Sponsoring Committee: Professor George Fisher, Chairperson
Professor Robert Rowe
Professor Ann Axtmann

MAPPING MOVEMENT TO MUSICAL RHYTHM:

A STUDY IN INTERACTIVE DANCE

Carlos Guedes

Program in Composition
Department of Music and Performing Arts Professions

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in
The Steinhardt School of Education
New York University
2005

Aos meus Pais, por todo aquele amor impossível de retribuir.

À Pascale, por motivos que levariam muito tempo a explicar.

À memória de Carlos Alberto Barros Guedes

À memória de Rui Guedes

À memória de Otília Cunha

ACKNOWLEDGEMENTS

I was fortunate to have the best committee one could find at NYU for doing a study like this one. I cannot find enough words, even in my native language, to thank George Fisher for all his help and orientation in order to bring this work up to its full completion. George's enthusiasm about the project from the very beginning; his highly perceptive reading and criticism of this text; his guidance on the text's internal organization and structuring — including the permanent advice to maintain this interdisciplinary study accessible to common readers, and all the English corrections on the use of terms and grammar — makes me wonder if I would ever be able to bring this project to its present form without his help. I embrace George's continued availability for orientation, discussion, and criticism until the completion of this dissertation as an act of truthful and sincere friendship, since he has not been acquainted with NYU for the past two and a half years.

I want to acknowledge Robert Rowe for the very precise and insightful criticism he gave about the text and the research, and for the invitation I got to be one of the first privileged readers of the draft of *Machine Musicianship*, whose "cognitive/music-theoretical" approach to making computational models was a great source of inspiration and model for shaping this study. I still want to thank

Robert for a fabulous seminar in Music Psychology he gave back in 1996, which has proven to be the seed for this dissertation.

I want to acknowledge Ann Axtmann, from the Gallatin School at NYU and my "dance" committee member, for all of her help in making me better understand human movement, for all of the long discussions we had about dance, and for the advice to do a weekend-long workshop on Laban Movement Analysis that has proven fundamental to better understand movement analysis. I also thank Ann for all the reviews and criticism she gave to the third chapter of this dissertation, which would have been much impoverished without her precious help.

I would also like to thank Melanie Percy from the Division of Nursing, one of the outside readers of this dissertation, for her valuable comments during the oral defense and her interest for this research.

Still at NYU, at Steinhardt's department of Music and Performing Arts, I want to thank Dinu Ghezzo, the director of the Composition program and the other outside reader of this text, for convincing me to do my PhD at NYU and for his valuable comments and ideas about this research during the oral defense. I want thank John Gilbert, the director of Doctoral Studies, for having "shared responsibilities" in the topic I chose for my dissertation, in part due to his great enthusiasm and stimulation for interdisciplinary research at the department. I also thank Paul Horan, the best advisor at the department, and a good friend who helped me to keep my academic status at NYU straight, especially during my years in Europe.

I am most indebted to the dancers that accompanied the development of this research in several stages, namely to (in order of their participation in the project) Marion Traenkle, Susanne Ohmann, Loup Abramovici, and, very especially, to Maxime Iannareli, the choreographer and dancer of the piece created for this study. The beautiful movement sequences these dancers provided, and their suggestions and criticism were crucial for the outcome of this research.

This piece of interdisciplinary research involved contacts with a lot of people and institutions in The Netherlands, Italy, and Portugal, besides NYU. In The Netherlands, I am especially grateful to the Institute of Sonology in The Hague, where I passed two years doing part of the research for this study. I am most thankful to Peter Pabon for his keen advice on DSP techniques and guidance on the (for me complex) implementation of the digital filters in one of the objects that were coded. I also want to thank Paul Berg, Clarence Barlow, and Joel Ryan for their seminars at this Institute, and for some good conversations that have proven influential for the compositional approach taken in the piece that portrays some of the latest shifts in my compositional interests. I also want to thank the STEIM studios in Amsterdam, especially Tom DeMeyer and Frank Baldé, for a residence at these studios in 2000, which has proven influential for the choice of using video analysis software to musically analyze dance. In Italy, I am grateful to the Laboratorio di Informatica Musicale in Genoa, especially to Antonio Camurri, Gualtiero Volpe, and Barbara Mazzarino, for a nice week I spent there in 2003 learning the analysis capabilities of EyesWeb. In Portugal, I want to thank INESC-Porto, namely directors Pedro Guedes de Oliveira e Artur Pimenta Alves, for letting me do there the first spectral analyses of frame-differencing data using Matlab. I want to thank Gustavo Martins for helping me with Matlab, and to Luís

Côrte-Real for suggesting a couple of good sources in computer vision. Still at INESC-Porto I want to thank very specially to Alberto Maia and Filipe Pinto for the two artistic collaborations we had so far involving interactive technology, the first of which led to my first collaboration in interactive dance.

I also have the privilege of having an excellent group of friends who provided great advice for the study from many discussions we had about dance, music, and interactive technology. I am most thankful to Kirk Woolford for teaching me the secrets of computer video analysis, and sharing some code he had written so that I could improve my knowledge in this area. I also want to thank Kirk for patiently letting me use his Blue Haptic Studios in Amsterdam to create the piece for this dissertation. I want to thank Ali Taylan Cemgil for some good discussions on tempo tracking and for suggesting certain good ideas during the conception of the adaptive clock. I want to thank Raúl Medina for once reminding me that periodic signals could be analyzed with FFT. I want to thank Saguenail for the insane discussions we had about how detecting rhythm using frame differencing would not work, and for other really good discussions that inspired the conception of this project among other things. I also want to thank Miguel Ribeiro-Pereira for his advice to do my graduate studies in New York.

My doctoral studies at NYU were possible thanks to the kind grants from Fundação Luso-Americana para o Desenvolvimento, and Fundação para a Ciência e Tecnologia (grant BD 13934/97). Part of the research at the Institue of Sonology was sponsored by NUFFIC, The Netherlands Organization for International Cooperation in Higher Education. I also would like to thank the director of the School of Music and Performing Arts at the Polytechnic Institute of Porto (ESMAE-IPP, in Portugal), Francisco Beja, and the directors of the music and theater departments, Jed Barahal and Claire Binyon, for authorizing two paid

leaves of absence in the school I currently teach in order to successfully complete this dissertation.

Finally, all of this work would have not been possible without the endless love support of my close family: Mom, Dad, Avó Mim, JP, Bárilu, Smi, Zgur, and Mimi.

TABLE OF CONTENTS

continued

continued

continued

continued

LIST OF FIGURES

continued

continued

LIST OF EXAMPLES IN THE DVD-ROM

CONTENTS OF THE DVD-ROM

FOLDER	
EtudeForUnstableTime	ForUnstableTime1.mov ForUnstableTime2.mov ForUnstableTime3.mov ForUnstableTime4.mov
Examples	ReadMeFirst.txt Example1.mov Example2.mov Example3.mov Example4.mov Example5.mov Example6.mov
CodeWarriorProjects	ReadMeFirst.txt m.bandit m.clock m.weights m.peak m.sample build-mac
PatchForUnstableTime	ReadMeFirst.txt UnstableTimeOsX.pat Grain hz2ms m.lp random-value slownoise~ rgrain2~ hanning.aiff tri.aiff SAMPLE.aiff ReasonVirtualSynth

CHAPTER I

INTRODUCTION

Back in 1998, in an East Village bar, I was having a heated discussion with a friend about the role of music in contemporary dance. At that time, I already had a vague idea of the work I wanted to do in interactive dance. My friend, a professional dancer who had danced with several companies in the New York area, was trying to explain to me how dance had liberated itself and had become an autonomous art form; it did not need music in order to subsist. Dance had turned into exploring the human body, and into interacting with other art forms such as video art and theater, and the role of the music was consequently diminished. And that was just fine.

I struck back, by saying that the proof music was so important to dance was that there had been several choreographies set to the same music throughout the 20th century, that there were choreographers who still addressed music as a very serious and undissembling counterpart to dance (e.g. Mark Morris); and that 99% of the shows of contemporary dance I had seen still had music, or at least some sort of figurative noise that was being played in the background. Therefore, music or sound *still* played an important role in dance performance.

Moreover, I told her I believed that with the advent of interactive-computer technology the interaction between music and dance could be reframed

at a different level. I told her about the good work of composers of interactive dance such as Mark Coniglio, Wayne Siegel, and Todd Winkler already pointing in that direction, and I hoped my work could also be included in that newly-created tradition of composers who attempt to renew the interaction between music and dance through the use of interactive-computer technology. She asked me "How?" I replied, "for example, by *punishing* dancers with the creation of an interactive system that generates *different* music for the *same* choreography." My friend laughed. I began this study.

Overview of the Study

In this study, I formulate the hypothesis that *dance movement possesses musical qualities at the temporal level. These qualities can be extracted in real time and provide a way for interaction in dance and music utilizing interactive computer systems*. By *musical qualities at the temporal level,* I mean that certain rhythms in dance bear qualities akin to musical rhythms in their durational and accentual character. Therefore, under certain circumstances these rhythms could be said to possess musical qualities.

This hypothesis will be tested through the creation of a software library that extrapolates what I call *musical cues* from dance movement. *Musical cues* are rhythms in dance that bear qualities akin to musical rhythms in their durational and accentual character. I also propose a framework for computer-mediated interaction between musicians and dancers within which this software is utilized. This proposed framework for interaction aims to open a *channel for musical*

communication between the musician and the dancer during the performance of a piece.

Essentially, the software library allows a dancer to control the tempo of an electronically-generated music score, and/or to generate musical rhythmic structures from bodily movement in real time in interactive dance performance. The movement data is gathered non-invasively, using a fixed video camera placed outside the area in which the dancer is moving. This library is implemented as a set of external objects, called the m-objects, for the modular programming environment Max/MSP (Cycling '74, 2001) using an external object or library that performs frame-differencing analysis of a video stream in real time. During the performance of a piece utilizing this software, the musician can give or take away the control over rhythm generation and/or musical tempo from the dancer, and substantially alter the musical content of a piece during performance. This will eventually stimulate the dancer to move away from a possible pre-established structure and improvise with the new musical content. This situation creates a feedback network between the musician and the dancer, in which the computer acts as a mediator, since it is providing the musician with *musical* data gathered from the dancer, but also providing the dancer with different musical stimuli arising from the musician's manipulation of the data.

Etude for Unstable Time is a piece for interactive dance created by me in collaboration with choreographer/dancer Maxime Iannarelli in which the formulated hypothesis is tested, through the utilization of the software and the

proposed framework for interaction. The compositional and collaborative approach taken for this piece is also discussed in the study.

Motivation

This study stems from my personal necessity for (a) carrying an interactive creative process throughout the actual performance of my music for dance, (b) improving the temporal articulation between my music and the movement performed by the dancers, and (c) creating a software library that can work as an addition to already-existing modular programming environments used in interactive dance.

My process of creating music for a dance piece is interactive in essence. I usually develop the music in close collaboration with the choreographers I work with. This is a step-by-step process in which the music and choreography gradually grow, influencing each other during the process of creation. This interactive process of creation between the music and choreography ends once the show moves from the rehearsal space to the theater where the performance will take place. The music, which is electronically produced, gets *frozen* on a CD in the last rehearsals before the performance.

However, other changes occur during the on-stage performance of the piece. The lighting, the costumes, and the set still transform the piece further, and the presence of the audience, night after night, makes no two performances the same — with the exception of the music. This frustrates the interactive process of collaboration I was referring to above. Even though I can *fade in* or *out* the music,

or press *play* or *stop* sooner or later in portions of the piece, I cannot *slow down* or *speed up* the music, or even *change its content* during performance. These latter aspects would improve dramatically the expressive significance of a choreographic work during performance, and allow for the interactive process of musical collaboration to carry on through the performance of the piece.

The improvement of the temporal articulation between my music and dance relates to the issue discussed above**.** I am particularly sensitive to the audiovisual impact created by the temporal interaction between dance and music during performance, in both its articulation in time and accentual character. The pace at which movement is articulated with the music, and the accents produced by the synchronization between music and movement, have an expressive value I greatly esteem when creating music for dance. However, during performance, there are often slight differences in the timing of dance movement. These differences in timing create pervasive unwanted *dissonances* in the overall temporal interaction between the dance and the music. This inadvertently affects the effect of the piece. Creating a software library that can adapt the tempo of the music, or synchronize the rhythm to the dancer's movement would help to circumvent this problem.

Finally, creating a software library that works in a widely used modular programming environment such as Max/MSP would allow the users of this library to utilize it as an *added facility* to these modular programming environments. For example, this library could be used with other libraries to synchronize digital video or imagery to electronic music and to a dancer's pace of movement. Even

though this software will be coded as a set of external objects for Max/MSP, it should be easily ported to other modular programming environments such as EyesWeb (DIST, 1999) or Isadora (Coniglio, 2002a). The code for the library will be provided as an appendix to this study.

A Musical Perspective

The literature reviewed for this dissertation includes topics from the areas of perception and production of musical rhythm, dance and movement theory, interactive computer music and interactive dance, digital signal processing (DSP), and computer science. The literature review and the study follow what I call *a path from music to dance and back to music*. I start by reviewing topics on the perception and production of musical rhythm in order to establish the important relationship between the perception and production of musical rhythm and the physical characteristics of the human body. I then analyze the concept of rhythm in dance and discuss how certain rhythms in dance can relate to musical rhythms. Further, I review the software commonly used in interactive dance performance, and the work by some of the most prominent composers and researchers in this field. Finally, I develop a computational process that analyzes rhythm in dance as if it was musical rhythm utilizing DSP techniques. This computational approach is inspired by some of the later computational approaches in musical tempo tracking in real time.

Although this study touches several research fields such as those mentioned above, the path I am taking throughout underlies the musical

perspective from which all these research areas will be observed. This is a study in music, more precisely, in musical composition. The theoretical discussion of this dissertation is oriented towards the creation of software and to the proposal of a framework for interaction that can enhance the compositional and (eventually) choreographical practice in interactive dance when utilizing already-existing modular programming environments widely used in the field**.**

A Reduction to Rhythm

In this study I follow a very thin and delicate line of theoretical thought that I call a *reduction to rhythm*. Music and dance are reduced to rhythmic art forms that produce a perceivable order in time through the succession of durations and accents. The term *perceivable* is used here in the strict sense, which means the ability of the brain to discriminate events in the time span that comprises the short-term memory. The idea of *perceived* (as opposed to *conceived*) rhythm as formulated by Paul Fraisse (1982) is thoroughly discussed in Chapter 2.

Consequently, I do not address, for example, harmonic rhythm or hypermeter when discussing rhythm in music. Even meter is given a much smaller role in the study than pulse for motives that are explained in Chapter 2. When discussing rhythm in dance, I acknowledge several types of rhythm in dance; yet, I pay much more attention to the aspect of the perceived articulations of dance in time than to articulations of dance in space, or to the spatial element altogether. Ignoring such fundamental elements as pitch and harmony in music, space in dance, and analyzing rhythm in its most abstract form in both disciplines,

was considered a necessary step to better bring to the surface the rhythmic similarities of music and dance.

This *theoretical reduction* proved helpful in developing the concept of *musically processing a dance* as explained in Chapter 5. The concept of musically processing a dance is implemented computationally and consists of analyzing dance movement from a musical perspective in order to facilitate the musical communication between musicians and dancers in computer-mediated collaborations.

Organization of the Text

This dissertation comprises two large parts of four chapters each. The first four chapters include this chapter and Chapters 2, 3, and 4 and correspond to the review of literature section of this study. Chapters 5, 6, 7, and 8, are the chapters in which I discuss the conception of the software library and its implementation, the piece that was created to test the software, and the conclusion in which I provide the guidelines for further study and discuss the original contribution of this study. A summary of each chapter's contents follows.

Chapter 2 discusses two important topics related to the perception and production of musical rhythm. These topics address the relationship between bodily movement and musical rhythm and pulse sensation and perceived meter respectively.

Chapter 3 (a) analyzes the concept of rhythm in dance as viewed by anthropology, philosophy, Laban Movement Analysis, and dance theory; (b)

investigates the temporal interaction between dance and music; and (c) provides a framework for the analysis of movement as musical rhythm.

Chapter 4 is the longest chapter of this study. This chapter, (a) reviews two types of interfaces commonly used in interactive dance — sensors connected to a MIDI digitizer and video cameras utilizing video analysis software; (b) reviews the most commonly used video analysis software utilized in interactive dance; and (c) discussess the approaches taken by prominent composers and creators of interactive dance in mapping movement to music.

Chapter 5 discusses the conceptual and technical considerations that assisted the creation of the software.

Chapter 6 presents a brief description of the software library and a detailed technical explanation of the core objects of the library.

Chapter 7 does a tutorial-like presentation of the possible applications of the software and discusses the piece *Etude for Unstable Time* in detail. This piece is a collaborative effort that served as the test bed for the software library.

Finally, Chapter 8 discusses the conclusions of the present study, its original contribution to the field of interactive dance performance, and provides the guidelines for further study.

CHAPTER II

TWO IMPORTANT TOPICS ON THE PERCEPTION AND PRODUCTION OF MUSICAL RHYTHM

Introduction

Research on the perception and production of rhythm is vast. This is witnessed by the amount of studies produced in this research area throughout the 20th century in the fields of experimental psychology, music theory, and music cognition.[1] These studies range from studies in tapping and synchronization with isochronous sequences of simple tones, to elaborate studies on expressive timing in music. The high degree of complexity and quality this research has attained in recent years, especially since the 1980s, has provided a fertile ground for creating computational models that emulate aspects of human rhythmic perception. These models have been used to either test or refine existing theories, or create interactive computer systems that are able to follow a live performance.

This chapter presents a review on two topics I found to be most relevant for the present study: (a) the relationship between rhythm, the human body, and movement; and (b) pulse sensation and perceived meter. The selection of these two topics as a point of departure for discussion, aims at providing a theoretical ground that:

---

[1] Michon (1985), Fraisse (1982), Parncutt (1994a) and Clarke (1999) provide excellent overviews of the research on several topics pertaining to the perception of rhythm. For a comprehensive article on rhythm that reviews many of rhythm's related aspects see London (2001).

1- Presents strong evidence about the relationships between the physical characteristics of the human body and the perception and production of musical rhythm. This will help to better discern the relationship between rhythm in dance and musical rhythm;
2- Provides excellent knowledge about how humans process musical rhythm. The fact that this knowledge is to a great extent obtained through experimental observation will optimize the creation of software that implements human musical processing;
3- Reiterates the musical perspective of this study.

Taking a theoretical path through a musical perspective will provide a better understanding of two important aspects that will emerge throughout the study: (a) the theoretical assumption made in the next chapter that movement in dance may possess musical qualities; and (b) the idea of musically processing dance, a metaphorical idea in the software conception presented in Chapter 5.

This chapter is divided into two parts entitled Movement, Rhythm, and Action, and Pulse Sensation and Perceived Meter respectively. In the first part, I will provide an overview of important concepts in the fields of experimental psychology, and music perception and cognition that relate musical rhythm, the human body, and movement. The second part will address relevant aspects pertaining to pulse sensation and perceived meter in music.

Movement, Rhythm, and Action

It is impossible to dissociate the role of the body and movement in the perception and production of musical rhythm. We use bodily motions to produce musical rhythm (e.g. when playing an instrument) and we often respond (consciously or unconsciously) to musical rhythm with movement through simple body motions such as rocking or foot tapping, or even by engaging in the act of dancing to the music.

The association between musical rhythm and human movement dates as far back as the Greeks. According to Paul Fraisse (1982), "rhythm" comes from the Greek *rhythmos* (rhythm) and *rheo* (to flow). *Rhythmos* appears as one of the keywords in Ionian philosophy generally meaning "form," but an improvised, momentary and modifiable form; it literally signifies a "particular way of flowing" (p. 150). Plato applied this term to bodily movements, which, like musical sounds, can be described in terms of numbers. Plato defined rhythm as being *the order in movement* in *The Laws* (Fraisse, 1982), and, departing from Plato's own definition, Fraisse defines rhythm as "the perception of an order" (p. 151). Implied in his definition is the fact that we can predict or anticipate what will come next in a rhythmic sequence. The characteristic of predictability distinguishes rhythm from arrhythmia.

Fraisse (1982) contends that the order in rhythm may be perceived or conceived. Conceived rhythm, such as the rhythm of very rapid and very slow

temporal successions,[2] is the rhythm in which the order is not assessed directly. All the rhythms we perceive "are rhythms which originally resulted from human [motor] activity" (p. 150). Fraisse also notes that "the possibility of rhythmic perception depends on tempo because the organization of succession into perceptible patterns is largely determined by the [Gestalt] law of proximity" (p. 151).

There are two notions in Fraisse's concepts about rhythm perception that are of great importance for the topics discussed in this chapter. The first is that rhythm can be perceived or conceived. This will help identify an important qualitative difference between pulse sensation and perceived meter in the next part of this chapter. The second is that the rhythms we perceive result from human motor activity. This is an important aspect to consider when addressing the relationship between rhythm and movement.

In a recent article that reviews the research in the temporal dimension of music, Eric Clarke (1999) considers two conceptual distinctions in analyzing the relationships among rhythm, timing, and movement:

> The relationship between rhythm and movement can be conceptually separated into rhythm and timing seen as the consequence of movement, and rhythm and timing seen as the source of, or motivation for, movement. The first of these two relationships is primarily an issue of motor control: timing information can either be seen as the input to a motor system, which then produces some kind of temporally structured behavior, or timing can be seen as the consequences of the intrinsic characteristics of the motor system and the body itself. . . .

---

[2] Fraisse (1982) gives as examples these rhythms the of day and night, of the seasons, or of the frequencies of light.

> Turning now to rhythm as the source of, or motivation for, movement, the most striking and obvious evidence for this relationship comes from the ancient association between music and dance . . . (p. 495).

The two conceptual distinctions in analyzing the relationship between rhythm and movement asserted by Clarke (1999) are of central importance for the present study. Analyzing how musical rhythm and timing are a consequence of movement will help support the idea that movement in dance may possess musical qualities. Analyzing how rhythm and timing are the source of, or motivation for, movement, will provide a better understanding of why dance and bodily movement are a natural response to music.

Movement as a Consequence of Rhythm

Musical rhythm bears a strong relationship to the physical characteristics of the human body. The musical tempi we are able to perceive and produce are related to certain periodicities of the human body such as those of heartbeats and footsteps (Parncutt, 1987). This confers on musical rhythm a universal character, as it relates its perception and production to the physical characteristics of the human species.

The periodicities found in musical rhythms lie between 200 and 1800 ms (Fraisse, 1956, 1982; Parncutt, 1987). This range is also the one that allows for motor synchronization with a given sound stimulus. According to Fraisse (1982), the optimal time intervals for motor synchronization lie between 500 and 800 ms. He also notes the striking similarity of these time intervals to certain human

bodily functions and activities such as the rhythms of the heart, walking, and spontaneous and preferred tempo.[3]

The periodicity or regularity inherent in the concept of rhythm bears, according to Fraisse (1974), strong affective connotations. This fact led him to prefer using the term *experience* to characterize what he calls the "plurisensual" perception of rhythm.[4] He states that although this experience is difficult to analyze, one can discern its main components. These components have two dimensions. The first is a personal dimension with two parts: (a) the "satisfaction" aroused by the return of what is anticipated in a rhythmic or periodic structure; (b) the ability of rhythm to induce motion,[5] and the ability of accompanying in pendular fashion — such as rocking — a perceived duration: "This motion is already a source of satisfaction by giving an easily maintained excitement that is increased by the harmony obtained between the perceptual and the motor" (p.114).[6] The intensity of this excitement increases if one yields to motor induction and performs more and broader movements like in dance. The second dimension of this rhythmic experience lies in its social character: the ability of

---

[3] Spontaneous tempo, also called personal tempo, or mental tempo, is the pulse which humans tap an ischronous sequence spontaneously. Preferred tempo corresponds to the speed of a succession of sounds or lights that is judged as being neither too fast or too slow (Fraise, 1982).

[4] According to Fraisse (1974), the term experience of rhythm was coined by C. A. Ruckmick (1927).

[5] Fraisse (1974) states that there is a natural tendency to synchronize body motion (such as hand-claps) with rhythmic patterns. The impulse for spontaneous body synchronization with rhythm does not disappear with age. According to Fraisse, Western cultures voluntarily inhibit this impulse.

[6] My translation. In the original: "Ce mouvement est déjà une source de satisfaction en procurant une excitation facilement entretenue qui est augmentée par l'harmonie réalisée entre le perceptif et le moteur."

periodic temporal patterns to synchronize humans in leisurely and working activities, contributes to enhancing rhythm's affective character.[7]

In regards to Fraisse's social dimension of the affective character of rhythm, Clarke (1999) notes that contemporary popular culture keeps the "ancient association between music and dance as vital as at any time" (p. 495). He also acknowledges the long history of working songs as a means of synchronizing people working together in order to optimize the efficiency and economy with which the energy is expended. Further, Clarke concludes:

> With this deep-seated association in the history of music and human action, it is hardly surprising that music with a periodic rhythmic structure tends to elicit accompanying movements, whether these are explicitly dance movements or less formalized responses and whether the music is intended to be dance music or not. (p. 495)

Richard Parncutt (1987) approaches the affective character of rhythm through a different perspective. He theorizes that the perception of pulse is acquired through prenatal conditioning, by the fetus's familiarization with sounds associated with the heartbeat and footsteps, as these sounds are more audible than outside world sounds. After acknowledging that musical pulses tend to vary between two extremes — strict pulses and rubato pulses — he suggests that strict rhythms (rhythms that evoke physical movement and dance) are associated with the sounds of footsteps, which are regular in general, and rubato rhythms (more suitable to emotion expression) are associated with the sounds of the heartbeat, since the speed of the heartbeat tends to vary with emotional state and with

[7] This "bodily" experience of rhythm suggests links to other concepts such as Merleau-Ponty's "lived-body" as explained by George Fisher and Judith Lochhead (2002).

breathing. According to Parncutt, rubato rhythms suggest physical movement less strongly and are more appropriate of romantic musical styles.

The short review presented above helps clarifying how physical movement can be understood as a consequence of musical rhythm. The fact the periodicities found in musical rhythm lie within a small range of values (200-1800 ms), which is also the range in which the motor synchronization with a sound stimulus is possible, bestows a close relationship between musical rhythm and bodily activity. Fraisse (1982) also notes the striking similarity between the optimal time intervals for motor synchronization (500 - 800 ms) and certain human bodily functions and activities, such as the rhythms of the heart and walking, and spontaneous and preferred tempo. Not surprisingly, these time values (75 – 120 beats per minute) comprise the musical tempi of most dance music. The greatest salience of pulse sensation also lies within this time interval — around 100 events per minute, or 600 ms (Fraisse, 1982; Parncutt, 1987, 1994a, 1994b).

The sensation of movement as an affective character embedded in the perception of periodicity and regularity in musical rhythm was also addressed through the studies of Fraisse (1974) and Parncutt (1987), in its lowest level.[8] A

[8] I call this a low level since it only addresses the aspect that musical rhythm can evoke physical movement. The research reviewed above does not address in detail the qualities of movement that can emerge from the experience of listening to music. This complex issue has motivated research in several fields, including music theory and experimental psychology. Some of these studies address the sensation of movement in the music listening experience (e.g. Dura, 1998); attempt to elaborate theories that consider music, motion, and emotion as being isomorphic (e.g. Gabrielsson, 1995); analyze the interaction of the auditory system with the motor system (e.g. Todd, 1993), or analyze how the central nervous system translates sound pulses into motor pulses (e.g. Clynes & Walker, 1982).

deeper analysis on the affective qualities of movement as evoked by music as well as qualities of movement performed as response, would certainly give a better insight towards the understanding of this intricate relationship. For sure, this would be of value in the creation of expert interactive dance systems. However, at the present time, this subject is outside the scope of this study. For now, it will suffice to acknowledge the notion that (a) musical rhythm bears a strong relationship to the physical characteristics of the human body, (b) the time values of periodicities found in musical rhythm relate to bodily functions and activities, and (c) the sensation of movement as perceived in music may lie in this latter relationship.

Rhythm as a Consequence of Movement

The production of accurate metronomic musical rhythm is an impossible task for humans as noted by Carl Seashore (1938). Eric Clarke (1999) asserts:

> If there is a principle on which virtually all rhythm research agrees, it is that small integer ratios of duration are easier to process than more complex ratios. And yet this principle is at first sight deeply paradoxical when the temporal properties of real musical performances are examined, for almost nowhere are small integer ratios of durations found — even in performed sequences that are manifestly and demonstrably simple for listeners to interpret. (p. 489)

When asked to produce perfectly isochronic sequences, even the most skilled performers introduce small random timing errors due to the neuromuscular system; that has been referred to as “performance noise” (Eck, Glasser & Port, 2000, p. 159). As pointed out by Douglas Eck, Michael Glasser, and Robert Port (2000), the auditory system is robust to this type of temporal noise, and is able to

filter out the small variances of a supposedly metronomic performance. In regards to this aspect related to the perception of timing, Clarke (1987) notes that rhythm perception is categorical, and Clarke (1987) and Parncutt (1994a) suggest that categorical perception is confined to simple categorical distinctions of integer ratios — between 1:1 or 2:1 or, more generally, between even (1:1) and uneven (N:1) (Clarke, 1999).[9]

Walking, dancing, or playing and instrument, are some of the physical actions that often produce perceivable periodicities. Understanding how the temporal noise in periodic actions can be modeled is of relevance for this dissertation. Analyzing "noisy" periodicities in dance movement through the eye of a video camera and transforming them into musical rhythms is the gist of this study. This section reviews some of the work by Henry Shaffer (1982, 1985) in skilled action in order to get a better understanding of how bodily action can produce perceivable periodicities.

Skilled Action

Henry Shaffer (1985) defines skilled actions as processes that structure the time domain. They (a) take definite forms in space and time, and (b) can be reproduced with great detail. A skilled action can be represented as a sequence of ballistic movements towards a definite goal, and these can be located in time

[9] Another aspect related to the perception of timing in music is expressive timing. Performers often lengthen or shorten the notated rhythm as a means to convey musical structure. These so-called expressive timing deviations are well felt by listeners (cf. Clarke, 1989). Its absence from a musical performance is often noted: performances lacking these expressive timing deviations are in general referred to as being wooden or artificial (Eck et al., 2000).

relative to their trajectory. Shaffer (1982) contends that the motor system acts as a timekeeper for these skilled movements, and the timing of skilled movement is based on an internal schedule of the targets of that movement.

Shaffer (1982) also notes that many skilled actions have the property of being nearly periodic. This "near-miss" periodicity happens at some level of the performance unit, and some skilled actions tend more towards periodicity than others do. He suggests that "[t]he strongest motive for periodicity is that it creates an iterative processing cycle that can facilitate the control of motor output. It can also help to regulate the accelerations in movement and so reduce the energy dissipated in performance" (p. 115). In this study, he examines tapping, handwriting, typing, playing music, and speech as instances of skills having this near-miss periodicity.

Skilled performance is often considered to be rhythmical. Shaffer (1982) notes that this is due to the inherent ambiguity about the meaning of rhythm:

> If we ask what is common between the rhythms of speech, playing tennis, and playing music, we find that it refers to different properties of time series, relevant in varying degrees to the different skills. Among these are properties of temporal pattern, periodicity, stress, expression, and quality of movement. (p. 109)

According to Shaffer, the only common aspect of rhythm in spatiotemporal action that applies to all skilled activities is the quality of movement, in which the skilled performer gives the appearance of being unhurried, fluent, and avoiding abrupt accelerations. In general, the production of rhythm is made possible because there is a preparation and coordination of these movements.

The properties of fluency and flexibility led Shaffer to formulate a theory of motor programming[10] whose major components are a performance grammar and a control system (Shaffer, 1982). The performance grammar provides the set of movements to be executed for a certain action, and the control system (a timekeeper, or clock) provides the schedule for the execution of these movements. At specific times, the timekeepers generate a motor command, which, after a certain motor delay, leads to a motor response (Beek, Peper, & Daffertshofer, 2000).

Shaffer (1982) proposes a model for periodic motor actions that assumes there is an internal timekeeper responsible for the control of periodic movement actions. This internal timekeeper is a stochastic clock that produces small random variations around its mean. Shaffer also accounts for another "noisy" source, which are the small random variations produced by the motor system in performing periodic actions (the motor variance). The motor variance and the internal timekeeper are thus unrelated stochastic variables. The equation he proposes for modeling periodic tapping is $I_n = C_n + I'_n - I'_{n-1}$ (p. 114). I_n is the interval between taps, C_n is the timekeeper and I' the movement referred to a beat. Shaffer's model stems from the one presented by Wing and Kristofferson (1973) that attempted to model isochronic tapping without the presence of a metronome. The model presented by Shaffer also considers synchronic tapping i.e., tapping in the presence of a metronome — in which case C would be a constant. This

[10] Motor programming theories belong to the class of information processing theories and cognitive constructivism. In general, motor programming theories are based on the distinction between a controlling system and a controlled system (Beek et al., 2000).

extension of Wing and Kristofferson's model accounts for the fact that musicians are able to move in and out of a reference beat during performance in a controlled fashion, and pianists can even go one step further, by letting one hand playing in and out of time while the other keeps the beat.

The interesting aspect of Shaffer's model is the assumption of the existence of an internal clock for producing periodic motor actions. This idea bears a strong resemblance with the ideas of Dirk-Jan Povel and Peter Essens (1985) that assume the existence of an "internal adaptive clock" in the perception temporal patterns (cf. Perception of Temporal Patterns below). The idea of an "internal clock" that adapts to timing variations is also used in the software that was coded for this study (cf. Chapter 6).

The important ideas expressed by Shaffer (1982, 1985) about skilled actions could be applied to dance. Movement in dance structures the time domain, it can be reproduced in great detail, and periodicity in dance can be understood as a means of producing and maintaining fluency in motion. In addition, rhythm is produced as consequence of movement and is considered as one of the defining characteristics of dance (cf. Beardsley, 1982). The interrelation between dance and music suggests that the processes of production of rhythm in dance may be closely related to those that produce rhythm in music. This aspect will be addressed in more detail in the next chapter.

Pulse Sensation and Perceived Meter

If we except Gregorian and some post-serial music, Western music has always been built around the idea of a beat or pulse. Meter, which establishes a hierarchy in a group of pulses, provides a temporal grid for the unfolding of musical rhythm in time. Amid the myriad of styles and periods Western "metered" music has known, a metrical hierarchy and a beat are more or less perceived by a listener. This tends to vary with style and period. For example, early Baroque dance music and the German songs from the Romantic period can provide two extremes of a spectrum in which on one side we have a clearly perceivable pulse in music, and, on the other, a musical style in which strong tempo fluctuations are idiomatic thus implying that a beat is not clearly felt.

In this section, I discuss some studies by Parncutt (1987, 1994a, 1994b) on pulse sensation,[11] and Fred Lerdahl and Ray Jackendoff (1983) on perceived meter. Comparatively, more emphasis and space are given to pulse sensation than to perceived meter. This is because much of this study is concerned with pulse detection in dance and completely ignores the detection of meter in dance. Musical pulse and phrase encounter strong parallels in dance whereas meter does not — at least in the dance literature reviewed for this dissertation. The inference of a metrical structure seems to be a musical-cognitive process; it deals with accents at the musical surface that encompass both harmony and pitch besides the accents caused by the duration and articulation of events. However, the literature

[11] Parncutt (1994a) uses "pulse sensation" as a blanket term for "beat," "swing," "rhythmic level," and so on. In this study, I use this term for the same purposes.

on rhythm perception/cognition seems at times to address pulse and meter as equivalent phenomena (e.g. Clarke, 1999, pp. 482-489). In a certain sense they are, since perceived meter has strong similarities with pulse sensation. The similarities, as well as the qualitative differences between pulse and meter, will hopefully be unfolded throughout this section and will be summarized in the subsection Similarities and Differences between Pulse Sensation and Perceived Meter.

This analysis will thus lay out the main aspects to consider in the creation of the software. Before discussing the selected literature on pulse sensation and perceived meter, I will describe the concept of serial and periodic grouping in music and review studies on the perception of temporal patterns in order to better support this discussion.

Serial and Periodic Grouping

Parncutt (1994a) notes that events in music can be grouped "serially" or "periodically" or both. Parncutt sees these terms as equivalent to Lerdahl and Jackendoff's (1983) grouping and meter and assumes that they are also related to the way sound events are grouped in speech as noted by Martin (1972). Serial, or figural, grouping depends on the proximity of adjacent events in time. Periodic grouping is hierarchical in nature and usually depends on the relative timing and perceptual properties of non-adjacent events (Martin, 1972). Parncutt (1994a) distinguishes two stages of periodic grouping in music: pulse sensation and perceived meter**.** Based on evidence from the literature, he conveys that serial and

periodic grouping are independent as they depend in distinct ways on the interonset intervals[12] (IOIs).

Perception of Temporal Patterns

In an attempt to describe how subjects process temporal patterns, Povel and Essens (1985) propose a model based on the assumption that listeners make use of an internal clock for the processing of the temporal structure of such patterns. This clock is hierarchically organized and adaptive. Its hierarchical organization resembles everyday life clocks, which are organized into hours, minutes, and seconds; yet, this clock's time unit and subdivisions are not fixed but flexible, since it is able to synchronize with a temporal pattern. The authors note that foot tapping can be itself interpretable as reflecting the listener's internal clock. Povel and Essens (1985) assert that this clock is configured based on the distribution of accents perceived in the pattern, and listeners tend to prefer clocks in which the coding is done economically.[13] They also assume that not every pattern may induce a clock. In this case, the pattern may be coded "figurally," by clusters of events. When this type of coding occurs, detailed information about the relative duration of the intervals seems to be left uncoded.

In a different study, Povel (1985) analyzes the rhythmicity of rhythmical patterns. Rhythmicity, according to Povel, is the feeling of tension in a rhythmic pattern as perceived by listeners that may induce movement or dance. In this

---

[12] Interonset intervals are the intervals between the onsets of note events.

[13] This assertion is supported by experimental observation reported in the study.

context, one rhythm may be said to be more rhythmical than another. For Povel, a rhythmical pattern is a temporal pattern whose structure can only be accessed by the means of an internal clock. He makes three assumptions for the determinants of rhythmicity:

1. Rhythmicity arises when the monitoring mechanism, on basis (sic) of one or more events in the sequence, registers a deviation from the prior pattern of such degree that the adjustment of the current clock must be made. This will create uncertainty and ambiguity which is experienced as perceptual tension or rhythmicity.

2. Rhythmicity will be higher when such adjustment must be made more often.

3. Rhythmicity will be higher the stronger the current clock is at the moment that contradictive evidence occurs. (p. 216)

Implied in these determinants is an idea of ambiguity without uncertainty; i.e. the rhythmicity of a sequence is high when the event accents contradict the induced clock accents, however the feeling of the clock must prevail throughout the sequence. The experiment results shown in the study confirm these determinants.

The two studies discussed above provide examples of the research done in the processing of rhythmic patterns. According to the theories of Povel and Essens (1985), the existence of an induced clock, or pulse, seems to be crucial for the accurate processing of temporal patterns in time. Temporal patterns may also be coded figurally, by clusters of events; in such case the correct timing information about the pattern seems to be left uncoded. An interesting concept is that of rhythmicity as defined by Povel (1985), associated to the degree of tension emerging from a rhythmic pattern. It is important to note that this tension can only exist due the presence of a clock or pulse.

Pulse Sensation

Coding of Auditory Patterns

Pulse plays an important role in the perception and coding of musical rhythm. Christopher Longuet-Higgins and Christopher Lee (1982) state that "the concepts of 'beat,' 'meter' and 'bar' are of central importance in the perception of music" (p.115). Parncutt (1987) notes that pulse may be regarded as the basis for musical rhythm as it distinguishes rhythm from non-rhythm. This reinforces what has been discussed above about the perception of temporal patterns, namely the importance that an "induced clock" has in the correct processing of musical rhythm.

The sensation of pulse, or beat, is confined to the auditory system (Parncutt, 1994a). This gives an advantage, in terms of perceptual coding, to musical rhythm over visual patterns that are rhythmically organized in time, when the periodicities in both types of organization (acoustic and visual) are within the time values in which pulse is felt. Parncutt (1994a) illustrates this aspect clearly:

> Rhythmically organized auditory patterns are easier to encode, recall, and reproduce than similarly organized patterns (Garner & Gottwald, 1968; Glenberg, Mann, Alman, Forman, & Procise, 1989; Handel & Buffandi, 1968). A feasible explanation is that presentation times are encoded more accurately in the auditory case (Glenberg & Swanson, 1986). This may be due to the confinement of the sensation of beat or pulse to the auditory modality (as demonstrated, e.g., by Grant & LeCroy, 1986). Perhaps the auditory rhythms can be perceived by encoding their constituent events relative to an underlying musical beat or pulse, but visual "rhythms" cannot. This view would be consistent with observations of Glenberg and Jona (1991) and Schab and Crowder (1989) that the advantage of auditory over visual rhythms diminishes for tasks involving relatively long durations, because the perceptual salience of pulse sensations diminishes as their period increases from about 600 to 2000 ms (Fraisse, 1982). (pp. 410-411)

At first glance, this aspect noted by Parncutt may defeat the purpose of this study. Attempting to create an algorithm that detects *the musical tempo of a dance,* when a fair amount of research portrays pulse sensation as being a perceptual property of the auditory system, seems to be a useless endeavor. The assertion that "visual" rhythms do not evoke the sensation pulse is of paramount importance for the present study, namely in the conception of the theoretical grounds for the creation of software that attempts to determine *the musical tempo of a dance*. In Chapter 5, when I discuss the conceptual considerations in the creation of the software and introduce the idea of *musically processing a dance,* by looking at rhythms in dance that bear the quality of musical rhythms, I am not suggesting that dance movement can evoke the sensation of pulse as music does. Instead, I am considering that certain rhythms in dance, because of their articulation, accentual character, and duration, bear the qualities of musical rhythms as they also convey the sensation of pulse in dance movement (cf. Chapter 3, The Rhythm(s) of Dance).

Nevertheless, the above citation may help explaining certain important aspects about the interaction between music and dance, namely the importance that pulse in music exerts over the perception of duration in dance. Musical pulse may facilitate the perceptual coding of the visual patterns performed in dance, by functioning as a "clock" in the audiovisual setting that allows shorter and more precise temporal segmentations to be done.[14]

---

[14] Cf. Chapter 3, Two Perceptual Studies on the Relationships between Music and Dance. The experiments of Carol Krumhansl and Diana Schenk (1997) noted that the temporal segmentations

Pulse Sensation

Pulse sensation is a form of pattern recognition (cf. Povel & Essens, 1985; Parncutt 1994a). In the presence of a rhythmic sequence, a listener will try to match the IOIs to an isochronous template that will fit a predominant pulse sensation. Pulse sensation, as defined by Parncutt (1994a) "is a measure of the probability that a listener will tap out that pulse when asked to tap the tactus (main or underlying beat) of a rhythmic sequence" (p. 433). Musical pulse sensations are illusory in the sense that they do not indicate the presence of a single periodicity in a rhythmic pattern.

According to Parncutt (1994a), a pulse sensation arises from the complex interaction of all the periodicities involved in a given sequence. He compares this situation to perceiving a musical chord in which the constituent notes seem to blend into a single sound, or the sensation of pitch while listening to complex tones. He views pulse perception as an acoustic phenomenon and proposes an interesting definition of musical rhythm that distinguishes it from other perceivable rhythms: "A musical rhythm is an acoustic sequence evoking a sensation of pulse" (p. 453). Pulse sensation is also confined to the limits of short-term memory. Short-term memory sets an upper limit to the sensation of pulse. Pulse sensations cease to exist at values exceeding approximately 1800 ms or below 200 ms (Fraisse, 1982; Parncutt, 1987, 1994a).

---

of a choreography set originally to music were done in larger segments with fewer subdivisions when the participants were asked to do a temporal segmentation of the dance watched in silence. They also noted that the delineation of temporal units by section ends and new ideas was clearest when the participants just listened to music. Hodgins (1992) noted that the perceived duration of a choreography was influenced by the temporal and textural elements of music.

Ambiguity of Pulse Sensation

A rhythmic sequence can evoke several levels of pulse sensations. That is demonstrated by the fact that subjects can tap at different rates in the presence of the same sequence. For example, musicians tend in general to tap at slower rates than non-musicians (Drake, Penel, & Bigand, 2000). Experimental observation reported by Parncutt (1994a) confirms that the tactus level (main pulse sensation) was found to be quite ambiguous, and the ambiguity of the tactus increased as the rhythmic patterns became more complex. Parncutt concluded that the tempo of most pulse sensations lay on the vicinity of moderate tempo (ca. 100 events per minute, or 600 ms) regardless of the rhythm's note rate. Departures from moderate tempo cause pulse salience to decrease. This led him to assume the existence of a pulse region that can be determined by a bell-shaped function centered at the period of 600 ms that is symmetrical with respect to the logarithm of the pulse period (Parncutt 1987, 1994a, 1994b). Most of the area under the curve is contained between the values 400 and 900 ms: 67-150 events per minute (Parncutt, 1994a, 1994b).

Pulse sensation is enhanced by the existence of parent or child pulse sensations. David Rosenthal (qtd. in Parncutt, 1994a) defines parent or child pulse sensations of a certain pulse as being isochronous levels that are rhythmically consonant,[15] and stand hierarchically above or below that pulse — e.g., the 1/4-note pulse is a child of the 3/4-note pulse in a 3/4 meter.

---

[15] Consonant rhythmic strata, as defined by Maury Yeston (1976), are those in which the divisor for one level is a factor for the other (e.g. one level consisting of half-notes and another consisting

Summary of Main Concepts about Pulse

The above paragraphs helped clarifying important concepts about pulse sensation that are of great importance for the present study. I will now summarize the most relevant aspects that emerged from the reviewed literature:

1- Pulse sensation is an acoustic phenomenon that arises from the interaction of periodicities in a musical sequence;
2- Pulse sensation is enhanced by the existence of child or parent pulse sensations;
3- Pulse sensation is ambiguous; a rhythmic sequence can evoke several levels of pulse sensations;
4- Most pulse sensations lay in the vicinity of 100 events per minute or 600 ms (moderate tempo) regardless of note rate. Departures from moderate tempo cause pulse sensation to decrease;
5- The region of greatest pulse salience is between 400 and 900 ms, and pulse sensations cease to exist outside the 200-1800 ms time value range;
6- The existence of pulse in a musical sequence facilitates its perceptual coding.

---

of quarter-notes). A rhythmic stratum is an isochronous level in a rhythmic hierarchy akin to Schenkerian level.

Perceived Meter

Perceived meter consists of the hierarchical grouping of pulses into holistic temporal patterns (Parncutt, 1987). According to Lerdahl and Jackendoff (1983), one fundamental aspect to the idea of meter is the periodic alternation of strong and weak beats. Phenomenal accents function as an input to metrical accent[16] as they serve as cues towards establishing a metrical structure. If these cues are not very regular or are in conflict, the sense of metrical accent becomes attenuated or ambiguous. If they are regular and mutually supportive,[17] the sense of metrical accent becomes definite and multileveled. Once a clear sense of meter becomes established, the listener renounces it only in the face of strongly contradicting evidence. Syncopation occurs when cues are in contradiction with an established metrical interpretation but are not strong enough to override it. Lerdahl and Jackendoff contend that it is the listener's cognitive task to match a given pattern of phenomenal accents to a permissible pattern of metrical accentuation. "Metrical accent, then, is a mental construct, inferred but not identical to the patterns of accentuation at the musical surface" (p. 18).

One final remark about perceived meter is that metrical grouping occurs within the limits of the psychological present. This limit (4 – 5 seconds), is also found in other rhythmic arts as noted by Fraisse (1982); "The slowest adagio in a

---

16 Phenomenal accents are any events at the musical surface that give emphasis or stress to a moment in the musical flow. This category includes pitch-events, local stresses such as sforzandi, sudden changes in dynamics or timbre, long notes, leaps to relatively high or low notes, harmonic changes, and so forth. Metrical accents are beats that are relatively strong in their metrical context (Lerdahl & Jackendoff, 1983).

17 Or consonant, in Yeston's (1976) view (cf. fn. 15).

9/4 bar is no longer than 5 sec, and the longest lines of poetry have from 13 to 17 syllables, the time necessary to recite them being no longer than from 4 to 5 sec (Wallin, 1901)" (p. 158). This latter aspect about metrical grouping is important to consider in addressing the qualitative distinctions between pulse sensation and perceived meter in the section Similarities and Differences between Pulse Sensation and Perceived Meter.

Ambiguity of Perceived Meter

Perceived meter, as a cognitive task that imposes a metrical hierarchy on a group of pulses, carries a fair degree of ambiguity. Perhaps the best way to start understanding this ambiguity is to look at the phenomenon of subjective rhythmization.

Subjective rhythmization consists in the perceptual grouping of non-accented events in an isochronous sequence.[18] Wilhelm Wundt (1880) found that subjects in the presence of an isochronous sequence of simple tones tended to group the events in pairs or in fours (often grouped as 2+2), less commonly in threes[19] (Quoted in O'Brien, 1998).

---

[18] Studies on subjective rhythmization have a long history in experimental psychology, as subjective rhythmization was some of the first phenomena to be analyzed in the field of rhythm perception (O'Brien, 1998). See Fraisse (1982, pp. 155-56) for a summary on the most relevant studies on subjective rhythmization.

[19] According to Parncutt (1994a), this may explain why the enhancement of pulse sensation may be expected to be greater in pulses that contain child pulse sensations of 1:4 proportion than in pulse sensations that have a ternary subdivision: in a grouping by fours we have a family of three consonant pulse sensations (1, 2 and 4), whereas in ternary subdivision we have only two consonant pulse sensations (1 and 3).

Some of the studies reviewed in this chapter, also account on the subjective nature of metrical grouping. For example, Fraisse (1982) reports research carried out by Vos (1978), for which subjects familiar with Western classical music were asked to tap in synchrony with the beginning of each bar in a commercial version of Bach's preludes: "For a 3/4 bar, 20% tapped each beat. Eighty percent tapped as if it were bar with two beats, grouping two of the three beats, the last beat of a bar being grouped one time out of two with the first beat of the following bar |**1**2**3**|1**2**3|" (p. 173). Parncutt (1994a, pp. 419-423) reports similar findings about the ambiguity of metrical accent as perceived in an experiment designed to test the salience of metrical accent.

Similarities and Differences between Pulse Sensation and Perceived Meter

The second part of this chapter addressed important aspects about pulse sensation and perceived meter. The following summary will bring to the surface the similarities and differences between the two.

The first similarity is that pulse sensation and perceived meter are two forms of periodic grouping in music (Parncutt, 1994a), which implies that in both cases there is an established hierarchy among the perceived periodicities. Pulse seems to be the first manifestation of a metrical hierarchy in music because once the sense of pulse becomes established, some sort of metrical grouping will occur even if it is by subjective rhythmization.

However, pulse sensation and perceived meter are qualitatively different. Pulse sensation is an acoustic phenomenon (Parncutt, 1994a) that emerges from a

perceived *consonance* between low-frequency periodicities present in an acoustic sequence, or "an objective reality of the acoustic signal" (Iyer, 1998, p. 75), and perceived meter is a cognitive task (Lerdahl & Jackendoff, 1983), i.e. an intellectual process. Also, they occur at different time spans: pulse sensation is felt between 200 and 1800 ms, the region of greater salience being between 400 and 900 ms (Parncutt 1994a, 1994b), and perceived meter can extend up to 5 seconds (Fraisse, 1982). On the other hand, pulse sensation is related to the physical characteristics of the human body (e.g. Fraisse, 1982; Parncutt, 1987) and perceived meter to the way the brain processes information in the psychological present (see Parncut, 1994a, pp. 450-452).

Finally, both pulse sensation and perceived meter are found to be quite ambiguous, and yet they are crucial for the better processing of musical patterns. As noted in Perception of Temporal Patterns, the existence of a "clock" or pulse in a musical sequence facilitates the coding of temporal patterns.

Concluding Remarks

The analysis of the selected topics on the perception and production of musical rhythm helped bringing important insights into this study. The first part of this chapter put in evidence the important relationships between musical rhythm and the physical characteristics of the human body, the sensation of movement as perceived in musical rhythm, and how rhythm and timing can be a consequence of movement. In section Movement as a Consequence of Rhythm, I identified the relationship between the time values of the perceivable periodicities

in musical rhythm and motor synchronization, as well as the important relationship between the optimal time values for motor synchronization and functions and activities of the human body (heartbeat and walking), and preferred and spontaneous tempo. In this section, I also addressed the important aspect of motor induction by musical rhythm and the affective connotations that aspect entails in what Fraisse (1974) describes as the experience of rhythm. In the section Rhythm as a Consequence of Movement, I looked at the concept of skilled action as defined by Shaffer (1982, 1985). From this definition it can be inferred that dance is a skilled action. Shaffer's theory that skilled actions can be modeled as "clocked" events, and that some skilled actions have the property of being nearly periodic will help support the concept elaborated in the upcoming chapter that certain rhythms in dance could be interpreted as musical rhythms (cf. Chapter 3, Finding the Common Denominator).

In the second part of this chapter, important aspects about pulse sensation and perceived meter were analyzed. This provided important knowledge for the creation of the algorithms discussed on Chapters 5 and 6, namely in two important concepts that emerged from this part of the literature review. The first concept is that pulse sensation is a form of pattern matching, in which an isochronous and adaptive "clock" is used to match the pulse of a rhythmic sequence (Povel & Essens, 1985; Parncutt 1994a). The second concept is that pulse sensation is enhanced through the presence of consonant rhythmic strata (Parncutt 1994a).

CHAPTER III

RHYTHM IN DANCE, TEMPORAL INTERACTIONS WITH MUSIC, AND MOVEMENT AS MUSICAL RHYTHM

Introduction

Dancing involves the production of rhythm. When we see dance, we clearly perceive patterns of movement that are organized in time and space. Dance is often set to music, and it is common to see certain ways of dancing corresponding to specific types of rhythmic patterns in music. Popular and folk dances illustrate that aspect very well: rumba is danced differently from samba and from swing. These dances differ from each other in the same way the rhythm of their music does. These differences pertain essentially to the metrical qualities, internal accent within the meter, and tempo of the corresponding musical rhythm.

As we saw in the previous chapter, the perception and production of musical rhythm are intimately linked to certain physical characteristics of the human body. The periodicities in musical rhythm have the ability of inducing bodily motion (cf. Fraisse, 1974). Bodily movement is a natural response to musical rhythm. At times, the degree of synchronization attained between bodily movement and music in dance gives the impression that dance is performing spatial realizations of the music's own rhythm. These aspects suggest that there are similar rhythmic realizations in both art forms, and provide a strong motivation towards investigating the similarities between musical rhythm and

rhythm in dance. The main goal of this chapter is to investigate such similarities.

The chapter is divided in three main sections — The Rhythm(s) of Dance, Temporal Relations with Music, and Movement as Musical Rhythm. In the first section, I will analyze the concept of rhythm in dance as portrayed by several authors in the fields of anthropology, philosophy, Laban Movement Analysis (LMA),[20] and dance choreography. The concepts gathered from this review will lead me to group them into three possible categories for rhythm in dance.

The second section will focus on problems involving the analysis of the temporal interaction between music and dance. Special attention will be drawn to two perceptual studies that analyze the temporal interaction between music and dance from different perspectives (Krumhansl and Schenk, 1997; Hodgins, 1992). Carol Krumhansl and Diana Schenk (1997) analyze the extent to which dance can reflect the structural and expressive qualities of music. Paul Hodgins (1992) analyses the impact of certain musical qualities in the structural perception of dance.

Finally, the third section will provide a framework for the analysis of dance rhythm as musical rhythm. I will compare the knowledge gathered in the previous chapter about the role of the body in musical rhythm production, skilled action, and pulse sensation, with the ideas developed in the first two sections of this chapter. This will have the goal of defining which types of dance rhythms

---

[20] Rudolf Laban (1879 - 1958) was one of the founders of modern dance in Germany in the 20th century. He and his followers devised a system for movement analysis — nowadays known as Labananalysis, or Laban Movement Analysis, or LMA. They also developed a notation system for body movement (Labanotation) that is analogous to music notation (Bartenieff & Lewis, 1980).

bear the characteristics of musical rhythms. Eventually, this comparison will lead to the formulation of the first main theoretical assumption of this study — that *movement in dance possesses rhythmic qualities akin to musical rhythms*.

The Rhythm(s) of Dance

Rhythm is considered to be a defining characteristic in dance. Anthropologist Anya Royce (1984) considers rhythm to be a structural characteristic of dance. She contends that "all arts of all cultures have in common such features as virtuosity of form, symmetry, and rhythm" (p. x). Philosopher Monroe Beardsley (1982), in defining what constitutes the act of dancing, considers rhythm to be one of its three important characteristics: "when a motion, or sequence of motions, does not generate practical actions, and is intended to give pleasure through perception of rhythmic order, it is dance"(p. 35). Anthropologist Judith Hanna (1979) provides a definition for dance practice that resonates with Beardsley's views:

> Human behavior composed, from the dancer's perspective of (1) purposeful, (2) intentionally rhythmical, and (3) culturally patterned sequences of (4a) non-verbal body movements, (4b) other than ordinary motor activities, (4c) the motion having inherent and aesthetic value. (Quoted in Sparshott, 1995, p. 103).

The "intentionally rhythmical" characteristic of dance in Hanna's definition led philosopher Francis Sparshott (1995) to investigate what type of rhythms are involved in a dance. Sparshott notes that since everyone agrees the "rhythm" of a dance is one of the most important characteristics, "we would expect the possibility of radical differences among dances to go with a

corresponding indeterminacy in the concept of rhythm. And so it does" (p. 154). After revisiting some concepts and definitions of rhythm such as Aristoxenus's definitions, Sparshott concludes that the degree of rhythmicality in a dance depends on four aspects:

> First is salience, the obviousness of the temporal . . . . Second is closeness to the organic, body-centered rhythms we identify in our lives. Third is the degree to which the rhythm enhances and enforces such a rhythm. And a fourth is the extent to which it departs intriguingly or captivatingly from its basic beat or sets up a counterpoint to it. Conversely, the unrhythmical is what conspicuously does none of these, neglecting them or pointedly avoiding them. (p. 180)

He also notes that "it must be hard to move in a way that is truly devoid of rhythm; it might be easier to move in a way that makes a parade of its unrhythmicality" (p. 180). This assumption made by Sparshott seems easily acceptable if one considers what was discussed in the previous chapter about the role of bodily functions in the perception and production of rhythm.

Based on a Laban Movement Analysis (henceforth LMA) effort/shape[21] perspective, Irmgard Bartenieff[22] (Bartenieff & Lewis, 1980) mentions that Rudolf Laban considered elemental rhythms to be based on polarities of exertion and recuperation (awake/asleep, work/rest, etc.). These rhythms are expressed in spatial and effort[23] patterns and in the use of body parts. Changes within

---

[21] LMA analyzes movement from three different perspectives: body (what moves), effort and shape (how), and space (where).

[22] Irmgard Bartenieff was one of Laban's students who later came to the United States and further developed his ideas in dance, anthropology, and physical therapy. She was the founder of the Laban/Bartenieff Institute of Movement Studies in New York (Bartenieff & Lewis, 1980).

[23] Laban identified four motion factors (space, weight, time, and flow) toward which performers of movement can have different attitudes when moving. The attitudes toward the motion factors of space, weight, and time, are what Laban called *antrieb* in German, a combination of *an* (on) and *trieb* (drive). These attitudes represent the organism's urge to make itself known. "In English

movement produce rhythmic patterns of stresses and phrasing that can be experienced in the spatial scales. A phrase is that organization of movement processes that consists of a beginning, middle, and end of a statement. Bartenieff considers that rhythm in movement is not only the perceived accents and stresses caused by the changes within movement, but also the alternation between Effort qualities in interaction with the spatial element:

> [R]hythm is not just a duration of time, accentuated by stresses. It is also the result of the interaction of Effort combinations with variations in spatial patterns.
>
> Since time relationships and all divisions of time are actually perceived in space, the movements of the body in space, the movements of sound in space and the movements of lines and colors in space can be related to the time divisions to produce visible and audible forms of time in dance, music and art as well as in functional activities. (p. 75)

The interesting aspect in Bartenieff's definition, that contrasts with the previous definitions, is the inclusion of the spatial element in the definition of rhythm in movement. The interaction with space is a crucial aspect in the definition of rhythm in dance movement. All dances evolve in space and the interaction with the spatial element is thus a crucial aspect to consider for a complete

---

translation *antrieb* has become Effort" (Bartenieff & Lewis, 1980, p. 51). The four effort elements of space, weight, time, and flow (tension) are used to describe general qualities of the movement attitude. These elements are usually perceived in combinations and sequences that express dominant characteristics of the mover. According to Bartenieff (Bartenieff & Lewis, 1980) every movement has some degree of the above-mentioned effort elements. For a certain movement each effort element is described as belonging to one of two opposing qualities: space can be direct or indirect; weight can be light or strong; time can be sustained or sudden, and the flow of tension can be free or bound.

Laban defined "inner states" or "incomplete efforts" as being combinations of two effort elements. There are six effort states: alertness (space/time), dream (flow/weight), remote (space/flow), near (weight/time), stable (space/weight), and mobile (time/flow) (Bartenieff & Lewis, 1980). The near state, or rhythm state has been consistently observed in numerous dance styles especially in the folk or indigenous styles (A. Axtmann, personal communication, January 8, 2004).

understanding of rhythm in dance.[24]

In another approach, choreographer Doris Humphrey[25] (1959) considers four sources of rhythmical organization in humans:

> First, the breathing-singing-speaking which leads to phrasing and phrase rhythm. Then the partly unconscious rhythms of function: the heartbeat, peristalsis, contraction and relaxation of muscles, waves of sensation through the nerve ends. Another is the propelling mechanism, the legs, which [humans] discovered would support [them], one after the other, while moving in space, and which provided also a conscious joy in beat as the weight changed. Lastly, there is emotional rhythm: surges and ebbs of feeling, with accents which not only supply strong rhythmical patterns but are a measure for judging emotional rhythms in others. (p. 105)

While Humphrey (1959) acknowledges four sources of rhythmical organization, only three types are used in dance: motor rhythm, breath rhythm, and emotional rhythm.

The motor rhythm, which Humprhey (1959) also calls the "beat" rhythm, is produced by the motor mechanism, and is the one she considers to be the most important for dancers: "Here is where the original dance began — with the feet — and here is where it still carries on, in the main" (p. 105). She also thinks that

---

[24] However, this aspect is not considered in the present comparison. Musical rhythm is often treated theoretically and from the perceptual/cognitive perspective as being one-dimensional, i.e., as consisting of durations and accents articulated in the time domain. One can also identify a spatial element in musical rhythm, namely the spatial interaction that may exist in a piece for large orchestra or ensemble, or even in electronic music where the spatial element occupies a more prominent role with sound spatialization. The discussion of the (bi- or tri-dimensional) spatial element in musical rhythm is outside of the scope of this study, and therefore only the one-dimensional aspect of musical rhythm is addressed (i.e. the temporal distance between events and articulations) as is seems to be the crucial aspect in its understanding. Consequently, the comparison between rhythm in movement and musical rhythm only addresses their articulatory nature in the temporal domain, and the method that is described for capturing the articulations of movement in dance in Chapter 5 *strips out* the spatial element in dance movement and only represents the movement inflexions of dance in the time domain.

[25] Doris Humphrey (1895 - 1958) is one of the founders of American modern dance during the 1930-50s and one of the few professional choreographers who has written an actual text on how to choreograph.

the awareness of accent, the energy punctuated by the beat, stems only from the change of weight in dance established by the human feet when walking. The act of walking is the key pattern of fall and recovery, it is the giving into and rebound from gravity she considers so fundamental towards providing the sense of beat. According to Humphrey, this play with gravity, primevally manifested by walking, is the very core of all movement.

Breath rhythm gives the sense of phrasing and is qualitatively different from motor rhythm since it has not a sharp accent. According to Humphrey (1959), breath rhythm is not concerned with the physical life in the sense that motor rhythm is. The idea of breath rhythm — the inhalation, suspension and the exhalation — can be transferred to other parts of the body, such as the knees, arms, or even the whole body.

Finally, emotional rhythm is related to the succession of dramatic states a piece of dance may have. It may be cast into a breath rhythm, motor rhythm, or movement sequences. Its radical distinction from the previous type of rhythms is that — although there is some sense of regularity inherent in both motor and breath rhythms — the emotional rhythm is unstable "as the human being is not capable of sustaining a feeling in an absolutely steady intensity and rhythm" (Humphrey, 1959, p. 108). Humphrey emphasizes this aspect of emotional rhythm by asserting that "emotion, by its very nature, fluctuates; hence the dramatic rhythm pattern must show variation if it is to be convincing" (p. 108).

Doris Humphrey's thoughts about rhythm in dance are particularly striking in their kinship to some of the possible "rhythms" in music. In music, one

can at least discern similar rhythmic levels: a "beat" (pulse) level, that establishes a local regularity in terms of rhythmic organization; a phrasing level, also regular, establishing longer term rhythmic organizations whose relation to breathing has also been noticed;[26] and an emotional rhythm level, perhaps not so apparent in all music styles as in dance, but bearing the fluctuating character proper to a dramatic/expressive rhythm as defined above.

The short survey presented above portrays rhythm as a characteristic universal to dance. The insights gathered from the several sources suggest that there exist several "rhythms" in dance. For the purpose of this study, these rhythms will be grouped into three categories akin to the ones proposed by Humphrey:

1- Rhythms produced by the motor system;
2- Rhythms that establish longer-term relationships that shape the phrases in movement;
3- Rhythms associated with the expressive qualities of a movement sequence.

The rhythms produced by the motor system are what Humphrey (1959) calls the motor rhythm and Sparshott (1995) identifies as the body-centered rhythms. These rhythms have an accentual character that serves to establish the sense of beat. The rhythms associated with phrasing do not have an accentual character

[26] E.g. Fisher (1995, pp. 43-60) in his extensive review about what constitutes a phrase in music provides a definition that encompasses both tonal and post-tonal music. This definition includes a physiological condition, which states that a musical phrase "corresponds to a single, complete breath" (p. 54).

according to Humphrey (1959). They are associated with breathing and therefore establish longer-term regularities. Bartenieff (Bartenieff & Lewis, 1980), on the other hand, refers to phrasing as a movement process consisting of a beginning, middle, and end of a statement, involving a sequence of changes within movement that produce rhythmic patterns of stresses. The rhythms related to the expressive qualities of movement deal with the interaction of the Effort elements with variations in spatial patterns. This relationship is what produces different kinds of phrasing. Effort elements describe general qualities of movement and they can be perceived in a movement sequence. Humphrey calls these rhythms the "emotional rhythms," which are related to the succession of the emotional states in a dance.

The above tripartite subdivision of the possible rhythms in dance is a necessary step for the present study. In order to build an algorithm that is able to interpret dance rhythms as musical rhythms, one has to initially figure out which rhythms in dance are suitable for musical interpretation. This is important because not all dance rhythms bear a close relationship to musical rhythms. As seen in the short survey presented above, the term "rhythm" in dance refers to different aspects of movement organization in the spatiotemporal domain. The following section will further distill the possible similarities between rhythm in dance and musical rhythm by analyzing the temporal interaction between dance and music. This is an intermediate and necessary step before laying out a framework for analyzing movement as musical rhythm, the last section of this chapter.

Temporal Interactions with Music

Music and dance are temporal arts. They both structure the time domain. They are made of patterns that unfold in time, and dance is often set to music. Some similarities between the rhythms of dance and the rhythms of music were already pointed out when we analyzed Humphrey's ideas about rhythm in dance. In numerous cultures dance and music are intrinsically inseparable, and, in Western culture, dance forms have often been associated with musical genres — e.g. Waltz, Polka, Sarabande, etc. This reinforces the idea that there may be similar temporal domain processes at play in both music and dance.

Analyzing the relationship between music and dance is a difficult task as Paul Hodgins (1992) points out:

> Music and dance share certain fundamental characteristics of rhythm, structure and function so innately that it is difficult even to differentiate each discipline's method of realizing such elements. (p. iii)

At the temporal level, these similarities are also apparent as noted further by Hodgins:

> Time is the unavoidable x-axis, the structural spine upon which all musical and choreographic events are ordered. The spiritual alliance of dance and music is apparent even in certain aspects of their vocabularies — phrase, breath, pause, tempo, pulse — not surprisingly since both deal with movements of the body measured in time. (p. iii)

Hodgins's statements suggest that the problems in analyzing the relationship between music and dance (including the relationships in the temporal domain) relate more to the similarities in the vocabulary of both disciplines than to the nature of their aesthetic interaction. The similarity of their vocabularies, both consisting of "movements of the body measured in time" suggests that there may

be rhythmic realizations in dance that encounter a strong parallel in music. This reinforces the idea that certain rhythms in dance may bear qualities akin to musical rhythms.

When viewing a dance set to music, one is in the presence of two temporal structures — visual/spatial, and acoustic — that overlap in time. The tempi of both structures, as well as other elements such as breathing and phrasing, share strong similarities because the nature of their rhythmic realizations is strongly grounded in the characteristics of the human body. In the previous chapter, one has seen how musical rhythm production relates to the physical characteristics of the human body (cf. Movement, Rhythm, and Action). In the previous section of this chapter, one has seen how walking is at the very core of all movement, and the importance of the accentual character of motor rhythms in establishing the sense of beat in dance (Humphrey, 1959).

This provides a strong argument towards Hodgins's claim (1992) that both music and dance share similar vocabularies in the temporal domain, structure, and function — hence the difficulty in analyzing their interaction. In my opinion, it may be the peculiar nature of this problem that underscores the richness in dance/music partnerships during the 20th century.[27] In essence, all different ways of collaborating resulted in and coexisted as valid ways of presenting dance with music. This may be because both music and dance represent different stylized realizations of the body, expressed spatially and acoustically.

---

[27] For a good and brief description of some of these collaborations as well as how choreographers have been treating the relationship of dance to music in the recent past (since 1960) see Sally Banes (1994, pp. 310-326).

The following report on two perceptual studies on the interaction between dance and music will hopefully shed more light into the subject. One of the studies concerns the expressive and structural similarities between dance and music in a Balanchine ballet. The other concerns the perception of duration, tempo, structure, and quality of movement in a choreography viewed with several musical accompaniments in which the correspondences between dance and music were purely accidental.

Two Perceptual Studies on the Relationships between Music and Dance[28]

Carol Krumhansl and Diana Schenk (1997) attest that there exist a large variety of elements which define mappings between music and dance. These operate on different hierarchical levels and suggest non-accidental relationships between dance and music. In their study, Krumhansl and Schenk used George Balanchine's choreography set to Mozart *Divertimento* No. 15 as the stimulus for an experiment in which the participants[29] were asked to perform the following four on-line tasks:[30] indicate the occurrence of section ends and new ideas, and judge the amount of tension and emotion expressed. The participants in the experiment had to perform these tasks in one of three conditions: by listening

---

[28] Special attention will be given to these studies in this text. These were the only two perceptual studies on the relationship between dance and music I found in the literature review for this study.

[29] The participants in this experiment were twenty-seven students from Cornell University and their music and dance experience varied widely: "[o]n average they had taken 9.2 years of music lessons and 7.3 years of dance lessons, and had participated for 10.4 years in musical activities and 3.5 years in dance activities" (Krumhansl & Schenk, 1997, p.68).

[30] The subjects had to press a foot switch while watching to the choreography in all three conditions to indicate the required judgments after seeing and/or hearing the choreography once in order to get familiar with it.

solely to the music (Music Only condition); by watching the dance in silence (Dance Only condition); and by viewing the choreography while listening to the music (Both Music and Dance condition).

In the discussion of the results, the authors assert that the analyses of the data showed correlations between Music Only and Dance Only conditions in all of the performed tasks. Occurrence of section ends and new ideas tended to coincide in the music and in the dance, and the curves of tension and emotion expressed were also similar even thought the stimulus materials were completely different — such as in the case of Music Only and Dance Only conditions. Krumhansl and Schenk (1997) also noted that the delineation of temporal units by section ends and new ideas was clearest in the Music Only condition. The temporal segmentation on the Dance Only condition appeared primarily at a higher hierarchical level with fewer subdivisions. Similar findings occurred in the tension and emotion evaluations. In the conclusion to their study Krumhansl and Schenk state:

> [I]n all three conditions the four on-line tasks showed consistent temporal patterns with one another. New ideas were judged as occurring at section beginnings when levels of tension and emotion were low. These levels tended to increase throughout the sections, reaching peaks just before the section ends, and then declining rapidly. The finding that this temporal pattern appeared in all conditions suggests that a general schema for temporal organization is operating. (p. 79)

The findings on the study reported above provide some important elements to the present discussion. Krumhansl and Schenk (1997) suggest that a general schema for temporal organization is operating due to the similarities found in the segmentations in the Music Only and Dance Only conditions. These

similarities entail temporal segmentation, judgment in the occurrence of new ideas, and tension curves. They also noted that the temporal segmentations in the Dance Only condition in general appeared at higher hierarchical levels than those done in the Music Only condition. This aspect will be further dissected later.

Although the findings reported in this study are nonetheless interesting for the present discussion, such clear correspondences between music and dance may not always be so apparent. Krumhansl and Schenk (1997) note that Balanchine was indeed a choreographer who paid close attention to the musical score when choreographing. He was an accomplished pianist and amateur conductor (Hodgins, 1992), and was known for giving prior importance to time and metrical pulse, which he believed provide a structure required for sustained movement in a dance (Krumhansl & Schenk, 1997). Krumhansl and Schenk acknowledge such limitations in their study's findings and note:

> Other choreographers, notably Merce Cunningham, are known for creating marked oppositions between [music and dance]. Psychologically, these disparities might produce more complex interactions between the two components. For example, tension may be produced not only by the music and dance, but also by the disparities between them. Similarly, the relative dominance of the music found in this experiment may be a function of the particular selection chosen. (p. 79)

In another perceptual experiment, Hodgins (1992, pp. 1-7) found that music interfered with the temporal and structural perception of dance. This, experiment, carried out at the University of California, Irvine, 1986, concerned viewing a videotaped choreography in silence or with three different musical accompaniments. The participants had to answer different questions about the perceived duration, pace and variability of tempo, structure, and quality of

movement. The dance consisted of a duet of about six minutes in length and was carefully choreographed in order to be as featureless as possible in its most important characteristics. “It was deliberately made amorphous, without strong indications of structure, motive, rhythm or prevailing speed (the tempo, in fact, varied)” (Hodgins, 1992, p. 2).

Four separate groups of participants[31] were used. Each group viewed the choreography in one of the following situations: (a) in absolute silence; (b) set to the final movement of Bach’s Brandenburg Concerto No.4; (c) set to an adagio movement from a middle period Beethoven piano sonata in which the structure and meter were obfuscated by the extremely slow tempo; and (d) set to a soundscape composed by the author himself, consisting of highly dissonant and harsh synthesized sounds, elements of musique concrète, whose effect was deliberately unsettling and disorienting. In each of the three accompaniments, the correspondences between the music and choreography were completely accidental.

The results of the experiment showed a fair degree of variability depending on the setting the choreography was viewed. In regards to figuring out the duration of the dance, the group that saw the choreography with the Beethoven music and the group that saw the choreography with the electronic soundscape erred considerably. The Beethoven group perceived in general a shorter length and the electronic music a longer one. The group that viewed the

[31] The participants consisted of undergraduate students that were divided evenly between dance majors and non-majors, “all of them with varying degrees of dance experience” (Hodgins , 1992, p. 2). Each group of participants had an average of about 4.5 (± 0.3) years of training.

choreography in silence and the group that viewed the choreography with the Bach music gave the best estimates.

Regarding the structural division of the piece, Hodgins (1992) noted that the vast majority of the groups that watched the choreography with the music saw the work sectionalized. Most of the group that watched the choreography in silence saw the work as being seamless and non-divisible.

The question concerning the tempo variability (the choreography had a varied tempo) gave the most disparate results. The group that watched the choreography in silence voted overwhelmingly in favor of a varied tempo. The group that watched the choreography with the Bach music gave a 3 to 1 response in favor of a fixed tempo. The "Beethoven group" favored a varied tempo in a 60/40 split and the "electronic music group" was evenly divided. Regarding the question pertaining to the quality of movement, Hodgins noted that musical influences were not so dramatic in the evaluation.

The findings from this "small-scale and imperfect survey"[32] unveil how influential a musical setting can be in perceiving aspects that pertain to the temporal structure in a dance. Temporal features such as duration, sectioning, and tempo variability in a dance are perceived differently depending on certain characteristics of the music such as tempo, structure, and texture. Music acts subliminally in the perception of dance's temporal features. As Hodgins (1992) notes in the conclusion to his survey:

---

[32] Even though the author acknowledges these imperfections he nevertheless assumes that "certain observable tendencies do become apparent, and these would probably remain fairly consistent in a larger survey" (Hodgins, 1992, p. 6).

> What is most surprising is the degree to which these subliminal expectations permeate the entire spectrum of the dance viewer's observational aesthetic. Elements of length, tempo, structure, texture, even movement quality are all dictated to some degree by the music, even before a single movement was choreographed. Choreographers may choose to work with or against these strong subliminal influences, but one thing is certain: in many different ways . . ., what we hear exerts a profound and inescapable influence over what we see.[33] (p. 7)

Some Conclusions

The two studies reported above bring in some valuable elements for the present discussion. Krumhansl and Schenck (1997) attest that dance can reflect the structural characteristics of music in the temporal domain. Although the study concerns a particular instance of dance/music collaboration — in which the dance is deliberately set to music — the authors note that subjects can temporally segment dance and music in a similar fashion. They therefore suggest that a general schema for temporal organization is at play in both music and dance. Hodgins's study (1992) reveals how certain structural characteristics of music — namely tempo, structure, and texture — can interfere subliminally in the perception of dance's temporal structure. Music acts as a sort of structural straightjacket to dance that conditions the viewer's temporal perception of a choreography.

Both studies reveal the complexity in the temporal interaction between dance and music. On one hand, as seen in Krumhansl and Schenck's study (1997),

---

[33] And the contrary is also true. What we see also exerts an influence on what we hear. The simplest example of that are the difficulties an audience has to remain interested in concerts of (what is commonly termed) tape music without a visual component (performers, dance, video, or light). However, addressing that issue in more depth is completely outside the scope of this study.

dance can reflect the structural and expressive qualities of a piece of music. It can portray the music's temporal structure and its sectional tension curves. On the other hand, as found in Hodgins's experiment (1992), even when the music is superimposed accidentally over a pre-existing choreography, it interferes subliminally in the choreography's structural perception and temporal segmentation.

From a perceptual standpoint, music seems to provide a more important role in the temporal segmentation of a choreography.[34] This aspect is even apparent when a dance is deliberately set to music as in Krumhansl and Schenck's study. The fact that subjects were able to perform shorter temporal segmentations in the Music Only condition, illustrates that when a subject is constructing a temporal hierarchy, music plays a more prominent role. This may be explained by the fact rhythmically organized auditory patterns are easier to encode, recall, and reproduce, than similarly organized visual patterns for durations that can evoke a pulse sensation (Parncutt, 1994a).[35] Pulse sensation is a form of temporal segmentation.

Another aspect may contribute to the fact that temporal segmentations occur at higher levels. Movement actions in dance tend to span longer than in music since they are in general performed by the whole body — in music, rhythm production is in general performed by smaller body parts (wrist, fingers, forearm,

---

[34] Cf. footnote 15 from the previous chapter (section Pulse Sensation) in which I hypothesize that pulse in music may serve as a "clock" in the audiovisual setting. The last movement of Bach's Brandenburg Concerto No. 4 was the only music that had a clear and strict tempo from the three musical examples Hodgins (1992) used in his experiment.

[35] Cf. Pulse Perception from the previous chapter.

etc.). This aspect will be addressed in more detail later in the text.

This section has enlivened two very important aspects concerning the temporal interaction between dance and music. The first aspect is that the temporal unfolding of dance and music in time can have similar characteristics from a perceptual standpoint. This provides a strong case towards assuming that there is a kinship between music and dance rhythmic realizations. The second aspect concerns the role of music in the temporal segmentation of dance. Music seems to provide stronger cues in temporal discrimination and segmentation, and can even influence the temporal segmentation of dance. These aspects will be taken into consideration in the following section when we discuss how movement in dance can be interpreted as musical rhythm.

Movement as Musical Rhythm

Music and dance can represent bodily movement patterns that are perceived acoustically and visually. The interaction of music and dance in time is a hard task to analyze for the reasons exposed above. In the temporal domain, they intimately share a vocabulary due to the affinities of musical rhythm to functions of the human body. Nonetheless, music and dance developed different rhythmic vocabularies and their own rhythmic idiosyncrasies. We will see shortly that there exists a “fundamental incompatibility” (Hodgins, 1992, p. 13) between the rhythmic realizations of instrumental music and those of dance.

Developing an algorithm that is able to track the pulse of a movement sequence should imply that the movement sequence in question contains

sufficient elements that can be interpreted as musical rhythms. As we saw in the previous chapter, musical rhythm is an acoustic sequence that evokes the sensation of pulse (Parncutt 1994a). The movement elements that bear the characteristics of musical rhythm should not be acoustic of course, but should bear the timing characteristics of musical rhythms as well as have an accentual character.

I will first present the elements that point out the fundamental differences between rhythmic production in music and dance. I will subsequently provide a framework for the analysis of movement as musical rhythm that is grounded on knowledge gathered from the previous chapter about musical rhythm production, motor synchronization, and skilled action.

A Fundamental Incompatibility

Hodgins (1992) points out the qualitative differences between the types of rhythms produced in music and those produced in dance. He asserts that there exists a "fundamental incompatibility" between the gestural tempi of dance and music:

> String tremolos, woodwind arpeggios, and piano trills are all highly idiomatic and widely used musical devices, the quintessential ingredients of musical gesture. They are, of course, executed primarily by actions of the fingers, wrists and forearms. Yet even relatively small gestures in dance involve feet, arms, lower legs, and the head — larger body parts, hence larger and more time-consuming gestures. (p. 13)

Musical rhythmic action and phrasing are largely conditioned by the physical characteristics of the musical instruments. The length and shape of phrasing in

instrumental music is dependent on instrumentally idiomatic elements, such as the length and speed of a string player's bow or the force and duration of a wind player's breath. To a dancer, Hodgins (1992) notes, "these units of measurement may seem arbitrary or irrelevant …. In dance, the phrase's shape and length is determined by a much larger and less temporally precise instrument — the human body in motion, within the confines of the stage or studio" (p. 14). The differences in timing of rhythmic action in dance and music create pervasive temporal consequences all the way up the structural hierarchy: "[t]he movement phrase [in dance] frequently translates into a larger musical structure: the duration of a period, say, or even an entire section in simple binary form" (Hodgins 1992, p. 14). According to Hodgins (1992), the problems concerning the relative duration of kinesthetic and musical gestures have created intractable philosophical disagreements about dance and music interaction. For example, John Cage, in expressing his dissatisfaction with the Balanchine-Stravinsky collaboration once said their collaborations "[had] been dependent on making ten fingers seem to be like two feet, which is impossible" (Cage, quoted in Hodgins 1992, p. 13).

These differences in both disciplines unveil their own rhythmic idiosyncrasies. The idiosyncrasies on the nature of rhythmic production in music and dance are in a sense linked to the size of the body parts used. Dance in general involves wider movements performed by larger body parts. Music playing involves smaller body parts and more spatially contained gestures. This aspect may explain why timing in music is in general more precise within the temporal confinements of musical rhythms. An instrumental player is able to produce

smaller pulse subdivisions when performing music, thereby giving a more precise sense of pulse and timing, since he performs more contained gestures in space and is working with smaller body parts. In terms of bodily rhythm production, this partially supports Cage's statement about the mistake of trying to make ten fingers seem to be like two feet.

However, the essence of musical rhythm is heavily grounded on the physical characteristics of the human body as it has been thoroughly emphasized in this study. The relationship between musical tempi and walking, heartbeat rate, spontaneous and preferred tempo (Fraisse, 1982), unveils that there are rhythms in music that closely connect to those in dance. Walking, as considered by Humphrey (1959), is at the core of movement. The accentual character in the act of walking provides the awareness of accent present in all motor rhythms, the *beat* rhythms. One would then easily assume that there is a region of rhythms produced in dance that intersects with those produced in music. This is what makes the act of dancing in synchronization to music possible. This may also mean that this region of rhythms in dance would be suitable for a musical interpretation.

Finding the Common Denominator

The above discussion has shown that rhythm in dance is multilayered. Although there are similarities between the rhythmic realizations of dance and music in time, we found that certain differences were also apparent. After analyzing the fundamental incompatibilities between the movement actions in dance and in music, we became aware that the phrase level in dance is in general

extended over larger time spans than in music. We also became aware that dance involves more full-body actions than music and therefore the movement tempi in dance tend to span longer than those of music.

When I surveyed the rhythms of dance in the initial section of this chapter, I grouped them into three types that were akin to Humphrey's distinctions of what might constitute rhythm in dance. Among these types, the motor rhythms — rhythms that have an accentual character that serve to establish the sense of beat in dance — seem to be the ones that can produce a series of stresses and accents that can convey the sense of pulse. The expressive rhythms and the phrase rhythms miss at least one of the two conditions to establish the sensation of pulse — the expressive rhythms are too irregular and the phrase rhythms, although regular, do not have an accentual character.

This common rhythmic region could thus be considered to range from the pulse level in music — eventually from a pulse's first subdivision for slower pulses — up to the metrical level in faster tempi in musical rhythm. At the phrase level, as pointed out by Hodgins (1992), we already encounter different rhythmic behaviors in music and dance, although as discussed earlier, there are still similarities between what constitutes a phrase in music and a phrase in dance.

The assertion that emerged from the literature review in these two latter chapters — that certain rhythms in dance bear the qualities of musical rhythms in their durational and accentual character — will thus allow for the creation of software that processes these rhythms in real time as if they were musical rhythms. This will open a *musical* channel of communication between musicians

and dancers in computer-mediated collaborations in real time. The way this channel of musical communication is idealized will be discussed in Chapter 5.

CHAPTER IV

TRANSLATING MOVEMENT TO MUSIC: INTERFACES, SOFTWARE, AND APPROACHES

Introduction

In the two previous chapters, I analyzed pertinent literature concerning the perception and production of musical rhythm and the possible relationships between musical rhythm and rhythm in dance. This chapter will now survey the interfaces and software applications that can help detect such relationships as well as the work by other composers and creators of interactive dance that has proven influential for this study.

This chapter is divided in three sections. The first section, entitled Interfacing Movement with a Computer System, will address some possible ways of gathering movement data for analysis. Before discussing the interfaces that can be used to interpret movement, I will introduce the concept of trans-domain mapping and a taxonomy for tracking techniques. Two types of interface will be thoroughly discussed — sensors connected to a MIDI digitizer[36] that measure the

[36] MIDI digitizers can interpret data as captured from sensors — such as accelerometers, flex sensors, or ultra-sound sensors — and translate it to MIDI. These interfaces include STEIM's Sensorlab, IRCAM's AtomicPro, Infusion Systems' I-cube, and Microlab from the Institute of Sonology in The Hague, The Netherlands. All of these interfaces work similarly, and allow the user to build custom systems that capture gestural data by means of analogue sensors connected to the digitizer. The digitizer is able to output in real-time a series of MIDI control messages whose values correspond to the measurements performed by the sensors. The MIDI data generated by these interfaces can be used directly to control a MIDI device, or transmitted to a computer for further processing. These interfaces are often utilized in today's interactive performance. They are

angle of joints of a dancer, and video cameras utilizing video analysis software. The first type of interface is important to mention in this study since the two existing systems utilizing this technique for gathering movement data were developed by composers. These composers, Mark Coniglio and Wayne Siegel, have written and talked about the musical use of such interfaces, and pertinent ideas expressed by these musicians about the dancers' musical interaction with a computer system will be discussed in the last section of this chapter. The use of video cameras as movement sensors is central to the research developed in this dissertation. This study concerns the development of analytic techniques that can be applied when using video cameras to interpret movement data. The motivation for developing such techniques is in part related to the fact that, when using a fixed video camera to gather movement data, one is not interfering with the dancer's freedom of movement.

The second section of this chapter, entitled Video Analysis Software, will survey several existing video analysis software that influenced and partially supported the approach taken in this study. At the time of this writing, this software is available commercially or free. The techniques for analysis employed in this study run on top of software programs or libraries that already provide an analytic representation of a video signal. Without the existence of such software, this work would have not been possible.

---

utilized to extend the control possibilities of normal acoustic instruments or to design custom gesture controllers for performance.

Finally, the section Making Movement Musical: Approaches and Discussion will review the approaches taken by other composers and researchers of interactive dance for setting modes of interaction in real-time between dancers and interactive music systems. Three main topics of discussion in this area will be unveiled through the analysis of these authors' views. These topics pertain to the limitations of the interfaces used to capture movement in dance, the mapping of movement to musical parameters, and the possible modes of interaction between the mover and the computer.

Interfacing Movement with a Computer System

Trans-Domain Mapping

Trans-domain mapping (TDM) involves the mapping of one creative environment onto another. Within the TDM framework, meaningful activities in one domain are digitized, captured, tracked and mapped onto another domain (Ng, Popat, Ong, Stefani, & Cooper, 2000). For the purposes of this study, the term *mapping* will always refer to trans-domain mapping. The applications of TDM are immense in the arts that encounter representation in the digital domain. These applications range from interactive installations, interactive musical, dance, and theatrical performances, to other multidisciplinary events that involve the interaction of digital image with music and/or dance and theater.

In essence, any parameter from any of the fields mentioned above can be mapped onto another parameter from a different field provided that both have a suitable representation in the digital domain. For example, one can map the

position of a person in a defined space in a way that triggers musical and/or visual events in certain regions of that space. One could also control the cutoff point of a filter by mapping the angle of flexion of the elbow of a performer to values of the cutoff frequency in the filter.

Tracking Techniques

There are several ways of meaningfully translating physical movement into digital data and using the data to control musical parameters.[37] Axel Mulder (1994) distinguishes three different techniques for tracking human motion: inside-in, inside-out, and outside-in movement-tracking techniques.

Inside-in techniques use sensors placed on the body to track movement of small body parts. These sensors can be data gloves, or piezo-electric flex sensors (sensors that measure the angles of joints), for example. In general, they do not allow for measuring rotation, and are not dependent upon a fixed point for reference. Since the sensors are placed on the body, this technique is considered obtrusive. The placement and eventual wiring of these sensors can interfere with the freedom of movement.

Inside-out techniques employ sensors on the body to track external sources. They can be used to measure the movement of larger body parts as well as give information about the position of a performer on a stage. These sensors

[37] For an excellent survey of sensing technologies that can be utilized in electronic music performance see Bongers (2000).

can be accelerometers, gyroscopes, or even video cameras placed in a person's body, and suffer from the same issue of obtrusiveness pointed above.

Outside-in techniques employ an external sensor that senses artificial sources or markers on the body. These include electro-optical systems that can track reflective markers, or natural sources on the body such as external video cameras to track motion of certain body parts. Outside-in techniques are the least obtrusive of the three. These techniques are good for tracking larger body-parts, but they pose problems in tracking small body parts such as fingers. The illumination of the environment can also interfere with proper operation of the system. Outside-in techniques are in general computer intensive and limited by occlusion. Since they employ an external source for movement sensing, objects on stage may interfere with a proper representation of the body parts to track. Certain parts of the body may also not be tracked properly due to the position of the body relatively to the sensing source.

The two types of interface discussed in this study employ inside-in and outside-in tracking techniques respectively. I will also mention some interactive dance performances and installations in which they were employed.

Gathering Movement Data Using MIDI: The MidiDancer and the DIEM Digital Dance System

The MidiDancer and the DIEM Digital Dance System were systems developed respectively by Mark Coniglio in 1989 (Coniglio, 2002b) and by Wayne Siegel and Jens Jacobsen in 1996 (Siegel & Jacobsen, 1998). They were

built for the purpose of interactive dance performance, and both systems work similarly. They employ a series of flex sensors placed on a dancer's body and connected to a MIDI digitizer in order to effectively control musical data in real-time.

The MidiDancer

The MidiDancer consists of eight flex sensors and two small boxes. The first box, an encoder/transmitter, is worn on the dancer's body and connected to the sensors. The sensors measure the angles of bending body parts such as knees, elbows, and wrists. The microcomputer chip of the encoder/transmitter samples the analogue voltage values from the sensors at a speed of thirty times per second and transmits the values over a radio frequency, broadcasting them over a three-hundred-foot range. The analogue values are digitized to numbers between zero and 100, the value zero corresponding to fully straight and the value 100 corresponding to fully bent.

The receiver box of the MidiDancer contains a radio receiver and another single-chip microcomputer. After error-checking is performed on the received data and the corrupted data ignored, the valid data is converted into a stream of MIDI messages. These messages indicate the position and acceleration of the performer's limb. The MIDI output of the receiver can then be connected to any personal computer with a MIDI interface for routing to the various media devices that are being controlled (Troika Ranch, *Media/Technology | MidiDancer*, n.d.). Coniglio typically uses a Macintosh computer running a software called Interactor

LPT (Coniglio & Subotnik, 1989) to meaningfully interpret the MIDI data and send it to devices being controlled.

Tactile Diaries (Coniglio & Stoppiello, 1990), *In Plane* (Coniglio & Stoppiello, 1994), and *Reine Rien* (Coniglio & Stoppiello, 2001) are some of the pieces in which Mark Coniglio and choreographer/dancer Dawn Stoppiello use the MidiDancer for the control of musical events and other digital media.

Tactile Diaries was Coniglio and Stoppiello's first work involving telecommunications. The piece was performed simultaneously at the Electronic Cafe in Los Angeles and the NYU Television studios in New York. They used slow-scan videophone techniques developed by Electronic Cafe founders Kit Galloway and Sherrie Rabinowitz to transmit images remotely from the local performance sites using ordinary phone lines. In the final section of the work, Stoppiello wore an early version of the MidiDancer in order to use the shape of her body to determine when the images should be transmitted from Los Angeles to New York (Troika Ranch,*Works | Tactile Diaries,* n.d.). In the work *In Plane*, the dancer used the MidiDancer to control the generation of music, the recall of video images, theatrical lighting, and the movements of a robotically controlled video projector (Troika Ranch, *Works | In Plane,* n.d.). In *Reine Rien*, each of the four dancers wear a MidiDancer suit to interactively control the music and video imagery (Troika Ranch ,*Works | Reine Rien,* n.d.).

The DIEM Digital Dance System

Wayne Siegel and Jens Jacobsen's DIEM Digital Dance System works in a fashion similar to Coniglio's MidiDancer. The system is comprised of a set of sensors and two small boxes called the Dancer Unit and the Receiver Unit respectively. The Dancer Unit consists of an encoder/transmitter to which the sensors are connected. The Receiver Unit consists of a receiver that converts the data sent by the Dancer Unit into continuous controller MIDI messages.

Up to 14 bending sensors can be connected to the Dancer Unit. The sensors measure limb angles ranging from 0° to 120° and can be worn on any part of the body that bends. Two separate systems can be used in the same room at the same time, making interactive performances with two dancers possible. Siegel and Jacobsen's Receiver Unit functions as a standard MIDI controller device and can be connected with a MIDI cable to any MIDI interface or device. Sensor data is received by this unit and sent as MIDI continuous controller values from zero to 127. When the sensors are in a straight position, the Receiver Unit will send a controller value of zero for that sensor. As the sensor is gradually bent, the controller value will gradually increase to a maximum of 127 (The Royal Academy of Music, Aarhus — DIEM, *Product Description*, n.d.).

Movement Study (Siegel & Saunders, 1997) and *Sisters* (Siegel & Brolin-Tani 1998) are two pieces by Siegel that employ the DIEM Digital Dance System. *Movement Study* is a solo for a dancer wearing the Digital Dance System and computer interactive music system. A program written in Max interprets the movement data captured by the sensors of the system, allowing the dancer's

movements to control and influence the music in real-time. According to the program notes for this piece,

> An essential idea in Movement Study is that the dancer is able to influence musical processes and not simply trigger sequences and events. Since the compositional program and digital dance system were developed first, the task of the choreographer was to experiment with the hardware and software to get acquainted with the instrument and then begin to develop a choreographic idea that could integrate an artistically viable choreography with the movements necessary to control the music. (The Royal Academy of Music, Aarhus — DIEM, *Movement Study,* n.d., Choreography, ¶ 1)

Several compositional algorithms were created by Siegel for this piece that enable the dancer to control musical processes. These range from simple mappings that allow the dancer to control filtering and amplitude of sounds to other rule-based algorithms that allow the dancer to control the density of notes in a six-voice polyphonic texture, or the pace at which the piece evolves in a given section (Siegel & Jacobsen, 1998).

Sisters was composed for two dancers, each wearing a DIEM dance system, and interactive computer music. Again, a program in Max/MSP is used to interpret the movement data from the sensors. In this piece, about fifty sound files are loaded into the RAM of the computer, and these files are played back and altered in real-time using comb and resonant filters. There is no prerecorded music in this piece. The sound score is created as a direct reaction to the dancers' movements, which control the playback and filter parameters of the sound files (The Royal Academy of Music, Aarhus — DIEM, *Sisters*, n.d.).

Gathering Movement Data Non-Invasively; The Use of Video Cameras

Video cameras have been increasingly used as movement sensors for interactive performance. The advantages of using this medium in dance performance relate essentially to the fact that video cameras are non-invasive when utilized as outside-in tracking devices. They do not require that the dancers use sensors attached to their bodies, and therefore do not interfere with the dancers' freedom of movement on stage since they are in general placed outside the stage. They also do not interfere with the dancers' visual appearance on stage as in general the dancers are not required to use any special equipment for tracking.

When video cameras are used as sensors for gathering movement data, one needs a software program that processes the incoming digitized video stream and produces video analysis data to enable the interaction between the mover and a computer system. In the upcoming section, I will give some examples of works in which video cameras were used as sensors. The longer section that follows will discuss in detail some of the most important available software applications that perform video analysis.

Applications of Video Cameras as Sensors in Interactive Performance and Installations

As early as 1975, computer scientist Myron Krueger utilized video cameras in his installation *Videoplace* in order to enable the interaction of people in remote places (Krueger 1977/ 2001). Krueger envisioned that when cameras

see a person in contrast to a neutral background it is easy to digitize their outline and determine its orientation. According to Krueger "[i]t is easy also to determine if one person's image touches another's, or if someone touches a computer graphic object" (p. 114).

Media artist David Rokeby has been developing interactive installations that utilize cameras as motion sensors since 1982. Rokeby is the creator of VNS (Very Nervous System), a real-time motion tracking system that monitors the user's actions through a video camera, analyzes the data in a computer and responds to those actions (Rokeby, 2001). Composer Todd Winkler has used VNS in his interactive dance composition *Dark Around the Edges* (Winkler & Ferrero, 1997), choreographed and performed by Walter Ferrero. VNS used to be a standalone system running on a Macintosh computer. This later evolved to a set of Max externals, called softVNS. One of the latest versions of softVNS is reviewed below.

Photographer/media artist Kirk Woolford has been developing software for motion tracking in dance since 1995. One of his early collaborations with dance was the development of a tracking system for *Moving Target* (Charleroi/Danses, 1996), a choreography by Frédéric Flamand for the Charleroi/Danses dance company. Other works by Woolford utilizing video cameras as sensors for interactive dance performance include the software for *Contours* (Kozel & Woolford, 2000), *Viking Shoppers* (Igloo, 2000), and *Winterspace* (Igloo, 2001). All the applications developed by Woolford for dance

are created for a specific purpose, and they use motion-tracking techniques to allow the dancers to control computer-generated imagery in real-time.

The Palindrome Intermedia Performance Group, directed by choreographer David Wechsler, has been using video analysis technology for the control of music and other media since 1995 (The Palindrome Intermedia Performance Group, *Repertory,* n.d). This performance group has developed several hardware and software platforms for the performance of interactive dance works that include Eyecon (a camera-based motion-recognition system), and wireless sensor systems that measure muscle contractions or heartbeat (The Palindrome Intermedia Performance Group, *Palindrome Performance Systems*, n.d.). Eyecon will be briefly reviewed in the upcoming section.

Most of the examples given above were custom-built systems that required the use of non-trivial computer hardware such as expensive video digitizing cards. Since the creation of BigEye in 1995 at the STEIM studios in Amsterdam, The Netherlands, several computer programs that perform motion capture and analysis in real-time have become available for desktop computers. These programs consist of stand-alone applications or libraries that run on other software programs including visual programming environments such as Max/MSP. This software in general allows the use of more trivial hardware for motion capture such as desktop webcams. A survey of some of the most important programs, especially those that motivated and inspired the current study, will be presented below.

Video Analysis Software

The video analysis software applications that will be surveyed for this study are intentionally designed for artistic purposes, and are used in interactive installations or interactive performance. Some of these computer applications helped shape the path of this study. BigEye, Cyclops, softVNS 2, and EyesWeb are the software applications that I will discuss in detail. These applications played an extremely important role in the development of the algorithms for this study. At the end of this section, I will also give examples of other existing video analysis software.

BigEye

BigEye (STEIM, 1995) is a program that converts real-time video data to MIDI messages. The user configures the program to extract objects of interest based on color, brightness, and size. These objects are singled out through the use of image filters. The video screen is divided into user-defined regions ("hot zones"), in which a set of MIDI commands is executed when the object is detected in the zone, moves within the zone, or leaves the zone. BigEye has two modes of operation: a *simple mode*, and a *scripting mode*. In the *simple mode*, the user can specify a set of MIDI commands when an object interacts with a region in the three ways mentioned above. The *scripting mode* uses a scripting language that allows for the programming of more complex interactions between the objects and the regions.

When operating in BigEye's *simple mode*, the user determines the behavior of each one of the messages by using pop-up menus with pre-defined MIDI messages and scrolling numericals for the attribution of values to those messages (see Figure 1).

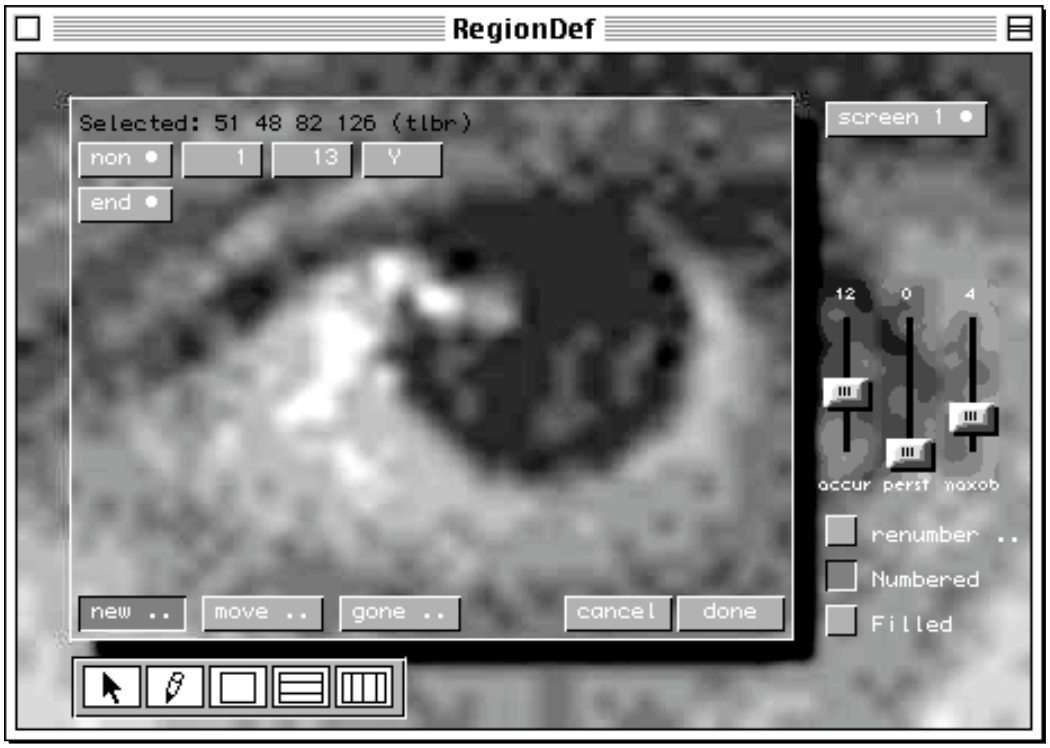


Figure 1. BigEye's "simple mode" of interaction.

In the above figure, we can see a screenshot of BigEye working in such mode. The "new" message (leftmost button in the bottom of the window) produces the MIDI note on message when an object enters the region defined in screen 1. The values of the note on message are: channel 1, note number 13, velocity equivalent to the object's Y coordinate when entering the region. One can in the same way define simple actions for when the object moves within the region or leaves it.

In BigEye's *scripting mode,* a user can write a script utilizing BigEye's scripting keywords and predefined variables. These can be combined into procedures that can be executed at the script's run time. A BigEye script usually consists of a *global* script, a *start* and a *stop* script for each screen, and a *new*, *mov*e, and *gone* script for each of the regions defined for that screen. The *global* script is never executed directly in response to any event in BigEye. It holds the code for the procedures called by the other scripts and the global variable definitions. The *screen* scripts consist of a *start* and *stop* script that get executed whenever a screen is selected (*start script*) such as in the "init & run" command, the command that initializes and runs a script in BigEye. The *stop* script gets executed whenever one changes the screen. The *new*, *move*, and *gone* scripts are executed whenever an object enters, moves within, or leaves a region defined in a screen.

Some libraries that do image grabbing and analysis became available to the Max programming environment. Max external objects such as **Cyclops** (Singer 2001) and libraries such as SoftVNS 2 (Rokeby 2003) offer robust video analysis facilities in the Max environment. These objects/libraries take advantage of Max's modular architecture, thereby allowing the user to customize its applications to a greater extent than in stand-alone applications like BigEye. Below I provide a survey of these objects and libraries available in Max.

Cyclops

Cyclops[38] (Singer, 2001) is a Max external object that receives and analyzes video input developed by Eric Singer. It receives input from a QuickTime video input source, analyzes the video frames, and outputs messages based on the images. In order to process video, **Cyclops** divides the digitized video image into a low-resolution grid of equally-spaced rectangles called blocks. Blocks average the pixel values within that area of the digitized image. They yield an 8-bit greyscale value and/or 24 –bit color value for that block, depending on the type of analysis to be performed (one can choose to perform greyscale and color analysis in the same block). Zones are locations in the video frame defined by the user for analysis. The zone location determines which block will be analyzed and the type of analysis to be performed. The types of analyses available in **Cyclops** are greyscale, threshold, difference, and color.

Greyscale analysis outputs the greyscale value of a zone from 0 to 255, with 0 corresponding to black and 255 to white. Difference analysis outputs the difference in greyscale value of a zone between consecutive frames. If the grey value of a zone goes from black (0) to white (255) from one frame to the next, the output for that zone will be 255. If the reverse happens –255 will be the output value. Threshold analysis checks if a certain greyscale value between 0 and 255 has passed a previously defined threshold value. It outputs zero or one depending if the threshold value was passed or not.

---

[38] Throughout the dissertation Max objects will appear in bold typeface in the text. This is a convention commonly used in literature about this software. See for example *Max Reference* (Zicarelli et al., 2001) or Winkler (1998).

Color analysis can be done in four different ways using the modes *color*, *match*, *closest*, and *closestmatch*. The mode *color*, outputs the RGB value for the color detected in a zone. The remaining modes for color analysis perform comparisons between a detected color and preexisting colors stored in **Cyclops**'s palette window. These colors can be stored either by grabbing a color from the Macintosh's Color Picker system extension or by sampling a color from the actual video input. *Match, closest,* and *closestmatch* are three different modes of color comparison a user can perform between the live input image and the colors stored in **Cyclops**'s palette window.

Cyclops played an important role in the development of this study. It was the first object I tested in the Max environment that performed the types of video analysis I wanted to do in order to start testing certain hypotheses for the real-time detection of periodicities in movement. In figure 2, we can see a Max patch that I designed that outputs the greyscale difference (*diff* mode) in 192 zones placed over 192 blocks, corresponding to a low-resolution grid of 16x12 blocks. The output of **Cyclops** in *diff* mode, gives an indexed value of the difference in greyscale values between frames of each zone. The output of the values of greyscale difference for each block in the frame is temporarily stored in the Max **coll** object and subsequently summed so that we get the total amount of differences in greyscale values in the whole frame. The graph in the bottom of the patch window shows a visual representation in time of the values produced from the movement of a waving hand in regular pendular motion. As it can also be seen, the graphical output is a sinusoid wave, with the peaks corresponding

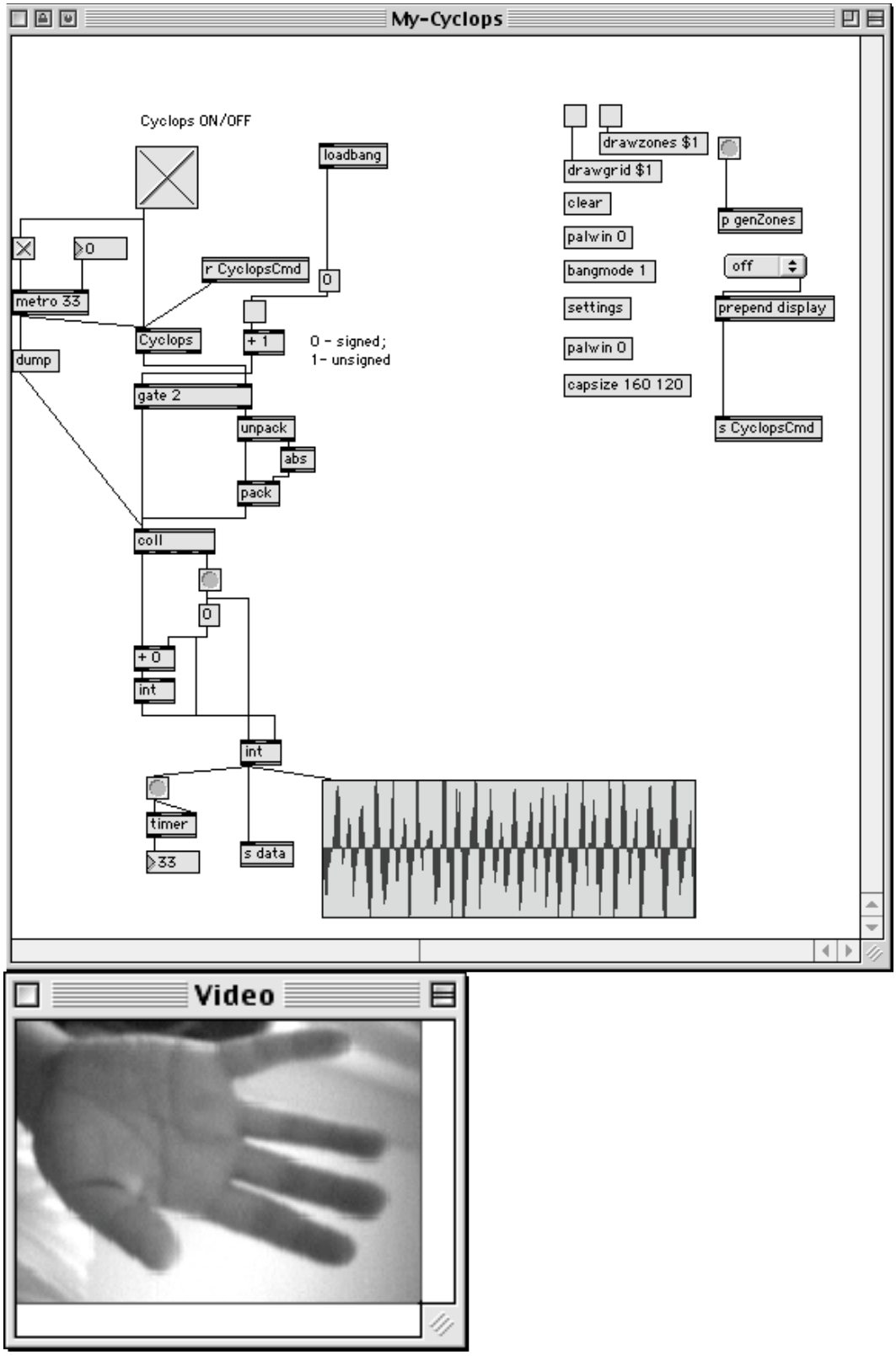


Figure 2. Max patch illustrating the use of **Cyclops** for movement analysis.

to the inflections in the pendular motion. This made me realize that it may be possible to detect periodicities in movement, as captured from a fixed video camera in real-time, utilizing a modular programming environment such as Max.

SoftVNS 2

SoftVNS 2 (Rokeby 2002) is the 6th generation of video processing systems created by David Rokeby for his own interactive sound and video installations (Rokeby 2003). SoftVNS is a software adaptation of VNS or Very Nervous System for the Max programming environment. The original *Very Nervous System* (Rokeby, 1983) is an interactive sound installation developed by David Rokeby, first exhibited in 1983. SoftVNS 2 is a collection of almost 100 external objects for Max. It runs only on Macintosh computers that have a G4 or G5 processor, since it makes full use of the velocity engine available to this generation of processors.

In softVNS 2, the video streams are passed from object to object through normal patch cords. A video stream is characterized by its width, height and format. The width of a video stream is the number of pixels in a single horizontal line of a frame of the stream. The height is the number of lines in a single frame of the stream. Finally, the format is the way in which the pixels are described. SoftVNS 2 supports three types of formats: *grays*, *yuv*, and *rgb*.

The greatest strengths of softVNS 2 are the quality and speed of its video analysis objects and its versatility. SoftVNS 2 has objects that perform various types of analysis on a video stream. The objects **v.motion** and **v.colourfilter**

perform frame differencing and color tracking respectively. The object **v.presence** can detect the presence of an object in a video stream through background subtraction.[39] **v.regions**, divides the video into regions for tracking. The objects **v.edges** and **v.silhouette** calculate the edges in an image, and find and highlight silhouettes respectively. The object **v.heads** can track heads from the output of **v.silhouette**. Finally, **v.track** follows a specific small object across the video field. The great advantage of a modular architecture containing objects with dedicated types of analysis is that the user can combine different modes of analysis for a specific need.

Another great feature of softVNS 2 is that it works with both live video input and with QuickTime movies. One can digitize live video input with the object **v.dig**, from a normal video camera connected to a video card, a USB webcam or Firewire DV camera, or read a QuickTime movie with the object **v.movie** in order to perform the video analysis. This feature has been of great help in order to refine the study I am developing, since one can devise an algorithm for analysis and apply it to both pre-existing videos and live input. **Cyclops**, for example, does not have this flexibility. Henceforth, all the Max/MSP examples in this study, including the patch built for the piece that demonstrates the m-objects, will use softVNS 2 to perform the analysis of the video stream.

---

[39] Background subtraction is a technique that stores a reference image and then subtracts that image from an incoming digitized video stream. This is a helpful technique for determining the silhouette of objects that entered the area previously stored in the reference image.

EyesWeb

EyesWeb (DIST, 1999) is another modular visual programming environment that was developed by Antonio Camurri and his colleagues at the Laboratorio di Informatica Musicale DIST, at the University of Genoa, in Italy. This project began approximately in 1997, and is based on previous research done by this group on the interaction between gesture and music. One of the main aims that supported the creation of EyesWeb was to surpass the common interaction metaphor of hyperinstrument (Machover & Chung, 1989). As stated by Camurri, et al. (2000):

> We are interested in interaction metaphors like multimodal dialogue and social interaction, including expressive content and emotional communication….
>
> A general goal of the EyesWeb project concerns the modeling of composition and performance environments in terms of communities of *emotional agents* (p. 58).

The use of emotional agents as the vehicle for interaction will be explained below in the section Making Bodily Movement Musical: Approaches and Discussion.

The EyesWeb project is built on an open hardware platform that incorporates different sensor systems including black and white, infrared, and color video cameras, special electronics for the multiplexing and synchronization of two video cameras, and wireless body sensors. The modular software program includes a visual interface that allows the user to design patches to configure the system, and to observe and extract the desired movement information (Carmurri et al., 2000). EyesWeb runs on PC-compatible computers.

The basic libraries of this system include movement capture modules, filters, output modules, and observers. The movement capture modules act as inputs for the different sensor systems. Filter modules include low-level filters for preprocessing, denoising, etc, and high-level filters that can extract, for example the barycenter (center of gravity) coordinates of a human figure. The output modules include MIDI, DMX, TCP/IP, including the communication of expressive content to external applications. The observers are high-level modules or patches that reconstruct a view or understand a particular aspect of the style of movement. Observers usually study the dynamic properties of movement (Camurri et al., 2000).

Camurri and colleagues have developed several patches implementing these observers that can be classified in two categories: observers that study the gestures performed by the dance inside the Kinesphere,[40] and observers able to identify the movement of the Kinesphere as a whole entity in the general space (Camurri & Trocca, 2000). The observer models reported so far by Camurri et al. (2000) include a posture-analysis observer; an observer for the tendency to movement and equilibrium; a microdance[41] analysis module; stability, mobility and Laban parameters; and a space-and-time rhythm observer.

---

[40] The Kinesphere is the three-dimensional space reachable by the human body in upright position (Bartenieff & Lewis, 1980).

[41] A microdance "is a minimal set of movements that permits us to create a context suitable to study the expressivity of movement" (Camurri et al., 2000, p. 60). A duration of a microdance ranges between 20 seconds and 2 minutes.

Other Video Analysis Software

The software reviewed above contributed significantly for the present study. Other existing programs that perform video analysis that are utilized in interactive dance performance will now be briefly described.

Isadora (Coniglio, 2002a) "is a software program designed to allow interactive, real-time manipulation of digital media, including pre-recorded video, live video, sound, standard MIDI files and more" (Coniglio, 2002a, p. 1). Isadora is a visual programming environment akin to Max/MSP and EyesWeb in which the users can create programs by linking together modules (*actors,* in Isadora jargon) that perform a specific function on the media. These programs can be made interactive by controlling the actors with a *watcher*, a type of module that looks for information from the outside world — e.g. MIDI messages, mouse or keyboard actions. Results of the actions and interactions performed by Isadora are presented through the computer's video screens, speakers, or MIDI interfaces. Isadora reflects Mark Coniglio's desire to make a powerful, flexible and reliable tool to make his own pieces for his theater dance company Troika Ranch, and to create a friendly working environment for those who do not have extensive experience in working with computers.

MvM, the acronym for Music via Motion, is a project carried out by Kia Ng and colleagues at the Interdisciplinary Center for Scientific Research in Music (ICSRiM), at the University of Leeds in the UK. MvM uses live video data as input, and detects and tracks visual changes (motion and color) of the scene under inspection. The software prototype is designed to be intuitive and user-friendly to minimize the time needed for familiarization (Ng et al., 2000). The basic mapping

functions of MvM include a distance-to-MIDI-events module with many configurable parameters (pitch range, scale types, etc). Musical mapping can be enhanced with a database of composed musical material, and several mapping layers can be overlaid in order to produce multi-layered and polyphonic effects. The aim of MvM is to turn the whole body of the user into a musical instrument interface, which determines the tempo, volume and audio generation of the performance (Ng et al., 2000).

EyeCon is another video analysis program for PC-compatible computers that is used to control other media such as lighting, sound, and digital image in real-time. This software platform was developed by the Palindrome Intermedia Performance group in Germany, and is commercially available. Eyecon digitizes the input from a video camera and allows the user to specify regions or portions of the video field that trigger events when a dancer passes by it. It can also measure the amount of motion occurring in a certain region. Additional features of the program let the user track the position of persons within the performance area, measure their height, width, overall size, or the degree of left-right symmetry in their shape. These control elements may be assigned to different outputs simultaneously or in any combination. Multiple cameras may be used, though not simultaneously (The Palindrome Intermedia Performance Group, *EyeCon Support and Download Page,* n.d.).

Making Bodily Movement Musical: Approaches and Discussion

The development and consequent generalization of interactive technology that enables some type of musical control through bodily movement has generated a discussion among composers and artistic developers of interactive dance systems. This discussion entails issues such as (a) what types of mappings could be performed between physical actions and musical parameters; (b) what are the roles of composers and non-music performers in such environments; and (c) what are the kinds of music that can be produced when one uses the body for the generation of musical material. Below, I will survey some of the approaches taken by other composers and researchers in addressing these issues. The views of Richard Povall, Wayne Siegel, Todd Winkler, Mark Coniglio, and Antonio Camurri will be analyzed. I will then discuss the main points in their views and conclude this chapter by presenting my own ideas on the subject.

Richard Povall

Composer Richard Povall provides an interesting perspective on the use of interactive technology in dance. By noting the return to a certain conservatism in compositional attitude from experimentalism — prevalent for the most part of last century's music until the late seventies — the composer sees the interactive environment as a new way of challenging the maker/performer paradigm. “[A]n interactive environment creates a framework that demands radical ways of thinking and creating” (Povall, 1998, p. 3).

In the recent work with *half/angel,* Povall (2000) speculates about ways of capturing movement data that may not provide a detailed interpretation of the human body:

> I have rejected the more common methodologies of moving further and further towards a literal, accurate, detailed interpretation of the physical body, and instead use speed, direction, acceleration, and size of moving objects to gain an approximation of the kind of movement that is occurring in the performance space. Although I am able to sense an object's position in space, I rarely use this data, usually considering it irrelevant or unnecessary. Instead, I am looking instead (sic) for information from the movement that tells me how, and therefore possibly why the performer is moving. I am searching as much for the emotional intentionality of the movement as I am attempting to map the actual, literal physical movement. (p. 64)

Povall puts these ideas into play using BigEye's virtual object feature. This feature in BigEye generates a virtual object that carries on where the physical object left off. If an object is traveling leftward on stage with a given acceleration and speed and suddenly stops, the virtual object will keep on going for a little longer. According to Povall, "[t]his gives [the] data a roundness, a fluidity, that it would not otherwise have. The data flow doesn't just stop when the body stops, it sort of gently fades and dies (like we all do, I suppose)" (Povall, 2000, p. 65).

In the discussion to his approach in dealing with interactive dance systems, Povall points out pertinent aspects about movement and its subsequent interpretation by a computer. He notes that the physical manifestation of a choreographed or improvised movement is more than a simple physical manifestation. He thus reminds us that when attempting to analyze or capture movement on stage one must be aware that there is more to movement than the

movement itself. Dance movement does not exist within a single body, but within the context of the performance space and the emotional space of the work:

> Much of what is communicated to the audience is implicit in the movement, not overt. Ever watched dance on video? How rarely it is successful. Dance is arguably the performing art that translates the least successfully to the screen, particularly the small screen. This is at least partially a function of what the camera – the objective electronic eye – cannot see. (Povall 2000, p. 66)

Povall considers that the objective physical phenomena of movement are just a tiny part of what we see when we watch dance. He consequently claims the need for designing software environments "that can supplement the dumb obeisance of the camera to the absolutely overt, and begin instead to let the machine see, and enjoy, the implied action and the emotional state that is driving the overt action" (Povall, 2000, p. 68). Rather than attempting to build a system that is objectively human-like, he attempts to design a system that models human subjectivity while interpreting movement data in performance.

Wayne Siegel

Wayne Siegel and Jens Jacobsen (1998) define interactive dance as dance utilizing a computer interface in which a dancer is able to influence the musical processes, while the music in turn affects the dancer. In defining interactive dance, Siegel and Jacobsen point out that the term "interaction" is often confused with automation. "Automation refers to a process that is self-operating or automatically controlled by mechanical or electronic means, as illustrated by the concept of pushing a button and having something happen" (Siegel & Jacobsen,

1998, p. 29). Interaction, the authors contend paraphrasing Goebel (1988), implies interplay between two equal parties. The authors' speculation about the aesthetic consequences of the use of an interactive dance system motivated the creation of the DIEM Digital Dance System and the composition of Siegel's *Movement Study* for interactive dance. Siegel's central concern was to find out the consequences of giving a dancer the control over the compositional process during a performance.

Siegel and Jacobsen (1998) say that giving the dancer control of musical performance aspects opens new possibilities for precise synchronization between dance and music outside the framework of linear time. However, they draw attention to three problems that arise in the transference of the musical instrument paradigm to a dance interface: (a) dancers are not musicians with specialized training in performing a musical work; (b) the visual appearance of the dancer's movement is of great importance whereas the instrumentalist's is not; (c) if a dancer is asked to perform as an instrumentalist in the context of a dance performance, the effort required to control the music may make it difficult or impossible for the dancer to concentrate on dancing. They finally conclude that "[t]he artistic success of a composition for interactive dance will, to a large degree, depend on the ability of the artists involved to find suitable models of mapping gesture to sound, and also on their ability to integrate music and choreography into a single coherent work" (Siegel & Jacobsen, 1998, p. 42).

Todd Winkler

Composer Todd Winkler (1995) doubts if movement data can be simply applied to pre-existing musical models. He notes that the timbral characteristics of instrumental sound reflect in some way the effort and energy used to create it. Physical constraints produce unique timbral characteristics, and suggest musical material that will be idiomatic or appropriate for a particular instrument's playing technique. Winkler wonders if human movements have constraints similar to musical instruments, which might suggest something akin to idiomatic expression.

He contends that the underlying physics of motion lends insight into the selection of the musical material, but simple one-to-one correspondences, such as weight to density or register, or tension to dissonance, may not always be musically successful. The composer's job then, is not only to map movement data to musical parameters, but also to interpret these numbers to produce musically satisfying results. According to Winkler, the awareness of the underlying physics of motion can lead to provocative and intriguing artistic effects by subverting the mappings of movement to musical parameters and creating models of musical response unique to virtual environments:

> More furious and strenuous activity, for example, could result in quieter sounds and silence. At the same time, a small yet deliberate nod of the head could set off an explosion of sound. Such "unnatural" correlations [make] motion all the more meaningful. (Winkler 1995, p. 263)

In two subsequent papers (1997, 2002), Winkler describes his approaches taken for mapping movement to sound and video in the creation of several interactive dance pieces using David Rokeby's VNS.

Mark Coniglio

In addressing sensor technology to capture movement, composer Mark Coniglio (2002b) recognizes that for dancers to really play the MidiDancer in a way that the audience perceives it, they need to move like musicians. The movement of the dancer needs to be in service of the sound or image that she is generating, thereby creating serious problems for the pieces using the MidiDancer to work choreographically. Coniglio acknowledges that putting sensors in the joints was a correct decision in order to make the audience perceive the changes caused by angular changes. Yet he has realized lately that this is not the gestalt that we perceive when we watch a dancer: "We really see energy — that's a bit vague, but it's the best word I think of to describe it. We're not looking at the individual angles of the joints, but the way that the dancer moves through space and the overall articulation of the movement" (Part I, ¶ 19).

Antonio Camurri

Antonio Camurri and his colleagues at the Laboriatorio di Informatica Musicale DIST in Genoa, Italy, have done groundbreaking work in interactive dance systems. Their modular programming environment EyesWeb has been systematically developed since 1997, and this group of computer scientists have

developed important concepts in human/computer interaction. The discussion about their work will take a little more space than the previous authors as I will discuss three important concepts developed by them: emotional agents, KANSEI information processing, and their computational model of Space Effort.

Emotional Agents

Emotional agents are software agents[42] that possess an emotional state[43] (Camurri & Coglio, 1998). An emotional agent integrates different components into its architecture: input processing for the surrounding world (real or virtual), cognitive (or rational) processing, emotional processing, reactive processing, and output processing. Emotional agents are also able to respond to the input in real-time (Camurri, Coglio, Coletta, & Massucco, 1997; Camurri & Coglio, 1998; Camurri & Trocca, 2000). The input to an emotional agent can be data captured from sensors or messages from other agents. The output produced by an agent can be sounds or animations, or messages to other agents. The reactive, rational, and emotional processing, or elaborations available in an agent, are intended to produce a richer and more stimulating interaction between agents and human users.

Reactive elaborations map inputs to outputs directly and instantaneously. Emotional elaborations use the input from sensors or other agents to modify their

---

[42] For a detailed description and discussion about several types of software agents see (Bradshaw, 1997).

[43] An emotional state in a software agent involves the codification of artificial emotive states in the software, akin to human emotions, in order to facilitate the interaction between computers and humans (e.g., see Camurri & Coglio, 1998, p. 24).

emotional state, which in turn modify their output. Finally, rational elaborations use the input to update high-level knowledge about the external world and/or the agent itself, and produce an output based on such inferences. These elaborations do not operate in isolation, and can influence each other in various ways — e.g. reactive elaborations may vary according to the emotional state (Camurri et al., 1997).

In the application of this agent architecture to interactive dance, an agent is able to extract from humans some gesture or movement features, and control the generation of sound and music. That is, an agent is able to construct an analysis of a gesture of a movement sequence and interpret it in some way:

> At the beginning, the agent is a *tabula rasa*, and nothing is evoked by movement; the system is simply trying to identify features of the "style of movement" of the dancer. If the dancer begins moving with nervous and rhythmic gestures in roughly fixed positions in the space, thereby evoking the gestures of the percussionist, the agent, after a few seconds of observation might construct a continuous transformation to a set of virtual drums located in the space that corresponds to the dancer's movement. 'Continuous' refers to the fact that 'neutral' sounds begin to emerge and transform progressively into drums, e.g. gradually reducing the attack duration and moving from a default to a specific timbre…. The timbral and amplitude characteristics of the drum can be associated with the interpretation of the dancer's movements. The dancer is therefore now allowed to play the instrument that has been created. (Camurri et al., 2000, p. 59)

Emotional agents are also able to adapt continuously their output and behavior toward a different context, or they might learn the behavior of the dancer and start behaving semi-autonomously becoming a clone of the dancer — "The dancer is then the creator of and molder of the characters emerging on stage. After the creation of a character, the dancer may interact with it or leave it and move to

other places on the stage and create other clone dancers" (Camurri et al., 2000, p. 59).

In a performance with emotional agents, the composer/designer introduces sound and music knowledge into the system, specifying goals and aspects of the integration between music and gesture, as well as the number of degrees of freedom left to the agent regarding compositional choices. This raises interesting issues about new perspectives for the integration of music and movement languages:

> We can easily conceive more sophisticated examples of interaction in which the dialogue between a dancer and the active space can evolve, basing for example on learning, imitation, dialogue: the interaction metaphor can therefore vary from 'musical instrument' to 'orchestra conduction,' to different kinds of 'social interaction' under the control of the composer. The role of the composer is now to define the 'character' and 'learning capabilities' the dancer interact[s] with, rather then writing a predefined score. (Camurri & Troca, 2000, p. 101)

Camurri and Trocca (2000) also call for a deeper analysis of movement in order to overcome naïve uses of interaction mechanisms that can be noticed in art installations and hyperinstruments. To overcome these issues, they developed models of movement analysis based on KANSEI information processing (Hashimoto 1997) and Laban's effort theory.

KANSEI

KANSEI information processing is a new branch of information processing technology born in Japan. KANSEI is a Japanese word that does not have a direct translation into English. The concept of KANSEI is strongly tied to

the concept of personality and sensibility. It relates to the abilities that humans have for resolving problems and processing information in a faster-than-usual and personal way. Traces of KANSEI can be noticed in human actions, and personal ways of thinking and resolving problems. This type of information processing is not a synonym for emotion, but instead it refers to the human ability of processing information in ways not just logical (Camurri & Trocca, 2000).

An artist expresses his KANSEI through his work and performances, and a skilled actor or dancer will mimic the KANSEI required to make the character she is simulating more believable. A spectator will use her KANSEI in order to evaluate the work of an artist, to extract meaning and information about the perceived KANSEI. "Basically, it is possible to summarize one of the aspects of KANSEI Information Processing as a process that permits us to personalize the way in which a message is sent, an action is performed, choosing solutions suitable for the personality and sensibility of the performer" (Camurri & Trocca, 2000, p. 97).

Computational Model of Effort Space

Camurri and colleagues developed a model of effort space they consider useful for a better understanding of effort theory. This model for the analysis of movement just takes into account two effort parameters — space and time.[44] In their model, space is intended as a measure of directness and of the path followed

[44] This computational model of effort space could be better termed as a computational model for the alert state, since this inner state is the combination of space and time efforts (cf. fn. 23 on Chapter 3).

in space by the movements; time is a measure of the impulsive or sustained nature of movement. They consider the flow and weight hard to evaluate parameters directly. Flow is a complex parameter that was almost neglected by Laban in the eight basic efforts (efforts that take into account pairs of simple efforts). The weight parameter may be difficult to evaluate since in the case of free body movements "the performer must mimic the use of strength through the use of appropriate muscles' counter tensions" (Camurri & Trocca, 2000, p. 99).

Discussion

In the above section, one can identify three trends of discussion in the problem of generating/transforming musical material out of human movement: the influence of the gestural interface on movement capture; the mapping of movement to musical parameters; and possible modes of interaction between the mover and the computer. Interfacing, mapping, and interaction modes are three different aspects in the problem of producing music out of bodily motion in dance. These aspects are interrelated. The type of interface being used conditions the type of mappings that can be done and the type of mappings to be done condition the possible modes of interaction between a performer and a system.

In regards to the use of the interface, Coniglio's remark (2002b) about the restrictions imposed on the dancers by the MidiDancer, and Povall's remarks (2000) about the limitations of video cameras are important to retain. Coniglio and Povall acknowledge that there is more to dance than bending of joints or even

the raw physical motion. There is an energy, a gestalt in dance performance that cannot easily be captured by an interface and therefore interpreted by a computer.

In regards to the way a dancer can interact with a computer system, two different approaches have emerged from this review. One approach, followed by Winkler (1995) and Siegel (Siegel & Jacobsen, 1998), consists in the direct mapping of bodily actions to musical parameters. The other approach, followed by Camurri and Povall, consists in establishing interaction modes that rely on some type of movement analysis being done by the computer in real time.

Mapping bodily actions to musical parameters is a challenging task. Winkler provides a good insight into the problem. Although he is aware that instrumental sound reflects in some way the effort and physical energy used to produce it, he questions if the same paradigm can be simply transposed for mapping bodily actions to musical parameters. He suggests that being aware of the underlying physics of motion can lead to provocative and intriguing artistic effects, by subverting the mappings of movement to musical parameters and creating models of response unique to virtual environments. Both Winkler and Siegel consider that it is the composer's job to find appropriate mappings between gesture and sound in a piece of interactive dance. However, they provide very few clues about what may be an appropriate mapping.

The establishment of interaction modes that rely on some type of movement analysis seems to provide a broader range of possibilities for the work in interactive dance. The opinions of Povall (2000) and the work of Camurri colleagues (Camurri & Coglio, 1998; Camurri et al., 2000) provide two examples

about how can one go beyond direct mappings between movement and music. Povall moves away from a detailed interpretation of the movement sequences being performed, by letting the computer make its own representation of a certain movement sequence and produce results that have a certain degree of randomness. Camurri and colleagues engage the task of creating models for movement analysis, borrowing from LMA and KANSEI Information Processing. The elaborate software agent architecture implemented by them allows the composers to set up modes of interaction between a dancer and the system based on knowledge about movement analysis.

A Personal View

As a composer/researcher of interactive music/dance systems, I think the developments to be made in this field should be in the direction of setting more elaborate and subtle ways of interaction between dancers and interactive computer systems. These ways of interaction should take the idiosyncrasies of dance into account. By *taking the idiosyncrasies of dance into account*, I mean essentially two things as it concerns interactive dance performance: (a) the interaction of a dancer with a computer system should be done by creating interaction modes that rely on some type of movement analysis; (b) the analysis of movement should be done non-invasively.

The creation of interaction modes that rely on some type of movement analysis provides a richer way for interaction than simply performing direct mappings of movement actions to music. Devising some type of computational

algorithm that can provide an interpretation of dance movement is an important stage in the process of taking the idiosyncrasies of dance into account for the interaction with a computer system. The work by Camurri and colleagues provides an excellent example of how higher levels of interaction can be attained by applying existing theoretical knowledge in the creation of models for movement analysis. This can lead to a higher level of interpretation of movement and therefore promote more refined ways of interaction.

The use of a non-invasive sensor for capturing movement in dance is also an essential aspect in this process. I personally do not favor the application of the hyperinstrument paradigm to interactive dance performance, or any other type of interaction mode that involves the operation of an interface by a dancer. Trying to make a dancer operate an interface for the detection of movement undermines the essence of dance itself because dancing is not about operating interfaces. On the contrary, in interactive music performance for example, making an instrumentalist operate an interface that is an extension of her own instrument does not undermine the essence of music performance — playing a musical instrument is to a great extent to operate an interface. Musical instruments are interfaces that translate physical actions into sound.

The software I am developing as a consequence of this study creates a *musical* mode of interaction between dancers and musicians in computer-mediated collaborations. In this particular study, a *channel of musical communication* is established by performing certain types of analyses in dance movement that interpret certain rhythms in dance in a musical way. The affinities

between certain types of rhythmic articulations in dance and musical rhythms were noted in the previous chapter; they provide a strong motive for developing analytic techniques that can identify *musical rhythms* in dance and thus facilitate the creation of a channel of musical communication between dancers and musicians. Determining such type of articulations in dance movement could allow for interesting ways of musical dialogue in interactive dance performance. For example, one could create musical interaction modes that trigger and transform musical algorithms according to the rhythms performed by a dancer. In other situations, this type of analysis could enable a dancer to control the tempo of a musical sequence.

Conclusion

This chapter ends the literature review part of this study. Gathering knowledge in these fields was a necessary step in order to provide the background for developing a software system that can trace musical periodicities in dance movement in real-time, and use them as the vehicle of interaction between a dancer and an interactive computer system.

In this three-fold chapter, I initially discussed the possible ways of interfacing movement with a computer system by introducing the concept of trans-domain mapping, a taxonomy for tracking techniques, and then by analyzing two common types of interfaces that are utilized in interactive dance performance.

Next, I presented a survey on existing programs and software libraries that do video analysis. This software survey was of great importance to the study, as it

helped finding the type of video analysis software and techniques that could be utilized in the analysis of dance movement. As I explained in the previous section, the creation of modes of interaction in dance that take the idiosyncrasies of dance into account should perform analyses of movement non-invasively. Video cameras, and the myriad of software libraries for video analysis available to visual programming environments such as Max or EyesWeb, currently offer the best solution for the creation of interaction modes based on non-invasive analysis of movement. This has to do with the fact that one can easily create additional software modules that utilize data from these libraries, in order to perform the analyses. Moreover, there has been work done in the area of real-time analysis of dance movement utilizing these libraries in visual programming environments, such as the work of Antonio Camurri and colleagues in EyesWeb.

In the third section of this chapter, I reviewed some approaches taken by composers and other creators of interactive dance in order to assess the strategies of these authors in establishing interaction modes between movement and music in interactive dance performance. I finally expressed my own personal view about this issue. I suggested an approach for the creation of more elaborate and subtle ways of interaction between music and dance that took the idiosyncrasies of dance into account; i.e. these modes of interaction should rely on some type of movement analysis, and this analysis should be done non-invasively. The analyses of dance should apply existing theoretical knowledge about movement in order to provide elements that can foster refined ways of interaction in real time. The theoretical study in the two previous chapters of this dissertation was a necessary

step in order to develop analytic techniques that can provide information about the rhythmic articulation of dance in time and its relationship to musical rhythms.

In that respect, this study has stronger affinities with the work by Camurri and colleagues than with the work by Siegel, Winkler, or Coniglio. It focuses on the application of theoretical knowledge that may yield more sophisticated ways of interaction based on movement analysis, instead of trying to find new direct mapping strategies between gesture and sound. In the upcoming chapters, I shall discuss approaches taken in the software development and implementation, its use in the creation of a piece, the contributions of this study in the field of interactive dance performance, and guidelines for further study in this area of research.

CHAPTER V

SOFTWARE IMPLEMENTATION I: HYPOTHESIS, CONCEPTUAL AND TECHNICAL CONSIDERATIONS

Introduction

In the last pages of the previous chapter, apropos the discussion of other authors' views about ways of interacting movement with music, I started expressing my own views about possible ways that movement and music may interact in interactive computer systems. In order to establish more refined ways of interaction between dance and music in these systems, I suggested an approach that took the idiosyncrasies of dance into account. This meant that the interaction modes should rely on some type of movement analysis, and that the movement analysis should be done non-invasively.

This study puts these two ideas in perspective. The approach I will take is based on movement-analysis techniques that may yield elements for musical communication in interactive dance performance. This analysis is done non-invasively through the use of a fixed video camera, utilized as an outside-in tracking device. The practical outcome of the study is a software library that can extract musical periodicities from dance movement in real-time. The main idea thoroughly hypothesized in this study, is that the movement in dance possesses musical elements at the temporal level and these elements can be utilized for the real-time interaction between dance and music. The extraction of musical

elements, such as musical periodicities, from a dance is what I call the *musical processing of a dance*. This concept will be further elaborated later in the chapter.

One of the issues to explore is how these elements can be used in computer-mediated collaborations between a composer and a dancer/choreographer. By capturing and analyzing these elements in dance movement, I am expecting to open a channel of musical communication in computer-mediated collaborations between dancers and musicians. This channel of communication should allow elaborate and subtle ways of interaction that go beyond the application of the hyperinstrument paradigm in interactive dance performance. In many aspects, the approach taken in this study resonates with that of Camurri et al. in the field of interactive dance. In a broader sense, it is also similar to Rowe's work (1993, 2001) in interactive music. In order to promote novel and elaborate ways of interaction, I am creating software that borrows relevant theoretical knowledge from other fields. Applying this knowledge should enhance the computer-mediated communication between artists from the fields of music and dance.

This chapter is the prelude to the chapter in which I present a detailed description of the software library that was developed as a result of this investigation. I will initially formulate the main hypothesis of this study, and then, after explaining the delimitations within which the hypothesis is formulated, I will explain the conceptual and technical considerations that assisted in the creation of the software.

The conceptual considerations include the proposal of a framework for

interaction in computer-mediated collaborations between composers/musicians and choreographers/dancers, the concept of musically processing a dance, and a conceptual framework for software design. The technical considerations include certain properties of the frame-differencing video analysis signal, its representation in the time-domain, and an explanation of how a video camera — whose digitized stream is processed by a frame-differencing algorithm — can be used as a sampler for the detection of musical periodicities in dance movement.

Hypothesis

The theoretical knowledge acquired in Chapters 2 and 3 helped establishing important relationships between (a) the perception and production of musical rhythm and physical actions of the human body, and (b) the way these physical actions are also manifested in the production of rhythm in dance. The review of the video analysis software in the previous chapter was the final step in order to ground the main assumption of this study: *Dance movement possesses musical qualities at the temporal level. These qualities can be extrapolated in real time and provide a way for interaction between dance and music utilizing interactive computer systems.*

The musical periodicities in dance movement are extrapolated in real time by capturing the dancer's movement non-invasively with a video camera connected to a computer. The computer then performs certain types of analysis on the digitized video stream that enables the creation of a musical channel of interaction between musicians and dancers. The next two sections will explain the

conditions under which such extrapolation is possible.

Frame Differencing

A camera does not see objects in space nor does it distinguish between bodies or other objects. “Cameras only see light, not shape” (Lovell, 2002, p. 198). In order to make a computer *see*, some type of processing needs to be done to the incoming digitized stream. According to Robb Lovell (2002), several processing techniques can extract particular kinds of low-level information from a video stream. These techniques include the detection of presence, motion, and objects based on brightness or color information extracted from the stream.

Frame (or image) differencing is a motion segmentation technique (Tekalp, 1995, Chapter 11) that subtracts the brightness value of the pixels between two frames. A common application for frame differencing in video analysis is the detection of the *quantity of motion* (the amount of light change) that occurred between two frames. Frame differencing is available in all the programs that were reviewed for this study. It is the basis of the first software version of VNS, now incorporated in the **v.motion** object on SoftVNS 2. The *diff* analysis mode in **Cyclops** also works in the same fashion. EyesWeb also contains objects that enable the implementation of frame differencing. The subsection Properties of the Frame Differencing Analysis Signal and its Representation in the Time Domain below will demonstrate frame differencing and explain how this analysis signal can be used for the detection of musical periodicities in dance.

Delimitations

The musical periodicities in dance movement are extrapolated in real-time by applying certain digital signal processing (DSP) techniques to the frame-differencing analysis signal of the video stream. For the purposes of this study, and in order to keep the number of computations performed to a minimum, this type of processing is always applied to a greyscale video stream with a frame size of 160 by 120 pixels at a rate of 30 or 25 frames per second (fps). The techniques utilized to determine such periodicities work under the following conditions:

1- The camera should be placed in a fixed position outside the area the dancer is moving and the background should be static;

2- There should be a good contrast between the dancer's body and the background against which the dancer is moving. For example, a dancer wearing a black suit against a light background;

3- The camera should be capturing a single dancer at the time.

These three conditions are essential for the system to work properly. Since frame differencing computes the difference in brightness of the pixels that constitute the image, these differences can correspond to the dancer's movement if nothing else changes— i.e. the background against which the dancer is moving should not change, the camera should not change position while tracking the dancer, and abrupt light changes should be avoided. Otherwise, noise will be introduced into the system thereby interfering with the proper detection of the periodicities.

It is also important to have a good contrast between the background and the dancer. This way one will have a better range in the difference values. If a

dancer is wearing a color that does not have a good contrast against the background in the greyscale image, smaller values in brightness difference[45] will be detected and the dancer may even become *invisible* to the computer.[46]

Finally, this type of processing requires that only one dancer is tracked at a time. Frame differencing does not detect by itself the number of dancers that are present in the image. For this to happen other types of processing should be employed[47] and this could result in slowing down the computation. I shall now proceed towards describing the conceptual and technical considerations that assisted in the creation of the software.

Conceptual Considerations

A Possible Framework for Computer-Mediated Interaction in Dance

The framework for interaction proposed here intends to promote certain ways of computer-mediated interaction in real time between a dancer/choreographer and a composer/musician. It is designed to be operated by specialists from the fields of dance and computer music. The main goal is to allow the composer of electronic music certain types of control over the temporal articulation of the electronic music score during performance, otherwise impossible to achieve with pre-recorded music. This control ranges from tempo

---

[45] In greyscale streams, brightness is equivalent to the greyscale value of a pixel.

[46] If, for example, the dancer is wearing the same color as the background, when the dancer is moving very little difference between the brightness of the dancer and the background is produced and therefore it will not be detected by the computer.

[47] Eyesweb has objects that can determine the Kinesphere of a dancer, and one should be able to apply the frame-differencing algorithm to the Kinesphere of a dancer instead of the whole frame. However, this was never tested for the purposes of this study.

adaptation of the score to the dancer's movement — thereby enabling the dancer to slightly accelerate or slow down the music being played — to the generation of musical rhythmic structures extracted from certain types of rhythms in the dancer's movement.

A camera connected to the computer observes the dancer and the computer produces movement analysis data that is interpreted musically at the temporal level. The musician/composer who operates the computer receives a temporal-level interpretation of the movement analysis data in real-time. It is up to the musician to decide what type of controls/parameters for interaction are given to the dancer over the musical content.

The computer thus acts as a mediator in the process and is responsible for providing the musical elements extracted from movement that can be used for interaction. It acts as a sort of a filter that can extrapolate musical elements from a dance, thereby opening a channel for musical communication/interaction between dancers and electronically-generated music in real time. This type of interaction framework thus calls for a more intense participation of the composer/musician in the performance of dance with electronic music. It should also open interesting possibilities for improvisation or for the performance of open-form music/dance structures. The musician operating the computer can always change the parameters for interaction during performance and feed back different musical content to the dancer, or map the musical elements extracted from the dance to different parameters in the electronic score.

Certain aspects of this approach are in many ways similar to the one

described by Camurri et al. (2000) and Camurri and Trocca (2001) for the performance with emotional agents discussed in the previous chapter. The approach taken in this study is however more focused. The software I am developing tries to find and interpret musical cues in dance movement and use them as a means to enable the musical interaction between a dancer and a musician. Unlike Camurri et al., the approach proposed here does not rely on previously feeding the system with knowledge, and then let the system work by itself during performance. Although some elements can be implemented previously (such as the expected tempo of a musical sequence) this system is designed to be operated in real-time.

The Musical Processing of a Dance

The idea of musically processing a dance outgrew from the theoretical study carried out for this dissertation. As noted by Hodgins (1992) music and dance share at the temporal level fundamental characteristics of structure and rhythm so intimately, that is difficult to differentiate each discipline's method of realizing such elements. This made me realize that it is perhaps possible to analyze certain aspects of dance from a purely musical perspective. By doing so, one would be opening a strictly musical channel of communication between dance and music in interactive dance performance. If a computer can engage in the task of doing a musical analysis and processing of dance movement, it can provide musical elements for interaction to the musician operating the system. The musician would thus be communicating with the dancer in strictly musical

terms, in the same fashion she would be communicating with another musician.

In chapters 2 and 3, I put in evidence strong relationships between bodily movement and musical rhythm, and dance and music at the temporal level respectively. In Chapter 3, I established three categories of rhythms that may be identified in dance: (a) rhythms produced by the motor system, (b) rhythms that establish longer-term relationships that shape the phrases in movement, and (c) rhythms associated with the expressive qualities of a movement sequence. From these three categories, motor rhythms encounter strong parallels in music. As noted in chapter 3, motor rhythms have an accentual and articulatory character that serve to establish the sense of beat in dance. They thus seem to be the ones that can produce a series of stresses and accents that can convey the sense of musical pulse in dance.

It is thus possible, or at least conceivable, to look for these musical elements at the temporal domain in dance movement.[48] The software developed for this study tries to identify musical elements at the rhythmic level in dance — rhythms present in dance movement that bear the qualities of musical rhythms. These musical rhythms, or *musical cues*, are extracted from a movement sequence by performing certain types of frequency-domain analysis on a representation of

---

[48] One could perhaps determine how *musical* is a dance by analyzing how these musical rhythms (and other musical elements) are apparent in the movement. Seen from this perspective, different dances can be said to have different degrees of musicality, and the degree of *musicality* could be a characteristic for classifying certain types of dances. Finding a heuristic in order to classify the degree of musicality of a dance is completely outside the scope of this study. The term *musical* is used here in a broad sense. It refers to an intuitive way of classifying rhythmic elements in dance that can be interpreted as musical rhythms. This degree of musicality as a parameter for classification hinted here would be akin to — for example— trying to classify the degree of musicality in languages by analyzing the amount pitched phonemes whose inflection resembles a melody.

the movement in the digital domain. The types of analysis performed on the digital representation of the movement will be discussed in detail in the upcoming chapter.

The concept, or metaphorical idea, of musically processing a dance is key to the present study. By doing this, I am hoping to foster a level of musical communication between dancers and musicians that relies on a common denominator, instead of relying on interaction metaphors that are more typical of music performance, such as the hyperinstrument approach.

Conceptual Framework for Software Implementation

The software developed for this study consists of a specialized modular library that can extract existing musical periodicities in dance movement as captured from a video camera. This modular library is designed to work with modular visual programming environments, such as Max or EyesWeb, that already possess video analysis libraries. The software that was programmed consists of a set of Max external objects that process video analysis data performed by other objects/libraries available to the Max programming environment such as **Cyclops** or softVNS2. The algorithms that were implemented should be easily ported to other visual programming environments containing libraries of image analysis such as EyesWeb or Isadora.

These external objects provide several facilities for extracting temporal domain data from the analysis of a video stream. These facilities include a frequency domain representation of the signal and musical tempo analysis and

control. The modular architecture of the software is intended to allow its users to combine the modules in several different ways in order to map the data creatively in ways that were not initially predicted.

Schematically, the software that was developed works as shown in the figure below.

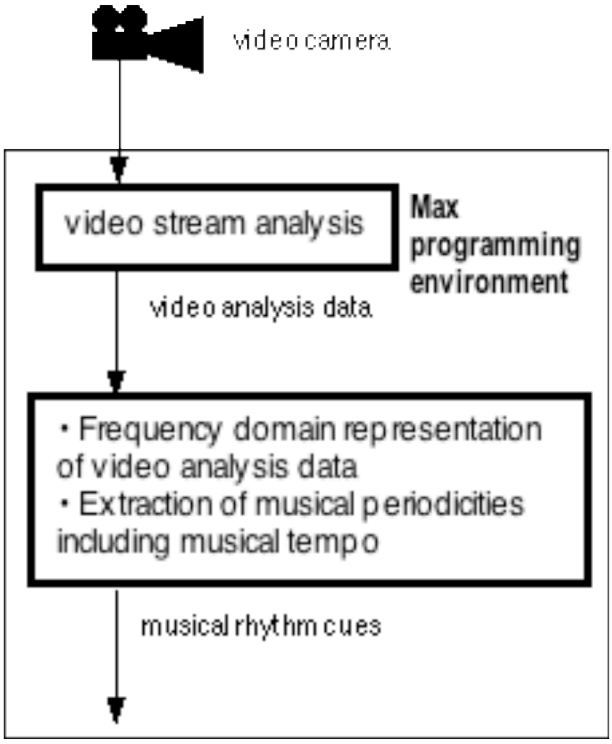


Figure 3. Schematic representation of the software.

A fixed video camera grabs the movement data at 25 or 30 frames per second.[49] The video signal is digitized and analyzed by one of Max's existing libraries for video analysis. The analysis of the digitized stream is represented numerically in the time domain as a time-varying signal. Subsequently, that time-varying data is given a representation in the frequency domain and the periodicities that have

[49] The frame rate of a video camera depends on the broadcasting system being used. In the PAL system the frame rate is 25 fps. The NTSC system uses 30 fps.

musical relevance are extracted. The user of the library can then perform mappings of these data directly, enabling the dancer to generate musical rhythmic structures from movement in real-time. The frequency-domain representation of the data can also be used to compute the underlying beat present in the stream. The value of the beat then feeds an adaptive clock that enables the dancer the control of musical tempo in real time.

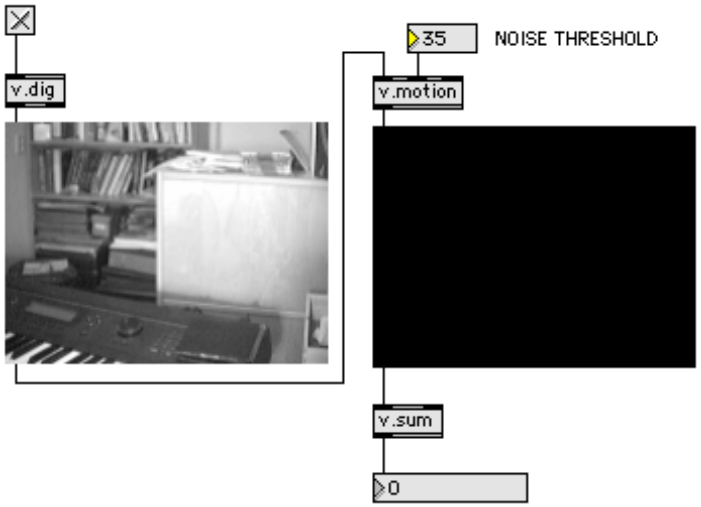
Technical Considerations

Properties of the Frame Differencing Analysis Signal and Its Representation in the Time Domain

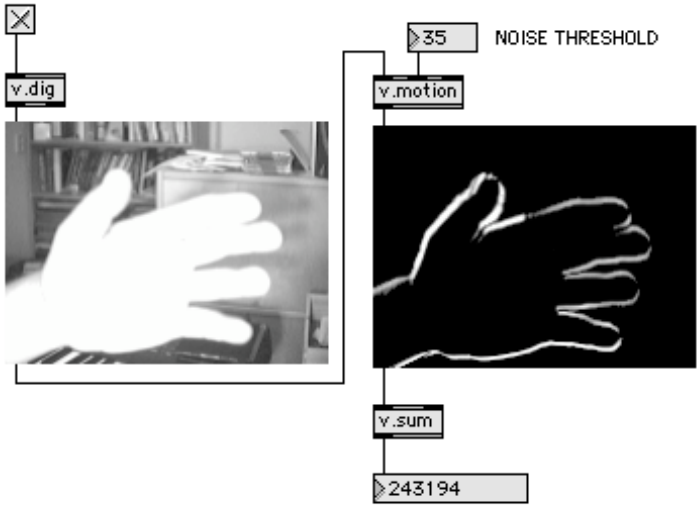
If a camera is looking at a controlled-lighting environment in a fixed position and the background is static, we can detect if motion has occurred at a certain instant in time by calculating the difference in brightness of each pixel of the image between consecutive frames. In the areas of the image in which no motion has occurred, the difference is zero, since the pixels of that area carried the same brightness onto the next frame. In the areas in which motion has occurred the absolute value of the difference will be greater than zero.[50] Summing all the brightness difference values of each pixel in two consecutive frames corresponds to the overall change that occurred in the pixels that changed their value. This is often equated to the *amount of motion* that occurred in between those two frames. The following softVNS patch illustrates the process (see Figure 4).

---

[50] This is true after some denoising is done. Digital cameras usually introduce some noise in the digitized stream. This can be easily overcome by *thresholding*, i.e., by introducing a threshold value in which the calculated brightness differences are forced to zero when they are below that value (Tekalp, 1995). In SoftVNS 2 the threshold value is the introduced in the middle inlet of **v.motion** as it can be seen in Figures 4a) and b).



a)



b)

Figure 4 a) Frame differencing applied to a static background; b) Frame differencing applied to a moving hand over the same static background.

In figures 4 a) and b) we can see the frame differencing algorithm applied to a complex static background, and to a hand moving over the background respectively. In figure 4a), the background remains static and the output from

v.sum is zero. Object **v.sum** sums the value in all of the calculated brightness differences. The **v.screen** below the **v.motion** object is completely black since there are no changes in the brightness of the pixels between consecutive frames. In figure 4b), we see a hand moving over the same background. Because there is movement, there are a number of pixels that are going to change their brightness every time the frame differencing is computed. The **v.screen** below the **v.motion** object now depicts *all* the pixels that changed their brightness level and the sum of all brightness levels is greater than zero. As it can also be observed, when we visualize the results of an image differencing operation, we can detect the edges of the moving object in the image.

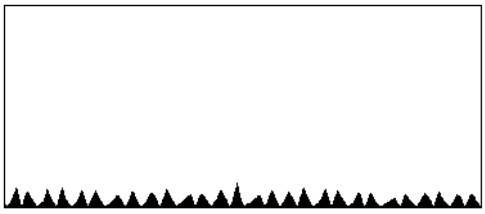
Another important feature about the frame differencing signal is that the computation of the sum of the absolute brightness differences in a digitized video stream does not give any information about *where* the motion occurred. The information obtained with this type of computation gives information about *when* the motion occurred and its amount, when one analyses this signal in the temporal domain. Furthermore, this technique can capture very well movement inflexion, since they in general translate into a tendency to zero in the sum of all brightness differences. This will be further explored in the next section, where the time-domain representation of this signal will be done and analyzed.

Signal Representation in the Time Domain

When we plot over time the instantaneous values given by the sum of the absolute value of the differences in brightness of all the pixels in a stream, some

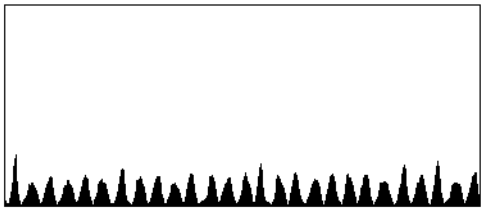
interesting conclusions can be drawn. [51] Since the background does not change, all the changes detected through frame differencing will correspond proportionally to the amount of movement performed by the moving body. The more the body moves, the greater will be the number of pixels that changed their brightness between consecutive frames. When there is movement inflexion (e.g. a change of direction) the value of the sum in the brightness difference tends to zero since there was a sudden stop in the movement for the inflexion to occur (see Example 1 in the DVD-ROM in folder 'Examples').

If we compute the number of pixels that changed their brightness in periodic movement actions over time, we can detect periodicities in the video analysis signal that are in direct correspondence to the actions performed. This works very well for simple periodic actions such as jumping or waving a hand, for example. Moreover, wider actions correspond to increasing the amplitude of the frame differencing signal and faster actions correspond to increasing the frequency (see Figure 5).

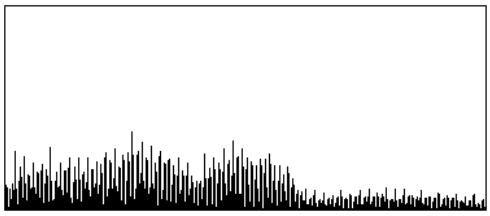


a)

[51] This applies to the conditions explained in the Delimitations section of this chapter.



b)



c)

Figure 5. Several representations in the time domain of frame-differencing graphs of movement of waving hand. Figures 5a) and b), same frequency, with two different amplitudes; 5c), faster frequency with amplitude variation.

If we remove the DC offset from the signal, the similarities to periodic acoustic signals are simply striking (see Figure 6). This means that if we apply an algorithm that detects the periodicities of this signal — such as the Fast-Fourier Transform (FFT) — that works for frequencies below the audible range, we can detect the rhythms present in the signal. Moreover, if we apply an algorithm for pitch detection that works in the same frequency range we can detect the fundamental frequency (tempo) of a periodic movement action.

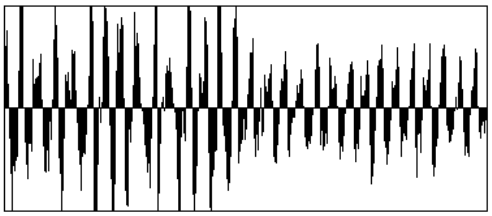


Figure 6. DC offset removal of the quantity of motion variation in a video caption of a waving hand.

Although this comparison seems promising, one still needs to address if the computation of the sum of the brightness differences could be used as a method for determining musical periodicities in dance. For this method to be implemented, two questions need to be answered. The first is if this method can be used for the detection of accents in dance movement that can evoke musical rhythms. The second is if these periodicities can be properly reconstructed in the time-domain so that we can perform the conversion of this signal into the frequency domain.

The first question has been partially answered in the above description of the properties of the frame-differencing signal. This signal detects movement inflexions with a good degree of accuracy. In chapter 3, it was established that the accentual and articulatory nature of motor rhythms in dance conveyed the sense of pulse, or beat in Humphrey's terms. Pulse is an inherent quality of musical rhythm (cf. Chapter 2). However, pulse sensation is an acoustic phenomenon (cf. Parncutt 1994a). The sense of pulse in dance is established somewhat differently. It is established by changes in spatial articulation that occur regularly, at a rate that can evoke the sense of pulsed movement. For pulsed dance movement

sequences that are performed with music, and thus evoke the sense of musical pulse and rhythm, these rates should be confined to the durations of musical beats — 300 to 1500 msecs.[52]

The frame-differencing signal detects well the spatial inflexions and accents in movement, and the rate of these articulations/inflexions is visible when we plot the output of this signal in the time-domain (see figure 5). The variation in the frequency is independent from the variation of the amplitude in the signal as seen in figure 5 c). This is an important factor to retain, since one might initially suspect that the dancer's proximity to the camera could be a factor that interfered in the proper representation of the frequencies in the signal.

This is actually not true.[53] The dancer's proximity to the camera will affect the amplitude of the signal. When one move closer to the camera, more pixels will change their brightness value since the area occupied by the body in the image is bigger; however, the rate of change in the movement remains unchanged. In order to better understand this phenomenon, one can establish a comparison to the situation of capturing periodic acoustic sounds with a microphone. When one captures a periodic sound with a microphone, the recorded pitch does not change if we place the source closer or farther away from the microphone. Instead, it is the amplitude of the recorded pitch that changes.

---

[52] These values for musical tempi are the ones that commonly appear in metronomes (40 - 200 beats per minute), and are good boundary values for musical tempo. As Rowe (2001) convincingly argues, "[m]etronome boundaries, are not arbitrary; they emerged from many years of performance practice during which tempi beyond these limits were not normally needed" (p.125).

[53] This is valid when the dancer does not move too close, to the point in which the whole body is not seen any longer to the camera, or too far, to a point in which the dancer's body is not well depicted by the frame size being used.

The second question, pertaining the fidelity in the reconstruction of the signal in the time domain, will be addressed in the upcoming section.

Utilizing Video Cameras as Low-Frequency Samplers for the Detection of Rhythm

As shown above, the time-domain representation of the frame-differencing analysis signal is in many ways similar to the time-domain representation of acoustic signals. This made me wonder if this type of processing of video signals could be somehow equivalent to the time-domain representation of acoustic signals in the digital domain. Empirically, it seems so.

The conversion of an acoustic signal to the digital domain is a process that could be generically exemplified as follows: a microphone transduces air-pressure variations into electrical voltages, and those voltages — which are an *analog* representation of air-pressure variations — are subsequently digitized by an analog-to-digital converter (ADC). The ADC converts the voltages into a string of binary numbers (see for example Roads, 1996, Chapter 1). The time-domain representation of a sound in the digital domain depicts the variation of the amplitude overtime. This is a proportional and discrete representation of the air-pressure variations captured by the microphone. For each sample in the digitized signal, we get an instantaneous value of the amplitude of that signal that corresponds to the same instantaneous air-pressure variation captured by the microphone.

The computation of brightness differences in frame differencing gives the instantaneous value of a variation too. This variation corresponds to the variation in the amount of motion. As one has seen above, if we have a fixed video camera capturing an object moving against a static background, the range of this measurement is proportional to the amplitude of the movement and to its frequency — the broader the movement the wider the variation, the faster the movement the faster the variation.

This has a resemblance to the case explained above in the analog-to-digital conversion of acoustic sounds captured by a microphone. In both situations, we get a discrete representation of a variation over time. In the case of the digitized acoustic signals captured by a microphone, each sample represents an instantaneous value of the air-pressure variation caused by the signal. In the case of the frame differencing of a digitized video stream, each sample represents an instantaneous value of the amount of movement variation.

The frame differencing video analysis signal can thus be represented as a signal of time-varying amplitude that bears properties akin to digitized acoustic signals, namely in its wave-like shape and periodicity, as seen in figure 6. This provides a good case for applying the same techniques utilized to find periodicities in digitized acoustic signals, such as FFT, to this type of signal. The utilization of techniques that can be employed for the analysis of musical signals in the analysis of digital signals carrying information about human motion, concretely dance, also helps reinforce the concept of musically processing a dance. Such techniques, although employed in the analysis of other phenomena,

encounter a wide range of applications in sound analysis and synthesis, and musical composition.

By applying this type of processing to movement analysis data, one is employing video cameras as low-frequency samplers for the detection of rhythm. One of the practical aspects to discuss is if video cameras can actually sample human movement with enough resolution to properly reconstruct the musical periodicities in the time-domain representation of the frame-differencing signal. This is actually possible.

According to the sampling theorem (Nyquist, 1928), " [I]n order to be able to reconstruct a signal, the sampling frequency must be at least twice the frequency of the signal being sampled" (Nyquist, 1928, quoted in Roads, 1996). A video camera samples images at a rate of 25 or 30 fps. This is equivalent to saying that video cameras sample images at a rate of 25 or 30 Hz. This means we can properly reconstruct periodicities of up to 12.5 or 15 Hz if we use a video camera as a low-frequency sampler for the detection of periodicities in movement.

Musical periodicities range between 200 and 1800 ms (cf. Chapter 2, Movement as a Consequence of Rhythm) or between 5 and 0.55 Hz. The movement articulations that can evoke such periodicities are in the same range, which is far below the highest frequency one could properly represent by utilizing a video camera as a sampler.

Conclusion

In this chapter, I initially presented the main hypothesis that was formulated as an outcome of the literature review, video analysis software review, and approaches taken by other creators of interactive dance, from chapters 2, 3 and 4. I also exposed the conceptual and technical considerations that assisted in the creation of software that was developed. In the technical considerations part of this chapter, I explained certain interesting properties of the frame-differencing signal for movement analysis, and discussed how this signal can correctly reconstruct existing musical periodicities from dance movement. In the conceptual considerations part, I put forth some ideas that may be interesting to consider as new ways for establishing interaction modes between dance and music utilizing interactive computer systems. These ideas include the proposed framework for computer-mediated interaction and the concept of musically processing a dance.

In the proposed framework for computer-mediated interaction, I presented a framework that calls for an active participation of the composer/musician during the performance of a work of dance with computer music. This idea is not new. Rowe (2001) also advocates that interactive music systems can be a vehicle for a return of the composer into active music-making during the performance of a piece. One of the main motivations that underlies the work developed for this dissertation was a personal need for carrying out what I called an interactive process of collaboration up to the performance stage, as mentioned in the introductory chapter to this dissertation. In the views that were gathered from

other creators of interactive dance in Chapter 4, the active participation of the composer in the process of performance did not seem to be a major issue or motivation for the use of computer interactive systems in dance. However, it would not be surprising if these composers also engage actively in the performance of their works of interactive dance as a natural outcome in the use of interactive computer systems.

The concept of musically processing a dance is an idea that gradually emerged from the analysis of the literature in the perception and production of musical rhythm, and the interaction between dance and music in the temporal domain. This idea is grounded on the fact that there are similar mechanisms for the production of rhythm in both dance and music, and these mechanisms can be extrapolated in real-time, by observing the dance *musically,* without interfering in the dancer's freedom of movement. This way, a level of musical communication can be established between dancers/choreographers and composers/musicians without forcing a dancer/choreographer to behave, or think, musically. The computer creates a musical interpretation of the dancer's movement and provides the musician/operator of the system with the musical elements for interaction.

This idea seems to provide a fertile ground for development in the field of computer-mediated collaborations between dancers and musicians. If new research findings are produced that, for example, establish correlations between certain types of expressive movement in dance and expressive qualities in music, this knowledge could be implemented algorithmically in order to promote higher levels of musical communication between dance and music utilizing interactive

computer systems. This ultimately may lead to new ways of thinking the music/dance interaction altogether.

In the next chapter, I will do a brief description of the objects that comprise the library and discuss in detail the techniques utilized for the extraction of musical periodicities in real-time from the time-domain representation of the frame differencing analysis signal.

CHAPTER VI

SOFTWARE IMPLEMENTATION II: BRIEF LIBRARY DESCRIPTION, DETERMINING THE SPECTRUM OF A DANCE, AND DANCE MUSICAL TEMPO TRACKING AND CONTROL IN REAL TIME

Brief Library Description

The m-objects are the set of Max external objects comprising the library I developed as an outcome of this study. These objects perform certain types of temporal analysis and processing on data coming from libraries that do real-time video analysis available to this programming environment. This library is intended to add extra functionalities for the work in interactive dance utilizing Max/MSP. As it was already emphasized, these algorithms should be easily ported to similar programming environments that possess video analysis libraries, such as EyesWeb or Isadora.

The library contains six objects: **m.bandit**, **m.clock**, **m.weights**, **m.peak**, **m.sample**, and **m.lp**. These objects can be combined in several ways to perform the temporal domain analysis of dance movement, and can be grouped into objects that do analysis (**m.bandit**, **m.peak**, **m.weights**), processing (**m.clock** and **m.lp**), and extras (**m.sample**). **m.bandit** and **m.clock** are the core of the library since they are responsible for doing the frequency domain representation of the frame-differencing signal and musical tempo control respectively.

Five from the six objects[54] comprising the library were coded in the C programming language utilizing Max/MSP's own Software Developers Kit (SDK). The main reference used to code these Max objects for Macintosh using Mac Os 9 was *Writing Max/MSP Externals* (Zicarelli, 2001). Other useful texts were consulted for the development and debugging of the code, such as Paul Berg's own tutorial introduction to writing Max externals, *Writing Max Objects with CodeWarrior: A Simple Introduction* (Berg, 2001), and Kirk Woolford's text on debugging Max externals with Code Warrior (Woolford, 2001). The original objects were coded in the CodeWarrior software development environment, version 7. The latest versions of the objects that run on Macintosh using Os X were coded in CodeWarrior version 8 following the guidelines of *Writing External Objects for Max and MSP 4.3* (Zicarelli, Clayton, & Sussman, 2003). The latest version of the code that can be found in Appendix C, and the CodeWarrior projects containing a compilable version of the code can be found in the accompanying DVD-ROM.

I will now present a summary description of each object's function. This summary presentation of the m-objects is intended to supply the reader with enough information to better follow the tutorial presentation I do in the next chapter. A detailed description of each object in regards to the messages and arguments it accepts, and the produced output, is presented in Appendix B and follows the style of the *Max Reference* (Zicarelli et al., 2001), the reference manual for Max version 4. This description is also done at this point in the text in

---

[54] **m.lp** is a Max subpatch.

case the reader wants to skip the two upcoming sections that present the mathematical formulae utilized in the core objects, and a detailed technical description of their functionalities. A detailed description of the techniques utilized by **m.bandit** and **m.clock** for the spectral representation of the frame-differencing signal and for the control of musical tempo will be done in the two upcoming sections, respectively entitled The Spectrum of a Dance: Representing the Frame Differencing Signal in the Frequency Domain and Controlling Musical Tempo from Dance Movement.

Analysis Objects

m.bandit

m.bandit is one of the core objects of the library. This object takes as input the time-varying representation of the frame-differencing signal and: (a) estimates and outputs the fundamental frequency of that signal, (b) outputs the amplitude and frequency of the initial four harmonics of that frequency, and (c) outputs lists for graphically displaying the signal spectrum and the spectrum of the four harmonics of the fundamental frequency.

m.bandit also allows the user to specify the object's sensitivity to movement. This is done by sending two messages to the object, 'idle_thresh' and 'idle_int,' which respectively specify the threshold value in the brightness difference below which the object considers that there is no movement and the minimum time interval after being below that value in which the object should consider that there is no movement. The object prints the word 'moving' in the

Max window when it considers there is movement (referred to in the text as ‘moving state’), and outputs 1 from the rightmost outlet, and prints the word ‘idle’ (referred to in the text as ‘idle state’) and outputs zero from the rightmost outlet otherwise. This is independent from the quantity of movement measurements provided by the frame-differencing algorithm. This enables the user to program the object to send control information to other objects only when the measurements of the quantity of movement pass a certain value. One useful and practical application of this feature is, for example, to allow the dancer to move in the backstage without triggering any events while preparing to start a piece. Other *artistic* applications of this feature are explained in the description of the utilization of **m.bandit** in *Etude for Unstable Time*.

m.peak

m.peak takes as input the time-varying representation of the frame-differencing signal and outputs a ‘bang’ message (order of execution) when a significant peak in the signal has occurred. This object is especially useful to trigger events (sounds, or messages to other objects, for example) when heavily accented movement actions occur.

m.weights

m.weights takes as input the fundamental frequency of the frame-differencing signal as calculated by **m.bandit** converted to milliseconds, rounds that value to the hundredth millisecond, and stores it temporarily. This object

outputs the hundredth millisecond value that was output more times by **m.bandit** in the past 60 frames.[55] One of the functions of **m.weights** is to provide a "short-term memory" to **m.clock** that can be used to find the tempo of a movement sequence without a previous estimate (see Chapter 7, Establishing the Musical Tempo from Dance Movement without an Initial Estimate). **m.weights** is also a good object to use during performance as a means of permanently monitoring the output of **m.bandit** and suggest tempi values to **m.clock.**

Processing Objects

m.clock

m.clock the other core object of this library. It is an adaptive clock that enables a dancer to control the musical tempo of a musical sequence. This object treats the fundamental frequency output of **m.bandit** (converted to milliseconds) as a beat candidate and allows to perform the tempo adaptation of a musical sequence if the beat candidate falls within a pre-defined margin for adaptation from a previous tempo estimate.

m.lp

m.lp is a simple low-pass filter that can be used to smooth brightness values computed by the frame-differencing algorithm.

---

[55] As an example of how this works lets suppose that a value of 468 ms arrives to the input of **m.weights**. That value is rounded to 400 and stored in **m.weight**'s 400 ms bin (**m.weights** has bins ranging from 300 to 1500 ms). After receiving and rounding the value, the object calculates which bin got more "hits" in the past 60 frames. If, for example, the values of the 700 ms bin were more prominent, the output of the object will be 700.

Extras

m.sample

m.sample acts as a sampler for the brightness difference values computed by the frame-differencing algorithm. This object was built after I realized that the digitized streams grabbed by USB cameras are not very steady. They can oscillate between 23 and 34 fps (Firewire cameras tend to be more stable, oscillating between 29 and 32 fps). Since the computations performed by **m.bandit** are dependent on the sampling rate, it is crucial to have a steady sampling of the values that will be computed by this object**.** This object thus samples the output of the brightness difference values using the processor's clock at the desired sampling rate, yielding a steady flow of values to be computed by **m.bandit**.

The Spectrum of a Dance: Representing the Frame-Differencing Signal in the Frequency Domain

As shown in the previous chapter, all the periodicities in movement whose frequency is akin to musical periodicities can be reconstructed from the time-domain representation of the frame-differencing signal. It was also stated that one could find the underlying beat of a movement sequence by calculating the fundamental frequency of that signal.

In order to get a spectral representation of the signal, **m.bandit** employs a bank of 150 second-order recursive[56] band-pass filters.[57] The center frequency of

[56] Recursive filters are also known as IIR (infinite impulse response) filters.

these filters spans from 0.5 Hz to one-half of the sampling/frame rate[58] is distributed logarithmically,[59] and their bandwidth is proportional to the center frequency (10% of the center frequency).[60] The Goertzel algorithm (Goertzel, 1958; see also Openheim & Schafer, 1975, Chapter 6, pp. 287-289) is applied to the output of the band-pass filters in order to get a magnitude and phase representation of each center frequency.[61]

The main reason for applying a bank of band-pass filters using the Goertzel algorithm, as opposed to FFT/DFT, in order to obtain the frequency-domain representation of the signal has to do with the speed of computation at runtime. The Goertzel algorithm can compute the Discrete-Fourier Transform (DFT) of a single frequency component (Blahut, 1985; Oppenheim & Schafer, 1975) faster than FFT (Banks, 2002; Sirin, 2004). The Goertzel algorithm thus allows the calculation of the magnitude and phase spectrum of a selected frequency band, and when using a bank of band-pass filters one can select the frequency range at which the object is operating, also sparing calculations of other

---

[57] I thank Peter Pabon from the Institute of Sonology, The Hague in The Netherlands, for suggesting this approach for obtaining the frequency domain representation of the signal, as well as his helpful comments and suggestions during its implementation.

[58] The sampling rate is the frame rate at which the movement sequences are grabbed.

[59] This is an important feature to consider for the detection of pulse in dance movement. **m.bandit**, the Max external object that performs the frequency-domain representation of the time-domain representation of the frame differencing signal, outputs a visual representation of the amplitude spectrum of the signal. Having a logarithmic distribution of the frequencies is very useful in order to visualize the relationships between the frequencies that comprise the signal. The prominent harmonics of a fundamental frequency will keep the same relative distance between them in the visual representation. Harmonic relationships between the frequencies comprising the signal are a fundamental aspect for the determination of pulse. This issue will be discussed in detail later in the chapter.

[60] The best value for the bandwidth was determined by trial-and-error, and seems to be the best value for this type of application.

[61] The Goertzel algorithm is a type of second-order recursive filter (see for example Oppenheim & Schafer, 1975).

unwanted frequency bands at runtime.[62]

The frame-differencing signal spectrum is calculated in two stages. The first stage consists in passing the time-varying representation of the frame-differencing signal through the bank of band-pass filters. The difference equation for the second-order band-pass filter is:

$$Y_{(n)} = X_{(n)} + 2CR \times Y_{(n-1)} - R^2 \times Y_{(n-2)} \qquad (1)$$

$$C = \cos(\frac{2\pi cf}{sr})$$

$$R = e^{(\frac{-\pi bw}{sr})}$$

cf = center frequency

bw = bandwidth

sr = sampling rate

e = 2.718281828459045... (Euler's number)

This equation can be found in Stan Tempelaars book *Signal Processing, Speech and Music* (1996) as an example of a formant filter for synthetic speech production.

The second stage in the calculation of the spectrum extracts the real and imaginary part of the output by applying a slight alteration of the Goertzel algorithm suggested by Peter Pabon (Personal communication, October 31, 2002). The real and imaginary part of each filter's output are calculated as follows:

$$Y_{real} = Y_{(n)} - R\cos(\frac{2\pi bw}{sr}) \times Y_{(n-1)} \qquad (2)$$

---

[62] A future implementation of **m.bandit** will allow the user to dynamically allocate to the object the number of filters to use, the frequency range and their bandwidth. This will allow to use this object to 'tap' selectively a certain frequency range, and to use several objects without sacrificing the speed of computation.

$$Y_{imag} = -R\sin(\frac{2\pi bw}{sr}) \times Y_{(n-1)} \quad (3)$$

The magnitude and phase components for each frequency band can then be easily calculated:

$$Y_{mag} = \sqrt{Y_{real}{}^2 + Y_{imag}{}^2} \quad (4)$$

$$Y_{phase} = \arctan(\frac{Y_{real}}{Y_{imag}}) \quad (5)$$

In the object **m.bandit**, each band-pass filter is represented as a C programming language structure, where all coefficients are stored upon the object's instantiation:

```
typedef struct filter      //filter sructure
{
       double y[3];        //output (y(n)), previous (y(n-1)),
                           // second previous (y(n-2))
       double f_cf;        // center frequency
       double f_bw;        // bandwidth
       double f_c;         //coefficient C
       double f_r;         //coefficient R
       double f_cr;        //coefficient CR
       double f_two_cr;    //coefficient 2CR
       double f_rsqrd;     //coefficent R^2
       double f_out;       //output
       double f_real;      //real part
       double f_imag;      //imaginary part
       double f_rsin;      //RsinPhi
} Filter;
```

Figure 7. C structure representing a digital band-pass filter and all of its pre-calculated coefficients in **m.bandit**.

The Max external object definition contains an array of 150 of such structures for spectrum calculation. The signal magnitude spectrum is output as an Atom data type— a data type used to output lists in Max — and can be visualized by Max's **MultiSlider** object.

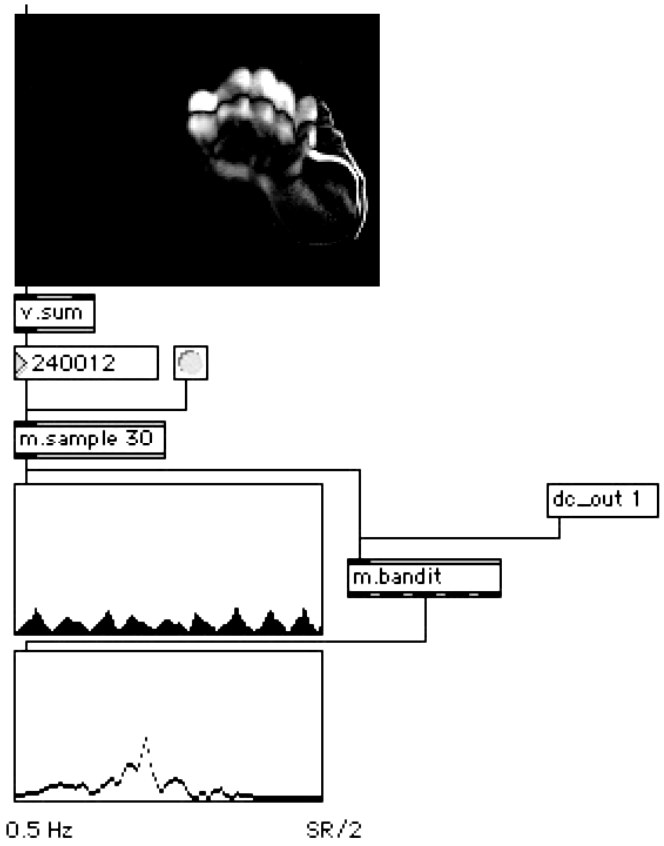


Figure 8. Magnitude spectrum output by **m.bandit**.

The Musical Tempo of a Dance: Determining the Fundamental Frequency of the Frame Differencing Signal

Beat Tracking as Pitch Tracking

Determining the pulse of a musical sequence is essentially determining the frequency and phase of a very low frequency (Scheirer, 1998; Rowe, 2001). The problem of beat tracking can be seen as a special case of pitch tracking in which phase detection plays a more predominant role. As noted by Eric Scheirer (1998) human pitch recognition is only sensitive to phase under certain conditions, whereas "rhythmic response is crucially a phased phenomenon—tapping on the

beat is not at all the same as tapping against the beat, or slightly ahead of or behind the beat, even if the frequency of tapping is accurate" (Scheirer, 1998, p. 590).

Certain existing computational algorithms of beat tracking attempt to find the tempo of a musical sequence by estimating the frequency and phase of the most prominent pulse in a musical sequence. These algorithms utilize adaptive oscillator models (see for example Large & Kolen, 1994; Toiviainen, 1998), banks of band-pass and comb filters (Scheirer, 1998), or adaptive filters (Cemgil, Kappen, Desain & Honing, 2000).

On the adaptive oscillator models approach (Large & Kolen, 1994; Toiviainen, 1998), a network of non-linear oscillators take as input a stream of onsets and continuously adapt the period and phase of an oscillator to match the tempo of a musical sequence. Scheirer's oscillator model (1998) is inspired on Large and Kolen's approach (1994) and uses a network of resonators (comb filters) to phase-lock with the beat of the signal and determine the frequency of the pulse. Scheirer's model operates in acoustic data rather than on event streams, and more pre- and post- processing is required in order to accurately extract the beat in a musical sequence. The adaptive filter model presented by Cemgil et al. (2000) formulates tempo tracking in a probabilistic framework in which a tempo tracker is modeled as a stochastic dynamical system. The tempo is modeled as a hidden state variable of the system and estimated by Kalman filtering.

The proposed approach for determining the underlying tempo of a dance sequence is largely inspired on the approaches mentioned above. This approach

— which has not been tested on musical data — seems to provide very good results for the work in interactive dance, in terms of determining the most prominent frequencies that can be considered the tempo of a movement sequence. The approach presented here should not therefore be considered as an alternative model for beat tracking in music.

In the previous section, it was shown how one could determine the frame-differencing signal spectrum by applying a modified band-pass filter bank to that time-varying signal. The object **m.bandit** can also output the most prominent instantaneous frequency of that spectrum. This is done by cross-correlating the calculated spectrum at a certain instant with the spectrum of a 1 Hz pulse train. Unlike the beat trackers presented above, **m.bandit** does not calculate the phase of the most prominent frequency. In fact, it ignores phase completely and that makes this algorithm very akin to other algorithms of pitch tracking. The reason for ignoring the phase spectrum of the signal will be explained later in the chapter.

Determining the Musical Tempo of a Dance

The musical tempo detection of a dance performed by the system implements the theoretical knowledge that was gathered in chapter 2 about pulse sensation. According to Parncutt (1994a) pulse sensation is an inherent quality of musical rhythm. It is an acoustic phenomenon produced by the interaction between the low-frequency periodicities present in a musical sequence. Parncutt equates the perception of pulse to the perception of pitch in complex tones. Pulse

sensation is enhanced by the existence of *parent* or *child* pulse sensations that are *consonant* with the most prevalent one.

If we want to determine the *musical* pulse in dance movement one should analyze if the periodicities in the motor articulations of dance movement evoke the sensation of regularity akin to musical pulse. That is, if these articulations bear the durational characteristics of musical rhythms and if they are in a certain proportion to each other in a way that pulse is reinforced. The approach utilized for determining the pulse in a movement sequence is the same that can be applied for finding the fundamental frequency of a pitched complex tone. It is based on a harmonicity criterion, i.e., that the pulse sensation is enhanced by periodicities that are integer multiples of a beat duration as proposed by Parncutt (1994a).[63] Hence, the cross-correlation with a 1 Hz pulse train — as opposed to a 1 Hz sine wave, for example — in order to find the frequency that can be a beat candidate. The spectrum of a 1Hz pulse train contains harmonic peaks, and the output of the spectral cross-correlation will thus favor signals that contain harmonic peaks.

---

[63] As seen in Chapter 2, Parncutt (1994a) actually says that pulse sensation is enhanced through the existence of parent or child pulse sensations. This means that pulse sensation should also be enhanced by the presence of periodicities that are in sub-harmonic proportion to the pulse. The calculation s performed by **m.bandit** only take into account the periodicities that are in harmonic proportion to a given pulse. This simplification does not seem to have an influence on the detection of the correct pulse.

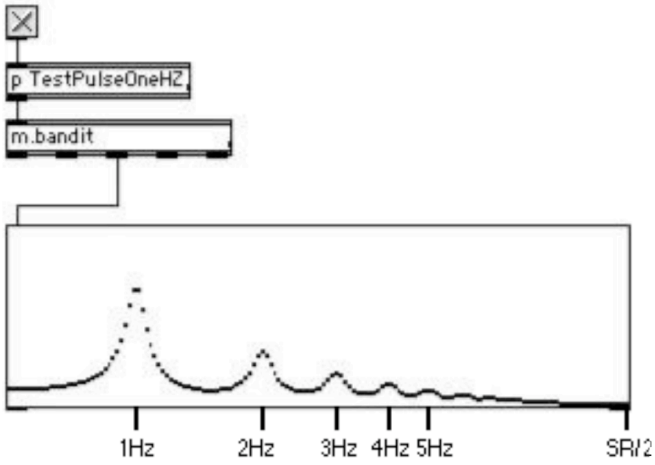


Figure 9. Spectrum of 1 Hz pulse train as calculated by **m.bandit**.

The object **m.bandit** embeds a file called testsig.h in which the amplitude spectrum of a 1 Hz pulse train is stored. This spectrum was collected from **m.bandit** itself. The array in which the pulse train spectrum is stored, contains a simplified version of the actual output: the amplitude value of the output for each frequency band was normalized to 1, and all the amplitude outputs for the frequency bands, except those that correspond to the harmonics of the one-hertz pulse, were forced to zero. The array only contains the amplitude values of the frequencies between 1 and 4 Hz (63 points), the first four harmonics of the one-hertz pulse train.

```
#define TEST 63
float testsig [TEST]={0.95362, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.52969, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.38007, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0.30061};
```

Figure 10. File testsig.h.

The cross-correlation between the instantaneous spectrum of the signal and the spectrum of the 1 Hz pulse train is done the following way: starting at the lowest frequency band (0.5 Hz), the first 63 points of the spectrum are multiplied by the corresponding values in the `testsig` array. All the multiplication results are added to each other and the final sum is stored temporarily. This operation is repeated up to the 88th center frequency band (around 3.4 Hz) of the band-pass filter bank. The index containing the highest sum value is the index of the center frequency that has highest correlation with the one-hertz pulse train. That center-frequency value is returned by **m.bandit**'s leftmost outlet (see Figure 11)**.**

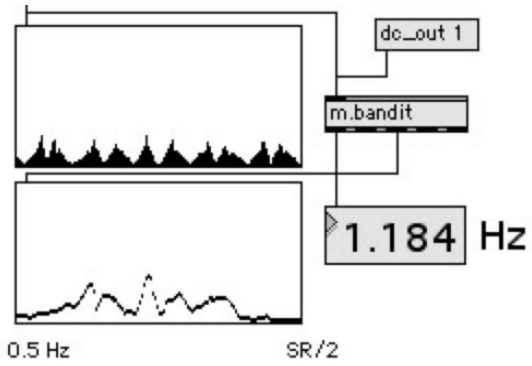


Figure 11. **m.bandit** outputting the instantaneous value of a beat candidate.

m.bandit is not a beat tracker in the true sense of the word. This object is able to do an instantaneous estimate of the fundamental frequency of a low-frequency signal, and outputs values that can be considered musical beats. It implements knowledge that helps providing good estimates about the most prominent beat in a movement sequence but, since it only outputs instantaneous values, it does not compare them or provides in anticipation what the next value will be. It is not even able to analyze if a movement sequence has, in fact, a steady

beat or not.

If one wants to do a temporal analysis of the output of **m.bandit** (beat tracking, for example) one should connect the outlet that outputs the fundamental frequency estimate to other existing objects in the library. The object **m.clock** models a simple tempo tracker that takes the fundamental frequency output from **m.bandit** as input, and produces a tempo estimate based on that data. **m.clock** can be effectively used to enable a dancer to control the tempo of a musical sequence in real time.

Ignoring Phase

As mentioned earlier, **m.bandit** does not calculate the signal phase spectrum and, consequently, the software that was developed is not phase sensitive to dance movement as captured by a video camera. This decision was taken after doing some experiments with earlier versions of **m.bandit** that calculated the signal phase spectrum and outputted a 'bang' message whenever a phase zero-cross occurred.

Having a system that is phase sensitive seemed initially a good and logical idea to implement. However, it was gradually abandoned. The main reason has to do with the fact that **m.bandit** is outputting the estimate for the fundamental frequency of the signal at frame rate, and this information tends to vary considerably since there is some noise in the system. The camera grabs the images, the computer digitizes them, and some processing still has to be done in order to provide the frame-differencing signal for analysis. Very small delays in

this processing cycle can introduce errors in the measurements, which means that a sudden jump in value in the frequency estimate would cause the phase synchronization to jump as well. A statistical approach to the measurement of the tempo estimate seems to be the appropriate thing to do in this system.

Also, when the dancer accelerated or decelerated the tempo in a movement sequence, the phase zero-cross would jump abruptly for every new fundamental frequency estimate. This implied that a lot of denoising on that information, such as ways of smoothing out these changes in phase, had to be done at runtime. Even so, I still thought that the phase zero-cross message output could be used as a musical rhythm generator, but this information needed to be quantized so that the rhythms created by this generator fell in places according to the beat. The idea of utilizing the phase of the detected fundamental frequency was therefore abandoned.

On the other hand, trying to build a system that is phase sensitive for musical tempo control in dance does not seem to be that crucial after all. In interactive music performance, having a system that can accurately follow a player's positioning in tempo is a crucial aspect since following the performer asynchronously can jeopardize the performance of the piece. As poignantly pointed by Scheirer (1998) the positioning in time of musical events is a crucial aspect in music performance. However, in dance performance (interactive or not), the music tends to provide a temporal grid from which dancers can move in and out, and synchronize to it or not. In fact, this aspect — somehow typical in the temporal interaction between dance and music — can create a powerful

audiovisual counterpoint between the rhythmic articulations of both during performance.

The musical tempo control features provided by the library model this latter aspect in dance performance. Since the system is not sensitive to phase, the dancer can control the tempo of a musical sequence asynchronously. However, triggering musical events or sequences from the dancer's actions is contemplated by the library through the object **m.peak,** that outputs 'bang' messages when a peak above a certain threshold is detected, thereby allowing the triggering of musical events or sequences when a dancer performs a rather accentuated body action.

Controlling Musical Tempo from Dance Movement

In order to enable a dancer the control of musical tempo, the beat frequency of **m.bandit** is analyzed by **m.clock** at frame rate. **m.clock** is an adaptive clock that outputs and adapts to the tempo according to certain rules. The best description for **m.clock** behavior is that of a filter. **m.clock** receives from **m.bandit** a tempo estimate, and filters out any unwanted information for musical tempo estimation and adaptation. It could be said that this object acts in part as a denoising filter for **m.bandit**.

The adaptive clock embedded in **m.clock** is modeled according to the following formula:

$$T_{(n)} = T_{(n-1)} + \Delta t \qquad (6)$$ [64]

$T_{(n)}$ is the clock value in milliseconds at frame n, $T_{(n-1)}$ is the clock value at the previous frame, and Δt is the difference between the measurement that was output by the band pass filter bank (converted to milliseconds) and $T_{(n)}$. This clock only works for values that can be considered musical beats — 300 to 1500 milliseconds (Rowe 2001), or 3.33 to 0.66 Hz. Each time **m.clock** receives a new value, the clock object checks if the value is within the musical beat boundaries. If the value is not within boundaries, that value is ignored and no calculations are performed. When a legal value arrives, if it is the first one to be within boundaries and if no default initial value was sent to **m.object**, $T_{(n)}$ gets initialized to that value.[65] For subsequent legal beat values, the clock object checks if the variation between the received value and the current clock time is within the allowable margin for variation. If it is, the new tempo estimate is computed according to

---

[64] Elsewhere (Guedes, 2003) I proposed a slightly different formula for the clock behavior, in which $T_{(n)} = aT_{(n-1)} + b(T_{(n-1)} + \Delta t)$. The coefficients *a* and *b* could be used to set the degree of "strictness" of the clock. *a* and *b* had to be both greater than zero and *a* =1-*b*. This was intended to allow further resistance to tempo change by the clock. If the user wanted the clock to be extremely strict, coefficient *a* could be set to a value close to 1. This would make the clock very little sensitive to tempo changes induced by the dancer. This formula worked well with videos that were being used for analysis at the time, in which the dancers were dancing to music with very strict tempos (e.g. samba). However, during the first experiments with the clock operation in real time using live input, I realized the simplification of the initial formula presented above worked better. The main reason has to do with the fact that the user only has to worry about the control of one free parameter for the clock's calculation (the allowable amount for tempo deviation) as opposed to two (amount of variation and strictness).

[65] **m.clock** can also be initialized with the message 'accept t' (t is the a time value in milliseconds between beat boundaries). This is a helpful feature in case one wants to start a musical sequence at a certain tempo to be subsequently changed by the dancer. This can also be used to optimize the **m.clock**'s initial estimate of the tempo. Since the clock, when not initialized, will accept the first value from **m.bandit** that is between musical beat boundaries, if the value is a 'noisy' measurement (for example, caused by a movement inflexion that outputs a completely different value from what is expected to be the tempo) it can be very hard for **m.clock** to retrieve the real tempo value. The message 'accept' can also be used to initialize the clock with a value at runtime.

equation 6.[66] If it is not, the current tempo estimate remains unchanged ($T_{(n)} = T_{(n-1)}$).

Performing the Tempo Adaptation

The allowable margin for variation (delta) is a percentage of the current beat value, defined by the user. This value, expressed as a floating-point number between zero and 1, is multiplied by the previous tempo estimate (delta* $T_{(n-1)}$) each time a new tempo estimate is calculated. When a new legal beat value is accepted for calculation, **m.clock** calculates the *absolute difference* between that value and the current tempo, and checks if that value is within the allowable margin. If the value is within that margin, Δt becomes the *difference* between the accepted value and $T_{(n-1)}$, and the current tempo estimate is updated. When one wants to control the tempo of a musical sequence through the output of **m.clock** (for example, to control the tempo Max's **metro** object), one should interpolate between values (for example, using the **line** object) in order to smooth the output. The amount of allowable variation for tempo adaptation can be input as an argument to the object. The amount of variation can also be changed by sending the message 'delta f' to the object (f is a floating-point value between zero and 1). If no values are introduced when **m.clock** gets instantiated, the object initializes the allowable margin for change to the default value of 15% of the current beat value (delta = 0.15).

---

[66] I thank Ali Taylan Cemgil for suggesting this approach for the clock behavior.

An adaptive clock that exhibits the behavior described above can have many interesting applications for interactive dance performance when applied as an instrument that enables the musical tempo control by a dancer. As mentioned before, this clock only adapts to the new tempo if the new measurement is within an allowable margin for change. This allows, for example, that a dancer moves rhythmically outside of the current tempo and no tempo changes will occur. If the dancer wants to change the tempo he has to synchronize with the current tempo and introduce slight changes to it gradually, within the allowable margin for tempo adaptation. This produces a beautiful effect of *carrying* the tempo over time. The faster the dancer moves in synchronization with the tempo, the faster the tempo will be, and vice-versa.

On the other hand, the operator of the system can change the margin of adaptation at runtime, so she can decide if the tempo will change or not according to the dancer's rate, and also determine how radical the change will be. This can be done through passing different values of the allowable margin for tempo variation to **m.clock** via the message 'delta.' Values for delta very close to zero will make it harder to change the tempo (the value zero will not allow changes at all). Values closer to 1 will allow a large range of values to be considered as a margin for tempo adaptation, and the clock will stop behaving like a clock since virtually any value can be considered a good estimate. As an example, let us suppose that delta has the value of 1 and the current tempo estimate is 750 ms. The margin for tempo adaptation will range from 750 – (1*750) to 750 + (1*750), i.e. 0 to 1500 ms, which exceeds the range of musical beats.

m.clock still offers other possibilities for a composer to give a dancer several degrees of control of musical tempo in real time. Besides utilizing the message ‘delta’ to change the allowable margin for adaptation, the user can also reinitialize the object at runtime with different values for the initial tempo estimate through the message ‘accept.’

CHAPTER VII

PRACTICAL APPLICATIONS OF THE SOFTWARE LIBRARY AND *ETUDE FOR UNSTABLE TIME*

Introduction

This chapter is divided into two parts. In the first part, I will discuss the practical applications of the software library. In the second part, I will discuss *Etude for Unstable Time,* the piece that was created to demonstrate the use of this software library. In the first part, along with the discussion about the practical applications of the library, I will give examples about how the objects should be connected in each of the situations discussed. The examples will be complemented with short video clips illustrating the practical application of the library. These short clips were taken from the performances of *Etude for Unstable Time*. In the second part, I will discuss important aspects pertaining to the conception and creation of the piece. I will describe important aspects related to the collaboration with choreographer/dancer Maxime Iannarelli, discuss other aspects that influenced the conception/creation of the piece, present a description of the Max/MSP patch, and describe the structure and music of *Unstable Time*.

Practical Applications of the Library

There are two main practical applications envisioned for the m-objects library: the generation of musical rhythms and the control of musical tempo from

dance movement. Although all the objects that comprise the library are not to be used exclusively in these situations, it was these two possible ways of mapping dance to musical rhythm that motivated this dissertation.

In this section, I will discuss the possible ways of treating the output of **m.bandit** in order to enable a dancer to generate musical rhythm and to control musical tempo in real time. Three types of generic situations will be discussed: generating musical rhythm from dance movement, changing the tempo of a musical sequence, and establishing the musical tempo of a musical sequence without a previous estimate. The types of connections between the objects in these three situations will be explained, and I will discuss the post-processing that sometimes needs to be performed on the output of the patches in order to obtain musically acceptable results.

Real-Time Generation of Musical Rhythms from Dance Movement

The generation of musical rhythms through bodily motion can be done utilizing the normalized output of the four harmonics of the signal as calculated by **m.bandit. m.bandit** outputs a list in the form of $f_1\ a_1 f_2\ a_2 f_3\ a_3 f_4 a_4$ from its second outlet, in which f and a correspond respectively to the frequency and amplitude of the harmonics (f_1 is the fundamental). This output from **m.bandit** can generate four independent consonant rhythmic layers, each one corresponding to one harmonic of the signal, which can be used to provide the rhythms for a melody or musical texture. Once this list is unpacked by Max's **unpack** object, one can convert the frequencies to milliseconds to feed the inlet of the **metro**

object, and use the amplitude values as thresholds that turn this object on and off. The example below illustrates this situation.

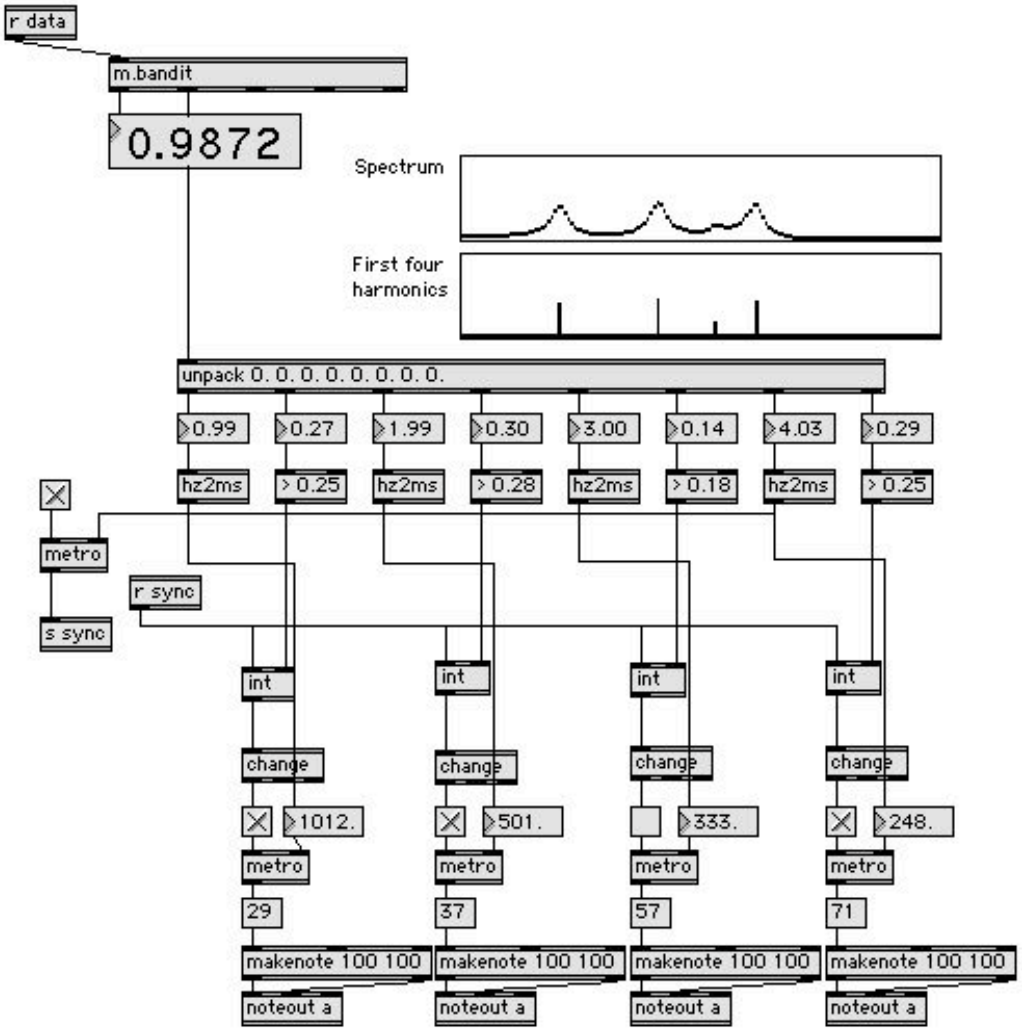


Figure 12. Musical rhythm generation with **m.bandit.**

This figure shows a patch fragment in which the harmonic list output of **m.bandit** is used to control four **metro** objects that trigger MIDI note on events. To produce output given above, I used a theoretical signal of fundamental 1 Hz containing three harmonics at different intensities. The rate at which each **metro** object outputs the 'bang' messages is the frequency of each harmonic converted

to milliseconds by the object **hz2ms** — a subpatch that converts Hertz to milliseconds (see Figure 13). Below the amplitude of each harmonic, the object **>** (greater than) has the threshold value as argument. This is used to turn on or off the **metro** object. The note on message is only triggered when its corresponding harmonic amplitude has passed the threshold value.

In Figure 12, the triggering of the MIDI notes is also synchronized by the rate of the highest harmonic. The leftmost **metro** object in the patch is outputting 'bang' messages, at the rate of the highest harmonic, that are sent remotely through the **s** (**send**) object. This serves as a *quantization grid* for the synchronization of the overall output. That is, the **metro** objects are turned on and off in synchronization with the highest harmonic value. This rough quantization procedure is performed in order to avoid that the **metro** object for each rhythmic layer turns immediately on, unsynchronized with the other objects,[67] when the threshold value is overtaken. In Figure 14 we can see the difference between the quantized and non-quantized outputs of the four rhythmic layers — fundamental frequency (f layer), second harmonic (f/2 layer), third harmonic (f/3 layer), and fourth harmonic (f/4 layer). Note that the f/3 layer practically disappears when the output is synchronized by the highest harmonic (the attacks that do not coincide with the quantization grid appear shaded in the figure).

---

[67] The user of the system can, of course, disregard this aspect and just have four unsynchronized rhythmic layers in proportion to the harmonic rates, without having any type of synchronization performed between them. However, when working with live input, one should interpolate the rate output (using **line**, for example) in order to smooth out the changes in between the values of each harmonic frequency.

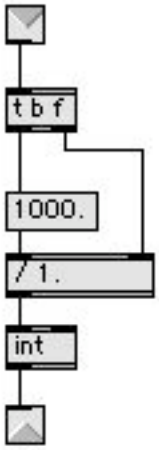


Figure 13. Subpatch **hz2ms.**

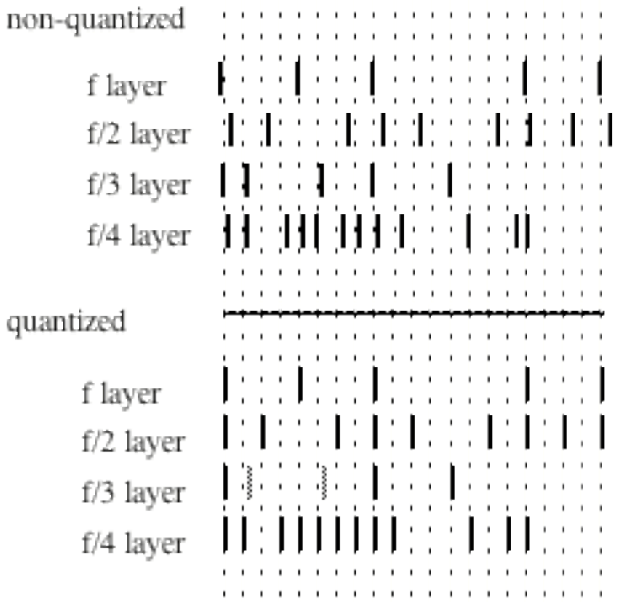


Figure 14. Difference between non-quantized and quantized outputs.

Generating rhythms directly from the output of **m.bandit** — by utilizing the amplitude and frequency of the harmonics of the signal to control the rate and triggering of rhythms — can produce very interesting effects in the real-time interaction between dance movement and music. Since the fundamental frequency varies at a rate that is proportional to the movement, a waving effect in the rhythm

that accompanies the dancer's movement can be heard. This produces a very organic relationship between rhythm and movement.

In the first part of the piece that was composed, a technique similar to the one presented above was used to generate the rhythms of the melodies that are played when the dancer moves. Instead of synchronizing the triggering of the **metro** objects by the highest harmonic rate, I synchronized them by the rate of the fundamental. The amplitude of each harmonic was scaled to range from 0 to 127 in order to provide note-on velocity values for the melody that were proportional to the amplitude of the harmonics. This was intended as an added expressive effect for this section of the piece. Examples 2 and 3 from the DVD-ROM show several moments from *Etude for Unstable Time* in which the dancer is generating musical rhythm from bodily movement.

Another way for generating musical rhythm through dance movement can be implemented utilizing this library. This way concerns the utilization of the exact harmonic proportions of the fundamental frequency. As noted in the previous chapter (subsection Determining the Musical Tempo of a Dance), **m.bandit** outputs the *center frequency of a frequency band* that correlates best with the fundamental frequency of the analyzed signal. The *real frequency* of this signal is within that frequency band. The output of the harmonic frequencies follows the same principle; i.e., **m.bandit** outputs the *center frequency* that is closest to the *real* harmonic frequency. This means that there is a slight error in the frequency proportion of the harmonics with the fundamental. That can be clearly seen in Figure 12. The fundamental of the theoretical signal is 1 Hz, and

the object outputs 0.9872 Hz as the fundamental, which corresponds to 1012 ms. The three harmonics above the fundamental have the frequencies 1.99, 3.0, and 4.03 Hz, corresponding to 501, 333, and 248 ms respectively. This slight error in the frequency values makes the proportion between the rhythmic layers to be not harmonic in the strict theoretical sense. If one wants to work with exact harmonic proportions (although the difference between this situation and the actual output is very feeble), one can multiply the output of the fundamental frequency by the harmonic rates in order to give the **metro** object the exact value, and still use the amplitude thresholds to trigger the rhythmic layers.

Real-Time Control of Musical Tempo from Dance Movement

The real-time control of musical tempo can be done by connecting the output of the fundamental frequency of **m.bandit,** converted to milliseconds, to the leftmost inlet of **m.clock.** As noted in the previous chapter, **m.clock** works as a filter to this output of **m.bandit.**

m.clock processes the input of **m.bandit** in two stages. The first stage of processing only accepts values for calculation that can be considered musical beats. The second stage of processing compares an accepted value with the current tempo estimate and performs the tempo adaptation if that value falls within a pre-defined margin for adaptation that can be input as an argument to the object. I will present two generic ways in which musical tempo control can be done using the library: the change of musical tempo in a sequence that starts being played at a given tempo, and the guessing and control of musical tempo by the

library without any previous estimate. I will also show in some detail the musical tempo control patch I built for *Etude for Unstable Time*, which shows the specific utilization of the tempo control objects in the library for this piece.

Basic Patch for Tempo Control

The simplest possible way of using **m.clock** with the output from **m.bandit** is shown below:

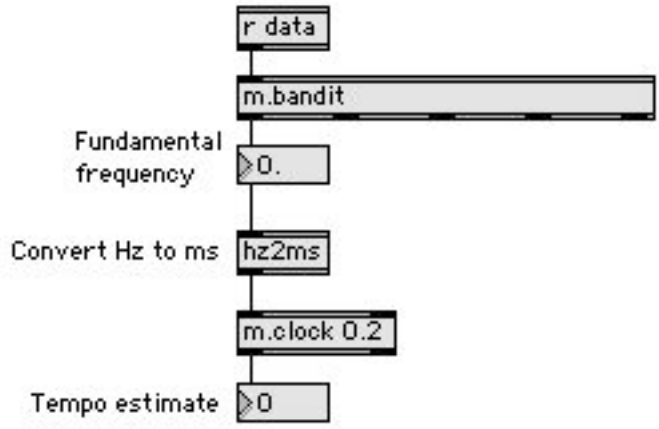


Figure 15. The simplest way of using **m.clock.**

The fundamental frequency output is converted to milliseconds and sent for analysis to **m.clock. m.clock**, which can be initialized with a value for delta as an argument to override its default (0.15), provides the tempo estimates based on its own system of rules as explained in the previous chapter.

This simple patch is by no means the ideal way to produce a correct tempo estimate for musical tempo control, although it provides a skeletal representation of the use of **m.clock** in conjunction with **m.bandit**. In the example above, the object was initialized with delta = 0.2, or 20% of the current beat estimate. If

m.bandit started receiving brightness-difference data, **m.clock** would start processing the output of **m.bandit** as soon it received a value that could be considered a musical beat. If a dancer started a pulsed movement sequence that was being analyzed by this patch, it would be very hard for **m.clock** to come up with the right tempo estimate. This is because the object would capture the first frequency that satisfied the condition of being considered a beat, and would initialize the first tempo estimate to that value, with a rather slight margin to readapt.

The fundamental frequency estimation is done at frame rate. When a dancer initiates a movement sequence, the signal's fundamental frequency will change considerably before it gets stabilized, since it takes at least one period of the actual frequency of movement for **m.bandit** to come up with a reasonable fundamental frequency estimate. If **m.clock** is initialized with a value that diverges considerably from the actual movement rate, it will never adapt to the actual rate because the margin given for adaptation is rather small. However, if one initiated the processing of **m.clock** after **m.bandit** started outputting a rather stable fundamental frequency estimation, such a simple way of calculating the tempo estimate would work provided that one inserted a **gate** object between fundamental frequency output of **m.bandit** and **m.clock** that only allowed the data to pass to **m.clock** after **m.bandit** produced a rather stable output. This would prevent **m.clock** from accessing the initial erroneous values.

Changing the Musical Tempo of a Sequence Being Played

In order to give a dancer the control over the tempo of a sequence being played with a known tempo, one should initialize **m.clock** to the tempo of the sequence by sending the message 'accept' to the object (see Figure 16). Let us suppose the object **s** (**send**) in the bottom of the patch is sending the tempo to a musical sequence that is being played with an initial tempo of 700 ms. Once **m.clock** gets initialized with 'accept 700' and 'delta 0.', the adaptive clock will not change its tempo estimate with the data coming from **m.bandit**, since the allowable margin for adaptation is 0. The sequence that receives the tempo information from **m.clock** can safely play at this tempo for as much time as one wants, since **m.clock** will not produce any tempo adaptation caused by the movement of the dancer. The figure below illustrates how can one initialize **m.clock** in order to exhibit the behavior just described.

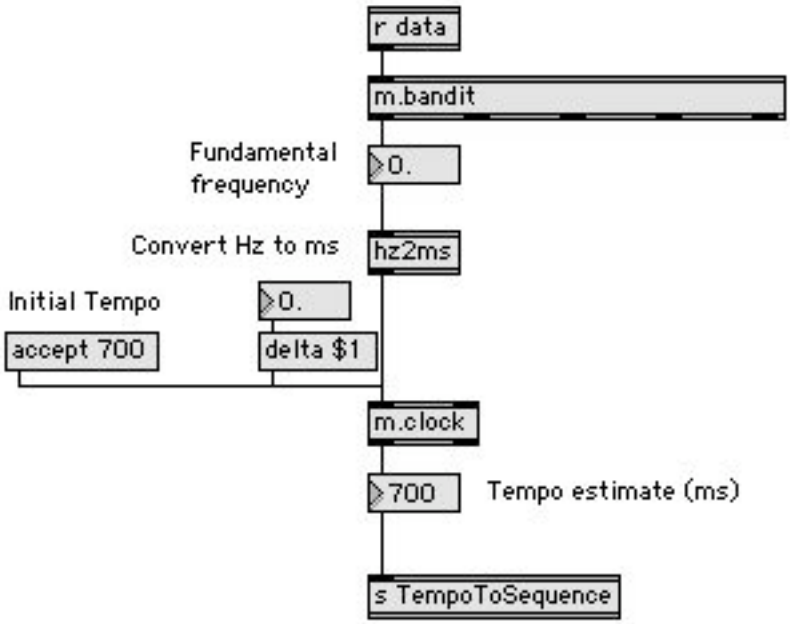


Figure 16. Initializing **m.clock** with a known tempo value.

To give the control of musical tempo to the dancer, one should gradually increase the value of the floating-point number box above the message 'delta $1' in order to enlarge the margin for adaptation, and therefore produce tempo estimates that take the output from **m.bandit**. Smaller values for delta (around 0.2 or less) will allow the introduction of *expressive* variations in the tempo; larger values for delta (up to 0.8) allow radical tempo changes such as fast *accelerandi* and *ritardandi* in the musical tempo. For larger values of delta, the output of **m.clock** should be interpolated in order to avoid abrupt tempo changes. Since the dancer controls the tempo asynchronously, it is up to the user to find the values that fit best for the situation in which is used. A little experimentation with the interpolation time is in general required to achieve the desired results. In Figure 17, I give an example of how the tempo estimates from **m.clock** can interpolated by **line.**

This type of patch can lead to very interesting effects for musical tempo control in real-time. Since the dancer can only introduce alterations to the tempo when she articulates the movement at the rate of the current tempo within the allowable margin for variation, different values of delta offer different levels of resistance to tempo change— smaller values of delta offer more resistance than larger values. Also, since **m.clock** only adapts to tempo once the values of the fundamental frequency given by **m.bandit** fall within the allowable margin for variation, the dancer can always *move away* from the current tempo and get back in, in order to change it.

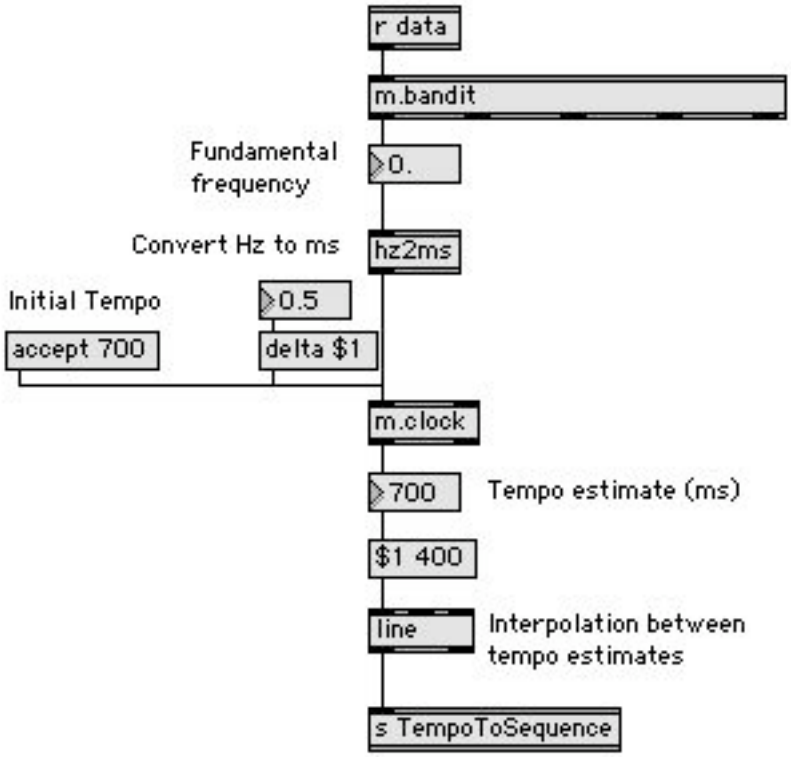


Figure 17. Example of interpolation between tempo estimates using Max's **line** object.

Establishing the Musical Tempo from Dance Movement without an Initial Estimate

Determining the tempo musical tempo of a dance without any previous hint is a much harder task than simply changing the tempo of a musical sequence in which the starting tempo is known. However, the m-objects library can still do that with a fair degree of success.

The process by which **m.clock** arrives at the first tempo estimate has been already explained. As already noticed, after arriving at the first estimate, **m.clock** updates the estimates based on the value given for delta. If delta is too small,

m.clock may never arrive at the correct estimate. A possible way to circumvent this situation is to provide a very large initial value for delta (e.g. 0.8) in the beginning of the capture, and gradually decrease that value as the tempo starts following the movement. Another possible approach, somehow more refined, consists in the utilization of **m.weights** as an "informer" to **m.clock** during the time the movement sequence gets initiated, until a stable tempo is reached. Since **m.weights** outputs the one-hundred millisecond 'bin' that got more hits in the past 60 frames, the output of this object can feed values to **m.clock** via the 'accept' message. This way, **m.clock** updates faster as it is being constantly reinitialized. Once the tempo gets stabilized one should stop sending 'accept' messages to **m.clock** so that the object updates the estimates according to its own system of rules. An example patch showing how this can be done is given below.

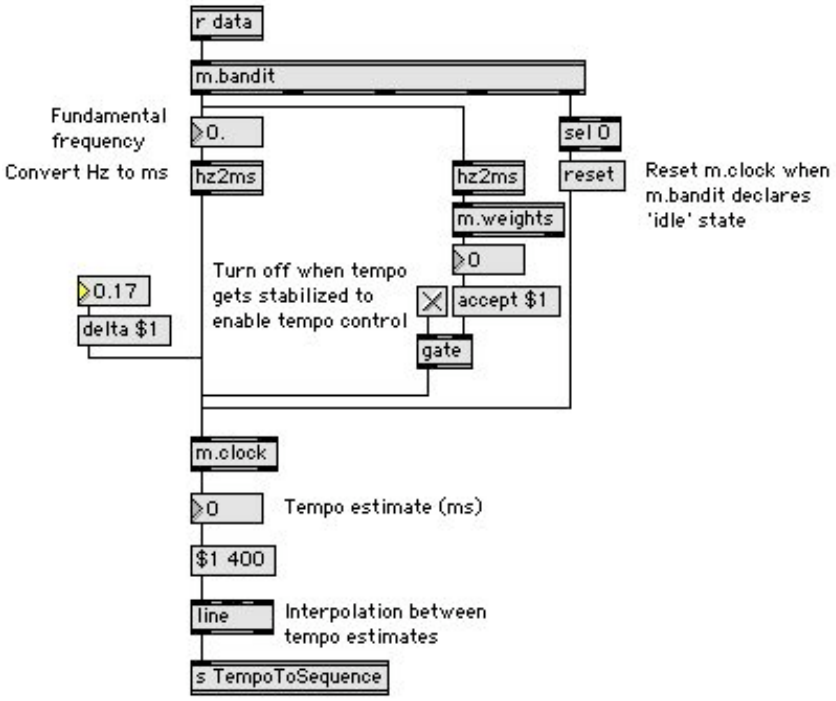


Figure 18. Utilizing **m.weights** to help **m.clock** finding a 'blind' tempo estimate.

The above patch illustrates how **m.weights** is used to effectively help **m.clock** finding a tempo estimate without previous initialization. As soon as the gate is closed, **m.clock** will produce a new tempo estimate if the new beat candidates fall within the allowable margin for adaptation — in the above case, up to ±17% of the current beat estimate. In the above example, I also show how one can utilize the moving state flag from **m.bandit** to reset **m.clock** each time a dancer stops moving. Such situation would be useful if one wants the system to adapt to the tempo of the dancer each time he starts a new phrase, for example.

The Max/MSP patch designed for the performance of *Etude for Unstable Time* applies the library in the ways described above with a few minor changes. These changes were necessary in order to accommodate the idiosyncrasies of the piece. Examples 4 and 5 in the DVD-ROM illustrate some applications of the m-objects in musical tempo control from dance movement. Example 4 from the DVD-ROM shows two short sequences illustrating the how the dancer controls the speed of sample grains produced by the granular engine used in the patch for the performance of *Etude for Unstable Time*. In this patch, every time **m.bandit** declares 'idle state' (moving flag equals zero) the speed of the grains increases. Example 5 shows a moment during the performance of the piece in which the dancer does not control the speed (tempo) of the grains. However, one can see slight accelerations in the tempo as the dancer is moving back stage as the sensitivity preset of **m.bandit** is set to declare 'idle state' when the dancer is back stage. This causes the speed of the grains to increase. In the upcoming section, I

will describe the piece *Etude for Unstable Time,* including the Max/MSP patch that was designed for its performance.

Etude for Unstable Time

Introduction

Etude for Unstable Time was the first piece composed with the m-objects library. This piece puts in evidence the utilization of the software library, and therefore some conscious options were taken for that purpose: the musical materials are limited, and controlled to a great extent by the m-objects. I wanted to put in evidence the performance of the software thereby restricting the temporal articulation of the music (musical rhythm and musical tempo) of the sonic material to the output of the m-objects library. This piece is thus an *Etude* on the utilization of this software, or, as I prefer calling it, a *prototypical* piece for interactive dance. In the upcoming section, I will note important remarks in the conception of the piece. Afterwards, I will provide an overview of the Max/MSP patch I utilized for this piece, and finally, discuss the structure and music of *Etude for Unstable Time*.

Technical Details

This piece was done using a Macintosh G4/400 MHz (256 MB of RAM), running Mac OS 9.2.1, for both the video analysis and processing and sound processing utilizing Max/MSP (version 4.2) with the softVNS 2 library (version FC2_3b). Besides doing some real-time sound processing, Max was also

controlling a MIDI virtual software synthesizer via the Open Music System (OMS) Inter-Application Communication (IAC) bus. A Philips USB webcam (model ToU pro) was used to do the video tracking. The four realizations of the piece that are included in the DVD-ROM (folder EtudeForUnstableTime) were recorded in Amsterdam, The Netherlands, on September 27, 2003, at the Blue Haptic studios. The sound was recorded directly from the computer's line output utilizing a minidisk recorder, and the video was recorded utilizing a mini DV camera, with a reference sound captured directly from the camera's microphone. The sound of the video clips was post-synchronized utilizing the sound from the camera as source.

The Collaborative Process

Elaboration of a Concept

This piece was created in collaboration with choreographer/dancer Maxime Iannarelli. Iannarelli is a young French choreographer/dancer who has demonstrated a great interest in developing this piece. He had never danced or choreographed utilizing interactive technology and was eager to have his first experiment in this type of environment. Since this was the first time we were collaborating, we first talked extensively about how the piece would be like. We both agreed not to have a story line for the piece. Iannarelli, like me, prefers to have a more *abstract* approach to the interaction between movement and music, instead of trying to *tell a story* with a piece.

Figure 19. Choreographer/Dancer Maxime Iannarelli dancing *Etude for Unstable Time*.

This piece is essentially about temporal instability. It is about the *illusion* one has of the body as a perfect mechanical entity, capable of maintaining a certain temporal order by performing repetitive actions that actually seem to be perfectly regular. In that respect, the piece attempts to unveil the differences between our perceptions of what seems to be an orderly/rhythmically articulated bodily action and the actual (musical) results these actions produce.[68] As can be witnessed from the video clips of the piece, the rhythms produced by the computer when it accompanies the actions of the dancer (notably in the first section of the piece) are rather unstable.[69] However, this *instability* is very

---

[68] In that respect, as noted by Hodgins (1992), full body actions tend to be *noisier* than actions of smaller body parts, as in music performance.

[69] In the second section of the piece, I also make use of this concept of temporal instability through a different perspective. In this section, which is about musical tempo control, the dancer controls the tempo at which fairly larger sonic grains — around 150 ms — are played. When **m.bandit** declares 'idle state,' the time between grains is accelerated up to 30 ms. At times, **m.bandit** declares this state when a dancer stopped briefly — or it fell into the idle_int margin of the object —, there is an acceleration in the tempo that is interrupted immediately by the dancer's movement. This causes again the effect of temporal instability as the dancer can only control the tempo at which the grains are played when he's moving.

organic as the rhythms accompany the bodily actions. There is a rhythmic flow that bears a strong relationship to the movements of the body, and one can actually feel how those rhythms are projected in time by the dancer's movement.

A Trio for a Dancer, Computer and a Composer

Etude for Unstable Time could have as subtitle *A Trio for a Dancer, Computer and a Composer*. The three entities involved in the performance of the piece all play important — yet distinct — roles. The dancer produces movement sequences that are being analyzed in real time by the computer. Besides analyzing the dancer's movement, the computer is producing music that is algorithmically generated. The computer is producing a non-deterministic musical output based on algorithms that produce a random-weighted output. The dancer and the composer (myself) are both responding to the output produced by the computer. The dancer responds through movement to the musical output; the composer responds to the movement and to the computer's output by letting or not letting certain musical processes develop and by controlling the degree of interference of the dancer over the musical processes, namely the temporal articulation of the music. A tripartite feedback network is thus established among the three entities involved during performance.

A Collaboration in Real Time

The *feedback network* described above allows for a collaboration in real time to take place between the three entities involved in the performance of the

piece. This aspect is also very important to consider in the conception of *Unstable Time*. One of the main goals of this study was to carry out what I called an interactive process of collaboration up to and including the performance of a piece. In Chapter 5, I proposed a framework for computer-mediated interaction in dance performance that called for an active participation of the composer/musician in the process of interaction. In the proposed interaction framework, the computer acts as a mediator by facilitating an otherwise-impossible-to-achieve channel of communication between the musician/composer and dancer/choreographer. In this piece, the design of the Max/MSP patch allows the three entities (dancer, composer, and computer) to collaborate in real-time. The patch not only facilitates the communication between the dancer and the composer, but also gives the computer the role of a collaborating partner.

The generative algorithms produce a musical output that seldom repeats itself. The musical material produced by those algorithms thus triggers different responses by the dancer and me every time the piece is performed. Since I am controlling how, when, and under which conditions those algorithms are played, I can always choose for how long — and under what circumstances — the algorithms are active during the performance of the piece. Even though the computer has a limited role in the decision-making process during the performance of the piece, it is constantly providing musical materials *for appreciation* that can be more or less utilized, or even transformed, during performance.

Composition

From a compositional standpoint, I wanted to explore the *instability* created in the temporal articulation of the music by the dancer's movement. This bodily controlled *instability* seems to provide possible ways of expressive temporal articulation in a piece of electronic music. I also wanted to create a piece that existed as a set of algorithms that produced a weighted-random output, which could be triggered alone or in combination by me during the performance of the piece. Instead of a plan, there was a concept I developed in collaboration with Iannarelli about possible ways of interacting with the musical materials that were being generated by the computer and controlled by me in real time. There was this idea of a piece that had a basic, simple structure, and an improvisational character. The piece evolved as the dancer and I decided, through the way the dancer was responding to the music, and as the musical result provided by the algorithms was considered satisfactory from a musical standpoint.

There was also a *game* being played between the dancer and the computer over the control of the musical tempo. During the rehearsals of the piece, Iannarelli became very aware of when he had the control over the musical tempo or not. I warned him that during the execution of the piece I would *steal* from him the control over the tempo unexpectedly. This was intended to create a tension between the movement and the music. There are sections in the piece in which one can see that the dancer has the control over the tempo, whereas in other sections, this control does not exist at all. Since it took a short while for the dancer

to realize this fact during the performance, there is this expressive tension created at times between the movement and the temporal articulation of the music.

The end result is a quasi open-form piece, in which a macro structure is being observed, but the musical content and the dance vary considerably due to its improvisational character, and to the musical content generated by algorithms that produce a weighted-random output. Each time the piece is realized, new musical content and dance are produced. To demonstrate this, I included four different realizations of the piece in the accompanying DVD-ROM.

Patch Structure

The Max/MSP patch that was created for the performance of the piece has a structure similar to the schematic representation of the software presented in Chapter 5 (Figure 3). In that chapter I propose a three-fold structure in which the processing performed by the computer is divided in three stages: video analysis, musical processing and generation of musical data for control.

The patch contains three main sections. The first section, called *Video Analysis,* performs the analysis of the video stream that is grabbed by the webcam. The second section, *Movement Periodicity Analysis,* utilizes the m-objects library to perform a musical rhythmic analysis, or musical processing, of the movement sequences, and outputs data that controls the musical output of the third section of the patch, called *Sound Engine*. The figure below shows the user-interface level of the patch. The arrows drawn in the picture represent the general flow of data within the patch (see Figure 20).

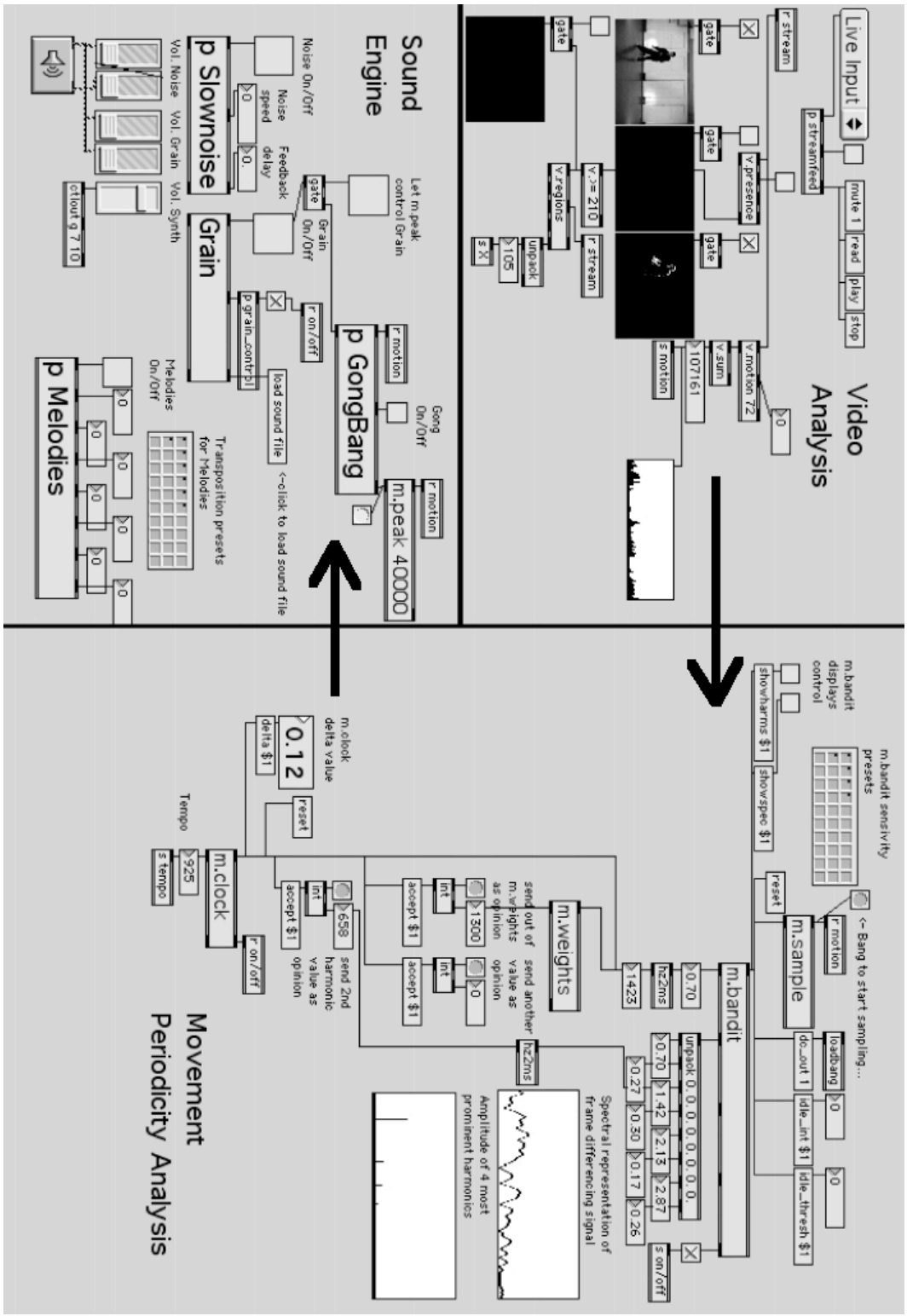


Figure 20. Patch for the Performance of *Etude for Unstable Time*.

Before the performance, the user has to tune the video analysis objects so that a good stream for analysis is produced. There should be a good contrast between the dancer and the background and one should set up the camera so that the full-body image of the dancer is visible at all times. After the initial setup and tuning are done, the operator of the patch has to focus pretty much on the *Sound Engine* part of the patch and on the tempo estimates provided by the *Movement Periodicity Analysis* section, in case they need to be corrected at run time. A detailed description of each section of the patch is given in Appendix D. For now, I will only describe how the extraction of musical rhythm and the control of musical tempo by the dancer are done in this patch.

Generation of Musical Rhythm in *Etude for Unstable Time*

The harmonic output of **m.bandit** is sent remotely to the object **Melodies** in the *Sound Engine* section of the patch for the generation of musical rhythm from dance movement in the piece. This object produces melodies whose rhythm is exclusively produced by the amplitude and frequency values of the harmonics of the fundamental frequency as extracted by **m.bandit**. The technique utilized to do this is akin to the one explained earlier in this chapter. The messages that send those values are not seen at the user-interface level; they were hidden in order to save space and to provide a clearer view of this section of the patch.

The Musical Tempo Control Patch for *Etude for Unstable Time*

The patch I created for musical tempo control for *Etude for Unstable Time* follows the principles explained above, this time adapted to the specific utilization of the library in the control of musical tempo in this piece (see Figure 21). In this section of the patch, one sees several objects and connections that lead to **m.clock** placed in the bottom of this patch section. **m.clock** broadcasts the message 'tempo,' the tempo estimate calculated by this object to other parts of the patch, namely to **Grain** and **GongBang** in the *Sound Engine* of the patch. **m.clock** accepts values for (re)initialization from three different sources: a value typed by the user in a number box, the output of **m.weights**, and the frequency of the second harmonic, besides the possibility to control the allowable margin for tempo adaptation. Since the patch was operated manually during the performance of the piece, all of the number boxes were connected to the right inlet of the **int** objects so that they could be sent to **m.clock** at the operator's discretion. The buttons above each **int** object allow to manually sending the values from each of the three sources when necessary.

This was a necessary functionality to implement in the patch for this piece. Since the piece was improvisational and experimental in character, I wanted to give the control of musical tempo to the dancer and perform corrections of the readings of **m.clock** whenever I found necessary. A special attention should be given to the fact that **m.clock** was also getting information from the second harmonic of the spectrum. This has to do with the fact that **m.bandit** often

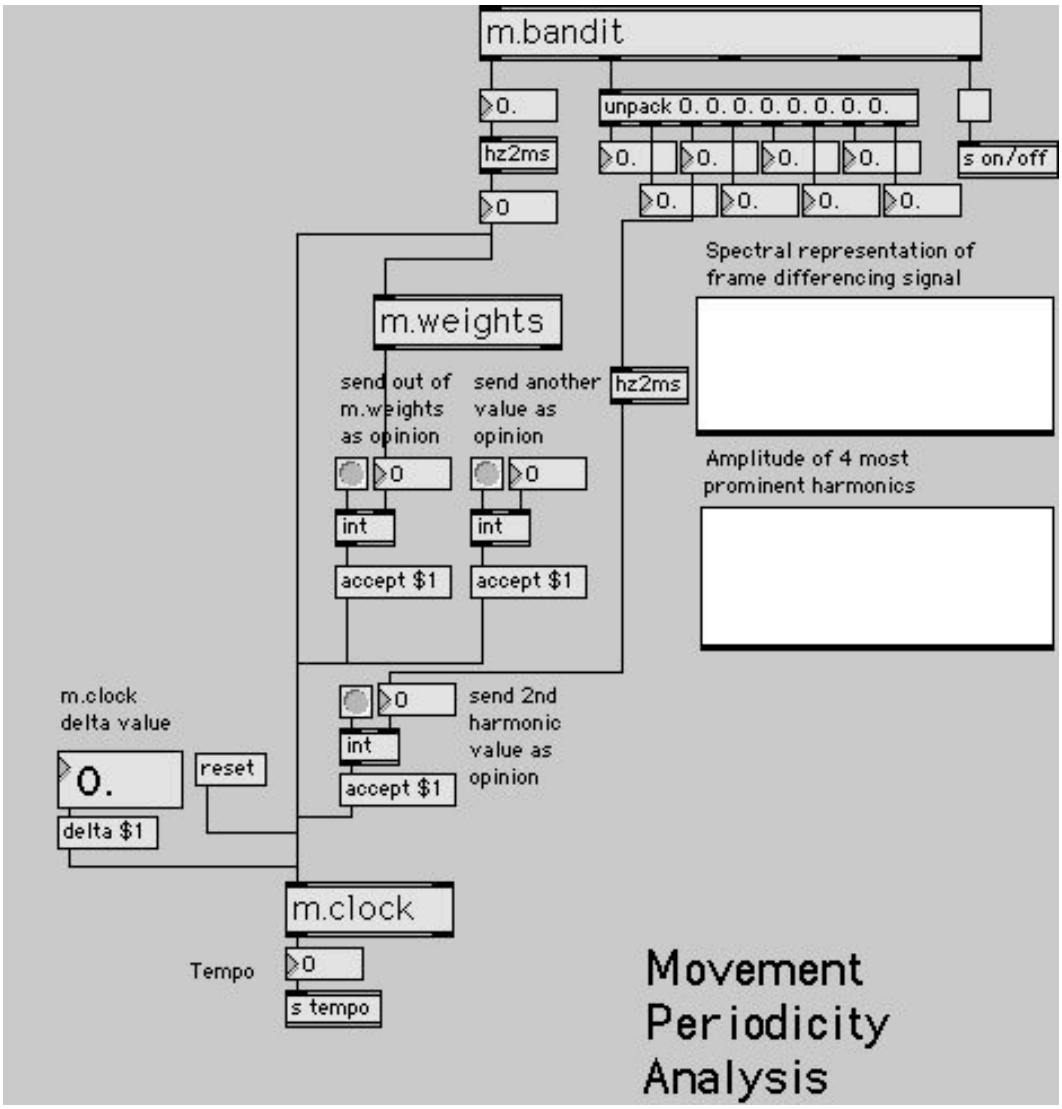


Figure 21. Patch fragment utilized for musical tempo control in *Etude for Unstable Time*.

performs “octave” jumps in the estimation of the fundamental frequency of the signal. This type of behavior in **m.bandit** should be expected since the band-pass filter bank tries to find the fundamental of the signal that is between 3.33 and 0.66 Hz. When the movement rate falls a little below or above these margins, and because the fundamental estimation is done by following a harmonicity criterion,

these “octave jumps” should be expected. Also, when a dancer is accelerating or decelerating the tempo it is common for **m.bandit** to produce readings that are an octave above or below the expected frequency when the rate of movement is changing.

Finally, in this piece I wanted to work with the partial indeterminacy of the readings by **m.clock** when performing ‘blind’ estimates. This was done with the purpose of inducing the dancer with an initial unexpected tempo that was generated from his impulse. As an example, see the beginning of the second section in each realization of *Unstable Time* (Example 6 from the DVD-ROM). Every time the dancer performs an impulse to start the music, the strings sample is played at different speed. Being able to send the second harmonic value to **m.clock** was sometimes necessary in order to effectively pass the control of the musical tempo to the dancer.

The Structure of the Piece

Etude for Unstable Time is divided in two main sections. Each section focuses respectively on the two possible utilizations of the m-objects in interactive dance performance: the generation of musical rhythms and the control musical tempo from dance movement. As I mentioned previously, *Unstable Time* was conceived around a very simple and basic structure that allowed for improvisation and experimentation to take place during performance.

The first section of the piece explores the generation of musical rhythms from the dancer’s movement. It was agreed that the dancer would start this section

in the upstage left position[70] and end it around center stage. This section is built around a gradual increase in movement actions by the dancer that trigger musical events. The piece invariably starts with the dancer moving very slowly, triggering very few musical events through accented bodily actions, and gradually increase the rate at which these actions trigger the musical events with a gradual increase of tension in these actions. This is helped, partially, by the increase in the sensitivity of **m.bandit** to the dancer's movement.[71] Several threshold values for the idle interval and idle threshold (from *less* sensitive to *more* sensitive) were stored in Max's **preset** object and were activated by me during the performance of this first section.

The second section of the piece is about musical tempo control. This section invariably starts with a strong impulse from the dancer around center stage (the end of the previous section) that triggers a gong-like sound and a granulated sample of string sounds starts being played. This section is about the *game* played by the dancer and the musical tempo control. As I mentioned earlier, during the performance of this section I warned the dancer that, at times, I would take the control of musical tempo away from him. He thus had to feel when he was controlling the tempo of the music or not. This aspect was complicated a bit further by an algorithm that accelerated the string sample grains whenever **m.bandit** declared 'idle state' to the patch.

---

[70] All spatial indications are given from the perspective of the viewer.

[71] As explained in Chapter 6 this is done through the messages 'idle_thresh' and idle_int.'

There is a feel throughout this section of a strong instability in the tempo. This is caused partly by the acceleration in the grains of the sample when **m.bandit** declares 'idle state,' but also by the fact that in the beginning of the section **m.clock** is producing 'blind' tempo estimates from the initial impulses of the dancer. As soon as one of the fundamental frequency values estimated by **m.bandit** fall within the acceptable margins that are considered musical beats, **m.clock** gets initialized to that value and then updates its musical tempo estimate when the subsequent values fall within the allowable margin for adaptation. In this section I also bring in the musical materials from the first section.

The Music for *Etude for Unstable Time*

The musical organization of the piece follows the two-fold structure described above. Certain objects in the *Sound Engine* of the patch are used in each section of the piece. Each object for musical generation embeds an algorithm that produces a certain type of output. Even though at any time the algorithms are activated they produce an audible musical output, the function of the composer in the piece is not to merely activate or deactivate the algorithms during performance. There is a compositional plan that is being executed, although within the constraints of a piece that is improvisational in character.

One could say that there are two compositional levels in *Etude for Unstable Time*. An initial level consisting of the algorithms that produce a certain musical output, and a second level, a *metacompositional* level, that consists in the

organization of the musical output of the algorithms in (real) time. [72] The musical algorithms designed for the piece provide the first compositional level. They already yield certain musical characteristics that will give the piece a certain type of sound and character. The output of these algorithms is then (re)*composed in real time*[73] during the performance of the piece, within the tripartite feedback network described earlier. Each time the piece is realized, the musical result will be different, although is bound by a certain type of *sound* — the musical output of the algorithms. The role of the composer in the piece is thus to manage and control the unfolding in time of the musical materials produced by the algorithms that operate at the so-called first compositional level. The algorithms of the patch produce a type of output that requires a composer to control the behavior of the algorithms during performance, by altering certain parameters, by combining their output with the output of the other algorithms, or simply by turning them on or off.

The two large sections of *Etude for Unstable Time* follow a rather simple musical plan that is in tandem with the simple structure and the purposes for which the piece was created. The first section uses objects **Melodies** and **Slownoise**. The object **Melodies** generates melodies whose rhythm is driven by the harmonic output from **m.bandit**. **Slownoise** is an object that generates

---

[72] This should not be mistaken with an interpretational level. Although a piece of music can in general have several interpretations, the level at which one interferes in the performance of *Etude for Unstable Time*, goes well above the mere interpretation of the output the algorithms. The operator of the patch can dramatically alter the musical result that is produced in each realization of the piece.

[73] The application of this term does not ignore Joel Ryan's ideas about composition in real time with electronic music.

resonant-filtered down-sampled noise bands whose centered frequencies are the frequencies of the MIDI notes generated by **Melodies**.[74] There is a tendency mask applied to the output of the algorithms that cause a progression in register up to the very high register. The MIDI notes being produced by the object **Melodies** that are heard in the form of glockenspiel/vibraphone-like sounds follow this path; **Slownoise** converts the MIDI notes to Hertz and uses the values to generate the center-frequencies for the down-sampled noise bands that accompany the melodies in this section.

As the melodies climb in register, their values go well beyond 127. Since the output of **Melodies** is being converted to Hertz, when the MIDI note values pass 136 the corresponding frequency will pass 22.05 KHz (one-half the sampling rate of the sound of the piece, 44.1 KHz). This causes a foldover effect in the frequencies of the resonant filters in the **Slownoise** object causing the bursts of noise that can be heard towards the end of the first section. This coincides with the point in the piece at which the dancer is performing more intense movement sequences. This *contradicts* the increase in **m.bandit**'s sensitivity throughout the section. As I mentioned earlier, there is a gradual increase in the response to the dancer's movement by **m.bandit** in this section. However, this increase in response is *counterpointed* by the gradual increase in register of the melodies that are generated in this section. At the point of maximum responsiveness by the system, the MIDI notes practically become unheard, as their values are so high, giving way to the noise bursts caused by the foldover effect in the center-

[74] A detailed description of all objects comprising the *Sound Engine* is given in Appendix D.

frequencies of **Slownoise**. Such type of contradictions between the response of the system to the dancer's movement and the musical output are common in *Etude for Unstable Time*. This is a way to circumvent the obviousness in musical response that certain movement actions can create.[75]

The music in the second section of the piece uses the objects **Grain** and **GongBang** as a base for the musical content. **Grain** granulates a strings sample and the duration between the grains is controlled by the output of **m.clock.** The position of the sample to be read is controlled by the position of the dancer in the horizontal axis. This enables the dancer to *travel* through the sample while moving. **GongBang** generates a chord of gong-like sounds when it receives a 'bang' message from **m.peak**. In the beginning of this section, the dancer triggers the gong-like sounds though strong body impulses. These impulses also start and stop the granulated strings sample. I gradually subvert this initial logic during the course of the section by interrupting the dancer's control of the playback of the sample, and by creating *surprises* to the dancer in regards to if his strong impulses actually trigger the gong-like sounds or not.

As mentioned earlier, this section is also about the game being played over the control of the musical tempo. At times the dancer has the control of the

---

[75] This type of preoccupation resonates with conceptual ideas presented in Chapter 5 about modes of interaction that can possibly be created in computer-mediated collaborations (cf. Chapter 5, A Possible Framework for Computer-Mediated Interaction in Dance). The idea of having a system that is operated in real-time by a specialist (in this case a composer/musician) aims to promote more elaborate ways of interaction. Having a system whose behavior can be modified in real time can diversify the output of the system and therefore create unexpected and intriguing effects to an audience in regards to the type of response given by the system. One can in a sense play with the spectator's expectations about the audiovisual interaction modes and thus establish a more sophisticated way of communicating with the audience by modifying the type of response from the system during performance.

musical tempo; at other times I take the control away from the dancer and gradually give it back. The tempo instability felt throughout the most part of the section is also enhanced by the fact that when **m.bandit** is not *seeing* the dancer, the IOIs between the strings sample grains decrease to a point that transforms what is otherwise heard as articulated chords into a granular texture. This situation is particularly visible at a point in the section (it happens in all performances) in which the dancer moves to upstage left and goes from left to right very slowly, *traveling* through sample without triggering any type of temporal articulation.

In this section, I also create a sort of return to the musical materials from the first sample by *bringing back* the objects **Melodies** and/or **Slownoise**, and by making the output from these objects interact with the musical output from **Grain** and **GongBang**. This return to the initial musical materials also signals a move towards the completion of the piece. The piece does not have a fixed end-point and it pretty much ends when the dancer and I feel like ending it.

The second section of the piece has a much looser plan in regards to the pre-defined *path* for the music. In this section, there is a lot more musical improvisation with the output provided by the *Sound Engine* of the patch and the dancer's response to the musical output that is being created. The structure of the piece and the simple musical plan that was utilized are summarized in Figure 22.

Figure 22. Summary of the music and dance plan for *Etude for Unstable Time*.

Olivia

The m-objects had their first public presentation in *Olivia,* a dance solo for children based on the cartoon character created by Ian Falconer. This solo was choreographed and danced by Isabel Barros, with music composed by me, and was premiered at Teatro Rivoli, in Porto, Portugal, on May 6, 2004. In this piece, I used the m-objects to create two *magical moments* in the show. The first was when Olivia went to the museum and pictured herself in a ballet class. When the dancer started dancing, a piano solo was generated utilizing the rhythms produced by her movement, thereby giving the impression that the character Olivia was controlling piano music. The second *magical moment* happened when Olivia was having insomnia at night and started rolling in bed until eventually began to dance. For this situation, I used a modified version of the **Melodies** object to generate glockenspiel-like sounds that accompanied the movement.

Closing Comments

This chapter is the last of three chapters in which I introduced important concepts that lead to the creation of a software library for musical rhythm generation and musical tempo control from dance movement, the creation of a *prototypical* piece for interactive dance in which the software is utilized and demonstrated, and the application of a concept for computer-mediated interaction in dance that yields for a collaboration to take place in real time.

In chapter 5, I initially formulated the hypothesis that existing musical qualities in dance rhythm could be detected in real time and provide an interesting

way for computer-mediated collaborations utilizing interactive computer systems. I also presented the conceptual and technical considerations that assisted in the creation of the software. In the conceptual considerations, I presented a possible framework for computer-mediated interaction, and the conceptual framework in which the software was developed. In the technical considerations part of the chapter, I showed that video cameras can actually be used to represent in the temporal domain all the periodicities in dance movement that bear the temporal characteristics of musical rhythms.

In chapter 6, I provided a summary description of the library, focused on the frequency-domain representation of the signal that carries the information about the periodicities in movement, and on techniques that can be utilized for dance musical tempo tracking and control.

Finally, in this chapter I showed the practical applications of the library in the two situations it was designed for: the generation of musical rhythm and the control of musical tempo from dance movement. I showed in detail the types of connections that have to be made to achieve such results, and illustrated the practical applications of the library with small video clips gathered from the four performances of *Etude for Unstable Time*. *Etude for Unstable Time* was the first piece I created that applied the m-objects library as well as the computer-mediated interaction framework that was proposed in Chapter 5. This piece is thus the test bed not only for the library that was produced as result of the theoretical study that took place, but also for a type of interaction framework that allows for a collaborative effort between a composer and dancer/choreographer to take place

in real time. This latter aspect was an aim that motivated (and justified) this study. As I mentioned in the introduction to this text, I consider my way of collaborating in dance to be interactive in essence, and feel a need to carry this interactive process of collaboration up to the performance of a dance piece. The software library produced as a result of this study — as well as the proposed framework for computer-mediated interaction — proved to be a possible way to achieve this aim.

CHAPTER VIII

CONCLUSION

In this dissertation I formulated the hypothesis that dance possesses musical qualities at the temporal level, that these qualities can be extracted in real time and provide a way for interaction between dance and music utilizing interactive computer systems. The formulated assumptions led to (a) the creation of a software library that musically interprets certain rhythms produced by the dancer while moving, (b) the proposal of a framework for musical interaction in real-time computer-mediated collaborations between dancers and musicians, and (c) the creation of a piece for interactive dance in which the software library is utilized within the proposed framework for interaction. In this *prototypical* piece, all the rhythm that animated the music was generated from the computer's interpretation of the dancer's movement. The following summary highlights the steps that were followed in order to conceive the software and to propose the approach for computer-mediated interaction that were ultimately tested in *Etude for Unstable Time*.

Summary

The Software

The software library I created was a consequence of important concepts and findings that emerged from this study. Some of them emerged from the literature review, some from experimental observation, and others from these two perspectives combined. The important theoretical findings that emerged from the literature review in chapters 2 and 3 are as follows:

1- The perception and production of musical rhythm are related to the physical characteristics of the human body;

2- Skilled actions produce rhythm and have the property of being nearly periodic. Dance could be considered a skilled action;

3- Pulse sensation and perceived meter are qualitatively different. Pulse sensation is an acoustic phenomenon and perceived meter is in general a musical cognitive task;

4- Whereas pulse has an equivalent in dance — the *beat rhythm,* in Humphrey's terms (1959) — meter does not seem to have one. The next temporal level in dance that encounters strong similarities with music is the phrase level;

5- Certain rhythms in dance movement, namely rhythms produced by the motor system, bear qualities akin to musical rhythms in their durational and accentual character.

These findings paved the way towards assuming that there may be a way of *musically* analyzing certain rhythms in dance, and to determine a dance's *musical* tempo.

The review of the currently-available video analysis software revealed that there are video analysis algorithms that can depict existing periodicities in dance movement. Concretely, the frame-differencing algorithm detects such periodicities under controlled lighting conditions and using camera in fixed position. In these conditions, this algorithm can be used to determine the *quantity of motion* (i.e. the amount of light change) that occurred between two consecutive frames in a digitized video stream. Under the more restrictive conditions that were expressed in Chapter 5 (cf. Delimitations), three important findings emerged from informal experiments I did with this video analysis signal:

1- The time-domain representation of the frame-differencing analysis signal depicts well the movement articulations and inflections in dance movement;

2- Simple periodic movement actions translate into periodicity in the time-domain representation of this signal;

3- The *musical* periodicities in dance movement, including the tempo of the movement articulations, can be extracted in real time by applying DSP techniques that are similar to recent approaches for detecting tempo in music.

In order to extract the *musical* periodicities from dance movement, including the *musical* tempo of the dance, the object **m.bandit** employs a bank of 150 second-

order recursive band-pass filters whose center frequencies range from 0.5 Hz to one-half of the sampling/frame rate. The Goertzel algorithm is applied to the output of these filters in order to compute the signal spectrum and its fundamental frequency. This approach's resemblance with similar approaches for tempo detection in music was thoroughly discussed in Chapter 6.

These findings also contributed to the elaboration of the concept of *musically processing a dance*. This concept, which I formulated in Chapter 5, consists of extracting *musical cues* from dance movement. (In this study, musical cues are rhythms in dance that bear qualities akin to musical rhythms.) The software library can musically represent certain types of movement actions in dance and thus opens a *channel of musical communication* in real-time computer-mediated collaborations between dancers and musicians.

The Proposed Framework for Interaction

In Chapter 1, I expressed two personal aims that motivated this study in interactive dance: (a) to carry my interactive process of collaboration in dance towards the performance of a piece, and (b) to improve the temporal articulation between my music and dance during performance. In Chapter 4, the analysis of the approaches taken by other composers and creators of interactive dance in translating movement into music unveiled two different ways for operating that translation: (a) by performing direct mappings of movement actions to musical parameters, and (b) by creating interaction modes that rely on some type of movement analysis that is being done in real time.

The analyses of other creators' approaches, complemented by the review of video analysis software utilized in interactive dance and motivated by my personal aims in this field, led me to formulate my approach towards analyzing *musical* rhythm in dance. This approach, which *takes the idiosyncrasies of dance into account* relies essentially on two aspects: (a) the creation of an interaction mode that relies on some type of real-time movement analysis; (b) the analysis of movement should be done non-invasively. I then proposed a framework for computer-mediated interaction in dance within which this software is utilized (cf. Chapter 5, Conceptual Considerations). The proposed framework calls for an intense participation of the musician/composer during the performance of a choreography.

The utilization of the software within the proposed framework for interaction utilizes the above-mentioned *channel of musical communication.* During the performance of a piece, the musician/composer using this software can give or take away the control of musical tempo to the dancer, change the temporal articulation of the music in synchronization with the dancer's movement, and therefore substantially alter the musical content during the performance of a piece. This can eventually trigger different responses from a dancer and lead to some improvisation with the music, creating a feedback network of communication between the musician/composer and the dancer in real time. In this respect, the piece *Etude for Unstable Time* provides an *artistic synthesis* of this study since it not only demonstrates the functionalities of the software library but also

demonstrates how the proposed framework for interaction can be applied during the creation and performance of an interactive dance work.

Guidelines for further Study

Before addressing the guidelines for further study, it is important to mention that this study developed a very specific way for extracting *musical cues* from dance movement and for utilizing this information as a means for computer-mediated communication between dancers and musicians in interactive dance performance. This was done through a *theoretical reduction to rhythm*, a necessary step in order bring to the surface the rhythmic similarities of music and dance. As I said in Chapter 1, music and dance were addressed as rhythmic art forms that produce a perceivable order in time. As a consequence of this, in order to facilitate the comparison between rhythm in music and rhythm in dance, and to better formulate the idea of *musically processing* a dance, this study ignored the spatial element in dance movement altogether.

Space is a fundamental aspect in dance movement as noted in Chapter 3. The human body is a three-dimensional entity, and all bodily movements are produced in space. Capturing the spatiality of movement in dance in order to better grasp its rhythmic quality and richness is certainly an issue to address in future studies in this area of research. However, besides suggesting that one could use two cameras placed at an angle in order to try to capture two different perspectives of movement, and apply the m-objects to both video streams, I am not in a position for suggesting other ways for capturing the three-dimensionality

of movement in dance at this point in time. Such attempt would probably require the utilization of other computer-vision techniques for analyzing the video stream — or even that a completely different theoretical approach was taken for analyzing this type of musical cues in a three-dimensional representation of dance movement. Nevertheless, realizing a three-dimensional representation of dance movement for the detection of (musical) rhythm would certainly improve the musical interaction mode as proposed by this study, an eventually lead to even more subtle ways for setting up other musical interaction modes.

The guidelines for further study will thus focus on two trends of work that might be built on the approach taken by this study. One pertains to the utilization of the m-objects in its present version, the other pertains to the improvements that can be done to the present algorithms.

Developments Utilizing the Current Version of the Software

There are five developments proposed for interactive dance performance that can be done utilizing the current version of the m-objects:

1- Porting the software to other modular programming environments;

2- Utilizing the library to extract the rhythms of more than one dancer;

3- Applying the library to a segmented representation of the human body;

4- Creating additional objects for improving the rhythmic interaction mode;

5- Controlling computer animations from dance movement.

Porting the Software to Other Modular Programming Environments

Porting this software to other modular programming environments, especially to EyesWeb or Isadora, would be a good contribution to the community of interactive dance performance. EyesWeb is the programming environment that has most systematically developed libraries that perform high-level analyses in dance movement. Isadora is a more recent programming environment that has become increasingly used.

Utilizing the Library to Extract the Rhythms of More than One Dancer

Utilizing the library to extract the rhythms of more than one dancer at the same time would involve the determination of each dancer's Kinesphere and use the frame-differencing algorithm in each Kinesphere instead of the whole image. The m-objects could then be used to track each dancer's rhythm. This could be done utilizing the software EyesWeb, which already provides objects that define the dancer's Kinesphere and could lead to interesting poly-rhythmic musical effects created by the differences in rhythm of both dancers.

Applying the m-objects to a Segmented Representation of the Human Body

Applying the m-objects to a segmented representation of the human body could help improve the accuracy of the tempo extraction from dance movement. Having a rough segmentation of the human body with different rectangles representing the head, the trunk, and the limbs would allow one to either detect the tempo of each body part selectively or to put a "perceptual weight" on the

different body parts to get a better estimation of the tempo. One of the problems in the sole utilization of frame differencing for the estimation of tempo is that the articulations of the limbs seem to provide more cues towards the estimation of the tempo than the trunk. Since the trunk occupies a bigger area in the image than the limbs, more pixels change brightness when the trunk moves. Putting a "perceptual factor" on each part of the body could help correct this problem.

Creating Additional Objects for Improving the Rhythmic Interaction Mode

This study stops short of creating a *full* interaction mode based on the rhythm extracted from dance movement. By *full* interaction mode, I mean, for example, the creation of algorithms capable of analyzing how *musically* rhythmical is the dance being performed, if there is a steady tempo in the dancer's movement, or if there is a tendency in the dancer to accelerate or decelerate the rhythm. Having such algorithms implemented would enable the users of the system to program more refined musical responses from the computer, based on these additional analysis of the rhythmic qualities of dance movement.

Controlling Computer Animations from Dance Movement

Controlling the speed of computer graphic animations from dance movement can be another interesting application for the m-objects in interactive dance performance. Since the library provides frequency-domain information about dance movement, the rhythms extracted by the m-objects (including the tempo) could be applied to control the rhythm of computer animations during the

performance of an interactive dance piece. Both animations and music can be synchronized to the dancer's movement and thus enhance the audiovisual impact in performance.

Improving/Modifying the Algorithms

The creation of other algorithms that utilize a different computational approach from the one that was proposed and applied would solidify the research of musical rhythm detection in dance movement. Taking as an example the research that has been done in the computational approaches for rhythm extraction and tempo detection in music, one realizes that are several algorithms that provide alternative ways for solving the same problem. I designed this software having in mind a framework for computer-mediated collaboration in interactive dance that calls for the active participation of the musician operating the software during performance.

Utilizing this software in a stand-alone mode (e.g. in an interactive installation) may not provide such a robust output like the one it can be obtained when this software is operated during its performance. One of the issues I pointed out in the behavior of **m.bandit** while calculating the fundamental frequency of the frame-differencing signal was the occurrence of "octave jumps" in the calculation of the fundamental frequency of the frame-differing signal (cf. Chapter 7, The Musical Tempo Control Patch for *Etude for Unstable Time)*. Whereas I tackled this issue for the utilization of this software while being

operated, no solution was created so far for circumventing this problem in a stand-alone mode of utilization.

Other modifications to the present computational approach may still be needed. I also envision a possible application for the m-objects in interactive music performance: One may want to utilize a video camera to trace the bodily movements of a conductor or a music performer in order to synchronize the generation of electronic music to bodily action in interactive music performance. In order to do so, the object that computes the spectrum of the frame-differencing signal should behave as a *real* musical beat tracker, and the information about the phase of the signal should be computed to enable the precise synchronization of bodily action to the generation of electronic music.

What have been presented above are just some possible guidelines for further work to be done in the real-time detection of rhythm in movement. Other applications for this software can still be found, especially if one uses it creatively. The code for the algorithms is provided as an appendix to this dissertation and in the accompanying DVD-ROM, and the reader is sincerely encouraged to modify or improve the present algorithms and explore new creative/innovative applications for this software.

Original Contribution

There are two aspects that make this interdisciplinary study an original contribution to the field of interactive dance. The first is the creation of a modular software library that enables the generation of musical rhythms and/or the control

of musical tempo from dance movement in real time. The second is the *musical approach* that was taken for the study of dance movement.

The creation of a modular software library that enables the generation of musical rhythm and/or musical tempo control from dance movement is perhaps the most concrete contribution this study gave to the field of interactive dance performance. None of the existing modular programming environments that are utilized in interactive dance performance, namely Max/MSP, EyesWeb, or Isadora, provides the facilities offered by this software library, and this study seems to be the first in the field that provides a viable solution for the generation musical rhythm and/or control musical tempo in real time from dance movement.[76] The fact that this library can be utilized in conjunction with other libraries available to these modular programming environments enables its users to take advantage of the facilities provided by this library and use it as an added facility to the work they have developed so far.

---

[76] To my knowledge, controlling musical tempo from dance movement in real time has been attempted once earlier, with a very limited degree of success, in MIDAS (MIC Interactive Dance System), an amusement system created by the Japanese company ATR Media Integration & Communication Research Laboratories in 1999. This system, designed for the general public, plays music and projects video imagery according to emotional features extracted from dance movement utilizing a video camera. MIDAS utilizes a computational model of time, space, and weight effort to extract the emotional features in movement, and it is in many ways similar to the effort space model of Camurri and colleagues. In its only paper available in English language about this system (ATR Media Integration & Communication Research Laboratories, 2001), ATR mentions that it tried to estimate the frequency of a dancer's movement using frame-differencing analysis without success, and got some success in calculating the speed of movement by measuring the difference of the area of the Kinesphere between consecutive frames. ATR states that in the future it will be possible to synchronize the system's beat to the dancer's frequency of movement. However, no further news about this project development was available at their website (www.mic.atr.co.jp/organization/dept3/midas.html, accessed 10/24/2004) at the time of this writing.

The *musical approach* taken to analyze movement in dance is seen as another original contribution that has proven fruitful for the work in interactive dance. By *musical approach,* I mean the way of looking for musical elements present in dance movement that was developed by this study. This type of *musical analysis* does not impose on the dancer additional musical responsibilities in performance nor does it condition the dancer's freedom of movement while performing. These two aspects concerned Siegel (the imposition of musical responsibilities on the dancer) and Coniglio (the MidiDancer imposed particular ways of moving on dancers) in mapping movement to musical parameters in interactive dance performance (cf. Chapter 4, Making Bodily Movement Musical: Approaches and Discussion). This approach involves the gathering of *musical* data non-invasively and it is up to the musician/operator of the software to make the decisions of how to use these data during performance, thereby letting the dancer remain focused on dancing.

This work idealized and created a *musical* interaction mode that relies on the extraction of *musical cues* from dance movement by *musically processing a dance*. This approach is in many ways similar to the interaction modes created by Camurri and colleagues for the extraction of *expressive cues* from movement in dance. Whereas their work utilizes computational models of KANSEI information processing and LMA for the extraction of such cues, this study applies computational models for the detection of musical rhythm in a time-domain representation of dance movement. This can be done because there is enough

theoretical evidence about the similarities between certain rhythmic realizations in dance and music's rhythmic realizations.

The idea of musically processing dance seems to provide a fertile ground for development in computer-mediated interaction between dancers and musicians. As I stated in the conclusion to Chapter 5, researching the correlations between certain types of expressive movement in dance and the expressive qualities of music to a point of implementing those theories algorithmically could lead to new ways of (re)thinking the music/dance interaction altogether. (As, for example, in the creation of systems that generate *different* music for the *same* choreography?).

Maybe it is now time to return to the reply I gave to my dancer friend as the outcome of the discussion we had, as this issue deserves a little reflection in the concluding paragraphs of this dissertation.

In fact, this study created a system that can generate different musical content for the same choreography. The musician operating the system can map the data generated by the software in different ways, either by utilizing different pitch structures for the rhythms extrapolated by the library, or by *playing tricks* to the dancer about the control over the generated rhythm or the musical tempo. The software library is also not robust enough to guarantee that all the readings are going to be the same all the time, and some errors are produced in the extrapolation of rhythmic data from movement. Therefore, producing different music for the same movement sequence is something one should expect and deal with when using the m-objects. However, besides the subversive postmodern

attitude of inverting an established paradigm in choreographical practice (in fact, during the 20th century *different* choreographies were created for the *same* music), what would be the artistic gains of doing so?

In my opinion, very few, and after the reflection and study about the use of interactive technology in reframing the compositional and choreographical practice in dance performance, I realized that this work points to more interesting ways of reframing this practice than merely subverting this established paradigm. If used wisely, this software and the proposed framework for interaction can allow composers/musicians and choreographers/dancers to work at a meta-level of creation.

This meta-level of creation allows both artists to play with pre-defined compositional and choreographical processes during the performance of a piece, utilizing an established channel of communication that also enables them to *mold* the piece's overall structure and form during performance. This allows for certain changes to occur while the piece is unfolding in time without compromising its pre-established aesthetics, overall structure, and goals. This situation is different from pure or free improvisation because improvisation is not so deterministic. This is also different from a complete musical and choreographic composition since both are fully determined before the performance of a piece.

This meta-level of creation could thus be portrayed as a level that falls between improvisation and composition but slightly above the two since one is (re)composing music and dance in real time using (pre)composed structures. In this precise sense, *Etude for Unstable Time* could be seen as a demonstration of

how this meta-level of creation can operate during the performance of an interactive dance piece. This piece is a collaborative creation that falls in that interesting level between improvisation and composition that is slightly above both, which foreshadows interesting new possibilities for computer-mediated collaborations in dance utilizing electronic music.

BIBLIOGRAPHY

ART Media Intergration & Communication Research Laboratories. (2001).*MIDAS : MIC Interactive Dance System*. Retrieved October 24, 2004, from http://www.mic.atr.co.jp/organization/dept3/papers/midas/Midas.html

Banes, S. (1994). Dancing [with/to/before/on/in/over/after/against/away from/without] the music: Vicissitudes of collaboration in American postmodern choreography. In *Writing dancing in the age of postmodernism* (pp. 310-326). Hannover, NH: Wesleyan University Press.

Banks, K. (2002). *The Goertzel Algorithm*. Retrieved March 8, 2005, from http://www.embedded.com/showArticle.jhtml?articleID=9900722

Bartenieff, I., & Lewis, D. (1980). *Body movement: Coping with the environment*. Amsterdam: Gordon and Breach Publishers.

Beardsley, M. (1982). What is going on in a dance? *Dance Research Journal, 15*(1), 31-36.

Beek, P., Peper, C. L., & Daffertshofer, A. (2000). Timekeepers versus nonlinear oscillators: How the approaches differ. In P. Desain & L. Windsor (Eds.), *Rhythm perception and production* (pp. 9-33). Lisse, The Netherlands: Swets & Zeitlinger.

Berg, P. (2001). *Writing Max objects with Code Warrior: A simple introduction*. Unpublished manuscript.

Blahut, R. (1985). *Fast algorithms for digital signal processing*. Reading, MA: Addison-Wesley Publishing Company.

Bongers, B. (2000). Physical interfaces in the electronic arts: Interaction theory and interfacing techniques for real-time performance. In M. Wanderley & M. Battier (Eds.), *Trends in gestural control of music* (pp. 41-70) [CD-ROM]. Paris: IRCAM.

Bradshaw, J. (1997). An introduction to software agents. In J. Bradshaw (Ed.), *Software Agents* (pp. 3-48). Cambridge, MA: MIT Press.

Camurri, A., & Coglio, A. (1998). An architecture for emotional agents. *IEEE Multimedia*(Oct.-Dec. 1998), 24-33. Retrieved October 11, 2004, from http://www.infomus.dist.unige.it/eywindex.html

Camurri, A., Coglio, A., Coletta, P., & Massucco, C. (1997). An architecture for multimodal environment agents. *Proceedings from the Italian Assoc. for Musical Informatics (AIMI) International Workshop on Kansei — The Techonology of Emotion*, 48-53.

Camurri, A., Hashimoto, S., Ricchetti, M., Trocca, R., Suzuki, K., & Volpe, G. (2000). EyesWeb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal, 24*(1), 57-69.

Camurri, A., & Trocca, R. (2000). Movement and gesture in intelligent interactive music systems. In M. Wanderley & M. Battier (Eds.), *Trends in gestural control of music* (pp. 95-110) [CD-ROM]. Paris: IRCAM.

Cemgil, A. T., Kappen, H. J., Desain, P., & Honing, H. (2000). On tempo tracking: Tempogram representation and kalman filtering. *Proceedings of the International Computer Music Conference*, 352-355.

Charleroi/Dances. (1996). *Moving target* [Choreography].

Clarke, E. (1987). Categorical rhythm perception: An ecological perspective. In A. Gabrielsson (Ed.), *Action and perception in music* (pp. 19-34). Stockholm: Royal Swedish Academy of Music.

Clarke, E. (1989). The perception of expressive timing in music. *Psychological Research, 51*, 2-9.

Clarke, E. (1999). Rhythm and timing in music. In D. Deutsch (Ed.), *The psychology of music* (2nd ed.). Orlando, FL; London: Academic Press.

Clynes, M., & Walker, J. (1982). Neurobiologic functions of rhythm, time, and pulse in music. In M. Clynes (Ed.), *Music, mind and brain: The neuropsychology of music* (pp. 171-216). New York: Plenum.

Coniglio, M. (2002a). Isadora (Version 1.0) [Computer software and manual]. Retrieved October 20, 2004, from http://www.troikatronix.com/izzy-download.html

Coniglio, M. (2002b). *Isadora "almost out of beta": tracing the development of a new software tool for artists*. Retrieved October 12, 2004, from http://www.sdela.dds.nl/sfd/isadora.html

Coniglio, M., & Stoppiello, D. (1990). *Tactile diaries* [Interactive Dance Work].

Coniglio, M., & Stoppiello, D. (1994). *In Plane* [Interactive Dance Work].

Coniglio, M., & Stoppiello, D. (2001). *Reine Rien* [Interactive Dance Work].

Coniglio, M., & Subotnik, M. (1989). Interactor LPT [Computer Software].

Cycling '74. (2001). Max/MSP (Version 4.1) [Computer Software]. San Francisco: Cycling '74. Retrieved October 3, 2004, from http://cycling74.com

Dell, C. (1977). *A primer for movement description: Using Effort-Shape and supplementary concepts* (Revised ed.). New York: Dance Notation Bureau Press.

DIST. (1999). EyesWeb [Computer Software and manual]. Retrieved Novemeber 10, 2004, from http://www.infomus.dist.unige.it/eywindex.html

Drake, C., Penel, A., & Bigand, E. (2000). Why musicians tap slower than non-musicians. In P. Desain & L. Windsor (Eds.), *Rhythm perception and production* (pp. 245-248). Lisse, The Netherlands: Swets & Zeitlinger.

Dura, M. (1998). *The kinesthetic dimension of the music listening experience*. Unpublished Dissertation, Northwestern University.

Eck, D., Glasser, M., & Port, R. (2000). Dynamics and embodiment in beat induction. In P. Desain & L. Windsor (Eds.), *Rhythm perception and production* (pp. 157-170). Lisse, the Netherlands: Swets & Zeitlinger.

Fisher, G. (1995). *Phrase Procedure in the Music of Seymour Shifrin*. Unpublished Dissertation, Columbia University.

Fisher, G., & Lochhead, J. (2002). Analyzing from the Body. *Theory and Practice, 27*, 37-67.

Fraisse, P. (1956). *Les structures rhythmiques*. Louvain, France: Publications Universitaires de Louvain.

Fraisse, P. (1974). *Psychologie du rythme*. Paris: PUF.

Fraisse, P. (1982). Rhythm and tempo. In D. Deutsch (Ed.), *The psychology of music* (pp. 149-180). Orlando, FL; London: Academic Press.

Gabrielsson, A. (1995). Expressive intention and performance. In R. Steinberg (Ed.), *Music and the mind machine: The psychophysiology and psychopathology of the sense of music* (pp. 35-47). Berlin, New York: Springer Verlag.

Goebel. (1988). Man-machine interaction. *Proceedings of the International Computer Music Conference*, 41-48.

Goertzel, G. (1958). An algorithm for the evaluation of finite trigonometric series. *American Mathematics Monthly, 65*, 34-35.

Guedes, C. (2003). Controlling musical tempo from dance movement: A possible approach. *Proceedings of the International Computer Music Conference*, 453-457.

Hanna, J. (1979). *To dance is human: A theory of non-verbal communication*. Austin, TX: University of Texas Press.

Hashimoto, S. (1997). KANSEI as the third target of information processing and related topics in Japan. *Proceedings of the International Workshop on Kansei - Technology of Emotion*, 101-104.

Hodgins, P. (1992). *Relationships between score and choreography in twentieth-century dance: Music, movement and metaphor*. London: Mellen.

Humphrey, D. (1959). *The art of making dances*. New York: Grove Press Inc.

Igloo. (2000). *Viking shoppers* [Interactive Dance Work].

Igloo. (2001). *Winterspace* [Interactive Dance Work].

Iyer, V. (1998). *Microstructures of feel, macrostructures of sound: Embodied cognition in West African and African-American musics*. Unpublished Dissertation, University of California, Berkeley.

Kozel, S., & Woolford, K. (1999). *Contours* [Interactive Dance Work].

Krueger, M. (2001). Responsive environments. In R. Packer & K. Jordan (Eds.), *Multimedia: From Wagner to virtual reality* (pp. 104-120). London: Norton.(Reprinted from *American Federation of Information Processing Societies 46*, pp. 423-429, 1977)

Krumhansl, C., & Schenck, D. (1997). Can dance reflect the structural and expressive qualities of music?: A perceptual experiment on Balanchine's choreography of Mozart's Divertimento No.15. *Musicae Scientiae, 1*(1), 63-85.

Large, E., & Kolen, J. (1994). Resonance and the perception of musical meter. *Connection Science, 6*(1), 177-208. Retrieved November 5, 2004, from http://www.cis.upenn.edu/~large/publications.html

Lerdahl, F., & Jackendoff, R. (1983). *A generative theory of tonal music*. Cambridge, MA: MIT Press.

London, J. (2001). Rhythm. In *The new Grove dictionary of music and musicians* (2nd ed., Vol. 21, pp. 277-309). London: Macmillan.

Longuet-Higgins, C. L., & Lee, C. (1982). The perception of musical rhythms. *Perception, 11*, 115-128.

Lovell, R. (2002). Video based sensing in interactive performance spaces. In S. Dinkla & M. Leeker (Eds.), *Dance and technology* (pp. 196-208). Berlin: Alexander Verlag.

Machover, T., & Chung, J. (1989). HyperInstruments: Musically intelligent and interactive performance and creativity systems. *Proceedings of the International Computer Music Conference*, 186-190.

Martin, J. G. (1972). Rhythmic (hierarchic) versus serial structure in speech and other behaviour. *Psychological Review, 79*, 487-509.

Michon, J. (1985). The compleat time experiencer. In J. Michon (Ed.), *Time, Mind, and Behaviour* (pp. 20-52). Berlin: Springer Verlag.

Mulder, A. (1994). *Human movement tracking*. Retrieved Oct. 11, 2004, from http://www.cs.sfu.ca/~amulder/personal/vmi/HMTT.pub.html#ii

Ng, K., Popat, K., Ong, B., Stefani, E., & Cooper, D. (2000). Trans-domain mapping; A real-time system for motion-acquisition and musical mapping. *Proceedings from the International Computer Music Conference*. Retrieved November 1, 2004, from http://www.satorimedia.com/hands_on/icmc2.htm

O'Brien, K. (1999). *Factors influencing the perception of rhythm in music*. Unpublished Dissertation, University of South Carolina.

Openheim, A., & Schafer, R. (1975). *Digital signal processing*. Englewood Cliffs, NJ: Prentice-Hall.

Palindrome Intermedia Performance Group. (n.d.). *Eyecon support and download page*. Retrieved October 11, 2004, from http://eyecon.palindrome.de/

Palindrome Intermedia Performance Group. (n.d.). *Palindrome performance systems*. Retrieved October 11, 2004, from http://www.palindrome.de/pps.htm

Palindrome Intermedia Performance Group. (n.d.). *Repertory*. Retrieved October 11, 2004, from http://www.palindrome.de/rep.htm

Parncutt, R. (1987). The perception of pulse in musical rhythm. In A. Gabrielsson (Ed.), *Action and Perception in Rhtyhm and Music* (pp. 127-138). Stockholm: Royal Swedish Academy of Music.

Parncutt, R. (1994a). A perceptual model of pulse salience and metrical accent in musical rhythm. *Music Perception, 11*, 409-464.

Parncutt, R. (1994b). A model of beat Induction accounting for the perceptual ambiguity by continuously variable parameters. *Proceedings of the International Computer Music Conference*, 83-84.

Povall, R. (1998). Technology is with us. *Dance Research Journal, 30*(1), 1-4.

Povall, R. (2000). Making emotional spaces in The Secret Project: Building emotional interactive spaces. In R. Ascott (Ed.), *Art, technology, consciousness mind@large* (pp. 64-68). Portland, OR: Intellect Books.

Povel, D.-J. (1985). Time rhythms and tension: In search of the determinants of rhythmicity. In J. Michon & J. L. Jackson (Eds.), *Time, mind and behaviour* (pp. 214-225). Berlin: Springer-Verlag.

Povel, D.-J., & Essens, P. (1985). The perception of temporal patterns. *Music Perception, 2*, 411-440.

Roads, C. (1996). *The computer music tutorial*. Cambridge, MA: MIT Press.

Rokeby, D. (1983). *Very nervous system* [Interactive Installation].

Rokeby, D. (2003). Soft VNS 2 (Version 2.14) [Computer Software and manual. Max library]. Retrieved November 9, 2004, from http://homepage.mac.com/WebObjects/FileSharing.woa/wa/default?user=davidrokeby&templatefn=FileSharing3.html&xmlfn=TKDocument.3.xml&sitefn=RootSite.xml&aff=consumer&cty=US&lang=en

Rowe, R. (1993). *Interactive music systems: Machine listening and composing*. Cambridge, MA: MIT Press.

Rowe, R. (2001). *Machine Musicianship*. Cambridge, MA: MIT Press.

Royce, A. (1984). *Movement and meaning: Creativity and interpretation in ballet and mime*. Bloomington: Indiana UP.

Scheirer, E. (1998). Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America, 103*(1), 588-601. Retrieved November 4, 2004, from http://web.media.mit.edu/~eds/papers.html#journ

Seashore, C. (1938). *The psychology of music*. New York: McGraw-Hill.

Siegel, W., & Brolin-Tani, M. (1998). *Sisters* [Interactive Dance Work].

Siegel, W., & Jacobsen, J. (1998). The challenges of interactive dance: Overview and case study. *Computer Music Journal, 22*(4), 29-43.

Siegel, W., & Saunders, H. (1997). *Movement study* [Interactive Dance Work].

Singer, E. (2001). Cyclops (Version 1.0) [Computer software and manual. Max external object]. San Francisco: Cycling '74. Retrieved November 10, 2004, from http://www.cycling74.com/products/cyclops.html

Sirin, S. (2004). *A DSP algorithm for frequency analysis*. Retrieved March 8, 2005, from http://www.embedded.com/showArticle.jhtml?articleID=17301593

Sparshott, F. (1995). *A measured pace: Toward a philosophical understanding of the arts of dance*. Toronto, Canada: University of Toronto Press.

STEIM. (1995). BigEye (Version 1.1.4) [Computer software and manual]. Amsterdam: STEIM. Retrieved October 11, 2004, from http://www.steim.org/steim/products.html

Tekalp, M. (1995). *Digital video processing*. New Jersey: Prentice-Hall.

Tempelaars, S. (1996). *Signal processing, speech and music*. Lisse, The Netherlands: Swets & Zeitlinger.

The Royal Academy of Music Aarhus — DIEM. (n.d.). *Movement study*. Retrieved October 11, 2004, from http://www.musik-kons.dk/english/diem/digdance/movement.php

The Royal Academy of Music Aarhus — DIEM. (n.d.). *Product description*. Retrieved October 11, 2004, from http://www.diem.dk/english/diem/digdance/prod.php

The Royal Academy of Music Aarhus — DIEM. (n.d.). *Sisters*, from http://www.diem.dk/english/diem/digdance/sisters.php

Todd, N. P. (1993). Vestibular feedback in musical performance: Response to *Somatosensatory Feedback in Musical Performance* (Ed. Sundberg and Verillo). *Music Perception, 10*, 379-382.

Toiviainen, P. (1998). An interactive MIDI accompanist. *Computer Music Journal, 22*(4), 63-75.

Troika Ranch. (n.d.). *Media/Technology | MidiDancer*. Retrieved October 11, 2004, from http://www.troikaranch.org/mididancer.html

Troika Ranch. (n.d.). *Works | In plane*. Retrieved October 11, 2004, from http://www.troikaranch.org/wrk/inplane.html

Troika Ranch. (n.d.). *Works | Reine rien*. Retrieved October 11, 2004, from http://www.troikaranch.org/wrk/reinerien.html

Troika Ranch. (n.d.). *Works | Tactile diaries*. Retrieved October 11, 2004, from http://www.troikaranch.org/wrk/tactile.html

Wing, A. M., & Kristofferson, A. B. (1973). Response delays and the timing of discrete motor responses.*Perception and Psychophysics, 14*, 5-12.

Winkler, T. (1995). Making motion musical: Gesture mapping strategies for interactive computer music. *Proceedings of the International Computer Music Conference*, 261-264.

Winkler, T. (1997). Creating interactive dance with the Very Nervous System. *Proceedings of the 1997 Connecticut College Symposium on Arts and Technology*. Retrieved October 12, 2004, from http://www.brown.edu/Departments/Music/faculty/winkler/papers/index.html

Winkler, T. (1998). *Composing interactive music: Techniques and ideas using Max*. Cambridge, MA: MIT Press.

Winkler, T. (2002). Fusing movement, sound, and video in Falling Up, an interactive dance/theatre production. *Conference on New Instruments for Musical Expression*. Retrieved October 12, 2004, from http://www.brown.edu/Departments/Music/faculty/winkler/papers/index.html

Winkler, T., & Ferrero, W. (1997). *Dark around the edges* [Interactive Dance Work].

Woolford, K. (2001). *Debugging Max externals using CodeWarrior*. Unpublished manuscript.

Yeston, M. (1976). *The stratification of musical rhythm.* New Haven, CT: Yale University Press.

Zicarelli, D. (2001). *Writing external objects for Max 4.0 and MSP 2.0*. Retrieved November 8, 2004, from http://www.cycling74.com/products/dldocold.html

Zicarelli, D., Clayton, J., & Sussman, R. (2003). *Writing external objects for Max and MSP 4.3*. Retrieved October 31, 2004, from http://www.cycling74.com/products/dldocold.html

Zicarelli, D., Taylor, G., Schabtach, A., Clayton, J. K., jhno, & Dudas, R. (2001). *Max reference*. Retrieved October 11, 2004, from http://www.cycling74.com/products/dlmaxmspprev.html

APPENDIX A

CONSENT FORM

You have been invited to take part in a study whose purpose is to test a computer program that allows dancers to control certain musical processes in real-time through bodily motion. This study will be conducted by Carlos Guedes, from the Department of Music and Performing Arts at the School of Education, New York University, as part of his doctoral dissertation. His faculty sponsor is Dr. George Fisher, who can be contacted at NYU School of Education, Department of Music and Performing Arts, phone number (212) 998-5410.

If you agree to be in this study you will be asked to do the following:

1- To improvise two or three simple movement sequences, in a place you find suitable for dance performance (for example, a dance studio), with the sole presence of Carlos Guedes besides yourself. These movement sequences will be captured by one of the following devices: (1) video cameras; (2) sensors placed in your body in a way that is neither painful or encumbering to you; (3) sensors placed unobtrusively in the space you will be moving on. You will be requested to repeat a certain number of times the same movement sequence in order to test the reliability in the computer program's response

2- To respond verbally to some informal questions about the tests you just performed: for example, how was your personal experience in using the program, if you enjoyed being in control certain musical aspects, and what do you feel it could be improved or implemented in the future in the computer program

Participation in this study will involve about one hour and thirty minutes of your time: 30 minutes to warm-up, setup and fine-tune the equipment, 45 minutes for testing, 15 minutes of informal conversation.

There are no apparent physical risks to be expected from this test, since you will be asked to perform movement sequences within your full range of capabilities. If, for any reason you find that any of these tasks or parts of the test are difficult or upsetting to perform, you should discuss that with Carlos Guedes at any point of the testing procedure.

Although you will receive no direct benefits, this research may help Carlos Guedes better understand the performance of the computer program being created.

Federal regulations require that all subjects be informed of the availability of medical treatment or financial compensation in the event of physical injury resulting from participation in the research. New York University can not provide either medical treatment or financial compensation for any physical injury resulting from your participation in this research project. Inquiries regarding this policy may be made to the principal investigator or, alternatively, the New York University Office of Sponsored Programs (212-998-2121).

Carlos Guedes has explained this study to you and answered your questions. If you have additional questions or wish to report a research-related problem, you may contact the him at (212) 614-0151, or by e-mail at carlos.guedes@nyu.edu.

For questions about your rights as a research participant, you may contact the University Committee on Activities Involving Human Subjects, Office of Sponsored Programs New York University, (212) 998-2121.

Participation in this study is voluntary. You may refuse to participate or withdraw at any time without penalty.

Confidentiality about your participation in these tests can be strictly maintained. If you wish that your participation in the study remains confidential, Carlos Guedes will ensure that all elements that can possibly identify you (such as name, sex, age, or social security number) will not be mentioned in the dissertation. Moreover, the records pertaining to your participation will be kept in a safe place under Carlos Guedes's direct responsibility, providing no access to other people. The records will be destroyed after a period of three years. If you wish that your participation remains confidential, you should check the box below the "Agreement to Participate."

Your tests will be videotaped. You may review these tapes and request that all or any portion of the tapes be destroyed immediately upon completion of the testing session. You should also inform Carlos Guedes if all or any portion of the tapes may be used as examples to illustrate the performance of the computer program in the dissertation.

Attribution Statement

___ Yes, I grant Carlos Guedes permission to use my name and attribute quotes to me directly in any written research from these tests.

___ Yes, I grant Carlos Guedes permission to use my videotape in connection with this research.

___ No, I would prefer that Carlos Guedes identifies me by pseudonym in any written research from these tests.

___ No, I would prefer that my videotape not be used in connection with this research.

You have received a copy of this consent document to keep.

Agreement to Participate

_____ _____

Subject's Signature Date

APPENDIX B

COMPLETE LIBRARY DESCRIPTION

Analysis Objects

m.bandit

m.bandit computes the fundamental frequency of a time-varying representation of the frame-differencing signal extracted from a digitized video stream. The object also outputs the amplitude and frequency of the initial four harmonics of that frequency, and outputs lists for graphically displaying the signal spectrum and the spectrum of the four harmonics of the fundamental frequency. **m.bandit** also allows the user to specify the object's sensitivity to movement. The object prints in the Max window the message 'moving' when it considers that there is movement, otherwise, it will print 'idle' (idle state). The rightmost outlet of the object outputs the value zero if in 'idle state,' or one otherwise ('moving state').

Input:

int	An integer value corresponding to the overall change brightness between consecutive frames in the stream.
reset	Resets the object.
bang	Prints in the Max window the sampling rate at which the object is

	operating, if it is removing the DC offset from the signal (dc_out), and the values for the messages 'idle_int' and 'idle_thresh.'
dc_out	The word 'dc_out' followed by integer 1 or zero turns on or off the removal of the DC offset (default: 'dc_out = 1').
idle_int	The word 'idle_int' followed by an integer sets the time interval (in ms) after which the object declares 'idle state' when the brightness difference has been under a certain threshold value (default: idle_int = 3000 ms).
idle_thresh	The word 'idle_thresh' followed by an integer sets the value representing the brightness-difference threshold under which the object interprets the stream as "not moving" (idle state). Default: idle_thresh = 200.
showharms	The word 'showharms' followed by 1 or zero tells the object to send out or not the spectrum of the harmonics for display (default: showharms = 1).
showspec	The word 'showspec' followed by 1 or zero tells the object to send out or not the spectrum of the harmonics for display (default: showspec = 1).
sr	The word 'sr' followed by an integer sets the sampling rate (the number of frames per second) at which the object will operate (default: sr = 30).

Arguments:

int	Sets sampling rate (frames per second).

Output:

float	On first outlet (leftmost): the fundamental frequency of the signal.
int	On rightmost outlet: zero or 1 depending if the object considers that there is movement (zero if it is idle, and the word 'idle' is printed in the Max window; 1 if there is movement, and the word 'moving' is printed in the Max window).
list	On second outlet: frequency and amplitude of the four most prominent harmonics in the form *f1 a1 f2 a2 f3 a3 f4 a4* On the third outlet: The spectrum of the signal (to connect to a **MultiSlider Object)**. On the fourth outlet: The spectrum of the four most prominent harmonics (to connect to a **MultiSlider Object**).

m.peak

m.peak outputs a 'bang' when a peak occurred in the brightness difference value. The user specifies what is value in the overall brightness change that the object should consider to be a peak in the signal. This is in general a big number. This object is especially useful to use when one wants to trigger events when heavily accented movement actions occur.

Input:

int	In left inlet: an integer value corresponding to the overall change brightness between consecutive frames in the stream.
	In right inlet: the threshold value to consider as a peak.

Arguments:

int	The threshold value to consider as a peak.

Output:

bang	Bang message when a peak occurred.

m.weights

m.weights collects the output of the frame-differencing signal fundamental frequency from **m.bandit** converted to milliseconds and outputs the one-hundred millisecond value that was most output by **m.bandit** in the past 60 frames.

Input:

int	Fundamental frequency of the frame-differencing signal converted to milliseconds as output by **m.bandit**.
reset	The word ‘reset’ clears all the values received in the past 60 frames.

Arguments:

None

Output:

int	In first outlet: the one-hundred millisecond value that was most output by **m.bandit**.
list	In second outlet: a list of all the one-hundred millisecond values that were output by **m.bandit** in the past 60 frames (connect to **MultiSlider** in order to visualize).

Processing Objects

m.clock

Adaptive clock that takes as input the fundamental frequency output of **m.bandit** converted to milliseconds and enables the control of the tempo of a musical sequence. The clock can be initialized with a previous value (tempo of the sequence) and a margin for adaptation (delta). If the fundamental frequency output by **m.bandit** falls within the margin for adaptation, the clock outputs a that new tempo value for control. If the clock is not initialized with an initial tempo value it will consider as the initial tempo value the first value it receives between 300 and 1500 ms.

Input:

int	In first inlet: beat candidate (in general, the fundamental frequency calculated by **m.bandit** converted to milliseconds). In second inlet: **m.bandit**'s 'idle state' (zero or 1)
accept	The word 'accept' followed by an integer (between 300 and 1500) initializes the tempo of the clock to that value and the message 'accepted x milliseconds' is printed in the Max window.
reset	The word 'reset' resets the object (i.e. clears the current tempo estimate).
delta	The word 'delta' followed by a floating-point value between zero and 1 sets the allowable margin for tempo adaptation. E.g. 'delta 0.2' sets a margin for adaptation that is 20% of the current tempo estimate (default: delta = 0.15).

Arguments:

float	The value of delta.

Output:

int	In first outlet: the estimated tempo value. In second outlet: zero or 1 corresponding to the 'idle state' message if **m.bandit**'s rightmost outlet is connected to the second inlet of the object.

m.lp

Simple IIR first-order low-pass filter that can be used to smooth the frame-differencing signal (see Figure 23).

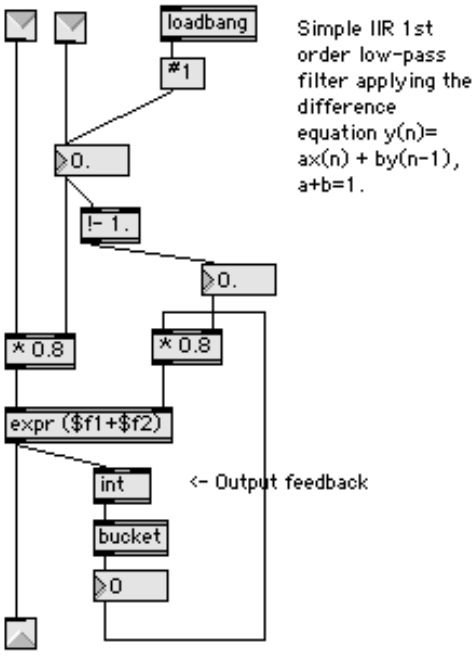


Figure 23. Internal structure of **m.lp**.

Input:

int	In first inlet. The brightness difference value.
float	In second inlet: feedback coefficient (between zero and 1).

Arguments:

float	The feedback coefficient for the filter.

Output

float	The calculated output of the filter.

Extras

m.sample

m.sample samples the output of the brightness-difference sum at a constant rate, in order to improve the accuracy of the calculations performed by **m.bandit**.

Input:

int	An integer value corresponding to the overall change brightness between consecutive frames in the stream.
sr	The word 'sr' followed by an integer sets the sampling rate in fps (default: sr = 30).
bang	A bang message causes the object to start sampling and the word 'sampling…' is printed in the Max window. If a second bang is received the object will stop sampling.
start	The word 'start' causes the object to start sampling and the word 'sampling…' is printed in the Max window.
stop	The word 'stop' causes the object to stop sampling.

Arguments:

int	The object's sampling rate.

Output:

int	The values received in the inlet at a steady rate.

APPENDIX C

CODE

m.bandit.c

```
/*
   m.bandit © Carlos Guedes 2002-2005
*/


#include "ext.h"    // contains basic function macros and
                    // type definitions for Max

#include <math.h>
#include <stdlib.h>
#include "testsoundfile2.h"
/*definitions*/
#define PI 3.141592653589
#define TWOPI 6.2831853072
#define NUMFILTERS 150
#define SUMS 88

typedef struct filter  //filter sructure
{
   double y[3];     //output (y(n)), previous (y(n-1)),
                    //second previous (y(n-2))
   double f_cf;     //center frequency
   double f_bw;     //bandwidth
   double f_c;      //coefficient C
   double f_r;      //coefficient R
   double f_cr;     //coefficient CR
   double f_two_cr; //coefficient 2CR
   double f_rsqrd;     //coefficent R^2
   double f_out;       //output
   double f_real;      //real part
   double f_imag;      //imaginary part
   double f_rsin;      //RsinPhi
} Filter;

/* bandit object data structure */

typedef struct bandit
{
```

```
    struct object m_ob;     // object definition
    long a_in;              // input to the filter
    long a_inMinOne;        // input with 1 sample delay
    long dc_out;            // dc offset removal
    short dc_onoff;         // dc offset on/off switch
    short spec_onoff;       // output spectrum switch
    short harm_onoff;       // output first four harmonics switch
    Atom output[NUMFILTERS];      // filter output
    Atom displayharms[NUMFILTERS];// list for harmonic spectrum display
    Atom harmons[8];              // list of frequencies and amplitudes
                                  // of the 4 most prominent harmonics
    double sumsig [SUMS];         // store correlation sums
    double harmonics[NUMFILTERS]; // store harmonics

    void *m_qelem;

    Filter bfilter[NUMFILTERS];   // Array of bandpass filters

    long a_sr;                    // sampling rate
    short flagmove;               // post flags so that it doesn't
                                  // repeat the same message forever
    short flagidle;
    short idle;                   // idle state
    long idleinterval;            // time interval to declare idle state
    long idlethresh;              // threshold value
    long idletime;                // time idle
    long prevtime;                //
    long currtime;                // logical time checking variables
    long deltatime;               //

    /* outlets */
    void *m_out5;
    void *m_out4;
    void *m_out3;
    void *m_out2;
    void *m_out;

} Bandit;

/* Function Prototypes*/
void bandit_bang(Bandit *x);
void bandit_int(Bandit *x,long n);
void bandit_setSR(Bandit *x, long n);
Boolean bandit_dc_out (Bandit *x, short n);
Boolean bandit_showspec (Bandit *x, short n);
Boolean bandit_showharms (Bandit *x, short n);
void bandit_getidleinterval (Bandit *x, long n);
void bandit_getidlethresh (Bandit *x, long n);
void bandit_reset(Bandit *x);
void bandit_assist(Bandit *x,void *b,long m,long a,char *s);
void bandit_calculate (Bandit *x);
void bandit_out (Bandit *x);
```

```
void bandit_checktime (Bandit *x);
void bandit_getidletime (Bandit *x);
void bandit_resetsum (Bandit *x);
void bandit_init (Bandit *x);
void *bandit_new(Symbol *s, short ac, Atom *av);
void bandit_free (Bandit *x);

void *class;     // global variable that contains the bandit class

void main(void)
{
   /* set up the class */
   setup((Messlist **)&class, (method)bandit_new,(method)bandit_free,
(short)sizeof(Bandit), 0L,A_GIMME,  0);

   // bind method to the bang message
   addbang((method)bandit_bang);
   // bind method for int input in the leftmost inlet
   addint((method)bandit_int);
   // bind method to message "sr" (set sampling rate)
   addmess((method)bandit_setSR,"sr", A_LONG, 0);
   // bind method to message "dc_out" (dc_out removal)
   addmess((method)bandit_dc_out, "dc_out", A_LONG, 0);
   // bind method to message "showspec"
   addmess((method)bandit_showspec, "showspec", A_LONG, 0);
   // bind method to message "showharms"
   addmess((method)bandit_showharms, "showharms", A_LONG, 0);
   // bind method to message "idle_int" (set the idle interval in ms)
   addmess((method)bandit_getidleinterval, "idle_int", A_LONG, 0);
   // bind method to message "idle_thresh"
   // (set pixel threshold value to declare idle state)
   addmess((method)bandit_getidlethresh, "idle_thresh", A_LONG, 0);
   // bind method for reset
   addmess((method)bandit_reset, "reset", 0);
   // bind method for assistance
   addmess((method)bandit_assist,"assist",A_CANT,0);
   // add to New Object List
   finder_addclass("Movement Analysis","m.bandit");

   // copy assistance string resource to Max Temp file
   rescopy('STR#',8003);
}

/*methods*/

// method for bang message
void bandit_bang(Bandit *x)
{
   post ("sampling rate %ld", x->a_sr);
   post ("dc_out %ld", x->dc_out);
   post ("idle int %ld", x->idleinterval);
```

```
    post ("idle thresh %ld", x->idlethresh);
}

// method for int in left inlet
void bandit_int(Bandit *x,long n)
{
    x->a_in=n;
    bandit_checktime(x);

    // if |a_in| below a certain threshold
    // calculate idletime
    if(abs(x->a_in)<=x->idlethresh)      {
        bandit_getidletime(x);
    // if idletime passes a certain value
    // declare idlestate
    // post in Max window idle state
    // turn on flag for idle state so that it does not repeat
    // reset moving state flag
        if (x->idletime>x->idleinterval)
            x->idle=1;
            if (!x->flagidle && x->idle){
                post ("idle");
                x->flagidle=1;
                x->flagmove=0;
                outlet_int(x->m_out5, x->flagmove);
                }
            }
    // else reset idle time and declare not idle
    // post in Max window moving state
    // turn on flag so that it does not repeat forever
    // reset idle state flag
    else{
        x->idle=0;
        x->idletime=0;
        if (!x->flagmove) {
            post ("moving");
            x->flagmove=1;
            x->flagidle=0;
            outlet_int(x->m_out5, x->flagmove);
            }
        }

    // grab input and calculate bank filter output
    if (!x->dc_onoff){
        bandit_out(x);
        }

    else if (x->dc_onoff){
        x->dc_out= n-x->a_inMinOne;
        x->a_in=x->dc_out;
        bandit_out(x);
        x->a_inMinOne=n;
```

```
      }
}

// set sampling rate
void bandit_setSR(Bandit *x, long n)
{
   x->a_sr=n;
   bandit_calculate (x);
}

// method for removing dc offset
Boolean bandit_dc_out (Bandit *x, short n)
{
   x->dc_onoff=n;
   post ("dc_out %ld", x->dc_onoff);
   return x->dc_onoff;
}

// method for displaying the spectrum
Boolean bandit_showspec (Bandit *x, short n)
{
   x->spec_onoff=n;
   return x->spec_onoff;
}

// method for displaying the harmonics
Boolean bandit_showharms (Bandit *x, short n)
{
   x->harm_onoff=n;
   return x->harm_onoff;
}

// method for defining the idle interval in ms
void bandit_getidleinterval (Bandit *x, long n)
{
   x->idleinterval=n;
   post ("idle_interval %ld", x->idleinterval);
}

// method for defining pixel threshold value to declare the idle state
void bandit_getidlethresh (Bandit *x, long n)
{
   x->idlethresh=n;
   post ("idle_trhesh %ld", x->idlethresh);
}

// method for resetting the object
void bandit_reset(Bandit *x)
{
   int i, j;

   qelem_set (x->m_qelem);
```

```
    for (i=0; i<NUMFILTERS; i++){
        x->bfilter[i].f_out=0;
        for (j=0;j<3;j++)
               x->bfilter[i].y[j]=0;
               }

    for (i=0; i<SUMS; i++)
        x->sumsig[i]=0.0;

    x->a_in=0;
    qelem_unset(x->m_qelem);

post ("reset!");
bandit_out(x);
}

// assist method
void bandit_assist(Bandit *x,void *b,long m,long a,char *s)
{
    assist_string(8003,m,a,1,2,s);
}


// calculate all coefficients
void bandit_calculate (Bandit *x)
{
    int i, j;

        for(i=0; i<NUMFILTERS; i++) {
            // calculate center frequencies in log scale
            // starting at 0.5 Hz until sr/2 with 150 steps
            x->bfilter[i].f_cf=0.5*pow(exp(log((x-> a_sr*0.5)/0.5)/
NUMFILTERS),i);
            // bandwidth
            x->bfilter[i].f_bw=x->bfilter[i].f_cf*0.1;
            // coefficient C
            x->bfilter[i].f_c= cos(TWOPI*x->bfilter[i].f_cf/x->a_sr);
            // coefficient R
            x->bfilter[i].f_r=exp(-PI*x->bfilter[i].f_bw/x->a_sr);
            // coefficient CR
            x->bfilter[i].f_cr=x->bfilter[i].f_c*x->bfilter[i].f_r;
            // coefficient 2CR
            x->bfilter[i].f_two_cr=2*x->bfilter[i].f_cr;
            // coefficient R^2
            x->bfilter[i].f_rsqrd=pow(x->bfilter[i].f_r,2);
            // coefficient Rsin
            x->bfilter[i].f_rsin=x->bfilter[i].f_r*sin(TWOPI*x->
bfilter[i].f_cf/x->a_sr);
            //reset filter outputs
            for (j=0; j<3; j++){
               x->bfilter[i].y[j]=0;
```

```
            }
         }
}

// Calculate magnitude spectrum, most prominent frequency, and first four
// harmonics
void bandit_out (Bandit *x)
{
   int i, j, index, count;
   short prevLock;
   long max;
   long sum;
   float tmp[8]={0,0,0,0,0,0,0,0};
   float harmExtract;
   qelem_set (x->m_qelem);
   index=0;

   //calculate output for each filter
   for (i=0; i<NUMFILTERS; i++) {
      // IIR Bandpass filter output calculation using the
      // formula Y(n)= X(n) + 2CRY(n-1) - R^2Y(n-2)
      x->bfilter[i].y[0] =x->a_in + x->bfilter[i].f_two_cr*x->
bfilter[i].y[1] - x->bfilter[i].f_rsqrd*x->bfilter[i].y[2];
      //real part= Y(n)-Rcos(2PIbw/sr)Y(n-1)
      x->bfilter[i].f_real=x->bfilter[i].y[0]  - x->bfilter[i].f_cr*x->
bfilter[i].y[1];
      //imaginary part=-Rsin(2PIbw/sr)Y(n-1)
      x->bfilter[i].f_imag=-x->bfilter[i].f_rsin*x->bfilter[i].y[1];
      //amplitude   spectrum
      x->bfilter[i].f_out=sqrt(pow(x->bfilter[i].f_real,2)+pow(x->
bfilter[i].f_imag, 2));
      //Feedback output
      x->bfilter[i].y[2]=x->bfilter[i].y[1];
      x->bfilter[i].y[1]=x->bfilter[i].y[0];
   if (x->spec_onoff)
      SETFLOAT (x->output+i, x->bfilter[i].f_out);

   }

   max =0;
/*
// Fundamental frequency calculation by correlation:
// 1- Sum all the multiplications between filter output and signal and
// divide by the number of points of the test signal
// 2- If the correlation sum is greater than the maximum and the center
// frequency is between beat boundaries (300 and 1500ms or 3.33 and
// 0.66Hz) return center frequency
*/
   for (i=0; i<SUMS; i++){
      for (j=0; j<TEST; j++) {
         x->sumsig[i]= x->sumsig[i]+x->bfilter[i+j].f_out*testsig[j];
         }
```

```
       x->sumsig[i]=x->sumsig[i]/TEST;
       if (x->sumsig[i]>max && (x->bfilter[i].f_cf>0.66 && x-
>bfilter[i].f_cf< 3.33)) {
          max=x->sumsig[i];   //find maximum between beat boundaries
          index=i;
          }
       }
   // Calculate relative amplitude of the first four harmonics
   sum=0;
   count=0;
   for (i=0; i<NUMFILTERS; i++){
       if (i<index)
          x->harmonics[i]=0;
       else {
          if ((i-index)< TEST){
             harmExtract=testsig[i-index];
             if (harmExtract!=0)
                harmExtract=1.;

             x->harmonics[i]=harmExtract*x->bfilter[i].f_out;
             if (x->harmonics[i]!=0) {
                tmp[count]=x->bfilter[i].f_cf;
                tmp[count+1]=x->harmonics[i];
                sum=sum+x->harmonics[i];
                count=count+2;
                   }
             }
          else
             x->harmonics[i]=0;
          }

       if (x->harm_onoff)
          SETFLOAT (x->displayharms+i, x->harmonics[i]);
   }

//pack list of the 4 initial harmonics in the form f a 2f a2 3f a3 4f a4
//with the sum of amplitudes normalized to 1
   for (i=0; i< 8; i++){
          if (sum !=0) {
             if (i%2 == 0)
                SETFLOAT (x->harmons+i, tmp[i]);
             else
                SETFLOAT (x->harmons+i, tmp[i]/sum);
          }
       }

       outlet_float(x->m_out, (float) x->bfilter[index].f_cf);
       outlet_list(x->m_out2, 0L, 8, x->harmons);
       if (x->spec_onoff)
          outlet_list(x->m_out3, 0L,NUMFILTERS,x->output);
       if (x->harm_onoff)
          outlet_list(x->m_out4, 0L,NUMFILTERS,x->displayharms);
```

```
        bandit_resetsum (x);

    qelem_unset(x->m_qelem);

}

////// Other functions //////
// check the time when val arrives at input
void bandit_checktime (Bandit *x)
{
    x->currtime=gettime();
    x->deltatime=x->currtime-x->prevtime;
    x->prevtime= x->currtime;
}

// get idletime when idle
void bandit_getidletime (Bandit *x)
{
    x->idletime=x->idletime+x->deltatime;
}

// reset correlation sum array
void bandit_resetsum (Bandit *x)
{
    int i;
    for (i=0; i<SUMS;i++)
        x->sumsig[i]=0.0;
}

//initialize filter bank
void bandit_init (Bandit *x)
{
    int i, j;

    for (i=0; i<NUMFILTERS; i++){
        x->bfilter[i].f_cf=0;
        x->bfilter[i].f_bw=0;
        x->bfilter[i].f_cr=0;
        x->bfilter[i].f_two_cr=0;
        x->bfilter[i].f_rsqrd=0;
        x->bfilter[i].f_out=0;
        x->bfilter[i].f_real=0;
        x->bfilter[i].f_imag=0;
        x->bfilter[i].f_rsin=0;
        for (j=0; j<3; j++)
            x->bfilter[i].y[j]=0;
    }

    //initialize other variables of the object
    x->m_qelem=qelem_new(x, (method) bandit_out);
    x->a_in=0;
    x->idle=0;
```

```
    x->dc_onoff=0;
    x->dc_out=0;
    x->a_inMinOne=0;
    x->idletime=0;
    x->deltatime=0;
    x->currtime=0;
    x->prevtime=0;
    x->idleinterval= 3000;  // 3 seconds default for idleint
    x->idlethresh = 200;    // 200 for idlethresh
    x->flagmove=0;
    x->flagidle=0;
    x->spec_onoff=1;
    x->harm_onoff=1;

    for (i=0; i< SUMS; i++)
        x->sumsig[i]=0.0;

}
/* function to create a new instance of the bandit class */

void *bandit_new(Symbol *s, short ac, Atom *av)

{
    Bandit *x;                      /* object to be created*/
    /* allocates memory and sticks in an inlet */
    x = (Bandit *)newobject(class);
    bandit_init (x);

    x->m_out5 = intout (x);
    x->m_out4 = listout (x);
    x->m_out3 = listout (x);
    x->m_out2 = listout (x);
    x->m_out = floatout (x);
    if (ac != 1)     // input in an Atom, ac in the number of arguments
                     // write to Max window
    {
        post("Default SR=30 will be used for m.bandit.");
        x->a_sr = 30;
    }
    else             // get values from the Atom
    {
        x->a_sr = av[0].a_w.w_long;
    }

    bandit_calculate (x);  //calculate all necessary coefficients
    post("m.bandit object © Carlos Guedes 2002-2005");
    return (x);             /* return a copy of the created object */
}

void bandit_free (Bandit *x)
{
    qelem_free (x->m_qelem);
```

```
}
```

testsig.h

```
//one hertz pulse signal as test

#define TEST 63

float testsig [TEST]={0.95362, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0.52969, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0.38007, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.30061};
```

m.clock.c

```
/*
   m.clock © Carlos Guedes 2002-2005

*/


/* the required include file(s) */
#include "ext.h"
#include <stdlib.h>

/* structure definition for the clock object */
typedef struct mclock
{
   Object m_ob;        // header

   long clocktime;     // time estimate after calculations
   short firstval;     // flag to test if first value arrived
   long candidate;     // value in milliseconds to be tested at input
   float delta;        // allowable margin for adaptation
   void *m_out2;       // outlets
   void *m_out;
} Clock;

/* global that holds the class definition */
void *Clock_class;

/* function prototypes */
void mclock_int (Clock *x, long n);
void mclock_in1 (Clock *x, long n);
void mclock_getdelta (Clock *, float f);
void mclock_acceptopinion (Clock *x, long n);
void mclock_reset (Clock *x);
void mclock_assist(Clock *x,void *b,long m,long a,char *s);
```

```
void mclock_calculate (Clock *x);
void *mclock_new (Symbol *s, float d);


void main(fptr *f)
{
   /* tell Max about the class. */
   setup((Messlist **)&Clock_class, (method)mclock_new, 0L,
(short)sizeof(Clock), 0L, A_DEFFLOAT, 0);

   /* bind methods to symbols */
   // method for int in second inlet (idle state from m.bandit)
   addinx((method)mclock_in1, 1);
   // method for int in first inlet (beat candidate)
   addint((method)mclock_int);
   // bind method to message "delta"
   addmess((method)mclock_getdelta, "delta", A_FLOAT, 0);
   // bind method to message "accept"
   addmess((method)mclock_acceptopinion, "accept", A_LONG, 0);
   // bind method to message "reset"
   addmess((method)mclock_reset, "reset", 0);
   // assist method
   addmess((method)mclock_assist,"assist",A_CANT,0);

   /* list object in the new object list */
   finder_addclass("Movement Analysis","m.clock");

   rescopy('STR#',8009); // copy assistance string resource to Max
                         // Temp file
}

/* methods*/
// int method
void mclock_int (Clock *x, long n)
{
   // if no first value was not recieved
   // check if it is within boundaries
   // turn firstval flag on
   // initialize clocktime to that value and output
   if(x->firstval==0){
      if (n>=300 && n<=1500) {
      x->firstval =1;
      x->clocktime=n;
      outlet_int(x->m_out, x->clocktime);
      }
   }
   // if first value has been received
   // check if it is a beat candidate
   // calculate new tempo estimate
   if (x->firstval) {
      if (n>=300 && n<=1500) {
      x->candidate=n;
```

```
        mclock_calculate(x);
        }
    }
}

// method for int in second inlet
void mclock_in1 (Clock *x, long n)
{
    outlet_int (x->m_out2, n);

}

// method for "delta" message
void mclock_getdelta (Clock *x, float f)
{
    if (f>=0 && f<=1)
        x->delta=f;

    else post ("0 <= delta <= 1 !");

}

// method for message "accept"
void mclock_acceptopinion (Clock *x, long n)
{
    // check if opinion is a valid beat candidate
    // if it is the first value (initialization)
    // turn on firstval flag
    // initialize clocktime to that value and
    // output
    if (n>=300 && n<=1500){
        if (!x->firstval)
            x->firstval= 1;
        x->clocktime= n;
        outlet_int(x->m_out, x->clocktime);
        post ("accepted %ld ms", x->clocktime);
    }
}

// method for message "reset"
void mclock_reset (Clock *x)
{
    x->firstval=0;
    x->candidate=0;
    x->clocktime=0;
    post("m.clock reset");

}

// assist method
void mclock_assist(Clock *x,void *b,long m,long a,char *s)
{
```

```
    assist_string(8009,m,a,1,3,s);
}

// function that calculates output
void mclock_calculate (Clock *x)
{
    long delta;

    // calculate the difference between beat candidate and
    // current estimate (delta)
    // check if that value is within allowable margin for
    // adaptation. If it is, recalculate clocktime
    delta= x->candidate-x->clocktime;
    if (abs(delta)<= x->clocktime*x->delta){
        x->clocktime = x->clocktime+delta;
        outlet_int(x->m_out, x->clocktime);
        }
}

/* function to create a new instance of the clock class*/
void *mclock_new (Symbol *s, float d)
{
    Clock *x;

    // get memory for a new object & initialize
    x = (Clock *) newobject(Clock_class);

    intin (x,1);               // additional inlet

    x->firstval=0;                 // initialize variables
    x->clocktime=0;
    post ("m.clock © Carlos Guedes 2002-2005");
    if (d<=0 || d>1){
        x->delta= 0.15;
        post ("default value for delta is 0.15");
        }
    else {
        x->delta=d;
        post ("delta: %lf", x->delta);
        }
    x->m_out2 = intout(x);  // ilde state from m.bandit (on/off to metro)
    x->m_out = intout(x);   // tempo outlet

    return (x);             // return newly created object to caller
}
```

m.weights.c

```
/*
   m.weights © Carlos Guedes 2002-2005
*/

#include "ext.h"

/* structure definition of the object */
typedef struct weight
{
   Object m_ob;            // header

   long m_value;           // store received value
   long collect[60];       // array that collects the previous 60 values
   long count;             // counter
   long occurences[12];    // array that stores the occurences of beat
                           // values from 300 - 1500 msec in 100 msec
                           // steps
   Atom out[12];           // output list of the percentage of
                           // occurrences in the past 60 values
   void *m_out2;           // list outlet
   void *m_out;            // maximum value outlet

} Weight;

/* globalthat holds the class definition */
void *weight_class;

/* function prototypes */

void weight_int(Weight *x, long n);
void weight_resetall (Weight *x);
void weight_assist(Weight *x,void *b,long m,long a,char *s);
void weight_findmax (Weight *x);
void weight_resetoccur(Weight *x);
void weight_init (Weight *x);
void *weight_new(float n);

/* initialization routine */

void main(fptr *f)
{
   /* set up the class. */
   setup((Messlist **)&weight_class, (method)weight_new, (method)0L,
(short)sizeof(Weight), 0L, A_DEFLONG, 0);

   /* bind methods to symbols */
   addint((method)weight_int);
   addmess((method)weight_resetall, "reset", A_LONG, 0);
   addmess((method)weight_assist,"assist",A_CANT,0);
```

```
    /* list object in the new object list */
    finder_addclass("Movement Analysis","m.weights");

    rescopy('STR#',8010);
}

// int method
void weight_int(Weight *x, long n)
{
    int i,index;
    long sum;
    x->m_value = n;
    // if val between beat boundaries
    if (x->m_value>=300 && x->m_value<=1500){
    // round values to 100 ms slot
        x->collect[x->count]=(int)((x->m_value-300)*0.01)*100+300;
    // increment count
        x->count++;
    // wrap around after receiving 60 values
        if (x->count>59)
            x->count=0;
        }
    // calculate percentage of occurrences
    // every 6 frames
    if (x->count%6==0){
        sum=0;
    // reset previous calculations
        weight_resetoccur(x);
    // find the number of occurrences of each 100ms bin
    // from 300-1500 ms
        for (i=0; i<60; i++){
            index= (int) ((x->collect[i]-300)*0.01);
            x->occurences[index]++;
        }
    // get the total of occurrences
        for (i=0; i<12; i++)
            sum+=x->occurences[i];
    // format list for viewing the percentage of occurrences
        for (i=0; i<12; i++){
            SETFLOAT(x->out+i, (float) x->occurences[i]/(float) sum);
            }
        outlet_list(x->m_out2,0L,12, x->out) ;
    // find the 100ms bin with more occurrences and output it
        weight_findmax(x);
    }

}

//reset method
void weight_resetall (Weight *x)
{
    int i;
```

```
    x->m_value = 0;
    x->count= 0;
    for (i=0;i<60; i++)
        x->collect[i]=0;
    for (i=0; i< 12; i++)
        x->occurences[i]=0;
    post("m.weights reset!");

}

// assist method
void weight_assist(Weight *x,void *b,long m,long a,char *s)
{
    assist_string(8010,m,a,1,2,s);
}

// other functions
// find bin with maximum occurrences
void weight_findmax (Weight *x)
{
    int i, index;
    int max=0;
    for (i =0; i<12; i++){
        if (x->occurences[i]>max){
            max=x->occurences[i];
            index=i;
            }
        }
    outlet_int (x->m_out, (index*100)+300);
}

// reset previous occurrences
void weight_resetoccur(Weight *x)
{
    int i;
    for (i=0; i< 12; i++)
        x->occurences[i]=0;
}

// initalize all variables
void weight_init (Weight *x)
{
    int i;

    x->m_value = 0;
    x->count= 0;
    for (i=0;i<60; i++)
        x->collect[i]=0;
    for (i=0; i< 12; i++)
        x->occurences[i]=0;
}
```

```
// object instantiation routine
void *weight_new(float n)
{
   Weight *x;
// get memory for a new object & initialize
   x = newobject(weight_class);
   weight_init(x);
   x->m_out2=listout(x);
   x->m_out =intout(x);
// return newly created object to caller
   post ("m.weights object © Carlos Guedes 2002-2005");
   return (x);
}
```

m.peak.c

```
/*
   m.peak © Carlos Guedes 2002-2005
 */


#include "ext.h"

/* object definition */
typedef struct peak
{
   Object m_ob;
   void *m_out;           // create outlet
   long values[6];        // array to store values (6 frames)
   short curdir;          // current direction (up=1, down=0)
   short prevdir;         // previous direction (up=1, down=0)
   short changedir;       // store direction changes
   long peakrange;        // range definition for a peak
   long curtime;          //
   long prevtime;             // logical time variables
   long deltatime;            //
   short counter;             // counter
} Peak;

/* globalthat holds m.peak class */
void *peak_class;

/* function prototypes */

void peak_in1(Peak *x, long n);
void peak_int(Peak *x, long n);
void reset_peak (Peak *x);
```

```
void peak_assist(Peak *x, void *b, long m, long a, char *s);
void swap (Peak *x);
void peak_gettime(Peak *x);
long findmin (Peak *x);
void *peak_new(long n);

/* initialization routine */
void main(fptr *f)
{
   /* set up the class */
   setup((Messlist **)&peak_class, (method)peak_new, (method)0L,
(short)sizeof(Peak), 0L, A_DEFLONG, 0);

   // method for int in left inlet (value to be analized)
   addint((method)peak_int);
   // method for int in right inlet (peak range)
   addinx((method)peak_in1,1);
   // bind method for reset
   addmess((method)reset_peak, "reset", 0);
   // bind method for assistance
   addmess((method)peak_assist, "assist", A_CANT, 0);
   /* list object in the new object list */
   finder_addclass("Movement Analysis","m.peak");
}

/*methods */
// method for int in right inlet
void peak_in1(Peak *x, long n)
{
   x->peakrange=n;
   post("peakrange: %ld", x->peakrange);
}

//method for int in left inlet
void peak_int(Peak *x, long n)
{
   long localmin;
   //store first 5 values without calculating if a peak occurred
   if (x->counter<5) {
      x->values[x->counter]=n;
   //calculate direction on the 5th stored value
      if (x->counter==4){
         if (x->values[4]-x->values[3]>=0)
            x->prevdir=1;
         else x->prevdir=0;
         }
   x->counter++;
   }
   //after the 6th value is received…
   else {
      x->values[5]=n;
   //check direction (if is going up or down from previous)
```

```
        if (x->values[5]-x->values[4]>=0)
            x->curdir=1;
        else x->curdir=0;
    //check if direction changed from up to down
        x->changedir=x->curdir-x->prevdir;
        if (x->changedir==-1){
    //if it did...
    //find minimum value in the array
            localmin=findmin(x);
    //if the difference between the previous-to-last value in array and
    //the minimum is greater than the peak range AND the time between
    //this peak and the previous is greater than 120 ms output a bang
    //message
            if (x->values[4]-localmin>x->peakrange) {
                peak_gettime(x);
                if (x->deltatime>120)
                    outlet_bang (x->m_out);
                }
            }
    //move values in the array to the left (discard older value)
        swap (x);
    //update direction
        x->prevdir=x->curdir;
    }
}

// method for message "reset"
void reset_peak (Peak *x)
{
    int i;
    for (i=0; i<6; i++)
        x->values[i]=0;
    x->counter=0;
    post ("m.peak reset!");
}

// assist method (alternative way of commenting inlets and outlets
// without a resource file
void peak_assist(Peak *x, void *b, long m, long a, char *s)
{
    if (m == ASSIST_INLET){
        switch (a) {
        case 0:
            sprintf(s,"Values to be analyzed");
            break;
        case 1:
            sprintf(s,"Peak threshold");
            break;
        }
    }
    else if (m == ASSIST_OUTLET) {
        switch (a) {
```

```
        case 0:
            sprintf(s,"Bang if there is a peak");
            break;
        }
    }
}

/*additional functions*/
// swap values in array
void swap (Peak *x)
{
    int i;
    for (i=1; i<6; i++)
        x->values[i-1]=x->values[i];
}

// get logical time
void peak_gettime(Peak *x)
{
    x->curtime= gettime();
    x->deltatime= x->curtime-x->prevtime;
    x->prevtime= x->curtime;
}

// find minimum value in array
long findmin (Peak *x)
{
    int i;
    long min;
    min= 10000000000;

    for (i=0; i < 5; i++){
        if (x->values[i]< min)
        min= x->values[i];
        }
    return min;
}

/* instance creation routine */

void *peak_new(long n)
{
    Peak *x;
    int i;


    x = newobject(peak_class);     // get memory for a new object &
initialize
    intin (x,1);
    x->m_out = bangout(x);

    for (i=0; i<6; i++)
```

```
        x->values[i]=0;

    x->counter=0;           // store value (default is 0)
    x->curtime=0;
    x->prevtime=0;
    x->deltatime=0;

    x->prevdir=0;
    x->curdir=0;
    x->peakrange=n;
    post("m.peak ©Carlos Guedes 2002-2005");
    if (x->peakrange==0){
        x->peakrange= 100;
        post ("default value for peak range 100");
    }
    else post ("peak range = %ld", x->peakrange);

    return (x);                // return newly created object to caller
}
```

m.sample

```
/* the required include file(s) */
#include "ext.h"

/* structure definition of the object */
typedef struct sample
{
    Object m_ob;

    long data;       // data to be sampled
    double m_sr;     // sampling rate
    void *m_clock;   // clock
    short bang;      // bang flag
    void *m_out;     // outlet
} Sample;

/* globalthat holds the class definition */
void *sample_class;

/* function prototypes */

void sample_bang(Sample *x);
void sample_start (Sample *x);
void sample_stop(Sample *x);
void sample_int(Sample *x, long n);
void sample_setSR(Sample *x, long sr);
void sample_assist(Sample *x,void *b,long m,long a,char *s);
void sample_tick (Sample *x);
void *sample_new(long n);
```

```
void sample_free (Sample *x);

/* initialization routine */

void main(fptr *f)
{
   setup((Messlist **)&sample_class, (method)sample_new,
(method)sample_free, (short)sizeof(Sample), 0L, A_DEFLONG, 0);

   /* bind the methods to symbols */
   // bind mesthod to bang message
   addbang((method)sample_bang);
   // int method
   addint((method)sample_int);
   // bind method to "sr" message
   addmess((method)sample_setSR, "sr", A_LONG, 0);
   // bind method to "stop" message
   addmess((method)sample_stop, "stop", 0);
   // bind method to "start" method
   addmess((method)sample_start, "start", 0);
   // assist method
   addmess((method)sample_assist,"assist",A_CANT,0);
   /* list object in the new object list */
   finder_addclass("Movement Analysis","m.sample");

   rescopy('STR#',8008);   /* copy assistance string resource to
                              Max Temp file */
}

/* methods */
// bang method
void sample_bang(Sample *x)
{
   if (x->bang==0)  {  // if first bang is received start clock
      x->bang=1;
      post ("sampling...");
      clock_fdelay(x->m_clock,0.);
   }
   else {           // else stop it
      x->bang=0;
      sample_stop(x);
      }
}

// method for "start" message
void sample_start (Sample *x)
{
   post ("sampling...");
   x->bang=1;
   clock_fdelay (x->m_clock, 0.);
}
```

```
// method for "stop" message
void sample_stop(Sample *x)
{
   x->bang=0;
   post ("m.sample stopped");
   clock_unset(x->m_clock);
}

// int method
void sample_int(Sample *x, long n)
{
   x->data=n;
}

// method for message "sr"
void sample_setSR(Sample *x, long sr)
{
   if (sr>0) {
      x->m_sr= 1000./(float) sr;
      post ("new sampling rate: %lf", x->m_sr);
   }
   else
      post ("sampling rate > 0!");
}

// assist method
void sample_assist(Sample *x,void *b,long m,long a,char *s)
{
   assist_string(8008,m,a,1,2,s);
}

// function that outputs values at clock rate
void sample_tick (Sample *x)
{
   clock_fdelay (x->m_clock, x->m_sr);
   outlet_int (x->m_out,x->data);
}


// function that ceates new object
void *sample_new(long n)
{
   Sample *x;

   // get memory for a new object & initialize
   x = newobject(sample_class);
   //initialize clock
   x->m_clock = clock_new(x, (method) sample_tick);

   if (n>0) {
      x->m_sr=1000./(float) n;
      post("sr= %ld", n);
```

```
        }
    else {
        x->m_sr=33.333333;
        post ("default value sr=30 will be used");
    }

    // other initializations
    x->data=0;
    x->bang=0;

    // create outlet
    x->m_out = intout(x);
    post ("m.sample @Carlos Guedes 2002-2005");

    // return newly created object to caller
    return (x);
}

// free clock
void sample_free (Sample *x)
{
    freeobject((t_object *)x->m_clock);
}
```

APPENDIX D

DETAILED DESCRIPTION OF THE MAX/MSP PATCH FOR *ETUDE FOR UNSTABLE TIME*

In this appendix, I do a detailed description of the three sections comprising the Max/MSP patch used for the performance of *Etude for Unstable Time*. Prior to reading this section, the reader may want to recall Figure 20 in Chapter 7 (p. 170) in which a full image of the user-interface level for the patch is shown.

Video Analysis Section

This section of the patch performs basic analyses of the video stream. After the video stream is digitized, there are two types of analyses performed on the digitized stream: the extraction of the absolute brightness difference value through frame differencing, and positional analysis through background subtraction. The instantaneous value of the absolute brightness difference is sent remotely via the 'motion' message using the object **s** (**send**) to several places in the patch — These include the inlet of **m.bandit** so that the musical rhythm analysis of the movement is produced, and the inlet of the object **m.peak**, so that the analyses of the signal's peaks are done**.**

The background subtraction is done utilizing **v.presence**. An image of the background is previously stored in this object so that the silhouette of the dancer

is extracted. This silhouette is subsequently binarized through the object **v.>=** .[77] This image feeds the object **v.regions** that outputs the position of the dancer in the horizontal axis via the message 'X.' The position of the dancer in this axis is mapped to the position in which the sample is read by object **Grain**. Figure 24 provides a detail of the video analysis section of the patch.

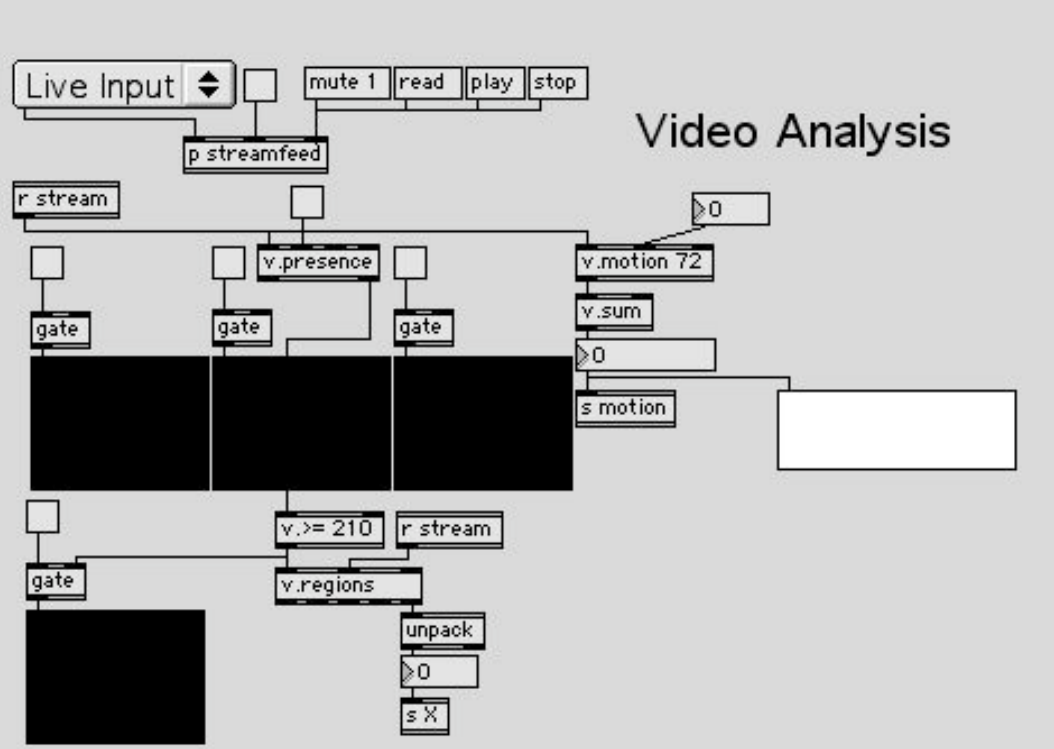


Figure 24. Detail of the Video Analysis section of the patch for *Etude for Unstable Time*

In the video analysis section of the patch, there are gates that turn on or off each **v.screen** displaying the output produced by each video analysis/processing

[77] SoftVNS object **v.>=** forces to black (0) all the pixels that are below the brightness threshold and to white (255) all the pixels whose value is above or equal the threshold vale. This way, one obtains a black and white (*binarized*) representation of the image.

operation. This is done in order to spare processing cycles at run time since the display of the streams is a very CPU-consuming operation.

Movement Periodicity Analysis

This portion of the patch is where the musical processing of the dance is done, and where the most part of the m-objects are utilized. A detail of this section of the patch is shown in Figure 25. This section takes the absolute value of the brightness difference from the digitized stream to perform all the necessary operations for the generation of musical rhythm and musical tempo control from dance. This value is received by the object **m.sample** via the message 'motion' and is sampled at 30 Hz (or 30 fps). After the values are sampled, they are sent to **m.bandit**. Once the sampling is turned on, **m.bandit** computes the fundamental of the signal and the amplitude and frequency of the four initial harmonics.

At the top of this section one can see a **preset** object. This object stores several presets for the messages 'idle_int' and 'idle_thresh.' This has the purpose of providing **m.bandit** with different degrees of sensitivity to movement during the performance of the piece. For example, in the beginning of the piece, **m.bandit** only processes the movement data when the dancer performs rather accentuated full-body actions, since the motion threshold is rather high. In the beginning, although the dancer is moving, **m.bandit** is not triggering musical data. During the first section of the piece the sensitivity to movement of **m.bandit** is gradually increased. This is a possible way to effectively control the density of musical events in this section.

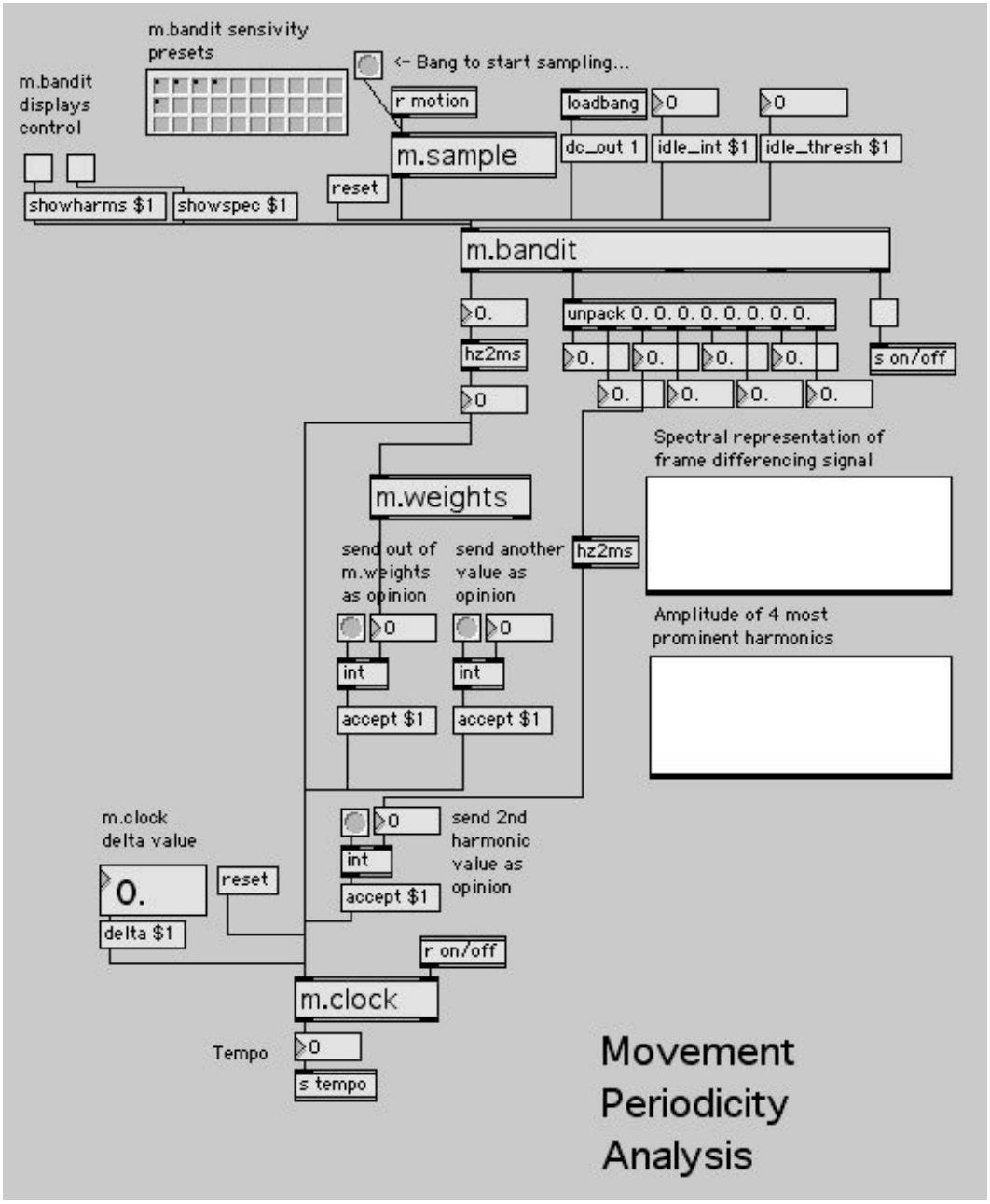


Figure 25. Movement periodicity analysis section of the patch.

Sound Engine

The sound engine section of the patch contains all the objects that generate the sound for *Etude for Unstable Time*. These objects receive the temporal articulation data from the m-objects. There are no pre-stored musical sequences in the patch, and the temporal articulation of the music of the piece is controlled by the m-objects and by me. In Figure 26, I show in detail the user-interface level of this section.

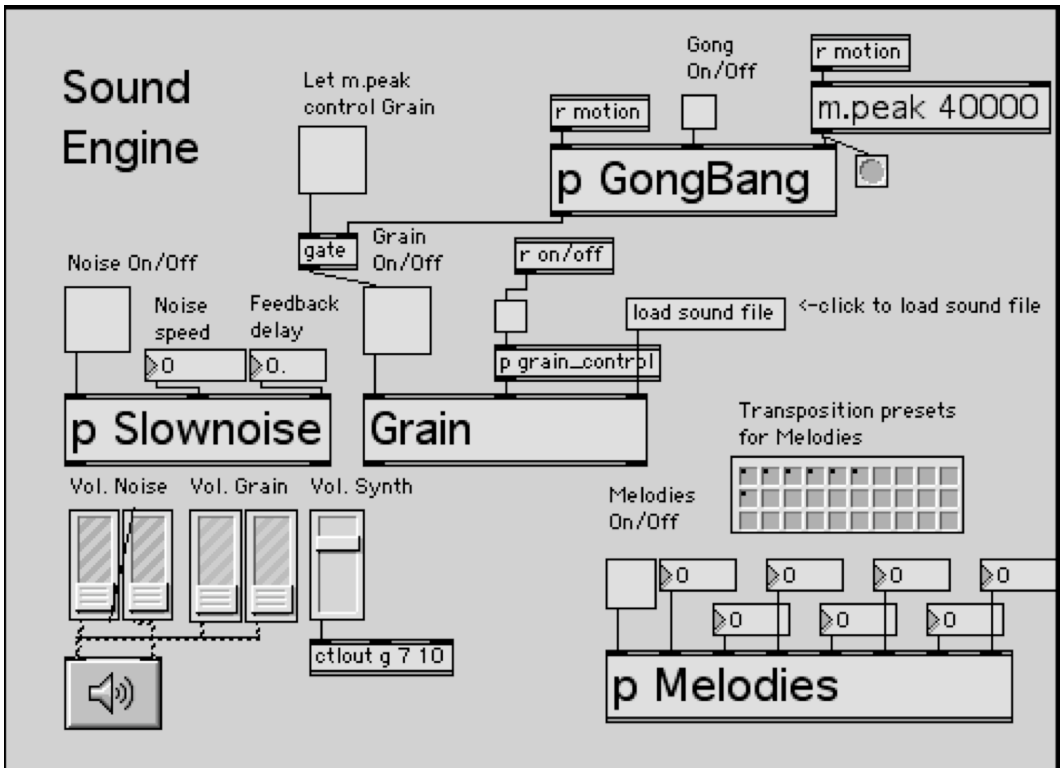


Figure 26. The user-interface level of *Sound Engine*.

Generation of Musical Material

Four Max/MSP objects were created to generate all the sound for *Etude for Unstable Time:* **Melodies**, **SlowNoise**, **GongBang**, and **Grain**. These objects

respond directly or indirectly to the data sent by the m-objects. According to Rowe's proposed taxonomy (1993) to classify the response methods in interactive systems, these objects could be grouped into algorithms that have a generative response (**Melodies** and **GongBang**) and algorithms that have transformative response (**SlowNoise** and **Grain**).

Melodies

Melodies is an object that generates melodies whose rhythm is driven by the harmonic output from **m.bandit. Melodies** contains three modules (subpatches **p sound1, p sound2,** and **p sound3**), each storing a set of MIDI note numbers that are respectively triggered when the amplitude of the harmonics 1, 2 and 4 pass a certain threshold. Their rhythm is the frequency of the harmonic converted to milliseconds. The process for the rhythmic generation of these melodies is very similar to the one explained above in the practical applications of the library. Each module containing its own collection of MIDI note numbers was designed similarly. The difference among them pertains to the note numbers they store, and to the MIDI channel used to output the MIDI note events. In Figure 27, we can see the internal structure of module **p sound1.**

The module works as follows: the MIDI note numbers are stored in the **coll** object. Whenever it receives a 'bang' message, the **urn** object generates a random number between 0 and 6. This number will be the index for the stored MIDI note number. The indexed MIDI note number is output by **coll** and then

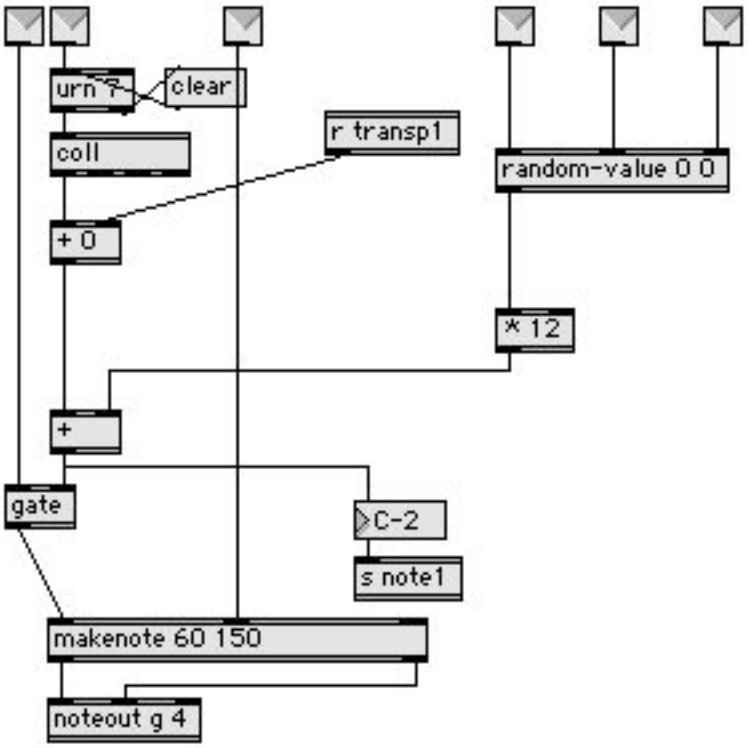


Figure 27. Internal structure of module **p sound1.**

is added a transposition value (sent through the 'transp1' message), and an octave index. The octave index is sent by the object **random-value**, an object that outputs random values between two integers. If the **gate** is open, this value is sent to the **makenote** object. The value for the velocity (sent to the middle inlet of **makenote**) is proportional to the amplitude of the harmonic whose frequency is the duration for that note (since **p sound1** receives the frequency of the first harmonic, the velocity would be proportional to the amplitude of the first harmonic). Finally, the MIDI note number and velocity are sent to the **noteout** object, that sends the MIDI message to a virtual synthesizer via OMS's IAC bus.

The MIDI note number is also sent remotely to the object **Slownoise**, via the message ‘note1.’

The values for the transposition and octave index for each of the three modules comprising **Melodies,** as well as a switch that turns the object on or off, are available at the user-interface level of the patch. For the performance of *Unstable Time,* I had seven presets with ascending transposition values for the notes generated by **Melodies**.

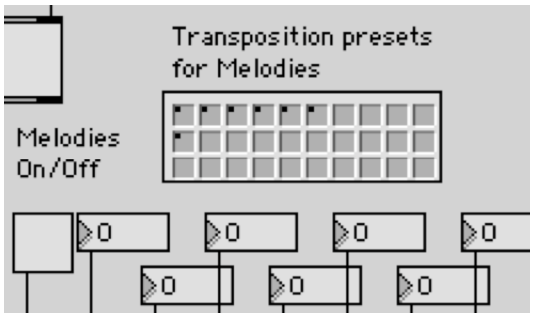


Figure 28. User interface for object **Melodies** with presets.

Slownoise

Slownoise is an object that generates resonant-filtered noise bands whose center frequencies are the frequencies of the MIDI notes generated by **Melodies.** The object receives remotely the three notes produced by **Melodies** at all times. The MIDI note numbers are converted to Hertz by MSP’s **mtof** object. After being converted to Hertz, these values are multiplied by several factors to produce other center frequencies for additional bands as shown in figure 29.

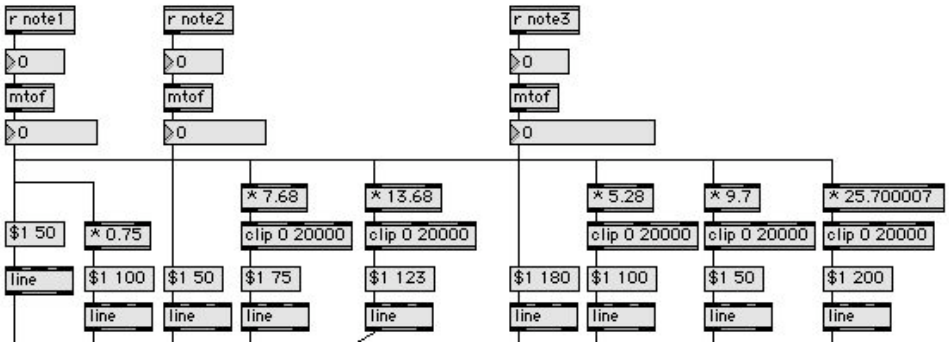


Figure 29. Creation of nine noise-band center frequencies from MIDI notes generated from **Melodies.**

Nine different center frequencies that are related to the MIDI notes are thus produced. After being interpolated by **line**, to avoid sudden jumps in frequency, these values feed corresponding **reson~** objects that filter *slowed- down* white noise with a very high factor Q.

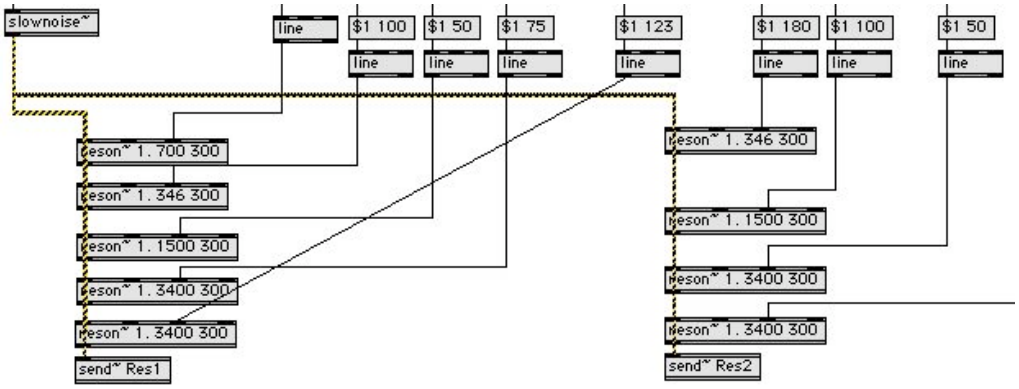


Figure 30. Creation of nine different slowed-down noise bands with **reson~** objects.

slownoise~ is a MSP external created by Paul Berg that allows for downsampling white noise by repeating the same noise value (a random real number between -1.0 and 1.0). The number of occurrences (integer value) of each noise value can be input as an argument to the object or sent to the object's inlet (when this value

is 1 no repetitions occur). At the front end of the patch of the patch, the user can turn the object on or off, define the number of occurrences for each noise value (the higher the number the *slower* the noise will sound) and set the delay feedback value.

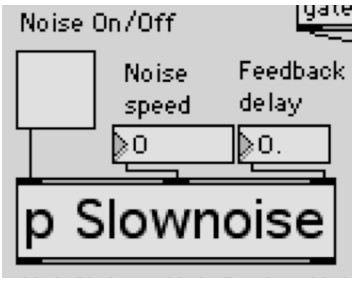


Figure 31. User interface for **Slownoise.**

Grain

Grain embeds in an object a patch for granular sample playback created by Richard Dudas and Leslie Stuck that is distributed with the later versions of Max/MSP. I did slight modifications to the patch for the performance of *Unstable Time,* namely by mapping the dancer's position in the horizontal axis to the reading position of the sample. That is, the object receives through the message 'X' the position of the dancer in the screen, and maps that value to the position at which the sample is played (see Figure 32). The user interface for **Grain** allows the user to load a sound file into the object (message box connected to the rightmost inlet), to turn the object on or off (leftmost inlet), and also allows **m.peak** to turn the object on or off when a 'bang' message is produced (the **toggle** below the comment 'let m.peak control Grain'). The middle inlet of **Grain**

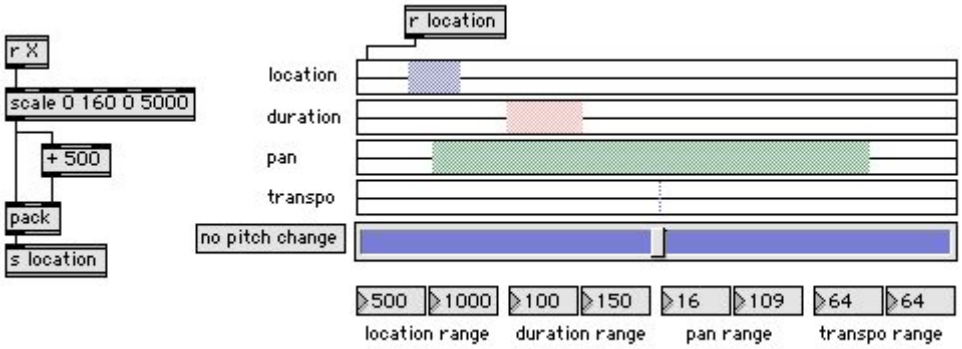


Figure 32. Mapping the dancer's position to the sample reading location in **Grain**.

receives the grain IOI values from the subpatch **p grain_control** that is controlled remotely by **m.bandit'**s 'idle' state. This received remotely through the message 'on/off.' The **toggle** below the **r** (**receive**) object allows the user to see if **m.bandit** is declaring 'moving' (1) or 'idle' (0) state (see Figure 33).

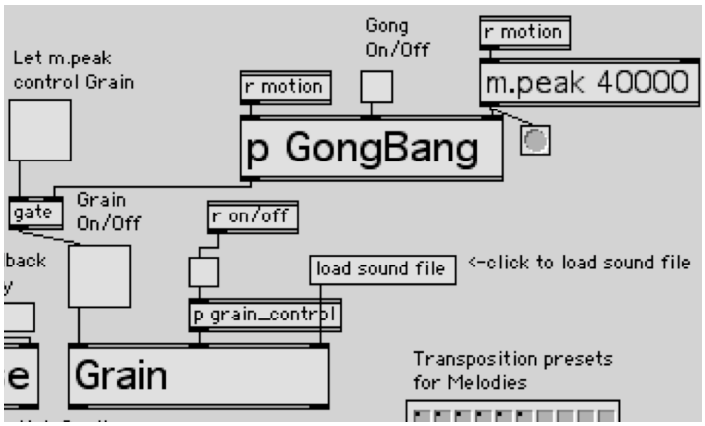


Figure 33. User interface for **Grain.**

The subpatch **p gain_control** sets the duration between grains to half of the tempo estimate sent by **m.clock** when **m.bandit** declares 'moving' state. When **m.bandit** declares 'idle' state, the subpatch gradually sets the time between grains to 30 milliseconds (Figure 34).

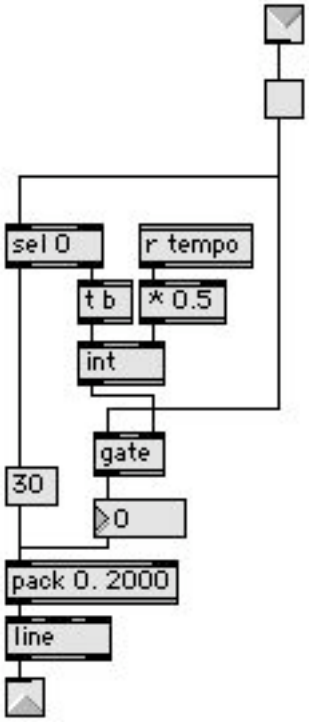


Figure 34. Subpatch **p grain_control**.

GongBang

GongBang responds to the output of **m.peak**. When **m.peak** detects a peak in the frame differencing signal, it sends a 'bang' message to this object, in order to trigger a chord of gong-like sounds. Like the other musical generation objects in the patch, **GongBang** can be turned on and off at the user-interface level (see Figure 35). The object receives at the leftmost inlet the instantaneous value of the quantity of motion through the remote message 'motion,' and at the rightmost inlet the 'bang' message from **m.peak.** The internal structure of this object can be seen in Figure 36.

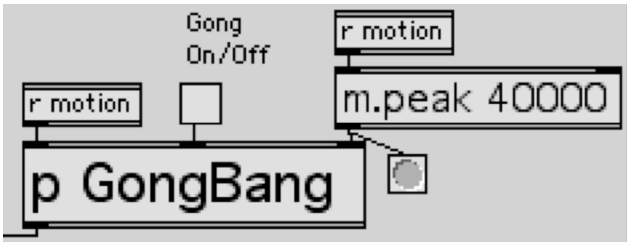


Figure 35. User interface for **GongBang**.

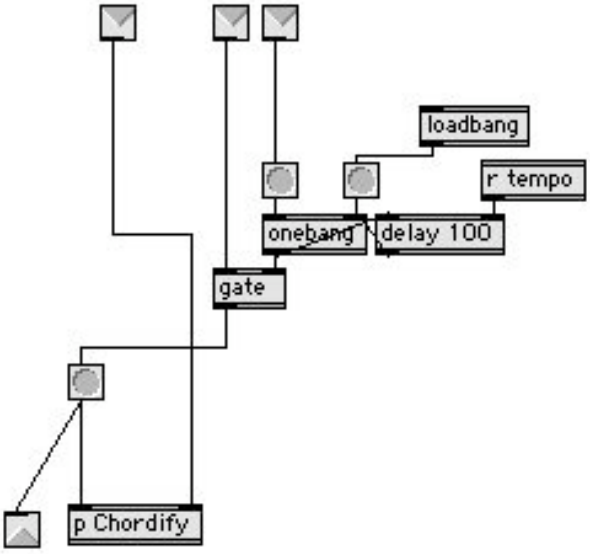


Figure 36. Internal structure of **GongBang**.

When **m.peak** sends a 'bang' message to the object and the **gate** is open, the 'bang' message is passed to the subpatch **p Chordify**. Note the **onebang** object at the right inlet of **gate**, in order to let the 'bang' messages pass to **p Chordify** at the maximum rate of the tempo estimate sent by **m.clock.** This is intended to filter subsequent 'bang' messages that may be produced by **m.peak** at a rate smaller than the actual tempo estimate. **p Chordify** generates chords of seven pitches distributed randomly by three octaves. It takes as input, in its left inlet, the 'bang' message from **m.peak,** and in its right inlet, the value of quantity of motion at

which the peak occurred. This is done in order to trigger the velocity of the MIDI notes in proportion to the peak value — i.e., larger peak values trigger louder gong sounds, smaller peak values trigger softer gong sounds. Like in **Melodies**, **GongBang** sends out the chord notes via the IAC bus to a virtual synthesizer. The subpatch **p Chordify** is depicted in Figure 37.

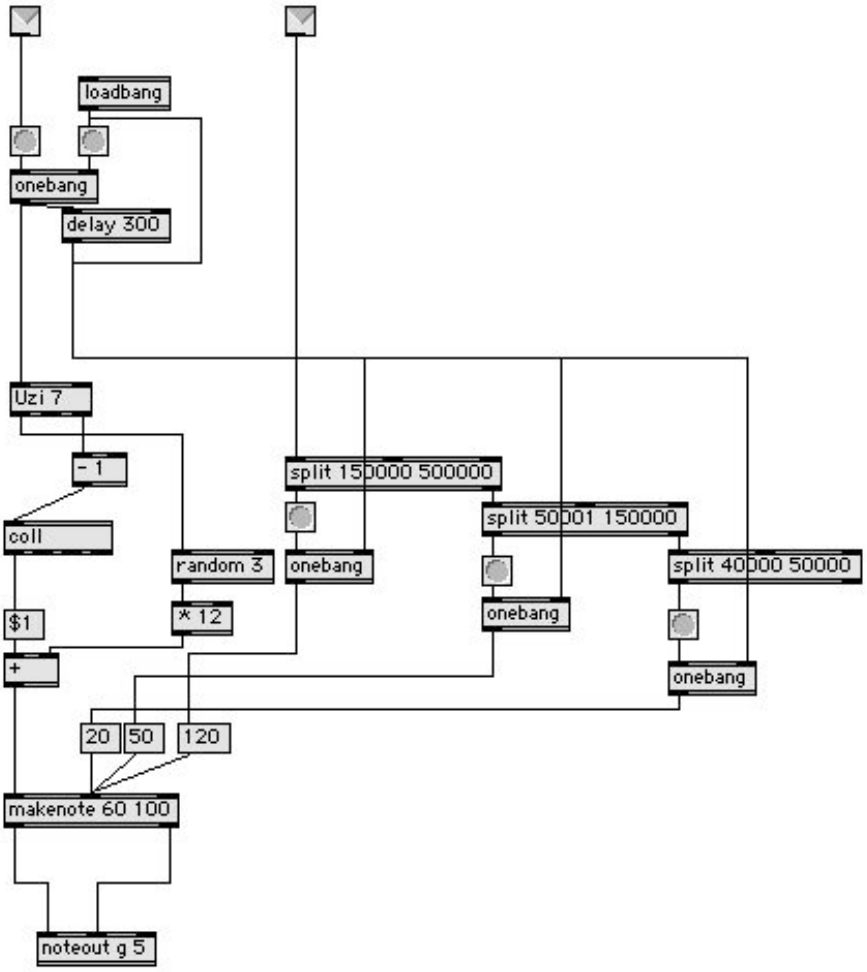


Figure 37. Subpatch **p Chordify.**