ACTION – SOUND

Developing Methods and Tools to Study
Music-Related Body Movement

Alexander Refsum Jensenius

Ph.D. thesis
Department of Musicology
University of Oslo
2007

Thesis submitted 11 July 2007
Thesis defended 31 January 2008

Advisors:

Professor Rolf Inge Godøy, Universitetet i Oslo
Associate Professor Marcelo M. Wanderley, McGill University

Committee:

Professor Sylvie Gibet, Université de Bretagne Sud
Dr. Roberto Bresin, Kungliga Tekniska Högskolan
Professor Stan Hawkins, Universitetet i Oslo

Series of dissertations submitted to the
Faculty of Humanities, University of Oslo
No. 234

ISSN 0806-3222

Author email: alexander@jensenius.no

If we knew what we were doing, it wouldn't be called research, would it?

Albert Einstein

Abstract

Body movement is integral to both performance and perception of music, and this dissertation suggests that we also think about music as movement. Based on ideas of *embodied music cognition*, it is argued that ecological knowledge of *action-sound couplings* guide our experience of music, both in perception and performance. Then follows a taxonomy of music-related body movements, before various observation studies of perceiver's *music-movement correspondences* are presented: *air instrument* performance, *free dance* to music, and *sound-tracing*. These studies showed that both novices and experts alike seem to associate various types of body movement with features in the musical sound. Knowledge from the observation studies was used in the exploration of *artificial action-sound relationships* through the development of various prototype music controllers, including the *Cheapstick*, *music balls*, and the *Music Troll*. This exploration showed that it is possible to create low-cost and human-friendly music controllers that may be both intuitive and creatively interesting. The last part of the dissertation presents tools and methods that have been developed throughout the project, including the *Musical Gestures Toolbox* for the graphical programming environment Max/MSP/Jitter; techniques for creating *motion history images* and *motiongrams* of video material; and development of the *Gesture Description Interchange Format* (GDIF) for streaming and storing music-related movement data. These tools may be seen as an answer to many of the research questions posed in the dissertation, and have facilitated the analysis of music-related movement and creation of artificial action-sound relationships in the project.

Acknowledgments

A number of people has been of great importance to this project. Going back to my undergraduate days as a physics student, Arnt Inge Vistnes was the person who introduced me to digital audio and the Fourier transform and laid the ground for my interest in studying music from a scientific perspective. Later, when I started studying music, I was warmly welcomed by Jøran Rudi and Bjarne Kvinnsland at NoTAM, both of whom also helped me carry out my first music technology project.

During my MA studies I was fortunate to work with David Wessel at CNMAT (UC Berkeley), who introduced me to timbre research and the world of live electronics using Max/MSP. There I also became interested in the interdisciplinary research world of music technology, something which was further strengthened by Mats Nordahl during my MSc studies at Chalmers in Gothenburg.

As a PhD student I have had the opportunity to spend a great deal of time in the laboratories of Marcelo M. Wanderley and Philippe Depalle at McGill University. I am particularly thankful to Marcelo for accepting to co-supervise my dissertation. At McGill I have also had the pleasure of attending the PhD seminars given by Gary Scavone and Stephen McAdams, as well as having great cooperation and discussions with Carmine Casciato, Rodolphe Koehly, Joe Malloch, Mark Marshall, Nils Peters, Steve Sinclair and Doug van Nort.

Neither should I forget the close collaboration over the last years with Tim Place, Trond Lossius, Pascal Baltazar and Dave Watson in the Jamoma project. Also, thanks to the EU Cost Action 287 ConGAS, the Integra project and the Enactive Network of Excellence, I have been fortunate to work closely with many researchers around Europe. The S2S^2 and ISSSM summer schools have also been of great value for my development as a researcher throughout this project.

In Oslo, I have had engaging musical collaborations with Åshild Ravndal Salthe, Marie Fonneløp, Kjell Tore Innervik, Ivar Frounberg and Else Olsen S, and have worked closely with Einar Sneve Martinussen and Arve Volsund on various projects. I have also benefited from many good discussions with colleagues in the Musical Gestures Group and other colleagues at the Department of Musicology. Here I would particularly like to thank Tellef Kvifte, who has given invaluable feedback on several chapters of this dissertation. Thanks also to Antonio Camurri, Albrecht Schneider and Marc Leman who have given valuable feedback on my project as international advisors to the Musical Gestures Project.

I should not forget to mention the Norwegian Research Council and their generous support of the Musical Gestures project, through which my fellowship has been funded. They also granted bilateral research funding to expand our collaboration with McGill, and awarded me an international travel scholarship in 2006. Thanks also to extra travel funding from the Department of Musicology and the EU Cost Action 287 ConGAS, which have allowed me to present my research at several international conferences over the last years.

None of this would have been possible without the support of my advisor throughout all these years, Rolf Inge Godøy, who has encouraged and guided me gently from the very beginning of my music studies.

Finally, thanks to Paula, Francesca, Grete, Jørgen and Marie for continued support and encouragement past, present and future.

This document was typeset in LaTeX 2_ε using the TexShop and TextMate editors. The bibliography was handled with BibDesk and DevonThink Pro. Figures were made with OmniGraffle Pro and Comic Life. The PDF version of the dissertation provides coloured hyperlinks to references, figures, tables and chapters, and to audio and video examples on the accompanying CD-ROM. Max/MSP objects are marked with **bold** text. The dissertation is published for non-profit educational purposes. Copyrighted material has been included under the guidelines of academic *fair use*.

TABLE OF CONTENTS

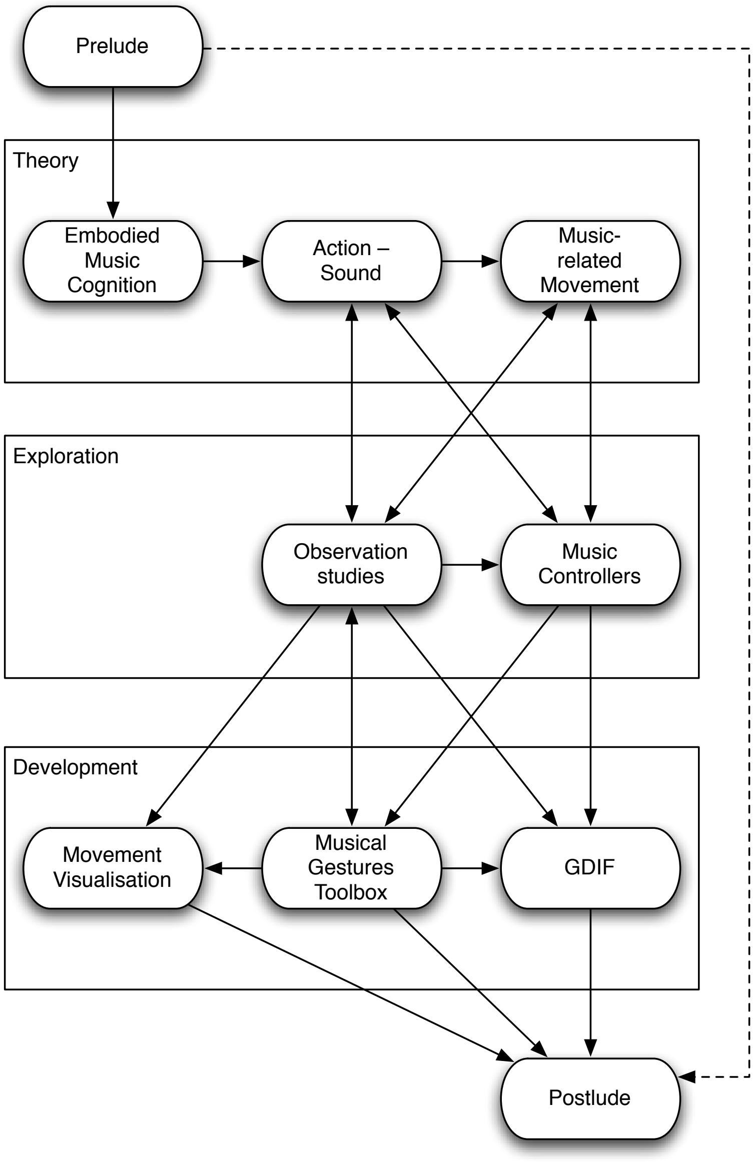


Figure 1: *Visual overview of the dissertation.*

Abbreviations

The following are some of the abbreviations used in the dissertation:

DMI Digital Musical Instrument

DoF Degrees of Freedom

GUI Graphical User Interface

HCI Human Computer Interaction

LMA Laban Movement Analysis

Max Max/MSP/Jitter programming environment

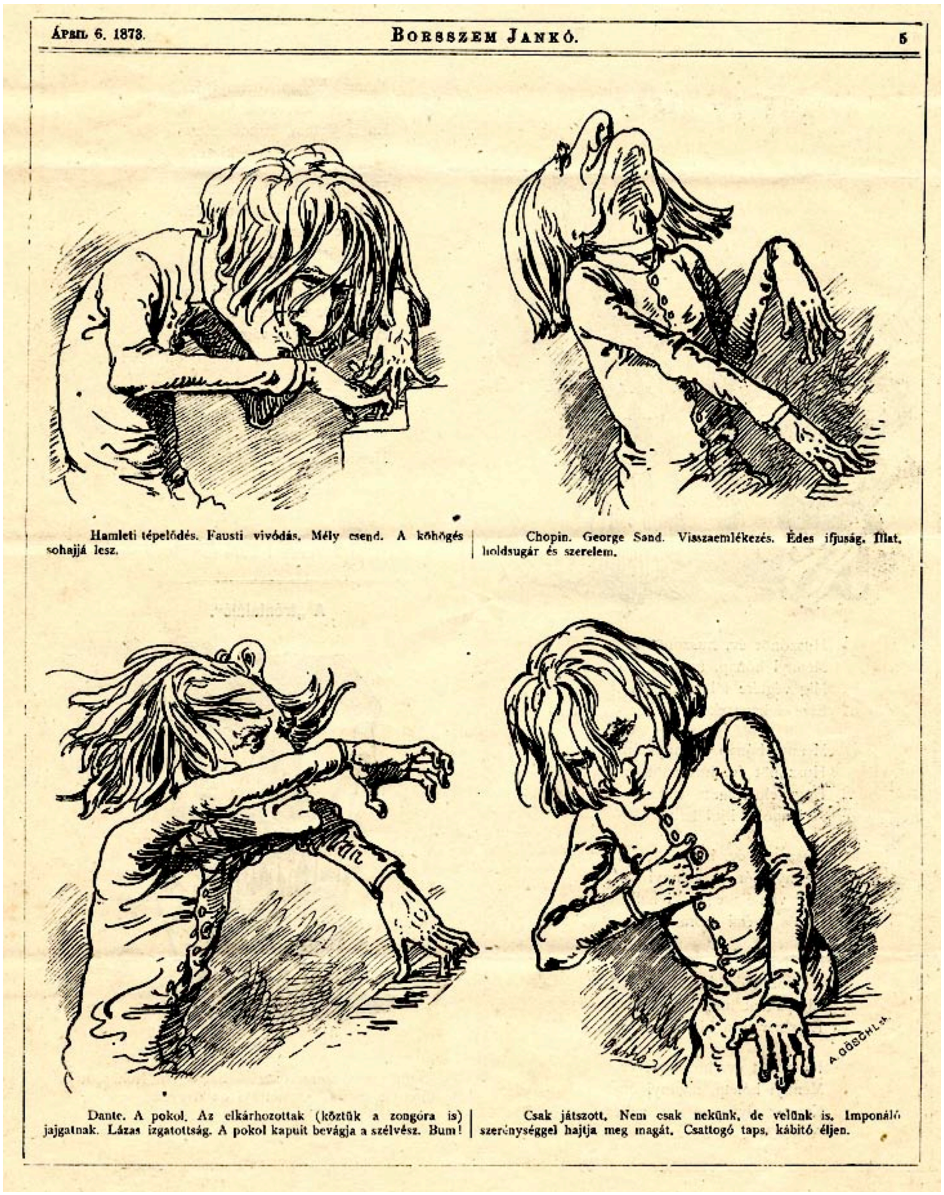
MGT Musical Gestures Toolbox

MI Motion Image

MHI Motion History Image

NIME New Interfaces for Musical Expression

QoM Quantity of Motion



ÁPRIL 6. 1873. BORSSZEM JANKÓ. 5

Hamleti tépelődés. Fausti vivódás. Mély csend. A köhögés sohajjá lesz.

Chopin. George Sand. Visszaemlékezés. Édes ifjuság. Illat, holdsugár és szerelem.

Dante. A pokol. Az elkárhozottak (köztük a zongora is) jajgatnak. Lázas izgatottság. A pokol kapuit bevágja a szélvész. Bum!

Csak játszott. Nem csak nekünk, de velünk is. Imponáló szerénységgel hajtja meg magát. Csattogó taps, kábító éljen.

Figure 2: *Caricatures of Liszt's performance movements (Göschl, 1873).*

CHAPTER 1 Prelude

Music, despite its phenomenological sonoric ethereality, is an embodied practice, like dance and theater.

Richard Leppert (1993, xxi)

This chapter starts out with an introduction to the importance of body movement in music performance and music perception, and continues by presenting the research questions, premises and limitations of the dissertation project. A discussion then follows of the traditions in which the project has been carried out, followed by an outline of the dissertation.

1.1 Music and Movement

Music is movement. A strong claim, and one which I will try to justify throughout this dissertation. The statement does not only refer to the physical movement of sound waves, but also to the fact that we *create* music by moving, and we *perceive*[1] music while moving. In fact, body movement seems to be an integral part of both performance and perception of music. To understand more about such *music-related movement*[2] is the core of this dissertation.

[1] In this dissertation I will use the terms *perceiver* and *perception* instead of *listener* and *listening*. This is to acknowledge that all *modalities* are involved in our experience of music. *Perceiver* is used instead of *receiver* to stress that perception is an active process. The term *listening* will only be used when referring to *auditory* music perception.

[2] I will use the term *music-related movement* to refer to all sorts of body movement found in music performance and perception.

Even though body movement seems to be an essential part of all musical activity in all cultures, it has received relatively little attention in traditional music research. A parallel can be drawn to the situation in linguistics in the 1970s, when it was reported that "the gesture specialist is a rare bird indeed — not so much a vanishing species, as one that has hardly yet begun to evolve" (Morris et al., 1979, ix). Morries et al. suggest that one possible reason for this is that gesturing has been considered subsidiary to verbal communication. Another reason might be that the multidimensional nature of gestures makes them harder to study and write about than other aspects of linguistics.

A similar situation can been seen in music research, particularly in the so-called Western art music tradition, in which music has been studied more as written text than as organised sound or bodily sensation. While there has been a slight shift towards researching the sonorous aspects of music, an *embodied* perspective still lacks in much of music research. This is strange, considering the fact that music is, and has always been, a movement based phenomenon. Furthermore, talking about meanings and emotions without referring to the human body seems to contradict research showing that actions are one of the most important factors when it comes to conveying feelings and emotions (Darwin, 1872), not only in humans but also in other species (Pollick and de Waal, 2007).

Body movement, or more specifically *gesturing*, has received a growing interest in linguistics, psychology and behavioural science over the last decades, with researchers like Adam Kendon, David McNeill and Susan Goldin-Meadow at the fore. Similarly, there have been some researchers focusing on body movement in music, for example Alf Gabrielsson, Eric Clarke and Jane Davidsson, but this has often been from a music psychological or music performance point of view. Recently, however, the interest in this field has increased, as for example the two interdisciplinary conferences on *Music and Gesture*[3] have shown (Gritten and King, 2006). Both these conferences drew researchers from several different fields of study, including musicology, music performance, composition, psychology, cognitive neuroscience and various types of engineering disciplines.

However, despite this growing interest in music-related movement, working on the relationships between body movement and music still seems radical and far from mainstream musicological discourse. This is the case, even though it has long been argued that music should be studied as a bodily activity. As Manfred Clynes wrote 25 years ago:

> There is much music in our lives — yet we know little about its function. [...] [T]he coming years are likely to see much progress towards providing answers and raising new questions. These questions are different from those music theorists have asked themselves: they deal not with the structure of a musical *score* [...] but with music in the flesh: music not outside of man to be looked at from written symbols, but music-man as a living entity or system. (Clynes, 1982, vii)

Considering the enormous progress that has occurred in the field of cognitive neuroscience since he wrote this, his words seem truer than ever.

[3]Organised at the University of East Anglia in 2003 and at the Royal Northern College of Music in Manchester in 2006.

1.2 Research Design

The main topic of this dissertation is music-related movement, and the main research question is:

- How can we define, describe, analyse and represent music-related movement?

From this main question, a number of sub-questions follow:

- What is music-related movement?
- Which terminology should we use when describing music-related movement?
- What are the relationships between music-related movement and musical sound?
- Which theories, methods and tools are available for studying music-related movement?

As these questions show, there is a great deal of unexplored territory surrounding the main topic, and it has therefore been necessary to carry out research of a rather basic character. The research questions formed the foundation of a set of aims and objectives formulated for my dissertation. These are to:

- clarify the terminology and definitions used to describe music-related movement.
- carry out observation studies of music-related movement.
- construct digital musical instruments to test relationships between movement and sound.
- develop software tools to study music-related movement.
- formalise descriptors for realtime control of digital musical instruments, and for storing music-related movement data and associated analysis data.

1.3 Tradition

This dissertation is founded on a tradition of *systematic musicology* dating back to the pioneering work of a number of German researchers in the 19th century (Leman, 2007), including the psychoacoustic discoveries by Hermann von Helmholtz (1821-1894), the cognitive psychology of Wilhelm Wundt (1832-1920), and the empirical psychology of Franz Brentano (1838-1917). Their ideas formed the basis of the *Gestalt* school of psychology, and *phenomenology* as developed by Edmund Husserl (1859-1938) and others.

Many of these ideas were also important for musical exploration by some composers in the 20th century, particularly the French composer Pierre Schaeffer (1910-1995).

Schaeffer was not only the father of *musique concrète*, but also developed an extensive theory of music focused on the concept of *sonorous objects*[4] (Schaeffer, 1966). It is Schaeffer's thoughts on the phenomenology of music, combined with Gibson's (1979) *ecological* approach to cognition, which may be seen as the immediate inspiration for this dissertation. Using Schaeffer's seemingly disembodied theory within an embodied framework may seem like a contradiction, since one of his main arguments was that sounds should be evaluated disregarding their context. However, I build this on the work of Rolf Inge Godøy, who has suggested that sound-related movements share many properties with the corresponding sounds, and that Schaeffer's theories may be extended to study *gestural-sonorous objects*, meaning the phenomenological relationships between movement and sound (Godøy, 2006, 150).

I also see my work as being part of the interdisciplinary field of *music technology*. Music technological research has grown rapidly in recent years, and is currently encompassing a number of different disciplines. As indicated in Figure 1.1, I see my dissertation somewhere between many of these disciplines: science versus art disciplines on the one hand, and the humanities versus technology on the other. These axes also reflect my personal educational background, since I originally studied physics, mathematics and computer science, before moving to musicology, music performance, multimedia art and music technology. In this dissertation project I have tried to draw on my experience from all of these fields.

Working between various disciplines and traditions has been both an advantage and a challenge in my dissertation project. It has been exciting to engage in interdisciplinary discussions, and enriching to combine ideas, theories and methods from many different disciplines. But manoeuvring through terminology and established traditions in such a diverse research landscape is also daring. Even though engineers, psychologists, composers and musicologists work on many of the same issues, there are many fundamental differences in how the material is approached. Thus, as will be evident throughout this dissertation, I have tried to work with a vocabulary that may be understood in all these disciplines. That is also the reason why a large portion of Chapter 4 is devoted to discussing why I have abandoned the term *gesture* when referring to music-related movement.

My approach in this dissertation project has been to study music as an embodied *process*. Furthermore, I have been focusing on short fragments of such an embodied process, typically in the range of 2-5 seconds. This is the approximate duration of sonorous objects, in the research tradition of Schaeffer, and also the duration of many body movements. It is not a coincidence that this is also the limit of our *working memory*, the running "time window" within which we experience the world. In my Master's thesis on *short-term music recognition* I discussed the importance of short musical fragments for our ability to recognise a song in less than one second (Jensenius, 2002). In this dissertation I have continued to study short-term processes, but this time focusing on the importance of body movement for our musical experience. To limit the project, I have

[4]The *sonorous object* (*l'objet sonore*) was presented in Schaeffer's *Traité des objets musicaux* (1966), and is a fragment of sound, usually lasting a few seconds, that is perceived as a holistic entity (Schneider and Godøy, 2001, 14).

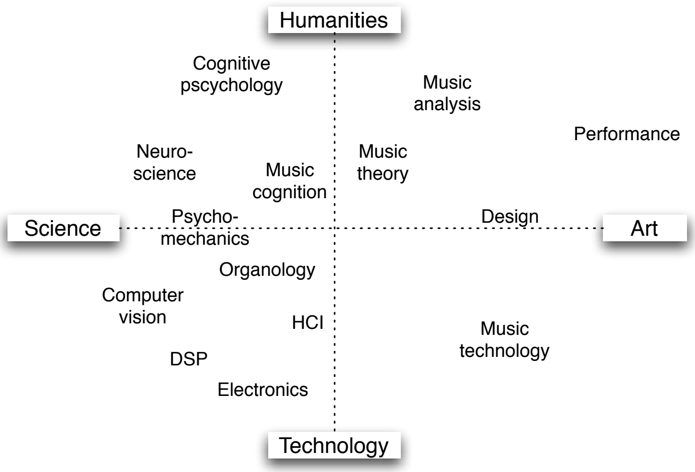


Figure 1.1: *This interdisciplinary dissertation project is placed between a number of different fields, which may be seen as defining four different "polarities".*

had to leave out contextual aspects, for example how the musical fragments I am studying fit into the context of songs, pieces or works, or the influence of society and culture on our experience of such fragments.

Although this study is biased towards so-called Western music traditions, there has been no attempt to limit the scope to a specific musical genre or socio-cultural context. Rather, I believe music-related movement is a key component in all musical cultures, and that our life-long experience of relationships between objects, actions and sounds is an essential part of our bodily and mental capacities. This is because we live in a world where our body and the objects we interact with are subject to mechanical and biomechanical laws, and their *psychomechanical* and *psychoacoustical* counterparts are a fundamental part of being human. A premise for this dissertation is therefore that relationships between objects, actions and sounds also govern our experience of music. This is the reason why I study music through movement.

1.4 Outline of Dissertation

As the visual overview (page xii) shows, this dissertation project has been manifold and has covered several theoretical and practical topics that are linked in various ways. For this reason I thought about writing the dissertation as a hypertext document that could be read non-linearly. I ended up with a traditional monograph, but the text may still be seen as the result of a non-linear process. For the sake of clarity, the dissertation is organised around three main parts: *theory*, *exploration* and *development*.

Part one: Theory

The chapters in the first part presents the theoretical basis for this dissertation, define relevant terms and topics, and outline some of my own theories and models on the main research topic.

Chapter 2 gives an overview of the emerging field of *embodied music cognition* based on *ecological psychology*. The concepts of *multimodality*, *embodiment* and *action-perception loop* are outlined, and it is argued that an action-perception approach to music cognition may force us to revise our understanding of a separation between the performance and perception of music.

Chapter 3 outlines the concepts of *action-sound relationship* and *action-sound coupling* (i.e. a natural action-sound relationship) and discusses our perception of such couplings and relationships. Examples of how *artificial action-sound relationships* are used creatively are given, and it is argued that knowledge about ecologically experienced action-sound couplings should be used when creating artificial action-sound relationships.

Chapter 4 starts with an overview of how the term *gesture* is used in various disciplines, and explains why this word is used sparingly in this dissertation. The rest of the chapter deals with an overview of different types of *music-related movement*, outlining a typology and framework for studying such movements.

Part two: Exploration

The chapters in this part present various explorations of relationships between movement and music, both in performance and perception. These studies may be seen as practical explorations of topics discussed in the theoretical chapters.

Chapter 5 presents three observation studies of *movement-music correspondences*, including a study of *air instrument* performance, *free dance* movement to music, and *sound-tracing* using a graphical tablet. These studies showed that both novices and experts alike seem to associate various types of body movement with features in the musical sound.

Chapter 6 focuses on the creation of artificial action-sound relationships in digital music controllers and instruments. Examples are given of various prototype music controllers developed during the project, including using game controllers in music performance, the *Cheapstick*, *music balls*, and the *Music Troll*. Examples are given of how the devices have been used in educational and performance settings, and they are evaluated in context of the action-sound relationships afforded by the devices.

Part three: Development

These three chapters present the development of software, methods and formats to solve some of the problems and needs that were identified in the exploration chapters.

Chapter 7 presents the *Musical Gestures Toolbox* (MGT), a collection of software tools that help analysing music-related movements. MGT is developed for and with the graphical programming environment Max/MSP/Jitter, using the modular framework Jamoma. Examples are presented of how applications built with MGT have been used for analysis and performance.

Chapter 8 starts with an overview of various types of computer visualisation of movement. This is followed by a presentation of my explorations of using *keyframe displays*, *motion history images*, *motiongrams* and *videograms* to help visualise and navigate in video material from our observation studies, music videos and for medical experiments.

Chapter 9 discusses the need of formats and standards for handling music-related movement data. This is followed by a presentation of the current state of development of the *Gesture Description Interchange Format* (GDIF), and how it may help to solve some of the challenges in streaming and storing music-related movement data.

Conclusion and Appendix

Chapter 10 provides a summary of the dissertation, presents some general reflections on the topic, and proposes several directions for future research.

Appendix A lists the contents of the Musical Gestures Toolbox, and my contributions to the Jamoma framework within which it is developed.

CD-ROM

The accompanying CD-ROM contains the audio and video examples used in the dissertation. Reading the electronic (PDF) version of this document, these audio and video examples can be accessed directly using the hyperlinks in the text.[5] The media files can also be accessed from the hyperlinked *index.html* file on the CD-ROM, or by manually selecting the files in the appropriate folders.

The CD-ROM contains the software and Max *patches* presented in the dissertation. A Max *patch* is a "program" developed in the graphic music programming environment Max/MSP/Jitter, and requires the full version of Max or Max Runtime,[6] as well as Jamoma[7] to be installed. More information about installation and setup is available on the CD-ROM.

[5]This only works with PDF applications by Adobe.

[6]A 30 day demo of the full version of Max/MSP/Jitter, and the free Max Runtime, can be downloaded from Cycling '74 (http://www.cycling74.com/downloads).

[7]The latest developer and stable versions are included on the CD-ROM.

Part I

Theory

CHAPTER 2 Embodied Music Cognition

In Africa the most important rhythms tend to be played the least. They don't have to since they are already in the body.

John Collins (2006a)

This chapter presents various topics relevant to the emerging field of *embodied music cognition*, based on *ecological psychology*. Embodied music cognition may be seen as an alternative to cognitive theories upholding a separation between mind and body, and focuses on the *embodied* and *multimodal* capacities of human cognition. The chapter ends with a discussion of theories related to perception as an active process.

2.1 Introduction

An *embodied* approach to cognition takes the body with its perceptual and motor capacities as the point of departure. This is to say that our mental processing is inseparable from the body, and that we may think of a matrix within which memory, emotion, language and all other aspects of life are meshed (Thelen, 1995, xx). As such, an embodied approach may be seen as an extension to *phenomenology*, which suggests that meaning is to be found in the world in which we act rather than in abstract concepts and ideas (Dourish, 2001). Historically, for instance in the work of Schaeffer (1966), a phenomenological approach to the experience of music has been concerned with the *sounding qualities* of music. An embodied music cognition extends this by suggesting that the whole body plays an important role in our experience of music.

An embodied perspective opposes traditions which uphold a separation of body and mind, a dichotomy which seems to have been particularly prevalent in the practice and thinking of so-called Western art music. In an analysis of the "rituals" of the classical concert hall, Small (1998) argues that such music is surrounded by a set of strict behavioural laws that are imposed on both performers and perceivers. In such a setting the audience is expected to sit quietly and *listen* with as little movement and emotional expression as possible. But, even with such rules suppressing the body and its movement, we cannot escape the fact that Western art music is, by necessity, an embodied activity and should be evaluated as such. In fact, even when studying only the *scores* of Western art music, there are numerous references to body movement in all the metaphors used to describe the intended qualities of the music. For example, expressions like *staccato*, *ritardando* and *lento* may be seen as indicative of the composer's imagined movement qualities, which again may be traced back to the sensation of music as an embodied activity.

The importance of the body on musical experience has long been suggested in *ethnomusicology*, for example in Alan P. Merriam's (1964, 32) model of understanding music at three levels: musical *sound* itself, *behaviour* in relation to music, and the *conceptualisation* of music. Such an embodied perspective may be more of a necessity when studying cultures in which there might be no clear separation between *perceiving* and *performing* music, or where the same word may denote *playing*, *dancing* and *singing* (Bjørkvold, 1989, 58-61). However, I believe that all musical thinking would benefit from using such an embodied perspective.

2.2 An Ecological Approach

The term *ecological psychology* was coined by James J. Gibson (1904-1979), a psychologist working mainly on visual perception and whose ideas have influenced a whole school of thought. Gibson's main contribution was to suggest that we do not only see the environment with our eyes, but "with the eyes in the head on the shoulders of a body that gets about" (Gibson, 1979, 222). He thereby stresses the importance of looking at our cognitive system not as a separate mental process, but rather that we are related to our environment as parts of a whole ecological system. Furthermore, at the core of his thinking is the idea of perception being an active process, in which the body, the mind and the environment are in constant interaction.

Inspired by Gibson's ideas, Eric Clarke (2005) suggests that an ecological approach to music perception should be based on *ecological listening*. This, he suggests, involves taking our everyday listening and the capacities of our auditory system as the point of departure for understanding meaning in music. The human auditory system has developed through evolution as a tool to help us live in the world. For example, our ability to discriminate between sound events, what Bregman (1990) calls *auditory scene analysis*, may be seen as a result of such an evolutionary process in which we have adjusted our hearing to what is presumably important for our survival. It is this evolutionary developed auditory system that is the basis for how music is composed, performed and perceived. Thus, our perception of musical sound should be studied with reference to the capacities (and limitations) of the auditory system.

2.2.1 Multimodality

A fundamental aspect of ecological psychology is that our perception is inherently *multimodal* in nature. In fact, multimodality seems to be the norm rather than a deviation in human perception in general (Berthoz, 1997). Multimodal perception is here used to suggest that all our *senses* and *modalities* (as summarised in Table 2.1) are at work when experiencing the world, and that they mutually influence each other.

Table 2.1: *The human senses and their corresponding sensory organs and modalities (Schomaker et al., 1995, 5).*

Sensory perception	Sensory organ	Modality
Sight	Eyes	Visual
Hearing	Ears	Auditive
Touch	Skin	Tactile
Smell	Nose	Olfactory
Taste	Tongue	Gustatory
Balance	Organ of equilibrium	Vestibular

Let us create a little thought experiment of the interdependence and inseparability of our modalities. Think of someone dropping a glass on the floor behind you. Most likely you will turn to inspect visually what happened when you hear the sound of the broken glass. Already at this point three modalities have been at work: the *auditory*, *visual* and the *vestibular*. First, the omni-directional nature of your hearing recognises the sound of broken glass, and guides your turning so that the directionally limited sight can see where the sound is emerging from. Here the vestibular modality is important for keeping you in balance and informing about the orientation and movement of your head and body. This seemingly simple example shows some of the richness of our multimodal capacities.

An interesting aspect of our multimodal perception is that we rarely pay attention to which modality is at work, and it often does not even matter. For example, whether we recognise a person by hearing, sight, smell or touch is less important than the fact that we recognise who it is. We can choose to focus on one of our senses if we like, but normally we do not think about which senses and modalities are at work. Multimodality does not only happen in perception, but can also be seen in how effortlessly we move between modalities in our interaction with others. An example of such *cross-modal* behaviour is that of answering a spoken question with a hand gesture. Trevarthen (2000) found that also infants and children often show such cross-model behaviour. Thus, it seems like multimodal perception and cross-modal behaviour are fundamental capacities of the human cognitive system.

Multimodal perception and cross-modal interaction also seems to be fundamental parts of our music experience. A performer is obviously very much dependent on hearing, and on the haptic and visual feedback of the instrument. All musicians are also heavily reliant on the vestibular modality to keep in balance while playing. Some musicians also need other modalities, for example wind instrumentalists who rely on their gustatory capacities to wet the reed. When it comes to music perceivers, they rely mainly on their sight and hearing to perceive musical sound and movement, but the smell of the concert environment, the tactile feel of the chair, and the sense of vestibular activity may all play an important role in the perception of music. Even today, when people's experience of music is often via some sort of digital medium, music continues to be a multimodal experience through *audio-visual* integration in various *multimedia* devices, such as TV, computers and media players. For this reason, both performance and perception of music should be studied as a multimodal phenomenon.

2.2.2 Classicist and Cognitivist Traditions

Embodied music cognition may be seen as an alternative to several other "schools" of philosophy, cognitive science, and computer science, such as *rule-based* artificial intelligence (AI) and *connectionist* paradigms. Rule-based AI models, which can be seen as a classical information-processing paradigm, has been popular in symbolic and rule-based music theories since they are easy to implement and test in various computational models, such as the *generative theory of tonal music* (GTTM) (Lerdahl and Jackendoff, 1983), the *implication-realisation* model (Narmour, 1990), *melodic peaks* (Eitan, 1997), *melodic similarity* (Hewlett and Selfridge-Field, 1998), and *melodic expectation* (Huron, 2006). Such systems may reveal many aspects of musical structure and form across large collections of data, but often fail to locate and explain structures that fall outside of the rules being used in the model.

An alternative to rule-based AI was offered with *connectionism*. Connectionism quickly became popular after *artificial neural networks* (ANNs) were proposed by, amongst others, Smith (1980) and Hopfield (1982). The idea of ANNs was to create a model based on how the brain's processing was believed to function; a network of connected *nodes* (or *neurons*). In such a network the changing strength of the connections (the *weights*) between the neurons would be the basis for learning in the system. Many different learning models have been suggested, one of the most influential being the *backpropagation* algorithm developed by Rumelhart et al. (1986). This is a so called *supervised learning* algorithm, meaning that both the input and the output in the system is known. The backpropagation algorithm is based on running in cycles, with iterative adjustments of the weights between the nodes in the network. The aim of the algorithm is to adjust the weights so that the output of the network matches the input.

One of the strengths of the backpropagation algorithm, and ANNs in general, is that the networks may be used to learn multidimensional structures without relying on any rules defining the relationships between the input and output data. This seems interesting for musical applications, and has for example been explored in timbre classification by Wessel et al. (1998). In my MA thesis (Jensenius, 2002), I built on Wessel's method using the backpropagation algorithm for training ANNs to recognise relationships between a low-dimensional input set (the fundamental frequency and loudness of a sound) and a multidimensional output set (the 60 first partials of the sound). The end result was a trained network that could play a rich set of timbres by inputting only a frequency and a loudness. Unfortunately, the method was not particularly efficient, as it took around an hour to train the network with only one short sound.

Another influential neural network algorithm is that of *self-organising maps* (SOM) (Kohonen, 1990, 2001). This is an *unsupervised* learning algorithm, meaning that only the input data to the network is known. The SOM algorithm may classify a complex input signal by reducing the multidimensionality of the input data to a low-dimensional output data set. The learning in a SOM is done by adjusting the weights in the network based on features in the input data only, so no prior knowledge about the outcome is necessary. As such, SOMs work well for classifying multidimensional material where the output of the data set is unknown, and has been used in various types of music research, for example on music and schema theory (Leman, 1995), and melodic expectancy (Krumhansl et al., 2000).

The difference between rule-based and connectionist systems has been described as analogous to the difference between computers and humans. While the computer is based on serial processing and a "brute-force" approach to problem solving, the human brain is believed to be based on distributed and parallel processing in large neural networks. However, the difference is not only in the type of processing, but also in the type of answers provided by such systems. A rule-based system may be able to provide a correct and precise answer, given the necessary rules and time to carry out the task. A distributed system, on the other hand, may not be able to give a precise answer, but will usually be better at finding relationships between data, and provide relative answers. As such, this seems to be more similar to the human capacity of reasoning and finding solutions to complex problems.

2.2.3 Multimodal Processing

One problem with both rule-based and connectionist systems is the focus on data processing as logical reasoning based on a passive retrieval of sensory information. If this was how human cognition worked, why do computers have problems carrying out tasks that all humans can do effortlessly? Even though today's supercomputers may approach, or even supersede, the brain's raw processing power,[1] we humans are still far better at carrying out many complex tasks. This is probably because of our embodied cognition, and thus also our cross-modal capacities.

An example of how multimodality helps our processing is our ability to hear separate events in a complex soundscape, or what is often referred to as the *cocktail party effect*. Mitchell et al. (1969) showed that this to a large extent was due to our fine auditory sensitivity of direction, based on the slight differences in time between when sounds reach our two ears. It has later been shown that our ability to combine visual and auditory modalities considerably improves our processing capacity (Allport et al., 1972; Frick, 1984). It has also been found that such gains in processing seem to be the strongest when modalities represent different, but related, segments of information (Sweller, 2002).

[1]This is a highly debated topic, and is also not easy to prove as long as it is based on speculations about the number of neurons in the brain and the speed at which connections work (DeGroot, 2005).

Our multimodal capacities may also lead to cognitive conflicts, as demonstrated in the *McGurk effect*. McGurk and MacDonald (1976) carried out experiments where the combination of different visual and auditory stimuli resulted in either stimulus being chosen and overriding the other, or in some cases even a third percept showing up. The latter was shown in famous experiment where subjects heard the spoken word "Ba" while at the same time seeing the video of a mouth saying "Ga". Interestingly, most of the subjects reported that they saw/heard "Da", a combination of the two stimuli.

Similar types of audio-visual integration have been found in music. In a study of performances by clarinettists, Vines et al. (2005) showed how visual information (video of the performer) enhanced the sense of phrasing and anticipation of changes in emotional content in perceivers. Thompson et al. (2005) report similar findings of the visual impact on the perception of music based on case studies of recordings of popular artists. In a series of experiments they have also shown that audio-visual integration seems to occur preattentively (Thompson and Russo, 2006). There is therefore evidence that supports the idea that multimodality is not only a reality, but rather a main feature, in music perception.

2.2.4 Affordance

One fascinating aspect of our embodied cognitive capacities is that of seeing an object and recognising the *function* of the object. For example, we have no problem seeing that a chair standing in front of a table is, in fact, a chair standing in front of a table. This type of identification we can do even if we have never seen the chair before, and even though it may be of any shape, size and colour. Many computer programs, on the other hand, would probably have great difficulties separating the chair and the table, due to overlapping visual elements (for example the large number of legs). The reason we are so good at solving such *visual scene analysis* tasks may be our capacity to identify *motor programs* associated with objects (Rosch et al., 1976). One of the most important functions of a chair is that of being something for sitting on, and thus the motor program of "sitting" is essential for categorising an object as a chair.

The aspect of an object which makes it obvious how it is to be used is what Gibson termed the *affordance* of an object (Gibson, 1977). For example, a chair may afford sitting, but it may also afford being a table, being thrown at someone, or used as a percussion instrument. As such, an object may have multiple affordances, and these affordances are dependent on the individual, the culture, and the context within which the object is located and used. It may be argued that some affordances are more basic than others, but most affordances are probably learned by living. In our daily interaction with objects in the world, we continuously expand our knowledge about the affordances of the objects. But we also seem highly capable of coming up with new types of usage of objects. For example, even though we have learned that a chair is made for sitting, we may easily create a long list of other types of usage that the chair affords. Thus, imagination and creativity also seems to be important aspects of our embodied cognitive system.

2.2.5 Emotion and Expression

Yet another reason for employing an embodied perspective to cognition is to better understand our perception of *emotion* in music. Several studies, for example by Gabrielsson and Juslin (1996) and Dahl and Friberg (2004), have reported that the communication of emotions in music is very much dependent on the *expressiveness* of the performer. Expressivity is here used to denote the ability to convey feelings and emotions, and this capacity is something which is prevalent in music performance.

However, even though everyone seems to agree that expressiveness is a key component in music performance, a recent study by Laukka (2004) reports that music performance teachers spend relatively little time on expressivity in music education. Similarly, Juslin and Sloboda (2001, 3) argue that there has been comparably little research on emotion in music in the 50 years since Meyer (1956) published his classic *Emotion and Meaning in Music*. This is exemplified by the fact that the two most seminal books in their fields, *The Psychology of Music* (Deutsch, 1999) and the *Handbook of Emotions* (Lewis and Haviland-Jones, 2000), have no chapters on emotion and music, respectively.

The situation seems to be changing, though, and the extensive literature reviews by Alf Gabrielsson (1999; 2003) show that there has been a growing interest in research into music performance and musical expressiveness during the last century. Particularly psychologists, and more recently computer scientists and engineers, seem to be interested in understanding more about music performance, emotion and expression. Psychologists and neuroscientists use music and musical expressiveness as examples of the exceptional capacities of human perception and cognition. Engineers, on the other hand, are often driven by the goal of creating computer systems that can perform and behave "expressively".

The lack of expressiveness is obvious when playing back symbolic music notation (e.g. MIDI files) on a computer. The result is a "perfect" rendition of the musical score, but one which sounds mechanical and "lifeless" compared to human performances. Computer renditions of music clearly show the importance of a human performer interpreting the score and "adding" expressivity to the performance. As Juslin (2003, 274) argues, it is the expressivity of the performance which motivates us to experience music in a concert hall rather than listening to a rendition played by the computer. Creating computer systems that model human expressiveness is therefore an interesting approach to understanding more about music performance and perception.

A number of computer models for musical expressiveness have been presented over the years, including the rule-based performance model *Director Musices* from KTH (Friberg et al., 2000), the tempo/dynamics model by Todd (1992), the artificial life and evolutionary models by Todd and Miranda (2004) and the rule-based (by machine learning) system by Widmer (2002). These models are all based on different methods, but they have a common aim in creating a more expressive rendition of the musical sound. It is interesting to see that these models are also taking human movement into account when creating the musical "expressiveness". Thus, it seems like an embodied approach is important for being able to convey expressions and emotions in music.

2.3 Perception – Action

An important aspect of Gibson's (1966; 1979) ecological psychology, is the close relationship between perception and action. This has been further developed in what is often called *motor theories of perception*, which suggest that we make sense of what we hear because we guess how the sounds are produced. This was first proposed in the *motor theory of speech perception*, claiming that language perception and language acquisition are based on learning the *articulatory gestures* of the human vocal apparatus (Liberman and Mattingly, 1985). Here articulatory gesture is used to denote a *phonological unit* or what Browman and Goldstein (1989) call the basic "atoms" out of which phonological structures are formed. Motor theories further postulate that the structures of language (phonological, lexical and syntactic) are based on the pre-existing neural systems which had evolved to control body movement. This implies that the perceptual mechanisms of the vocal apparatus were in place long before language entered the stage, and that speech and language adapted to phylogenetically older structures rather than the other way round (Lindblom, 1991).

The motor theory of speech perception and other similar theories, e.g. the *direct realist theory* (Fowler, 1986), have met much criticism throughout the years, such as illustrated by the dispute between Fowler (1996) and Ohala (1996). Ohala and other critics argue that speech perception can be understood through acoustic cues alone, and that involving a cross-modal perspective is "extravagant" (Ohala, 1996, 1724). This extravagance is what makes animals survive, Fowler (1996, 1740) replies, and argues that speech perception should be considered no different to other types of perception.

The motor theories of perception received renewed interest after a series of neurocognitive experiments by Gallese et al. (1996) found the so-called *mirror neurons* in the brain of monkeys. These experiments showed that a particular set of neurons were activated both when the monkey performed a *goal-directed action*, and when it observed a similar action being performed by the experimenter. Applied to us humans, this suggests that we mentally "simulate" the movements and actions we see, which may be an explanation for how we manage to learn by only watching others carry out actions. This type of mental "replay" of actions may also be seen as an explanation for our ability to make predictions. Wilson and Knoblich (2005) argue that we mentally simulate our own actions in parallel to the physical actions carried out, and that such imitative motor activation feeds back into the perceptual processing and allows us to adjust our actions accordingly. This supports the *sandwich* conception of mind suggested by Hurley (1989, 1998, 2001), as illustrated in Figure 2.1. This model suggests that cognition is "sandwiched" in between perception as input from the world to the mind, and action as output from the mind to the world, or a so-called *action-perception loop*.

Though the early experiments on mirror neurons were of goal-directed actions observed in monkeys, Gallese et al. (1996) also briefly suggested that mirror neurons may be active in the speech perception of humans, as is suggested by Liberman and Mattingly (1985). Such a link between speech and action was, indeed, shown in PET[2] studies of

[2] *Positron Emission Tomography* (PET) is a nuclear medical imaging technique producing three-dimensional

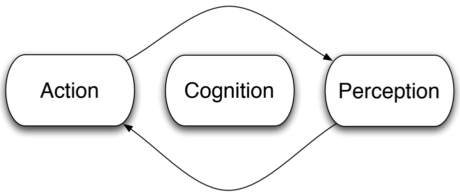


Figure 2.1: *My illustration of Hurley's* sandwich *conception of mind, suggesting that cognition is between perception and action.*

humans by Rizzolatti and Arbib (1998), where motor areas of the brain were activated during speech perception. Such an *auditory-motor interaction* has also been shown in fMRI[3] scans of humans carried out by Hickok et al. (2003). Here similar neuronal activity was found when subjects only listened to speech or music, and when they covertly "hummed" the auditory stimuli.

Another set of studies has shown the existence of so-called *audiovisual mirror neurons* in the brain. This was found in experiments on monkeys, where the same neuronal response was measured whether the monkey could both see and hear a sound-producing action or only hear the sound of the same type of action (Kohler et al., 2002). A follow-up study showed that for half of the tested *audiovisual mirror neurons*, the response did not differ significantly whether the action was heard, seen or both heard and seen (Keysers et al., 2003). Some of these early experiments on audiovisual mirror neurons were criticised for being based on only short everyday sounds, for example hand clapping or knocking, but there has since been several studies of more complex sound material. For example, Haueisen and Knösche (2001) used MEG[4] techniques to show that pianists show involuntary motor activity while listening to well-trained piano music.

A more recent study by Lahav et al. (2007) reports that similar motor activity may also occur in non-musicians. They carried out an experiment where non-musicians were practising to play a melody for a couple of weeks (learned by ear). After this practise, fMRI scans showed that motor areas in the brain were active when the subject listened to the same melody as had been practised, also in the cases when the person was not moving. This activation would only occur when listening to the same melody the person had practised; variations on the melody resulted in little or no activation. This follows the idea of an action-perception loop presented earlier, and Lahav et al. suggest that we may think about an *action-listening* process as illustrated in Figure 2.2. Here the idea is that we constantly and continuously simulate the sound-producing actions related to the sounds we hear. Lahav's model fits with Godøy's (Godøy, 2003) idea of music perception and cognition being *motor-mimetic*, i.e. that we mentally simulate the sound-producing actions when we listen to music. This forms the basis for my understanding of *action-sound relationships* which will be discussed in Chapter 3.

images of functional processes in the body (Purves et al., 2001, 28).

[3]*Functional Magnetic Resonance Imaging* (fMRI) is another functional brain imaging technique, which displays functions in the brain by detecting increased blood flow to activated areas (Solso et al., 2005, 56).

[4]*Magnetoencephalography* (MEG) is a technique for measuring the weak magnetic fields present outside the head due to electrical activity in the brain, using a *superconducting quantum interference device* (SQUID) (Tanzer, 2006). The technique is completely non-invasive and harmless and makes it possible to measure as precisely as with *electroencephalography* (EEG), but without any physical contact of sensors.

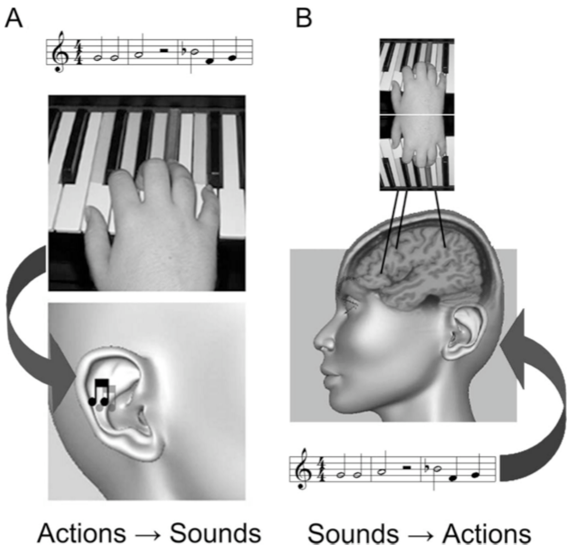


Figure 2.2: *Lahav's (2007, 309) model of* action-listening, *where the memorisation of performance of actions and sounds (A) results in motor activity in the brain when only listening to the same music (B).*

2.4 Summary

In this chapter I have tried to provide a brief overview of some topics that are relevant to an *embodied* approach to music cognition. One premise for such an approach is the idea of music being a *multimodal* phenomenon, and that our cognition of music is based on the capacities and limitations of our body in relation to the environment. Such a view is an alternative to music theories based on a mind/brain separation, and classicist and cognitivist views of human cognition.

An embodied approach suggests that there is a coupling of human action and perception. While this often was regarded a speculative idea some decades ago, it is now supported by quite a large body of research in cognitive neuroscience. A number of studies show that motor areas of the brain may be triggered by watching someone else carrying out an action, by thinking about the action, or even only by hearing the sound of the action. These findings may explain why we are able to "hear" the sound of a sound-producing action we only see, and "see" the sound-producing action of a sound we only hear. Such relationships between actions and sounds are the basis for what I call *action-sound couplings*, which is the topic of the next chapter.

CHAPTER 3 Action – Sound

It is easy to play any musical instrument: all you have to do is to push the right keys at the right time and then the instrument will play itself.

J.S. Bach

This chapter discusses the concepts of *action-sound coupling* and *action-sound relationship*, how they are constituted, and their importance for our music experience.

3.1 Introduction

Think about a glass falling towards the floor. While still in flight, you will imagine both the visual and auditory result of the glass breaking to pieces when it hits the floor. Knowing that the sound will be unpleasant, you may even try to cover your ears to protect your hearing. This simple example effectively demonstrates that we have a clear and immediate understanding of the relationships between actions, objects and the resultant sounds. To discuss such relationships I suggest two different terms:

- *Action-sound coupling*: this denotes relationships between actions and sounds where there is a mechanical and acoustical *coupling* between the action and the sound.
- *Action-sound relationship*: this denotes any type of relationship between action and sound, ranging from *natural* (i.e. an action-sound coupling) to *artificial*. Examples of artificially created action-sound relationships may be found in for example electronic devices.

3.1.1 Couplings versus Relationships

As the definitions imply, all action-sound couplings are also action-sound relationships, but only *natural* action-sound relationships are action-sound couplings. Note that I use these concepts to refer to the nature of the relationships between the objects and actions, not our perception of these relationships. Thus, in an acoustic piano we will find an action-sound coupling, while a digital piano is based on an action-sound relationship. This is the case even though we may be fooled to believe that the artificial action-sound relationship of the digital piano is a coupling. However, as will be discussed later, I believe that such an artificial action-sound relationship will never be as solid as an action-sound coupling (i.e. a natural action-sound relationship). For example, no matter how well the action-sound relationship of a digital piano is designed and constructed, it will fail the moment the power is turned off.

A number of concepts have previously been suggested for discussing the relationships between action and sound, including *gesture-sound relationships* (Dobrian, 2001), *gesture-sound links* (Godøy, 2004), *action-sound relation* (Luciani et al., 2005), *gestural-sonorous objects* (Godøy, 2006), and *auditory-gesture links* (Godøy et al., 2006b). There are several reasons why I have decided to use two new concepts rather than sticking with one of the above. First, as will be more evident after the discussion of *gesture* definitions in Chapter 4, I prefer to use *action* to describe coherent and goal-directed movements. Second, concerning the order of words in the expression, I find it important that *action* comes before *sound*, since actions necessarily happen before sounds appear. Third, I find it necessary to differentiate between natural (i.e. coupled) and artificially created action-sound relationships. As will be discussed throughout this chapter, this is because I believe that our life-long experience of action-sound couplings is essential for guiding our experience of artificially created action-sound relationships.

3.1.2 Object-action-object Systems

Another concept I will be referring to in this chapter, is that of an *object-action-object system*. As illustrated in Figure 3.1, such a system defines the objects and action(s) involved in producing a sound. In nature, the features of such a system are defined by the acoustical properties of each of the objects involved in the interaction (e.g. size, shape and material), and the mechanical laws of the actions that act upon them (e.g. external forces and gravitational pull). It is our life-long experience of acoustical and mechanical properties of objects and actions, that makes us able to predict the sound of an object-action-object system even before it is heard.

To use Gibson's (Gibson, 1977) term, we may say that an object-action-object system *affords* a specific type of sonic result. A falling glass will always result in the sound of a falling glass, and will never sound like a baby's scream or someone playing the violin. That is why I find it important to differentiate such action-sound couplings from other types of action-sound relationships, and to indicate that couplings are based on natural laws that we know will never fail.

An important aspect of Gibson's thinking was that an object may have multiple affor-

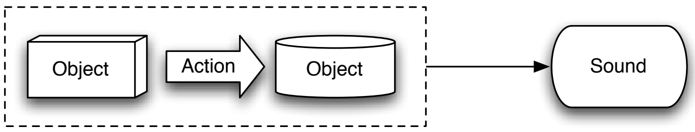


Figure 3.1: *The human capacity of imagining and predicting sounds seems to be based on our knowledge of the acoustical features of objects and the mechanical properties of actions, what I call an* object-action-object system.

dances. Similarly, an object-action-object system may also afford different sonic results. If the falling glass was made of plastic instead of glass, it would probably bounce off with a "plastic-like" sound, rather than break to pieces. Thus, if we thought the glass was made of glass, we would probably be surprised by a plastic-like sound, but it would still be a possible outcome if we were not absolutely certain that the glass was made of glass. However, if we heard a baby's scream when the glass hit the floor, it would be beyond the laws of nature and we would assume that the scream was coming from somewhere else. This is simply because a baby's scream is not within the span of sonic qualities afforded by the object-action-object system. I will call these multiple sonorous affordances the *action-sound palette* of the system. As shall be discussed later, I believe that the action-sound palette of a system is important for our perception of the action-sound couplings or relationships of the system.

The main argument I will be making in this chapter is that knowledge of action-sound couplings is deeply rooted in our cognitive system, and that this knowledge guides the way we think about other types of action-sound relationships, for example in *electronic devices* and *virtual realities*. Similarly, I will also argue that knowledge of action-sound couplings is vital for the understanding of the creative use of action-sound relationships in music, film, and other multimedia arts.

3.2 Action-Sound Couplings

To understand more about our perception of action-sound couplings we need to look closer at their origin and production. In this section I will mainly focus on sound-producing actions that are produced *intentionally*. When carrying out a music-related sound-producing action, for example hitting a drum with a hand, we may talk of an *action-sound chain* such as depicted in Figure 3.2. This chain starts with neurological activity in the brain, followed by physiological activity in a series of muscles, and physical (biomechanical) activity in limbs of the body. The interaction between the body and the object occurs as an *attack* when an element of the object (e.g. the drum membrane) is *excited* and starts to *resonate* creating the sound. As illustrated in Figure 3.2, there is multimodal feedback in all parts of the action-sound chain. This feedback may be seen as part of the action-perception loop, and the perception of the feedback allows for making corrections to the sound-producing action along the way.

Considering only the *attack* part of the action-sound chain, Godøy (2008) has sug-

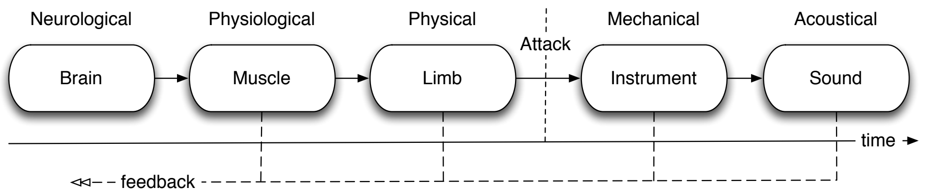


Figure 3.2: *The* action-sound chain *from cognitive process to sound, and with multimodal feedback in all parts of the chain.*

gested that this can be seen as an *excitation phase* with a *prefix*[1] and a *suffix*, such as depicted in Figure 3.3. The prefix is the part of a sound-producing action happening before the excitation, and is important for defining the quality of the excitation. The suffix is the return to equilibrium, or the initial state, after the excitation.

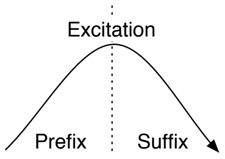


Figure 3.3: *A sound-producing action may be seen as having an excitation phase surrounded by a prefix and suffix. These three are closely connected and are important for both the performance and perception of the sound-producing action.*

The prefix, excitation and suffix are closely related both for the performance and the perception of a sound-producing action. Following the idea of our perception being based on an active action-perception loop as discussed in Chapter 2, a prefix may guide our attention and set up expectations for the sound that will follow. For example, if we see a percussionist lifting a mallet high above a timpani we immediately expect a loud sound. We will also expect the rebound of the mallet (the suffix) to match the energy level of the prefix, as well as the sonic result. As such, both prefixes and suffices help to "adjust" our perception of the sound, based on our ecological knowledge of different *action-sound types*.

3.2.1 Action-Sound Types

Combining terminology from Schaeffer (1966) and Cadoz (1988), we may talk about three different *action-sound types*, as presented in Godøy (2006):

- *Impulsive*: the excitation is based on a *discontinuous* energy transfer, resulting in a rapid sonic attack with a decaying resonance. This is typical of percussion, keyboard and plucked instruments.

[1]The prefix of a sound-producing movement has also been called *anticipatory movement* (Engel et al., 1997) and *preparatory movement* (Dahl, 2005).

- *Sustained*: the excitation is based on a *continuous* energy transfer, resulting in a continuously changing sound. This is typical of wind and bowed string instruments.

- *Iterative*: the excitation is based on a series of rapid and discontinuous energy transfers, resulting in sounds with a series of successive attacks that are so rapid that they tend to fuse, i.e. are not perceived individually. This is typical of some percussion instruments, such as guiro and cabasa, but may also be produced by a series of rapid attacks on other instruments, for example rapid finger movements on a guitar.

There are obviously several variants here, and many instruments can be played with both impulsive and sustained actions. For example, a violin may be played with a number of different sound-producing actions, ranging from *pizzicato* to bowed *legato*. However, the aim of categorising sound-producing actions into three action-sound types is not to classify instruments, but rather suggest that the mode of the excitation is directly reflected in the corresponding sound.

As shown in Figure 3.4, each of the action-sound types may be identified from the energy profiles of both the action and the sound. Here the dotted lines indicate where excitation occurs, and show that the action will always start slightly before the sound and also usually end before the sound (due to resonance and reverberation). Note that two action possibilities are sketched for the iterative action-sound type, since iterative sounds may often be the result of either the construction of the instrument or the action with which the instrument is played. An example of an iterative sound produced by a continuous action can be found in a cabasa, where the construction of the instrument makes the sound iterative. Playing a tremolo, on the other hand, involves a series of iterative actions, but these actions tend to fuse into one superordinate action. In either case, iterative sounds and actions may be seen as having different properties than that of impulsive and sustained action-sound types.

3.2.2 Perception of Struck Objects

As the example of the falling glass showed, we are capable of estimating both the timbral qualities and loudness of the sound to emerge based solely on the visual information of the glass and its trajectory in the fall. Similarly, we are also able to tell a lot about the objects and actions involved only by *hearing* the sound. This could be anything from the material of the glass, the surface it hit, the distance it fell, to whether it was dropped or thrown. Understanding more about our auditory perception of objects that are struck, or excited in other ways, is at the core of a field which can be called *sound-source perception* (Giordano, 2005), but which is also referred to as *ecological acoustics* (van der Veer, 1979) and *psychomechanics* (McAdams, 2000).

Not only do humans have the ability to quickly identify natural sounds that originate from a wide range of objects and interactions in the world (Lass et al., 1982; Ballas, 1993; Gaver, 1993a,b), but we are also very good at associating the sounds with various

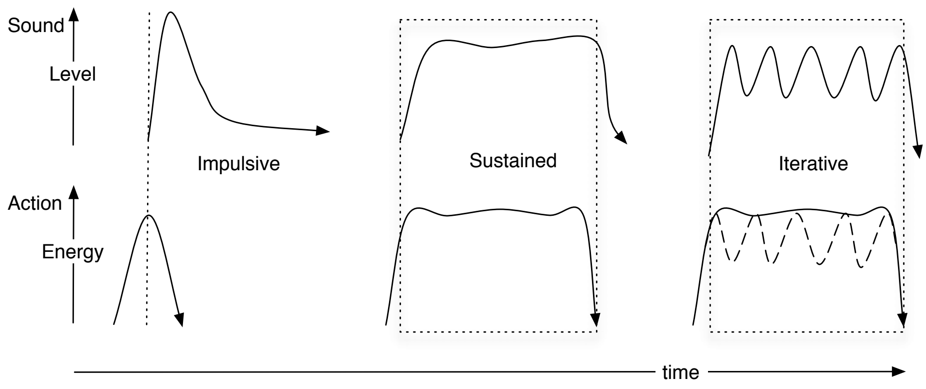


Figure 3.4: *Sketch of action energy and sound levels for the different types of sound-producing actions. The dotted lines/boxes suggest the duration of contact during excitation. Note that iterative sounds may be the result of different types of actions.*

features. For example, by listening to a voice we may be able to recognise the gender, age, origin, social status and physical health of the person speaking (Juslin and Scherer, 2005, 65). This ability to identify the properties of objects based solely on sound seems to be surprisingly reliable and accurate (Carello et al., 2005, 79), and has been shown to not only be robust during attentive listening, but also in the everyday perception of impact sounds and sources (Rocchesso and Fontana, 2003). From an ecological perspective this seems logical, considering that being able to characterise and identify sounds is important in the interaction with objects in their environment.

Several studies have investigated our ability to recognise qualities of specific types of sounds, such as bouncing and breaking bottles (Warren and Verbrugge, 1984), clapping hands (Repp, 1987), mallets striking metal pans (Freed, 1990), and walking (Li et al., 1991). Other studies have shown our ability to recognise physical properties based on hearing only the sound, for example the amount of liquid in a cylinder (Cabe and Pittenger, 2000), the direction of a sound source in three-dimensional space (Neuhoff, 2001), the length of objects (Carello et al., 1998), the hollowness of objects (Lutfi, 2001), the shape of objects (Lakatos et al., 1997; Kunkler-Peck and Turvey, 2000; Rocchesso and Ottaviani, 2001) and the gross material categories of struck objects of variable size (Giordano and McAdams, 2006).

When it comes to the question of which acoustical features are more important for our perception of the sound, Giordano (2005) argues that in his experiments the size and material of the objects were more important than the interaction between the objects. However, this conclusion is drawn from the results of experiments focusing on single impulsive sounds made by hammer strokes on various materials. I would assume that studies involving other action-sound types, as well as more complex interactions, such as in music, will place more perceptual emphasis on the interaction.

3.2.3 Perception and Cognition of Action-Sound Couplings

As the many studies of sound-source perception have shown, we have a remarkable ability to perceive qualities of the materials and actions associated with sounds only from listening. Godøy (2001) has suggested that our cognition of sound-producing actions may be founded on *motor images* of the excitation (what we do), and *materials images* of the resonance (the effects of what we do), as indicated in Figure 3.5. The term *image* is here used in the sense of *mental imagery*, denoting quasi-perceptual experiences that resemble perceptual experiences but with the absence of external stimuli (Thomas, 2007). Despite the name, mental imagery is used to describe mental representations of all modalities, such as "seeing in the mind's eye" or "hearing in the head". As such, *musical imagery* can be understood as "our mental capacity for imagining musical sound in the absence of a directly audible sound source" (Godøy and Jørgensen, 2001, ix).

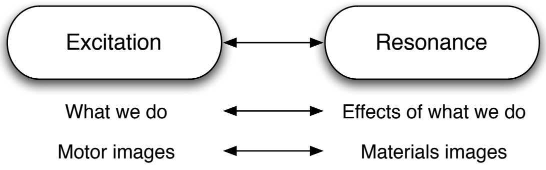


Figure 3.5: *Godøy's (2001, 242) schematic overview of the separation between* excitation *and* resonance*, suggesting that our cognition of these is based on* motor images *and* materials images.

In a discussion on reinterpreting Schaeffer's concept of the sonorous object, Godøy (2006, 150) argues that our perception and cognition of sonorous objects are closely linked to that of *gestural objects*. Godøy suggests a model where our mental images of musical sound is based on an incessant process of mentally *tracing* features of the sound when listening, or even when only imagining, music (Figure 3.6). This tracing can be imagining (or even carrying out) movements in for example hands or fingers in relation to onsets, contours, textures, etc. in the sound. From this it follows that the corresponding *gesture sensations* and multimodal *gesture images* have visual and motor components based on biomechanical constraints of what we imagine that our bodies can do.

In Godøy's model, listening can be understood as a continuous and multimodal process where the musical sound is *recoded* into multimodal *gestural-sonorous images*. This is not to say that we *see* an object, for example a piano, when hearing a piano sound, but rather that we get a sensation of the objects and actions involved in producing the sound. One important point here is the bi-directionality of the model, suggesting that the gestural-sonorous images are not only the result of listening to musical sound, but may also play an active part in the perception and cognition of the sound. The model suggests that our mental image of a sound will evoke the mental image of a sound-producing action, but also that the mental image of a sound-producing action will evoke a mental image of the corresponding sound. As such, the model can be seen as supporting the idea that our ecological knowledge of action-sound couplings is based on experience with both the action and the sound. Following this, I believe that these mental images of action-sound couplings guide our perception of artificial action-sound relationships.

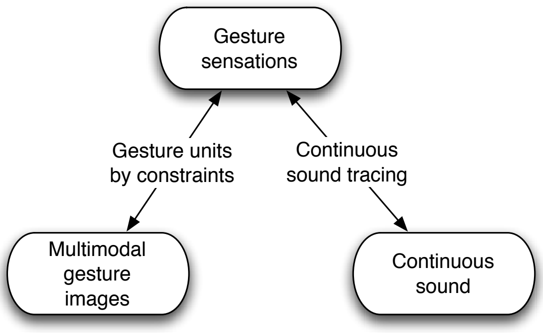


Figure 3.6: *Godøy's (2006, 150) tripartite model of relationships between multimodal gesture images and musical sound.*

3.3 Action-Sound Relationships

As discussed above, action-sound couplings are based on the mechanical and acoustical properties of the object-action-object system, and seem to be well established in our minds. However, there is an increasing number of action-sound relationships that are not natural (i.e. not coupled), but rather designed and created electronically. I choose to divide such artificial action-sound relationships into two categories:

- *Electronic devices*: this category includes everything from door bells to mobile phones and musical instruments. In such devices the sound is typically produced electronically and output from a speaker in or outside the device. The sound often results from a physical interaction between the user and some mechanical parts of the device, e.g. pressing a button.

- *Virtual realities*: this category includes TV, movies, computer software and computer games. Here the action-sound relationships are based on virtual object-action-object systems, even though these virtual systems may be controlled with a physical controller.

The main difference between these two categories of action-sound relationships is the type of *interaction*. In electronic devices the action-sound relationship is based on a direct physical interaction, for example pressing a button on a mobile phone. In virtual realities, on the other hand, the action-sound relationships are either not based on interaction at all, e.g. in TV or movies, or the interaction is indirect, e.g. using a joystick to control a car in a computer game. In the latter example the user will have a physical interaction with the game controller, but the action-sound relationship is based on an object-action-object system virtually constructed in the game. Obviously, there are examples where the difference between an electronic device and a virtual reality is not clear. For example, some computer games may be played with a physical controller similar to the virtual objects in the game, such as a car game controlled with a physical wheel controller and pedals.

In both electronic devices and virtual realities, sounds are used as a feedback modality to guide our interaction, and to enhance our experience with the device or reality. The corresponding action-sound relationships may be strong or weak, direct or indirect, or good or bad, for that matter, but they still influence our experience of the devices or virtual realities with which we are interacting. But how do we perceive such artificial action-sound relationships and how do they change our interaction with the device or reality?

3.3.1 Perception of Action-Sound Relationships

As argued in previous sections, our perception of action-sound couplings seems to be governed by our ecological knowledge of the actions and objects involved. This knowledge is in continuous development as we experience and learn new action-sound couplings. Similarly, as we surround ourselves with an increasing amount of technology, our knowledge of artificially created action-sound relationships is also in continuous development. However, even though action-sound relationships may become familiar to such an extent that they can feel natural, I believe that our perception of such relationships may never be as strong as that of an action-sound coupling.

The most important reason why I believe an artificial action-sound relationship never can be as strong as that of a coupling, is that we can never really trust the stability of an artificial action-sound relationship. For example, the action-sound relationship in an electronic door bell may always have been intuitive, direct and strong, but we still cannot be absolutely certain that this will always be the case. One day the power may be out, or the door bell may have been changed. If this happens we may be surprised, but it will not be an impossible outcome considering that we are dealing with an electronic door bell. On the other hand, we will never expect that a falling glass will result in no sound, simply because that is not a possible outcome considering the objects and actions involved.

As briefly mentioned above, we may think of an *action-sound palette* as being a range of possible combinations of sound-producing actions and resultant sounds. For action-sound couplings an action-sound palette is restricted to possible combinations of material and action properties. For example, the sound of a breaking glass will be dependent on the material, size and shape of the glass, as well as the distance it is falling, etc. These parameters may vary, but only within a fairly narrow palette as compared to the possible combinations of an artificial relationship.

The possible action-sound relationships of a door bell, on the other hand, are potentially infinite, and the sound may range from "ding-dong" or "beep" to a musical piece. Here a "ding-dong" sound may be perceived as more natural than a "beep" sound, since it inherits some of the action-sound qualities of a mechanical door bell upon which the design of the electronic door bell may have been modelled. However, if a pop song starts playing when pressing the button, it will probably be experienced as a rather unnatural action-sound relationship, what I call a weak action-sound relationship. As illustrated in Figure 3.7, action-sound relationships may range from being almost as strong as a coupling, to being very weak.

A large action-sound palette may be problematic in some contexts, but it may also

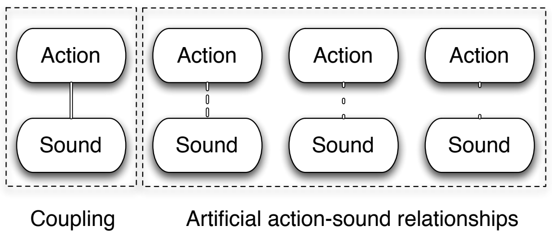


Figure 3.7: *As the vertical lines connecting the boxes indicate, action-sound relationships are coupled for natural objects (left), and are weaker in electronic devices where action-sound relationships are artificially created.*

be the source of creativity in others. For example, experimental composers and performers increasingly tend to extend the palette of their instruments in various ways. This may be done acoustically, for instance by *preparing* their instrument by adding various types of mechanical parts such as coins, paper clips, steel wool etc. It may also be done electronically, for example by using various types of sound effects. In some cases, such action-sound palette extensions may become part of the standard action-sound repertoire, such as the use of distortion and wah-wah pedals with electric guitars. Even though such effects originally may have seemed like a drastic change of the sonic result of a guitar, there are few people that would find this perceptually problematic nowadays.

In other cases, palette extensions may work poorly, and may feel unsatisfactory to both the performer and the perceiver. An example of this is how for many digital music instruments the relationship (the *mapping*) between the *controller* and the *sound engine* is weak. I particularly find that the action-sound relationships available in many commercial synthesisers built with a keyboard interface are problematic. In such instruments piano-like actions (impulsive) are used to control any type of sound model. The end result is that the relationship feels neither good, natural, nor creative for the performer and the perceiver. Clearly, there is a need for better action-sound design.

3.3.2 Design Strategies

When it comes to designing action-sound relationships for electronic devices and virtual realities, we may identify two different strategies:

- *Practical design*: focuses on creating action-sound relationships that enhance the usability of a device, or reality, and is often aimed at creating "natural" and strong relationships. Practical action-sound design is found in a lot of consumer electronics, and is often based on qualities of action-sound couplings.

- *Creative design*: is focused on creating novel action-sound relationships, often for entertainment. Here the aim is to create interesting, funny and surprising relationships.

The basic idea of *practical* action-sound design is that sound may greatly enhance the interaction between humans and technological devices. This has been an important topic in human-computer interaction for many years, as discussed by for example Gaver

(1986, 1989) in his experimental sound design for the computer desktop in the 1980s. The recent growth in the fields of *sonic interaction design*, *auditory display* and *sonification* shows that action-sound design is becoming more and more important as the world becomes increasingly technological. In these fields, creating intuitive and solid action-sound relationships is at the core, and this is often achieved by using properties of action-sound couplings. Examples of such practical action-sound design are readily available in computers, where many of the *system sounds* are designed to resemble action-sound couplings, such as the trash can sound heard when emptying the virtual trash can on the desktop. This and other computer system sounds may be seen as adding realness to the virtual reality.

Another example of practical action-sound design is found in movies. In most movies the sound design is intended to be as natural as possible. Interestingly this is most often achieved by recording the sounds separate from the visuals, using *dubbing*, *foley artistry*[2] and various other sound effect techniques. While the extensive use of such sound post-production may seem unnatural, it is in fact often done to make the sounds appear more natural than the real sounds recorded with the acting.

The film industry is also a good example of the *creative* use of action-sound relationships. One type of such creative relationships can be found in comedies and animations, which often make use of exaggerated action-sound relationships. An example is the *Road Runner* ("beep-beep") cartoons; every time the coyote runs off a cliff an extended whistling sound followed by a huge "boom" is heard. Here the sound effectively exaggerates the actions and adds to the comic aspects of the situation. The same type of sound effects has also been used in slapstick comedies throughout the years, again to exaggerate an action. In fact, often such sounds are so unnatural that we immediately identify the comic aspect of the movie. Yet another example of creative use of action-sound relationships may be seen in the extensive use of playback laughter in many sitcoms (situation comedies). This makes us believe that there is a large audience enjoying the show, while the laughter is often played on a keyboard by a sound engineer.

Action-sound relationships are also used in movies to create ambiguous effects. A common *transition* technique between shots or scenes is to fade in the sound a couple of seconds before the image appears. Here the sound will create an expectation of the image that will follow. Another transition technique is to morph sounds with similar timbral qualities into each other, a classic example being the fan/helicopter transition in the beginning of *Apocalypse Now* (Coppola, 1979). In this example the images and sounds of a hotel fan are morphed into the sounds and images of a helicopter propeller, creating an ambiguous situation both for the main character in the story and for the perceiver.

There are also examples of creative use of action-sound relationships in music. Many orchestration techniques are based on making separate sounds merge into new holistic textures which may be perceived as new and interesting. An example of this is the common technique of making chords with sounds from both impulsive and continuously sounding instruments. This creates the effect of an impulsive attack but with an unnaturally long "decay" prolonging the attack. Such techniques have also been explored

[2]Manually creating sound effects, e.g. footsteps, door handles and explosions.

extensively in electroacoustic music. Here new technology has opened for creating all sorts of new "instruments" that challenge our ecological knowledge. One example is the physically informed (and inspired) sonic model *blotar*, which is a combination of a flute, an electric guitar, and a mandolin (Trueman et al., 2001). Hearing such sounds may evoke several different and opposing mental images, and thereby open for interesting musical experiences. Yet another example of the creative use of action-sound relationships in music is the commonly used reversed cymbal sounds in electronic dance music. Such reversed sounds play with our ecological knowledge of prefix, excitation and suffix, and can lead to an upbeat and alert feeling which works well on the dance floor.

All in all, practical and creative action-sound design may be seen as two opposing design strategies, as shown in Figure 3.8. While the practical side is mainly focusing on ease of use, the creative side is focusing on constructing new and interesting relationships. From a perceiver's point of view, I find that much of the practical action-sound design found in electronic devices is often boring and uninspiring, while creative designs are often too confusing. A challenge here is to balance between the two axes, so that practical action-sound designs may also feel interesting, and creative designs not too bewildering.

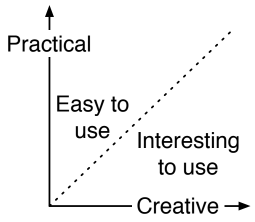


Figure 3.8: *I see practical and creative action-sound design as opposing design strategies. Practical designs are often easy to use but may be boring, while creative designs are often more interesting yet can be bewildering.*

3.4 Summary

Ideas from embodied music cognition presented in the previous chapter formed the basis for the discussion of *action-sound couplings* and *relationships* elaborated in this chapter. I argue that our perception of sound-producing actions and the resultant sounds are based on both the action and the sound, and that our mental imagery is based on the action, the sound and the coupling between them. Furthermore, I suggest that our ability to make predictions about the outcome of sound-producing actions arise from our ecological knowledge of action-sound couplings, and that these expectations are based on the *action-sound palette* afforded by the *object-action-object system*. This enables us to predict the sonic result of a sound-producing action we only see, or imagine the sound-producing action of a sound we only hear.

Similarly, I believe that ecological knowledge about action-sound couplings also guide our perception of artificially created action-sound relationships. Thus, if we want to design more *practical* action-sound relationships, they should be based on qualities found in action-sound couplings. In cases where the interaction itself is made to simulate an action-sound coupling, for example in a digital piano, the challenge is to develop engineering solutions that get as close as possible to the original object-action-object system.

In other types of practical action-sound design, for example in a mobile phone, the design should focus on the qualities of the interaction between the user and the device. For example, an impulsive action should lead to an impulsive sound, and a sustained action should lead to a sustained sound. Similarly, the timbral qualities and loudness of the sounds should match that of the sound-producing action. Following such an approach to action-sound design will probably lead to more intuitive interaction for the end user. An example may be found in the clicking sound heard when rotating the control wheel of an iPod, or the scroll button on Apple's Mighty Mouse. These action-sound relationships work so well that users may not realise that they are made by a tiny speaker in the devices. Developing such intuitive action-sound relationships requires further knowledge about our perception of action-sound couplings and relationships. For this reason I have carried out various observation studies together with other people in the Musical Gestures research group. These studies will be presented in Chapter 5.

Creative design of action-sound relationships, on the other hand, may be more flexible and may also intentionally violate action-sound couplings to surprise us. This sometimes works well, but it is a problem when the action-sound relationships are poorly designed. In such cases, the relationship neither works as a simulation of action-sound couplings, nor works in a creative way playing on our senses. I believe that designing better action-sound relationships that either simulate or exploit the potential of action-sound couplings is of crucial importance. Examples of such action-sound exploration will be discussed in the context of digital music instruments in Chapter 6. But before presenting these exploratory studies of movement and music, I shall discuss various types of music-related movement and how they can be classified.

CHAPTER 4 Music-Related Movement

First we move. Before we sing, we must be aware of our body. We take a deep breath, and the heart pumps faster, the blood flows to the extremities, the muscles flex and stretch, and we take a step forward.

Mark Ross Clark (2002, 3)

This chapter starts with an overview of various *gesture* definitions, and discusses why I have decided not to use this term in my research. There then follows an overview of various music-related movements and actions, and a taxonomy for describing such movements and actions.

4.1 Introduction

The previous chapter described one specific type of music-related movement: sound-producing actions. This chapter will give a general overview of different types of music-related movements, and suggest a terminology and a classification system that may be used to discuss such movements. Working on this project I have been struck by the lack of a common vocabulary, and how several of the main terms are used differently in various disciplines. In particular, I find the word *gesture* to be problematic, and particularly when it is used in the term *musical gesture*. The growing interest in "gesture research" in recent years is certainly fortunate, but I have found that the word gesture itself is becoming increasingly difficult to use. This chapter will therefore start out with a discussion of this term, before moving on to the presentation of various types of what I prefer to call *music-related movement*.

4.2 Gesture Definitions

Several suggestions for categorising the broad range of gesture research have been offered. Zhao (2001, 6) suggests dividing the field into two tracks: the *qualitative* and *conceptual* frameworks developed in linguistics, psychology, neurology, choreography and therapy; and the *quantitative* and *system oriented* frameworks in computer vision, human-computer interaction, human motor control and computer animation. Another categorisation was suggested by McNeill (2000, 9), looking at the *function* that gestures play in various fields: *human communication*, *cognitive psychology* and *modeling*. Combining the categorisations by Zhao and McNeill, I suggest a slightly different tripartite division of gesture research and associated gesture definitions:

- *Communication*: using gestures to denote aspects of human communication, focusing on how they work as vehicles of social interaction. This is how the term is often used in linguistics, psychology, social anthropology, etc.
- *Control*: investigating gestures as a system, focusing on computational models and the control possibilities of gestures in interactive systems. This is typical of the fields of human-computer interaction (HCI), computer music, etc.
- *Mental Imagery*: studying gestures as mental processes, which may be the result of physical movement, sound, or other types of perception. This is common in cognitive science, psychology, musicology, etc.

The following section will present examples of gesture definitions following each of these three categories. The selection of definitions is not exhaustive, but rather centred on the ones that can shed light on my main interest: music-related movement.

4.2.1 Gesture as Communication

In everyday life, *gesture* is often used to denote bodily actions associated with speech, particularly hand movements and facial expressions. This is also the gesture definition most commonly used in linguistics, psychology and behavioural studies, in which Adam Kendon has defined gesture as "visible action as utterance" (Kendon, 2004, 7). In his first papers on the topic, Kendon used the term *body motion* (Kendon, 1972) and later moved on to use *gesticulation* (Kendon, 1980), before finally settling on the word *gesture* (Kendon, 1982).

Gesture has also been the preferred term used by David McNeill in his research into the *co-existence* of gesture and speech (McNeill, 1992, 2005). Through a series of observation studies of people's storytelling, McNeill showed how hand movements and facial expressions are not just random movements accompanying speech, but are actually an integral part of the communication. Following this, he developed a taxonomy in McNeill (1992, 12-19) for different gestural functions based on the five types of nonverbal behaviour outlined by Ekman and Friesen (1969):

- *Iconics* represent a particular feature of an object, and can be described in terms of the shape and spatial extent of the gesture. Iconic gestures are often used to illustrate an action, for example imitating a knocking movement with a hand while saying "knocking on the door".

- *Metaphorics* are similar to iconics, but represent an abstract feature of an object. An example of a metaphoric gesture may be to say "something happened" while holding up the hands to refer to "something".

- *Beats* occur together with spoken words to highlight discontinuities and stress specific words. Beats are typically carried out as in/out or up/down movements, and may be seen as emphasising the most important words in a narrative.

- *Deictics* indicates a point in space, for example pointing in a specific direction while saying "over there".

- *Emblems* are stereotypical patterns with agreed meaning, such as the goodbye or OK sign.

McNeill's theory of gesture is built on the idea that gestures *coexist* with speech. This is not to say that they have to *co-occur*, but rather that gestures and speech are *co-expressive*. Here, McNeill (2005, 15) adopts Damasio's saying that "language is inseparable from imagery" and argues that mental imagery is embodied in the gestures that co-occur with speech. To explain the relationships between gesture and speech, McNeill (1992, 37) outlined what he calls the *Kendon continuum*, based on the typology of gestures suggested by Kendon (1982): *gesticulation*, *emblems*, *pantomime* and *sign language*. As shown in Figure 4.1, this continuum covers two extremes: *gesticulation* is used to denote the types of gestures that always co-occur with speech, and *sign language* the types of gestures that are linguistically self-contained.

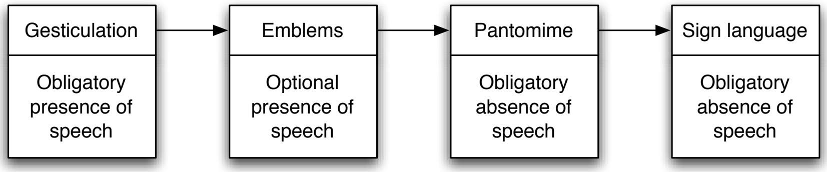

Gesticulation
Obligatory presence of speech
Emblems
Optional presence of speech
Pantomime
Obligatory absence of speech
Sign language
Obligatory absence of speech

Figure 4.1: *McNeill's (2005, 7)* Kendon continuum *of gestures, and how they relate to speech.*

Susan Goldin-Meadow has followed a similar line of thought in her work on how gestures help guide our thinking. The difference is that she uses the term gesture to denote only hand movement, and leaves out other types of body movement, including facial expression (Goldin-Meadow, 2003, 3). She, along with Kendon and McNeill, argue that gestures may not only *support* speech, but may also *contradict* speech. An example of this is how it is possible to spot when people are lying, since their facial expression and body movements contradict what they are saying. McNeill (2005, 4) therefore argues that studying *overt* gestures may reveal interesting aspects of our *covert* mental imagery. This is an interesting idea, and one which we have explored in the observation studies of music-movement correspondences that will be presented in Chapter 5.

The definitions presented above all focus on the communicative aspects of gestures. As such, the term *gesture* does not refer to body movement or expression per se, but rather to the intended or perceived *meaning* of the movement or expression. This is similar to how Feyereisen and de Lannoy (1991, 3) use the term gesture, but they provide a slightly wider definition:

> To some extent, any movement or change in position of a body segment may be considered a gesture. Accordingly, the very notion of gesture refers to a great variety of phenomena. In an extended sense, the term gesture encompasses technical gestures that are used in various professions and that often involve tool use. From such a perspective, gestures are mainly actions before becoming means of communication.

Here they open for using the term gesture not only in human-human communication, but also for describing the communication between humans and machines. This is probably why this definition also has been referred to in the human-computer interaction (HCI) literature, focusing on gesture for control.

4.2.2 Gesture for Control

An important topic in HCI is that of communication between humans and computers. Here, as for human-human communication, the point is *what* is communicated and not necessarily *how*. A difference between these fields, though, is that computers traditionally have had fairly limited sensing capabilities. This is probably the reason why several of the gesture definitions used in HCI are fairly restricted, for example a definition by Kurtenbach and Hulteen (1990, 310):

> A gesture is a motion of the body that contains information. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on it's way to hitting a key is neither observed nor significant. All that matters is which key was pressed.

This is a classic example of a computer-centric approach to interaction design, in which the constraints of the computer define the interaction. Here pressing the key is highlighted as the meaning-bearing component, while the rest of the movement of the person is considered irrelevant. Much has happened in the HCI community since Kurtenbach and Hulteen's definition was written, and today the focus is much more on creating computer systems that can sense human communication. The challenge is now to develop sensor and computer vision solutions, and corresponding computational algorithms, for separating gestures (here used in the communication sense) from a continuous stream of movement. While humans have few problems with separating a hand gesture from other types of movement (e.g. waving away a fly), this is much more problematic for computers. This is not only because of our capacity of *visual scene analysis*, as discussed in Chapter 2, but is also due to the fact that we may directly understand the intended meaning of the gesture based on our life-long experience of multimodal communication.

Rather than creating computer systems that can understand the meaning of gestures, some researchers are working on solutions for extracting the *expressiveness* of body movement. Here the term *expressive gesture* is introduced to denote aspects of body movement that convey information about affect and emotion:

> It seems likely that expressiveness in gestures is conveyed by a set of temporal/spatial characteristics that operate more or less independent from the denotative meanings (if any) of those gestures. In that sense, gestures can be conceived as the vehicles which carry these expressive characteristics and it is likely that expressiveness as such subsumes certain universal patterns and general rules. (Camurri et al., 2001, 1)

Yet another type of gesture definitions is offered by researchers working on the control aspects of computer music:

> If we call first of all, "gesture" all physical behaviour, besides vocal transmission, by which a human being informs or transforms his immediate environment, we may then say that there can be no music without gesture, for music is not exclusively vocal. (Cadoz, 1988, 65)

> [G]esture is used in a broad sense to mean any human action used to generate sounds. The term refers to actions such as grasping, manipulation, and noncontact movements, as well as to general voluntary body movements. (Miranda and Wanderley, 2006, 5)

Both these definitions are broad in scope, and like the other HCI definitions they focus on gestures for controlling various systems. An important difference between these control definitions and the communication definitions presented in the previous section, is that of what types of body movements are being studied. The control definitions are mainly focusing on *manipulative* gestures (Quek et al., 2002, 172). Manipulative is here used to denote gestures that are based on physical contact, or what may also be called *ergotic*, *haptic*, or *instrumental*. As such, they are clearly different in nature from the *empty-handed*[1] gestures that are studied in the fields of human-human communication. It is therefore easy to imagine the potential problems that may arise when such definitions are mixed up, something I often experience in musical settings where people come from both types of backgrounds.

[1]Empty-handed gestures have also been called *semaphoric*, *free*, *semiotic*, or *naked* gestures (Miranda and Wanderley, 2006, 6).

4.2.3 Gesture as Mental Imagery

A third group of gesture definitions focuses on gesture as mental imagery. While the two previous groups of definitions have both been using gesture in reference to some kind of physical body movement, there are also some definitions that use gesture in a metaphorical sense, for example:

> [B]oth [physical and auditory gestures] present the ability to communicate musical intentions at a higher level than an audio wave form. The similarity of their level of abstraction motivated the author to label them both as Musical Gestures. (Métois, 1997, 16)

Here the term *musical gesture* is used to denote the combined sensations of physical movement and sound. This is along the lines of how several musicologists have used term musical gesture in recent decades. From a popular music perspective, Middleton (1993, 177), referring to Coker's (1972) discussion of affections and emotions which could be associated with gestures, writes:

> [H]ow we feel and how we understand musical sounds is organised through processual shapes which seem to be analogous to physical gestures.

Middleton argues that the idea of gestures in music should be founded on the concept of rhythm. This seems similar to Todd's (1995, 1941) idea of relationships between musical sound and the body. Todd claims, without actually using the word gesture, that musical movement is similar to, and imitates, motion in physical space.

A similar way of thinking about gestures, as a mental entity that can be evoked from musical sound, is suggested by Hatten (2004, 95), who argues that a musical gesture is "significant energetic shaping through time". His theory of *musical gesture* is based on bodily action, or what he calls *gestural competency*. Such competency, he argues, arises from *physical* (i.e. biological and cognitive) and *social* (i.e. cultural and multi-stylistic) experience:

> Musical gesture is biologically and culturally grounded in communicative human movement. Gesture draws upon the close interaction (and intermodality) of a range of human perceptual and motor systems to synthesize the energetic shaping of motion through time into significant events with unique expressive force. (Hatten, 2003)

François Delalande offers a gesture definition that may be seen as a combination of the definitions that focus on gesture as mental imagery evoked by sound, and the definitions based on meaning conveyed by visible body movement. In a study of pianist Glenn Gould, Delalande (1988) notes that the term *musical gesture* lies in the intersection between observable actions and mental representations. He further argues that musical gestures may be studied at various levels, ranging from the purely functional to the purely symbolic: *effective*, *accompanist*[2] and *figurative* gestures (Cadoz and Wanderley, 2000, 77-78). Here *effective gesture* is used to denote what I call sound-producing actions, while *accompanist gestures* are used for the movements that support the effective gestures in various ways. Delalande suggests the term *figurative gesture* to refer to a mental representation that is not directly related to any physical movement, but which may be conveyed through sound only.

A somehow analogous definition to that of Delalande and Hatten is suggested by Gritten and King (2006, xx):

> [A] gesture is a movement or change in state that becomes marked as significant by an agent. This is to say that for movement or sound to be(come) gesture, it must be taken intentionally by an interpreter, who may or may not be involved in the actual sound production of a performance, in such a manner as to donate it with the trappings of human significance.

This definition implies that there is a flow of communication between the performer and the perceiver, and movement "becomes" a gesture if it is understood as such by the perceiver, as illustrated in Figure 4.2.

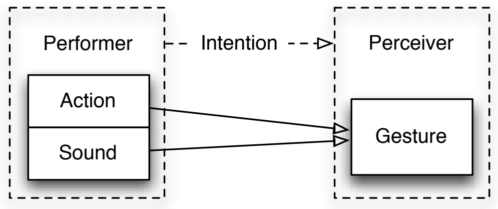


Figure 4.2: *My illustration of how Gritten and King (2006, xx) suggests that a performer's movements may become gesture if perceived (and understood) as such by the perceiver (the interpreter).*

An interesting question then arises when it comes to consciousness: does an action have to be carried out consciously to be seen as a gesture? In human communication, Kendon has argued that gestures have to be carried out consciously since they are intentional (Kendon, 2004, 15). Hatten, on the other hand, argues that musical gestures may be performed unconsciously but still be valid as gestures if they are observed as significant by the perceiver (Gritten and King, 2006, 162). I would assume that there are also ambiguous cases where one person may perceive an action as intentional and another person may see it as unintentional. In that case it will have to be the subjective experience that will decide whether the action should be seen as a gesture or not.

4.2.4 Terminology

The gesture definitions presented above range from using gesture more or less as equivalent to body movement, to using gesture in a purely metaphorical sense. There are

[2]*Accompanist gesture* is the translation by Cadoz and Wanderley (2000) of the term *geste accompagnateur* used by Delalande (1988). It might be better to use the word *accompanying* in English, so as not to confuse such movements with those of an accompanist.

certainly many other types of gesture definitions that have not been discussed, such as the concepts of *articulatory* and *phonological gesture* used in the motor theory of speech perception (Liberman and Mattingly, 1985). There are also several research fields focused on topics similar to the ones discussed, but which do not use the term gesture at all, such as *kinesiology* and *biomechanics*. The same can be found in some of the music performance literature, where terms like *motion/movement* (Shaffer, 1980; Gabrielsson, 1985; Clarke, 1993; Davidson, 1993) and *expressive movement* (Pierce and Pierce, 1989; Davidson, 1994) have been used. Yet another term is that of *corporeal articulation*, which Leman (2007) has used in the study of various types of music-related movement.

As I have witnessed at several conferences during the last years, the multitude of gesture definitions often results in much confusion and discussion. Particularly the rather imprecise use of gesture in parts of the HCI literature seems to cause confusion when contrasted with stricter definitions. Some have argued that it would help if gesture always was used in conjunction with another term, e.g. *instrumental gesture* (Cadoz, 1988) or *ancillary gesture* (Wanderley, 2001). My solution has been to avoid using the term gesture at all, and rather settle on the following terms:

- *Movement* is used to describe the act of changing the physical position of a body part or an object. This may be any type of displacement, whether it be carried out consciously or unconsciously.

- *Action* is used to denote a movement *unit*, or a *chunk*. This is a limited and goal-directed movement which is perceived as a holistic entity, or a manipulative action based on touch or force. For example, hitting a piano key with a finger may be seen as a displacement action, while holding the finger on the key may be seen as a manipulative action. As such, I use action to denote a cognitive entity, something which cannot be directly measured but rather inferred from movement.

- *Fidgeting* is used to denote movements that are not goal-directed and which may be unintentional and unconscious. Fidgeting can be thought of as "movement noise".

- *Interaction* is used to denote the reciprocal actions of objects (including body parts) that influence each other.

Figure 4.3 illustrates how I see the relationships between the three first concepts. Movement is shown as a continuous stream of displacement over time, while actions are separate units. The parts of a movement that are not categorised as an action are considered fidgeting. Note that actions can be nested, e.g. several actions that follow each other may be perceived as one coherent action. For example, playing a scale run on a piano may be seen as a series of separate actions if we focus on the finger movements, but can also be perceived as one coherent action if we look at the movement of the hand or the upper body. It seems as if we are able to perceive many such actions simultaneously, and that the actions mutually influence our perception, something which will be discussed in context of our observation studies presented in Chapter 5.

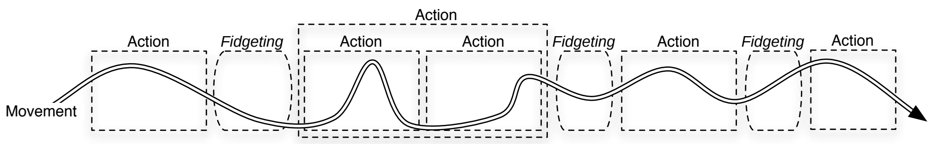


Figure 4.3: *A sketch of how I see the differences between the concepts of* movement, action *and* fidgeting. *Movement can be seen as the continuous displacement of a limb or object. Action may be seen as chunks of movement, while fidgeting refers to the movements between the actions. Note that an action may be a composite of other actions.*

The rest of this chapter will focus on various types of music-related movements, with an emphasis on developing a taxonomy for understanding the similarities and differences between such movements.

4.3 Music-Related Movement

Although there may be ambiguous cases, generally we can divide music-related movement into two main categories: the movement of *performers*, and the movement of *perceivers*. The movement of perceivers has received relatively little attention in the literature, and will be discussed more in Chapter 5. When it comes to the performers, I prefer to split this category into three subcategories, each having their own set of music-related movements:

- *Musicians*: the movements of musicians may be categorised as *sound-producing*, *ancillary*, *sound-accompanying* and *communicative*. These categories will be discussed more in the rest of this chapter.

- *Conductors*: the movement of conductors may be thought of as *sound-coordinating*. They are not sound-producing in themselves, but are still important for the sounding result (Nakra, 2000; Kolesnik and Wanderley, 2004; Gambetta, 2005), and as a specific type of theatrical movements (Small, 1998). Conductors' movements fall outside the scope of this project, and will therefore not be discussed any further in this dissertation.

- *Dancers*: The movements of dancers may be thought of as *sound-accompanying*, since they often follow or contrast with the musical sound. Such movements will be discussed more in Chapter 5.3.

There are many approaches to the study of music-related movements movements. In his dissertation on *instrumental gestures* (what I call sound-producing actions), Ramstein (1991, 29) suggested that music-related movements can be studied on three analytical levels:

- *Phenomenological*: this approach focuses on the *descriptive* aspects of the movement, such as describing the movement in terms of its *cinematic* (e.g. the speed), *spatial* (the size of space), and *temporal* dimensions (e.g. frequency range).

- *Intrinsic*: this approach deals with the conditions for movement production, for example that the hands are more suitable for fine motor control than the feet.

- *Functional*: this approach looks at the purpose of a movement or action in a certain context, e.g. whether it is sound-producing, sound-modifying, etc.

The rest of this chapter will focus on the *functional* aspects of the music-related movements of musicians. But first I shall start by defining some general properties of the performance environment and the performer's relationship to the environment.

4.3.1 Performance Scene, Position and Space

To have a clear idea of the performance environment in which music-related movements are carried out, I suggest three concepts: *scene*, *position* and *space*, as summarised in Figure 4.4. First, the *performance scene* may be thought of as a physical space that both the performer and the perceiver recognise as one in which a performance is carried out. This is evident as both performers and perceivers tend to change their attention when the performer enters the performance scene. In a typical concert situation the performance scene is clearly defined as a part of the *stage*, where everyone's attention is naturally focused due to the seating and general expectation of what will come. But a performance scene may also refer to a social construct, and may thus be created anywhere. A typical example is how street musicians claim a part of the pavement as their performance scene, which people walking past will usually respect.

When it comes to the *performance positions*, we can define the *home-position* of a performer to be the resting position in which the performer sits or stands before starting to act (Sacks and Schegloff, 2002). In a musical context, and particularly in Western classical music, this can be understood as when a musician is standing or sitting at ease with the instrument before starting to perform. When in home position, the perceiver will usually know that the performance has not yet begun and will wait until the performer moves into *start position* before expecting any sound to be produced. Finally, the *performance position* is the one from which the performance action originates.

We may also refer to a set of *performance spaces*, or a *personal space*. First, we may use Laban's term *kinesphere* to denote an imaginary box surrounding a person, which defines the maximum movement possibilities from a certain point in space (Laban, 1963). Laban argues that the kinesphere is a mental construct that we are always aware of in our interaction with the environment and with others. Within the kinesphere we may talk about different *performance spaces* or *action spaces*, i.e. imaginary "bounding" boxes for various types of music-related movements. For example, when playing the piano we have a well-defined sound-producing action space in the visual part of the keyboard, as indicated in Figure 4.5. This action space can usually be observed by both the performer

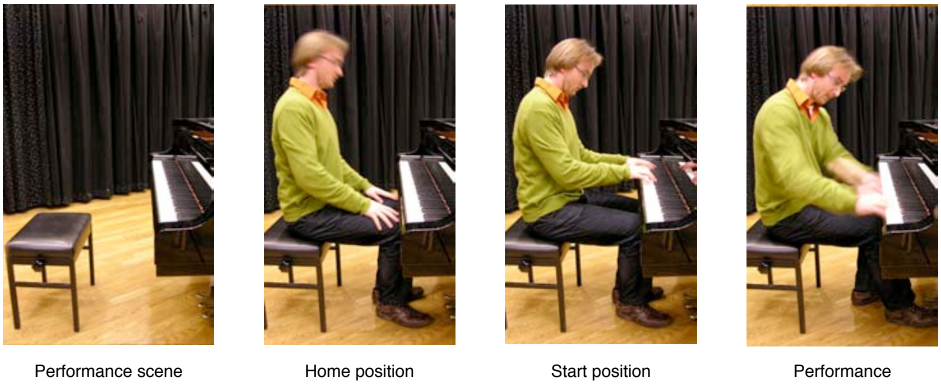


Figure 4.4: *The* performance scene *is the imagined area in which performance can happen. The* home position *is the position where the musician is sitting (or standing) at ease before starting to perform. The* start position *is where the performance starts from, and the* performance position *is the position(s) of the musician during performance.*

and the perceiver, making it possible to identify where the sound-producing actions are carried out.

Figure 4.5 also indicates the performance spaces of other other types of music-related movements (as will be presented in the following sections). The idea of identifying these spaces is to illustrate that we have a clear understanding of where different types of movements and actions should be carried out in relation to an object (e.g. an instrument). This knowledge of performance spaces for various types of music-related movements also helps us set up expectations when perceiving a performance. This is why we may get surprised if a musician happen to perform outside of such conventional performance spaces, for example by playing with the fingers on the strings of the piano. Much musical experimentation happen due to such exploration of the boundaries of established performance spaces.

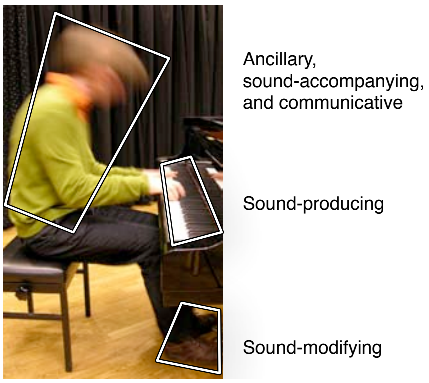


Figure 4.5: *The* action space *can be seen as an imaginary box surrounding the space in which movements can be carried out. Here the action spaces for various music-related movements are indicated, including sound-producing and sound-modifying actions, and ancillary, sound-accompanying and communicative movements.*

4.3.2 Action Units and Phrases

Let us now look more closely at the construction of an action. In Chapter 3, a sound-producing action was defined in terms of an excitation with a prefix and suffix. More generally, we can talk about an *action unit*, analogous to Kendon's *gesture unit*, defined as a *goal-directed movement excursion* which starts and ends in the home position (Kendon, 2004, 111). As shown in Figure 4.6, Kendon suggests that such a unit may contain multiple *phrases*, each of which is built from a *preparatory* part, a *nucleus* and a *recovery*. The nucleus can be further split into a *stroke* and a *post-stroke* part. The stroke part is the *goal-directed* part of the movement excursion. In a sound-producing action this is typically the *excitation*, while for a communicative action the stroke is what is usually identified as the "gesture". The stroke may end in a recovery phase, or may be extended in a *post-stroke hold* prolonging the effect of the stroke. In music performance, this can often be seen in the performance of the final chord at the end of a piece, at which point the performer will hold onto the stroke position for much longer than is necessary for the sound-producing action itself. As such, the post-stroke hold can be seen as a type of movement embellishment prolonging the perception of the musical sound.

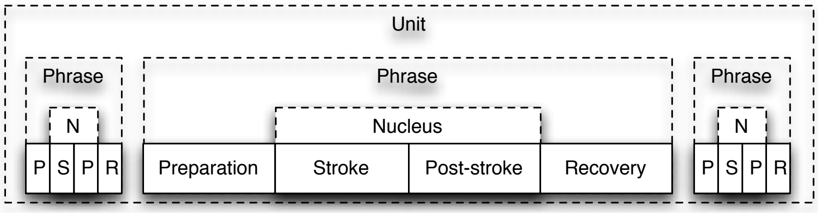


Figure 4.6: *The construction of an* action unit*, based on Kendon's (2004, 111)* gesture unit*. The two phrases on the left and right of the middle phrase are compressed to save space in the illustration (N = nucleus, P = preparation, etc.)*

4.3.3 Functional Aspects of Musicians' Movements

To understand more about the functions of various music-related movements, I suggest to divide music-related movement into four functional categories based on a combination of Cadoz' (1988) taxonomy for *instrumental gestures*, Delalande's (1988) typology, and Wanderley's (2004) divisions:

- *Sound-producing actions*[3] are the ones that are effective in producing sound. They can be further subdivided into actions of *excitation* and *modification*.

[3]Sound-producing actions are called *instrumental gestures* by Cadoz (1988), and *effective gestures* by Delalande (1988).

- *Ancillary movements*[4] support the sound-producing actions in various ways. As will be discussed and exemplified in a later section, such movements can be subdivided into *support*, *phrasing* and *entrained* movements. I refer to these as ancillary *movements* (and not actions) since they may not be of a goal-directed nature.

- *Sound-accompanying movements/actions* are not involved in the sound production itself, but follow the music. They can be *sound-tracing*, i.e. following the contour of sonic elements, or they can *mimic* the sound-producing action. Such sound-accompanying movements will be discussed in Chapter 5.

- *Communicative movements*[5] are intended mainly for communication. As will be discussed later in this chapter such movements can be subdivided into *endogenous*, *performer–performer* or *performer–perceiver* types of communication.

Figure 4.7 shows an illustration of different types of music-related movements and actions in piano performance. Note that the different categories are not meant to be mutually exclusive, as several actions have multiple functions. For example, hitting a final chord followed by a theatrical lift can be seen as having both sound-producing, ancillary and communicative functions. This functional multiplicity is illustrated in the *dimension spaces* in Figure 4.8. Dimension spaces are commonly used to analyse interactive systems (Graham et al., 2000), and have also been used to analyse the functionality of digital musical instruments (Birnbaum et al., 2005). Here they are utilised to visualise how the movements of a musician and a dancer cover different music-related movement functions.

4.3.4 Sound-Producing Actions

Based on the typology presented by Cadoz (1988), we can divide the sound-producing actions into two categories: *excitation* and *modification*.[6] As already discussed in Chapter 3.2.1, excitation actions may be subdivided into *impulsive*, *sustained* and *iterative* actions, each having distinct energy profiles. Excitation actions are either *direct* or *indirect*, depending on whether or not there is an object between the sound-producing element of the instrument and the object carrying out the excitation. For example, the actions of playing the harp or congas are *direct* since fingers and hands are directly in control of the resonating objects (strings and drum membrane). But there are also many *indirect* instrumental actions which involve one or more objects in the interaction, for example the bow in string instruments, the key mechanism on the piano, or sticks for drums.

[4]Ancillary movements are called *accompanist gestures* by Delalande (1988), *non-obvious performer gestures* by Wanderley (1999), and *ancillary gestures* by Wanderley and Depalle (2004).

[5]Communicative actions are called *semiotic gestures* by Cadoz and Wanderley (2000). Several of these can also be considered *gestures* in the way Kendon (2004) and McNeill (1992) use the term.

[6]Cadoz uses the term *instrumental gesture* to denote what I call sound-producing actions. He further uses *sound-producing gestures* and *sound-modifying gestures* for the subcategories of *instrumental gesture*. I have chosen to use other terms to avoid conflict with the rest of my terminology.

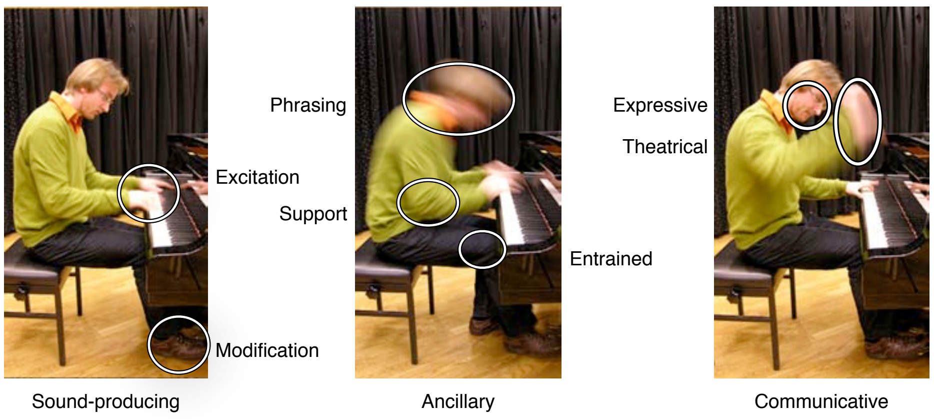


Figure 4.7: *Examples of where different types of music-related movements (sound-producing, ancillary and communicative) may be found in piano performance.*

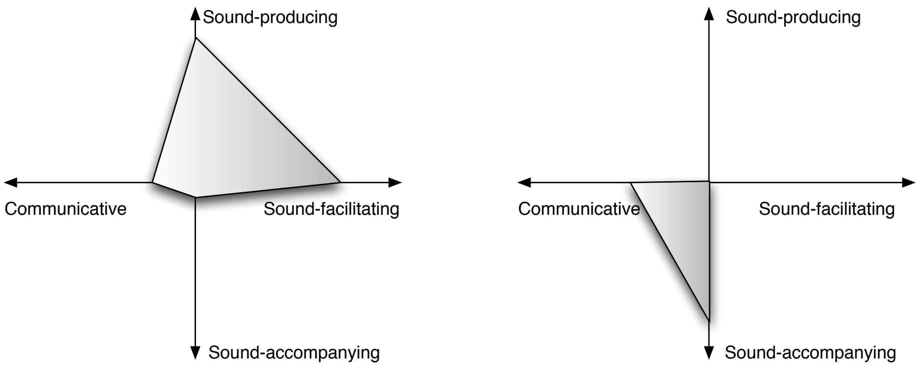


Figure 4.8: *Dimension spaces illustrating how the music-related movements of a musician (left) and a dancer (right) may be seen as having different functions. Here the musician's movements have a high level of sound-producing and ancillary function, while the dancer's movements have a high level of sound-accompanying and some communicative function.*

Modification actions is the other subcategory of sound-producing actions. Such actions do not actually produce sounds themselves, but they modify the quality of the sound. Cadoz (1988) suggested to subdivide such modification actions as:

- *Parametric*: actions which continuously change a parameter, e.g. bow pressure in violin playing.
- *Structural*: actions which modify or change the structure of the object, e.g. opening or closing a key on a wind instrument.

Most musical instruments are played with both excitation and modification actions. In some instruments the two types are easily separable, such as in string instruments where the two hands play entirely different roles: the left hand is mainly modifying the sound (choosing the pitch) while the right hand is carrying out the excitation. Another example is that of wind instruments, where the excitation is often done in the mouth, and modification actions are carried out with the fingers. Wind instruments are also an example of how excitation and modification actions are coupled, since the mouth can also effectively modify the quality of the sound. As such, the aim of this typology is not to create an absolute classification system, but rather to point out some of the different functions that actions may play in sound production.

4.3.5 Ancillary Movement

Ancillary movement is used to cover three different types of music-related movements that are not directly involved in sound production, but still have an important impact on the resultant sound: *support*, *phrasing* and *entrained* movements. As briefly mentioned above, I refer to these as *movements* rather than *actions*, since they may not be goal-directed and chunk-based.

It does not make sense to talk about sound-producing actions without also referring to the other body movements that co-occur with the sound-producing actions. For example, hitting a piano key involves not only the active finger, but also the hand, arm, and upper body. Such movements are *support* movements of the sound-producing actions. In fact, it is the preparatory movements of this complex *multi-joint system* that determine the trajectory and velocity of the finger before and after it hits the key. Thus, such support movements play an important role in supporting the sound-producing actions, and they may even have audible components, as shown by Wanderley (1999) in a study of clarinet performance. Here the performer's ancillary movements were seen in the movement of the clarinet bell, and this clarinet movement was shown to have an audible component due to the changing sound diffusion pattern of the instrument.

A different type of ancillary movement is what I prefer to call *phrasing* movements, since they are closely connected to musical phrasing. Wanderley (2002) has shown that the ancillary movements of clarinettists are an integral part of the instrumentalists' performance and are stable and reproducible even after long periods of time. Many of these repeatable movements and movement patterns seem to be closely connected to the phrases in the music being performed, and are often related to movement of the clarinet bell (Campbell et al., 2005; Quek et al., 2006).

Examples of support and phrasing movements are shown in Video 4.1, an except from Beethoven's Sonata Op.31 No.2 *Tempest* (III Allegretto) performed by François-René Duchable (2003). Figure 4.9 shows an image from the video where the rotating elbow movements support the hand rotation needed to carry out the passage. There is also an image showing a phrasing action when the performer is bending forwards to accentuate a salient point in the melody.

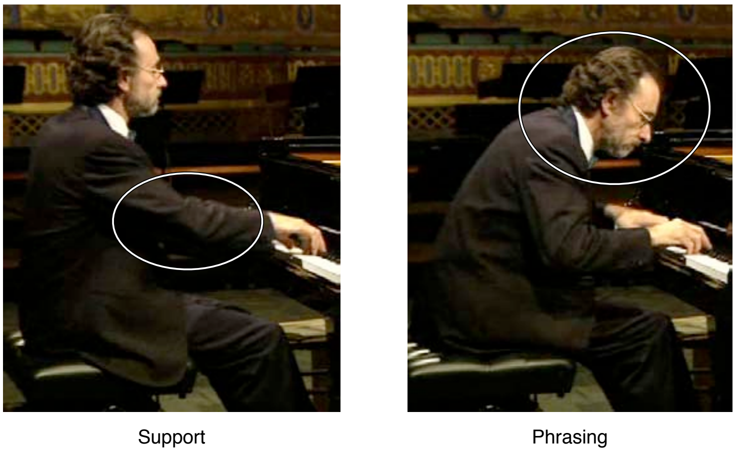


Figure 4.9: *Two types of* ancillary *movements seen in the performance by pianist François-René Duchable in Video 4.1: circular elbow movements which can be seen as* support *movements for the the circular hand movements (left), and moving the head forwards as a* phrasing *movement which helps accentuate the musical passage (right).*

A third type of sound-facilitating movements is what I call *entrained* movements. The concept of *entrainment* is based on the synchronisation of two or more independent rhythmic processes (Clayton et al., 2005), and can be said to have an ecological basis in biological, physiological and cultural rhythms (Aschoff, 1979; Hall, 1984; Strogatz and Stewart, 1993; Glass, 2001). Entrainment may happen as interpersonal synchrony, but here I am primarily interested in the synchronisation between the body, or body parts, and the music. Entrained movements are not directly linked to sound-production or the musical structure, but rather synchronise with a continuous underlying feature of the music, for example the characteristic rhythmic "feel" or the *groove* of the music.

Examples of entrained movements are tapping a foot, nodding the head or moving the whole upper body in synchrony with the music. Although such movements vary considerably between performers and performance styles, they may be thought of as important for the *timing* (or lack thereof) in a performance. As such, entrained movement can be a generator of rhythm and timing, in the same way as the rhythm and timing in music can be a generator of movements (Clarke, 1999). This was also observed in the clarinet experiments by Wanderley et al. (2005), where the performers continued to move, albeit less, when asked to play "immobilised".

An example of entrained movements is shown in Figure 4.10, a screenshot from Video 4.2 of the song *We Work The Black Seam* from the music movie *Bring on the Night* by Sting (1985). Here all the musicians show examples of entraining with the music: Branford Marsalis (tenor saxophone) is moving his right leg slightly, Darryl Jones (electric bass) is moving his right foot and the upper body, and Janice Pendarvis and Dolette McDonald (vocals) are moving in a choreographed manner, but are also swinging their arms more freely with the music. Sting has a very visible rhythmic movement starting in the left leg and extending through the body. I find this movement to be a bit too extreme to be only entrainment, and it can perhaps also be seen as a type of conductor movement, giving the pulse to the other musicians. As such, all the musicians show some type of entrained movements even though they are embodied differently.

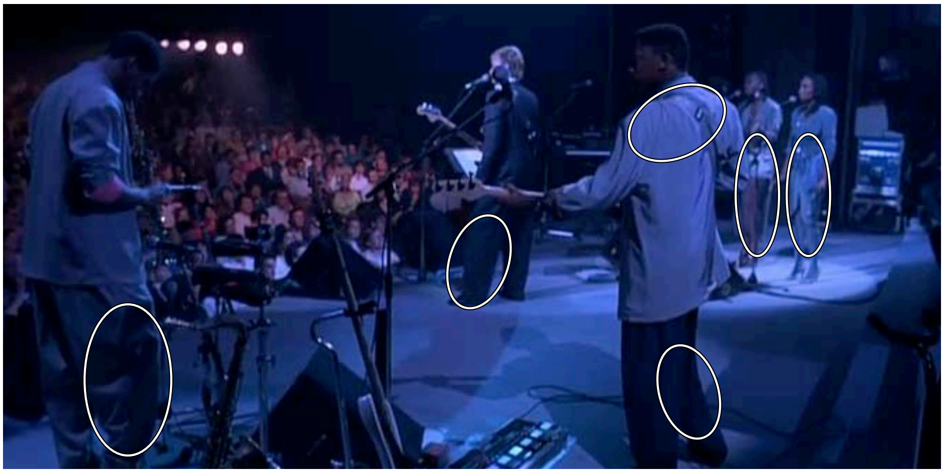


Figure 4.10: *An example of entrained (groove) movements in the song* We Work The Black Seam *from Sting's (1985) music movie* Bring on the Night *(Video 4.2). The circles identify where I see examples of entrained movement.*

Obviously, the three categories of support, ancillary and entrained movements are not always separable. Consider the excerpt of Rachelle Ferrel's solo on *Autumn Leaves* in Video 4.3 from The Manhattan Project (1991). The images in Figure 4.11 show some of her rich movement repertoire; including hand and finger movements to accentuate the rhythmic pattern she is singing, various groove-based entrained movements in knees and the right arm, and phrasing movements to create musical accentuations. This type of bending movement also has an important sound-producing effect, since the flow of breath dramatically changes when she leans towards the microphone.

As mentioned previously, the main idea of developing this taxonomy of music-related movement is not to devise a strict categorisation. Rather, I am trying to create a vocabulary that can point out some of the different types of functions these movements have in

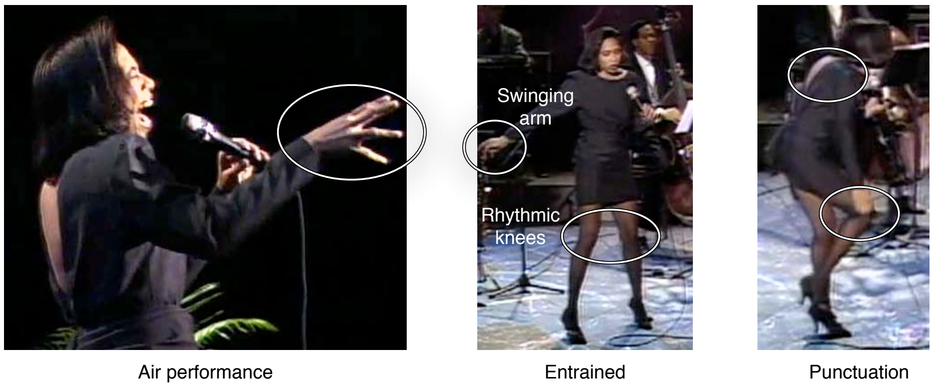


Figure 4.11: *Images from Video 4.3 of various kinds of sound-facilitating and communicative movements in a solo by Rachelle Ferrel (The Manhattan Project, 1991), including air performance with the fingers, entrained movements in the right arm and the knees, and a phrasing movement when she bends forwards to create a musical punctuation.*

music performance. When it comes to the ancillary movements, it is important to stress that these originate from the performer's needs, and should be separated from movements that are intended mainly for communication with the other performers and the audience. As such, ancillary movements are usually not carried out with a specific intention other than being the basis for, or result of, the sound-producing actions.

4.3.6 Sound-Accompanying Movements

A third type of music-related movements is what can be called *sound-accompanying* movements. These are neither part of, nor ancillary to, the sound-production, but rather intended to *follow* qualities in the sound or music. Such movements consist of *tracing* the melody of a song in the air, *mimicking* the sound-producing actions in "air instrument performance", or showing some other type of relationship to the musical sound. Sound-accompanying movements are carried out by performers (musicians, dancers, actors), but they are also observable in the movement of perceivers listening to music. Examples of such sound-accompanying movements will be shown in the context of the various types of observation studies presented in Chapter 5.

4.3.7 Communicative Movements

All performance movements can be considered a type of communication, but I find it useful to have a separate category for movements that are primarily intended to be *communicative*. These may be *endogenous*, *performer–performer* and *performer–perceiver* types of communication, and range from communication in a linguistic sense (*emblems*), to a more abstract form of communication.

An example of *endogenous* communicative movements can be seen in a performance by Glenn Gould in Video 4.4 and Figure 4.12 (Gould, 1974). It is impossible to know whether these movements were carried out for the sake of the camera or not, but, as Delalande (1988) suggests, they seem to be a type of communicative movement which involves Gould conducting himself. They may also be understood as a type of sound-accompanying or even ancillary phrasing movements.

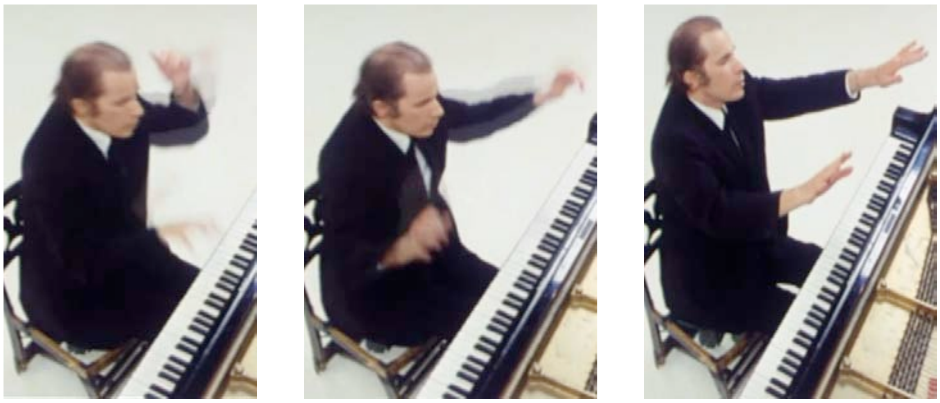


Figure 4.12: *Examples of Glenn Gould's hand movements as seen in Video 4.4 (Gould, 1974). These seem to be endogenous communicative movements and look like conducting movements.*

Several different types of *performer–performer* communicative movements can be seen in the images in Figure 4.13 taken from Video 4.2. Here Sting conducts the other musicians using eye contact and nodding. Notice also the passage during which Sting walks over to Branford Marsalis (tenor saxophone). The rhythmic swing Sting displays in his walk sets up expectations for the following contact improvisation, during which the two musicians keep focused on each other's gaze and movements.

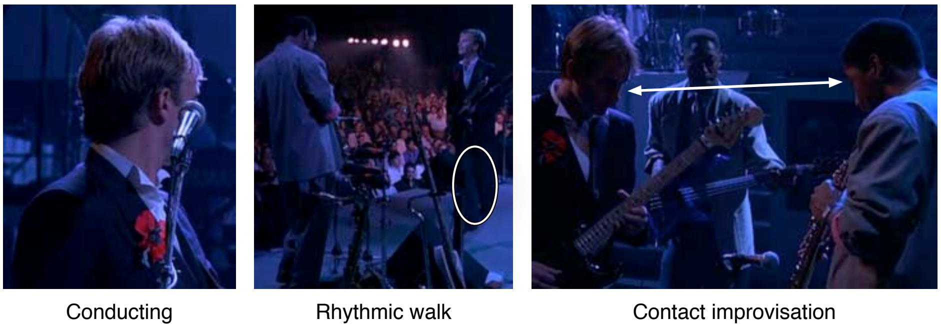


Figure 4.13: *Examples of performer–performer communicative movements seen in Video 4.2 (Sting, 1985). Sting conducts the other performers (left), indicates the groove while walking over to Branford Marsalis (middle), and keeps a close eye connection during Marsalis' solo (right).*

Examples of *performer–perceiver* communicative movements may be seen in the performance by Nigel Kennedy in Video 4.5 from the *Spirits of Music* DVD (McFerrin and Kennedy, 2005). The first image in Figure 4.14 shows a snapshot of how Kennedy turns and looks at one of the other musicians creating a sense of unity between the musicians on stage, or what could be called a *regulator* in the terminology used by Ekman and Friesen (1969). An example of what Ekman and Friesen called an *affect display* can be seen in the second picture in which Kennedy closes his eyes and mouth to enhance the mellow musical passage. Another affect display can be seen in the last image in Figure 4.14, in which Kennedy surprisingly outbursts an "ah" sound in the middle of the performance. All of these examples show various types of communicative movements that enhance the music performance, yet have little direct sound-producing function.

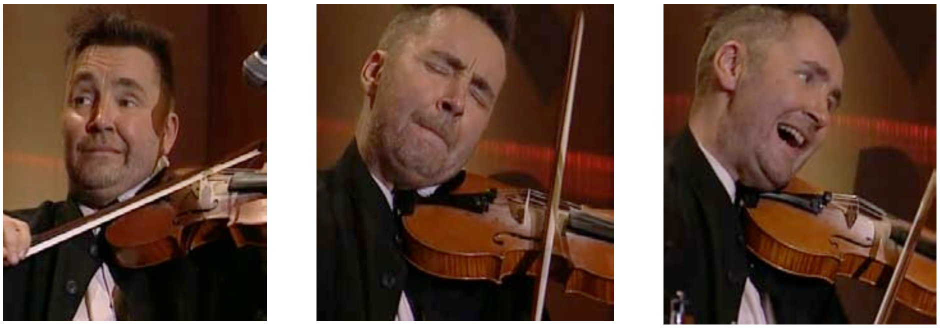


Figure 4.14: *Three types of communicative movement seen in Video 4.5 (McFerrin and Kennedy, 2005): contact movements through a turn and gaze (left), showing affect and emotion while playing (middle), and a surprised "ah" sound and facial expression (right).*

Several of the movements in Nigel Kennedy's performance in Video 4.5 can also be seen as having strong, theatrical components. Figure 4.15 is a picture summary of how Kennedy starts walking to the right side of the stage, turning slightly to be sure that everyone is following his movements. Then he punches into the air, and returns to continue playing with a smile on his face. These movements are clearly intended for communication, but may also serve to build musical expectation. The apparent expectation is further prolonged by the rubato playing style before they finally return to the melody of the piece.

4.3.8 Movement Repertoire

One of the things I have noticed while watching a number of music performance DVDs, is how many musicians seem to develop a rather consistent *movement repertoire*. This is particularly evident in many artists/musicians in different kinds of popular music. Figure 4.16 shows three images from Video 4.6 (Hendrix, 1973), where Jimi Hendrix performs a "swinging arm" movement. This swinging arm movement consistently co-occurs

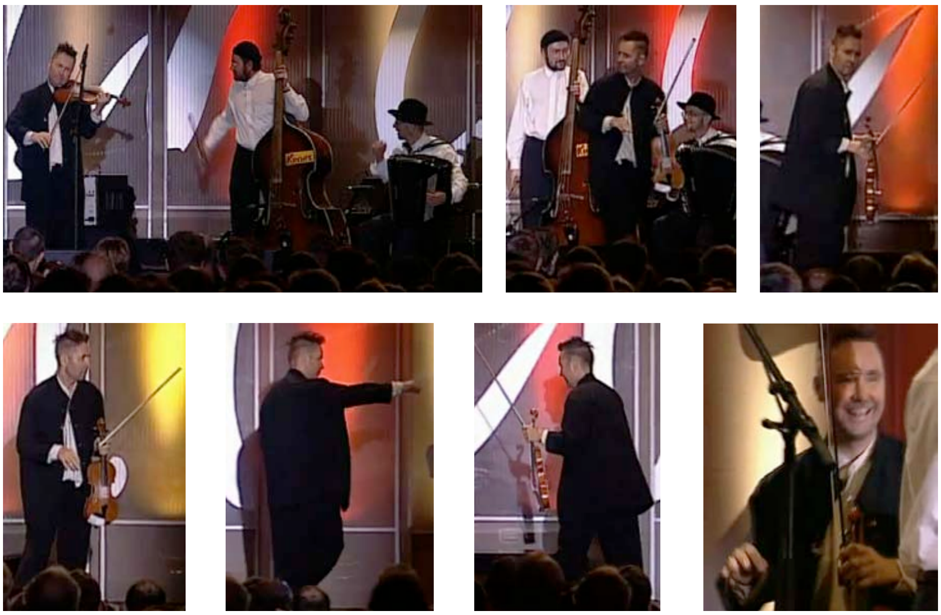


Figure 4.15: *An example of Nigel Kennedy carrying out theatrical movements in Video 4.5. The movements may also be seen as increasing the musical expectation.*

with a specific "screaming" guitar sound in the video example, and can also be seen in several of his other performances.

It is difficult to say whether this swinging arm movement is a sound-accompanying movement following the "upwards" movement in the sound, or whether it is an ancillary movement that supports the sound-producing action to actually help produce the sound. Perhaps he started using the arm movement for theatrical purposes, but that it has later become internalised as an integral part of his sound-production. Whatever the function, such a swinging arm movement can be seen throughout his career and can also be seen in the performances by several other rock guitarists. I have only observed this phenomenon informally, but it seems as if this action-sound relationship has become part of a rock guitarist's movement repertoire.

In the same way as musical phrasings and "licks" are often picked up by musicians playing together, I have seen that also the movement repertoire of a musician can be copied by others. An example of this can be seen in Videos 4.7 and 4.8, and Figure 4.17, where Eric Clapton and Tim Carmon play solos after each other in the song *Old Love* on the *CrossRoads* DVD (Clapton, 1999). Not only does Tim Carmon pick up many melodic, harmonic and timbral features from Clapton's solo, but he also adopts his movement style. This includes how they both move the torso and bend backwards with closed eyes to emphasise salient points in the melody.

I shall be careful about drawing extensive conclusions based on my brief observations

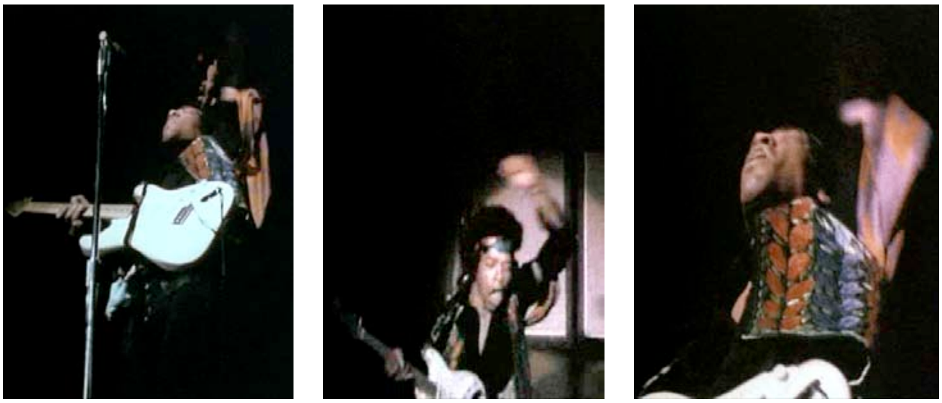


Figure 4.16: *Three images from Video 4.6 showing how Jimi Hendrix repeatedly performs a "swinging arm" movement with long sustained sounds in his solo, taken from Hendrix (1973). He used this action-sound relationship throughout his career, and also seems to have influenced other rock guitarists.*

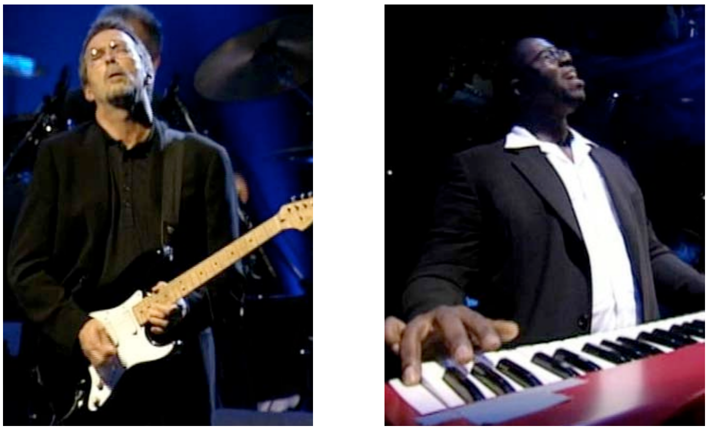


Figure 4.17: *Examples from Videos 4.7 and 4.8 of how music-related movements may be transferable between musicians (Clapton, 1999). Notice how similar Eric Clapton and Tim Carmon move in their solos, in these images a backwards bending movement with closed eyes occurring at a salient point in the melody line.*

of movement repertoires. However, it would be interesting to see more detailed studies of this, and with an ever-growing catalogue of commercial music DVDs it should be easy to find material for comparative studies. Such studies could focus on the development of performance styles over time for individual musicians, and look at how the movement repertoire of famous musicians have influenced the movements of other musicians.

4.3.9 Link to Sound

The four different types of music-related performance movements may be seen as representing a continuum when it comes to their connection to the musical sound. Figure 4.18 illustrates this continuum, showing that the sound-producing actions are, by necessity, very closely linked to the sound. Of the ancillary movement types, the support movements are often closely linked to the sound, while the phrasing and entrained movements tend to have a somewhat looser relationship. Similarly, sound-accompanying movements may have either a close or loose link to the sound. For example, "air instrument performance" may mimic closely the sound-producing actions, as will be discussed in more detail in Chapter 5.2. Free dance movements, on the other hand, often have a more loose link to the sound as we will see in Chapter 5.3. When it comes to communicative movements, they are often very loosely (if at all) connected to the musical sound.

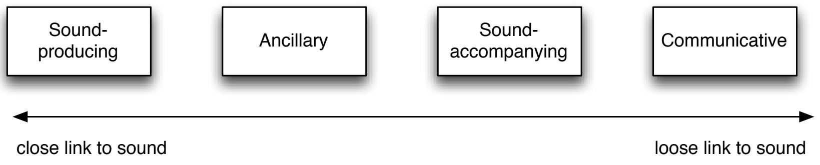


Figure 4.18: *Relationship between the four different music-related movement types and their connection to sound. Sound-producing actions are necessarily closely linked to sound, while communicative movements are often only loosely (if at all) connected to the sound. The others may vary within this continuum.*

4.4 Summary

The chapter started with a presentation of different types of *gesture* definitions, categorised in three groups: *communication*, *control* and *mental imagery*. The definitions all have in common that they focus on body movement in various ways. However, there are some important differences between these definitions that can potentially lead to confusion, particularly when we speak about intention and meaning. For this reason I have decided not to use the term *gesture* at all, but rather use *movement* to describe physical displacement, and *action* for goal-directed movements and manipulation. The rest of the chapter focused on the body and music-related movement, and particularly on the various functions music-related movements and actions of musicians play in a musical context.

Throughout the chapter examples from commercial music DVDs have been used to illustrate the various types of music-related movement. This is a deliberate choice, since I am interested in studying material of real performances. As Caroline Palmer (1997) argues, much of the early empirical studies of music performance focused on laboratory experiments, and particularly on piano performances of the Western classical repertoire. The reason for this is probably because the piano is an instrument from which a digital representation is readily available using the MIDI output of a digital piano. The many problems related to MIDI will be discussed in Chapter 6.2.6, but one of the main drawbacks is that MIDI is essentially centred around describing attacks of a sound-producing action. Analyses based on MIDI data will therefore inevitably result in a focus on sound-producing actions, omitting the other types of music-related movements.

More recent performance studies, on the other hand, are often based on motion capture recordings which reveal more of the full-body movement of the subjects. However, since most of these studies are confined to a laboratory space and setting, I find that many of the communicative movements found in stage performance are absent from such studies. There is a big difference between performing in a laboratory in front of a few researchers, and performing on stage in front of an audience. The DVD examples presented in this chapter show some of the richness of music-related movement as found in stage performances, and it would be interesting to carry out more controlled motion capture experiments on stage to study some of these movements.

This chapter has focused on the movements of performers and particularly musicians, but there are also many music-related movements carried out by perceivers. There have been few studies of such movements, so in the Musical Gestures project we decided to carry out a series of observation studies of people moving to music. These studies will be presented in the next chapter.

Part II

Exploration

CHAPTER 5
Observation Studies

How does it come that when someone voluntarily listens to a song with ears and mind [...] his body responds with motions somehow similar to the song heard?

Boethius ca. 500 (1989, 8)

This chapter presents a series of observation studies of *air instrument performance*, *free dance* to music and *sound-tracing*. The aim was to study various types of sound-accompanying movements to see what they can reveal about the perception and cognition of music.

5.1 Introduction

To try to understand more about the role of movement in our perception and cognition of music, we[1] have conducted a series of observation studies of *sound-accompanying movement*. As defined in Chapter 4.3.3, sound-accompanying movements are movements that follow some features in musical sound, as may be seen in for example marching, dancing, or "conducting" to music. In fact, it suffices to walk down the street, take the metro or go to the gym to find a multitude of examples of how people move to music, such as tapping a foot while listening to the radio, jogging in rhythm to music from a portable device, playing air drums at a concert, or dancing when going out clubbing. These are all examples of various types of spontaneous sound-accompanying movements.

[1]The air performance, free dance and sound-tracing studies have been carried out together with Rolf Inge Godøy and Egil Haga at the University of Oslo. This chapter will focus on my contributions to the studies.

In the Musical Gestures Project, we wanted to see what sound-accompanying movements might reveal about the the role of movement in the perception and cognition of music. This was based on the idea that moving to music necessarily involves choosing some features in the musical sound (e.g. melody, timbre, rhythm) to follow. Sound-accompanying movements may therefore be seen as a manifestation (although crude) of the musical features a person finds to be perceptually salient. By watching such movements we might then get a better understanding of the person's perception of the sound. For this reason we have carried out a number of observation studies in the Musical Gestures Project: free dance to music, ballroom dancing, music "conducting", "air performance" (piano, guitar, drums), and sound drawing and walking to music.

The observation studies have been exploratory in character, and the aim was not to carry out full-fledged experiments, but rather to gather some material to continue our theoretical explorations of the field. My main interest in these studies has been to develop setups, methods, techniques and computational tools with which to study music-related movement. I will not present detailed analyses of our findings in this chapter, but rather point out some of the challenges encountered, and some of the needs that inspired the technologies that will be presented in the rest of this dissertation.

5.2 Air Instrument Performance

The first set of observation studies we carried out were of so-called "air performance", what may be seen when a person is mimicking sound-producing actions, such as playing "air guitar" or "air drums". Air performance movements may be regarded as an indication of the perception of musical sound in general, and may also be more specifically related to the mental imagery of sound-producing actions perceived in the sound. A person performing in the air shows the ability to quickly extract salient features from a complex sound signal, and reproduce these features as body movement. This type of sound-accompanying movement is what we called *motormimetic sketching* (Godøy et al., 2006b, 258), because it is an approximate mimicry of the motor activity underlying a sound-producing action.

Motormimetic sketching may be seen as a kind of *goal-directed imitation* (GOADI) (Wohlschläger et al., 2003), in which the "goal" may be the onset of a tone, the high-point in a melodic line, etc. Motormimetic sketching seems to be something everyone can do. This we observed in a pilot study of air instrument performance during the Norwegian Research Days 2004, in which hundreds of people volunteered to perform on air guitar in our booth located in the city centre of Oslo. It is also interesting to see that air performance is being cultivated to virtuosity in the "air guitar" contests that have become popular in recent years.[2]

One of our main interests in the study of air instrument performance was to look at the differences along what may be called a *novice–expert continuum*. *Experts* (musicians) have extensive experience of carrying out real sound-producing actions, and we therefore

[2]The Annual Air Guitar World Championship contest has been organised as part of the Oulu Music Video Festival in Oulu, Finland since 1996 (http://www.airguitarworldchampionships.com).

assumed that they would also be able to carry out air performance effortlessly. Based on the idea of an action-perception approach to music cognition, we also believed that *novices* (people with little or no musical training) would be able to produce movements with qualities somehow similar to the original sound-producing actions. Even people that claim to be unmusical have had life-long, musical "training" (both auditory and visual) in everyday life. Thus, we assumed that everyone should be able to imitate sound-producing actions even though they may never have touched the instrument they imitate.

Preparing for the observation study, we expected that subjects would produce a number of different types of movements, everything from mimicry of sound-producing actions, to mimicry of a specific performance style, or emotional states evoked by the music. Since the subjects were allowed to perform to the same musical excerpt three times in a row, we also assumed that on the first attempt people would be able to create a rough outline of the sound-producing actions as well as some global features (mood, sense of effort, sense of speed, etc.), while during subsequent trials they would be able to perform with more details. Obviously, such a task depends on a person's motor abilities, and these abilities may influence the final result.

Air performance observation studies were carried out of three different instruments (piano, drums and guitar), but only the air piano study will be discussed in this section. This is because the imagined keyboard, representing pitch-space and point of impact for the onsets, facilitated the analysis. The next section will present a summary of our joint analysis, which has also been presented in Godøy et al. (2006b), and introduce some of my own reflections and how they led to further developments in my project.

5.2.1 Method

Location and Setup

The recording sessions took place in October 2004 at the *Intermedia* video studio at the University of Oslo, featuring a bluescreen background and high-end, DV cameras. As shown in Figure 5.1, the cameras were placed in front and to the left side of the subjects, each at a distance of 4 meters from the marked centre spot. Speakers were oriented towards the recording space, and sounds were played from one of the computers. Recordings were made to DV tapes in the cameras, while Firewire web cameras mounted on the stands below the DV cameras allowed for realtime video analysis using *EyesWeb* and the *Musical Gestures Toolbox* (see Chapter 7).

Subjects

Five subjects with different musical and movement-related training were recruited:

A. *Novice*: No musical or movement-related training.

B. *Intermediate*. Some musical training on different instruments, and some movement-related training.

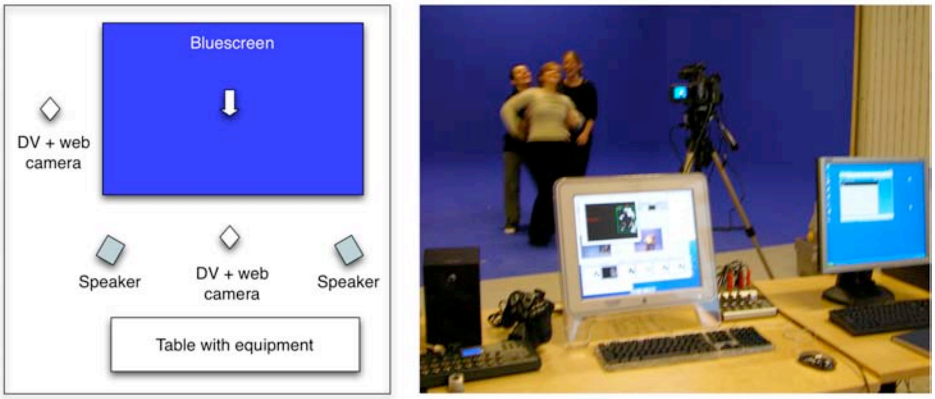


Figure 5.1: *A sketch of the setup for the observation study of air performance (left), and a picture of the equipment and recording space at the* Intermedia *studio at the University of Oslo.*

C. *Semi-expert*: Extensive musical training on several instruments and university level music studies, but no movement-related training.

D. *Semi-expert*: Extensive musical training on the piano and university level music studies, but no movement-related training.

E. *Expert*: Professional pianist with extensive university level training in performance, but no movement-related training.

Instructions

After filling out the consent forms, the subjects were informed of the purpose of the study and the procedure of the recording session. They were instructed to listen to the musical sound and try to imitate the imagined piano actions of the person performing. The subjects were also informed that the musical excerpts might or might not be familiar to them, and that they should start to move immediately when they heard the sound, even though they did not know the music. As such, their actions would lag behind the music on the first attempt, but as each excerpt would be played three times, so they would be able to adjust their actions on each repetition. The subjects were not allowed to see each other's performance, and only one subject would perform at a time. Four researchers operating the equipment and managing the session were present during recording sessions.

Musical Examples

Five music examples were selected, covering different types of piano techniques and musical styles:

1. Excerpt from the opening of Chopin's *Scherzo no. 2* in B^b minor op. 31 (17 seconds) (Pogorelich, 1999).

2. Excerpt from the opening of Scriabin's Sonata no. 5 op. 53 (10 seconds) (Austbø, 1989).

3. Excerpt from the opening from the third movement of Beethoven's 3rd Piano Concerto (16 seconds) (Duchable, 2003).

4. Excerpt from the opening of Messiaen's *Regard des Anges* from *Vingt regards sur l'enfant Jésus* (22 seconds) (Austbø, 1993).

5. Excerpt from *Encore* from *Tokyo '84* by Keith Jarrett (16 seconds) (Jarrett, 1984).

The excerpts were deliberately chosen so as to present only a few musical dimensions each: large pitch-space and salient phrases and attacks (excerpts 1 and 2), periodic and distinct textures (excerpt 3), percussive and dense textures (excerpt 4), and groove-based textures (excerpt 5). The excerpts were extracted from commercially available CDs, and recorded on one continuous track to facilitate playback and analysis. Each excerpt was repeated three times with 2 seconds of silence between similar excerpts, and 5 seconds of silence before new excerpts.

Pre-processing

The DV-tape recordings were transferred to a computer and one file was saved for each subject. Since we had recorded the audio using the microphones in the cameras, we manually aligned the digital audio tracks to secure better sound quality in the videos. The video files were cropped to align with the digital audio tracks, which ensured that all files started at the same time and were of the same duration. Since all videos are based on the same time code, it is easy to refer to time codes between files, as well as make systems which can play back multiple files in synchrony. For comparing the performances of various subjects at the same time, I developed a solution for creating *composite videos* with the Musical Gestures Toolbox. The implementation of this will be discussed in Chapter 7, but an example of such a composite video can be seen in Video 5.1.

5.2.2 Analysis

As an exploratory study, the idea was to test different types of analytical techniques to see which approaches would work better with the material. This included qualitative analysis based on observation, and quantitative analysis based on various computer vision techniques. My main focus was on the quantitative analysis, but I also participated in the qualitative analysis.

Quantitative Analysis

From the outset, we were interested in extracting various movement features from the video files. For this I used the EyesWeb[3] platform (Camurri et al., 1999), which gave access to tools for calculating the *quantity of motion* (QoM), the *stability index* and the *contraction index* of a person from a video recording, as shown in Figure 5.2. The *QoM* refers to the level of movement of the subject and is in EyesWeb calculated from the *silhouette motion image* of the person (Camurri et al., 2002). Motion images will be discussed more in Chapter 7, but generally a motion image is created by calculating the difference between consecutive frames in a video. The *contraction index* refers to the contraction/expansion of the limbs of the body, and is calculated with respect to the *kinesphere* (i.e. the imaginary bounding box) of the subject. As such, the contraction index has a high value if a person is standing with the arms and legs stretched out, and a small value when the arms and legs are close to the trunk. The *stability index* refers to the equilibrium of the subject, i.e. how much the subject moves around over time.

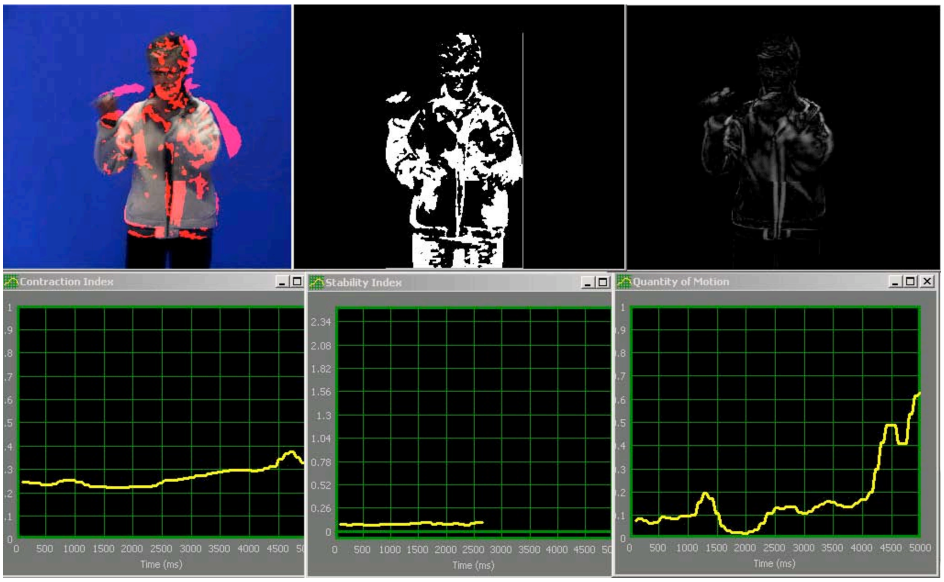


Figure 5.2: *Screenshot from an EyesWeb patch made for analysing air piano performances. The original video image overlaid with a motion image (top left), black and white motion image with a bounding box (top middle), grayscale motion image (top right), a graph of the contraction index (bottom left), stability index (bottom middle) and the quantity of motion (bottom right).*

Of the various quantitative measures mentioned above, I found the QoM to be most interesting for our material. This was because we were interested in comparing overall activity in the movements with overall activity in the music, and here the QoM seemed

[3]http://www.eyesweb.org

like a useful measure. To compare values from different subjects, I exported the results from the EyesWeb analysis patch to text files and plotted them in Matlab. Figure 5.3 shows a comparative plot of the QoM of the five subjects performing air piano. Time runs from left to right in the graphs, and the vertical axis represents the energy level of the QoM, a high value meaning that there is more movement. No smoothing (for example low-pass filtering) or normalisation was applied to the values before plotting them. The graphs reveal some general aspects of the movement that may not be so visible when studying only the videos. For example, they facilitate detecting periodicities and repeated patterns in the movement sequences, both for individual performances, but also between the subjects. An example of differences that can easily be seen, is that all subjects seem to have fairly high and even QoM values in the groove-based excerpt 5, compared to the more scattered values in excerpt 2. As such, the graphs serve as a visual overview of the movement, and help perform quick, comparative analysis.

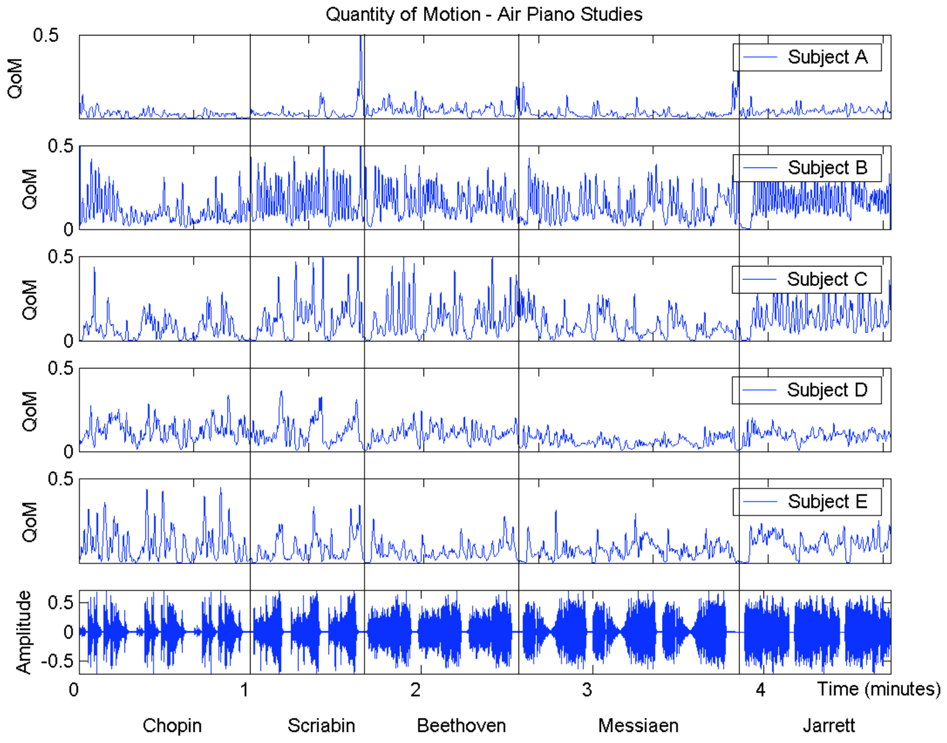


Figure 5.3: *Comparative display of the* quantity of motion *(QoM) of 5 different subjects performing air piano. Time runs from left to right, and the QoM values are measured by the number of changing pixels in each video frame, 1 being all the available pixels in the frame. A waveform of the sound is plotted at the bottom for reference. The graphs allow for the comparison of the overall level of movement throughout the session, and between excerpts and subjects.*

The graphs in Figure 5.3 also show some of the fundamental problems with using QoM in analysis. Consider for example the differences between Subject A and B. As can be seen in Video 5.2, Subject A stood rather still throughout the session, mainly moving her hands. This resulted in generally low QoM values, except for a few peaks when the subject changed position. Subject B, on the other hand, swung back and forth with the whole body, as can be seen in Video 5.3. Thus the QoM values of the performance of Subject B ended up much higher than for the performance of Subject A. The main difference between these two data sets, however, is that the QoM values for Subject A mainly refers to the movement in the subjects' arms and hands, since the body was moving so little. For Subject B, on the other hand, the QoM mainly represents the movement of the whole body, and the smaller movement in the hands are probably masked by the movement of the trunk. Observing Subject B (Video 5.3), I tend to perceptually "filter" out the movements of the trunk, and mainly focus on the movement of the hands. The problem with QoM is therefore that it only returns one value about the overall movement, and there is not really any way of seeing *where* in the image (and thus in the subject) the movement happened.

To overcome some of the problems of QoM plots, I tested some EyesWeb patches for tracking the head and the hands of a subject using a *colour blob tracking* technique (Camurri et al., 2004). This technique is based on defining the colour of a "blob" which should be followed in the image, in this case the head and the hands. Fortunately, the head and the hands were easily visible in our recordings, something which simplified the colour blob tracking process. This was due to the even background and that the subjects wore clothes with a different colour than their skin. Also, in the air piano recordings the subjects were standing in one spot and facing in one direction, so the head and the hands were always visible. As can be seen in a screenshot from the blob tracking patch in Figure 5.4, the algorithm effectively managed to locate the size and position of the head and the hands of a subject.

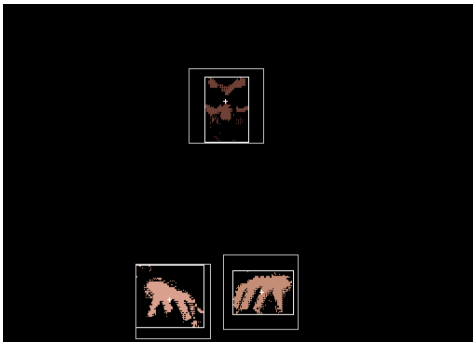


Figure 5.4: *Screenshot from an EyesWeb patch tracking the location and size of the hands and head of a subject performing air piano performance, based on colour blob tracking.*

Storing values from the blob tracking patch to a text file, I tested various types of graphing and analysis in Matlab. Figure 5.5 shows a graph of the location of the head and hands from a selection of the recording of the expert performer. This graph makes

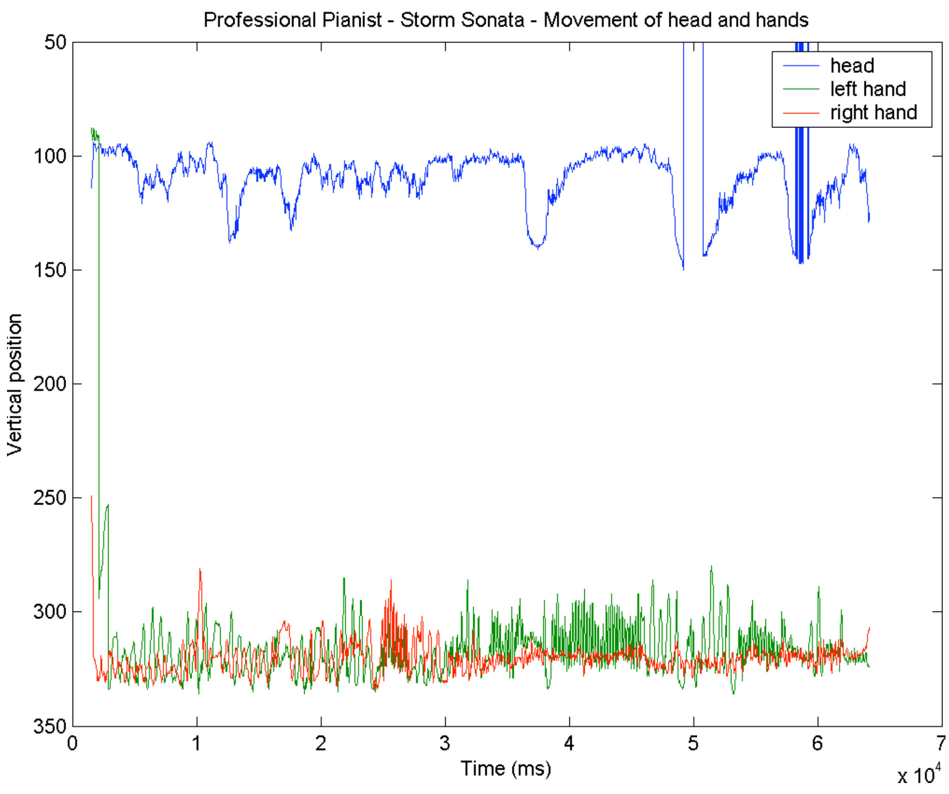


Figure 5.5: *Plotting values output from the colour blob tracking patch reveal how the head and hands were moving over time for an expert performer. While this may be give an indication of overall movement and development for the head and the hands, we found it difficult to relate these values to subjective evaluation of the source material.*

it possible to follow the *location* of the head and the hands over time, but the *movement* of the head and the hands are only indirectly present in this display. As such, we found these displays to be interesting for getting a quick overview of the movement, but not for carrying out more detailed analyses of relationships between movement and music. The complexity of the material, both the movements and the music, called for a qualitative analysis based on visual inspection.

Qualitative Analysis

The basis of our qualitative analysis of the air piano performances, which is presented in more detail in Godøy et al. (2006a), is what I call *action correspondences*.[4] Here

[4]We used the term *gestural correspondences* in Godøy et al. (2006a), but as discussed in Chapter 4 I prefer to use *action* rather than *gesture*.

correspondence is used to denote the relationship between observed movements and the sound-producing actions assumed necessary to produce the sound in the musical excerpts. A full list of these correspondences is presented in Table 5.1, showing how we proceeded from evaluating general to more detailed correspondences.

The first type of correspondence is the relationship between the *overall density* of the movement compared to the density of the onsets in the music, which may be seen as an indication of how the subject perceived the general qualities of the music. The second and third correspondences concern the *spatial distribution* of movements in relation to the pitch space in the music, in this case defined by the imaginary, vertical piano keyboard. The fourth and fifth correspondences concern the *temporal distribution* of movements, or the relationship between onsets in the movements compared to onsets in the sound. The spatial and temporal correspondences were evaluated at both a crude and detailed level, to get an idea of the "attentional resolution" of the subjects. Finally, we looked at the correspondence of *dynamics*, such as the size and speed of the movements, and *articulation*, such as movements simulating accents, staccato, legato, etc. As such, the correspondences represent many of the same features as are often evaluated when judging real musical performance, for example when grading student performers.

Judging the correspondences was based on visual inspection of the videos by each of the three researchers participating in the study. Grades were given for the performances of each subject using the following labels for the level of correspondence:

- *No correspondence*: The required, sound-producing actions are not visible. Score value = 0.
- *Poor correspondence*: The required, sound-producing actions are barely visible. Score value = 1.
- *Approximate correspondence*: The required, sound-producing actions are clearly visible, but inexact or wrong in details. Score value = 2.
- *Good correspondence*: The required sound-producing actions are clearly present and also match quite well with details in the musical sound. Score value = 3.

The average results of our judgements are summarised in Table 5.1. It is important to stress that these results are based on a limited group of subjects, and the subjective judgements of only a few researchers. Nevertheless, we found our individual judgements to be fairly consistent, and they do seem to give an indication of the differences between novices, intermediates and experts. The general impression is that the novice and intermediate subjects performed relatively well when it came to the overall activity in the music. We also found that the novices had quite a good synchrony of onsets, and an understanding of the pitch space in the music. Here we were positively surprised, as we did not expect a person who claimed never to have touched a piano to know that the bass is on the left and the treble on the right side of the instrument, and also be able to hear changing registers in complex piano music.

Figure 5.6, a screenshot from Video 5.1, shows a novice, intermediate and expert performing a part of the Messiaen example (excerpt 4) with activity in both the extreme

Table 5.1: *Average correspondence ratings of air performance actions of the subjects (A–E), on a scale from 0–3, where 3 is the best correspondence (Godøy et al., 2006a).*

	Feature	A	B	C	D	E
1.	Overall activity correspondence, i.e. *density of actions in relation to density of onsets in the music*, but regardless of pitch and onset precision	1.4	1.8	2.6	2.6	3
2.	Crude pitch-space/keyboard-space correspondence (spatial), i.e. *relative locations of hands left-to-right on an imagined keyboard at phrase/section level*	0.8	1.4	2.0	2.4	2.8
3.	Detail pitch-space/keyboard-space correspondence (spatial), i.e. *relative locations of fingers on an imagined keyboard at note-by-note level*	0.2	0.6	0.8	1.6	2.4
4.	Crude onset correspondence (temporal), i.e. *synchrony at downbeat or event level* (event instead of downbeat in cases of less or non-periodic music)	1.6	1.4	1.8	2.6	2.6
5.	Detailed onset correspondence (temporal), i.e. *synchrony of finger and/or hand movements at note-to-note level*	1.0	0.2	0.8	1.8	2.2
6.	Dynamics correspondence, i.e. *size and speed of hands/arms/body gestures in relation to loudness*	1.0	0.8	2.2	2.8	2.8
7.	Articulation correspondence, i.e. *movements for accents, staccato, legato*, etc.	0.2	0.2	0.8	1.8	2.4

low and high register. As can be seen in the video, the expert recognises the large register span immediately, while the intermediate takes about a second to adjust to the register. The novice, on the other hand, never really reproduced the extreme low part, but still seemed to notice a span in the register.

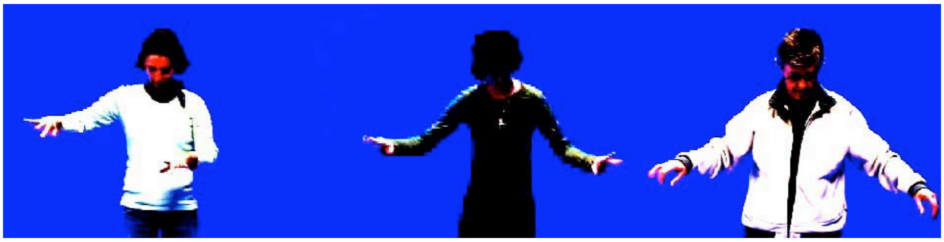


Figure 5.6: *A screenshot of the composite video (Video 5.1), showing movement activity in the extreme high and low registers for the Messiaen excerpt. The novice (left), intermediate (middle) and expert (right) subjects all managed to hear and reproduce the span in register (with increasing accuracy).*

An example of both spatial and temporal correspondence can be seen in the screenshot in Figure 5.7. This is taken from the upwards movement in the Scriabin example (excerpt 2), and shows how all subjects performed with an upward, sweeping movement and final hand lift. Even the novice performer did this sweeping movement very clearly in all three attempts, as can be seen in Video 5.4 and in the picture sequence in Figure 5.8.

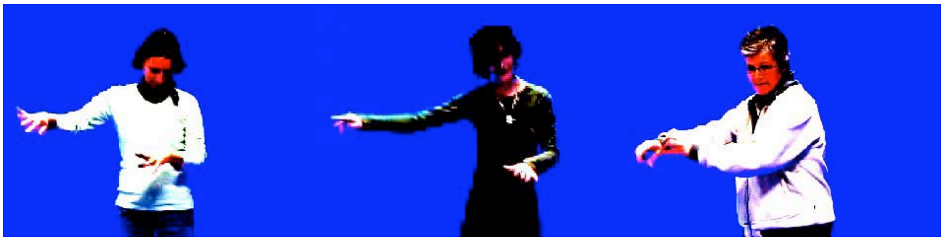


Figure 5.7: *All subjects performed the upwards movement in the Scriabin excerpt clearly, and with relatively good spatial and temporal correspondences.*

When it came to details in the spatial and temporal correspondences, as well as dynamics and articulation, only the expert excelled. Consider for example the expert's performance of the Chopin excerpt, as can be seen in Video 5.5. The picture sequence in Figure 5.9 shows how the expert performer carries out a hand lift to move from the low to the high register quite similar to what can often be seen in a real performance of this piece. Notice how the movement starts with the eyes focusing on the imaginary upper part of the piano, followed by a head movement, and a high preparatory hand lift. None of the other subjects managed to show this type of detail in their air performance.

We had deliberately chosen several musical excerpts without any clear pulse (excerpts 1, 2, 4), to see how the subjects would perform with and without a pulse to follow. As can

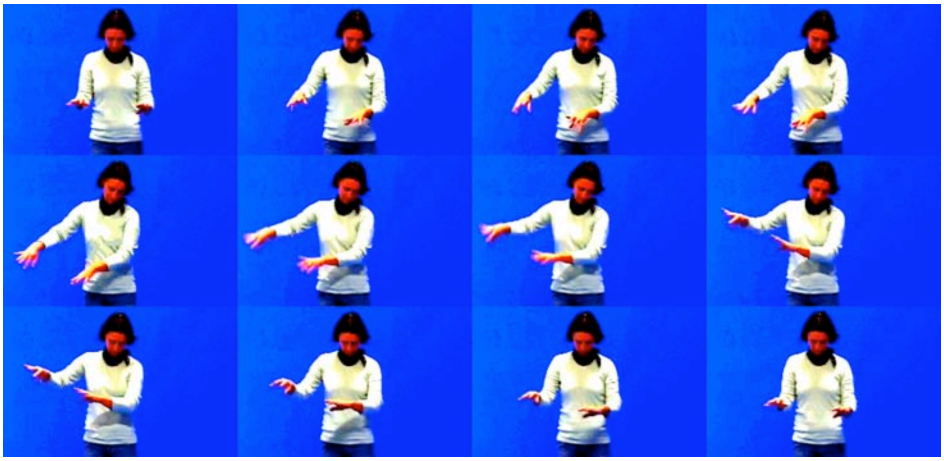


Figure 5.8: *A picture sequence of the novice performing the upwards movement in the Scriabin excerpt (Video 5.4). Images run row-wise from top left corner. Although quite approximate, this sequence shows that the novice has a good understanding of the spatial and temporal characteristics of the sound-producing actions necessary to perform this sequence.*

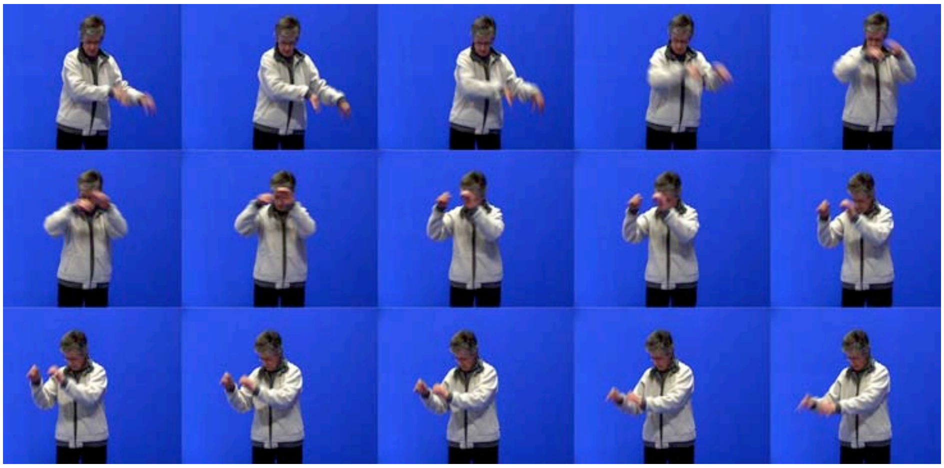


Figure 5.9: *A picture sequence of a preparatory lift found in the expert's air piano performance (Video 5.5), showing the level of detail and expressivity used in the air performance.*

be seen in Video 5.1, the subjects are more relaxed in their performances to the excerpts with a defined pulse (3 and 5). Here the subjects were able to make predictions about what would follow, and this made them able to adjust their movements accordingly. In the groove-based example in excerpt 5, it is even possible to see how they relax and to some extent *entrain* to the groove of the music.

An interesting observation can be made of the subjects' movements when the orchestra enters in the middle of excerpt 3 (which can be seen/heard from around 1:48 in Video 5.1). As shown in Figure 5.10, the novice continues to play air piano following features in the sound of the orchestra. The intermediate can be seen to hesitate a little when the orchestra enters, but also continues to play air piano following the sound of the orchestra. The expert, on the other hand, stops playing and only listens in the first trial, but follows the running arpeggios in the piano part in subsequent trials. This shows that all subjects except for the expert followed the *foreground* in the musical sound (the orchestra) and transferred this into the air instrument at hand (air piano).

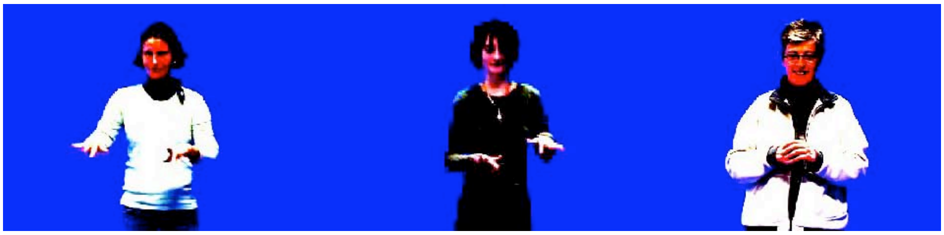


Figure 5.10: *The novice and intermediate performer continued to play piano when the orchestra entered in Excerpt 3, while the expert stopped playing.*

5.2.3 Discussion

We should be careful about drawing too strong conclusions from the rather limited material presented in this observation study, but there are certain interesting trends that can be investigated further in future studies. First, all our findings support the idea that air performance is something most people can do regardless of their musical training and capabilities. Second, there seems to be a continuum from novice to expert in the correspondences between movements and musical sound. While novices had quite good overall correspondences, they showed less detailed renditions of spatial and temporal features. Experts, on the other hand, showed a high level of detail for both spatial and temporal features, as well as articulation and dynamics. We found a similar tendency in the other air performance studies (guitar and drums/percussion), where the level of detail increased with the musical training of the subjects.

For some of the air performance studies we also experimented with the subjects wearing a wireless, homemade sensor glove, with bend sensors in the fingers and a two-dimensional accelerometer and gyroscope at the wrist. However, since we only got data from the glove and did not have any reference points in the body, these values turned out to be difficult to work with. Furthermore, using a glove seemed to distract the subjects and made them focus more on moving the hand wearing the glove and less on using full body movements. Thus, we abandoned the idea of using sensors and focused on the less intrusive camera-based analysis.

One of the interesting points about performing in the air is that there are no spatial limitations imposed on the movements being carried out. However, this might also be seen as a drawback, since performing on a real instrument involves tactile feedback. Thus, mimicking sound-producing actions in the air actually results in movements that are quite different in nature from the real sound-producing actions. For example, when performing on a real piano the focus is on hitting a key, and the movement of the finger will necessarily have to stop in the key. When performing in the air, this resistance will have to be supplied from the arm itself, which imposes a different type of movement throughout the whole action. For this reason we discussed other types of setups, for example performing on a silent digital piano, on a table with a drawn keyboard, or on a blank table. The latter we actually tested, which can be seen in Figure 5.11 and Video 5.6. This setup was more similar to a piano performance situation, because the subjects were sitting while playing, and could get resistance from the table. However, it also reduced some of the spontaneity that we were looking for, since the situation was getting closer to that of a real performance. Our main intent was to study movements as they appear in everyday life, and then performance in the air seemed more natural.

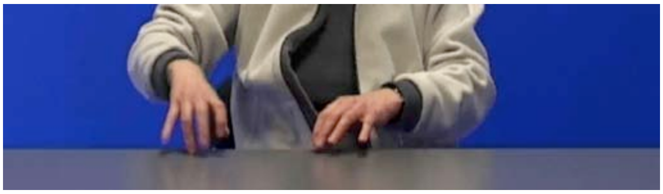


Figure 5.11: *Performing air piano on a table helped structuring the movements, and gave the performer tactile feedback (Video 5.6). However, it restricted the overall movement compared to a full body air performance.*

Working with the videos of the air performance studies, we quickly realised that manoeuvring around several hours of single-shot camera recordings was impractical. This was not only because all the video files looked similar, but also because the repeated excerpts made it difficult to know which excerpt we were watching in each of the files. For this reason I started exploring solutions to help navigating in the video files.

The first attempt to create a solution to help navigation was by adding spectrograms to the *QuickTime* movie files, something I discovered in a tutorial by van der Meer (2004). The process starts by exporting a spectrogram image of the sound from *Praat*,[5] a freeware sound analysis software developed for linguistic analysis (Boersma and Weenink, 2001). This image can then be added to the movie file using some of the functionalities of QuickTime Pro. The result is a file containing both the video, and the spectrogram in

[5] http://www.praat.org

the same display (See Figure 5.12 and Video 5.5). Since the spectrogram aligns with the progress bar in the QuickTime player, it is possible to use the spectrogram for navigation by looking at the structure of the audio. While this improved the situation somewhat, the spectrograms were so small that they did not reveal much of the sonorous features. And even if they did, they would still not say anything about the *movement* in the videos.

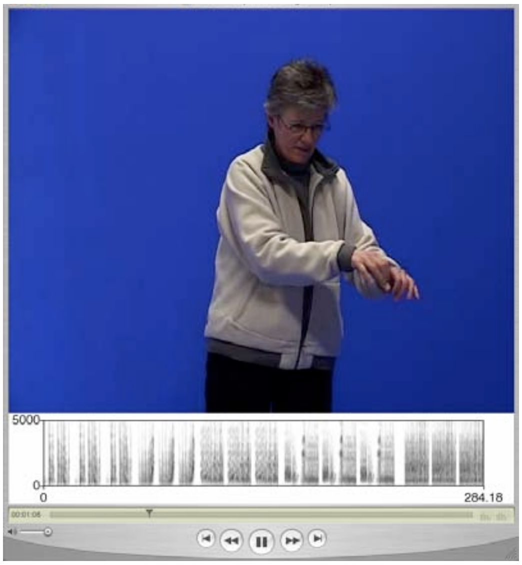


Figure 5.12: *Adding a spectrogram image of the audio below the video image helped navigate in the video files. However, it did not solve the problem of visualising the* movement *in the video.*

The second solution to help navigating in the video files was by using the timeline display in video editing software such as *iMovie* and *Final Cut Pro*. Such software usually presents *keyframes* of the video files along a timeline. In animation, keyframes are used to denote the images defining the beginning, high-points and ends of an image sequence. The term keyframe is also often used in digital video to denote the frames which are fully encoded during compression and not reliant on the surrounding frames for decoding. The "keyframes" displayed in video editing software, however, are often sampled at a regular rate and are not based on the content of the video. Thus, keyframe displays show some information about what happens over the course of the displayed images, but they still do not show the *movement* in and between the frames. Consider for example the two picture sequences in Figures 5.8 and 5.9. These images are of static postures, and do not really show the movement occurring in between the captured frames. What we needed was a tool for visualising the movement over time, and we started talking about the possibilities of creating a "spectrogram" for video. This discussion led to my exploration of *motion history images* and the development of *motiongrams*, which will be presented in Chapter 8.

Working with both the quantitative and qualitative analyses, I also discovered a number of challenges related to the videos themselves. The first problem was the lack of brightness and contrast in the video images. Since the bluescreen background was rather dark, people wearing dark clothes ended up with lower QoM values than people wearing lighter coloured clothes, since the contrast between their clothes and the background was weaker. Another problem with the bluescreen was due to the uneven lighting in the room, which resulted in the colour of the screen ranging from light blue in some areas to dark blue in others. Without any colour correction this influenced the quantitative measures calculated from the videos.

The second problem was related to the lack of pixels in the image. The recordings were done with a fixed camera placement adjusted to cover a tall person standing with his hands above his head. The idea behind this was to ensure equal recording conditions for all subjects. However, the result was that some of the short subjects filled only a small part of the image. Having to zoom in the image made the final video much smaller (in pixels) than it could have been if we had recorded all subjects close up.

Third, carrying out a two-dimensional computer vision analysis of three-dimensional body movement is a challenge, since the sense of depth in the image is lost. It is also problematic that parts of the body, particularly the hands, may become visually occluded in a two-dimensional video image. Having recordings from both the front and side of the subjects, I tried to carry out *triangulation* by combining values retrieved from the two images. However, due to the skewed angles between the images, this turned out to be impractical considering the tools, knowledge and time available.

As we shall in the next section, the challenges related to navigation in, and analysis of, the video files recorded during the observation study became even more apparent when studying free dance to music.

5.3 Free Dance to Music

The air performance studies focused on the mimicry of sound-producing actions, but we were also interested in observing how people behaved if they were asked to move freely to music. This led to another series of observation studies of "free movement" to music, where we recorded the movements of various people with different types of musical and movement experience. While all of these studies are interesting, this section will focus on the recordings of *free dance* movements to music carried out by three dance students. This material has also been presented in Casciato et al. (2005) and Haga (2007).

5.3.1 Method

Location and Setup

The setup for the free dance study was identical to the one used for the air performance studies presented in Section 5.2.

Subjects

Three female students were recruited from the Norwegian Academy of Dance. They had all been trained in classical ballet and modern dance from an early age, and were studying modern dance at the time of the recording. All three dancers had played musical instruments for several years in their childhood, but none of them had continued this into adolescence.

Instructions

To simplify working with computer vision analysis the dancers were asked to wear black clothes and differently coloured gloves and socks during the recording session. The session started with the dancers filling out the required consent forms, and they were informed about the purpose of the study. They were told that they would hear five excerpts of different types of music, each repeated three times. The instruction was to start moving when they heard the music and to move freely to the music. They were free to use different movement patterns for repeated musical examples, but they could also repeat their movements if they so preferred. The subjects were not allowed to see each other's performance, and only one subject would perform at a time. Four researchers operating the equipment and managing the session were present during recording sessions.

Musical Examples

Five musical examples were selected, each with only a few musical dimensions so as to simplify the analysis. We also made an attempt to select excerpts without a clear pulse, so that the dancers would have to continuously follow the music rather than entraining with the pulse. Each excerpt was rather short, in the range of 15-25 seconds, so that the length of each session would not exceed five minutes per person. The five selected excerpts were:

1. Excerpt from Lento (3rd movement) from Ligeti's *Ten Pieces for Wind Quintet* (Ligeti, 1998).

2. Excerpt from Prestissimo leggiero e virtuoso (4th movement) from Ligeti's *Ten Pieces for Wind Quintet* (Ligeti, 1998).

3. Excerpt from Allegro con delicatezza (8th movement) from Ligeti's *Ten Pieces for Wind Quintet* (Ligeti, 1998).

4. Excerpt from "Winter": Allegro Non Molto from Vivaldi's *The Four Seasons*, Concerto in F Minor, Op. 8/4 RV 297 (Il Giardino Armonico, 1994).

5. Excerpt from *Improvisation* by Alexander Refsum Jensenius (piano and live electronics), Marie Fonneløp (voice), Arve Voldsund (live electronics).

Excerpts 1–4 were extracted from commercially available CDs, while excerpt 5 was from a private concert recording. The excerpts were recorded on one continuous track to facilitate playback and analysis. Each excerpt was repeated three times with 2 seconds of silence between similar excerpts and 5 seconds of silence before new excerpts.

Recording and Pre-processing

The recording and pre-processing was similar to the one used for the air performance studies presented in Section 5.2. The end result was a set of time-aligned and time-cropped computer movie files for each of the subjects. A composite video of the three recordings can be seen in Video 5.8.

5.3.2 Analysis

Quantitative Analysis

Using different types of computer vision techniques to analyse the free dance performances, I encountered many of the same problems as in the air performance studies. In fact, the quantitative features extracted from the free dance videos turned out to be even more perceptually problematic than those from the air performance studies. The hand tracking in particular turned out to be difficult since the hands were often occluded when the dancers turned around. Also, calculating the QoM was problematic, since the dancers moved the whole body most of the time. This resulted in high QoM values even though the dancers did not necessarily carry out any movements which we considered perceptually relevant.

Qualitative Analysis

When it comes to qualitative analysis of dance, one of the most popular methods is called *Laban Movement Analysis* (LMA), named after the dancer, choreographer and movement analyst Rudolf Laban (1879-1958). Laban's theories are often connected to dance movements, but his analytical studies were of a general nature and have drawn more interest from other disciplines in recent years.[6] LMA should not be confused with *Labanotation*, the extensive and systematic notation system for describing dance movements (Guest, 2004). While Labanotation is a descriptive system which may be used to write and notate choreographies, LMA focuses on the functions and qualities of movement.

LMA is based on four basic elements: *body*, *space*, *shape*, and *effort* (Newlove and Dalby, 2004). The *body* element describes the body and which parts of the body are moving, while the *shape* element describes how the shape of the body changes over time. The *space* element describes how the body moves in relation to the environment, and defines the *kinesphere* as an imaginary box surrounding the body within which movements may occur. Finally, the *effort* element, one of the key elements of Laban's thinking, describes the *dynamics* of the movement. The three first elements are concerned with describing the body, its shape and its relationship to the space (Laban and Lawrence, 1947). The effort element, on the other hand, is more about describing subtleties of the movement, which may be seen as revealing the *qualities* of the movement. Laban (1980) suggested four

[6]Although Rudolf Laban's work is also used to help solve various physiological problems and improve people's movement habits, it is more general in scope than other "movement schools", e.g. *Alexander Technique* (Barlow, 1975), *Rolfing* (Feitis, 1978, 86) and *Eurhythmics* (Findlay, 1971).

subcategories of the effort element: *time* (fast-slow), *space* (small-large), *weight* (strong-light), and *flow* (direct-indirect). These subcategories should be approached by looking at the *intention* of the movement, or "placing yourself within the body of your subject", as I was explained during a seminar on LMA at the University of Oslo in December 2005 (Austvoll, 2005). It is through the subtle differences in effort that the intentions of a movement are revealed.

But how can LMA be used to analyse movement? Schrader (2004, 75) suggests starting asking questions about various features that may be observed in the movement:

- How is the movement pattern structured in time? Is it fast or slow? Is there a rhythmic structure and is it even or uneven?
- How is the movement pattern structured in space? What are the directions of the movement? Does it move horizontally or vertically? Is it straight or curved?
- How is the movement pattern structured in effort? Is the movement bound or free? Are there accents? How is the pattern different from other patterns?

LMA is a rich and well developed system for analysing movement, and we also have many different methods for analysing musical sound. The challenge, however, is to analyse the *relationships* between movement and sound. To analyse air performance, we defined a set of correspondences, mainly focusing on sound-producing actions. These correspondences were not directly useful in the free dance studies, since here we were more interested in general relationships between movement and various musical features, or what could be called *music-movement correspondences*. There have been several attempts to create systematic approaches to compare dance and music, and one of these is Hodgin's (1992) model for *choreomusical* analysis, which is summarised in Table 5.2.

Hodgin's model suggests a set of *parallels* (correspondences) between features in dance and music: *rhythmic*, *dynamic*, *textural*, *structural*, *qualitative* and *mimetic*. The *rhythmic* parallel, he suggests, relates the speed of *gesture*[7] to accents, meter or density in the music. This I understand as relating the temporal aspects of dance movement to temporal aspects in the musical sound. The *dynamic* parallel I understand as relating the size and volume of dance movements to dynamics (i.e. intensity and loudness) in the musical sound. The *textural* parallel is more straightforward, as it refers to the polyphony in either dance or music, based on the number of dancers or musicians. Similarly, he suggests a direct *structural* parallel between phrases and motifs in the dance and the music. The *qualitative* parallel correlates to the "nature of movement" in dance to timbre or articulation in the music. It seems as is Hodgin here is trying to relate some of Laban's effort element to musical timbre. This is reasonable, considering that the effort element focuses on subtleties in the movement, which may be seen as similar to timbral qualities and articulation in the musical sound. Finally, the *mimetic* parallel is used to denote relationships between (evoked) imagery in dance and music.

[7]Hodgin uses *gesture* for what I would call *action*, or in some cases only *movement*, but he also writes about musical gesture in a sense which seems similar to how it is used by Hatten (2004), i.e. an imaginary gesture evoked by musical sound.

Table 5.2: *Hodgin's (1992)* choreomusical parallels *of relationships between features of dance movements and musical features.*

	Dance	Music
Rhythmic	Speed of gesture	Accent, meter, event density
Dynamic	Volume of gesture	Dynamics of musical gesture
Textural	Number of performers	Number of instruments, homophony vs polyphony
Structural	Phrases, motifs	Phrases, motifs
Qualitative	Nature of movement	Timbre, articulation
Mimetic	Imitation, imagery	Imagery

The choreomusical parallels were initially intended for analysing choreographed dance, but they are also useful to analyse our material of free dance to music. Some of the parallels are more obvious than others; for example, the rhythmic, dynamic and structural parallels. These all relate to the level of activity in one way or another, the rhythmic at an event level, and the structural at a phrase level. Similarly, the sense of dynamics in musical sound can often also be directly reflected in the dynamics of the movements. The qualitative parallel, on the other hand, is more problematic, which is probably since both the "nature of movement" in dance and timbre in music are challenging concepts in their own rights. Thus, drawing a parallel between these two concepts does not help to make things clearer. That being said, we have seen that there are some observable relationships between movement qualities and timbral qualities in the sound in our material.

We have also seen examples of the *mimetic* parallel in our material, one which can be seen in the screenshot in Figure 5.13 from excerpt 3. Here all dancers use upwards movement in both hands and head, corresponding to the upwards arpeggios in the music. In this example the arpeggios seem to serve the function of *lifting* the dancers and their movements. Several other similar parallels may be seen in the performances, although they are often carried out differently by the dancers. For example, one dancer may display a floating movement using a hand, another a foot, and the third the whole body. While such movements may at first look different, the main point of LMA is to identify the underlying qualities of the movements, and the qualities of the movements may be equal even though the movements themselves may differ on the surface.

5.3.3 Discussion

We recorded more subjects in this observation study of free dance to music than was reported above, but we decided to focus our attention on the recordings of the three dance students. This was because they generally moved more, and in a more controlled manner, than the other subjects. This provided a rich set of movement material which I have used throughout the whole project, and which will also be discussed in Chapters 7 and 8. Similar to the quantitative analysis of the air performance studies, I found that many computer vision techniques (e.g. calculating QoM) were not particularly useful for

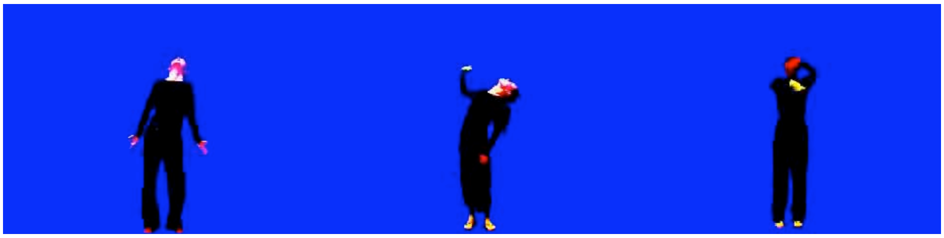


Figure 5.13: *Even though the dancers moved differently, there were certain similar characteristics, for example the tendency to "open" the movements at certain phrases in the music. This snapshot is from around 2:51 in Video 5.8 (Excerpt 3, third repetition).*

capturing the *qualities* of the movement as defined by LMA, and Hodgin's choreomusical parallels. As discussed in the previous section, this free dance study confirmed our need for tools to help navigate in large collections of video material, and display the *movement* within the videos. Thus, developing tools for displaying trajectories and the dynamics of full body movement seemed to be more relevant to our needs than trying to extract the positions of discrete points of the body.

Working with the videos from the observation studies, we also found the need for a video playback system that could allow for quickly moving around in the video files, change the playback speed and loop parts of the files. Furthermore, since we had problems with the lighting in the recordings, we found it important to be able to change brightness and contrast in realtime. We also needed to be able to zoom and crop the videos to study a particular movement or certain parts of the body in more detail. Since none, or only a few, of these features were available in typical software video players, I started developing a player myself that would cover our needs. This ended up as the *Musical Gestures Toolbox* which will be presented in Chapter 7.

5.4 Sound-tracing

The air performance and free dance studies presented in the previous sections were open and exploratory in nature, and focused on full body movement and fairly long musical examples. We were also interested in carrying out a more constrained and controlled study of the movements that people associate with sounds. As discussed above, even people without musical training seem to grasp quickly the main features of musical sounds and are also easily able to reproduce movements that correspond to some of these features. This may be seen as somehow similar to our ability to sketch quickly visual objects on paper. Thus we decided to set up a small experiment where subjects would be instructed to "draw" the movements they associated with sounds on a graphical tablet. This we called *sound-tracing* in Godøy et al. (2006a), as the aim of this study was the reproduced actions (i.e. the tracing) and not the drawings themselves. For this reason we asked subjects to draw with a digital pen so that they could not see the result of their drawing.

Sound-tracing may be seen as yet another type of sound-accompanying movement, and one which does not aim primarily to reproduce the sound-producing action or any other specific music-related movement. Rather, the idea of sound-tracing is the approximate rendition of any actions corresponding to some perceptually salient features in the musical sound. Our main interest here was to see what types of sound features the subjects would respond to, and how they would trace these features with the digital pen, for instance:

- mimicking the sound-producing action(s), e.g. impulsive, sustained and iterative.
- tracing features in the musical sound, e.g. dynamic envelope, pitch contours, timbral development, etc.
- drawing something which reflects the emotional response to the sound, e.g. being "lifted" or "floating".
- a combination of the above.

Some of these alternatives are related to specific features present in either the sound itself or in the imagery of the sound-producing action, while others are more general in nature. This span from a detailed rendering of features to general sensations of the musical sound, is what we called "variable acuity" (Godøy et al., 2006a). Another aspect of this study, and to some extent also of the previous studies, is that of *motor equivalence*. Motor equivalence relates to our ability to carry out the same task in many different ways, for example opening a door with a hand, an elbow, or even the cheek if necessary (Rosenbaum, 1991, 5). In this study, the subjects would only get a digital pen to make a tracing of the movements associated with a sound, and this would call for a "translation" from multimodal imagery into the two-dimensional graphical tablet. Sound-tracing may therefore be seen as a reduced representation of perceptually salient features in the musical sound.

5.4.1 Method

Location and Setup

The observation study was carried out in the Musical Gestures laboratory at the University of Oslo in March 2006. Figure 5.14 shows the setup in the laboratory, where an A4 Wacom Intuos 2 graphical tablet was positioned in the centre of a table. Two loudspeakers were placed on the table in front of the subject, and the sounds were played from a computer which also recorded data from the graphical tablet. A video camera was positioned to the front-left of the subjects and was used to record the sessions for later reference and analysis.

Instructions

After filling out the consent forms, the subjects were interviewed about their musical and movement-related training. Then the purpose of the study was explained, and they were

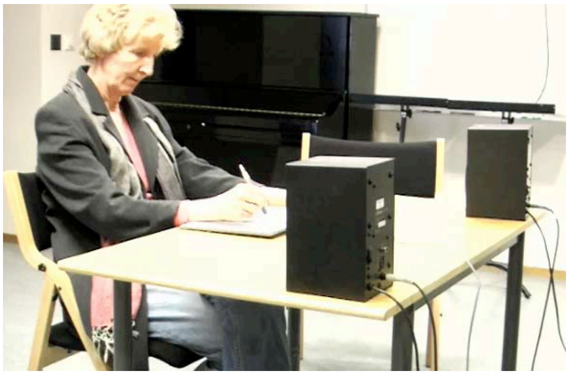


Figure 5.14: *Picture of the setup used in the sound-tracing studies. Subjects were sitting at a table with a Wacom graphical tablet in front of them. Two speakers were placed on the table facing the subject. The picture is taken with the video camera used to record the sessions.*

told that they would hear two sets of short sounds, one with 30 and the other with 20 sounds. Each of the sounds would be followed by a duration of silence equal in length to the sound being played. The instruction was to first listen to the sound, and then draw the movement associated with the sound during the silence. Note that we did not specify what type of drawings they should make nor which features in the sounds they should follow. This rather imprecise instruction was intentional, as we wanted to observe what types of features they would spontaneously respond to.

The subjects were given one minute to rest between the two sets, and to allow for saving the recorded data set. No new instructions were given in the break between the two sets. The subjects were not allowed to see each other's tracings, and only one subject would be in the laboratory at a time. Two researchers operating the equipment and managing the session were present during recording sessions. A short interview was carried out after the two sessions to check whether the subjects had had any particular listening or drawing strategies during the task, or if they had any other comments after their participation in the experiment.

Subjects

Nine subjects were recruited ranging from novices to experts: 3 subjects with no or little musical training, 1 subject with little musical, but extensive dance training, and 5 subjects with university level music performance studies. The subjects were aged 20-59 and the gender distribution was close to equal (4 male / 5 female).

Sound Examples

Two sound sets, one with 30 and the other with 20 sounds, were selected from sound compilations by Schaeffer (1998) and Delalande et al. (1996), as well as from various sound sample banks. Some sounds were also sampled from *Hyperprism* by Edgar Varese (1993) and *Concertino for 12 Instruments* by Igor Stravinsky (Parsons, 1992) to get some more complex musical sound excerpts.

All 50 sounds were between 2 and 6 seconds in duration and were chosen to represent a variety of the three main action-sound types described in Chapter 3: *impulsive*, *sustained* and *iterative*. All of the sounds were categorised beforehand as belonging to one of these three action-sound types, or a combination of these. We also categorised all of the sounds according to the pitch and timbral/textural content: *stable*, *changing/unstable* or *undefined*. The reason for this was our interest in looking at how movement in either pitch or timbre would affect the tracings made by the subjects.

The two sound sets were ordered by a gradually increasing complexity. The first set contained sounds with only one sonic event, or a group of events that we perceived as coherent. The second set contained composite sounds, several of which were composed from separate sounds presented in the first set, and with an increasing level of musical feature-dimensions. The idea was to see how the subjects would respond to the two different types of sounds (i.e. simple or composite), and whether they would perceive the more ambiguous composite sounds as one holistic sonorous object or as a combination of individual sound objects (or both).

Recording and Preprocessing

For the experiment I developed a Max patch to play the sounds and to record data from the Wacom tablet. The patch was built around the **wacom** external,[8] and data were recorded from the tablet at a fixed sampling rate of 100 Hz. The tablet may be sampled faster than this, but I found 100 Hz to be a good compromise between data quality and memory/CPU usage. The data were recorded using the **coll** object in Max and stored in a text file. Each row in the text file consisted of the time code (in milliseconds from the beginning of the experiment) as an index number, followed by the five tablet output variables (x-position, y-position, pressure, x-tilt, y-tilt) and finally the number of the sound being played. The structure of the file looked like this:

```
Time,  x-position y-position pressure x-tilt   y-tilt sound;
34114, 0.763965   0.442155   0.       0.233301 -0.4   5;
```

As may be seen from the example above, the 16-bit decimal range (0–65535) returned from the **wacom** external was normalised to a decimal range (0.–1.) in the text file. Values were recorded continuously in each session, so for each subject two data files were stored, containing 30 and 20 recordings respectively.

After the sessions the data files were processed in another Max/MSP patch which read the text files, parsed the content and saved an image for each of the sounds. This was done by drawing pixels in a **jit.lcd** object using the position (x,y) values, and controlling the "pensize" parameter of **jit.lcd** with the pressure values from the recordings. This make the output images three-dimensional, since the thickness of the line is based on the pressure of the pen used when tracing. We did not inform the subjects that the pressure had any effect in their tracings, but we still thought it would be interesting to include these values in the analysis since they were readily available. The two other recorded

[8]By Jean Marc Pelletier (http://www.jmcouturier.com).

dimensions (x-tilt, y-tilt) seemed to be less interesting in this study, so they were left out of the analysis.

We soon realised that it was difficult to analyse each of the 450 generated image files separately, so I made a small script[9] to output combined images of the 9 tracings for each sound (see Figure 5.15). In these combined images, the 9 tracings are ordered after the session number, counting from the top left corner:

- Image 1: the subject with dance experience.
- Images 2, 4, 8: the subjects with no or little musical training.
- Images 3,5,6,7,9: the subjects with university level musical training.

However, even with only 50 combined images we had problems finding our way around the material, so I ended up making QuickTime movies containing both the images and the sound files, as can be seen in Videos 5.9 and 5.10. These videos made it possible to carry out comparative analysis of the tracings with respect to the sounds.

5.4.2 Analysis

As described earlier, we had been uncertain about what features in the sounds the subjects would react to, and how they would choose to represent these in their tracings. For example, in one of the trial experiments leading up to the study, one of the subjects consistently sketched the physical properties of the imagined objects, for example drawing a trumpet for the trumpet sound, and a cymbal for the cymbal sound. This was probably due to a misunderstanding of the instructions and did not happen during the actual recording sessions, but the subjects still chose to focus on different features in the sounds.

Consider the different tracings of the impulsive sound of a cymbal (Audio 5.1) in Figure 5.15. Here one subject traces the loudness envelope of the sound, some imitate the sound-producing action (pressing down, which results in only a dot), while others draw a circle or spiral which may be seen as imitating some of the "circularity" in the timbre of the sound. These examples may be seen as three different types of sound-tracing, focusing on sound-production, timbral features or temporal development, respectively. In general it seems as if many subjects chose one of these three different types of tracing styles and stuck with that throughout the sessions.

Figure 5.16 shows the tracings of the iterative sound of a bamboo rainstick (Audio 5.2). Here all subjects represent the iterative character of the sound by drawing "broken" lines, but the directions used for tracing are less consistent. Most people drew from left to right, a tendency which seems to be general throughout the sessions, and which may be bound to our life-long experience with writing from left to right in Western languages. Similarly, we also noticed that the subjects would tend to use the vertical axis of the tablet for pitch space, with higher pitched sounds placed in the upper part of the tablet. This may also be culturally based, and fits well with our use of metaphors like "the sound is

[9]This was done with the "catalog" functionality in the image software *GraphicConverter*.

going up". It would be interesting to see a follow-up study of this experiment in a culture which is not based on these up-down and left-right concepts of time and pitch.

We also found a high level of agreement in the parsing and rendering of sound events. An example of this can be seen in Figure 5.17 where all subjects parsed the three descending chirp sounds (Audio 5.3) as three separate events, even though there were some individual differences in the tracings. Some traced upwards, some traced downwards, while some moved up and down. Most of the subjects made the three events equal in size, shape and direction, while some made them slightly different. Despite these individual differences, everyone seemed to have a solid perception of the sound as consisting of three separate events.

More divergent results were found in the tracings made for the composite sound set. Figure 5.18 shows the tracings of a composite sound consisting of an attack (by a triangle), a descending, sustained sound (violin and cello) and a final shorter, iterative percussive sound (Audio 5.4). It is interesting to see that all subjects traced only one sustained sound, grouping the sounds of the violin and the cello into one sound event. Furthermore, none of the subjects indicated the triangle attack in their tracings, and only some of the subjects traced the iterative sound suffix. Whether the subjects that did not trace the iterative sound did not perceive this sound as a separate sound event, or whether they just thought of it as an embellishment of the sustained sound remains an open question.

Obviously, this sound-tracing study is dependent on the subjects' drawing styles and skills. For example, a person who is more experienced in drawing with a pen will have an advantage in carrying out this task. On the other hand, a person with poor drawing skills but with a high level of musical training may be able to create tracings that more precisely reflect features in the musical sound. To study these types of discrepancies between subjects, I made a series of composite images containing all the tracings made by each subject. Figure 5.19 shows all the tracings of the short sounds by subject 1, who had little musical training but extensive training in dance and drawing. This I believe can be seen in the tracings, where the subject used the pen actively, and mainly followed various sonorous qualities in the sounds. Figure 5.20 shows the tracings by Subject 9, a music student, who used the pen more carefully, and also tended to imitate the sound-producing actions rather than only following features in the sounds. Whether people with musical training listen more in terms of sound-producing actions than others is still an open question, but this seems to be a tendency in our material.

We were initially interested in finding some statistical methods with which to analyse the data. However, we quickly realised that there were many interesting aspects that could easily be observed only from visual inspection of the static images. As the material showed, there are many underlying qualities in the tracings that may not be easily detectable by machine-learning techniques. For example, consider the tracings in Figure 5.15. Here, two subjects pressed the pen down (tracings 2 and 5 when counting row-wise from upper left right), which resulted in a dot in the static image. These tracings seem different from tracing 4 and 6, but the video recording from the session reveals that the pen actions produced by the subjects have many of the same qualities. From the video, we can see that all of these tracings share many qualities with the three circular

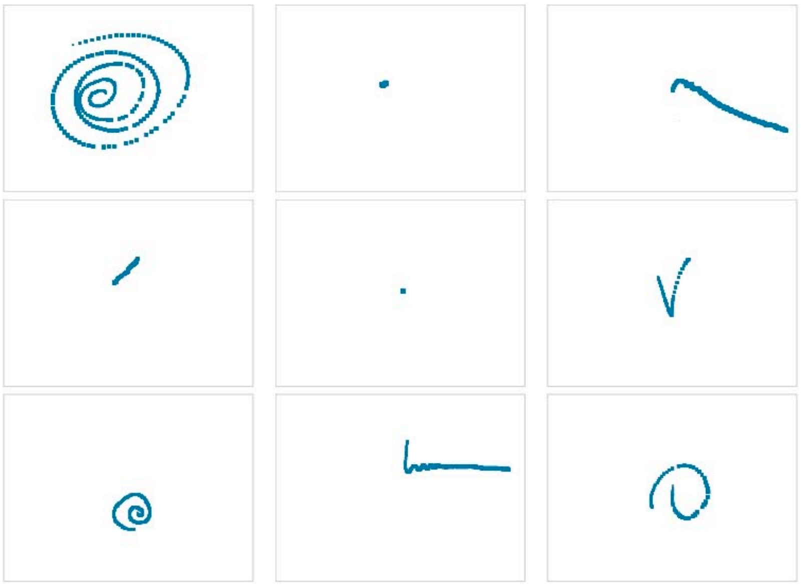


Figure 5.15: *The 9 subjects' tracings of the impulsive cymbal sound (Audio 5.1). Notice the three different tracing types: imitating the sound-producing action (pressing down which results in a dot), tracing sonic features (e.g. loudness envelope) and tracing the "circularity" in the timbre.*

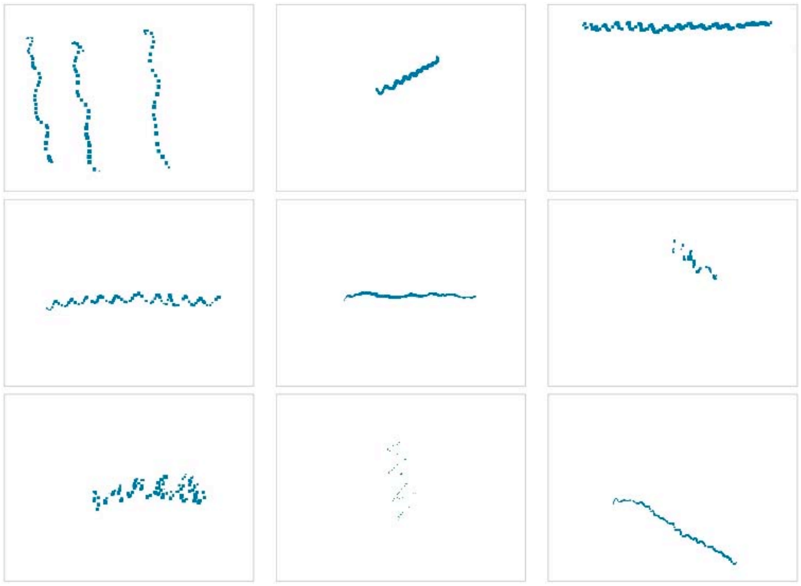


Figure 5.16: *The tracings of the iterative sound of a bamboo rainstick (Audio 5.2) all represent the iterative character of the sound, although different directions were used in the tracings.*

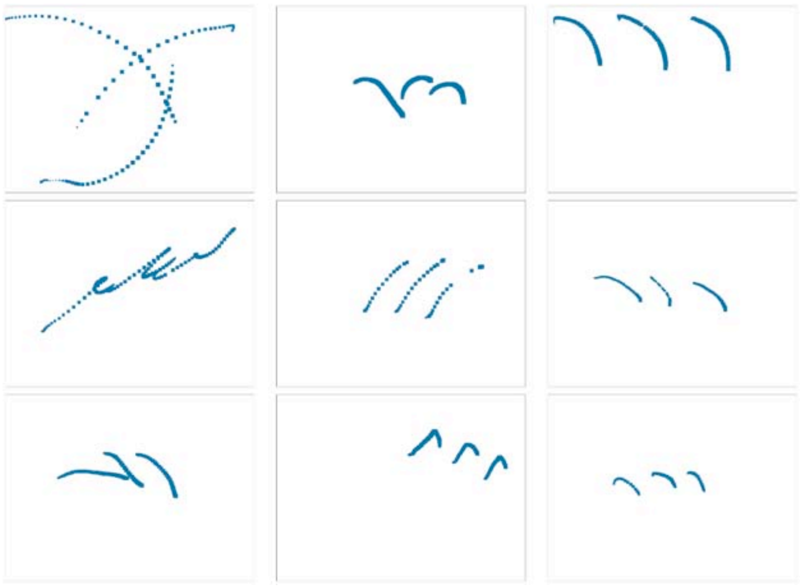


Figure 5.17: *Tracings of three separate descending chirp sounds (Audio 5.3). All subjects identified the sounds as three discrete events, but traced them somewhat differently.*

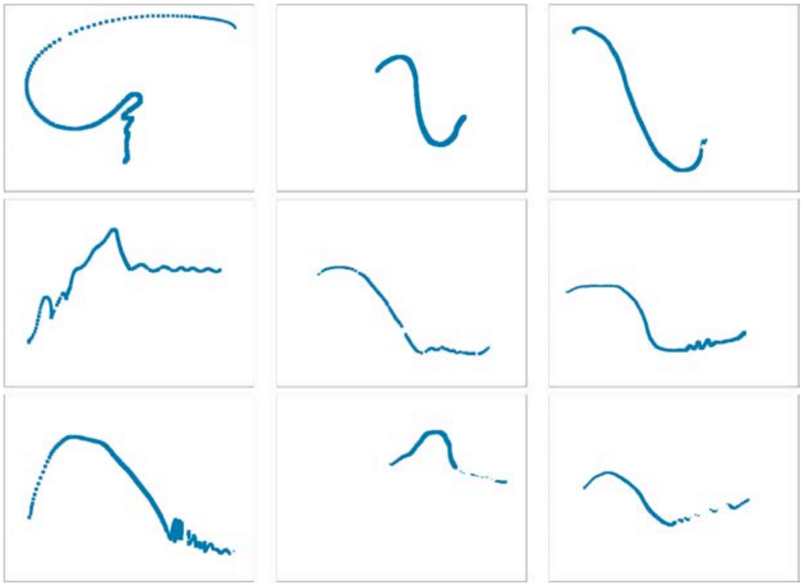


Figure 5.18: *Tracings of a composite sound consisting of two descending sustained sounds with an iterative suffix (Audio 5.4). All subjects traced one sustained sound, but only some subjects included the iterative suffix in their tracings.*

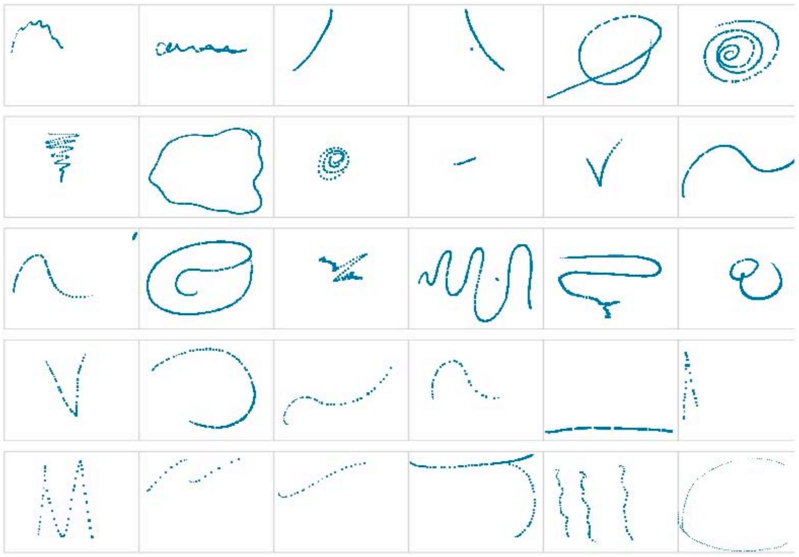


Figure 5.19: *All tracings made by Subject 1 for the first sound set (short sounds). This subject used the whole space of the tablet while tracing, and also varied the pressure level considerably for the different sounds.*

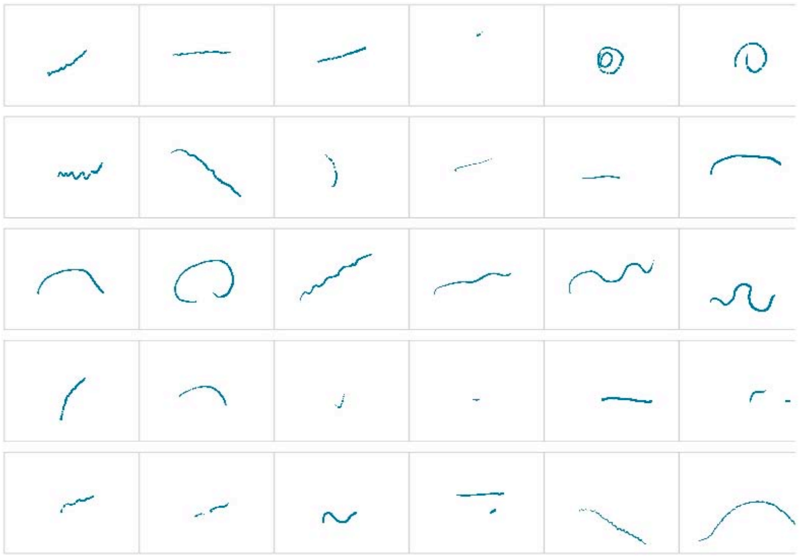


Figure 5.20: *All tracings made by Subject 9 for the first sound set. This subject used a more constrained drawing space, and focused on imitating sound-producing actions more than tracing sound features.*

tracings (1, 7, 9) in that they all start with an attack reflecting the impulsive quality of the sound. Such observations may be difficult to make with a quantitative approach, but are easily detectable by visual inspection.

We did not go into details when it came to the temporal aspects of the tracings, but there were some striking differences in how the subjects handled the concept of time in their representations. Looking at the subjects when they started to draw, there seemed to be a tendency for subjects to start tracing the impulsive sounds right after hearing the attack of the sound and long before the resonance of the sound had decayed. This was the case even though the subjects had been told clearly to start tracing *after* they heard the whole sound. Thus, starting to draw after the attack seems to support Godøy's (2001) idea of a clear distinction between how we perceive the excitation and resonance of objects, as discussed in Chapter 3. This was most obvious with the short impulsive sounds, even though it could also be observed with some of the short sustained and iterative sounds. However, for the composite sounds all subjects tended to wait for the silence before they started tracing. This implies that in these composite sound examples the subjects identified a more complex sound and/or sound pattern and felt a need to pay more attention to the whole sound before they could make their tracings.

5.4.3 Discussion

The underlying idea of the observation study presented in this section was to use sound-tracing as the basis for understanding more about the perception of short sounds. A general observation is that subjects with no or little musical training tended to focus on overall sonic features of the sounds (e.g. envelope) in their tracings. Subjects with musical training, on the other hand, seemed to imitate sound-producing actions to a much larger extent. The musically experienced subjects also seemed to be better at discriminating between separate sound objects in composite sounds, while the non-trained subjects more often traced such composite sounds as coherent units. Again, we should be careful about drawing too strong conclusions from the limited material in this study, but these are some phenomena that would be interesting to study further.

Several objections may be made to the setup of this study, particularly related to using a graphical tablet. One problem with using a graphical tablet is the lack of visual feedback. This was deliberate, however, as we wanted the subjects to focus on creating a movement that matched their experience of the music, rather than thinking about the visual part of what they had drawn. Another disadvantage is that a graphical tablet has few degrees of freedom. Even though we recorded all of the five independent degrees of freedom (x-position, y-position, pressure, x-tilt, y-tilt) of the tablet, we only instructed the subjects to use two of them (x-position, y-position). This was also intentional, since one of the main aims was to study how the subjects would adapt to the constraint of a 2-dimensional graphical tablet, by the idea of translation by motor equivalence. In the future, it would be interesting to carry out a similar study with different input devices, for example joysticks or haptic devices, or even a full-body motion capture system, to see whether and how that might change the results.

One rather unfortunate problem with the setup used in this observation study was the way data were recorded in the Max patch. As described earlier, the values from the Wacom tablet were recorded into one large text file for each session. One of the columns in the text file contained the number of the sound being played, and this number would change when a new sound was loaded and played back. This in itself worked well, but problems arose when subjects were still drawing when a new sound was loaded and the sound number was updated in the text file. This would result in parts of the tracing being recorded with the succeeding data set. For most of the simple sounds this was not a problem, but for many of the sounds in the second set, parts of the tracing of one sound were shifted onto the following image, as illustrated in Figure 5.21.

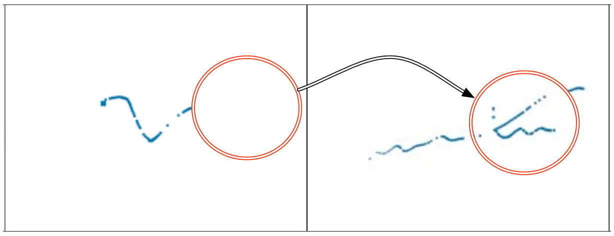


Figure 5.21: *Two subsequent tracings exemplifying how data from one tracing has been shifted onto the next tracing when the subject continued drawing after a new sound had started playing.*

There was no easy way to solve this problem for the data that we had already recorded, except for manually copying and pasting parts of the images, so I left the error in the images (as can be seen in some of the tracings in Video 5.9 and 5.10). In future experiments, it would be wise to redesign the recording patch to avoid this problem. One possible solution is to record data only when the sound is not playing, but this would also cut out any tracing that subjects might do before a sound has stopped playing. Another solution is to indicate more clearly when tracings should be carried out, or to make subjects advance manually to the next sound when they are done tracing. However, these solutions would work against our initial idea of getting a quick and spontaneous response to the sound stimulus, as subjects then would have time to start thinking about the sounds and their tracings. Presenting the sounds as rapidly as we did forced the subjects to react spontaneously. Nevertheless, figuring out a way to solve this problem of data recording is important for similar future studies.

5.5 Summary

This chapter has presented three observation studies of sound-accompanying movements: *air instrument performance*, *free dance* to music, and *sound-tracing* of short sounds. The aim was to understand more about the mental representation of action-sound relationships and music-movement correspondences by studying various types of body movement to music. Even though I should be careful about generalising too much from our limited material, I would say that the analyses support the idea that action-sound couplings are an integral part of our experience of music. We also see that both novices and experts alike are able to (re)produce movements that correlate well to features in musical sound. Not surprisingly, people with musical training show a better ability to discriminate details in the musical material, and they also tend to chunk segments in complex music and composite sounds at a more detailed level than people with no or little musical training. Still, we find that even people who themselves claim to be "unmusical" show a higher level of sound perception and discrimination than we expected.

The observation studies presented in this chapter also showed the need for better methods, tools and technologies. These needs formed the basis for development being undertaken during my project, including:

- *navigation* in multidimensional data and media. Working with large collections of data and media is a challenge, and resulted in my exploration of *motion history images* and development of *motiongrams* as will be discussed more in Chapter 8.
- *synchronisation* of audio, video, recorded data and associated analysis data. In several of the observation studies we recorded video separately from audio and other data, and had to manually align the different recordings. It would have been more efficient to use a standard for storing these media and data in a coherent and consistent manner. This will be discussed further in in the context of the *Gesture Description Interchange Format* (GDIF) in Chapter 9.
- *formalisation* of how movement data are stored. Working with other researchers and institutions it is important to have formats and standards that can be used to share data. This will also be discussed in Chapter 9.
- *tools* for working with all of the above. For each of the observation studies I developed customised software in Max/MSP. This collection of tools developed into the *Musical Gestures Toolbox* which will be presented in Chapter 7.

However, before getting to these chapters on methods, tools and technologies, I will present my explorations of the design of music controllers. The observation studies in the present chapter have focused on relationships between movement and music in *perception*. But I have also found it important to study relationships between movement and music in *performance*. Or, more precisely, I have been interested in studying sound-producing actions in electronic instruments in which the action-sound relationships are created artificially. By combining the theoretical knowledge from Chapter 3 with the analytical knowledge from the present chapter I shall discuss how we can create more intuitive action-sound relationships in electronic devices.

CHAPTER 6 Developing Music Controllers

We need new instruments very badly.

Edgar Varese, 1916
(Levenson, 1994, 294)

This chapter starts by discussing some design principles of electronic controllers for musical applications. Then follows a presentation of how game controllers may be used in music, and of various prototype music controllers which have been built during the project: *Cheapstick*, *music balls* and the *Music Troll*.

6.1 Introduction

The previous chapters have focused on the theoretical foundations for, and observation of, action-sound couplings and relationships. This chapter will present my practical exploration of such couplings and relationships in *musical instruments*. Figure 6.1 shows how a musical instrument, whether it is acoustic or electronic, may be seen as a mediator translating body movement into sound.

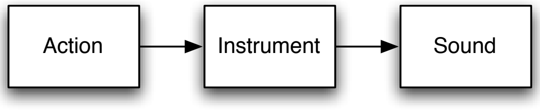


Figure 6.1: *A musical instrument can be seen as a mediator between action and sound.*

In an acoustic instrument, sound is usually produced by supplying energy to an object which starts vibrating. An electronic musical instrument, on other hand, is often based on a *controller*, a *sound engine*, and *mappings* between the controller and the sound engine, as illustrated in Figure 6.2. Here *controller* is used to denote the physical device with which the user interacts. A controller will not produce any sound in itself, but will have to be connected to a *sound engine* which can output sound. The sound engine may be implemented in either hardware or software, and may be based on various types of sound synthesis or sampling techniques. In analogue electronic instruments the sound engine is usually made in hardware using analogue circuitry, while most digital electronic instruments are run as *programs* on a computer or microcontroller.

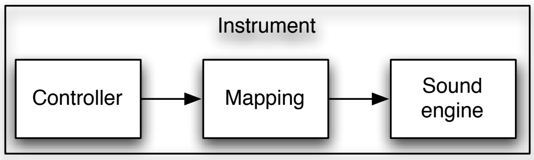


Figure 6.2: *A digital musical instrument is built up of a controller, a sound engine and the mappings between controller and the sound engine.*

For both analogue and digital electronic instruments, the *mappings* are made by either physically or virtually connecting outputs from the controller to inputs in the sound engine. Since there is no direct connection between the controller and the sound engine in an electronic instrument, they are usually designed and implemented separately. This is different from the design of most acoustic instruments in which it is usually difficult to separate the control part from the sound engine. As such, designing controllers and testing them with various types of sound engines is a potentially powerful way of exploring action-sound relationships in practice, and may provide valuable insight into some of the underlying features of our perception of action-sound relationships.

Following the ecological and embodied perspectives presented in Chapter 2, I have been particularly interested in developing controllers that are simple and intuitive for humans. Traditionally, a lot of technological design seems to have been constrained by the technology rather than focusing on the interaction with the human body. This has changed slightly over recent years, particularly thanks to the commercial success of devices like Apple's iPod, where simplicity has been a key factor in both the hardware and software design. This success has led to a revival in the interaction design community, where for example John Maeda has formalised the "ten laws of simplicity", summarised as "[s]implicity is about subtracting the obvious, and adding the meaningful" (Maeda, 2006, 89). My aim has been to explore simplicity in music controllers, and to create controllers that are so easy and cheap to build that they can be affordable for "everyone".

6.2 Some Design Issues

Recent years have seen a rapid escalation and growth in the international community surrounding the annual conferences called *New Interfaces for Musical Expression* (NIME).[1] These conferences have institutionalised the development of music controllers and instruments, and have resulted in an alternative community based on interdisciplinarity

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

and artistic/musical experimentation. It is interesting to note that the people in this community are often not based at music departments and conservatories, but come from disciplines such as engineering, psychology, architecture, design and art. This interdisciplinarity is probably one of the reasons why there has been such a rapid development in the field over recent years.

The recent innovation in the research-based and underground world of music technology may be seen as a response to the decline in the number of new *commercial* digital musical instruments over the last decades. Piringer (2001) argues that while the commercial interest in manufacturing new controllers and instruments has grown rapidly since the advent of the MIDI standard in 1983, the industrial *innovation* has largely declined. Consider, for example, some popular commercial music controllers and instruments shown in Figure 6.3. Even though these are different *models*, most of them represent just a few basic design *types*:

- controllers modelled on an acoustic instrument. These are most often based on the piano keyboard, although there are a few controllers modelled after wind instruments and drum kits.
- controllers based on a "knobs-and-sliders" paradigm. Here the idea is often to add as many sliders, buttons, knobs, etc. as can fit on the controller.
- a combination of the above.

As argued previously, a piano controller may be good for playing impulsive sounds, which is also the natural action-sound coupling found in acoustic pianos, but it is far from ideal when it comes to controlling other types of sound models. Playing a continuous sound model (e.g. of strings) with a piano controller leaves the user with little direct control over the sound. Similarly, controllers based on the "knobs-and-sliders" paradigm may be good for directly controlling the myriad of parameters in some sound synthesis models, but are often unintuitive for users that are not so familiar with the inner workings of the sound synthesis model.

The lack of innovation in commercial music technology is something that has been discussed extensively in the music technology community over the last decades. For example, more than a decade ago Winkler (1995, 4) wrote:

> Indeed, many of the important discoveries presented at previous ICMC[2] conferences regarding interactive systems, and issues surrounding real-time processing, sensing, and scheduling, seem to have gone largely unnoticed by the corporate world and popular media as they take their first baby steps toward creating intuitive sensors and systems that still await appropriate content. [...] there will be a strong need and increased opportunities to create music in response to human movement.

[2]Here Winkler is referring to the annual *International Computer Music Conference* (ICMC) organised by the *International Computer Music Association* (ICMA).

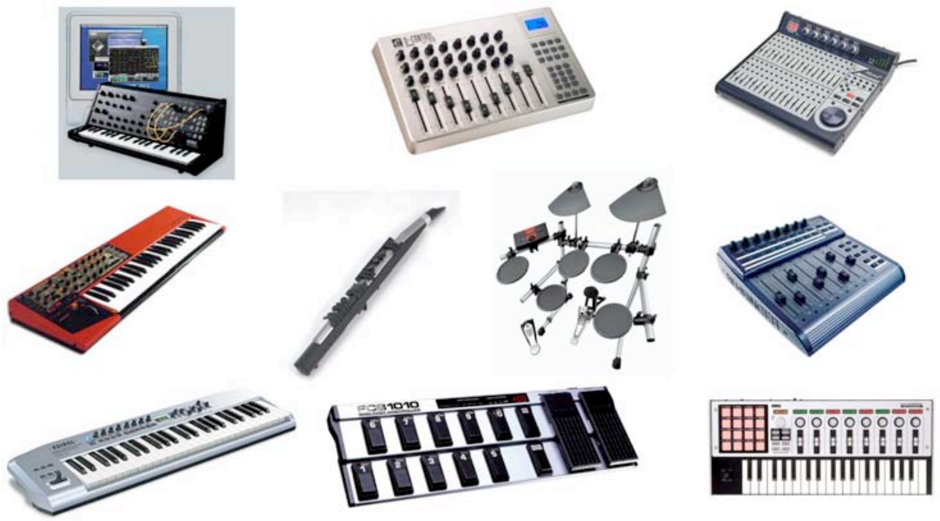


Figure 6.3: *Various commercial digital instruments and controllers, from top left:* Korg Legacy MS-20 *controller emulating the old analogue* MS-20*,* M-Audio UC-33e *controller with lots of faders and dial buttons,* JLCooper CS-32 *microcontroller with even more faders,* Clavia Nord Lead 2 *virtual analogue synthesiser and keyboard,* Yamaha WX5 *wind controller,* Yamaha DTXplorer *electronic drum kit,* Behringer BCF-2000 *controller with motorised faders,* Edirol PCR-50 *PC keyboard,* Behringer FCB1010 *foot controller, and* Korg microkontrol *keyboard with faders, buttons and pads. Note that all these commercial devices are either modelled on an acoustic instrument or are primarily based on a "knobs-and-sliders" paradigm.*

The same could probably have been written today, as the music technology industry continues to release controllers and instruments based on a design philosophy that has changed little over the years. On the positive side, today's controllers are generally faster, sturdier and cheaper than they used to be. The advent of the MIDI standard has also made controllers flexible when it comes to interchanging them and using them with various sound engines.

The last decade has seen a shift from hardware engineering to software implementation in both research based and commercial music technology. This could have resulted in a radically new approach to music technology design, since few of the traditional hardware restrictions apply when creating software. Unfortunately, many commercial software synthesisers and sequencers are often modelled on the hardware they have replaced. One thing is creating virtual copies of old analogue synthesisers, with the complete looks and functionality of the hardware device, which may be said to have some historic interest. But creating on-screen graphical user interfaces of new software synthesisers with old-style sliders and buttons, and giving the user only a mouse to control them, is in my opinion not very user-friendly. In fact, the large market for MIDI controllers designed to work with these software synthesisers shows that users prefer physical to virtual control. Unfortunately, what could have been a radically new approach to music technology has resulted in hardware controllers and software synthesisers that resemble analogue mixing desks and hardware synthesisers. Thus, what might be seen as a leap when it comes to implementation and engineering, has not resulted in a radically different working environment for electronic musicians.

6.2.1 Action-Sound Relationships

Developing controllers separately from sound engines is practical since it allows for the reuse and exploration of action-sound relationships. However, the focus on *generic control* is unfortunate and breaks with my ideas of action-sound relationships being important to our performance and perception of music. Creating a music controller without specifying the types of sounds that it can be used to control, or creating a sound synthesis model that can be played with any generic controller, violates the idea of our perception being based on action-sound couplings. The result is that we end up with *music systems*[3] that are unintuitive for both the performer and the perceiver. Going to electronic music concerts, of any genres, I often hear people complain that they do not *see* the relationship between what the performer does on stage and the sound that comes out of the speakers. This separation of actions and sounds is unfortunate both for the performer and the perceiver, as it alienates the performer from the perceiver due to the lost communication in the performance.

Early analogue electronic instruments had a close relationship between control and sound, and the separation of controllers and sound engines emerged with the advent of digital devices, and particularly with the computer. One reason for this was that com-

[3]Here I use *music system* rather than *musical instrument*, since there may be only a loose connection between the controller and the sound engine.

puters could not run the sound synthesis models in realtime. Thus, much of the early computer music focused on sound synthesis techniques and algorithmic composition, and the music was played from tape at concerts.

As computers became faster and smaller, *interactive* computer music became a reality, and the performer was re-introduced on stage in music technological performances. Garnett (2001, 25) argues that this new interactive computer music opened up for new aesthetic qualities, since the performer could add nuances to the performance through interpretation of the scores, and through the physical and cognitive constraints that a computer does not have. But adding a musician to perform with the computers also showed the need for better and more fine-grained controllers, to get the level of control a musician may have over an acoustic instrument. It also became apparent that it was necessary to "reestablish a balance between the body and the intellect" in the design of interactive computer music systems (Zicarelli, 1991, 76).

6.2.2 Design principles

Expanding the model of an electronic instrument in Figure 6.2, we may arrive at a model of a digital musical instrument as shown in Figure 6.4. In this model, the *controller* is represented as a device for capturing and analysing motion, and the *sound engine* is represented as a sound synthesis model. Miranda and Wanderley (2006, 4) argue that deciding upon the motion capture solution and that sound synthesis model to use are at the heart of the design process of digital musical instruments.

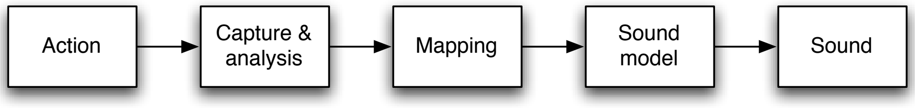


Figure 6.4: *A slightly more complex mapping chain, including the motion capture and analysis stage and the sound model.*

Starting with the motion capture part of the model in Figure 6.4, a key challenge is to decide which *sensors* to use in the controller. A sensor converts physical energy into electricity in the machine, and may therefore be called the "sense organ" of a system (Bongers, 2000, 44). While any standard industrial sensor may be used for musical applications, Marshall and Wanderley (2006) argue that it is important to evaluate carefully sensors with respect to the application they are intended for. For example, a bend sensor may be used to control anything from pitch to loudness, but it may not be the best sensor considering the actions offered by the controller, or the parameters available in the sound model. Making an intuitive instrument therefore calls for developing a controller to capture the intended actions, and choosing a sound synthesis model with input parameters that fit the outputs of the controller. This may help to create action-sound relationships that work well for the performer, and result in a performance that is more interesting to the perceiver.

6.2.3 Mapping

Mapping from controller to sound synthesis is one of the most important aspects of creating a digital musical instrument. One of the challenges in mapping is the many parameters available in sound synthesis models. A typical sound synthesis model consists of a number of technical and/or abstract parameters that are often not directly correlated to any perceptual qualities. A common solution to master such a *complex* system, i.e. consisting of many different and connected parts, is to create *presets* that define a set of parameters that work well together. These presets may then be used for further exploration of the sound synthesis model. However, the result is often that many people only use the presets, and never actually control the sound model more than changing from one preset to another. This again makes for a static and point-based type of control.

Several solutions to controlling large and unintuitive sound models have been suggested over the last years. One approach is to create mappings from a low-dimensional control space to the multidimensional sound model. Momeni and Wessel (2003) suggested doing this by assigning presets to points in a three-dimensional geometrical representation, and then control the sound models by interpolating values while moving around in the three-dimensional graphical model. Dahlstedt (2001) developed a system where an evolutionary algorithm would generate new presets by keeping some information from the previous presets in the system. In this way it was possible to generate a set of presets with similar characteristics to the presets of the "parents". Yet another approach to control complex sound models was suggested by Bevilacqua et al. (2005), who used statistical models to learn relationships between multidimensional control parameters and sound synthesis parameters. In this system the user would make an action with a controller, and select some sound synthesis parameters that the action should control. As for the models by Momeni and Wessel, and Dahlstedt, this model would also be able to interpolate between different actions, which would result in combinations of the assigned sound synthesis parameters. An interesting aspect of all these systems is that none of them require any specific knowledge about the parameters in the sound engine, as they rely on creating relationships between control output and sound synthesis input parameters.

The systems mentioned above are based on *pointwise* mappings, defining points (presets) in a multidimensional space where control parameters and sound synthesis parameters are related. van Nort and Wanderley (2006) have argued that it is not only important to define *what* to map, but also *how* it is mapped. This acknowledges the importance of the *trajectories* (or paths) taken between points in a mapping space and how these trajectories may change the "feel" of the controller. The experiments by van Nort and Wanderley showed that the complexity of an interface is linked to the dynamic quality of the mapping, and that this also influences the perceived musical expressivity of a controller.

A somewhat different approach is to create mappings between only a few output and input parameters. This was something Wanderley (2001) explored by using abstract mapping layers and *couplings* between parameters. The basis for this idea was the observation that parameters in acoustic instruments are usually coupled. For example, the breath input in a clarinet may control the timbre, loudness and vibrato of the sound at the same time. Similarly, these sound parameters may also be controlled by lip pressure. As such, most acoustic instruments seem to be based on what may be called *many-to-many* mappings, and couplings between these. Hunt (1999) showed that humans often find such coupled mappings to be more intuitive and interesting than seemingly "simple" mappings. Hunt's study therefore confirmed that what might seem to be an efficient mapping from an engineering point of view may not work so well in practice.

A very different way of looking at the problem of mapping is by using *physical modelling* techniques. First implemented by Karplus and Strong (1983), physical modelling was extended to *digital waveguide synthesis* by Smith (1992), and has later developed in several different directions. Physical modelling may be seen as an ecological approach to sound synthesis, as the models reflect the physical processes underlying sound production. As such, the control parameters of a physical model are related to physical actions and objects rather than some abstract technical parameters. This calls for creating controllers and sound models that match each other closely, and where the outputs of the controller may be mapped directly to the input parameters of the sound model. Examples of a physical model approach to musical instrument design is the CORDIS system by Cadoz et al. (1984), the *physically inspired stochastic event modeling* (PHISM) and related controllers developed by Cook (2001), and the re-creation of Russolo's *Intonarumori* by Serafin et al. (2006), to name but a few. These projects show that it is, indeed, possible to create electronic instruments where there is a tight and continuous relationship between control actions and sound output.

6.2.4 Feedback

One of the topics that seems to have received comparatively little attention in the design of digital musical instruments is that of *feedback*. All musical instruments have feedback in the form of musical sound, but electronic instruments often have much less tactile, or *haptic*, feedback than acoustic instruments. This is particularly the case when playing on a controller where the sound emitting device (the speaker) may be located far from the performer. Thus the performer will not feel any sound vibrations in the controller as would be the case from an acoustic instrument.

There are many types of feedback in an acoustic instrument, but as argued by Kvifte and Jensenius (2006) and illustrated in Figure 6.5, we may identify three layers of feedback. First, the performer will receive some direct feedback when performing an action on the controller. For example, hitting the key on a piano will result in multimodal feedback in the key. This feedback is primarily tactile and visual, but may also be auditory (e.g. when hitting a key with a finger nail). The second layer of feedback may be from the controller, e.g. feeling the vibrations in the instrument after carrying out a sound-producing action. The third layer of feedback is the sound resulting from the interaction.

In many electronic instruments the focus is often on sonic feedback, and many of the other modalities are either forgotten or only sparsely catered for. While it is possible to perform on an instrument that is mainly based on auditory feedback, I would argue that it may be difficult to develop virtuosity on such an instrument. There are some

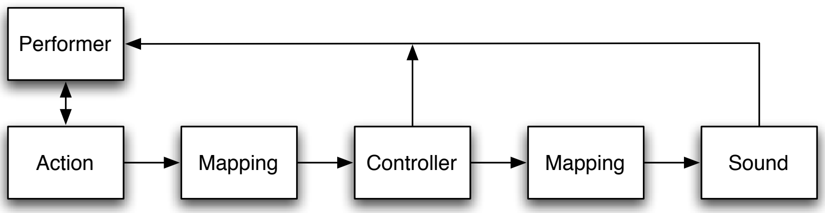


Figure 6.5: *A sketch of the chain from performer to sound in an acoustic instrument, adapted from Kvifte and Jensenius (2006). The model suggests two mapping layers and three multimodal feedback layers between action and sound.*

exceptions, though, such as the non-tactile nature of the Theremin, but in general much of the "feel" of an instrument lies in its tactility. Thus trying to develop better feedback modalities in electronic instruments seems to be important for future music technological development.

6.2.5 Simple or Complex

How easy should it be to play a musical instrument? This question has received quite some attention in the music technology community over the years. Acoustical instruments are often argued to be *complex*, in the sense that they require years of practice to master, and only a few will ever reach a level of virtuosity. However, most acoustic instruments are fairly intuitive and easy to get started on. For example, most people will be able to produce sound the first time they pick up a guitar, and many also learn to play a few chords and a scale in a rather short period of time. But to get from that stage to virtuosity is, indeed, a long journey.

In the world of electronic instruments things are somewhat different. Some electronic instruments may be so difficult to get started with that beginners never manage to produce any sound at all before they give up. This is particularly the case with computer-based instruments, which require quite some expertise to connect and install all the relevant hardware (controllers, audio interface, etc.) and software (soft-synths, samplers, etc.). Even people with degrees in music technology often struggle to get any sound at all out of such "instruments", and some controllers and synthesis models may be so difficult to use that nobody will ever be able to perform on them.

In the fairly short history of digital musical instruments, there are few instruments that have existed long enough for anyone to actually perform on them for an extended period of time. Therefore few instruments have had the chance to go through enough development cycles and be played by enough musicians to become stable and well functioning. In the most extreme cases, some performers play new prototype instruments at every performance. Obviously, this results in little time for practice, which calls for developing "simple" and intuitive controllers that can be mastered quickly. Wessel and Wright (2001) have argued that creating *simple* controllers only allows for *simple* interaction, and makes the device feel more like a toy than an instrument, and one which is not tempting for continued musical exploration.

This focus on simplicity has led to a tendency to *reduce* the complexity of controllers, but *increase* the complexity of the mapping and hence the musical output. An extreme example may be to think about a CD player as a musical instrument, on which it is possible to "play" music by pressing only one button. A CD-player allows for simple interaction, but it also leaves the user with little control over the sound. As such, the complexity of the interface is clearly linked to the complexity of the interaction and the output of the system. A not so extreme example is that of the so-called "accompaniment keyboards" which have become popular over the last decades. Such keyboards play an accompaniment (primarily chords and rhythms) following the melody played by the user. As such, the musical output is much more complex than the input actions would normally facilitate. Nevertheless, the user is somewhat in control of the musical output, but not beyond the musical structures pre-programmed into the instrument.

Some may argue that such semi-automatic instruments move into the territory of what Rowe (2001) has called *machine musicianship*. Rather than being an "instrument", the device seems more like another "musician", since the user has so little control over the musical outcome. While this may be considered problematic, Tod Machover (2004, 171) argues for such an active approach to music:

> What if we could unlock the expressive mysteries of music first, before learning the technical foundations, if we could help young people – and others – fall in love with the joys of music first, subsequently demanding deeper knowledge once they were 'hooked'?

Machover's approach in the Toy Symphony[4] and other projects carried out at the MIT media lab, has been to explore semi-automatic instruments and how they allow for *active music* exploration. Here the users, often children, may perform complex music with simple, toy-like controllers, often in collaborative performances. The idea is that the instruments should be immediately easy-to-use, yet provide for rich musical experiences. Though they do not allow for any fine control that may be developed to virtuosity, they function more as an entrance to the world of computer music.

6.2.6 From MIDI to OSC

The *Musical Instruments Digital Interface* (MIDI) format was standardised by the The MIDI Association (1983), and has been the de facto communication standard in the computer music world ever since (Jungleib, 1996). MIDI provides a simple way of mapping keyboards, controllers and pedals to musical sounds, both by defining the physical cables used to connect devices, as well as the protocol used for communication. The MIDI standard itself does not define any specific type of mappings between controller and sound, but the *General MIDI* (GM) and GS specifications by Roland define sets of timbres that

[4]http://www.toysymphony.net

synthesisers should follow. This makes it possible to play a MIDI file on any GM compliant system and get a more or less similar rendition of the music.

The simplicity and large commercial adaptation have made MIDI a standard not only for connecting controllers and sound systems, but also for storing symbolic music notation. Even with the advent of MusicXML[5] and other symbolic music notation exchange formats, MIDI files still seem to be the most popular solution for transferring notation between various types of notation software and sequencers. As such, MIDI is one of the most successful and oldest standards in the computer industry, and one which is still going strong, at least in the world of commercial music technology. The research community, on the other hand, has long pointed out MIDI's many shortcomings (Loy, 1985; Moore, 1988), including the low resolution,[6] high latency in the cables and interfaces,[7] number-based messaging,[8] and its serial nature.[9] While each of these weaknesses may not be that problematic in themselves, the combination may often lead to challenges, particularly in large setups.

Another fundamental problem with the MIDI standard, and one which may be considered its greatest weakness, is its foundation on keyboard type messaging. MIDI communication is built around "noteon" and "noteoff" messages using a tempered, twelvetone scaling system to denote the pitch values. This fits well with the discrete nature of keyboard-style controllers, but causes problems when used with other type of controllers. For example, trying to code a continuous pitch slide with MIDI requires the use of a series of "noteon", "noteoff" and "pitch bend" messages. Working with micro-intervals or complex attack forms is equally problematic. We may wonder whether these limitations inherent in the MIDI standard are the reason why there are so few electronic instruments based on continuous controllers, and so many based on keyboard and "knobs-and-sliders" paradigms.

Another example of how MIDI may have changed the way electronic instruments are designed, is the extensive use of *envelopes* in many synthesisers. Rather than giving the performer direct control over the sound, such semi-automatic envelopes, often based on *low-frequency oscillators* (LFOs), are used to change the timbre over time. This may be seen when hitting a key on any modern synthesiser, which often results in a sound that automatically evolves by itself over long periods of time. Lossius (2007, 77) argues that these pre-programmed envelopes, such as the *attack-decay-sustain-release* (ADSR) envelope, seem to be designed so that the performer will know when to release the key. Thus the instrument seem to control the performer more than the performer controls the instrument.

Throughout the years there have been several attempts to create standards to over-

[5]http://www.recordare.com/xml.html

[6]Based on 7-bit messaging, which gives a range of 0-127 integer values. Some controllers have started to use two MIDI channels for transmission, which gives a 14-bit resolution.

[7]This has been improved in devices in which the USB protocol is used to transfer MIDI messages.

[8]This is particularly problematic in large setups as it is difficult to figure out what the controller numbers and messages actually mean.

[9]In a MIDI stream, nothing actually happens at the same time. Often this is not noticeable, but it may be problematic when the polyphony increases. For example, dense chord structures may sometimes be heard as arpeggios rather than chords.

come the problems and limitations of MIDI. While MIDI still prevails in the world of commercial music technology, it has been superseded by far by *Open Sound Control* (OSC)[10] in the research community (Wright and Freed, 1997). OSC is a platform[11] and transport[12] independent protocol based on specifying messages with a URL style *name space*,[13] and is used for communication in and between hardware devices and software.

As the name implies, OSC is an open standard where few limitations are forced upon the user. While this openness has certainly been liberating compared to other standards, it has also made it difficult for OSC-enabled systems to communicate efficiently. This is because no standard namespaces have been set up, and so everyone uses it somewhat differently. The end result is that OSC-based devices and software will not be able to communicate directly without first setting up specific mappings between them. This is opposed to the limited nature of MIDI, where basically any controller can be used with any sound engine. For OSC to gain popularity, it is important to work towards common guidelines and to create uniform namespaces. This was suggested by Wright et al. (2001), and there have been discussions in the OSC community, but no consensus has been reached so far. Working towards a more coherent way of creating OSC namespaces been one of the goals of the *Gesture Description Interchange Format* (GDIF) which will be discussed in Chapter 9.

6.3 Game Controllers

Trying to overcome some of the limitations of commercial *music* controllers, I started to test how various *generic* computer controllers could be used in music performance. The market for generic computer controllers has grown rapidly in recent years, and prices have dropped swiftly. However, just like in the music technology industry, there seems to be a tendency to focus on a few main product designs among manufacturers of generic computer controllers. Standard input controllers are mainly limited to *mouses*, *keyboards* and *graphical tablets*. The variety of game controllers is somewhat larger, and includes the traditional *joysticks*, *gamepads* and *steering wheels*. More recently, it h as been interesting to see the large growth of alternative game controllers, such as the guitar controller for *Guitar Hero*,[14] the step controller in *Dance Dance Revolution*,[15] the conga drum controller for *Donkey Konga*,[16] and the wireless Nintendo Wii controller.[17]

[10]http://www.opensoundcontrol.org

[11]OSC is implemented in most popular operating systems, including OS X, Windows, and various types of Unix and Linux.

[12]While OSC is often communicated over Ethernet or WiFi, there is nothing in the standard that prevents it from being sent over USB, FireWire, or even MIDI cables.

[13]An OSC namespace may be thought of as a structure organising the way messages are communicated, for example the hierarchical message /instrument/controller/parameter. Each of the three levels in this URL-style namespace can be parsed separately.

[14]http://www.guitarherogame.com

[15]http://www.ddrgame.com

[16]http://www.donkeykonga.com

[17]http://www.wii.com

In general, I find game controllers to be cheaper, faster and sturdier than most commercial music controllers. As such, they would make for great music controllers if they could only be connected to some sound engines. Fortunately, most game controllers adhere to the *Human Interface Device* (HID) standard,[18] which make them "plug-and-play" on most systems. This also makes them easily accessible in many music programming languages, e.g. Max/MSP, Pure Data (PD) or SuperCollider, in which it is possible to map the controller values to various sound synthesis parameters.

While most game controllers use the HID protocol for communication, their messaging is not always coherent. I have tested a number of game controllers over recent years, and have found that the controller numbers and ranges are slightly different across devices and operating systems, even for similar devices from the same manufacturer. The result is that a mapping which is set up to work with one specific controller does not necessarily work with another. Such differences are not particularly difficult to correct one by one, but I have found them to be quite problematic when running workshops at which everyone is using different game controllers. Thus, making a program to automatically adjust for such differences seemed important. Thus, planning a workshop for children, I decided to create an application that could facilitate using game controllers with MIDI based sound engines.

6.3.1 MultiControl

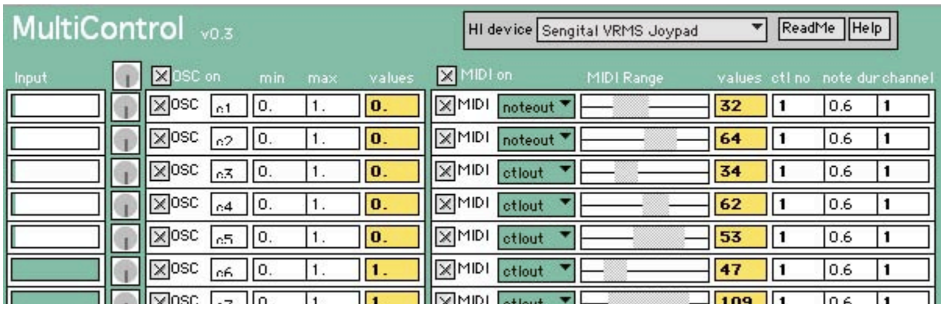
MultiControl is a standalone program, built with Max, that facilitates mapping values from any HID-compliant game controller to various musical applications. This makes MultiControl somewhat similar to *Junxion* from Steim,[19] but there are some important differences between them. MultiControl is cross-platform (Mac OS X and Windows XP), it can output both OSC and MIDI messages, and it facilitates low-pass filtering the controller values. While originally made for children, it has later been developed to be used in a music technology course I have been teaching to music students.

Figure 6.6 shows a screenshot of the main interface of MultiControl, and a short demonstration can be seen in Video 6.1. The user starts by choosing the desired input device in the menu in the top right corner. Then, when using the device, the different controller numbers and ranges will be recognised automatically. It is possible to output both OSC and MIDI messages and to choose the output range used for communication. For OSC messages it is possible to manually type in the name which will be used in the namespace, as well as setting up the network address which will be used for communication. The MIDI part of the interface allows for selecting the type of MIDI message to send out (noteon, control out, or bend out) and the associated controller numbers, durations and channels. Sending MIDI internally on the computer makes it possible to use MultiControl together with a software synthesiser running on the same computer.[20] It is also possible to choose to output to any MIDI device connected to the system.

[18]http://www.usb.org/developers/hidpage/

[19]http://www.steim.org/steim/junxion.html

[20]On OS X this can be done by using internal MIDI routing within Max/MSP, while on Windows it is necessary to install third party software, for example MIDI Yoke (http://www.midiox.com/myoke.htm).


MultiControl v0.3

HI device: Sengital VRMS Joypad — ReadMe — Help

Input	OSC on		min	max	values	MIDI on		MIDI Range	values	ctl no	note dur	channel
	☒ OSC	c1	0.	1.	0.	☒ MIDI	noteout		32	1	0.6	1
	☒ OSC	c2	0.	1.	0.	☒ MIDI	noteout		64	1	0.6	1
	☒ OSC	c3	0.	1.	0.	☒ MIDI	ctlout		34	1	0.6	1
	☒ OSC	c4	0.	1.	0.	☒ MIDI	ctlout		62	1	0.6	1
	☒ OSC	c5	0.	1.	0.	☒ MIDI	ctlout		53	1	0.6	1
	☒ OSC	c6	0.	1.	1.	☒ MIDI	ctlout		47	1	0.6	1
	☒ OSC	c7	0.	1.	1.	☒ MIDI	ctlout		109	1	0.6	1

Figure 6.6: *Screenshot from a part of the interface of MultiControl. The user may choose which device to use in the top right corner, and the controller numbers will automatically be detected and assigned to a separate "line" in the interface. The user can select to output OSC and/or MIDI messages, and choose the types and ranges used for these output values.*

I will not go into detail about the implementation of MultiControl, but will point out a couple of the main features. The MultiControl patch is built around the **hi** object in Max. Unfortunately, **hi** does not return a list of the available controller numbers, so I had to create a patch for dynamically recognising new controller numbers as they appear. These are assigned to one of the 16 "lines" in the main interface of the software, each of which is loaded in a **bpatcher** object. As can be seen in the implementation of a line in Figure 6.7, each line is self-contained in terms of processing. All scaling and conversion of numbers is done with a patch called **arj.scale**, which is based on the objects **scale**, **peak** and **trough**, as shown in Figure 6.8.

MultiControl has been of great use in both performance and educational contexts. An example of its educational use may be seen in Figure 6.9, which is a picture from our booth in the city centre of Oslo during the Norwegian Research Days in 2004. At this, and several other similar public events, thousands of people have used MultiControl to map values from game controllers to various musical applications. One of my main observations from such events is that children are generally very familiar with game controllers, and are eager to experiment with different types of mappings. Adults, on the other hand, are usually a bit more reserved in the beginning, but they too find it interesting to use game controllers in a musical context once they get started.

I have also used MultiControl in a number of concerts, controlling everything from sound synthesisers to video animation and sound *spatialisation*. Spatialisation is here used to denote the placement of sounds in space. I find joysticks to be particularly suitable to control sound spatialisation, since rotation of the joystick may be directly mapped to the placement of sounds in space. Then I have more problems when it comes to find mappings that work well with regular sound synthesisers. A typical joystick has three continuous axes (left, right and rotation) on the stick, and a set of discontinuous controllers (buttons). When controlling MIDI-based software synthesisers that expect a discontinuous excitation, I often use the fire button on the joystick to play the tone, and then

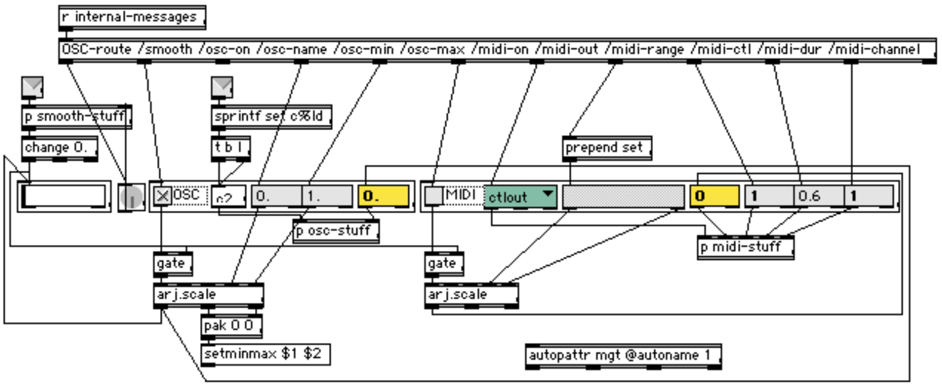


Figure 6.7: *The implementation of one of the "lines" in MultiControl. Values from the* ***hi*** *object are coming in from the left inlet. The right inlet is used for setting the instance number of the line, which is used for OSC output.*

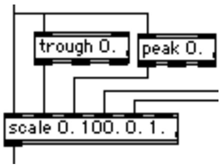


Figure 6.8: *The* ***arj.scale*** *patch is based on recognising maximum and minimum values using the* ***trough*** *and* ***peak*** *objects. The outputs of these objects are connected to a* ***scale*** *object which will automatically scale the numbers to the given range.*

Figure 6.9: *Pictures of children testing MultiControl at the Musical Gestures booth in the city centre of Oslo during the Norwegian Research Days 2004. In this setup the outputs of a Saitek Cyborg 3D Gold joystick were mapped to various sound parameters in Native Instruments' Absynth 2 software.*

use the three axes to control continuous parameters of the sound. I find such a mapping to be fairly intuitive, and it is also quite similar to how a joystick is often used in games.

As presented in the discussion of design principles of music controllers, it is important that the capacities of the controller match the control parameters of the sound model. Thus, the most basic question we should ask when trying to create a mapping is whether the controller is suitable for the chosen sound model. If not, either the controller or the sound model should be changed. One solution would be to develop custom made controllers to match the sound model. However, building hardware is either expensive, if you rely on easy-to-use commercial solutions, or difficult, if you try the traditional engineering approach. Inspired by the way game controllers could be used in musical applications, I set out to look for other cheap and easy solutions to create music controllers.

6.4 Cheapstick

The *Cheapstick* project started out as an attempt to create a solution for a music controller that could be used in music technology courses for music students. Therefore the controller had to be easy to build, and it should be so cheap that most people could afford it. As mentioned earlier, the core elements of a controller are the *sensors* which sense the input actions, and the *sensor interface* which converts the electric signal in the sensor into a digital signal in the computer. The challenge was,[21] and largely still is, that both sensors and sensor interfaces are fairly expensive, and/or often difficult to work with. I teamed up with Mark Marshall and Rodolphe Koehly at McGill University and we set out to create a complete and easy-to-build music controller for "€10". The end result was the prototype *Cheapstick* controller, which we presented in Jensenius et al. (2006).

The cheap solution to our problem would have been to take an electrical engineering approach, using cheap electronics components and microcontrollers. However, this was clearly far beyond the scope of what an average music student would be capable of accomplishing during a one semester course in music technology. The "easy" solution was to use one of the commercial "plug-and-play" sensor sets that have emerged for musical and artistic purposes in recent years. This would be easy enough, but would cost everything from 10 to a 100 times more than our budget. So we set out to find some other solutions.

6.4.1 Comparing Sensor Interfaces

Looking for a sensor interface that could suit our needs, we found it important that it should be so small that it could be integrated with a controller or instrument. It should also be easy to build, and easy-to-use, preferably not requiring any extra hardware or software to work with common platforms and programs.

When it comes to commercially available sensor interfaces for musical applications, there has been a growing number of solutions made available over recent years, all of

[21]This discussion refers back to early 2005.

which differ in size, number of inputs and outputs, speed, resolution, transport type, communication protocol, and price. Some of these are shown in Figure 6.10, and their features are summarised in Table 6.1.[22]

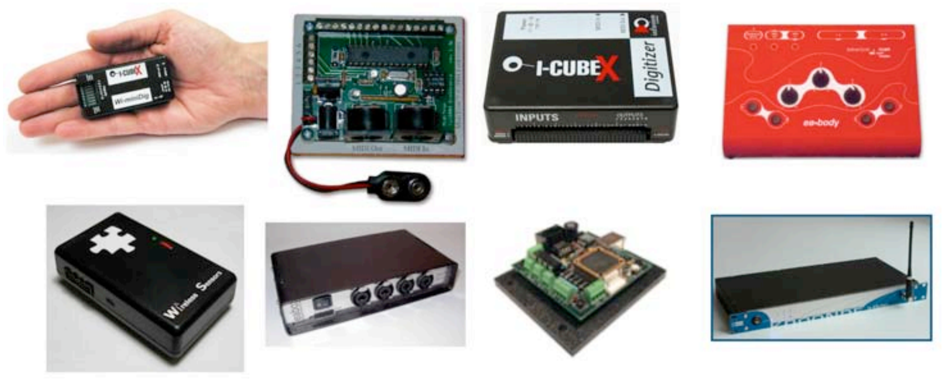


Figure 6.10: *Various commercial sensor interfaces for musical or artistic purposes. From top left: the* Wi-miniDig *from Infusion Systems, the* MIDITron *by Eric Singer, the* Digitizer *from Infusion Systems,* eo-body, WiseBox *from IRCAM,* Teabox *from Electrotap, a* Teleo *module from Making Things, and a* Kroonde *from La Kitchen. All of these interfaces differ widely in size, number of inputs and outputs, speed, resolution, transport type, communication protocol and price.*

As may be seen from the overview in Table 6.1, none of these interfaces came close to our budget. Also, all the cheapest interfaces used MIDI for connection and communication, something we tried to avoid considering MIDI's many limitations. On a positive note, MIDI-based sensor interfaces may be directly connected to a hardware sound device, for example a synthesiser or sampler. However, nowadays most people mainly seem to use computer-based sound engines, so a MIDI based sensor interface would also require a separate MIDI interface to connect to the computer. This would again increase both the price and the potential problems of the setup.

Several of the USB-based sensor interfaces in Table 6.1 are comparably smaller and cheaper than the other interfaces, and might therefore be a good option to use in a cheap controller. Unfortunately, many of these USB based sensor interfaces require proprietary drivers to work with the operating system, and also special objects to be accessible in various music programming languages. As such, they are not particularly flexible and "future-proof", since they often rely on the existence of a particular manufacturer. There are several examples of excellent interfaces and controllers that are useless because the manufacturer went out of business, and no new software updates are available. Another problem with USB-based interfaces is that many of them seem to be unreliable when used for longer periods of time. This is regularly discussed on the Max/MSP mailing

[22] I have made a more complete list of sensor interfaces at http://www.sensorwiki.org.

Table 6.1: *A comparison of a number of sensor interfaces for musical and artistic purposes available in 2006. The prices are approximate conversions based on the prices from the web pages of the manufacturers. Inputs refer to the number of analogue and digital inputs. The speed denotes the sampling rate of the interface for all inputs. Resolution refers to the range used for the sampled values. Protocol refers to the transport and communication type used in the interface.*

Product	Manufacturer	Price	Inputs	Speed	Resolution	Protocol
8/8/8	Phidgets	€80	8/8	-	10 bit	USB
Pocket El.	Doepfer	€80	16	-	7 bit	MIDI
MIDItron	Eroktronix	€125	8	-	7/10 bit	MIDI
Teleo	Making Things	€130	4	-	10 bit	USB
miniDig	Infusionsystems	€330	8	100 Hz	7/14 bit	MIDI
Teabox	Electrotap	€350	8	4000 Hz	12 bit	SPDIF
GluiOn	Glui	€445	16	1000 Hz	16 bit	OSC
Eobody	IRCAM	€480	16	-	10 bit	MIDI
Wi-miniDig	Infusionsystems	€500	8	100 Hz	7/14 bit	MIDI
Digitizer	Infusionsystems	€580	32	24-244 Hz	7/14 bit	MIDI
Wise Box	IRCAM	€950	16	200 Hz	16 bit	OSC
Kroonde	La Kitchen	€1200	16	200 Hz	10 bit	OSC

list,[23] and also by Allison and Place (2005) who report on the instability of USB based interfaces in long-running installations.

For Allison and Place, the solution to the many problems experienced with various sensor interfaces was to design and construct their own interface. Their *SensorBox* was based on *multiplexing*[24] the sensor values as a digital audio channel (Allison and Place, 2003), and has later been commercially released as *Teabox*[25] (Allison and Place, 2005). This currently seems to be one of the fastest and most reliable sensor interfaces available for musical applications, but unfortunately it requires an S/PDIF optical digital audio connector in the computer. S/PDIF now comes standard with some computers, but otherwise it requires an extra audio interface to work, which boosts the prize and complexity of the setup. These drawbacks, combined with the rather high price and large size of the box, made that Teabox was not an option for our cheap music controller.

Sensor interfaces based on Ethernet or WiFi communication (e.g. *Ethersense*, *Kroonde*, *Toaster*, *GluiOn*), seem to be more stable and flexible than USB and MIDI-based interfaces. Sporting OSC communication, such devices typically allow for higher resolution and sampling rates, and long-distance communication over standard, high-speed networking connections. But the problem with these networking-based sensor interfaces is

[23] http://www.cycling74.com/forums/

[24] Multiplexing is a method used in electronics and computing to combine multiple signal streams into a single stream. This allows for transmitting only one stream of data, which saves bandwidth in the communication. The multiplexed stream may be *de-multiplexed* in the receiver to retrieve the original streams.

[25] http://www.electrotap.com/teabox

that they are generally too large and far too expensive for our budget.

Summing up, none of the commercial products at the time met with the requirements for a cheap and small sensor interface for our music controller. This was when we thought about the possibility of "modifying" cheap commercial devices.

6.4.2 Game Controllers as Sensor Interfaces

As discussed above, game controllers are generally cheap, stable, small, and easily available. Furthermore, they do not usually require any drivers or software installs, they draw their power from the USB port, and they work with my MultiControl software. As such, they seemed to meet most of the requirements we wanted of a sensor interface. The good thing is that a game controller, like most other electronics devices, is actually a small sensor interface in its own right, since its purpose is to translate gaming actions into digital signals in the computer. In fact, it is enough to remove the plastic cover and solder on a few cables to turn a game controller into a generic sensor interface.[26]

Of the many different types of game controllers available, I find *gamepads* to be the most interesting to modify. They are generally cheaper and smaller than other types of controllers, such as joysticks, flight-sticks and steering wheels, and usually have the largest number of continuous and discrete controllers. This means that they also have the largest number of sensor inputs, which is what we were interested in. Taking a typical gamepad apart reveals a motherboard with 4 analogue and 12 digital inputs (see Figure 6.11). These inputs comply to a 0-5 volt sensor input range, which is the standard range used for many electronic devices. This means that it can be used directly with a large number of commercially available sensors.

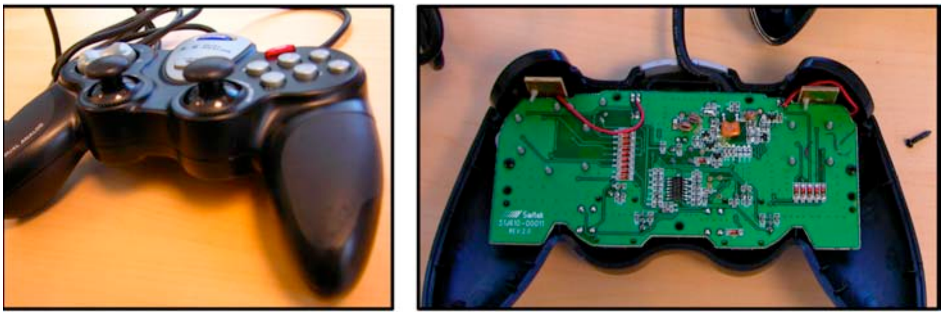


Figure 6.11: *Removing the plastic cover of a standard gamepad (left) reveals a motherboard which may be easily transferred into a generic sensor interface (right).*

The design of gamepads differ slightly between manufacturers, but they usually all have more or less the same construction. The circuitry on the motherboards also are often clearly marked, which makes it fairly easy to see where the different connectors are.

[26]Since we carried out and published this project, the details of modifying a game controller has also been presented in more detail by Collins (2006b, 195-200).

Otherwise it is not particularly difficult to use a multimeter to check which connectors carry the electric signal, the power (+5 volt) and the grounding. It is possible to leave the circuitry just as is, and solder on the necessary cables, but I usually prefer to remove all the redundant cables and parts of the game controller. In some gamepads this can done by cutting off a few cables, while in others it might be necessary to de-solder a few components. Figure 6.12 shows a picture of how cables have been soldered onto the relevant points on the motherboard. Here we also soldered 3-pin connectors to the cables, so that different sensors could more easily be plugged in to test the interface.

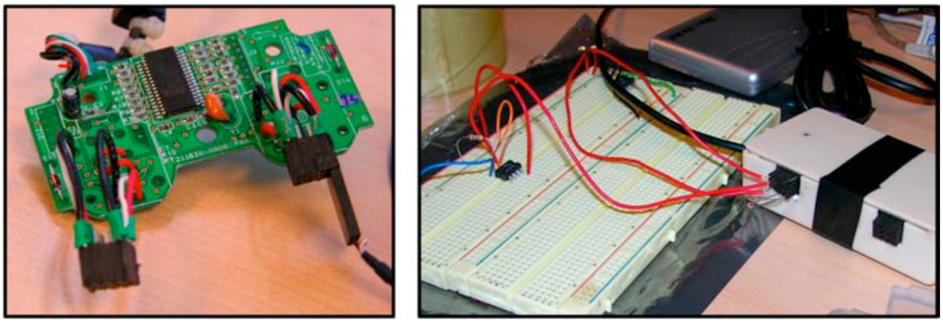


Figure 6.12: *The motherboard of a game controller with cables and three-pin connectors soldered on (left), and put into a small plastic box and used to test various types of sensors (right).*

The result of this process is a generic sensor interface with 4 analogue and 12 digital inputs, 8-bit resolution and 100 Hz sampling rate. Modifying a "rumblepad", a gamepad with built-in motors, allows for getting a couple of analogue outputs to control motors, and it is also possible to modify a wireless gamepad to get a wireless sensor interface. Unfortunately, wireless gamepads often turn off after a few minutes of inactivity, which makes them less ideal as music controllers. Thus, in general I have found USB-based gamepads to be the most stable and reliable devices to turn into generic sensor interfaces.

6.4.3 Homemade Sensors

The second part of the challenge to make a really cheap music controller was that of finding sensors that would suit our budget. As we had already spent a little too much on the gamepad sensor interface, we had to find a very cheap sensor solution. Again, most of the commercially available sensors sold for musical and artistic applications were priced at 10-100 times more than we would pay. These sensors are often *conditioned*[27] so that they can be connected directly to one of the sensor interfaces they are sold with. Buying unconditioned sensors directly from the manufacturer would make the price go down, but not enough for our budget. Another problem with commercial sensors, as discussed by Koehly et al. (2006), is that they usually come in standard sizes and with a fixed

[27] Adding a small circuitry with the necessary resistors or amplifiers to make the sensor adhere to the standard 0-5 volt range.

resistance. These standards restrict the design of the controller, since the design often has to follow the shape, material and resistance of the sensors.

Another problem with commercial sensors is the lack of tactile feel and response. For example, most commercial pressure sensors[28] are usually rather thin and have a standard resistance. Also, they are often either too small or too large to be interesting for musical applications, and trying to change their size may shortcut the sensor. The solution is often to use a small sensor, and pad it with extra material in the controller. This may be an efficient way of increasing the tactility of the controller, since the padding material is important for the "feel" of the instrument. However, such padding will reduce the direct control of the sensor, since the extra material will effectively dampen the applied force. A solution to overcome this may be to add extra sensors, but this will again add to the cost and complexity of the controller.

Rather than adding lots of sensors to increase the response of a controller, one solution is to use the padding material itself as a sensor. This was demonstrated by Paradiso and Hu (1997) in the design of a "shoe controller" where the sole of the shoe served as a sensor. This, and other similar projects, was what inspired my colleague Rodolphe Koehly to create various types of pressure and bend sensors from conductive ink, adhesive, rubber, tape, elastics and paper (see Koehly et al. (2006) for details). He also managed to make position sensors by using the conductive properties of regular VHS videotape based on a design presented by Escobedo (2005). All of Rodoplhe's sensors are made from cheap components that are readily available in many different sizes in most craft stores. This makes for easily creating sensing surfaces that may be customised to the size and shape of the controller.

Figure 6.13 shows some pictures of position and pressure sensors made for the Cheapstick project. As may be expected, the conductive qualities of these materials vary considerably, so further research is needed to test different types of materials and improve the consistency and reliability of the response of the sensor. Also, since the range and conductivity of these materials are unknown and sometimes change over time, they require improved electronic circuits for conditioning the signals to match the 0-5 volt range used in most sensor interfaces. Nevertheless, these early prototypes show that it is possible to create homemade sensors that have the same, or even better, response and tactility than commercial sensors, and at a fraction of the price.

6.4.4 The CheapStick Prototype

The final *CheapStick* prototype consisted of the modified game controller sensor interface, three pressure sensors made from conductive paper, and a position sensor made from videotape (see Figure 6.14). The interface and sensors were mounted on a leftover part of an IKEA bookshelf, and the total cost of the complete controller was around €25. This was a little more than our budget, but still much cheaper than what it would have cost to build a similar interface with commercially available interfaces and sensors. The

[28] *Pressure* is here used to denote a force applied to a surface, such as pressing your finger towards the table. One type of sensor which measures force is called a *force sensing resistor* (FSR)

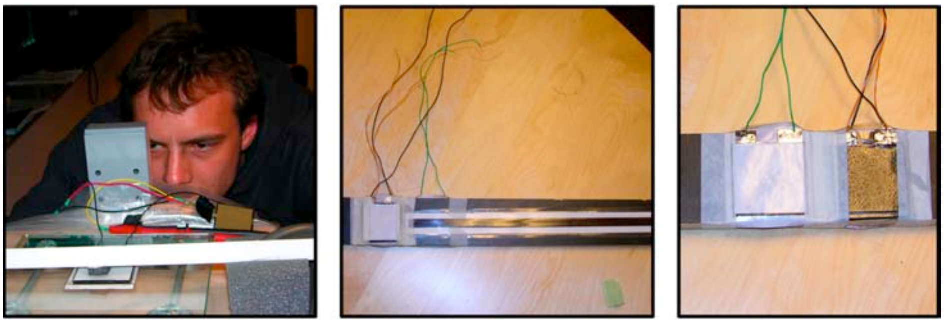


Figure 6.13: *Rodolphe Koehly measuring the resistance in his home-made paper FSRs (left). The position sensors made from video tape and a paper FSR were mounted on a leftover part of an IKEA shelf (middle). The two paper FSRs were mounted on the rear side (right).*

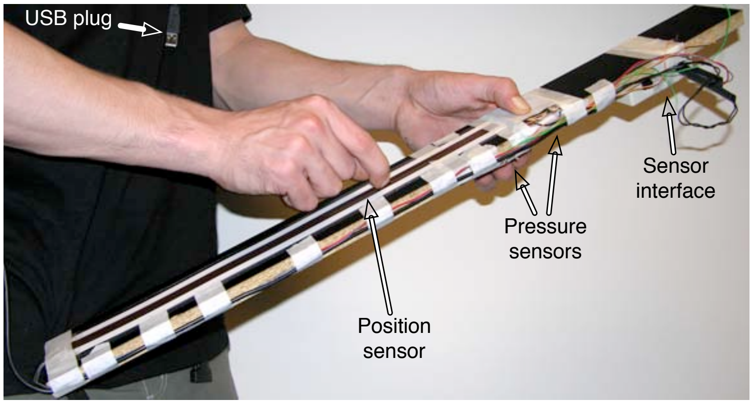


Figure 6.14: *The CheapStick is a self-contained music controller built for around €25.*

end result was a simple, handheld controller which could be plugged directly into the computer with the USB plug, and used with the MultiControl software to control any OSC or MIDI-based sound module. The prototype had an excellent tactile feel, and the sensor response was comparable to commercial sensors, although it detoriated quickly.

A problem with the gamepad sensor interface of CheapStick was that it only returned a 7-bit (0–127) range after it was modified, while it used an 8-bit (0–255) range as a gamepad. This has happened with some, but not all, gamepads I have modified, and seems to be the result of some kind of multiplexing between the inputs in the controller. We tried various kinds of conditioning on the sensor signals, and shortcutting parts of the circuitry on the board, but did not manage to improve this loss of resolution. It turned out that the sensors themselves did not have such a large resolution, so this resolution loss in the interface was not so critical to the overall performance of the device.

6.4.5 Discussion

Since we made the CheapStick, a number of cheap, research-based sensor interface solutions have been presented. These include McGill's *AVR HID*[29] based on the ATMEL ATmega16 microcontroller (Marshall, 2005), and the *CREATE USB interface*[30] based on a PIC18F4550 microcontroller from Microchip Technologies (Overholt, 2006). Both of these microcontrollers are popular in engineering, but have previously required quite some electronics and programming experience to work with. The simplified McGill and CREATE solutions provide a ready-made circuit for interfacing the microcontroller, but I would still argue that they are too difficult to work with for the average music student taking a music technology course.

A somewhat easier microcontroller solution can be found in the *Arduino* board,[31] an open-source, physical computing platform based on simplified *Wiring*[32] programming language. Currently, the Arduino is probably the cheapest, easiest and most flexible solution for people wanting to create their own electronics. However, for music students taking a music technology course I still find that modifying a game controller and using MultiControl for controlling sound engines is the cheapest and easiest solution.

When it came to the homemade sensors, we were generally pleased with their feel, touch and response. The paper sensors were much more tactile than comparable commercial pressure sensors, since they were anything from 3-10 millimetres thick. Another positive thing about the homemade sensors was that they could be fit to the size and shape of the controller. One problem, however, was that they were not particularly durable, and lost much of their resistance after a fairly short period of use. This was also the case with the position sensor made from videotape, which was very precise in the beginning, but which also lost some of its resistance over time. As such, the homemade sensors showed potential, but their durability and lifetime are critical factors that need to be improved in future research.

6.5 Music Balls

Besides creating *cheap* controllers, I have also been interested in exploring the concept of *simplicity* in the design of music controllers. Although rarely found in nature, the *sphere* is arguably one of the simplest and most uniform of geometrical constructs. I therefore wanted to create a music controller from a ball, to test what types of musical interaction it would allow for. This led to the development of a series of "music balls", a collection of controllers and instruments with one thing in common: they are ball-shaped and they may generate and control sound. Figure 6.15 shows some pictures from the development and use of different music balls, and the following sections will discuss three different design types: *acoustic*, *electronic* and *electroacoustic* music balls.

[29]http://www.music.mcgill.ca/∼marshall/projects/avr-hid/
[30]http://www.create.ucsb.edu/ dano/CUI/
[31]http://www.arduino.cc
[32]http://wiring.org.co

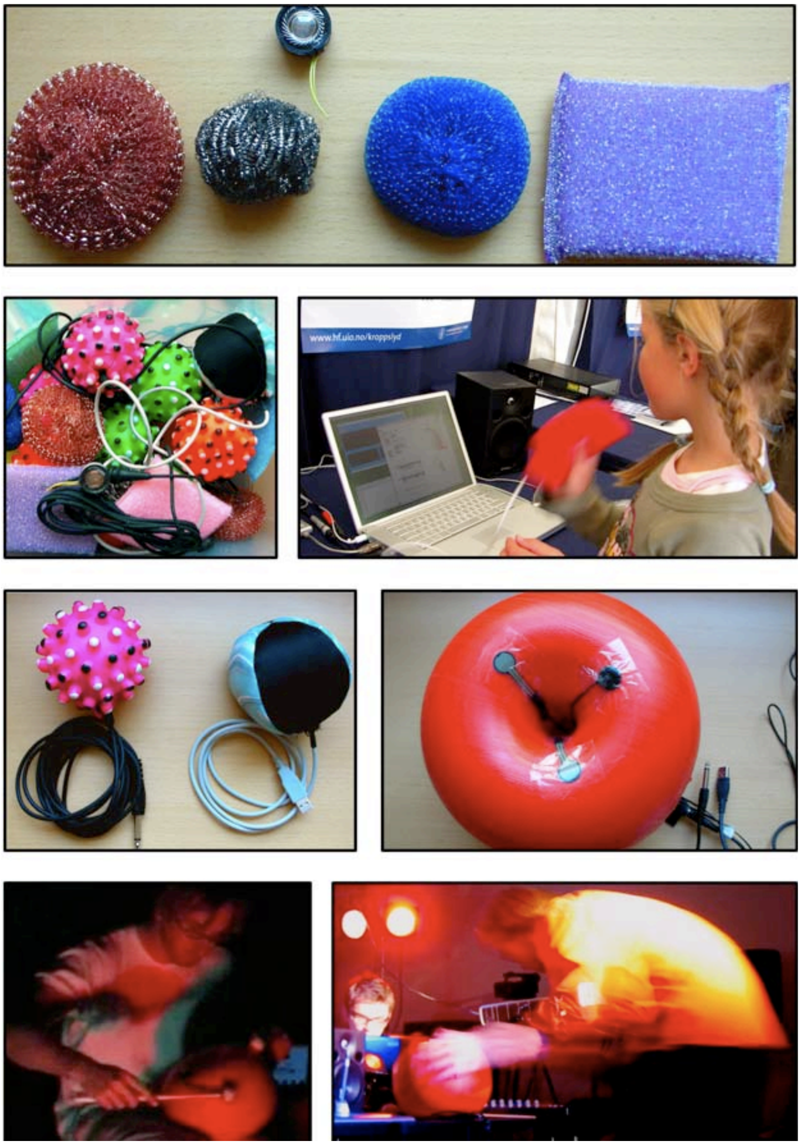


Figure 6.15: *Some pictures from the development and use of different music balls, including balls based on various materials and dynamic microphone elements (picture 1 and 2 from the top left corner), USB accelerometers (picture 3 and 4) and an electroacoustic ball built from a buoy with contact microphone and pressure sensors (picture 5-7). The balls have been used in various installations and in performances (picture 3, 6, 7).*

6.5.1 Acoustic Music Balls

Besides kicking or throwing, a ball typically affords squeezing or shaking. Thus, I started out by making a series of acoustic music balls based on various types of materials, with the intention of creating sound when squeezed. Cheap consumer shops are great for exploring various types of sound-producing materials, everything from kitchen utensils (plastic sponges and steel wool) to various synthetic materials. Many such shops also have lots of different types of ball-shaped toys, both for children and for animals. I am particularly fond of some hand-sized dog toy balls, since they are solid and easy to work with. Stuffing various types of materials (paper, peas, steel wool, synthetic fibres, etc.) inside such balls allow for lots of interesting sounds. While some materials sound nice, they may not be particularly durable over long periods of time. Newspaper sheets, for example, usually provide a nice and crispy sound, but tend to contract to a small ball inside the balls. Then many synthetic materials work better, for example plastic sponges, because they are elastic and expand quickly after being squeezed.

One of the problems with the sounds coming from such acoustic music balls is that they are not particularly loud. It is therefore necessary to amplify the sounds to use them in a musical context. This may be done by using a condenser microphone pointed at the ball from the outside, but a better sounding option is often to place one or more microphones inside the ball. Here there are several possible solutions. Sometimes a *contact microphone* placed in the middle of the sounding material inside the ball works well, but usually only when the microphone element is in close contact with the material. In general, I have had more success with the elements from *dynamic microphones*. The easiest solution is to buy cheap ($1) consumer "karaoke" dynamic microphones and remove all the plastic surrounding the microphone element. This leaves the element, a cable, and a jack connector (1/4 inch) which can be plugged directly into an amplifier or audio interface for further processing.

In general, such elements from dynamic microphones are more flexible in picking up sound than contact microphones, and they are able to amplify sounds from newspaper and other materials that do not produce particularly loud sounds. However, when it comes to different materials, my personal favourite is some balls made from different types of steel wool and dynamic microphones. These balls produce very clear and crispy sounds, and are fun to play with even without applying any digital audio effects. As the excerpt from a performance with an acoustic music ball in Video 6.2 shows, they also work well with various types of granulation and delay effects.

6.5.2 Electronic Music Balls

Acoustic music balls have the advantage of being a full instrument, since they are based on natural action-sound couplings. But is it possible to create electronic controllers that allow for the same type of control over sound? To test this, I created a series of electronic music balls. The main idea here was to explore various kinds of sensors that would fit inside the balls, such as flex sensors and pressure sensors. I find flex and bend sensors to be more "natural" than "technical" sensors, such as buttons, sliders and potentiometers, since they may sense bending and squeezing. However, one of the greatest problems with such sensors is that they are often small in size, so many of them are needed to cover a controller. Even for a small music ball, I needed 8 bend sensors positioned around the surface to get data from all sides of the ball. This also required an 8 channel sensor interface for each ball, which caused both the price and time of development to increase rapidly. Receiving and processing data from 8 separate sensors also calls for quite a complex preprocessing and mapping process in software. It could be possible to use fewer sensors in the ball, but this would also reduce the controller's sensitivity.

Shaking is another interaction possibility of a ball, and this calls for using an accelerometer to measure the movement of the ball. Dependent on the type of accelerometer used, the values returned are typically correlated to various types of manipulation of the device, including vibration, motion and inclination. All of these are interesting for musical applications, as they may be easily mapped onto various parameters in sound engines. Accelerometers can be bought separately and connected to a generic sensor interface, but currently my favourite is the USB accelerometers available from Phidgets.[33] These accelerometers come with all the necessary electronics embedded on one small chip, and with a USB cable which can be directly connected to a computer. Unfortunately, they require the installation of a proprietary driver to make them work with the operating system, and they also need a special object for Max. Nevertheless, the fully integrated hardware solution and possibility to run multiple accelerometers on one computer, far outweigh the software restrictions in my opinion, and greatly simplifies adding an accelerometer to a music ball, or any other device.

When it comes to creating mappings from a music ball with bend sensors and an accelerometer, I prefer to use movement data gathered from the accelerometer to control excitation and use the squeezing of the ball for modulating the sounds. While any type of sound model may be used, I particularly like *metashaker*, a *physically informed sonic model* (PHISM) (Cook, 1997, 2001), implemented in the Synthesis Tool Kit (STK) (Cook and Scavone, 1999; Scavone and Cook, 2005) and available as **metashaker~** in PeRColate[34] for Max. Using such a model, it is possible to directly control the energy of the sound with values from the accelerometer, and control the other parameters from the other sensors.

6.5.3 Electroacoustic Music Balls

Even though some of the electronic music balls are quite responsive and feel intuitive to play, I have not been able to make them as expressive and intuitive as the acoustic balls. Since the acoustic balls are based on natural action-sound couplings, it is difficult to make an artificial action-sound relationship which seems more direct. On the other hand, electronic music balls provide for much more flexibility and sonic exploration than the acoustic balls. Trying to get the best of both worlds, I started combining the two approaches, making *electroacoustic* music balls.

[33] http://www.phidgets.com

[34] http://www.music.columbia.edu/PeRColate/

By adding various sensors to an acoustic music ball it is possible to use the acoustic sound as the basis for sound manipulation controlled by the other sensors. This worked well, but after a while I discovered that it was also interesting to combine this type of natural sound-production with artificial sound-production. While acoustic sounds would be created by squeezing the ball, I added the possibility of creating artificial sounds by shaking the ball (using data from the accelerometer). At first, I expected this to be perceptually confusing, but have rather found it to be creatively interesting. One reason why having two different sound-producing action possibilities works, is probably because the two different action types involved, i.e. shaking and squeezing, are so different. Thus, the actions and sounds produced by the controller feel natural, even though they are based on an electronic sound engine.

Some of the electroacoustic music balls I am particularly satisfied with are built from various types of buoys and boat fenders, as seen in Figure 6.15. Not only are they extremely durable, which has proven to be important when making controllers for children and installations, but many of them also have some great acoustic properties. Particularly some of the hardshell buoys sound very nice when played with various types of percussive actions. Adding a few microphones, an accelerometer and some pressure sensors gives a flexible and durable musical instrument with both acoustic sound and electronic control. When it comes to mappings, I have found it to be important to use effects that support the percussive character of the natural sounds from the ball, some examples which be seen in Videos 6.3 and Video 6.4.

Summing up, I find that these electroacoustic music balls are the most interesting from a musical point of view. The combination of a direct acoustic response and related tactile feedback, combined with the electronic possibilities from the sensors, allow for the best of two worlds. This I have also seen in various installations and concerts, where the electroacoustic music balls are the ones that people seem to find the most interesting. This is probably because they have some natural properties that people can easily recognise and follow, while still providing for new and creatively interesting sounds and interactions.

6.6 Music Troll

The *Music Troll* project[35] was a natural extension of the exploration with the music balls presented in the previous section. Having a number of different music balls, a typical performance or installation rig required a fair amount of equipment, including multiple computers, sensor interfaces, audio interfaces, amplifiers, speakers, etc. What had started as a project trying to create simple and intuitive controllers, had ended up in a complex performance and installation rig. This called for simplification of the rig by visually removing all the electronics. The idea was to leave only the music balls, so that the user could focus on the musical interaction and forget about everything else.

[35] A *troll* is a Scandinavian fairy tale creature, often many-headed. This name was chosen because of the many-headed design of the musical controller.

The Music Troll project was a joint project with master students Einar Sneve Martinussen (interaction design) and Arve Voldsund (music technology). We decided to build all the electronics into a box, including computer, sensor interfaces, audio interface, amplifier and speakers. The idea was that only the music balls would be visually and physically present outside the box, and also that the box itself could be more easily placed in a public space without worrying about electronics and cables lying around.

Figure 6.16 shows some pictures from the design and construction phase of the Music Troll. The box was built out of plywood, and was big enough for all the necessary electronics. Four car speakers were mounted to the walls of the box so that it could play sound in all directions. One of the sides of the box served as a door to access the interior, and could also be locked so that the box could stand without supervision in a public space. Since the box would also serve as the base for the arms hanging out with the balls, it was built heavy to secure stability.

The Music Troll was built for public installations and not for performance, so we decided that the interaction design should be simple. While I think that a controller intended for performance needs to have some level of complexity to be interesting enough to perform on, a controller for installation can be much more sparsely designed. I have found that most people only spend a few seconds evaluating an installation before they move on, so it is important to simplify the interface to catch people's attention. We therefore decided that each of the "heads" (the balls) of the Music Troll should have only a few control possibilities each. This would limit the interaction possibilities, which we hoped would lead to more intuitive interaction.

As shown in Figure 6.17, the four heads were designed with a separate "identity", each having different shapes, types of sensing, and interaction and sound design:

- *Head 1:* A large spherical ball with an acoustic sensor built from steel wool surrounding a dynamic microphone element. Sensing was done by measuring the amplitude of the sound signal and this was used to drive the speed and velocity of a sampler playing the sound of a voice backwards at a slow pace. This resulted in a deep voice sounding when someone squeezed the ball, and could be thought of as the troll "speaking".

- *Head 2*: A cone-shaped ball with "antennae" hanging out of the head. This was an acoustic ball based on steel wool surrounding a dynamic microphone element placed inside a plastic container. Some plastic strips were attached to the plastic container, and hidden inside the "antennae" hanging out of the head. The head could be played by squeezing, or dragging the "antennae". The sound from the microphone element was amplified and compressed considerably, and the sonic result was somewhat squeeky and metallic sounding.

- *Head 3*: A small spherical ball hanging from a more flexible arm so that it could be thrown around. The sensing was done with a two-dimensional USB accelerometer from Phidgets. The motion calculated from the accelerometer was used to control the energy in the **metashaker**$\sim$ PHISM model.

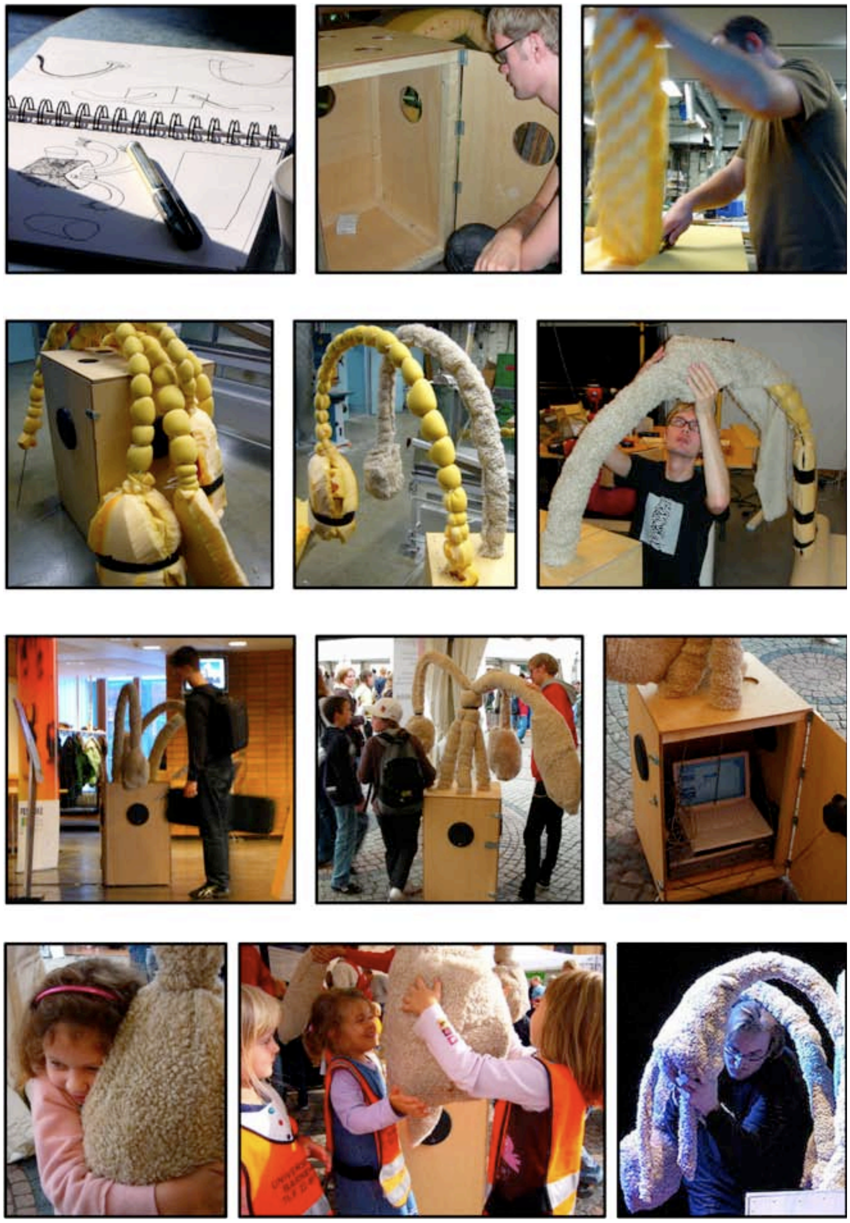


Figure 6.16: *Pictures from the design phase and construction of the Music Troll (pictures 1-6 from top left corner), its use at the Norwegian Academy of Music (7), and at the Norwegian Research Days in the city centre of Oslo (8-11) and during a performance in Oslo Concert Hall (12).*

- *Head 4*: A long flat head intended for bending. Sensing was done with a long bend sensor connected to a Phidgets 8/8/8 sensor interface. This was used to control the amplitude and centre frequency of a *peaknotch* filter (implemented with **biquad∼** in Max) applied to a *white noise* signal. The result was a sweeping sound when the user bent the head.

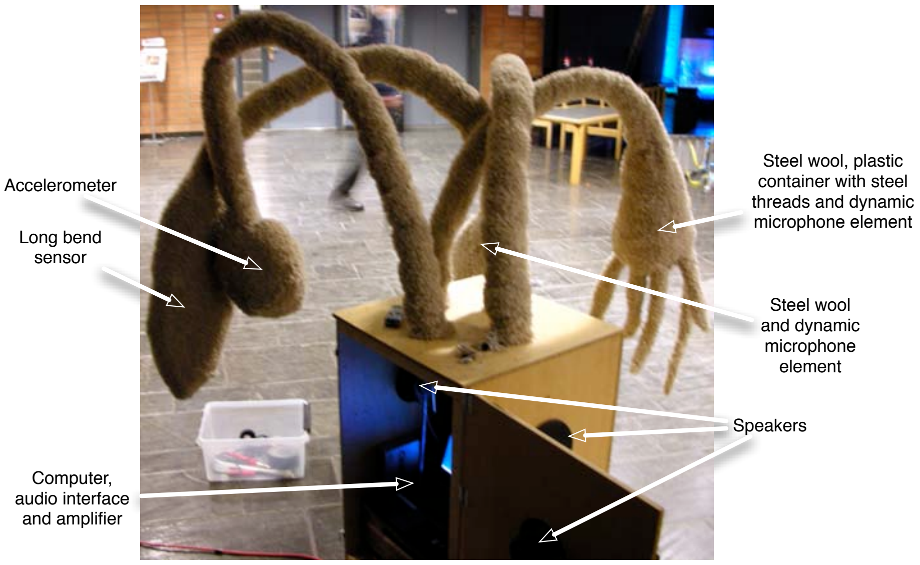

Accelerometer
Long bend sensor
Computer, audio interface and amplifier
Steel wool, plastic container with steel threads and dynamic microphone element
Steel wool and dynamic microphone element
Speakers

Figure 6.17: *The Music Troll on display in the foyer of the Norwegian Academy of Music, January 2007. The box was built with speakers on each side, and with space for all the electronics inside. Each of the heads were designed with a specific musical function in mind, which is reflected in the physical, interaction and sensing design.*

6.6.1 Music Troll in Installation

The Music Troll premiered at our booth in the city centre of Oslo during the Norwegian Research Days in the autumn of 2006. Here it quickly became a big success, perhaps too big, as it was shaken and squeezed by some thousand children of all ages. The Troll was left in a fairly poor condition, something which led to the identification of several design problems related to the robustness of the construction.

The first problem was due to the 6 millimetre steel wires used in the arms not being strong enough to stand children hanging on the heads. This resulted in the wires bending so that the heads touched the ground. It was easy to bend the wires back up, but they should ideally have been somewhat stronger to allow for rough usage. The second problem was that the stitching in the arms and heads started to come loose since the children pulled at the arms from all sides. This rough use was also the reason why some of the wires inside the heads broke and shortcut some of the sensors. This could quickly be fixed in the heads where the electronics were easily accessible, but in others it meant opening the entire arm to fix the problem. During the autumn and winter we fixed and improved several of these design problems, and the Troll has survived several installations and a couple of concerts since then.

Despite some of these weaknesses in the design, the Troll has worked more or less as expected in installations. Most people understand how they can interact with the heads to produce sounds. It is also interesting to see how positive they are to the non-technological look and feel of the troll. The wooden box and the soft fur-like material used in the arms and heads, have made several people, who would normally refrain from testing an interactive art installation, interested in trying it out. This was particularly evident when the Troll was exhibited in the foyer at the Norwegian Academy of Music during the "Open Week" in January 2007. Here, I observed how people of all ages came back to play with the Troll every day, and many tended to touch it every time they passed through the foyer.

Even though the main ideas for the sound design used with the Troll have been the same for all installations, I have made some changes over time. Initially, the idea was that the heads should play sound through one speaker each, so that there would be a sense of direction of the sound. For the display in the city centre, however, I quickly realised that the background sound level was so loud that the sound of the Troll could barely be heard. To improve the situation, I ended up running all sounds through all the speakers, and I also added a software compressor to level out the dynamic range.

When the Troll was displayed in the foyer of the Music Academy, there was much less background noise. This allowed for using the directional sound as first planned. Here I also decided to use the four speakers to create various types of spatialisation effects, and created a drone that pulsated slowly and quietly in the background. This made the Troll sound like a "living" creature, with its slow "breathing" patterns, and this also worked as an attractor for people walking by. For later installations we have been talking about adding a distance-based sensor to detect people's presence, using for example an ultrasound sensor or a video camera. If the Troll can "see" people when they approach, it could start playing sounds to catch the interest of the people. This could open for a richer interaction between the Troll and people in the installation.

6.6.2 Music Troll in Performance

The Music Troll was always intended for installation, and this was also reflected in its design. I therefore hesitated a bit when I was invited to perform with the Troll during the Norwegian Research Council's annual gala event in Oslo Concert Hall in the autumn of 2006. I had three main concerns about this performance:

- would the action-sound relationships in the Troll work in a large concert hall?
- would it be expressive enough as a musical instrument?
- would it be stable enough for performance?

The interaction design of the Music Troll was deliberately made simple so that it would be intuitive as an installation. This was the reason why each of the heads only had one type of sensor, which greatly limited the interaction possibilities. In fact, the entire Music Troll had five degrees of freedom. This had not been a problem in an installation context where people would normally interact with the Troll for only a short while, but I felt the interaction possibilities were too limited for a 5-minute musical improvisation in a large concert hall.

To improve the musical possibilities, I decided to add a small MIDI controller which could be used to trigger cues along the way. This, of course, violated the whole idea of the Troll being a standalone "instrument", and reduced it to more of a large effects controller. However, the fact that it had so few degrees of freedom made this the only possible solution if I wanted to do an improvisation that was musically interesting.

The third question, concerning hardware and software stability, was probably what I was the most concerned about. There is no room for failure when you only have 5 minutes in front of a 1200 person audience. When it came to the hardware, the Troll had proven fairly durable, particularly taking into account the rough treatment it had received during installations. The software had also proven fairly stable, although it had crashed a handful of times during installations. This is not so bad considering that it was running for 12 hours per day, but the crashes had occurred somewhat randomly, and I was not entirely sure that it would be stable enough for a musical performance.

The initial plan was to use an Intel Mac Mini to run the software for the Troll. Unfortunately, the concert happened in the middle of the transition to *Universal Binary* software,[36] and I encountered a series of problems connected to incompatibility with various drivers, Max objects, libraries and software that did not work well together on either the old or new versions of OS X. Finally, I ended up using a Windows laptop to run the patch, which proved stable enough once I got everything to run. However, a software crash during the dress rehearsal did not settle my unrest about the stability of the setup, so I ended up creating a playback track that was ready in case of total failure. Fortunately, everything worked fine, and I could carry out the performance as planned.

After the performance several people commented that they found my large performing actions to work well even when sitting at the back of the large hall. As shown in Figure 6.18, I could use my whole upper body to shake and squeeze the different heads of the Troll. While most instruments usually look very small in such a large concert hall, the Music Troll was clearly visible.

6.6.3 Conclusions

Several conclusions can be drawn from the experiments with the Music Troll:

- *Durability*: the lesson learnt is that a construction can never be solid enough, especially not when it is to be used for installation. As we quickly discovered, both the construction of the arms and the padding of the sensors were too weak.

[36]Software that can run on both PowerPC and Intel Mac computers.

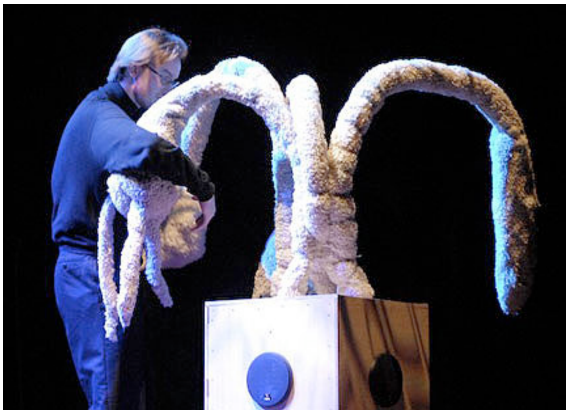


Figure 6.18: *Playing the Music Troll in Oslo Concert Hall in September 2006.*

- *Heat*: the temperature inside the box got very high after running the installation for a whole day. In fact, it got so warm that I feared the electronics might fail. The high temperature was mainly due to the large surround sound amplifier, so in future setups we will have to consider a smaller amplifier, or developing some kind of cooling system.

- *Stability*: the main patch for the Music Troll was running in Max using various third party externals (including PeRColate), and the drones were made with Native Instrument's Absynth. For the sake of portability and stability, it would have been better if everything could have been run in Max, using fewer third party externals. This would have made it easier to move the patch from one system to another, and would also have made it easier for other people to start the system. For the current setup I was the only person that managed to get the Troll up and running, which required me to always be around in case of failure.

- *Drivers*: we experienced a number of problems related to drivers for the USB-based audio interface, sensor interface and USB accelerometer. Due to various driver incompatibility problems with both PowerPC-based or Intel-based systems, I ended up running everything from a Windows laptop. While this system managed to install all devices and run all the necessary software, we experienced some random USB driver failures after some hours of operation. Every once in a while either one of the sensor interfaces or the audio interface would freeze, and the system had to be rebooted. For this reason I will probably use network-based or audio-based sensor interfaces in future installations.

- *Non-electronic feel*: one of the things I like the best about the Troll is its non-electronic feel, something which makes it easily approachable and likeable. Hiding all the electronics is crucial when it comes to an installation, but I believe this is also a good design strategy in general. In the future I will continue to build electronic instruments in which most of the electronics are hidden away.

- *Action-sound relationships*: I think the reason why the Troll felt so natural to use was that the heads were designed with a clear interaction and sonic design in mind. As such, the Troll is an example of how carefully designing the action-sound relationships in electronic devices does in fact improve the interaction and perception of the device.

6.7 Discussion

I was interested in comparing the many different controllers and instruments I had developed during my dissertation project, as well as various types of commercial interfaces I had been testing. So for a workshop and concert in January 2007, I set up a table with several of these devices so that we could test and compare them in a musical context. Figure 6.19 shows a picture of some of the equipment, including game controllers, music balls, graphical tablets, MIDI controllers, a Lemur graphical interface, screen-based interfaces, and a couple of computer vision-based systems. All of these were used in a workshop with professional musicians and music students, and this workshop ended up in an improvised performance. The following reflections are based on the possibilities of the devices as seen in the workshop and performance.

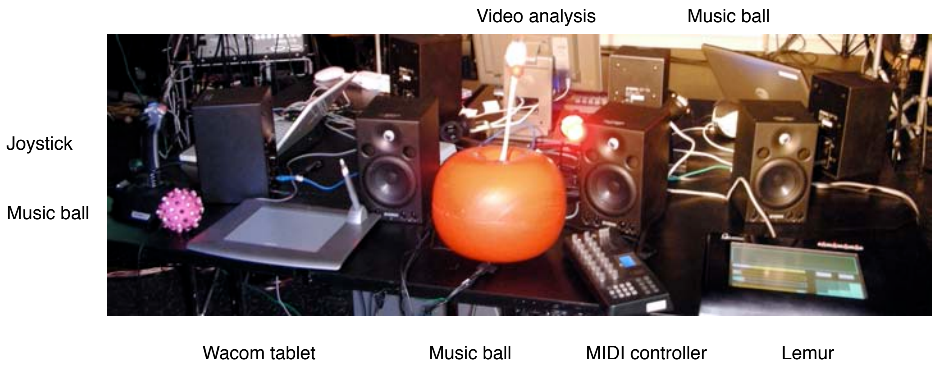


Figure 6.19: *Various types of commercial and homemade controllers set up for performance, including joysticks, music balls, graphical tablets, video analysis, etc. Note that each controller had a separate speaker to secure some directional sound.*

6.7.1 Comparing the Devices

Starting with the game controllers, I have already argued that they are cheap, stable, durable and easy to work with using my MultiControl software. One problem with game controllers, though, is that I often find it difficult to create action-sound relationships that feel immediately intuitive. Usually I end up using the fire button to play notes, and use the stick to control effects on the sound. This works well for people who are used to playing games with a joystick, but is otherwise not particularly intuitive. Another problem with joysticks is that they often have to be played sitting, and the control actions used are rather small. It is easier to stand up when playing a gamepad, but the actions used are still small and restricted, and not very visible on a large stage. As such, I often find game controllers to be more fun for the performers than for the perceivers.

The same problem can be seen when using *graphical tablets* in music performances. While their resolution, speed and fine control possibilities are great, they also force the performer to sit and look down at the table rather than communicating with perceivers. While it is certainly possible to create large theatrical movements when playing such a graphical tablet, these are not afforded by the controller itself. Also, the sheer physicality of using a pen and a small two-dimensional surface to control potentially large soundscapes does not feel very natural, in my opinion.

I find many of the same problems when performing with the *Lemur* controller.[37] This is a multitouch, screen-based controller, which allows for creating your own screen interfaces. It seems like many people often create lots of sliders and buttons when using the Lemur, something which brings it back to the concept of traditional MIDI controllers with a one-to-one relationship between an action and a sound synthesis parameter. This is not necessary, since the controller also allows for creating interfaces based on physical models, for example balls bouncing around the screen. This type of a dynamic interface is where the Lemur excels. Unfortunately, the Lemur is small and often ends up lying flat on a table. In small performance venues this might work well, since people can sit or stand close to the performer and actually see the screen and the actions performed on it. But in larger performance spaces, the Lemur is often too small. One solution to this problem might be to project a video of the interaction with the controller. This certainly helps guide people's attention in performance, but it still does not solve the problem of the performance space being reduced to a small screen.

While I have spent a great deal of time complaining about MIDI controllers, I do use them from time to time. For example, MIDI controllers based on the "knobs and sliders" paradigm are useful when having to control a large number of parameters or values directly. For this type of interaction, the physicality of a MIDI controller is certainly better than doing everything on screen with a mouse. While easily accessible, the computer keyboard and mouse do not work well in music performance in my opinion. Then a MIDI controller on which you can use all fingers at the same time is far superior. Also, as motorised MIDI controllers become cheaper and more readily available, they can be used to create a dynamic system with haptic feedback in a MIDI controller.

Video control through computer vision will be discussed more in Chapter 7, but generally I find that using computer vision as an instrument is problematic for many reasons. First, playing in the air in front of a video camera is difficult since there is nothing to touch, and it is difficult to know the boundaries of the performance space. Second, most computer vision models are based on calculating the *motion image*, and are thus sensitive to changing light. As such, they are often difficult to use in performance venues where the stage light is changing continuously. Third, computer vision, at least compared to

[37] http://www.jazzmutant.com

physical controllers, still does not allow for the temporal and spatial resolution which is required for musical performance. However, while I find computer vision techniques to be problematic for pure music control, I do find them interesting for creating interactive dance performances. Using a camera-based solution, dancers can move over a large space without having to use any on-body sensors and interfaces. An example of how I have been using computer vision in interactive dance will be presented in Chapter 7.

Comparing all the different controllers and instruments shown in Figure 6.19, I find the music balls to be the most interesting. It seems somewhat bizarre that after spending a considerable amount of time, money and effort on developing and testing various types of controllers and interfaces, I ended up preferring the cheapest and simplest ones. But this all comes back to the concept of action-sound couplings. The type of immediate sonic, tactile and haptic feedback you can get from an acoustic device will always be more direct than the one you get from an electronic device. Combining this with built-in sensors that allow for controlling sound effects calls for instruments that are both direct and intuitive, yet still provide rich, sonic interaction. It is interesting to see that many others think the same, as the music balls generally tend to attract a lot of interest before, during, and after performances.

6.7.2 Coding Actions

Mapping is often one of the most important and difficult challenges when it comes to developing electronic instruments. This is not only a conceptual problem, but also very much a practical one. While developing the controllers and instruments presented in this chapter, I found that the problem of mapping is highly related to the fact that we always work with *technical* parameters. While we may learn to understand what they mean, such technical parameters are, by definition, not very human-friendly. Trying to create mappings between such technical parameters is equally difficult. For example, mapping the x/y coordinates of a joystick onto the centre frequency and gain of a filter may seem logical to people with music technological training and experience, but it can hardly be called intuitive. Rather, this is a typical example of how we let ourselves be controlled by technology.

A more ecological approach to mapping would be to identify the action used (moving my right hand to the right) and the perceptual quality of the sound (changing the brightness and loudness of the sound) we want to control, and use these for mapping. This would result in systems where mappings are based on perceptually relevant parameters, which again would be more intuitive for people. Using meaningful parameters in mapping could also allow for more flexibility when it comes to interchanging controllers and sound engines. Looking at mappings as relationships between perceptually meaningful parameters, it should be possible to retain these relationships if the perceptual parameters stay the same. For example, using a joystick or a mouse to control a sound model would result in very different technical parameters, but the underlying qualities of the movements involved may be the same. Thus, if the mappings had been done at a perceptual level, it would be possible to interchange the joystick for the mouse and the mapping would still work. Similarly, this would allow for interchanging perceptually similar sound models even though they use different technical parameters. Such meaningful mappings will be discussed more in the context of the Gesture Description Interchange Format presented in Chapter 9.

6.7.3 Changing Roles

One of the things that has occurred to me over recent years, is how the new international trend of developing music controllers and instruments challenges many traditional roles in music. A traditional Western view has been that of a clear separation between *instrument constructor*, *musician* and *composer*. The idea has been that the *constructor* makes the instrument, the *composer* makes the score, and the *performer* plays the score with the instrument. However, in the community surrounding the NIME conferences, there are many people that take on all three tasks themselves. They make their own instruments, compose the music, and also perform themselves. It is interesting, but perhaps not too surprising, that a lot of this is happening outside the traditional music arenas, such as conservatories, concert halls and established festivals. This obviously threatens the traditional hierarchical and institutional systems in the music world.

This new trend in music technology also challenges the traditionally separated concepts of *performer*, *instrument* and *composition*. Using various types of neurophysiological, physiological or biomechanical sensors, performers themselves may become part of the instrument. Similarly, the instrument may become part of the composition through various types of algorithmic processing. The perceivers may also become part of both the instrument and the composition in systems based on audience participation and collaborative performance. As such, the notion of the traditional *concert* is changing, since many "instruments" and "compositions" may be used as installations in which the perceivers take an active part. In this way perceivers are turned into performers, and the composers end up as perceivers to the performance.

While some argue that this mixing of roles is unfortunate, I see it as a positive shift towards Machover's (2004) concept of *active music*. Even though music surrounds us everywhere, the musical experience is often *passive* in the sense that perceivers have little control over the music they are perceiving. However, the increasing interest in different performances, recordings, remixes and alternative versions of the same songs, show that people are interested in richer musical experiences. Active music may thus be seen as a way of giving perceivers music that they can control themselves. While such ideas have been explored by experimental composers and sound artists for decades, we now see that this type of activity is becoming more popular even outside dedicated art galleries. Examples are various types of collaborative instruments and performances, such as the *ReacTable* (Jordà et al., 2005), and various types of interactive web performances, for example by Freeman et al. (2005). These and other similar types of open performance systems show the possibilities of opening for a more active approach to music performance and perception.

6.8 Summary

The development of controllers and instruments presented in this chapter has been an exploration of many of the topics theoretically discussed in Chapter 3 on action-sound relationships. The aim has not been to create fully working controllers and instruments, but rather to make prototypes that show the potential of various concepts and ideas. As such, all of my controllers and instruments carry the tag "in development" and have only been used creatively on a few occasions each. I find this type of real-world testing important, as it gives some immediate answers to the usability and level of expression in a musical context.

It is interesting to note that some of the devices that I from a theoretical point of view would have thought to work well, turned out to be less interesting and intuitive in a performance situation. In addition, some of the most basic prototypes I developed, for example some of the electroacoustic music balls, turned out to be versatile, intuitive and exciting musically. Thus, such a practical "trial-and-error" approach has proven valuable to testing my theoretical ideas in practice, and the results from this practical work were important when developing the theories presented in the previous chapters.

The development of music controllers presented in this chapter, and the observation studies in the previous chapter, showed the need for various tools, methods and formats related to handling music-related movement material and data. The next chapters will present the development of some of these tools, methods and formats in more detail.

Part III

Development

CHAPTER 7

The Musical Gestures Toolbox

Technology at present is covert philosophy; the point is to make it openly philosophical.

(Agre, 1997, 240)

This chapter presents the *Musical Gestures Toolbox* (MGT), a collection of software tools for the graphical programming environment Max/MSP/Jitter, implemented within the *Jamoma* framework. Examples are given of how MGT has been used for analysis and performance.

7.1 Introduction

Starting out on the observation studies presented in Chapter 5, we looked around for software solutions that could help analyse relationships between body movement and musical sound, and that could:

- record and analyse both audio and video, and preferably also MIDI and sensor data
- annotate the recorded data
- carry out both quantitative and qualitative analysis
- work in realtime and non-realtime
- carry out comparative analysis
- work on Mac OS X

None of the tools we tested covered all our needs, and this was why I started to develop the Musical Gestures Toolbox (MGT). Before presenting the MGT, I shall review in brief some of the software solutions tested.

7.1.1 Software Review

There are a number of commercial software applications for video analysis. The greatest market seems to be software for surveillance purposes (e.g. in shops), but these solutions typically offer resolution and speed too low for our needs. Another large market share is applications available for sports and medical training. Such programs often have many quantitative features, but lack the qualitative perspective that we were interested in. Also problematic was the fact that many of these programs lack support for audio analysis, and that they only run in Windows.

Another branch of movement analysis software springs from the needs in behaviour and speech studies. Many researchers in these fields work on relationships between body movement and sound (speech), which is quite similar to many of our needs. Here we found *Anvil*,[1] a program intended for annotating videos of behaviour and gestures (used in a linguistic sense) (Kipp, 2001, 2003). The annotations appear as multiple colour-coded tracks of annotations, similar to an audio sequencer, and the input data can be exported as XML for further use and analysis in other programs. Adding to the list of positive things, Anvil is freely available, and since it is written in Java it works on all platforms. Furthermore, Anvil allows for importing spectrograms from the audio analysis software Praat,[2] which can be placed alongside the annotations. As such, Anvil is a good option when it comes to annotating movements and music, and several people in our group have used it successfully. However, Anvil does not carry out any quantitative analysis, nor does it allow for comparative studies of multiple video files. So we had to look around for more solutions to cover all our needs.

Standalone applications have the advantage of being easy-to-use, but are generally not very flexible. So for special needs it might be better to look at some of the many different types of *development environments* available. Such environments allow the user to create a custom-made application based on various types of building blocks. One such environment is *Matlab*, a scripting/programming language which has become standard in numerical computing. There are several libraries that simplify working with audio and video analysis in Matlab, but its text-based programming style was too challenging for some people in our group, and made us look for an easier solution.

As described in Chapter 5, *EyesWeb* is a multimedia environment developed at the University of Genoa (Camurri et al., 1999, 2004). EyesWeb is free (but not open source), and offers a large collection of tools for movement analysis, and in particular video. Since it is developed by researchers working on movement and music analysis and synthesis, the tools are highly relevant also for our research. This is one of the reasons we chose to work with EyesWeb at the beginning of the project. A major problem with EyesWeb,

[1] http://www.dfki.de/∼kipp/anvil/

[2] http://www.praat.org

however, is its Windows-only existence, which made it somewhat difficult to integrate with the rest of our software and hardware solutions.

Throughout the project, I also tested a number of multimedia development environments, including *vvvv*,[3] *Quartz Composer*[4] and *Processing*.[5] They all offer interesting solutions, but seem to be more focused on video synthesis than analysis, and none of them has any advanced audio functionality. As such, they are probably more interesting for video performance than analysis of music-related movement.

7.1.2 Choosing Max

We had found standalone software tools too limiting, the usage of Matlab too complex, and the multimedia solutions too narrow, and came to realise that we might be able to accomplish what we needed with the graphical programming environment *Max/MSP/Jitter*. *Max* was developed by Miller Puckette at IRCAM in the 1980s, and was originally a tool to control MIDI equipment (Puckette, 1985, 1988). The introduction of *MSP* in the late 1990s made it possible to carry out digital sound processing in realtime (Zicarelli, 1998), and the addition of *Jitter* in 2003 allowed for matrix operations, and video analysis and synthesis. Though it is a commercial application, Max has remained true to its experimental music origin, which is seen in its open structure and the ability to write new *externals*[6] to add functionality. This, combined with the excellent documentation and help files, cross-platform availability (Mac and Windows), and the possibility to make standalone applications, has made Max our development platform of choice. It also helped that several people in our group had experience of using Max for music creation, which simplified starting using it for analysing music-related movement.

The greatest objection to Max is its commercial nature and closed source code. As such, Pure Data (PD)[7] could have been a better option. PD is Miller Puckette's open source environment modelled on Max, and is much more flexible when it comes to hardware and software compatibility (Puckette, 1996). PD runs on most platforms available (everything from iPods to computers), which makes it a good choice when having to create solutions that are small and flexible. Similar to Max, PD also has a number of third-party extensions, and can even handle video analysis and synthesis with the *Graphics Environment for Multimedia* (GEM).[8]

However, while PD has many of the same features as Max, it is less developed when it comes to installation, use, documentation, help files, support and user group. So for most people (i.e. non-developers), Max is probably a better solution. The conclusion of our software review was therefore to use Anvil for annotations, EyesWeb for computer vision analysis, and Max to develop the other solutions needed in the project.

[3] http://www.vvvv.org
[4] http://developer.apple.com/graphicsimaging/quartz/
[5] http://www.processing.org
[6] An *external* is a small software extension to Max, written in the programming language C.
[7] http://www-crca.ucsd.edu/ msp/software.html
[8] http://gem.iem.at

7.2 The Musical Gestures Toolbox

The Musical Gestures Toolbox (MGT) started out as a small collection of patches and *abstractions*[9] to carry out various types of analysis for our observation studies. During the project, MGT grew into a more general collection of tools for working with audio and video analysis, and various sensor interfaces and devices. MGT was presented and made publicly available at the *International Computer Music Conference* (ICMC) in Barcelona in 2005 (Jensenius et al., 2005), and have been developed in several different directions since that time.

Developing MGT has been much more than only development. In many ways MGT can be seen as an answer to many of the research questions of this dissertation. As such, the development process has been one in which I have had the chance to test theories in practice, reformulate research questions, and find solutions to problems encountered in the observation studies.

MGT development also exposed a number of challenges related to programming in general, and Max and OSC in particular, several of which I have had to address to achieve what I wanted in MGT. For this reason, parts of this chapter (particularly Section 7.3) will present and discuss some details of the implementation, and these discussions necessarily require some knowledge of Max. For the most part, however, I will try to focus on the functionality of MGT rather than the implementation, and exemplify how MGT can be used for analysis and performance. But before presenting the current implementation, I shall give a quick overview of the early development of MGT.

7.2.1 Sound Analysis

SoundAnalysis was one of the first patches developed as part of MGT, and is a program for analysing sound in realtime. As can be seen from the screenshot in Figure 7.1, this patch displays a number of different types of audio analyses outputs. The patch was built around the **analyser**∼ object[10] by Tristan Jehan (2005), which takes a sound signal as input, and outputs a perceptually related analysis based on the categories *noisiness*, *brightness* and *loudness*, as well as a spectrogram using the psychoacoustical Bark scale. The patch displays these values, and also includes a sonogram, onset detection and pitch estimation.

The SoundAnalysis patch was originally developed for some pilot studies leading up to our first observation studies, as I could not find any easy-to-use applications for realtime audio analysis. The idea was that we could play the video recordings from our observation studies and look at correspondences between movement data and audio analysis data. In later versions of MGT, the functionality of SoundAnalysis has been implemented in the **jmod.analyzer**∼ module, so that it is possible to use it together with all the other tools in MGT.

[9] An *abstraction* is a Max patch developed so that it can be loaded as a separate *object* in another Max patch. Wrapping functionality into abstractions is an easy way to reuse patches and save some time and effort.

[10] http://web.media.mit.edu/∼tristan/

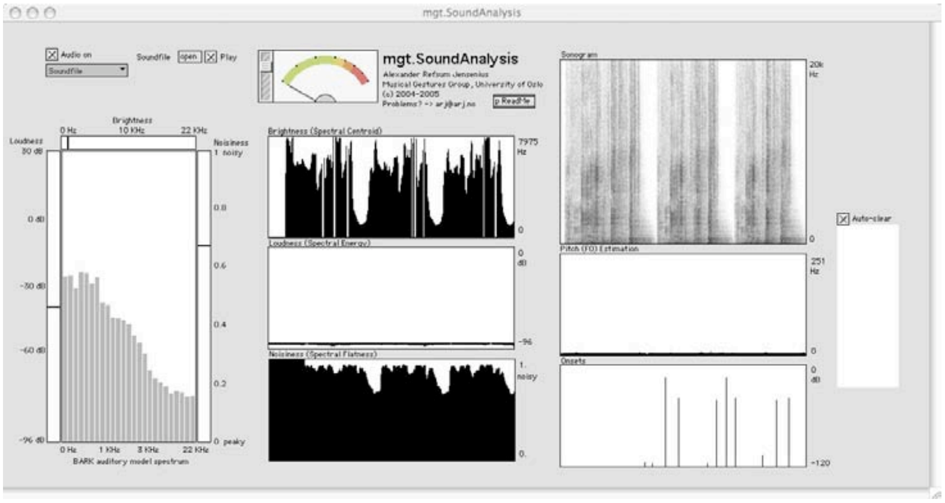


Figure 7.1: *Screenshot from* SoundAnalysis*, a patch built to analyse musical sound in realtime. The patch is based on* ***analyser~*** *which outputs various perceptual sound categories (noisiness, brightness, loudness).*

We only used the SoundAnalysis patch briefly for our observation studies, but I have continued to update the patch since several other people got interested in its functionality. To make it possible to use the patch without having Max installed on the computer, I created standalone applications in both English and Norwegian. These applications have been used at a number of workshops, summer schools and installations. For detailed audio analysis it is certainly better to use one of the many non-realtime audio analysis software available, but the realtime nature of SoundAnalysis has proven to be very useful in educational contexts.

7.2.2 Patch for Observation Studies

Another example of an early MGT patch is shown in Figure 7.2. This patch was used in our first observational studies, and provided tools for both qualitative and quantitative analysis of our material. The left side of the patch contains the playback functionality of video files with possibilities for scrubbing in the timeline display, setting looping points and adjusting the playback speed of the files. Here it was also possible to adjust the resolution of the video playback and whether the file should be played in colour or grayscale. There were also several types of video adjustment features available in this patch, for example possibilities for zooming in the image, changing brightness, contrast and saturation etc. As will be described in more detail later, all this functionality is still available in the latest version of MGT.

The four video displays on the left side of the patch in Figure 7.2 show various types

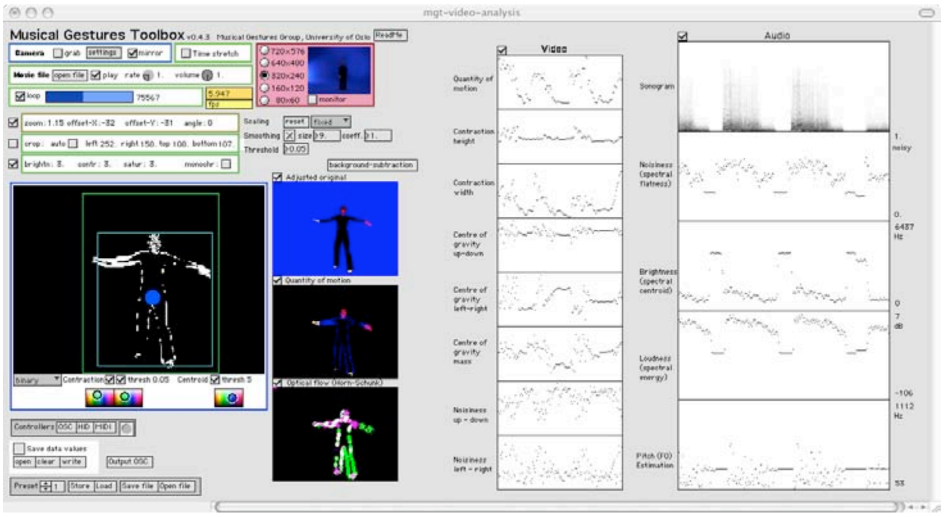


Figure 7.2: *Screenshot of a patch used in qualitative and quantitative analyses of music-movement correspondences in our first observation studies. The patch is an example of the early modular approach of MGT, where common functionality is built into self-contained modules that can be combined and reused in various ways.*

of motion images that we used for visual analysis. These displays were inspired by EyesWeb patches I had been working with in the beginning of my project. Just like in EyesWeb, I decided to include graphs of various types of quantitative video analyses, including the QoM, centre of gravity, etc. The functionality of the SoundAnalysis patch was also included in this patch, which facilitated studying relationships between movement and sound. All in all this patch proved useful in analysing the material from our first observation studies, as we had tools with which to inspect the videos visually, including flexible playback solutions and image adjustments.

7.2.3 A Modularised Approach

Working with Max, or any other type of programming environment, often results in re-implementing the same functionality over and over again. I quickly realised that all the patches I made reused parts from other patches. Rather than just copying and pasting these every time I made a new patch, I realised that it would be much more efficient to make a set of self-contained *modules*, small and self-contained patches that carry out a specific task. A *module* is similar to an abstraction in that it wraps functionality into one patch. The difference is that a module contains a graphical interface so that it can be loaded in a **bpatcher** object. A module typically contains more functionality than an external or abstraction, but less functionality than a whole patch. As such, a module is a practical building block since it carries out a specific set of operations, for example loading and playing back a video file. Since a module embeds both the interface and the algorithm(s) carrying out the processing, it is possible to change either the interface or the algorithm without having to update the patches where the modules are embedded. This results in faster and more flexible development and maintenance of a set of patches.

A modular approach not only benefits experienced Max users, but may be even more valuable for beginners and intermediate users. While it is fairly easy to get started with Max, the learning curve is quite steep and many beginners may not have enough experience to build efficient patches from *objects*.[11] One of the aims of MGT was therefore to create a collection of modules that could easily be connected and (re)used, and that would allow researchers in our group to set up complex patches without having to know all the details about the implementation.

Another reason for a modularised approach to Max development is that it allows for the creation of a common patching structure. A large part of MGT deals with various types of video analysis, and this typically requires a lot of the computer in terms of CPU and memory. It is therefore important to implement solutions to optimise data flow and video displays. Using a modularised approach, such optimisation features can be implemented in all modules, so that it is possible to change the size of the videos and turn off various processes as needed. A common patching structure also allows for more efficient data flow between different parts of the system, as well as saving *presets* in a structured way, i.e. storing a collection of system settings for later use.

The modular approach of MGT greatly helped in creating patches for our observation studies. However, as many Max developers have experienced, the structure and clarity of patches are often inversely proportional to the complexity. At some point I realised that I had outgrown the patching structure I had chosen, which made me spend too much time and effort on building the core functionality of the system itself rather than making modules. For this reason I started looking for solutions that could help solve structural problems in my patching. This was when I stumbled upon *Jamoma*, an open source project for standardising the development of modules and patches in Max.

7.3 Jamoma

Jamoma started out when composer and developer Tim Place decided to make the core structure of his then commercial *Jade*[12] interactive performance environment publicly available as an *open source* project.[13] The idea behind Jade, and later Jamoma,[14] was to develop a clear structure and a flexible framework for a modular environment in Max.

One of the main ideas in Jamoma is that a standardisation of modules will allow users to share their work more easily. Modules typically contain more features than objects, which make them more powerful, but also less flexible when it comes to integration with

[11] *Object* is often used to denote either an external or an abstraction, since it refers to functionality and not the implementation. In the Max world, many people have made their third-party objects available for others to use. A list of many of these third-party Max objects are available at http://www.maxobjects.com.

[12] http://www.electrotap.com/jade/

[13] Jamoma is released under the LGPL license (http://www.gnu.org/licenses/lgpl.html).

[14] In fact, Jamoma is an abbreviation for *Jade Modules for Max* (Place and Lossius, 2006).

other modules. This is because a module relies on a certain patching style and structure that may not be easily integrated with another patching style and structure. Creating a standard defining the *data flow* and *messaging* in and between modules would therefore allow for sharing and reusing modules between users.

Jamoma therefore seemed to fit well with my ideas in MGT, and I joined the project in 2005. At that time there were only two developers, Tim Place and Trond Lossius. The three of us worked together for some time, before Pascal Baltazar, Dave Watson and Nils Peters joined the project in 2006. As of writing, several other developers and users have become interested in Jamoma, and with the increased stability and functionality it is likely that it will receive more attention in the future.

7.3.1 The Structure of Jamoma

The modules are the main building blocks in Jamoma, and there are currently three different module types: *control*, *audio* and *video*. As can be seen in Figure 7.3, the three different module types share many properties, including the general look and size. This simplifies laying out modules in patches, and streamlines working with the modules.

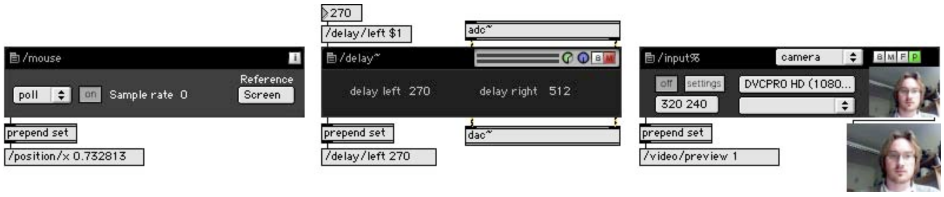


Figure 7.3: *The three types of modules in Jamoma:* control *(left),* audio *(middle) and* video *(right).*

The basic structural idea of a Jamoma module is that *control* messages are sent and received in the leftmost *inlet* and *outlet*,[15] while the other inlets and outlets are used for *signal* flow (audio and video). This clear division between control and signal data makes for a clear patching structure, and avoids conflicts between the different types of data. There are no restrictions on the number of inlets and outlets, so both audio and video modules can be made with any number of inputs and outputs.

Other common module features include the menus in the top left corner of each module. As the left screenshot in Figure 7.4 shows, clicking on the leftmost menu button will give access to core functionality such as options to freeze the GUI, save and load presets and view the help file or HTML documentation. As may be seen in the right screenshot in Figure 7.4, clicking on the name of the module will open a list with all the available control messages that can be sent to the module. This menu is practical when working with modules, since it is not necessary to open the documentation to check the name or syntax of a control message.

[15] An *inlet* is where messages are being sent into an object or module, and an *outlet* is where messages are returned from the object or module. This is common to all Max objects.

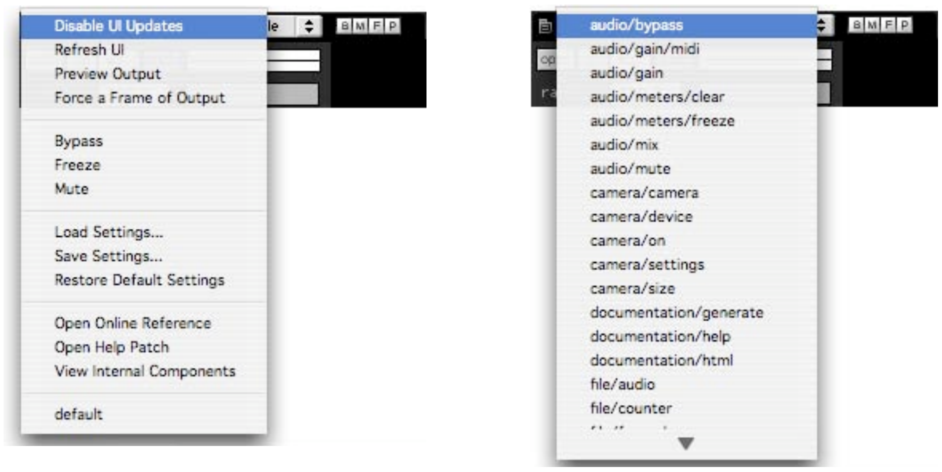


Figure 7.4: *The module menu (left) and namespace menu (right) of Jamoma modules*

In addition to the common module features, each module type also has some specific features. As may be seen in Figure 7.3, the *control* modules have few extra features. The *audio* modules, on the other hand, have small horizontal stereo VU meters to show the signal level, and a gain control to change the volume of the audio signal returned from the module. The audio modules also have a continuous mix control (*dry/wet*), which may be used to control how much of the processed audio signal should be returned from the module. This is not only practical when checking what is happening in a module, but may also be used creatively since it is possible to fade the effect in and out using this mix control. The audio modules also have a *bypass* (B) button, which will pass the incoming signal to the output without doing any processing, and a *mute* (M) button for stopping all audio processing in the module. These allow for quickly monitoring the signals, and saving CPU if some of the modules are not in use.

The *video* modules also have *bypass* (B), *mute* (M) and *freeze* (F) buttons that work on the video signal, as well as a small preview window which may be turned on and off with the *preview* (P) button. Modules can also have *inspectors* (i), windows that open and give access to features that are not available in the main user interface. All of these features are built into the core structure of Jamoma, and are present in the templates that can be used as the basis for developing Jamoma modules.

7.3.2 The Construction of a Module

An important concept underlying the construction of a Jamoma module, and which makes it somewhat different from traditional Max patching, is a clear separation between user *interface*, *processing* algorithm, and *presets*. In traditional Max patching, these three are usually combined in one patch, while in Jamoma they are implemented in separate files. A Jamoma module therefore consists of a number of different files, each having a certain functionality. For example, the **jmod.motion%**[16] module, which calculates the motion image from a video stream, contains the following files:

- **jmod.motion%.mxt**: the graphical interface of the module.
- **jalg.motion%.mxt**: the algorithm doing all the processing.
- **jcom.motion%.mxt**: the component sitting at the heart of the algorithm.
- **jmod.motion%.help**: the help file showing how the module can be used.
- **jmod.motion%.xml**: the preset file including the initial values of the module.
- **jmod.motion%.html**: a documentation file which can be loaded in a web browser.

Using different types of prefixes and suffices in the file names makes it possible to understand the function of a file by only looking at its name. The prefix of the name indicates the function of the file. For example, all files starting with *jmod* are files related to the module. As may be seen from the list above, there are four files starting with *jmod*, so here it is necessary to look at the suffices to understand their function. Files with the suffix *help* are help patches, *html* files are documentation files, *xml* are preset files, and the module file itself uses Max's patch extension *mxt*.[17]

There are two other prefixes in use in the Jamoma structure: *jalg* and *jcom*. The files with a *jalg* prefix contain the *algorithms* used inside modules. While the actual module file (**jmod.motion%.mxt**) only contains the user interface and the structure of the module, the algorithm file (**jalg.motion%.mxt**) is where the processing actually occurs. The *jcom* files represent *components* that may not be directly related to any modules, but may be more focused on general aspects of the inner workings of Jamoma. In the **jmod.motion%** module, I use a component called **jcom.motion%** to calculate the motion image inside the algorithm file (**jalg.motion%**). The functionality of this component could easily have been implemented directly in the algorithm, but I often find this functionality interesting to use separately from the module, so I decided to implement it as a standalone component. Usually, most components are not so closely connected to a module, and are more focused on solving a more general task. In fact, many components are independent from the structure of Jamoma, and so general that they may be used as normal third-party objects.

[16]In Max all audio objects have a tilde (∼) after the name to differentiate them from the control objects. Similarly, the video objects have the prefix *jit* (for example **jit.window**). In Jamoma, Tim Place chose to use % as the suffix of video modules, since all modules already had a *jmod* prefix. The idea of the % was because it resembles a "film roll", and it could therefore be seen as consistent with the use of the "wave-like" tilde (∼) for the audio objects in Max. Using the % is inconsistent with the naming convention used for Jitter, but we have found it to be easier to work with since the file names become shorter.

[17]The reason we save all files as *text* files (and not *binary*) is because this makes it possible to read the source code and make comparisons of different development versions in the repository at SourceForge (http://sourceforge.net/projects/jamoma/).

I will not go into detail about the core functionality of Jamoma, but only give an overview of how the modules and algorithms work. Figure 7.5 shows parts of the **jmod.motion%** module. The user interface part of the module is visible in the upper left corner. Trying to avoid too many *patch chords*,[18] all communication between the interface elements and the rest of the patch occurs through named objects and corresponding **pvar** objects. The **pvar** objects are connected to **jcom.parameter** or **jcom.message** objects that handle the state of these parameters and messages, including their data *types*,[19] *ranges*,[20] *clipmodes*,[21] *rampmodes*[22] and *descriptions*. Defining all these (and some other) parameters inside the patch makes the module self-contained, and makes it possible to query the module for its state of parameters and messages. This functionality is also something Trond Lossius exploited in a script which can generate HTML documentation files for each module based on the descriptions defined in the **jcom.parameter** and **jcom.message** components of the module.

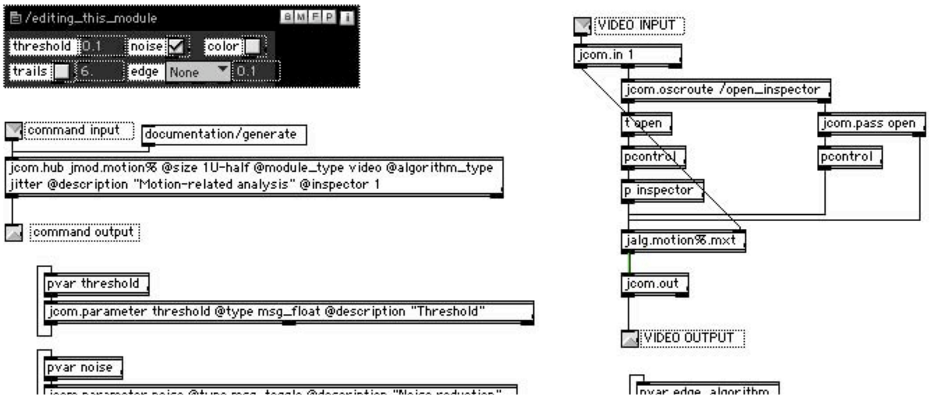


Figure 7.5: *Parts of the inner workings of a Jamoma module, with the GUI in the top left corner, and all the objects defining the messages, parameters, attributes and data flow of the module.*

The heart of all Jamoma modules is an instance of **jcom.hub**. This is the core object which handles all the "magic", including the preset handling[23] and communication between modules. As can be seen in Figure 7.5, the **jcom.hub** is instantiated with several messages, defining the size of the module (in "rack units"), the module type (control, audio, video), the number of inlets and outlets of the module, and a description which is used for documentation. The **jcom.hub** is responsible for adding the correct user interface *skin* to the module, which defines the looks of the module, and the extra func-

[18]The *patch chords* are the lines drawn between objects, and which define the data flow in patches.
[19]For example integer, float, list, etc.
[20]Set with low and high values, for example 0-100.
[21]The clipmode denotes whether a value should be *clipped* when it goes beyond the defined range. For example, if the range is 0-100, this will result in all values above 100 to be returned as 100.
[22]The *rampmode* defines whether messages should be *ramped*, or interpolated, when new values are set.
[23]The preset system in Jamoma is based on the **pattr** family of objects in Max.

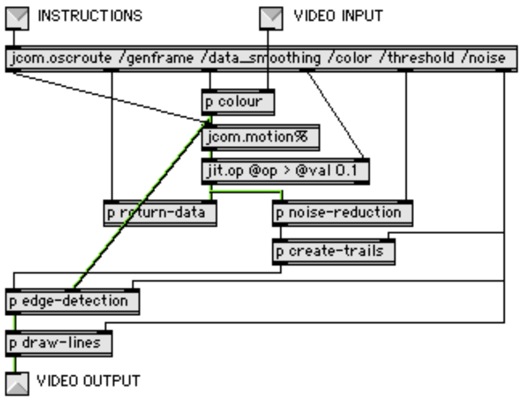


Figure 7.6: *The algorithm file of the motion% module contains multiple layers of processing that can be turned on and off. All incoming OSC control messages are routed to the relevant subpatches.*

tionality of audio and video modules. The right side of the patch is where the signal flow in the module is set up, by connecting the **jcom.in** and **jcom.out** objects (or **jcom.in**∼ and **jcom.out**∼ for audio modules) to the algorithm (in this case **jalg.motion%.mxt**). Notice that no processing is done in the module patch, it is only used to set up the communication to and from the algorithm.

As for the modules, the algorithms are structured in a specific way. As can be seen in the screenshot of **jalg.motion%.mxt** in Figure 7.6, the leftmost inlet and outlet of algorithms are used to receive and send control messages. These OSC messages are parsed with the **jcom.oscroute** component and sent to the correct places inside the algorithm. The other inlets and outlets are used for signals (in this case video). The **jalg.motion%.mxt** algorithm is a rather big and complex patch, consisting of several layers of processing, each of which is broken into a separate subpatcher which can be turned on and off at will. The functionality of this algorithm will be described in more detail later in this chapter.

7.3.3 Flexible Communciation

One of the strengths of Jamoma is the flexible communication between modules. This is made possible through the **jcom.hub** components inside each of the modules, which can dynamically create links between parameters or messages. In the early days of Jamoma, such communication was done using regular Max messaging through **send** and **receive** objects. The problem with using such "hard" messaging techniques (regular patch chords included), is that patches often are difficult to reuse in various ways. A more flexible approach was suggested by Wright et al. (2001) and Zbyszynski and Freed (2005), who have been using OSC messages to communicate not only between systems but also inside patches. This makes it possible to use global **send** and **receive** objects for handling all messaging, and use OSC objects for routing messages to the relevant destinations in the patches. Such an approach also makes it easy to create multi-computer setups, since messages can quickly be sent out on the network. I had been using such an OSC approach to Max patching in the early versions of MGT, so adding support for OSC messaging in Jamoma was my first initiative when I joined as a developer. The other developers accepted, and since Jamoma version 0.3 OSC has been used for all messaging.

An example of such OSC communication is shown in Figure 7.7. Here the global **jcom.send** and **jcom.receive** objects are set up to send and receive messages to and from a module. Since these messages can be sent from anywhere inside Max, or even from another program or computer, it is possible to control anything from anywhere.

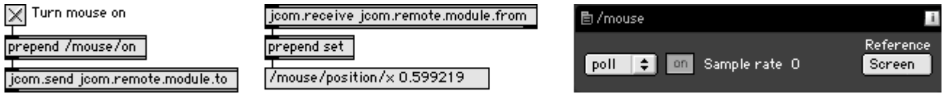

Turn mouse on
prepend /mouse/on
jcom.send jcom.remote.module.to
jcom.receive jcom.remote.module.from
prepend set
/mouse/position/x 0.599219
/mouse
poll
on
Sample rate 0
Reference
Screen

Figure 7.7: *Jamoma's global send and receive objects make it possible to control and receive messages outside modules. This example shows how processing in the* ***jmod.mouse*** *module (right) is turned on through a* ***jcom.send*** *object (left), and messages are received from a* ***jcom.receive*** *object (middle).*

In the above example the first part of the OSC message (/mouse) refers to the name of the module. Parameters and messages can be accessed directly by their names (and subnames) inside the modules, but from outside it is necessary to add the module name in front of the message, using this syntax:

```
/module/parameter <values>
```

The name of modules can be chosen by the user and set as an *attribute* in the **bpatcher** object when loading a module. Multiple instances of the same module can therefore be loaded and given different names so that they can be addressed separately. An interesting feature is that it is also possible to give multiple modules the same name. This allows for the control of multiple modules with the same control messages, and has been particularly practical for setting up patches for comparative studies where there are multiple versions of the same modules.

7.3.4 Mapping

The ability to access modules from anywhere greatly simplifies creating mappings between modules. Several modules are created for this purpose, such as the **jmod.mapper** module made by Tim Place, which can "learn" mappings when the user activates the messages or parameters that are to be connected. Pascal Baltazar has made another mapping module, **jmod.cont_mapper**, which allows the user to create mappings by choosing from menus containing lists of all available modules and their parameters.

This mapper module is using internal Jamoma functionality to query the namespace for all its addresses, a querying technique which is also used in **jmod.cuelist** made by Trond Lossius. The **jmod.cuelist** module allows for listing the state of all parameters and messages. Here it is possible to select the desired states and save them in a preset file. It is also possible to create a cuelist with multiple states that may be triggered by calling the relevant cue. All in all, the mapping and cuelist modules are some of the most exciting features of Jamoma, and allow for quickly creating rather complex setups that are also easy to maintain and change.

7.3.5 Meaningful Messaging

When it comes to message handling, we have decided to use names and namespaces that are *meaningful*. If the messages passed around are cryptic, it does not help if the communication system is flexible. For this reason, the aim has been to create namespaces that are fully understandable, and that are typed out in plain language. For example, we prefer to use "frequency" instead of "freq" or "f", to be sure that it cannot be misunderstood. This results in long namespaces that are inefficient from a processing point of view, but in my experience the efficiency gains on the human processing side far outweigh the extra CPU cycles that are needed to process such messages. There are exceptions, though, for example when it comes to passing high speed motion capture data. These types of data should probably not be communicated together with the control messages anyway, but should rather be passed on as "signals" just like audio and video data.

Another important aspect when it comes to communicating meaningful messages and data is to standardise the values, ranges and units used. As far as possible we try to normalise data to a 0-1 decimal range for control values. For audio modules, pitch is coded as both frequency (Hz) and MIDI values, while loudness is coded using a decibel (dB) and MIDI range. The reason for using dual messaging for in these modules was to be able to use either a technical parameter (Hz and dB) or data from a MIDI device to control the modules. However, as will be discussed later in this chapter, we have found that it is probably better to create a "unit library" which can handle conversion between units automatically.

7.3.6 Automatic Scaling

To handle scaling between various ranges, I developed **jcom.autoscale**, a component which will automatically scale values to a given range. This component is based on "learning" the maximum and minimum values of the input data and using these as the basis for scaling the output data (similar to the method used in MultiControl, shown in Figure 6.7). As may be seen from a screenshot of the help patch in Figure 7.8, **jcom.autoscale** operates in different *scaling modes*:

1. scaling based on continuous learning
2. scaling based on learned range (requires training with mode 1 first)
3. scaling based on a running window (in samples)
4. scaling based on the mean and standard deviation of a running window (in samples)

Choosing which scaling mode to use in **jcom.autoscale** depends on the type of input values. If the range of the values is known and stable, such as in a game controller, it

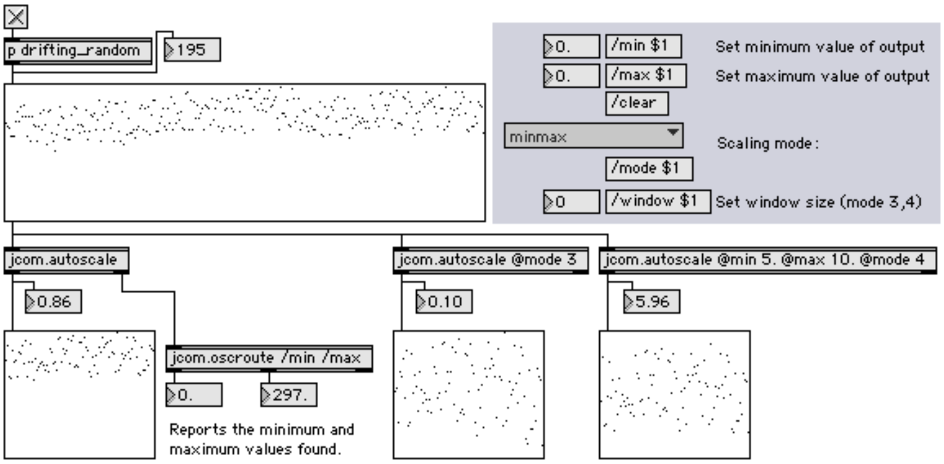

p drifting_random
195
0.
/min $1
Set minimum value of output
0.
/max $1
Set maximum value of output
/clear
minmax
Scaling mode:
/mode $1
0
/window $1
Set window size (mode 3,4)
jcom.autoscale
jcom.autoscale @mode 3
jcom.autoscale @min 5. @max 10. @mode 4
0.86
0.10
5.96
jcom.oscroute /min /max
0.
297.
Reports the minimum and maximum values found.

Figure 7.8: *The **jcom.autoscale** component is handy for automatically scaling values to a given range. It operates in four different modes defining whether the learning algorithm in the module should be based on all received values, or only a given window (in samples).*

is most efficient to use mode 1 to learn the range, and then switch to mode 2 as soon as the range is learned. However, if the input values are changing and/or unknown, it might be better to use mode 3 or 4. When working with sensors that may not have a clearly defined range, for example accelerometers, these are my preferred modes. Whether to use mode 3 or 4 depends on how scattered the values are. Since mode 4 is the most computationally demanding, I typically only use that mode if the values are noisy and change considerably. In that case it might also be worth filtering the data before scaling them, for example using a simple low-pass filter such as the **slide** object.

7.4 Functionality

Rather than dwelling too much on general aspects and implementation details of Jamoma, the rest of this chapter will focus on presenting some of the functions and modules of MGT, as they are now implemented in Jamoma. All the examples are available on the accompanying CD-ROM, together with a full version of Jamoma. A list of my contributions to the development of Jamoma is available in Appendix A. Since much of the following discussion will focus on digital video processing, I shall start with a quick overview of some basic features of digital video.

7.4.1 Digital Video

A digital video file can be thought of as a series of digital photographs. Each of these photographs has a certain *size*, usually defined in pixels. For a long time, the relationship between the height and the width in video images has had a 4:3 ratio. Computer video files therefore often have a size of 640x480 pixels or 320x240 pixels. Nowadays, other types of ratios are becoming popular, for example 16:9, but the 4:3 ratio still seems to be a standard in many programs, including Jitter.

In Jitter it is common to refer to the images in a video file as *matrices*, since they are basically a matrix of numbers. In fact, Jitter is actually a general purpose matrix processing tool, even though it is optimised for working with video matrices. Each of the matrices in a digital video sequence consists of one or more *planes*. The number of planes defines the colours available in the image. A grayscale video only has one plane, while a typical colour video usually has four planes. There are several different types of colour representations, and one of the most popular used in computers is called ARGB (alpha channel, red, green, blue). The number of planes is important for the computational cost of processing a video, and an ARGB colour video requires four times the processing power needed for grayscale video.

The *resolution* of the video image is another factor which is important for the processing load. A normal digital video file has a resolution of 8-bits, which means that each of the pixels in each plane is measured in a range of 0–255, where 0 is black and 255 is white. Sometimes it is necessary to carry out calculations using a higher resolution, and this will therefore increase the processing load.

Finally, the *frame rate* of a video file defines how fast each of the images in the video is played back. There are several different frame rates in use, two of the more popular being the European *PAL* standard using 25 *frames per second* (fps), and the North American *NTSC* standard uses 30 fps.[24] Several new video formats use higher frame rates for recording video, such as the 60 fps used in *High Definition Video* (HDV). Obviously, the frame rate is another important factor influencing the processing load when working with digital video, and the computer may *skip* frames in the video if it cannot keep up with the pre-defined frame rate.

All the above mentioned factors (size, planes, resolution and frame rate) are important for the processing load and final visual result of the digital video. Therefore, creating software solutions will always have to balance quality versus efficiency. For this reason I have implemented a number of tools in MGT that can change the features of a video in realtime.

7.4.2 Video Input

The **jmod.input%** module is the heart of most video patches built with MGT. This module provides basic input functions from a digital video camera, a computer video file or simple video synthesis. Gathering all of these in one module ensures that patches can quickly be used for both live input and video file playback, since no patching has to be done to switch from one to the other. This has proven useful when working on experimental setups in which it is possible to use a recorded file for testing, and then use live input for the observation studies.

[24]NTSC actually uses 29.97 fps, but this is referred to as 30 fps.

Figure 7.9 shows a screenshot of **jmod.input%**. The user can start by choosing the input mode (camera, file, or synthesis) in a drop down menu at the top of the module. The *camera* pane provides access to menus containing lists of the available devices in the system,[25] as well as access to the internal settings for the device, and size of the video (in pixels). I typically prefer to work with small video sizes (for example 320x240 pixels) when testing patches, since this allows for some overhead in the system. When adjustments are done, it is possible to turn up to a larger size for outputting files or data analysis. As will be discussed in later sections, changing the size of the video can be done in realtime, and subsequent modules will adjust automatically.

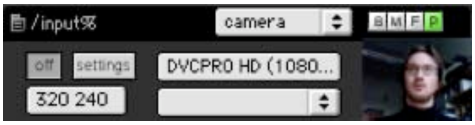


Figure 7.9: *The camera pane of the **jmod.input%** module allows for the retrieval of live video from any video camera that works with the system, for example USB or FireWire cameras.*

The *video file* input pane of **jmod.input%** is shown in Figure 7.10, and allows for the loading of any QuickTime readable video file for playback.[26] This pane also gives access to move around in the file during playback by clicking in the timeline display. To be able to study only a part of a video recording from our observation studies, I implemented the possibility to select loop points using the slider above the timeline display. For creative applications it is also possible to define whether such looped segments should play forwards, or in *palindrome* mode (playing forwards and backwards).

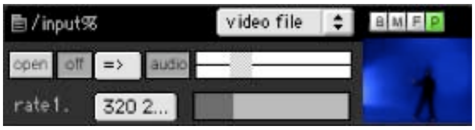


Figure 7.10: *The video file pane contains controls to load and play back any QuickTime readable video file, as well as scrubbing and moving around in the video file using the timeline display.*

When studying the videos from our observation studies, we have often found the need to control the playback speed of the video, either to study movements in slow motion or to navigate quickly through longer files. However, changing the playback rate of the movie in **jit.qt.movie** will also influence the pitch of the sound of the movie. To preserve pitch during playback, I have added a digital phase vocoder for pitch shifting the audio proportional to the change in the playback rate. As shown in Figure 7.11, this is implemented with the **gizmo~** object, in which the pitch scaling factor is control by the

[25]There is a subpatch checking whether the patch is running on a Mac (QuickTime) or Windows (Direct X) system, and the relevant input objects (either **jit.qt.grab** or **jit.dx.grab**) will be chosen accordingly (using Max' scripting abilities).

[26]It is also possible to connect the **jmod.file_browser** module to quickly select between multiple files in a folder or folder hierarchy.

inverse of the playback rate of the video file. This allows one to study the videos in slow motion, while still hearing the audio at the original pitch. The pitch shifting functionality is also useful when analysing different performances for which it is necessary to adjust the playback speed to maintain synchronisation, but the pitch of the sound of which would be preserved.

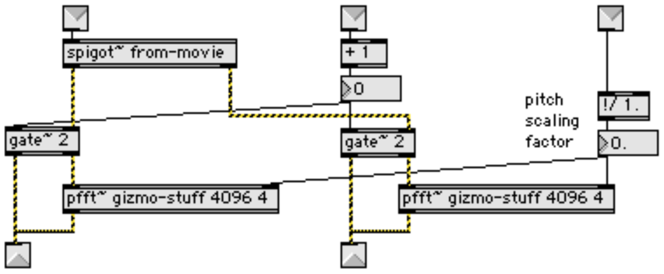


Figure 7.11: *The solution for keeping the pitch of the audio when changing the speed of the video. This is implemented with the* ***gizmo∼*** *object which is controlled with the inverse of the playback rate of the video file.*

The third pane of **jmod.input%** is shown in Figure 7.12, and implements a simple *noise synthesiser* for playing a noise function of different sizes, planes and frame rates. This is mainly intended for testing purposes in case there are no video cameras or video files at hand.

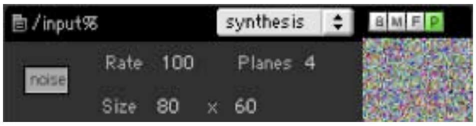


Figure 7.12: *The synthesis pane allows for playing video noise, something which might come in handy for testing purposes.*

All in all, **jmod.input%** has proven to be one of the most flexible and useful modules in MGT. It is usually the first module I include in new video patches, and one which often saves time and effort due to its many features.

7.4.3 Adjustment Modules

As mentioned in Chapter 5, one of the most important needs when studying the videos from our observation studies was to adjust the videos during playback. Our needs ranged from simple image adjustments (brightness and contrast) to rotation, zooming and cropping. Such features are not only useful in visual inspection, but often important when pre-processing the video before passing it on to analysis modules.

Zooming and rotation has been implemented in the **jmod.orsize%** module,[27] which is presented in Figure 7.13. Here the *rotation* and *flipping* functions can be used to rotate

[27]The name *orsize* is an abbreviation for *orientation* and *size*.

videos recorded in *portrait* view, as it is necessary to shift the image 90° to see the video correctly. The *zooming* function works from the upper left corner of the image, and it is usually necessary to adjust the location of the image in the frame after zooming. This can be done by using the *offsetting* parameters in either X or Y direction. To simplify this process, I have implemented the possibility to zoom and offset the image by clicking the mouse in the preview window of the module. Zooming can be done by holding down the shift key while click-dragging with the mouse, while only click-dragging will result in an image offset. This is a small feature, but one which I am very satisfied with, since it allows for quick adjustments.

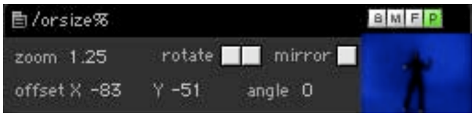


Figure 7.13: *The **jmod.orsize%** module is used for zooming, orientating and sizing the video.*

Sometimes it is necessary to remove unwanted and distractive elements from an image. It is possible to use **jmod.orsize%** to zoom, but this will result in an interpolation of the image matrix to match the original size of the video. A better option is to *crop* the video by removing the unwanted parts of the image, something which will also change the size of the matrix. I have implemented realtime "drag-and-crop" in the **jmod.crop%** module, which lets the user click-drag with the mouse in the video, and select the area of the video which should be cropped. This cropping functionality is not only useful in removing distractive elements, but can also be used to study a specific part of the image, for example the hand region in air piano studies (Figure 7.14).

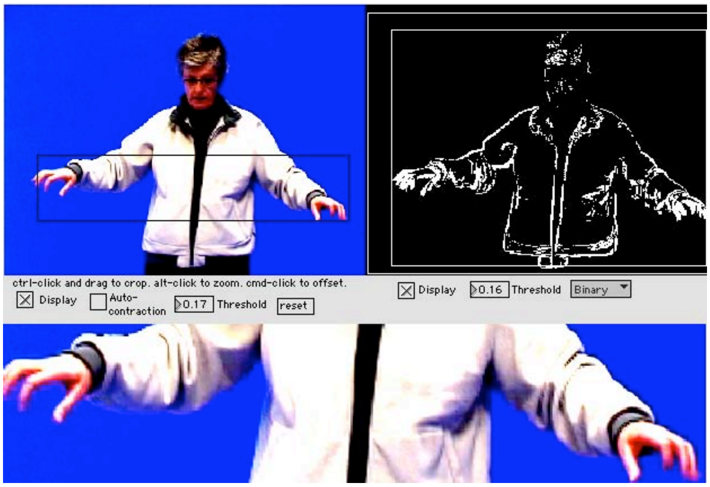

ctrl-click and drag to crop. alt-click to zoom. cmd-click to offset.
Display
Auto-contraction
0.17
Threshold
reset
Display
0.16
Threshold
Binary

Figure 7.14: *The cropping function in **jmod.crop%** makes it easy to zoom and crop the video by clicking and dragging in the source window.*

For videos where the subject is clearly separated from the background, it is possible to use the automatic cropping functionality of **jcom.autocrop%**. As shown in Figure 7.15, this component is based on the **jit.findbounds** object which finds the boundaries of an object in the image. By looking for the minimum and maximum values returned from **jit.findbounds**, **jcom.autocrop%** outputs a video which fits the maximum bounding box found in the video stream. This has been useful in our studies of free dance movements, since this component can effectively crop these videos so that only the part of the image which the subjects moved in is visible.

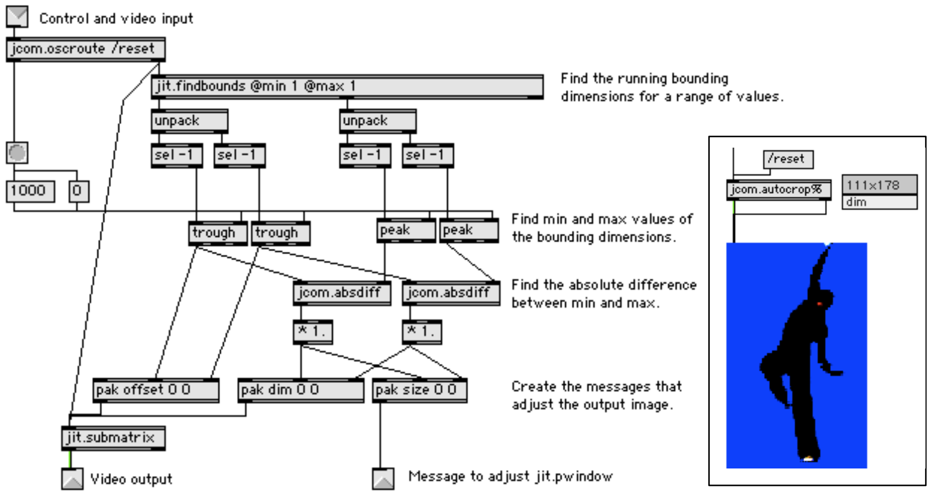

Control and video input
jcom.oscroute /reset
jit.findbounds @min 1 @max 1
Find the running bounding dimensions for a range of values.
unpack
unpack
sel -1
sel -1
sel -1
sel -1
1000
0
trough
trough
peak
peak
Find min and max values of the bounding dimensions.
jcom.absdiff
jcom.absdiff
Find the absolute difference between min and max.
* 1.
* 1.
pak offset 0 0
pak dim 0 0
pak size 0 0
Create the messages that adjust the output image.
jit.submatrix
Video output
Message to adjust jit.pwindow
/reset
jcom.autocrop%
111x178
dim

Figure 7.15: *The algorithm of the* ***jcom.autocrop%*** *component (left) is based on finding the minimum and maximum values returned from the* ***jit.findbounds*** *object. This allows for automatical cropping of the output video (right).*

Another useful pre-processing video module is **jmod.brcosa%** which may be seen in Figure 7.16. As the name of the module implies, it can change the *brightness*, *contrast* and *saturation* of the image. This module helps improve the *signal-to-noise ratio* of the image, which is important for both visual and machine analysis. The module also allows for the conversion of colour videos to grayscale, which helps reduce the processing load in large patches. Similar to **jmod.orsize%**, the parameters in **jmod.brcosa%** can also be changed by click-dragging with the mouse in the preview window in the module.

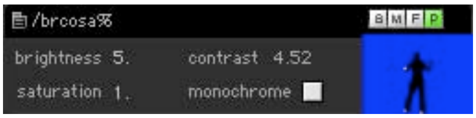


Figure 7.16: *The* ***jmod.brcosa%*** *module allows for simple image adjustments: brightness, contrast, saturation and grayscale.*

7.4.4 Matrix Variability

As mentioned earlier, the standard ratio in video images for a long time used to be 4:3, and this is also the native ratio used by several Jitter objects. This has been a major concern in developing MGT, since we often use video files with other ratios. Many new DV cameras record using the 16:9 ratio which has also become standard in new TVs. We have also increasingly been recording in portrait view to get a better "person-to-pixel" ratio in the video. This results in ratios of 3:4 or 9:16 depending on the camera used. Cropping videos also results in equally non-standard image ratios. Using such non-standard video ratios with objects that expect 4:3 video often leads to the image being stretched to wrong proportions. Thus, one of the premises in MGT development has been that all video modules should adapt dynamically to the properties of the input matrix.

Many Jitter objects are "transparent" to the properties (size, planes, etc.) of the incoming matrix,[28] and will automatically adjust to the incoming matrix. However, there are some objects that are not transparent, and one of the most notable ones is **jit.pwindow**, the object which is used to display videos inside a patch. Figure 7.17 shows an example of a 4:3 video recorded in portrait view, and how it is displayed sideways when played with **jmod.input%**. This can easily be corrected with the rotation functionality of **jmod.orsize%**, but, as may be seen in the middle of Figure 7.17, the **jit.pwindow** which is used to display the image does not update to reflect the change in video matrix size. The result is that the 3:4 ratio image is squeezed into the native 4:3 frame of **jit.pwindow**. To solve this recurring problem, I developed the **jcom.autosize%** component, which will automatically and dynamically change the size of the window to match that of the input matrix.

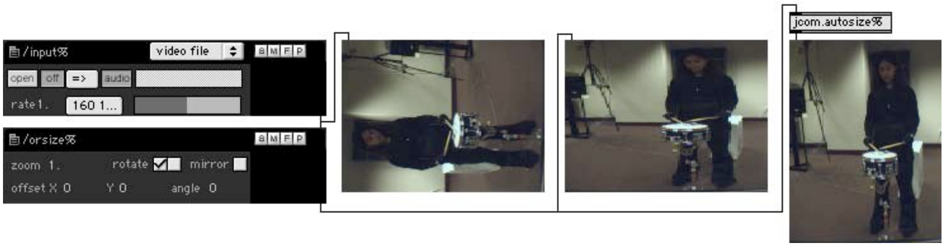


Figure 7.17: *An example of how a video shot in portrait view can be rotated with* ***jit.orsize%*** *(middle). Using the* ***jcom.autosize%*** *component adjusts a* ***jit.pwindow*** *to fit the input video size and ratio (right).*

The dynamic nature of **jcom.autosize%** is useful to change the size of **jit.pwindow** objects, but this may lead to overlapping **jit.pwindow** objects if they are laid out too closely. This is often the case when changing between small and large matrices. To account for this it is possible to use the attributes of **jcom.autosize%** to specify either the desired height or width of the following **jit.pwindow**. An example of this is shown in Figure 7.18, where either the width or height of the video is defined.

[28] This is the case when using the adapt=1 flag in for example **jit.matrix**.

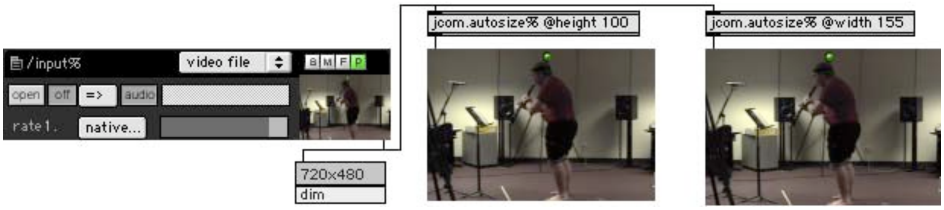


Figure 7.18: *An example of how a DV video in NTSC resolution (720x480, ratio 3:2) is scaled correctly in* ***jit.pwindow*** *while constraining either the height or width.*

Changing the number of planes in a matrix is another problem which often leads to problems. Typically, Jitter will complain if a 1 plane matrix (grayscale) is sent to an object which expects 4 plane matrices (ARGB colour), and vice versa. To adjust for this, I implemented **jcom.checkplanes%**, a small patch which can be seen in Figure 7.19. This patch automatically updates the following object so that it can receive the incoming matrix without any problem.

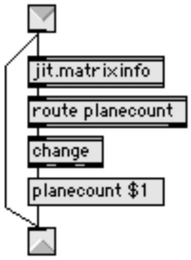


Figure 7.19: *The* ***jcom.checkplanes%*** *component adjusts subsequent objects to update the number of planes they expect in the incoming matrix.*

Another challenge working with different planes is that some Jitter objects cannot change the number of planes, and only accept either 1 or 4 planes. This calls for objects that can convert from 1 to 4, or 4 to 1 planes respectively. This can be solved with the **jit.luma2rgb** or **jit.rgb2luma** objects, but these native Jitter objects require you to know the number of planes you are converting from, which is often not the case in the dynamic nature of MGT. For example, I often prefer to use grayscale images to create motion images. This requires converting from colour to grayscale, which may be easily done with **jit.rgb2luma**. However, sometimes the video may have been converted to grayscale in an earlier module before it gets to the **jit.rgb2luma** object. This will result in an error from **jit.rgb2luma**, since it only accepts 4 plane matrices as input. To solve this problem I developed **jcom.luma2rgb%** and **jcom.rgb2luma%**, which convert to or from 4 and 1 planes respectively. They are similar to **jit.luma2rgb** and **jit.rgb2luma**, except that they will both accept either 1 or 4 plane videos as input. As shown in Figure 7.20, this is implemented by checking the number of planes in the input matrix before doing the conversion. These components also help save some CPU since they only carry out the conversion if it is necessary.

Yet another example of how MGT is focused on matrix variability is the implementa-

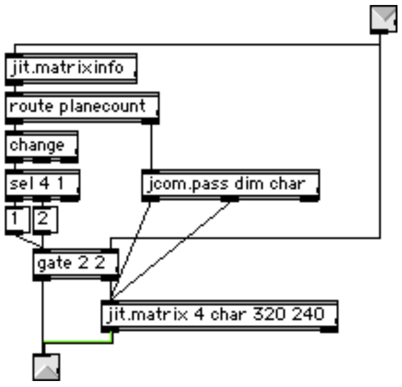


Figure 7.20: *The **jcom.luma2rgb%** component converts a 1 plane matrix (grayscale) to 4 planes (ARGB colour), but only if necessary.*

tion of components to facilitate changing the resolution of the matrices. Some operations typically have to be carried out at a greater resolution than regular video. For example, the **jcom.mean%** component would start to clip values if calculations were done in regular 8-bit resolution. In such cases **jcom.char2float%** and **jcom.float2char%** come in handy, as they can convert between Jitter's *char* type (8-bit) and *float* type (32-bit) numbers.

I have implemented several other similar components to help working with changing matrices in MGT (see Appendix A for details). Each of these components only carries out a small task, but together they allow for changing the properties of matrices without having to worry about whether this will cause problems in other parts of the patch.

7.4.5 Background Removal

In computer vision it is common to talk about the *background* and the *foreground* of an image. The foreground is typically an object or a person which is the focus of the analysis, while the background may be considered "noise" that should be removed to enhance the foreground. The background was not so much of a problem in the recordings from our observation studies, since most of these videos were recorded using a plain background. It was therefore possible to use fairly simple background removal techniques for our material.

Figure 7.21 shows a patch for simple background removal. This is based on recording the background and then subtracting this recorded image from the input image. Note that I am using a low-pass filter (**jit.slide**) before storing the background image. This is to remove any unwanted movements, changing light, or other types of artefacts that may reduce the quality of the background removal process. The end result is an image where only the foreground is visible. This example is a very simple and computationally efficient way of removing the background, but does not always work that well. As may be seen in the resultant foreground image in Figure 7.21, parts of the background image are visible in the foreground.

A more advanced background removal is implemented in the **jmod.background%**

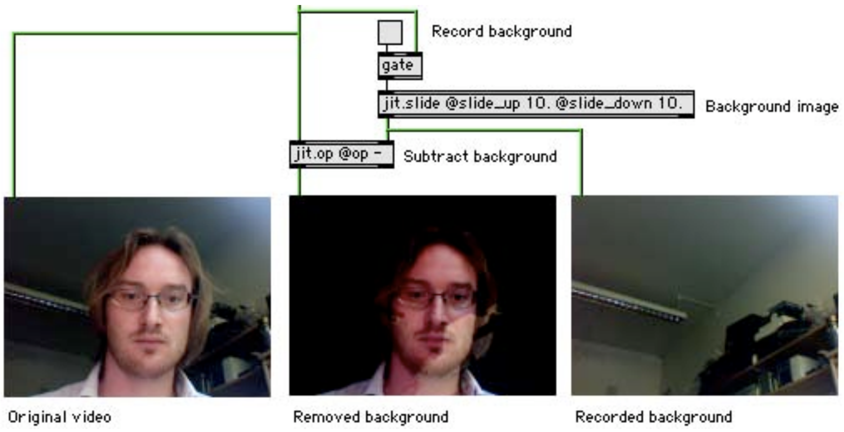


Figure 7.21: *An example of simple background removal, based on recording the background separately, and then subtracting this image from the input video stream. The end result is that only the foreground appears in the image, but the background may sometimes be visible in the foreground image (as seen in this example).*

module. This algorithm is based on a patch found in Jean-Marc Pelletier's cv.jit collection,[29] but it is modified to use only native Jitter objects.[30] The main idea of the algorithm underlying this module (**jalg.background%**) is similar to the simple background removal algorithm presented above. The main difference is that this algorithm finds a "foreground mask" which is used for subtraction. As the output images in Figure 7.22 show, this technique results in a better foreground/background separation, even in this example with a fairly complex background. Unfortunately, this technique is much more computationally expensive than the simple background removal technique presented above. Choosing one of these therefore depends on the processing load of the rest of the patch.

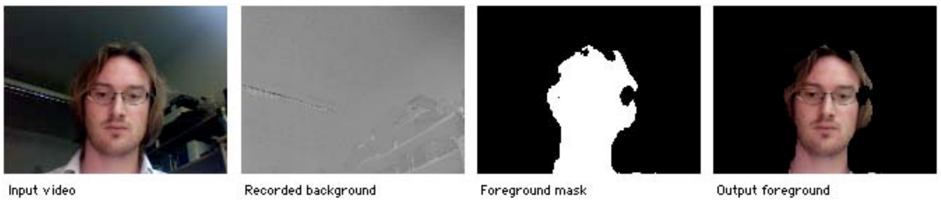


Figure 7.22: *A more advanced background removal technique, based on finding the "foreground mask" and subtracting this from the running input, implemented in the* ***jmod.background%*** *module.*

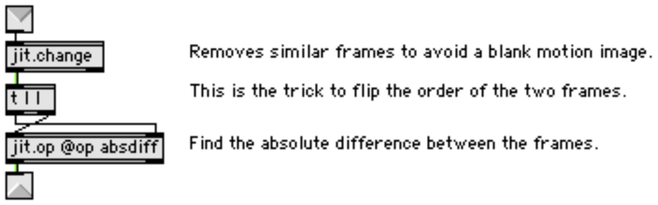
[29]http://www.iamas.ac.jp/ jovan02/cv/

[30]We are trying to avoid third-party objects in Jamoma, to reduce the risk of broken objects after software and hardware updates.

Another way of removing the background is to use a technique called *chromakeying*. This technique is common in TV and film production, and is based on removing a given colour from the image and substituting this with another colour or image. Chromakeying is implemented in the **jmod.chromakey%** module by Pascal Baltazar, and works very well on the video recordings from our observation studies with an even coloured background.

7.4.6 Creating Motion Images

A popular method for analysing movement from videos is the concept of *frame differencing*. This is simply based on subtracting consecutive frames in a video stream, which will leave only pixels that have changed between the frames. The result is what is often called a *motion image*. Simple frame differencing is implemented in the **jcom.motion%** component as shown in Figure 7.23. This component is also at the core of the **jmod.motion%** module.

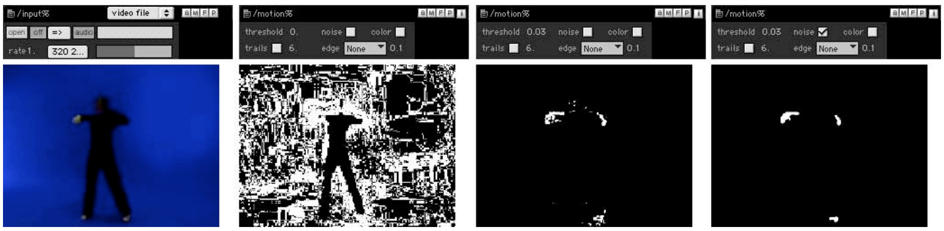


```
jit.change          Removes similar frames to avoid a blank motion image.
t l l               This is the trick to flip the order of the two frames.
jit.op @op absdiff  Find the absolute difference between the frames.
```

Figure 7.23: *The* ***jcom.motion%*** *calculates the absolute difference between consecutive frames.*

Using simple frame differencing on a normal video stream often results in a very noisy motion image. An example of a noisy motion image can be seen in Figure 7.24 (image 2 from the left), made from Video 7.1 of one of our free dance recordings. There are several reasons why this video results in such a noisy motion image, one being the fluorescent lighting used in the room. Another reason is that the video has been stored with a *lossy* video compression technique. One way to overcome such problems is by pre-processing the video before calculating the frame difference, by changing for example the brightness and contrast of the image. It is also possible to filter the noisy motion image, for example by using a low-pass filter. This is implemented in the **jmod.motion%** module, and as may be seen in image 3 from the left in Figure 7.24, applying a rather small threshold value in the module improves the quality of the output image considerably. By also applying a noise removal algorithm, based on removing all single pixels, the motion image becomes even clearer (image 4).

The **jmod.motion%** module has a number of other features, of which the visual outputs are summarised in Figure 7.25. The "edge" menu in the module gives access to several different types of edge detection, which will draw the edge of the objects in the video on top of the motion image. The "trail" functionality is a simple video feedback (using **jit.slide**), which results in a *motion history image*. As will be discussed more in



/input% video file
open off => audio
rate 1. 320 2...

/motion%
threshold 0. noise ☐ color ☐
trails ☐ 6. edge None 0.1

/motion%
threshold 0.03 noise ☐ color ☐
trails ☐ 6. edge None 0.1

/motion%
threshold 0.03 noise ✓ color ☐
trails ☐ 6. edge None 0.1

Figure 7.24: *The* ***jmod.motion%*** *module has options for filtering and removing noise from the motion images.*

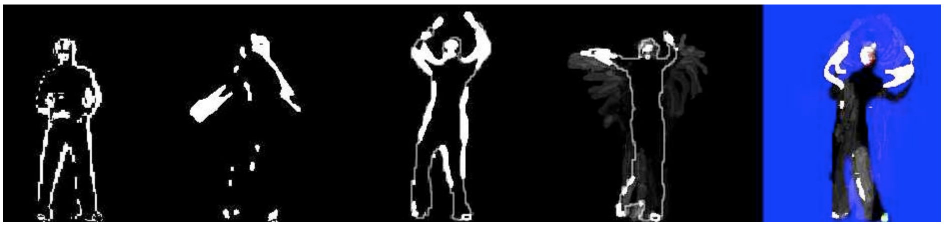


Figure 7.25: *Different motion images (left to right): raw frame-difference, with smoothing and threshold functions, with edge detection, with a feedback function, and added to the original video.*

Chapter 8, using such visual representations has proven to be a useful tool when studying videos from our observation studies.

The above mentioned features of **jmod.motion%** are mainly intended for qualitative analysis. But the module can also be used for quantitative analysis, since several OSC streams with analysis data are output from the left outlet of the module. Figure 7.26 is a screenshot from the help file of the module, and shows how it is possible to route the different data streams using the **jcom.oscroute** object. The returned values include the *quantity of motion* (QoM), and the spatial centre of the QoM.[31] As discussed in Chapter 5, the QoM is correlated to the level of movement in the image and is therefore a perceptually interesting measure, even though it should be used with caution.

The **jmod.motion%** module will also calculate the size of the contraction box around the object in the image (using **jit.findbounds**), which tells something about how much of the space of the frame the person fills. These values are not only available as data streams, but can also be visualised using thin lines overlaid the output video,[32] as shown in Figure 7.27. The module allows for the drawing of a bounding box which follows the subject, similar to the bounding box shown in the EyesWeb patches in Chapter 5. The **jmod.motion%** module can also draw a box displaying the maximum movement space that the subject has been using, which correlates to the kinesphere of the person. I find it practical to see the relationship between the running and the maximum bounding boxes,

[31]The QoM is here calculated as the sum of all pixels in the motion image divided by the number of pixels in the image, implemented in **jcom.sum%** made by Tim Place.

[32]Turning on these features can be done in the inspector panel of the module.

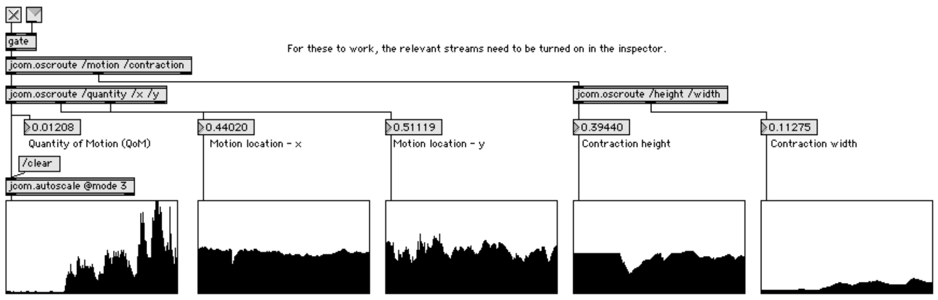

gate
jcom.oscroute /motion /contraction
For these to work, the relevant streams need to be turned on in the inspector.
jcom.oscroute /quantity /x /y
jcom.oscroute /height /width
0.01208
Quantity of Motion (QoM)
/clear
jcom.autoscale @mode 3
0.44020
Motion location - x
0.51119
Motion location - y
0.39440
Contraction height
0.11275
Contraction width

Figure 7.26: *Analysis data are output from the left outlet of the module and can be used for quantitative analysis or visual inspection of graphs.*

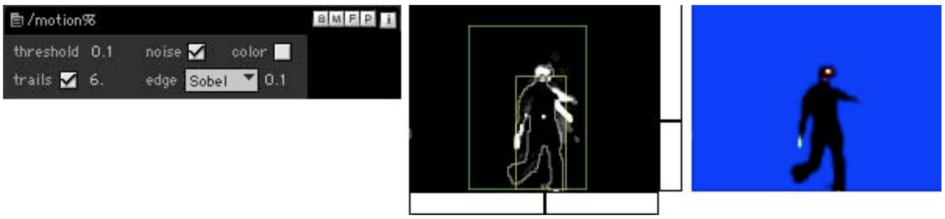


Figure 7.27: *It is possible to turn on the display of running and maximum contraction boxes as well as the "centre of gravity" in the inspector panel of the module.*

as this relationship effectively displays the contraction and expansion of the person. The **jmod.motion%** module can also plot the centre of the QoM in the video image, which may be seen as correlated to the centre of gravity of the person. The ability to visualise all these quantitative measures is interesting for qualitative analysis, since they enhance features that may not be easily seen in the video.

With its many features, the **jmod.motion%** module is one of the MGT modules I use the most, alongside **jmod.input%**. It is also the basis for creating *motion history images* and *motiongrams* which will be discussed in more detail in Chapter 8.

7.4.7 Various Input Modules

This chapter has so far mainly focused on tools for video processing and analysis, but MGT and Jamoma are not restricted to this. As mentioned in Chapter 5 and 6, I have also been working on many different types of sensors systems and controllers, and there are several modules with this type of functionality. There are modules for getting data from standard devices such as the computer mouse (**jmod.mouse**), joysticks (**jmod.hi**), *Wacom* graphical tablets (**jmod.wacom**) and midi devices (**jmod.midiin**). I have also made modules for retrieving data from *Polhemus* electromagnetic trackers (**jmod.polhemus**) and several of the sensor interfaces from *Phidgets* (**jmod.phidgets.accelerometer** and **jmod.phidgets.interfacekit**). Other Jamoma developers have contributed modules for working with some popular sensor interfaces, including the *Teabox* (**jmod.teabox** made by Tim Place) and the *Arduino* (**jmod.arduino** made by Nils Peters). This makes Jamoma able to work with most popular interfaces and controllers.

Most of these input modules are just *wrappers* around third-party externals. A wrapper is a module or object which does not implement any new functionality, except for being a "shell" around another external, and which may therefore be considered redundant. There are several reasons why I still think it is worth creating such wrapper modules. One reason is that of getting access to the preset handling, mapping possibilities, and documentation system available in Jamoma. Another reason is our attempt to standardise the messaging used to control modules, and the data returned from modules. As discussed in Chapter 6, various game controllers typically use different messaging and values, and it gets even worse when comparing the messaging of generic devices, such as mouses, joysticks, tablets, etc. This is not so much a problem with Max itself, but rather a result of different drivers and implementations. In Max this is often worsened by the fact that several objects for retrieving data from such devices are made by third-party developers, which results in rather inconsistent implementations.

An example of this type of inconsistency may be seen in Figure 7.28. This patch shows an example of two standard objects, **mousestate** and **hi**, which retrieve data from a mouse or a joystick, respectively. Notice that the **moustestate** object has to be *polled*[33] with a metro object to return values, while the **hi** object has polling built into the object. When it comes to the output, **mousestate** returns values in pixels relative to the upper left corner of the screen, while the **hi** object usually returns values between 0 and 255 under Mac OS X and from 0 to 65 536 under Windows XP.

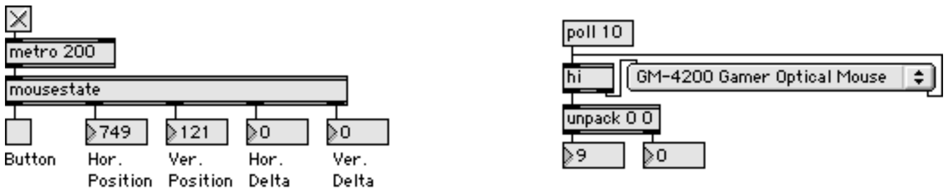


Figure 7.28: *An example of the inconsistency in the control and output of similar Max objects.*

The control messages and output data from other input device objects are equally different, and this inconsistency is a challenge when developing systems that are scalable and flexible. Imagine swapping the mouse for a joystick to control a synthesiser, and all the problems this leads to in setting up the objects correctly and scaling the data. Thus, creating a set of input modules that look and behave similarly has been one of my main goals in developing Jamoma, and will be discussed further in the context of the Gesture Description Interchange Format in Chapter 9.

[33]Sending a message asking for an output to be sent.

7.5 Applications

The previous sections have described the building blocks of MGT and Jamoma. This section will present a few examples of how I have combined these modules into patches used for analysis and synthesis.

7.5.1 Comparative Analysis

In Jamoma one module can be loaded multiple times in the same patch, something which is useful for comparative analysis. An example of such is shown in Figure 7.29, and was used to create the combined videos presented in Chapter 5. The patch is made from three equal "blocks" of modules: **jmod.input%** for playing the video files, **jmod.orsize%** for zooming and offsetting the image, **jmod.brcosa%** for changing brightness and contrast, and **jmod.motion%** for creating motion history images overlaid with edges. The outputs from the modules are combined into one video with the **jit.glue** object, and can also be saved as a video file using **jmod.record%** (made by Trond Lossius).

One of the challenges of working with video processing is that it requires a lot of computer processing power. This becomes even more apparent when working with multiple video files at the same time, which typically results in skipped frames during playback. To prevent this from happening I have implemented functionality for using MGT in *non-realtime*, processing the movie "as fast as possible" while retaining the intended quality. This is done by sending a *framedump* message to **jit.qt.movie**, which will only output a new frame when the patch is finished processing the former. This facilitates saving high quality video files to disk, an example of which can be seen in Video 5.4.

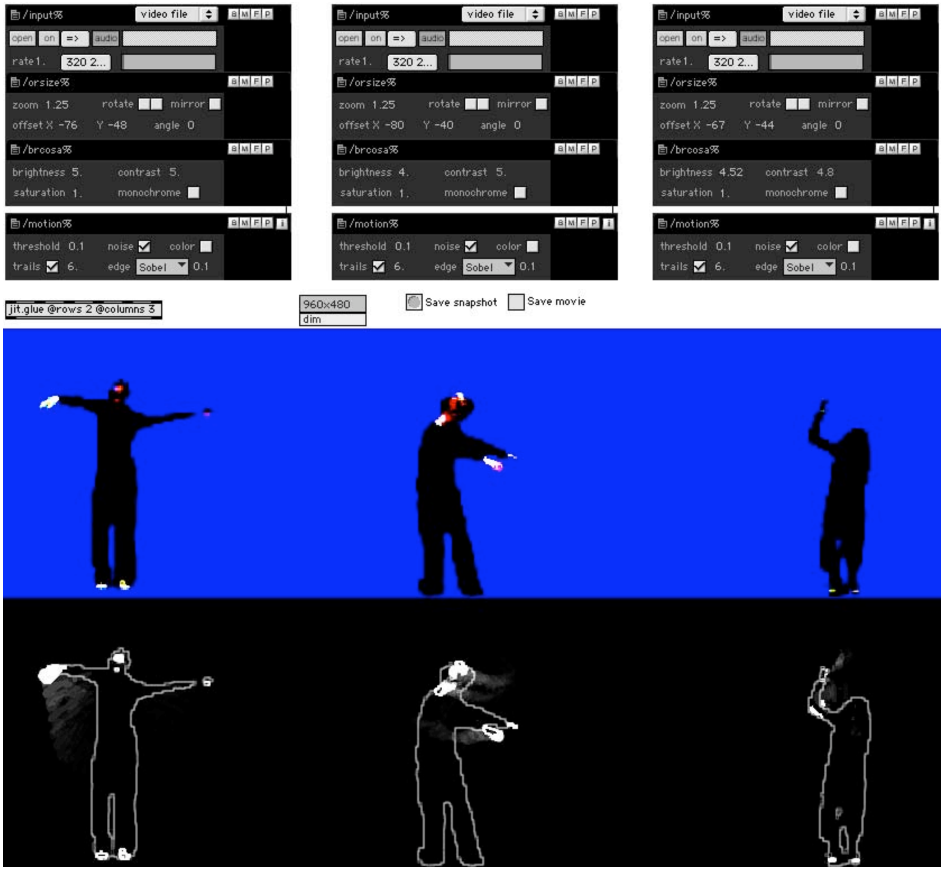
The non-realtime mode is also practical for carrying out various types of *batch processing*, processing multiple files with the same type of algorithm. In such cases, the non-realtime mode may even be used to process files *faster* than realtime, depending on the size of the files and the computer processing power available.

7.5.2 Interactive Dance

Though my focus in developing MGT has mainly been on analysis, I have also been interested in testing the tools for creative applications. This interest grew from the observation studies, in which we had studied how dancers moved to music. I wanted to turn this around, and look at how dance movements could be used to control musical sound.

The field of dance technology has developed rapidly over the years, and a number of different solutions have been presented in research and performance. Many of these solutions are based on sensor techniques, where the user has to wear lots of electronics on the body. Guedes (2005, 98) argues that such invasive sensor solutions violate the idea of dance as an art form, since the dancer is turned into an "interface operator". I agree with Guedes that it is more interesting to explore various types of non-invasive solutions for interactive dance, and that computer vision techniques stand out as a good candidate.

Several software environments have been developed over the years for working with interactive dance, one of the most flexible platforms being EyesWeb. However, a num-



rate 1. 320 2... | rate 1. 320 2... | rate 1. 320 2...

zoom 1.25 rotate mirror | zoom 1.25 rotate mirror | zoom 1.25 rotate mirror

offset X -76 Y -48 angle 0 | offset X -80 Y -40 angle 0 | offset X -67 Y -44 angle 0

brightness 5. contrast 5. | brightness 4. contrast 5. | brightness 4.52 contrast 4.8

saturation 1. monochrome | saturation 1. monochrome | saturation 1. monochrome

threshold 0.1 noise ✓ color | threshold 0.1 noise ✓ color | threshold 0.1 noise ✓ color

trails ✓ 6. edge Sobel 0.1 | trails ✓ 6. edge Sobel 0.1 | trails ✓ 6. edge Sobel 0.1

Figure 7.29: *A patch made for comparative analysis of free dance movements, based on adding three equal "blocks" of modules, and combining them in one image. The patch may be run in realtime, or files may be saved to disk using the non-realtime mode.*

ber of other standalone software solutions exist, for example *BigEye*[34] from STEIM, *Music via Motion* (MvM)[35] by Kia Ng (2004), and *EyeCon*[36] by Frieder Weiss and the Palindrome company[37] (Wechlser et al., 2004). There are also several computer vision solutions available for Max/MSP, the most extensive set being available in *Jitter*, and extended by the *cv.jit* collection as mentioned above. Another solution for Max/MSP is David Rokeby's *SoftVNS*[38] objects, as well as Cyclops[39] by Eric Singer. Several of the solutions found in these software applications have inspired my approach to using MGT for interactive dance.

My approach to using MGT in interactive dance was mainly focused on the relationships between movement and music, and the ability to make the dancer able to directly control musical sound as a "musician". Together with dancer Åshild Ravndal Salthe, I have created a series of pieces for interactive dance over recent years. Working with singer Maria Fonneløp, we developed a piece called *Whiteness I Remember* in the autumn of 2005. The musical idea was to create a "noise" piece based on short musical grains that could be controlled by the movements of the dancer. Sonically, the textures would be very thin, scattered and high pitched at the beginning of the 5 minute piece, and end up being large, heavy and low pitched at the end. The idea was that the dancer should improvise with the sounds, and be in dialogue with the improvisation of the singer.

The implementation for *Whiteness I Remember* was built around **metashaker∼** from PeRColate. As discussed in Chapter 6, I prefer such physical, or physically inspired, sound models, since they simplify the mapping process due to their intuitive parameters. The idea was that the movements of the dancer should directly control the sound, so that there would be no sound when the dancer stood still, and lots of sound when the dancer moved energetically. For this I created a direct mapping between the QoM of the dancer and the energy level of the sound model. Turning down the threshold function in **jmod.motion%**, even the smallest dance movements was detected and ensured that the dancer was in direct control of the sound generation, something which can be seen and heard in a short excerpt from a performance of the piece in Video 7.2.

When it came to controlling the other features of the sound model, I used several different movement analysis techniques. Due to the limitations of the small stages we performed on, we created a 2x2 meter "box" that the dancer could move within. As shown in Figure 7.30, this box was divided into a grid with 3x3 zones, where the QoM was calculated for each of the 9 zones (using **jcom.motion%**). This made it easy to detect which zones the dancer was moving in, and this information was used to control the *grain frequency distribution* of the sound model. Since movements within each zone resulted in a slightly different timbral quality, it was possible to "see" the zones through listening only. This was practical for the dancer, since she could hear where in the grid she was located, and could adjust her position if she was moving out of the "box".

[34]http://www.steim.org/steim/bigeye.html
[35]http://www.kcng.org/mvm/
[36]http://www.frieder-weiss.de/eyecon/index.html
[37]http://www.palindrome.de/
[38]http://homepage.mac.com/davidrokeby/softVNS.html
[39]http://www.cycling74.com/products/cyclops

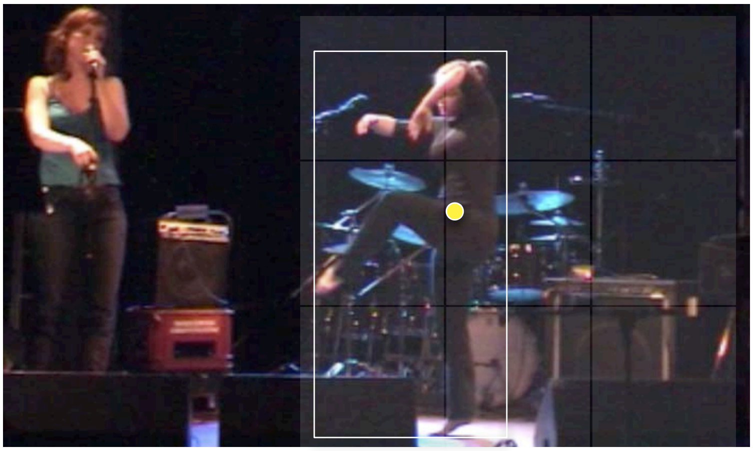


Figure 7.30: *From a performance of* Whiteness I Remember *at the Blå jazz club in Oslo, December 2005. The dancer would control the grains of the sound texture based on the quantity of motion, the expansion/contraction of the body, the centre of gravity, and the position within a 3x3 grid.*

As may be seen in Figure 7.30, I also used the bounding box and the centre of gravity of the dancer as control parameters in the piece. The running bounding box was used to control the "randomness" in the **metashaker**~ module, which resulted in a "looser" sonic texture when the dancer moved her arms or feet far from the body. The centre of gravity was used to control the centre frequency of the sound grains, which made the overall "pitch" of the sound grains correlate to the vertical position of the dancer. Thus the mappings between movement and sound may be seen as fairly "simple", since any movement would create sound, and the quality of the sound was controlled by the horizontal and vertical position of the dancer. Nevertheless, we found that this mapping worked better than many of the "complex" mappings we tested. Most importantly we found this simple mapping to be more intuitive both to the dancer and the perceivers. This simplicity made it possible for the dancer to "play" the sound model through her movements, and also simplified the interaction with the singer.

Using MGT for performance has presented very different challenges than those found in the observation studies. All our observation studies were recorded in controlled conditions, with an even background and stable lighting. This is very different from performing on stages with changing lights. In the beginning, I spent quite a long time adjusting the modules to work every time I set up the system in a new location, but over time I have been able to build in functionality and presets in the various modules to facilitate stable and reliable patches. This is obviously important to being able to create successful mappings later on.

A number of conceptual questions arise when working with dance movements as the source for sound control. Traditionally, dance is a sound-accompanying activity, where dance movements *follow* the musical sound. Using dance movements to *generate* musical sound may therefore result in a perceptual conflict, since the movements of the dancer happen before the musical sound appears. This may not only be novel for perceivers, but I have also experienced that many dancers have problems generating sounds themselves. This is probably because they often use sound as an inspiration for their movements, and use cues in the music as triggers for various sequences or patterns. Having to control the sound themselves, they have to start thinking about both their movements and the sonic result of their movements, which means having to start thinking like a musician.

This "conflict" between dance and music was used creatively in *Whiteness I Remember*. Here we had a very direct and immediate relationship between movements and sound, so the dancer had to learn to physically stop and keep standing still to give sonic space for the singer. Similarly, the singer would be able to adjust her improvisation to the fit the rhythms of the movements of the dancer. As such, it turned out to be a very interesting collaboration, where the three of us involved all had to push our ideas about music, movement and improvisation.

7.5.3 Polhemus Tools

The applications presented above have all been video-based, but MGT is more general in scope. Figure 7.31 shows the interface of a patch made to visualise and analyse the data retrieved from a *Polhemus Liberty* electromagnetic tracker.[40] This is a high speed (240 Hz) and high accuracy (in the range of millimetres) motion capture device with cabled sensors each having 6 degrees of freedom (x, y, z, azimuth, elevation, roll). Polhemus trackers are efficient motion capture devices, since they return data from the sensors without having to carry out any pre-processing to find the location of the sensors. This is opposed to camera-based motion capture devices (for example Vicon systems), where markers have to be identified semi-automatically after recording sessions. The downside to using a Polhemus tracker is the long and heavy cables attached to its sensors,[41] and the fairly short sensing distance (approximately 3 meters with the standard set).

The original Polhemus software is limited to only recording and visualising the position of the sensors. It was therefore necessary to develop our own software to carry out realtime analysis. Unfortunately, the Polhemus trackers only ship with drivers for Windows XP, and I wanted to use the device in OS X. To solve this problem, Julien Boissinot wrote a small program in the *Python* programming language, running on a Linux computer, which could retrieve data from the device. This program would format and pass on the data as OSC streams on the network (using multicast UDP for communication), and allowed for receiving the values in Max using regular networking objects. The benefit of this distributed system was that multiple computers could receive data from the Polhemus tracker simultaneously, and carry out different types of analysis in parallel. This proved useful in a collaborative study testing body movement control of sound spatialisation, which was presented in Marshall et al. (2006). In this project Mark Marshall improved and implemented various types of movement analyses in the Python script, and these

[40] http://www.polhemus.com

[41] This is improved in the *Polhemus Liberty Latus* tracker which sports wireless sensors.

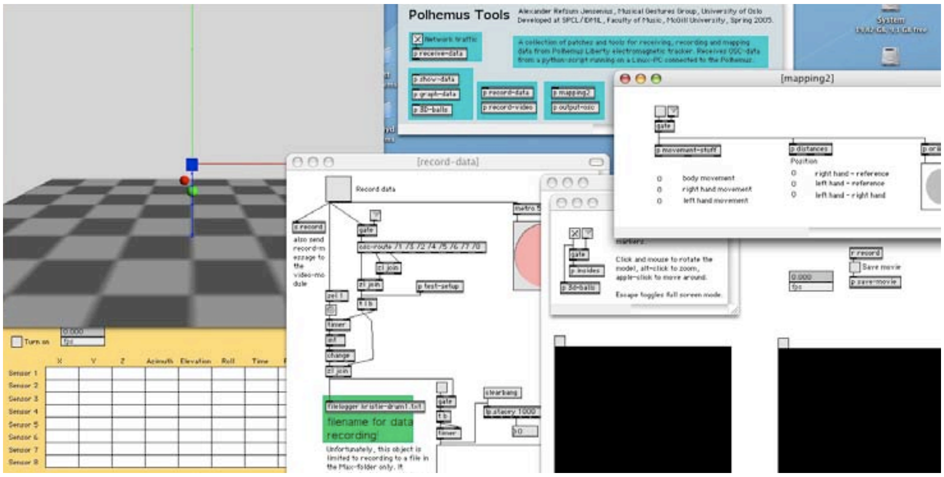


Figure 7.31: *Screenshot of a patch built for analysing data from a Polhemus electromagnetic tracker, as well as audio and video.*

data were then analysed further in my Polhemus patch, and also used by Nils Peters to generate sound and control sound spatialisation in realtime.

The intention of creating the Polhemus patch was to create one solution to work with motion capture data, MIDI, audio and video at the same time. A major challenge here is that these all rely on different software and storage solutions, and use different types of ranges, sample rates, etc. As may be seen in Figure 7.31, the patch has a 3D display visualising the location of the sensors, video modules for analysing the movements of the user, and MIDI and audio modules for working with sound. The end result was a prototype software tool that incorporated a number of the different solutions needed to study relationships between movement and sound. That was also its greatest problem, since the patch turned out to be so computationally demanding that the computer would run out of memory after only a couple of minutes of recording data. Still, the patch worked as a proof of concept, and also revealed several problems that will have to be addressed in future development.

7.6 Discussion

Throughout the development of MGT and Jamoma, I have discovered a number of challenges that need to be tackled. One of the most important ones is *standardisation*. Working with so many different types of devices and data makes it necessary to create standards for how messages should be communicated and stored. This will be discussed more in Chapter 9. Here I will focus on two issues that are related more specifically to the further development of Jamoma.

7.6.1 Units

The current implementation of Jamoma does not have a solution for handling *units*, such as meters, decibels, MIDI, etc. This is also a problem for Max in general, since there is no systematic approach to how units are defined and used. For this reason, I have argued for a *unit library* (UnitLib), which we will implement in the next version of Jamoma. The idea is to include information about units as attributes for all parameters and messages defined in the module files, and use the UnitLib to handle conversions between data sets using different units. Ideally, the UnitLib will facilitate using values of any unit type to control any parameter in any module. For example, the gain slider in an audio module could be controlled with data formatted in meters, MIDI, percentage, etc. Obviously there are several challenges related to this, which we have discussed at length in the Jamoma developer community.

The first challenge is related to the fact that changing units will also involve changing the other properties of the values being sent. For example, imagine changing the messaging of a gain control set to receive MIDI values (0–127), to receive values using a decimal range (0.–1.). The conversion between these two ranges is easy to carry out, but changing only the communicated value is not enough. The MIDI data range is typically set to clip at 0 and 127, so we will have to implement a solution to also change the ranges and their clip values when changing the units.

Another problem occurs if there is no linear relationship between the two unit types in question. Taking the example of the audio gain control, imagine changing from using a dB scale to a decimal range (0.–1.). This involves converting from a logarithmic to a linear scale, and may potentially lead to several problems in implementation, not only in the translation between the values, but also the values of the ranges and clip values.

A third problem is connected to the use of *ramps*, or interpolation between values. From Jamoma 0.4.2, a ramping library (RampLib) facilitates creating various types of ramps between values. This is a powerful feature, which may be used to create various types of dynamic effects in patches. However, since ramps are based on the ranges defined in the parameters, there might be some problems with active ramps if the ranges change while the ramp is running. This has to be taken into account when implementing the UnitLib.

Implementing units in Jamoma also poses several challenges in OSC messaging. For example, should the units be sent together with these values?

```
/module/parameter 10 dB
```

Or should they be set using a separate message?

```
/module/parameter/unit/set dB
/module/parameter 10
```

The former is probably better in terms of clarity and human readability. It would also make the system more robust in network situations, where packets might be lost using UDP for communication. However, sending the units with the messages may be seen as redundant, and would lead to more traffic and processing load. The question is also how this should be handled if there are multiple values in one string, e.g.:

```
/module/parameter 10 20 50 40 20 dB
```

Would that message mean that they all are using the same unit (dB)? And what would happen if several different units were sent as one string? For example:

```
/module/parameter 10 dB 20 MIDI 50 BARK 40 MIDI 20 dB
```

This would make for some rather complex OSC messages and may cause problems when parsing them. These are still open questions that will have to be addressed in the future development of Jamoma.

7.6.2 Control and Engine

An important concept in Jamoma is that of a separation between *control* and *engine*; or *module* and *algorithm*. The idea is that the user interface should be independent from the algorithm carrying out the processing, a technique which is common in various software on different platforms. The current Jamoma implementation reflects this to some extent, since the algorithm is actually stored in a file separate from the module. This makes it possible to use the algorithm without the interface, but since the algorithm is loaded inside the module file it is not possible to use the interface without the algorithm.

In a truly separated system, it would be possible to run the algorithms and interfaces independently. This is the approach taken in the *Integra* project,[42] which I have also been involved in over recent years. The idea here is that interfaces and algorithms should be implemented and run separately, and that they would communicate using OSC messages. This will facilitate having multiple interfaces controlling one algorithm, such as illustrated in Figure 7.32.

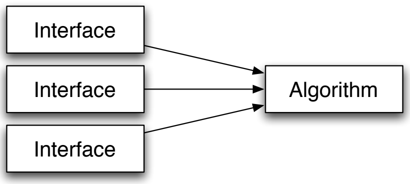


Figure 7.32: *Many-to-one mapping between the graphical user interface and the algorithm.*

The ability to use different types of interfaces to control the same algorithm is practical for many reasons. First, the looks of the Jamoma modules may not work aesthetically in all patches. For example, in the Polhemus patches I decided to hide all the modules since they would otherwise take up too much space. Another reason for having multiple interfaces is that the algorithms may be used in very different contexts, requiring different types of control. For example, when analysing videos from an observation study, I prefer

[42] http://www.integralive.org

to have an interface that gives access to lots of functionalities. This, however, is not very practical in a performance situation where I usually need a large and clear display and minimal direct control of technical parameters. In the current implementation this can be accomplished by hiding the module, and building another interface referencing that module. However, in Jamoma the modules also define the functionality, so it is not that easy to use the algorithms in different ways when they have to be controlled through a defined module. Splitting the interface from the algorithm would probably be a more elegant solution to this problem.

Similarly, separating the interface and the algorithm would also facilitate using one interface to control several different algorithms, as shown in Figure 7.33. This could be practical when using multiple algorithms that have the same functionality, but which are implemented differently. For example, most reverb algorithms work more or less the same way, but the parameters used to control them often have different names and use different ranges. In an ideal system it should be possible to have one reverb interface, and be able to use this with any type of reverb algorithm. This would be possible if the algorithms understood the same type of control messages, and the interfaces were made to communicate with them accordingly. An important topic here is that of creating *meaningful namespaces*, something which will be discussed more in Chapter 9.

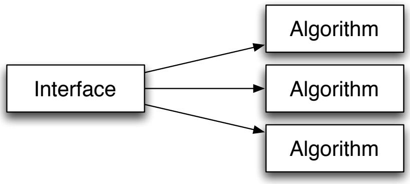


Figure 7.33: *One-to-many mapping between GUI and module.*

7.7 Summary

This chapter has presented MGT and its development from a handful of patches to a large collection of modules and components merged into the open source project *Jamoma*. Working on MGT has been much more than only developing software. Rather, development has also been an exploration of analytical methods with which to study music-related movement and musical sound. MGT is the answer to many of the challenges encountered during the project, solving many of the problems that other tools could not.

While we may sometimes be fooled to believe otherwise, technology is never neutral. Technology, both hardware and software, is always developed to solve certain problems. By testing various types of software intended for sports, medical and behavioural applications, we quickly realised that we needed tools developed by and for music researchers. Only music researchers know what is interesting from a musical perspective, and thus music researchers should develop software that can solve their needs. That is what I have done.

In MGT I have found it particularly important to implement tools for both quantitative and qualitative analyses. Quantitative data are useful to provide an overview of the material, and to look at large scale relationships. Here MGT can output data that can be used for numerical analysis in for example Matlab. However, the complexity of our material (both movements and musical sound) calls for a qualitative approach to study differences and similarities. For this reason, MGT can also be used to create various types of visual representations, both in realtime and for non-realtime analysis.

The idea of simplicity in design was discussed in the context of controllers in Chapter 6. This has also been an underlying idea in my development of MGT. One focus has been on developing self-contained modules that facilitate the quick and easy assembly of rather complex patches. I have also tried to work towards standardising the messages passed between modules, to allow for a more streamlined development process. Hopefully we will be able to create a clearer separation between interface and algorithm in future versions of Jamoma, so that it is possible to use multiple interfaces with the same algorithm, and multiple algorithms with one interface. This, however, requires a much more uniform way of passing information around in the patches, something which will be discussed more in Chapter 9.

One feature I am particularly satisfied with in MGT is the click-and-drag control of parameters implemented in some of the modules (**jmod.orsize%** and **jmod.brcosa%**). I rarely use the number boxes to control these parameters anymore, and the question is whether these number boxes have to be there at all. I will continue to explore this type of interface simplification in the future development of MGT.

This chapter has been fairly technical and focusing on implementation and some specific features of MGT. The next chapter will look at some motion visualisation techniques that I have developed during the project, and that are also implemented in MGT. They show more clearly how MGT may be used to analyse music-related movement.

CHAPTER 8 Movement Visualisation

If you take a photograph of something [...] you separate it from the rest of the world and you say, "This deserves special attention".

Brian Eno (Kalbacher, 1982)

This chapter starts with an overview of some movement and video visualisation techniques developed over the past centuries. Then follows a presentation of my exploration of various techniques for visualising movements from the videos of our observation studies, including *keyframe displays*, *motion history images*, *motiongrams* and *videograms*. Finally, examples are given of how motiongrams and videograms have been applied to the analysis of various types of video material.

8.1 Introduction

One of the key challenges when working with music-related movement is to create suitable displays and visualisations that can be used for analysis and documentation. The problem is that movements unfold in time and space, and are "gone" the moment they are finished. As argued in previous chapters, video recordings are efficient for presenting movements in time, but they are not very efficient when it comes to analysing, comparing and representing the movements. Taking a picture is one possible visualisation technique, but a picture will only display a *posture* and not the *movement* of a movement sequence. In some cases using a static picture may work, but in general I find that it would be better to have displays that more effectively show the continuous stream of movement.

Ideally, we should have visualisation techniques that simulate our perception and cognition of movements. As discussed previously, we tend to see and remember movements as goal-directed action units. For example, watching someone reaching for a glass is usually perceived as a coherent unit consisting of a *goal* (the glass), a *trajectory* (the path of the hand), and the *effort* involved. Not only are we able to identify and chunk such continuous movements as they happen, but we are even capable of "seeing" the whole action unit before it is finished. The challenge, then, is to be able to create computer displays that "simulate" such mental imagery, and that can guide our analysis of the movements. Creating such movement visualisation techniques is the topic of this chapter, and I shall start with an overview of various movement visualisation techniques.

8.2 Overview of Movement Visualisation

Looking back at the history of photography, it is interesting to note how the exploration of this new medium was flourishing by the end of the 19th century. At that time many of the most important inventions were done by researchers of anatomy who worked on techniques for visualising the movement of animals and people. Through this research they also happened to develop cameras, films and photographic techniques that were the predecessors of modern-day cinema.

8.2.1 Timelapse Photographs

British-American photographer and movement analyst Edward James Muybridge (1830-1904) pioneered early photography with pictures of the *locomotion*, i.e. the ability to move from one place to another, of animals. Of his many interesting observations, he is probably best known for a series of horse pictures from the late 1870s where he effectively demonstrated that at certain moments in time a horse has no legs on the ground while galloping. These *timelapse photographs* were taken by aligning a series of high-speed cameras (1/500 second shutter speed) on a race track. Electric wires attached to the camera shutters were suspended over the field, and the horse passing by the cameras would release the shutters when stepping on the wires (Mozley, 1972). The end result was a series of photographs shot in rapid succession that effectively showed snapshots of the movements of the horse.

One problem with Muybridge's "step wire" technique was that the temporal aspect of the movements was not reflected in the sequence, since each photograph was triggered by the horse (or other animal) stepping on the wires. This was not a problem as long as the animal moved evenly, but to study movements that were not so evenly distributed in time, it was necessary to develop other techniques. Muybridge therefore invented a clockwork mechanism to activate the cameras, which secured the photographs being shot at an equal time interval (Prodger, 2003, 188). An example of a timelapse photograph shot with such a clockwork mechanism is shown in the image sequence of a woman walking down steps in Figure 8.1. Here it is possible to get an idea of the movement of the person, even though this timelapse photograph is only a series of still pictures.

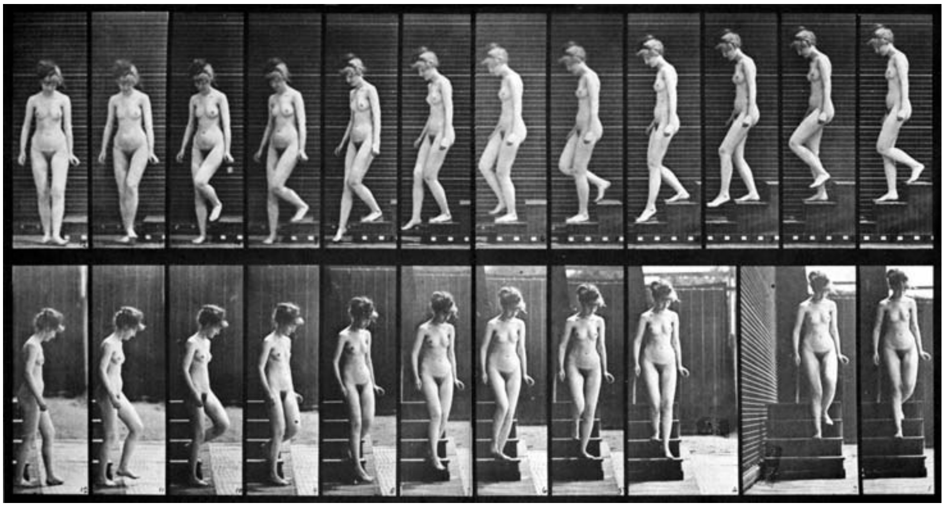


Figure 8.1: Woman walking down steps, *a* timelapse photograph *by Muybridge (1887).*

8.2.2 Chronophotography

Around the same time as Muybridge was carrying out his photographic experiments in California, another type of photographic exploration was going on in Paris under the direction of Étienne-Jules Marey (also 1830-1904). He developed different types of what he called *chronophotography*, or "pictures of time". One such technique is *strobophotography*, which is based on creating multiple exposures of a moving subject on the same photographic plate. An example of strobophotography is shown in the picture of a flying pelican in Figure 8.2 (Braun, 1992). Here it is possible to get an idea about both the temporal and spatial characteristics of the movements, as opposed to Muybridge's photographs which mainly represent temporal aspects.

Marey also developed techniques to reduce the display of the subject in the photographs. An example of this is the picture of a "walking stick man" in Figure 8.3. This photograph was taken with multiple exposures of a walking subject wearing a black suit with small reflective markers attached to the joints (Blake and Shiffrar, 2007). As for the pelican photograph, this picture also shows both temporal and spatial characteristics of the movement, and here it is also possible to clearly see the continuous stream of movement of the person walking.

8.2.3 Point Light Displays

Almost 100 years after Marey's experimentation with strobophotography, Swedish psychologist Gunnar Johansson used a similar technique in his *point light displays*. Johansson used reflective tape on the main joints of the body, and a special video recording technique to capture only the markers (Johansson, 1973). The idea was to study the per-

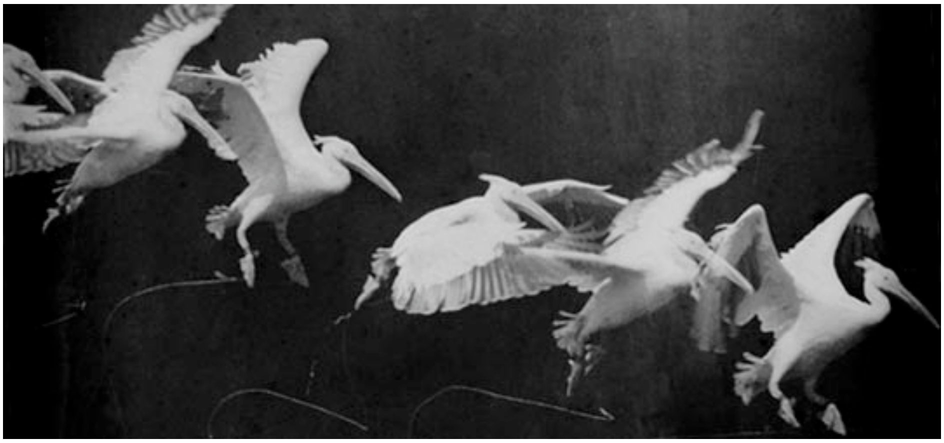


Figure 8.2: Vol du pélican*, an example of* strobophotography *by Marey (1882). Notice how both temporal and spatial aspects of the movement are presented in the photograph.*

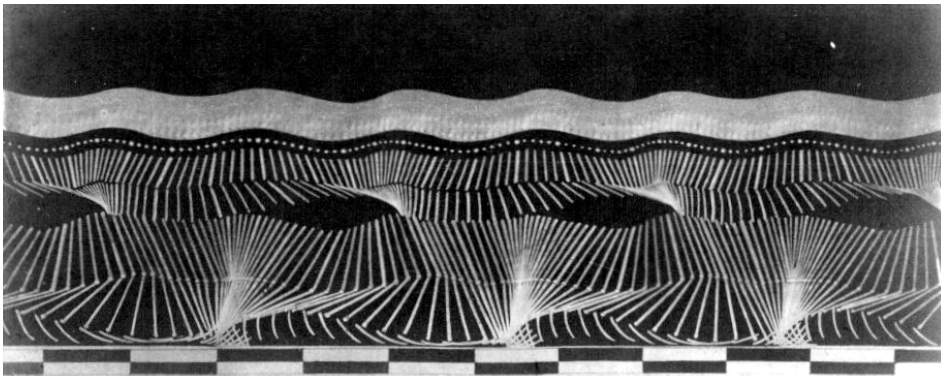


Figure 8.3: *An example of* chronophotography *of a walking man, captured by Marey (1884). This was based on adding reflective markers to the body of a person walking.*

ception of reduced kinematic models, and Johansson showed that people could efficiently recognise various movement features from the point light displays only. The point light display technique has later been used in both psychological and behavioural research, although it is now more common to create the recordings with *motion capture systems* using infrared cameras. Such systems are often based on multiple cameras surrounding a person and recording the light reflected from markers on the body. The final result is a three-dimensional display with a high level of both temporal and spatial detail.

During my project, I have had the chance to work with a Vicon[1] System 460 optical tracker and M2 cameras in the *Input Devices and Music Interaction Laboratory* (IDMIL)[2] at McGill University. Figure 8.4 shows a screenshot from the Vicon Workstation software, in which it is possible to play back the motion capture recording and watch the movements from different angles. This example is from a motion capture session of myself playing the piano, and uses a five second "movement window" to represent movement trajectories. As such, this display can be seen as a visualisation somehow similar to the way we perceive such a continuous movement sequence.

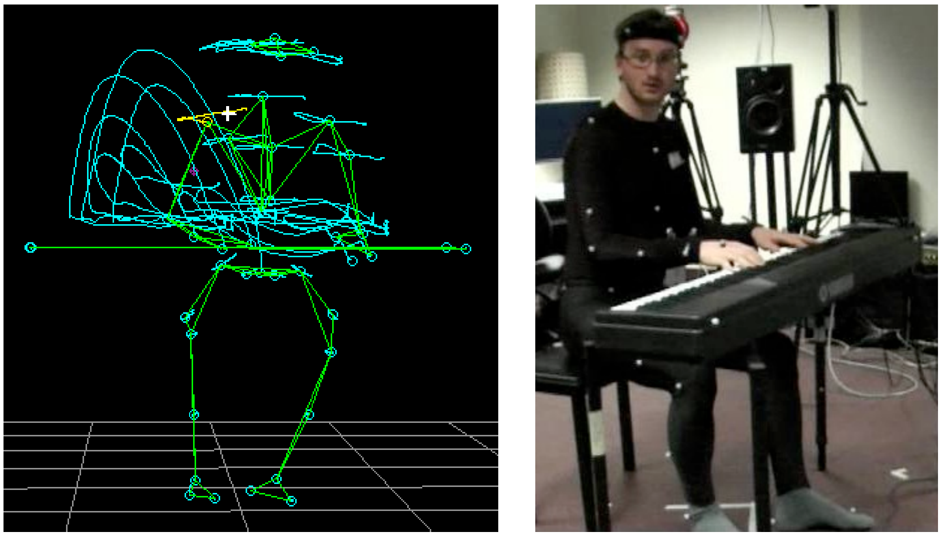


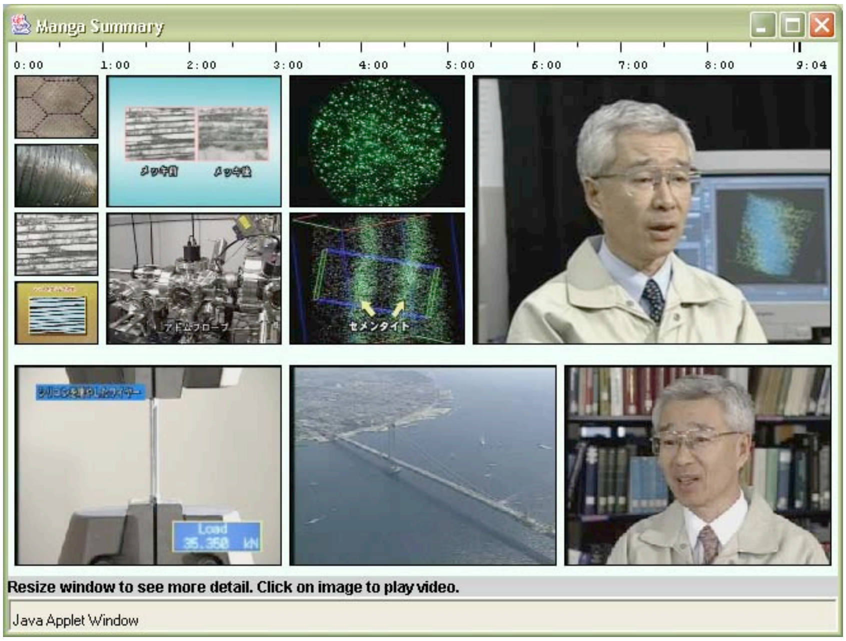
Figure 8.4: *A picture from a motion capture session of me playing the piano, with reflective markers attached to the body and to the instrument (right). The screenshot from the visualisation software (left) shows the piano as a line in the middle of the image, and the reflective markers used in the motion capture are shown as circles. A biomechanical model is applied to show an outline of the body, and a 5-second preview buffer is used to visualise the movement trajectories of the markers. This movement sequence displays a long run from low to high register on the piano, and a final lift and return to start position.*

[1] http://www.vicon.com
[2] http://www.music.mcgill.ca/musictech/idmil/

8.2.4 Video Mangas

The above mentioned techniques are all focusing on creating movement displays. Also interesting for my project are various techniques focusing on creating *video abstractions*, or "summaries" of a video file. This has become a large research field in recent years, due to the demand for searching and navigating large collections of digital video. This follows from the rapid growth in the use of digital video on various media (including DVDs) and as digital computer files. With an increasing amount of such video material there is a need to create solutions to find certain material without having to watch all the videos in person.

One approach to video navigation is called *video mangas*, named after the Japanese word for "comic book" (Uchihashi et al., 1999). Figure 8.5 shows an example of such a video manga, where the *keyframe images* extracted from the digital video file are organised in a comic book-style *keyframe display*. In the following discussion I will use *keyframe image* to refer to single, salient images extracted from a video sequence. A *keyframe display* I will use to denote the collection of multiple keyframe images in one visual display.



Resize window to see more detail. Click on image to play video.

Java Applet Window

Figure 8.5: *A video manga from (Girgensohn, 2003), summarising the content of a video file. The keyframe images are extracted from a digital video file and rated according to the "perceptual relevance" of the sequence they are representing. This rating is used to decide the size of the keyframe image in the keyframe display. The result is a video summary which can be used to navigate in the video file.*

The implementation of video mangas is based on extracting "salient" keyframe images from the video, by creating *self-similarity matrices* based on colour features of the video frames. Self-similarity matrices are an efficient tool for detecting recurring patterns in any type of material, and are used here to find similar *segments* in the videos. This segmentation is used to evaluate the "importance" of each keyframe image, and is used to find the size of the keyframe in the video manga (Girgensohn, 2003). Video mangas have also been extended by adding verbal descriptors such as keywords and salient phrases (Ding et al., 1999), and audio snapshots from the segmented video frames (Foote et al., 2002). The end result is a video summary that simplifies finding specific parts or scenes in video files.

8.2.5 Keyframe Trees

Similar video segmentation techniques have been used in the development of systems for automatically creating *keyframe trees* (Figure 8.6), showing the relationship between segments of a video file (Girgensohn et al., 2001). Here the user will have access to a hierarchical structure in which it is possible to see different levels of details of a video sequence. Presumably, this type of display will work better with video material containing clearly defined shots and sequences, and will probably not work so well with our studio recordings. That is because both the process of extracting the keyframe images, and the way these images are organised and displayed, focus on features of the *images* of the video and not the *movement* within the images.

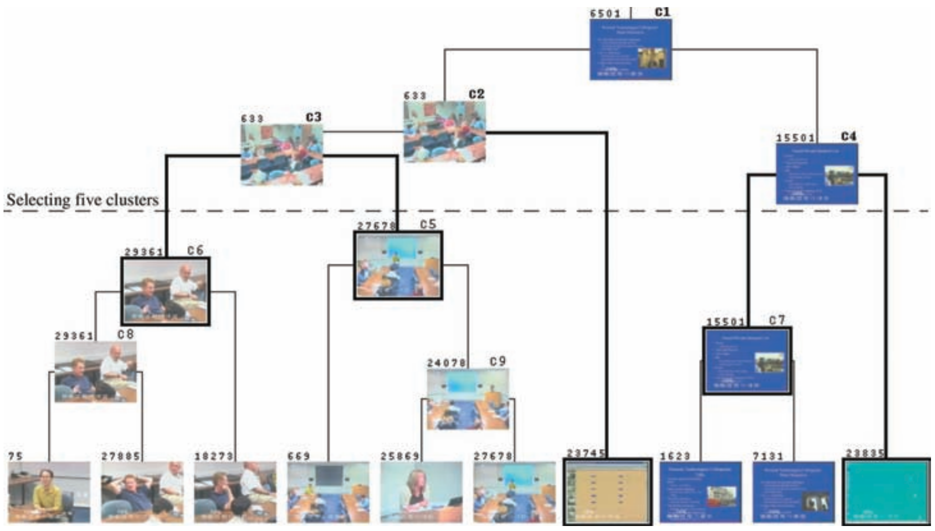


Figure 8.6: *A* keyframe tree *from (Girgensohn et al., 2001), based on grouping video segments according to colour similarity. This makes it possible to see the hierarchical structure of the content of a video file.*

8.2.6 Storyboarding

An example of a video abstraction technique that focuses on movement within the video images is that of automatic *storyboarding*. Storyboards are used in TV and film production to describe scenes and shots, and how actors and objects should move within the shots. Traditionally, storyboards have been created by professional storyboard artists, based on descriptions from the director and manuscript writer. Trying to reverse the process, Niikura et al. (1999) suggested a method to automatically create storyboards from video files, and this has later been developed into semi-automatic software by Goldman et al. (2006).

Figure 8.7 shows an example of a semi-automatically generated storyboard. This was created by manually choosing four keyframe images from a movie sequence, and selecting a few reference points to identify the location of common objects in the keyframe images. These four images and the selected points are then used to automatically generate a composite image showing the person moving through the frame, as well as arrows indicating the movement of the person. As can be seen in the figure, the semi-automatically generated storyboard is fairly similar to the storyboard created by a professional storyboard artist. What I find the most interesting about this semi-automatic storyboarding technique, is that the algorithm effectively manages to capture salient features about the movement, and create a display which shows the location and movement of a person moving inside the frames.

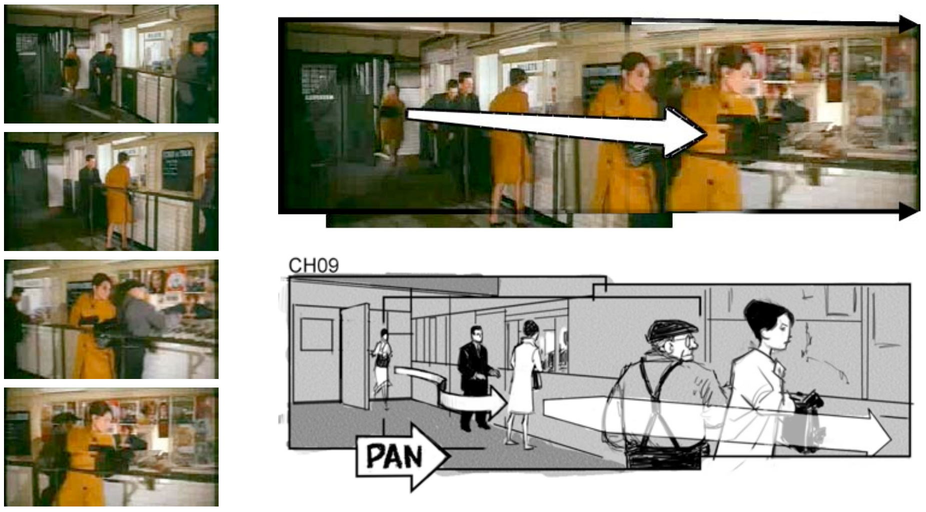


Figure 8.7: *A schematic* storyboard *from (Goldman et al., 2006), based on a video sequence from the film* Charade. *The four manually selected keyframes that were the basis for the automatic storyboard are shown to the left. The automatically generated storyboard is shown at top right, and a storyboard created by a professional storyboard designer is shown at bottom right.*

8.3 Keyframe Displays

Inspired by the approaches mentioned above, I have explored different methods for visualising the recordings of our observation studies, starting with creating keyframe displays. A keyframe display may represent movement, but only if the time between each captured frame is not too long. As such, I do not think that keyframe displays are particularly useful for displaying longer movement sequences, either because the display itself gets too large or because the time between each image gets too long.

An example of a keyframe display that does not work so well can be seen in Figure 8.8. This is a time-based keyframe display of the first 32 seconds of one of the recordings from our free dance studies (Video 7.1). The display makes it possible to imagine the movement happening in the sequence, but I do not find this display to be particularly efficient. That is because the frames are sampled at an interval of 2 seconds, something which is too much even for the slow movements in this sequence. Also, to be able to create a fairly small display, it was necessary to make each keyframe image very small. This again reduces the readability of the final display.

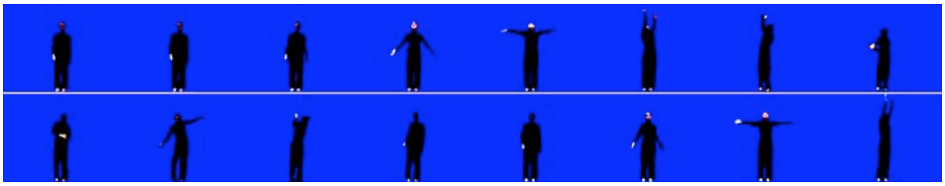


Figure 8.8: *A keyframe display of the first 32 seconds of Video 7.1 from our free dance observation studies. Such a keyframe display shows some information about the movements of the dancer, but several of the frames contain little perceptually relevant information.*

There are several possible solutions to improve keyframe displays such as the one shown above. One is to use larger keyframe images in the display, but this will also increase the overall size of the keyframe display. Or, if the display size was given (for example in a printed document), it would have been necessary to reduce the number of keyframes in the display. Both of these alternatives would also reduce the readability of the display, either because it would get too large, or because there would be too few keyframes. The latter would probably also decrease the chance of sampling a perceptually salient frame, and we might end up with a keyframe display with little relevance to the actual movement. A better solution would then be to sample frames based on their "perceptual salience".

8.3.1 Salient Keyframe Displays

One way to improve keyframe displays is to remove the frames that contain less perceptually relevant information. This technique has been referred to as *salient stills* by Teodosio and Bender (1993), but here I will use the term *salient keyframe display*. My approach to creating salient keyframe displays is based on finding frames in the continuous video stream that have a high level of difference from the neighbouring frames. This would imply that there is a rapid change in movement between these frames, which may again be correlated to perceptual salience.

An example of a salient keyframe display generated from the same free dance video as used above (Video 7.1), is shown in Figure 8.9. Since this keyframe display is based on the change of level of movement, each image will be different from the neighbouring images. In many ways such a salient keyframe display is more interesting than the time-sampled keyframe displays shown in Figure 8.8, since the salient keyframe display reveals more about the qualities of the movement in the frames. The salient keyframe display also makes it possible to visualise longer movement sequences (here approximately 70 seconds), since redundant frames are removed.

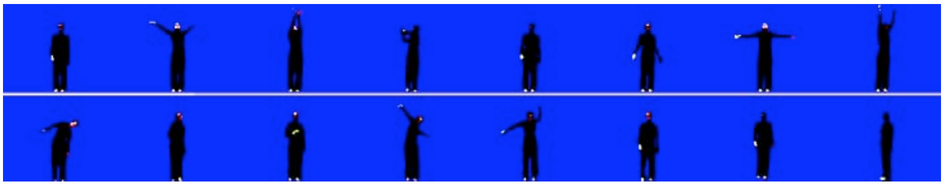


Figure 8.9: *A salient keyframe display from the beginning of Video 7.1. Such a display contains more perceptually relevant images than a time-based keyframe display, but there is no indication of the temporal relationship between the images.*

The salient keyframe display shown above was created with the patch **keyframe-display.mxt**. The implementation is based on calculating the QoM using a combination of the **jcom.motion%** and **jcom.sum%** components, and sending this into the segmentation patch shown in Figure 8.10. Here the first derivative (the difference between subsequent values) of the running QoM values is calculated using **jcom.delta** (implemented by Trond Lossius). These values are then normalised to a 0.–1. decimal range with **jmod.autoscale** using a running time window of 100 samples (i.e. video frames). This results in the values always being scaled relative to the surrounding values, something which ensures that the following threshold algorithms work for sequences with both high and low changes in QoM. For example, a large movement in a sequence with a high QoM may be perceptually similar to a comparably smaller movement in a sequence with a low QoM. The autoscaling function effectively levels out such context differences over time. Finally, the values are sent through two threshold functions, one removing all values that fall below 75% of the normalised values, and one removing values that are closer than 500 milliseconds apart. The output of this segmentation patch is a message sent every time the movement changes above the given threshold values. This message is then used to save and display the corresponding keyframe image in the main patch.

In many ways salient keyframe displays are better at visualising movement sequences than time-based keyframe displays. The biggest problem with salient keyframe displays, however, is that the temporal dimension is lost. We do get to see the most important frames, but there is no indication of how close or far apart they are in time. This could

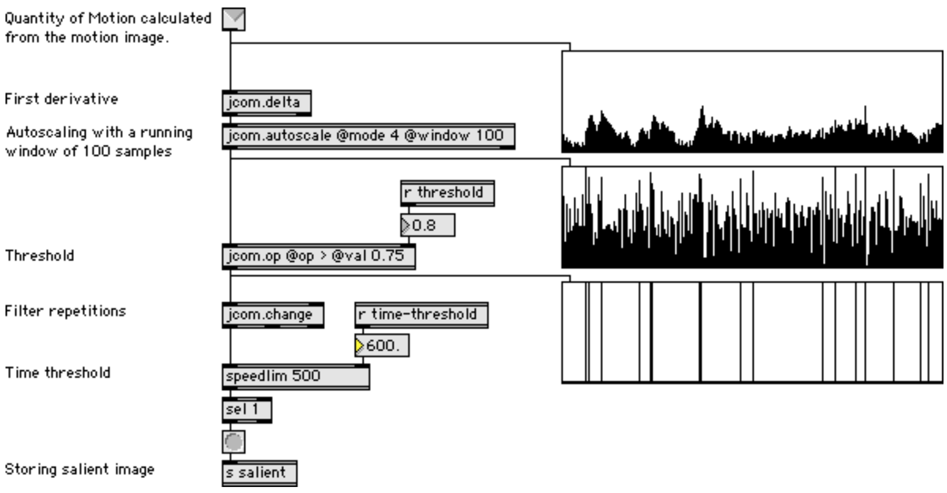


Figure 8.10: *The segmentation algorithm used in the* ***keyframe-display.mxt*** *patch. The first derivative is calculated from the running QoM values coming in at the top. Then the values are normalised with a running time window, and sent through two threshold functions, before the "salient" frames are selected.*

have been solved by displaying the frames using weighted keyframe sizes similar to the video mangas, but I realised that it was probably not worth the effort. I find keyframe displays to be fundamentally problematic due to their inability to represent the movement occurring *inside* the frames. So rather than continuing creating keyframe displays, I decided to explore other methods for visualising movement.

8.3.2 Motion History Images

Inspired by Marey's use of multi-exposure photographs, I started making images based on an "open shutter" technique. This was implemented by calculating the running average of whole video sequences, using **cv.jit.mean** by Jean-Marc Pelletier.[3] An example of this is shown in Figure 8.11, depicting the whole 5-minute free dance recording in Video 7.1. This display reveals some information about the spatial aspects of the sequence, for example that the dancer was standing more or less on the same spot throughout the whole session, but it tells little about the qualities and temporal development of the movements.

A better approach is to create *motion history images* based on a running "time window". A number of different techniques have been developed for creating such motion history images, for example *motion-energy images* (MEI) (Bobick and Davis, 2001), *timed motion-history images* (tMHI) (Bradsky and Davis, 2002) and *Silhouette Motion Images* (SMI) (Camurri et al., 2003). I have experimented with different approaches to

[3]http://www.iamas.ac.jp/ jovan02/cv/

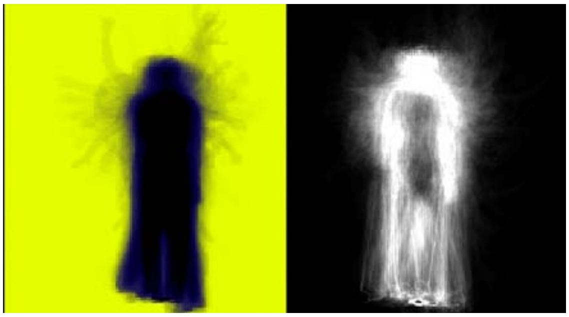


Figure 8.11: *An "open shutter" image of the 5-minute recording of free dance in Video 7.1. The left image shows the inverse image of the average of the regular video, while the right image shows the average of the motion image of the same video.*

creating such motion history images, seven of which can be seen in Video 8.1 and Figure 8.12. These motion history images are based on various types of video feedback algorithms and are implemented in the *motion-history-images1.mxt* patch. Notice how the motion history images effectively visualise some of the movement trajectories of the dancer. Interestingly, these coloured trajectories (red and yellow lines) are not a result of the algorithms in use, but based on video feedback of the red and yellow socks and gloves worn by the dancer during the recording session.

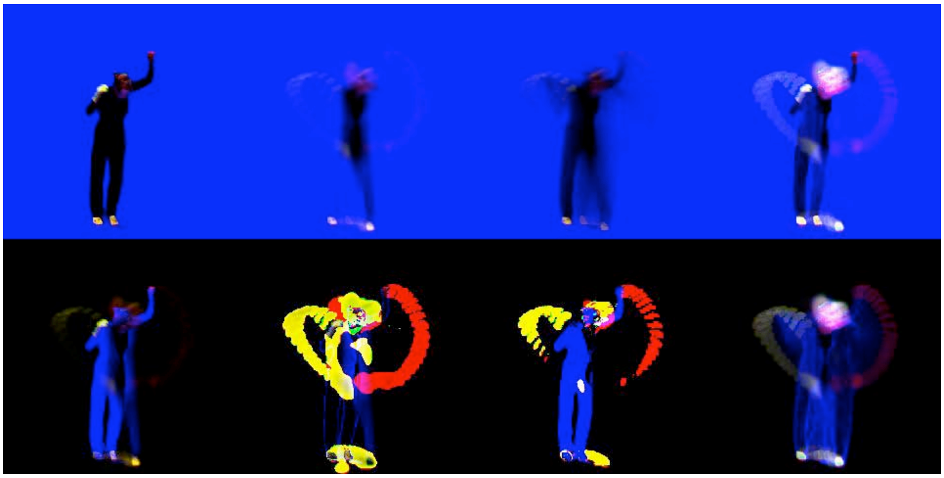


Figure 8.12: *A screenshot from Video 8.1, showing seven different types of* motion history images *all based on the original video presented in the top left corner.*

A different type of algorithm for creating motion history images is implemented in the patch *motion-history-images2.mxt*, and a snapshot of the output is shown in Figure 8.13. These displays are based on time delaying the input frames and shifting the delayed frames spatially, something which gives a "waterfall" type of display. To make the foreground more visible I have added the edge around the dancer. The resulting displays show spatio-temporal evolution of the movement sequence, but I find them less clear to read than the displays in Figure 8.12.

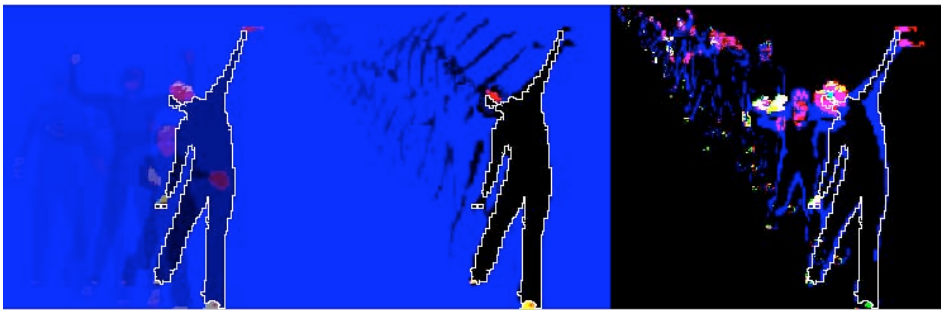


Figure 8.13: *Various motion history images based on time delaying and spatially shifting the images. These displays resemble "waterfall" displays, and give more spatio-temporal information.*

A challenge when creating motion history images is to find a suitable time window for the "motion history", something which can be seen as analogous to the problems of choosing a time window in audio analysis. If the history is too long, the image will often become blurred and difficult to interpret. This is particularly problematic if there is a lot of movement happening in the sequence. Similarly, a too short motion history may result in no visible effect if the movement in the sequence is slow.

One solution to the problem of choosing a duration of the motion history, is to dynamically change the duration over time. I implemented this by using QoM to control the duration of the time window. The algorithm worked fine, but the resultant images were difficult to work with since the temporal information was lost. This was because the trails of a slow movement ended up similar to the trails of a fast movement. My current solution is therefore to work with multiple motion history images at the same time, such as those shown in Figure 8.12. This facilitates looking at different displays dependent on the character of the movements. I also find that having multiple displays seem analogous to the way we humans see and think about movements, since we are capable of simultaneously analysing movements at various temporal resolutions.

Motion history images may be used to represent movement over shorter periods of time, usually up to around 10 seconds. Extending the motion history beyond this will result in images that are too blurred. One solution to represent longer movement sequences may be to create keyframe displays from motion history images. As argued above, one of the problems with keyframe displays is that they do not visualise the movement very well. However, using motion history images as the basis for keyframe displays will allow for both visualising movement, as well as representing changes in movement over longer periods of time. Two examples of *motion history keyframe displays* are shown in Figure 8.14. Notice how much more movement information is visible in these displays than in the keyframe displays shown in Figures 8.8 and 8.9. Rather than only showing the static postures of the dancer, it is possible here to get an idea of how much she is moving, as well as the trajectories of the movements.

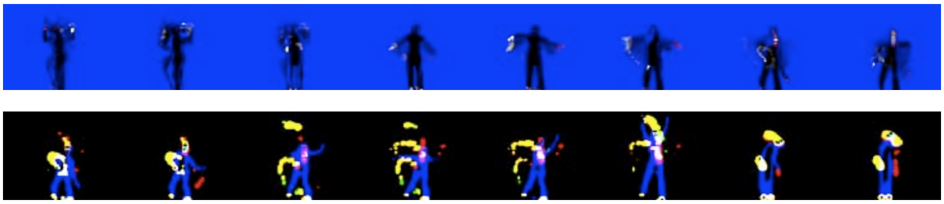


Figure 8.14: *Two different motion history keyframe displays, based on the regular image (top) and one based on the motion image (bottom). Each motion history image represents 5 seconds of movement. Such motion history keyframe displays are capable of showing movement inside frames, as well as relationships between frames over longer periods of time (here about 1 minute).*

Motion history keyframe displays are perhaps closer to our mental images of movements than the other types of displays presented above. One problem, however, is that such displays are not compact enough to represent longer sequences of movements. Figure 8.14 only displays about 40 seconds of the video, and creating displays with more images may be visually confusing. For this reason I started thinking about creating displays that could visualise longer sequences of movement.

8.4 Motiongrams

In the audio domain we are used to working with waveform displays and spectrograms. These are only rough displays of the original sound, but are still efficient in visualising some important features of audio material. I therefore started thinking about how it would be possible to create a "video spectrogram" for visualising longer sequences of movement, anything from a few minutes to several hours. One approach is to plot the QoM, as shown in Figure 5.3 on page 67. However, the QoM is only a rough estimation of the *quantity* of movement, and does not tell anything about the *location* of the movement inside the frame. This makes it difficult to know where the movement happened in the frame, for example whether movement was happening in the head, torso or feet. A better solution would be to create a display that could visualise the QoM as well as the distribution of QoM in time and space. Inspired by some of Marey's photographs, I stumbled upon a technique for creating displays of "collapsed" video frames. The resulting visual displays is what I have called *motiongrams*.

8.4.1 Process and Implementation

An overview of the process of creating a motiongram is shown in Figure 8.15. The process is based on reducing the matrix of a motion image to a 1 pixel wide or tall matrix which is plotted over time. More specifically, a matrix of size *MxN* is reduced to 1xN and Mx1 matrices by calculating the mean value for the column and row respectively. Thus for each frame of a video file, a 1xN or Mx1 matrix is calculated. Drawing these 1 pixel wide (or tall) stripes next to each other over time, results in either horizontal or vertical displays of the "collapsed" motion images. These running displays are motiongrams that make it possible to see both the location and level of movement of a video sequence over time.

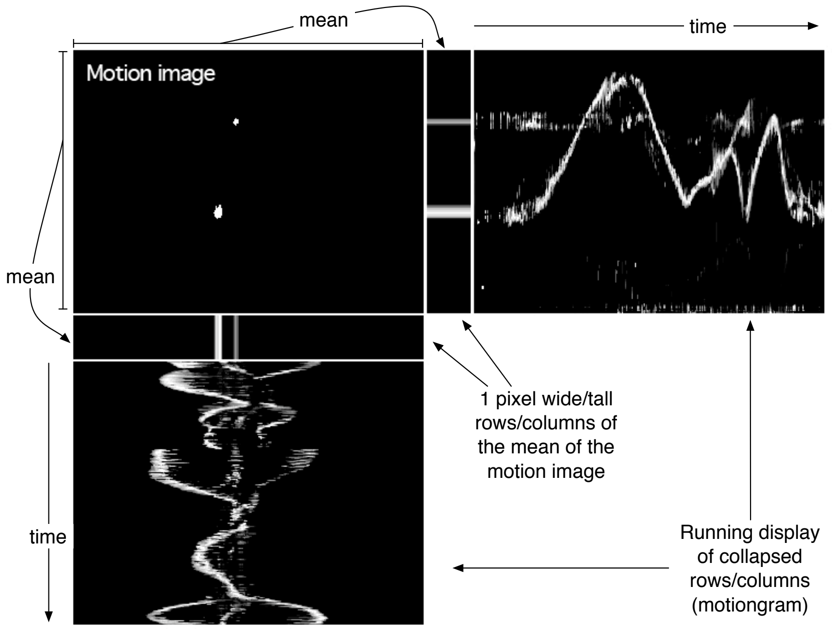
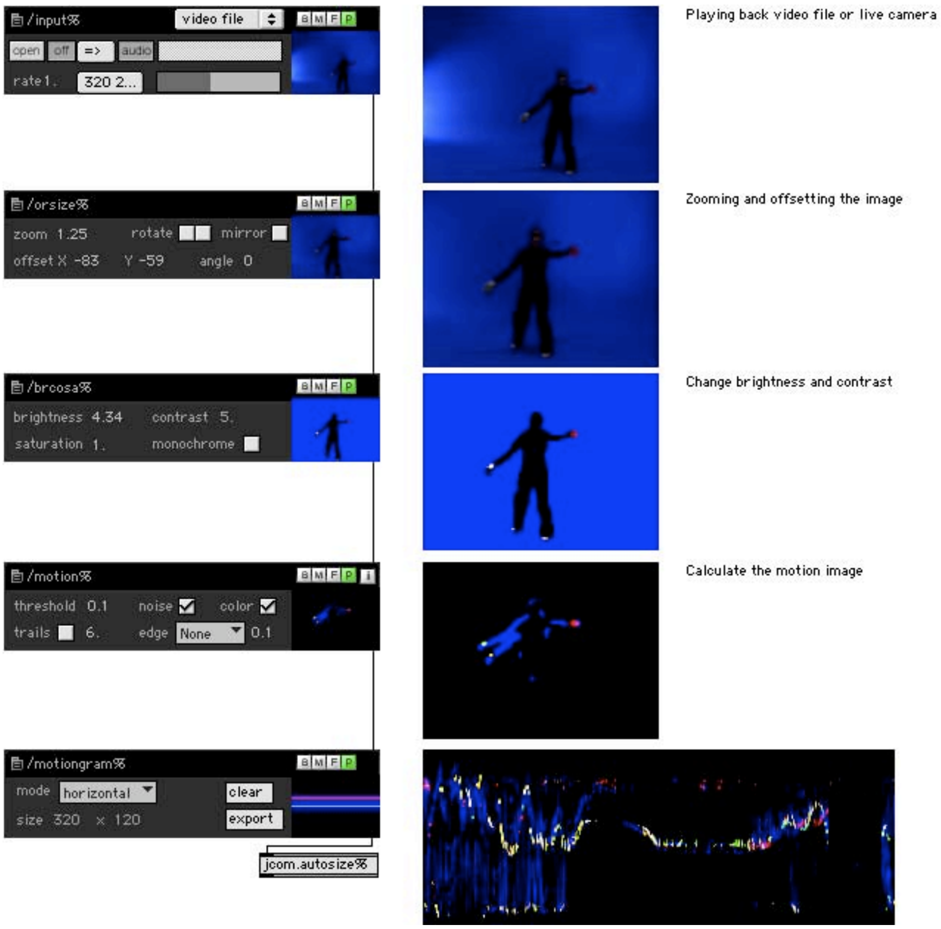

mean
time
Motion image
mean
time
1 pixel wide/tall rows/columns of the mean of the motion image
Running display of collapsed rows/columns (motiongram)

Figure 8.15: *An overview of the process of creating a motiongram, showing the motion image, the averages of the motion images and the running motiongrams.*

An example patch (*motiongram.mxt*) of the process of creating a motiongram is shown in Figure 8.16 and demonstrated in Video 8.2. The process starts by pre-processing the video to get the best possible input signal. This is done by cropping the image to only cover the performance space of the dancer (with **jmod.orsize%**), and adjusting the brightness and contrast to get a good separation between the foreground and background layers (using **jmod.brcosa%**). The pre-processed image is the basis for creating the motion image (using **jmod.motion%**), before finally using the **jmod.motiongram%** module to calculate the motiongram. This module can output either a horizontal or vertical motiongram by averaging over the columns or rows in each matrix respectively. In the first implementation of **jmod.motiongram%**, I developed the **jcom.mean%** component to do this averaging. This, however, is only an abstraction based on native Jitter objects, and was therefore not that efficient. Later I have found the external **xray.jit.mean**[4] to be faster than my algorithm, and I therefore use that one instead in the current implementation of the **jmod.motiongram%** module.

[4]By Wesley Smith, available from http://www.mat.ucsb.edu/ whsmith/xray.html.


/input%
video file
B M F P
open
off
=>
audio
/orsize%
/brcosa%
/motion%
/motiongram%
clear
export
jcom.autosize%
Playing back video file or live camera
Zooming and offsetting the image
Change brightness and contrast
Calculate the motion image

rate 1. 320 2...

zoom 1.25 rotate mirror
offset X -83 Y -59 angle 0

brightness 4.34 contrast 5.
saturation 1. monochrome

threshold 0.1 noise ✓ color ✓
trails 6. edge None 0.1

mode horizontal
size 320 × 120

Figure 8.16: *Screenshot from* ***motiongram.mxt****, an overview of the process of making a motiongram, starting with pre-processing the video, creating the motion image and sending this to* ***jmod.motiongram%*** *which creates the motiongram.*

8.4.2 Reading Motiongrams

It takes some time to get used to reading motiongrams. To guide one's attention, I have added some descriptors to the horizontal motiongram of the 5-minute dance sequence (from Video 7.1) shown in Figure 8.17. Note that time runs from left to right in the display, and that the motiongram only represents the vertical movement of the dancer.

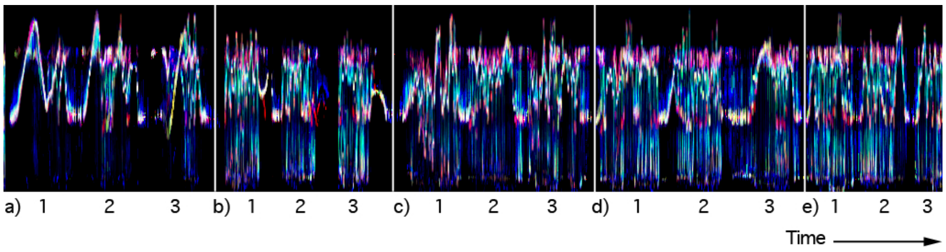


Figure 8.17: *Motiongram of 5 minutes of free dance movements from Video 7.1. The dancer moved to 5 different musical excerpts (a-e) and each excerpt was repeated three times (1-3). Although quite rough, it is easy to see differences in the QoM and similarities in the upward/downward patterns between the sequences.*

As opposed to a spectrogram in which the colours are used to show the energy level of the frequency bands, colours in a motiongram come from the video used to creating the motiongram (here the motion image). In this example, the body looks blue, which is because the dancer wore black clothes, and thus the colour of the background (the blue-screen) becomes visible. It is also possible to see traces of the red and yellow coloured gloves and socks worn by the dancer, something which is even more prevalent when zooming into the motiongram (as can be seen in Figure 8.18).

The added markers in Figure 8.17 show that it is possible to identify the structure of the movement sequence in the motiongram. For example, it is possible to see that the dancer was mainly moving the upper regions of her body in the beginning (section A), while moving the whole body more actively in the rest of the sequence. This motiongram also makes it easy to spot when the dancer was standing still between the music excerpts, since this is shown by black areas in the display (for example in section B).

It is important to remember that a motiongram is a series of *reduced* motion images, in this example reduced in the horizontal plane. This means that the motiongram visualises the movement and its distribution in the vertical plane only. Hence all information about the spatial distribution of movement in the horizontal plane is lost, since this is only represented by 1 pixel for each row (the mean of the row). It might help to think of the motiongram as a display of a collapsed series of pictures, or "stripes", where each "stripe" summarises the content of a whole motion image. When creating motiongrams it is therefore necessary to evaluate in which plane(s) the movement is occurring, before deciding whether to create a horizontal or a vertical motiongram (or both).

It may be argued that the reduction which happens in the creation of a motiongram is unfortunate. Nevertheless, I find that keeping one dimension is better than not keeping any, which is the case in for example QoM plots. I also find motiongrams to be useful since they are based on reduction, and not a specific type of analysis. This makes the method flexible and makes the resultant images more useful for carrying out further analysis. That said, the process of creating a motiongram is necessarily subjective, since it is important to adjust the filtering of the original video image, so as to obtain the most relevant visual result in the motiongram.

8.4.3 Filtering the Motiongram

The output of the **jmod.motiongram%** module is dependent on the input image, so the only way to change the quality of the motiongram is by pre-processing the video and choosing different settings in the **jmod.motion%** module. In general, I find that adjusting the threshold level in the motion module is one of the most important parameters, as it decides how much information is visible in the output motiongram. A low threshold will result in more active pixels in the motion image, which again will result in more information in the motiongram.

The five motiongrams in Figure 8.18 are based on applying different types of filtering and noise reduction. Using no filtering and no noise reduction results in a very noisy image. Applying a little filtering, or noise reduction, greatly improves the output motiongram. There is no rule of thumb, as the levels will have to be adjusted individually based on what the motiongram shall be used for. Setting the correct filtering is important since it influences the analysis. For example, using little filtering and noise reduction makes it possible to see both large and small movements in the whole body. Using noise reduction will make the largest movements stand out more clearly in the motiongram, but it will also remove the smaller movements in the display.

Another important decision that has to be made when creating a motiongram, is whether the motiongram should be in colour or grayscale. Often, I find that grayscale motion images and motiongrams are easier to read since they enhance movements and remove distractive colours. This, however, also depends on the material being studied. For example, in the case of our free dance recordings I find colour motiongrams to be better, since the red and yellow gloves and socks produce noticeable traces which are useful in the analysis.

8.4.4 Videograms

While I usually create motiongrams with the **jmod.motiongram%** module, I have also found that it is sometimes useful to create what can be called *videograms*. These are equal to motiongrams, except that videograms use the regular video image as source material rather than the motion image. As the the two videograms with different filtering in Figure 8.19 show, it is possible to see traces of the movement also in such displays. However, since the dancer is filling up a relatively small part of the entire image, the videograms mainly end up showing the colour of the background. So for steady-shot camera recordings I prefer motiongrams to videograms. However, as will be shown later, videograms can be useful for displaying video material in which the camera is moving.

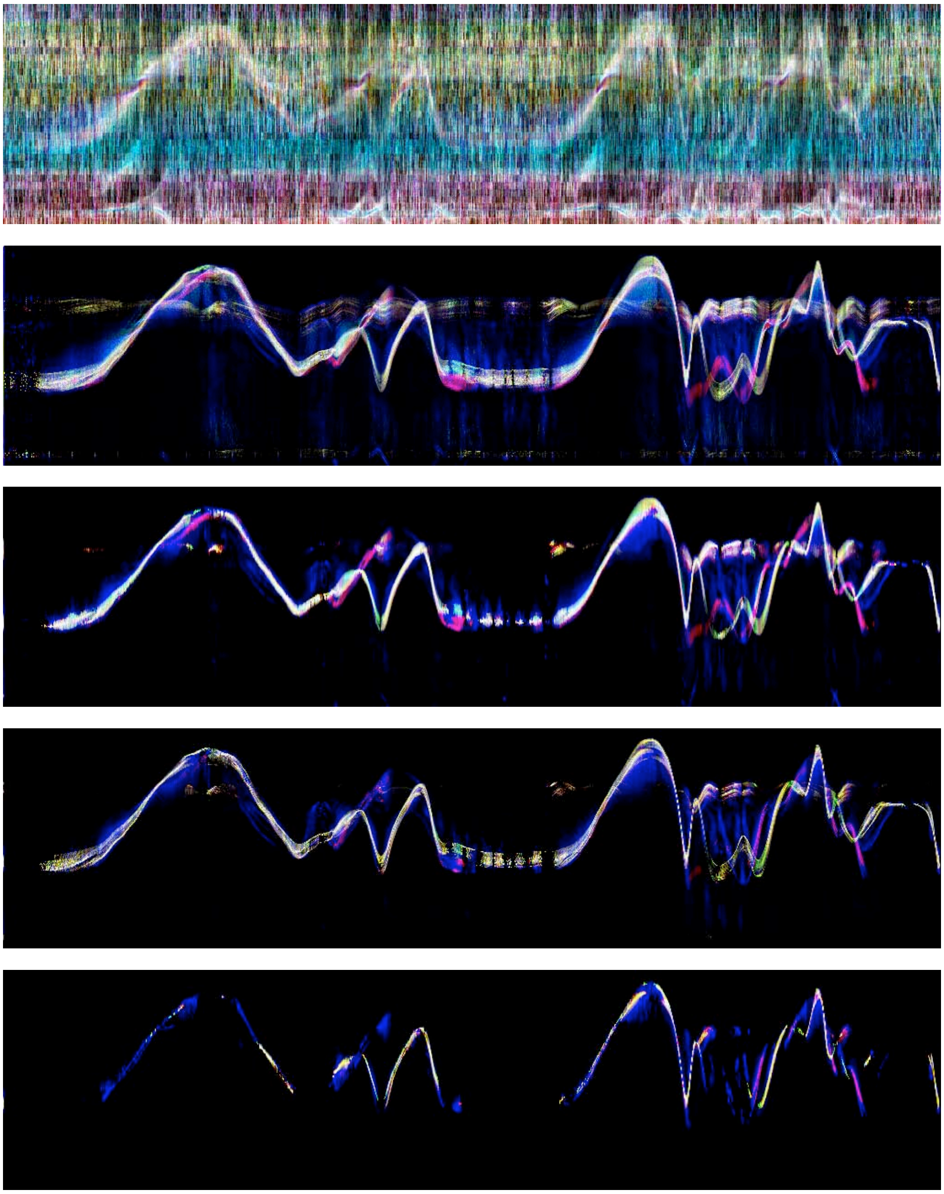


Figure 8.18: *Different motiongrams of the first 40 seconds of Video 7.1, showing the importance of adjusting the threshold functions (t) and noise removal (n) in the* ***jmod.motion%*** *module. From top: 1: t=0.0, n=0. 2: t=0.05, n=0. 3: t=0.05, n=1. 4: t=0.2, n=0. 5: t=0.2, n=1.*

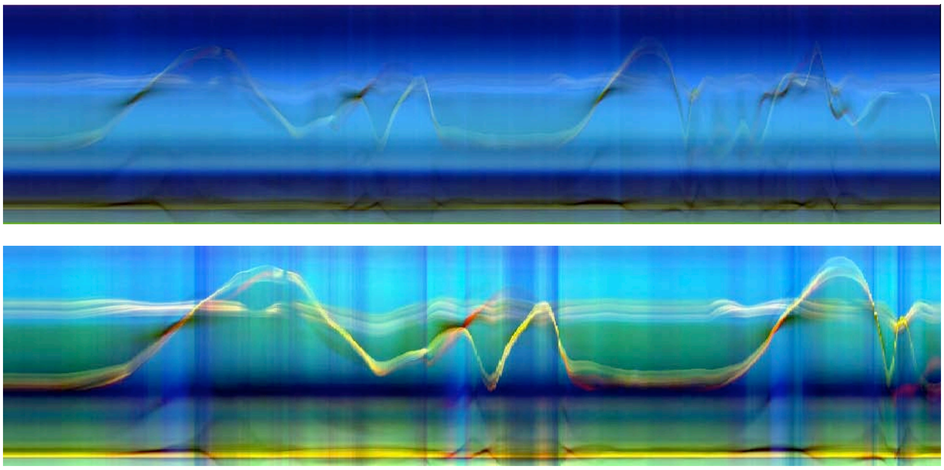


Figure 8.19: *Videograms of the first 40 seconds of Video 7.1. These are made by collapsing the matrices of the original video file (top), and from a filtered video file (bottom).*

8.5 Working with Motiongrams and Videograms

In this section I shall give some examples of how motiongrams and videograms can be used for the analysis of various types of music-related movements, music videos and for medical applications.

8.5.1 Motiongrams in Comparative Studies

I find motiongrams useful to quickly get an overview of video files, particularly in comparative studies. Figure 8.20 shows three motiongrams each representing the 5-minute recordings of the three dancers from our free dance studies. Notice how it is possible to spot similarities and differences when it comes to the spatial distribution of movements as well as the density of movements for the different musical excerpts. For example, the second motiongram shows that this dancer moved much more rapidly than the others in the two last musical excerpts. It is also possible to see how the third dancer covered more of the space with her arms, particularly at the beginning of the sequence.

Figure 8.21 shows the first 40 seconds of the three dance sequences, representing the three repetitions of the first musical excerpt used in the observation study (marked with white lines). These motiongrams reveal more details, and facilitate following the trajectories of the hands and heads of the dancers. For example, in the first motiongram the dancer used quite similar movements for the three repeated excerpts; a large and slow upwards movement in the arms followed by a bounce. The third dancer, on the other hand, varied her movements more, and covered the whole vertical plane with her arms. Such structural differences and similarities can easily be seen in the motiongrams, and can later be studied in more detail in the original video files.

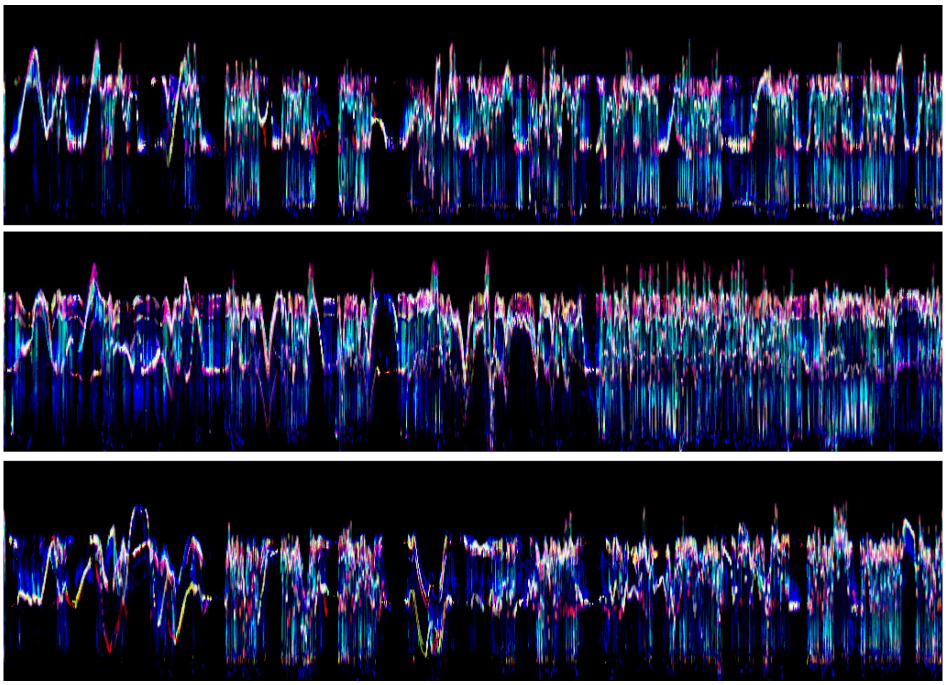


Figure 8.20: *Motiongrams showing the free dance movements of three dancers from our observation studies. Each motiongram represents a 5-minute sequence containing five musical excerpts each repeated three times. Notice how it is possible to get a quick overview of the levels and location of the movement, and make comparisons between the three dancers.*

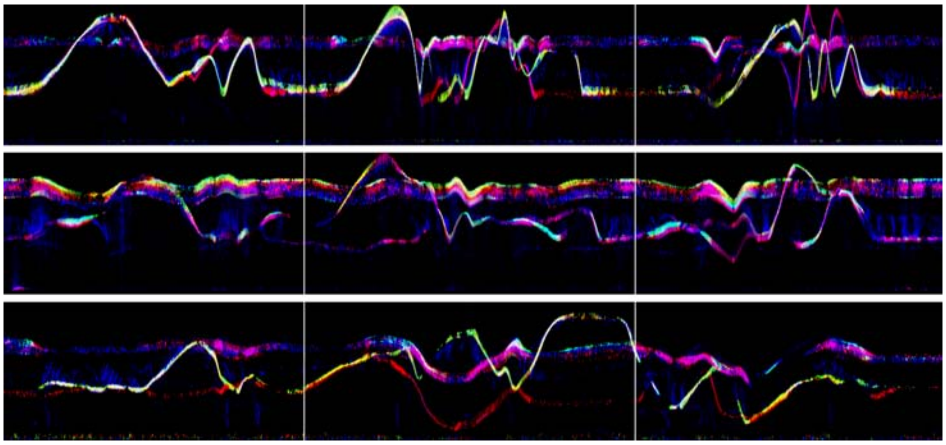


Figure 8.21: *Zooming in on the first 40 seconds of the above motiongrams makes it possible to follow the trajectories of the hands (red and yellow) and head (pink due to saturation) of the dancers.*

8.5.2 Motiongrams of Instrumentalists

Motiongrams are not only useful for studying dance movements, they work equally well with other types of video material in which both the video camera and the performer are stationary. Figure 8.23 shows the motiongram of a clarinettist performing the beginning of the *Allegro appassionato* from the Clarinet Sonata Op. 120, No. 1 in F minor by Brahms (1894). This motiongram is made from Video 8.2, which was recorded for *Workshop on Motion Capture for Music Performance* at McGill University in October 2006,[5] and Figure 8.22 shows a snapshot from the video. In this example I chose to create a horizontal motiongram since most of the interesting movement occurs in the vertical plane. These movements include the up and down movement of the clarinet bell, and the movement of the centre of gravity of the performer. Note how the motiongram visualises the rhythmic sway and weight changes of the clarinettist, which seem to correspond to the breathing patterns and musical phrasings as discussed by Wanderley et al. (2005).

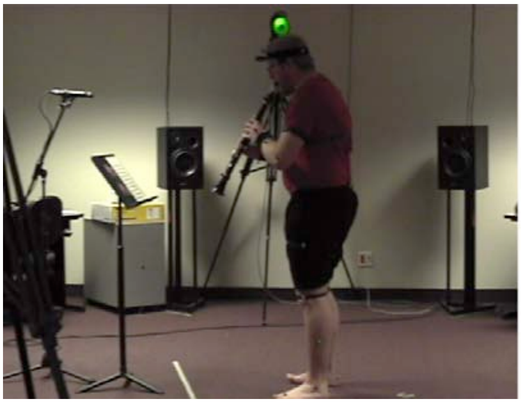


Figure 8.22: *Snapshot from Video 8.3 of a clarinettist performing the beginning of the* Allegro appassionato *from Clarinet Sonata Op.120, No.1 in F minor by Brahms (1894).*

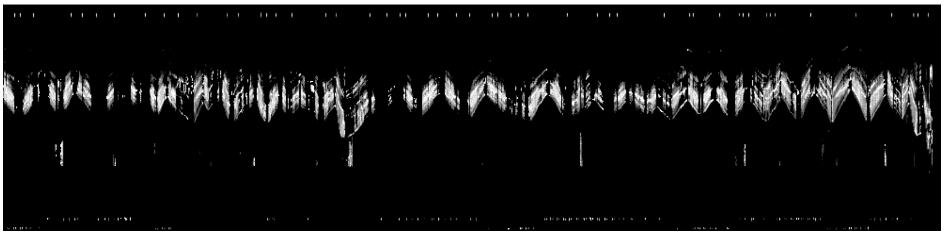


Figure 8.23: *Motiongram made from the clarinet performance in Video 8.3. The thin dotted line in the top part of the motiongram represents the changing light from the infrared motion capture camera, and the thin dotted line at the bottom is the rhythmic tapping of the clarinettist's toes. The few vertical lines below the movement of the clarinet, shows where the clarinettist shifted his weight during performance.*

[5] http://www.music.mcgill.ca/musictech/idmil/mocap/mocap.html

The motiongram in Figure 8.23 is a good example of the importance of pre-processing and filtering the input video before creating the motiongram. As can be seen in Figure 8.22, the background of this video recording is full of visual "noise" (speakers, music stand, etc.). This is not a problem as long as none of the objects is moving, since such elements will effectively be "removed" when creating the motion image. More problematic is the green circle which can be seen above the head of the clarinettist. This is one of the cameras of the Vicon motion capture system which was also used in the recording session. Since this is an active camera, emitting infrared light, it also shows up as a thin line in the motiongram.

Trying to remove the presence of the green light in the final motiongram, I removed the green plane from the video matrix, converted the video to grayscale before creating the motion image and turned up the threshold used in the motion module. This made the green light disappear from the motiongram, but unfortunately it also filtered out many interesting movement features. One of the challenges with pre-processing and filtering is to remove unwanted elements, while keeping the interesting ones. As can be seen in Video 8.3, the clarinettist is tapping his toes while playing, changing between toes when shifting weight from one side to the other. Unfortunately, the heavy filtering also removed these subtle movements, and for this reason I ended up readjusting the threshold levels so that both the camera and the toe movements are visible in the final motiongram in Figure 8.23.

Another example of a motiongram created of a music performance can be seen in Figure 8.24, made from Video 8.4. This video was recorded for the same workshop as mentioned above, and shows a percussionist performing the Violin Concerto in A minor BWV 1041 by Bach (1717) on the xylophone. As can be seen in a snapshot from the video in Figure 8.25, a blank sheet was here used to clear up the background and remove the visual presence of the motion capture camera. This facilitated making the motiongram, since much less filtering had to be applied to attain a good result. The final motiongram clearly visualises the beginning and end of the performance, and shows the regularity in the movement pattern over the course of the piece. In fact, here it is even possible to identify the specific parts of the piece, due to repetitions in the movement pattern. Also interesting is how the end of the performance, when the performer slowly lifts the mallets and steps back, can be clearly seen in the motiongram.

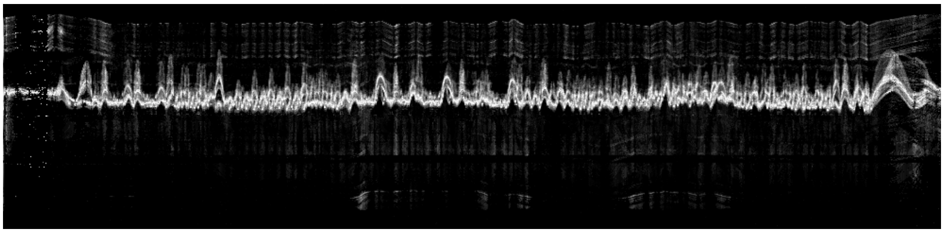


Figure 8.24: *Motiongram of the percussionist recording in Video 8.4. It is possible to see the structure of the piece through the periodicities of the hand movements. Note also the final, sustained action when the performer finishes performing and steps back.*

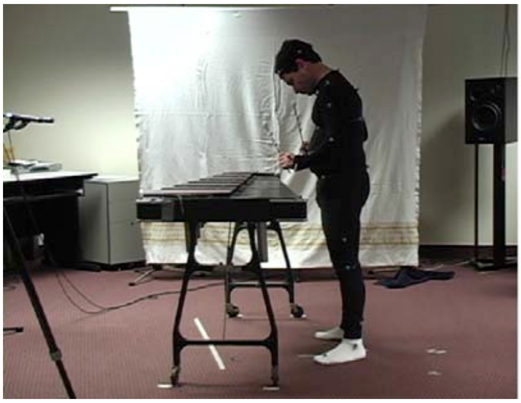


Figure 8.25: *Snapshot from Video 8.4 of a percussionist performing J.S. Bach's Concerto in A minor BWV 1041 on the xylophone (Bach, 1717).*

As these examples show, motiongrams may be an efficient tool for visualising movements in recordings of music performances. Such recordings share many properties with our observation studies in that the subjects are standing still and the camera is not moving. These two factors are important, because if the subject is moving too much around, it is usually the movement of the torso which is displayed, and not the movement of the hands or head. Similarly, camera movement will result in a motiongram which represents the camera movement more than the movement of the subject. I therefore believe that motiongrams work better for single-shot, steady camera recordings, but the next section will show that the technique may also be useful for other types of video material.

8.5.3 Videograms of Music Videos

One of the main reasons for creating motiongrams was the idea of having a tool for visualising and navigating in large collections of video material. This worked well for the video material from various observation studies, but I was also interested in seeing whether the technique could be used for studying other types of music-related material, for example commercial music DVDs. However, such videos typically contain lots of shots and scenes, and camera zooming, panning and tilting. The resultant motiongrams therefore contain little information about movement inside the frames, but mainly display the movement of the camera. For such video recordings I therefore find that videograms work better, since they are better at displaying reduced features of the original videos.

Figure 8.26 shows two videograms of the concert documentary movie *Bring on the Night* by Sting (1985). The videogram at the top is of the whole movie (1 hour and 40 minutes), while the videogram at the bottom is of the first 5 minutes. These videograms show a high level of visual activity and may be seen as a rather chaotic display. However, I find that such videograms do show some interesting features of the movie. For example, the full videogram gives an overview of the general features of the whole movie. The short videogram more effectively shows colour differences between scenes, and camera zooming and panning. These features may be used to study relationships between different parts of the movie, as well as for navigating between different shots and scenes.

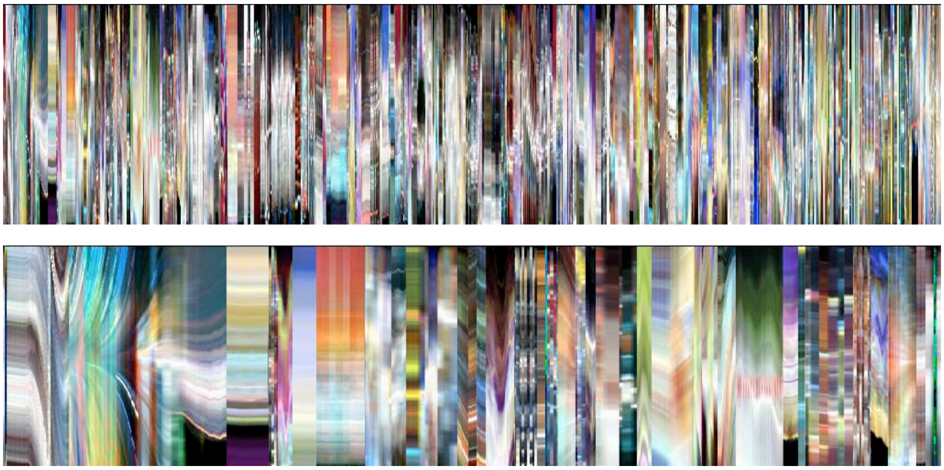


Figure 8.26: *Videograms of the whole movie (top) and first 5 minutes (bottom) of the music movie* Bring on the Night *by Sting (1985). The display is fairly chaotic, but shows that the movie contains lots of shots and different colours. The videogram of the first 5 minutes reveals a lot of camera movement and zooming, and it is possible to detect different scenes and shots.*

Another set of videograms made from a commercial music DVD is shown in Figure 8.27. These videograms are made from *Minimum–Maximum* by Kraftwerk (2005), a highly visual performance DVD. This visuality can also be seen in the resultant videograms, and I am particularly fascinated about how clearly it is possible to identify the structure of the performance in the short videogram. Here it is possible to study in detail how different parts of the performance are structured in time.

An example of videograms that do not work so well can be seen in Figure 8.28. These videograms are of a concert DVD of Pierre Boulez conducting the Chicago Symphony Orchestra in Alban Berg's Lulu (Boulez, 2000). This is a typical Western art music video production, featuring an orchestra on stage and a multi-camera setup. The reason why I believe that these videograms do not work so well, is because of the many and frequent changes in shots between the whole orchestra and various musicians. This recording is also much less visual in its production, so the full videogram looks fairly monotonous. Nevertheless, in the short videogram it is possible to see some of the structure of the recording, for example the opening titles, and when there is a close-up of the soloist. Going even closer reveals more details, so videograms may also be useful for such material.

As the three examples in this section have shown, videograms may be useful for analysing "normal" video material, particularly if there is a certain level of visuality in the production. I believe that videograms could also be an efficient tool for navigation in such files. This, however, requires the development of solutions for more efficiently playing back the original video file by "scrubbing" around in the videogram, for example with a mouse. Coupled with a more flexible system for zooming in and out of the displays may call for a potentially powerful playback solution.

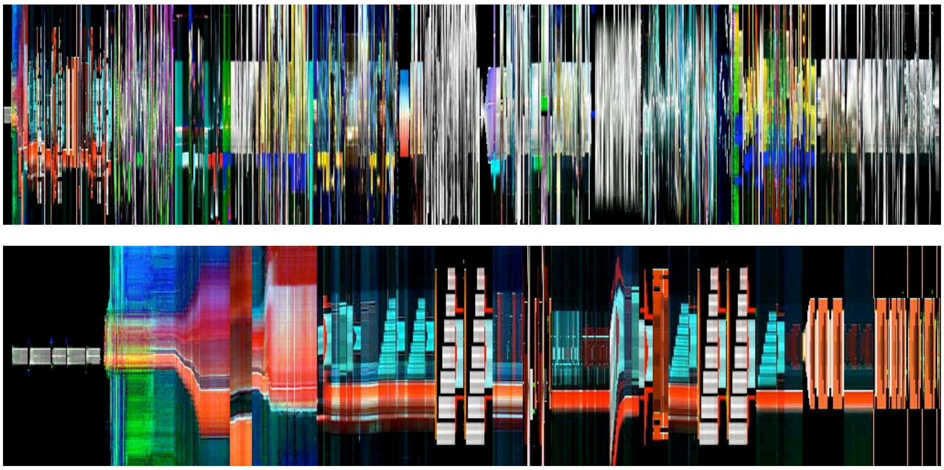


Figure 8.27: *Videograms of the whole (top) and first 5 minutes (bottom) music DVD* Minimum–Maximum *by Kraftwerk (2005). The videograms facilitate seeing relationships between different parts, and the temporal development in the performance.*

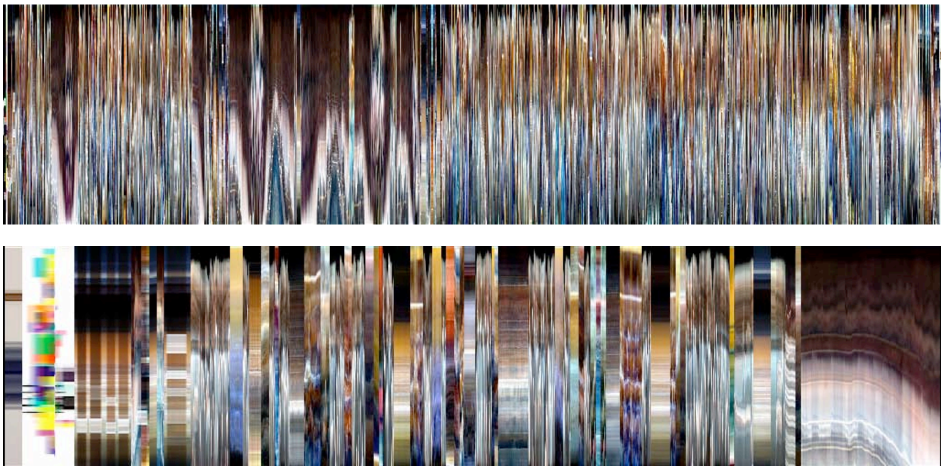


Figure 8.28: *Videograms of the whole (top) and first 5 minutes (bottom) of a concert DVD of Pierre Boulez conducting the Chicagoy Symphony Orchestra (Boulez, 2000). This is an example of how videograms do not work so well, since the movie is composed of a number of different shots with the same colour. The frequent change in camera position, as well as camera zooming and panning further complicates the visual appearance of the videogram.*

8.5.4 Motiongrams in Performance

I originally developed motiongrams for working with analysis, but have also found that they work well in a performance context. Figure 8.29 shows a picture from a demonstration of MGT at Oslo Concert Hall in September 2006, where motion images and a motiongram of the movements of dancer Åshild Ravndal Salthe. Here motiongrams were used mainly as a visual effect, but they also served the purpose of enhancing the performance by displaying the unfolding of movements in time.

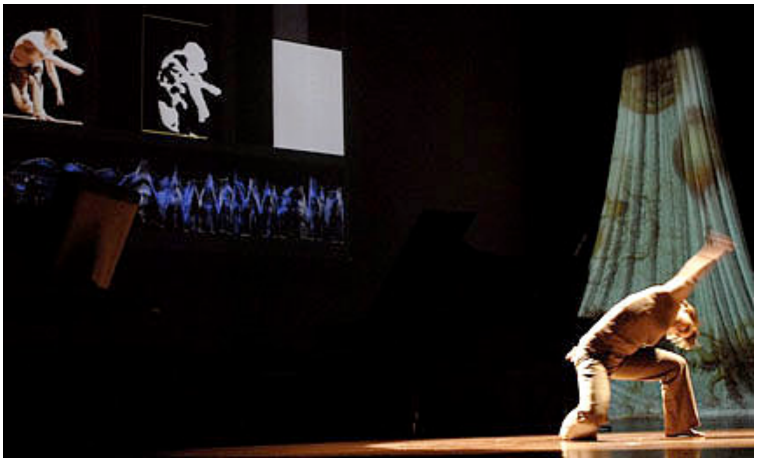


Figure 8.29: *From a performance in Oslo Concert Hall, September 2006. Different types of motion images and motiongrams were projected onto the screen behind the stage.*

8.5.5 Motiongrams for Medical Use

In the autumn of 2006, I was approached by Professor Terje Sagvolden from the Department of Physiology at the University of Oslo. He had seen my performance at Oslo Concert Hall, where I used motiongrams to display dance movements, and he was interested in using my tools to study the movements of rats in his experiments on *attention deficit hyperactivity disorder* (ADHD).

ADHD is a seemingly heterogeneous group of behavioural disorders affecting between 2% and 12% of young children (Swanson et al., 1998; Taylor et al., 1998). It usually manifests itself before the child is 7 years old, and is characterised by inattentiveness, hyperactivity and impulsiveness (Applegate et al., 1997). Around 50% to 70% of children diagnosed with ADHD will have problems relating to social adjustment and functioning, and are also more likely to have psychiatric problems as adolescents and young adults (Cantwell, 1985). It is therefore important to identify children with ADHD at an early age so that they can receive necessary treatment and support (Sagvolden et al., 2005).

There are three subtypes of ADHD diagnosis and two behavioural dimensions (American Psychiatric Association, 1994):

- *ADHD* (*attention deficit hyperactivity disorder*) is a predominantly hyperactive and impulsive subtype, typically more common in boys.
- *ADD* (*attention deficit disorder*) is a predominantly inattentive subtype, typically more common in girls.
- A combination of the two.

Sagvolden's group carries out experiments using genetically engineered rats that have symptoms equal to those of clinical cases of ADHD. They have found that spontaneously hypertensive rats (SHR) from Charles River, Germany demonstrate overactivity and poor sustained attention, which is similar to clinical cases of ADHD. Another breed of rats, called Wistar/Kyoto (WKY), also from Charles River, Germany, shows poor sustained attention similar to the SHR rats, but without overactivity. They can thus be seen as a model for ADD. A third group of rats, WKY rats from Harlan UK, shows normal behaviour and is used as a control group in the experiments (Sagvolden et al., 2005).

Detailed descriptions of the setup and method used in the experiments can be found in (Sagvolden, 2006). Briefly, the rats are placed inside a cage (as shown in Figure 8.30), and given the task of pressing one of two levers when a lamp lights up above one of the levers. If the task is carried out correctly, the rat will receive a drop of water as a reward. The rats used are young, and are trained for some weeks before the actual experiments are started. After the training period, experiments are run daily for several hours, and data about the activity of the levers are logged and used for further analysis. The aim is to study patterns of overactivity, impulsiveness and inattentiveness over sustained periods of time, and to see whether various types of medical treatment will change the behaviour of the rats.

One problem with the current experimental setup is that only the activity of the levers is recorded. This results in highly discrete and time-spaced measurements which provide no information about how the rats behave the rest of the time. This is where motiongrams may be interesting, since they can visualise the movement of the rats over long periods of time. For this reason we have carried out a pilot study in which a few rats were recorded with a web camera placed inside the cage. Videos of the three different types of rats (SHR, WKY ADD, and WKY) can be seen in Video 8.5, 8.6 and 8.7, and horizontal and vertical motiongrams of these videos are shown in Figure 8.31 and 8.32, respectively. These motiongrams reveal some differences between the movement patterns of the rats. In general, the normal WKY rat moves the least while the SHR rat moves the most. The motiongram of the WKY ADD shows that this rat moved almost as much as the SHR rat, but its movements were more controlled than those of the SHR rat.

Based on the promising results of the pilot study, video cameras are now being installed in all the experimental cages. The idea is to record video of the sessions, and

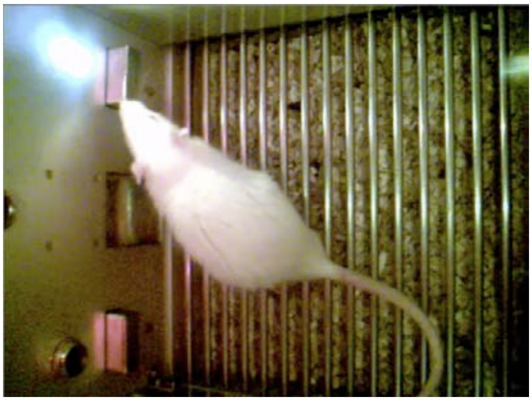


Figure 8.30: *A WKY ADD rat in the experiment cage, taken from the upper right corner of the cage. The two levers that the rats are supposed to push in the experiment can be seen on the left side, each having a light above them. The feeding mechanism is seen between the two levers.*

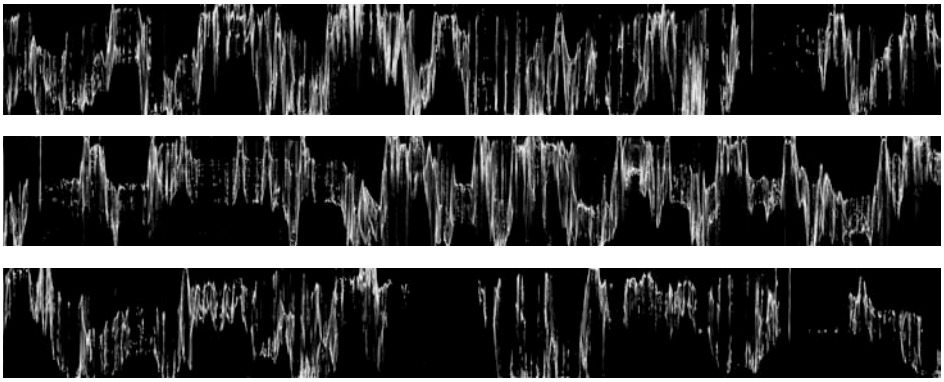


Figure 8.31: *Horizontal motiongrams of SHR rat (top), WKY ADD rat (middle) and a normal WKY rat (bottom). Time is running from left to right. The motiongrams clearly show that there are differences in activity patterns between the three rats.*

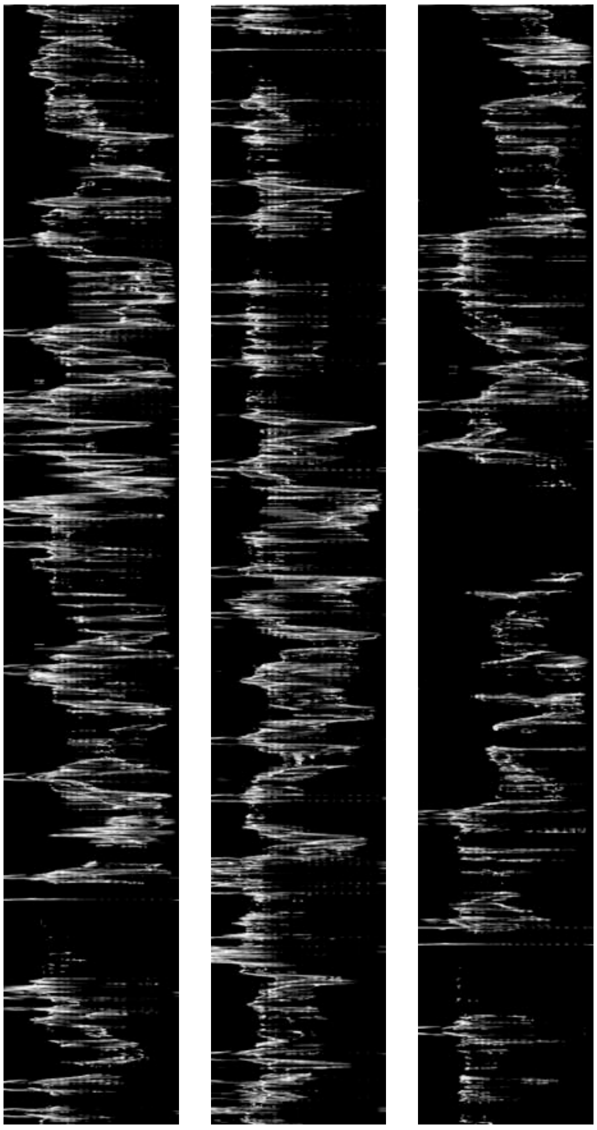


Figure 8.32: *Vertical motiongrams of SHR rat (left), WKY ADD rat (middle) and a normal WKY rat (right). Time runs from top to bottom, and left and right refers to the left and right sides in the original image (Figure 8.30). These vertical motiongrams makes it possible to see when the rat goes to the levers or feeder (located on the left side).*

use motiongrams for analysis. We have also tested other tools from MGT to analyse the material, and will continue to explore this in future experiments.

8.6 Summary

Inspired by some of the early experiments in photography by Muybridge and Marey, combined with recent video abstraction techniques, I have explored and developed various solutions for movement visualisation: *salient keyframe displays*, *motion history images*, *motion history keyframe displays*, *videograms* and *motiongrams*. These types of displays all address different needs in representing music-related movement. I am particularly fascinated by how motion history images may be used to visualise movement trajectories over short periods of time (up to 10 seconds). Such displays may be seen as resembling our perception and cognition of the movements, and are therefore useful when studying and analysing movement segments. Motion history images are also useful for displaying movements in "static" media, for example printed documents.

Motiongrams are my preferred solution for representing longer movement sequences. They can be used to visualise movements from a few minutes to several hours, and helps quickly get an overview of both the location and quantity of movement over time. This makes them useful as movement summaries, when navigating and in comparative studies. Motiongrams are not based on any particular analysis, only matrix reduction, and this makes the approach stable and flexible as long as the videos are recorded with a steady camera. But, as the videograms of music videos showed, the method is even useful for more complex video material.

The techniques presented in this chapter were developed to study music-related movement, but as the recent research collaboration with the department of physiology shows, they can also be useful in other fields of study. This collaboration is still in its infancy, but we have many ideas about the continued development of motion history images and motiongrams. Some of the topics that will be explored further in future development include:

- creating three-dimensional motiongrams to display both horizontal and vertical motion in one image. I have tried creating various types of motiongram "waterfall displays", but these quickly became too complex and difficult to work with. A better solution would probably be to implement motiongrams using a three-dimensional OpenGL model[6] which can be controlled in realtime by the user.

- creating motiongrams based on multi-angle camera recordings. All our observation studies were recorded with two cameras, one at the front and one from the side. It would be interesting to see if it is possible to create a three-dimensional motiongram by combining these two images.

[6]The *Open Graphics Library* (OpenGL) is a computer graphics standard which facilitates making three-dimensional models that can be run on the graphics card in the computer.

- improving the processing speed when creating motiongrams. The current implementation is not very efficient, since it is based on running through all frames in the whole video file to create a motiongram. I will explore solutions with which to create motiongrams based on *non-linear sampling*, i.e. sampling frames progressively at various locations in the video file.

- improving the quality of large motiongrams. The current implementation is based on sampling frames based on a given frame number. As the distance between frames increases, this will decrease the chance of sampling a perceptually relevant frame. A better method is to average the frames within each analysis "window", for example by using a running low-pass filter such as implemented in **jit.slide**. This will boost the processing load, but will result in nicer looking motiongrams.

In future development I will also continue to explore how we can create tools that more effectively simulate our perception and cognition of movements. I believe such perceptually based displays will allow for better solutions when it comes to navigating and analysing videos of music-related movement.

CHAPTER 9

The Gesture Description Interchange Format

Computers have promised us a fountain of wisdom but delivered a flood of data.

(Frawley et al., 1992, 57)

This chapter presents the background and current state of development of the *Gesture Description Interchange Format* (GDIF)[1] for streaming and storing music-related movement data. A multi-layered approach to data handling is suggested, including everything from raw data to higher level descriptors. A prototype implementation is presented and discussed in view of meaningful and flexible mapping.

9.1 Introduction

A major obstacle to music-related movement research is the lack of appropriate *formats* and *standards*. *Format* is used here to denote a specification of the way information should be encoded in a computer file. A format can eventually become a *standard* if it receives widespread usage and/or is accepted as a standard by the *International Organization for Standardization* (ISO).[2] Working with hardware and software tools that cannot efficiently interchange information is a limitation not only for the single researcher, but it also effectively blocks collaboration between researchers and institutions. Rather than analysing data, this makes researchers spend a lot of time and effort just getting their data

[1] Although I have personally found the term gesture somewhat problematic (as discussed in Chapter 4), it is still a commonly used term in the music technology community. For this reason I decided to keep the name of GDIF as it was first suggested.

[2] http://www.iso.org

transferred from one system to another. Developing formats that can be used in music-related movement research therefore seems of great importance, and is something that can also benefit the whole music technology community.

9.1.1 Background

Problems related to data streaming and storage have been following me since our first observation studies in 2004. At that time I was interested in storing data coming from a Kroonde sensor interface, and synchronise these data with simultaneously recorded audio and video files. The solution I came up with was to save the raw data in text files, with references to the corresponding audio and video files. However, as I did not store any information about the setup, scaling and filtering used, etc., these data files quickly became useless as I could not recall what they referred to. In later studies I was more careful about taking notes, but I still ended up with lots of problems relating to synchronisation between the data files and the audio and video files.

After talking to other researchers in the field, I realised that nobody seemed to have a general solution to store their experimental data coherently and consistently. With some other PhD students at the S2S^2 summer school in Genoa 2005,[3] I started thinking about creating a solution to our needs. The idea slowly developed, and we presented a rough proposal for GDIF at the Conference on New Interfaces for Musical Expression (NIME) in 2006 (Jensenius et al., 2006). Here I received a lot of interesting feedback, and many suggestions for further development. Over the last year several prototype setups have been developed to test out GDIF ideas in practice: using hand movements to control sound spatialisation in realtime (Marshall et al., 2006), analysing violin performance (Maestre et al., 2007), and using multilayered GDIF streams in a networked setup (Jensenius, 2007). GDIF was also discussed in a workshop of the EU COST Action 287 ConGAS[4] in Oslo in December 2006, and will be discussed in a panel session on music-related movement formats at the International Computer Music Conference (ICMC) in Copenhagen in August 2007 (Jensenius et al., 2007).

9.1.2 Various Formats and Standards

A number of formats exist that are related to, and cover parts of, our needs to stream and store music-related movement data. One type of format is related to the many different motion capture systems available, including the *AOA*[5] format used with the optical tracker systems from *Adaptive Optics*,[6] the *BRD*[7] format used with the *Flock of Birds* electromagnetic trackers,[8] and *C3D*[9] used with the *Vicon* infrared motion capture sys-

[3]http://www.s2s2.org

[4]http://www.287.org

[5]http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/AOA.html

[6]http://www.aoainc.com/technologies/technologies.html

[7]http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BRD.html

[8]http://www.ascension-tech.com/products/flockofbirds.php

[9]http://www.c3d.org

tems.[10] Several formats have also emerged for using the data from such motion capture systems in animation tools, for example the *BVA*[11] and *BVH*[12] formats for technologies from *Biovision*,[13] and the *ASF* and *AMC* formats[14] from game developer company *Acclaim*[15] (Lander, 1998), as well as formats used by animation software, such as the *CSM* format used by *Autodesk 3ds Max*.[16]

Some of these motion capture formats are used in our community of researchers on music-related movement, but often these formats create more problems than they solve. One problem is that they are proprietary and often closely connected to a specific hardware system and/or software solution, which makes them inflexible and impractical for our needs. Another problem is that they often focus on full-body motion descriptors, based on a fully articulated skeleton. This may not always be convenient in setups where we are only interested in a specific part of the body, and it also makes such formats less ideal as a starting point for a more generic format to encode movement data. Yet another problem is that most of these standards are intended to store raw data and/or basic movement descriptors, which leaves little room to store analytical results and annotations, as well as various types of music-related data (audio, video, midi, OSC, notation, etc.).

Another type of formats that could be interesting for our needs is the large number of XML based formats used in motion capture and animation, for example the *Motion Capture Markup Language* (MCML) (Chung and Lee, 2004), *Avatar Markup Language* (AML) (Kshirsagar et al., 2002), *Sign Language Markup Language* (Elliott et al., 2000), *Multimodal Presentation Markup Language* (MPML) (Prendinger et al., 2004), *Affective Presentation Markup language* (APML) (De Carolis et al., 2004), *Multimodal Utterance Representation Markup Language* (MURML) (Kranstedt et al., 2002), *Virtual Human Markup Language* (VHML) (Beard and Reid, 2002), etc. As far as I can see, none of these formats stands out as a candidate to meet our needs, but they should be evaluated more closely to see what we can learn from their efforts.

The MPEG series of formats and standards may then be more interesting. Developed by the *Motion Picture Expert Group* (MPEG),[17] these standards have gained a strong commercial position and wide distribution, and allow for audio and video compression (*MPEG-1*, *MPEG-2* and *MPEG-4*) (Hartmann et al., 2002), as well as for storing metadata and annotations (*MPEG-7*) (Martinez et al., 2002; Manjunath et al., 2002). As such, these standards could potentially solve many of our storage problems, including synchronisation with audio and video. Unfortunately, they seem to be too geared towards general commercial multimedia applications, and may therefore be less ideal for music analysis and synthesis applications.

When it comes to streaming and storing low-level movement data, the *Gesture Mo-*

[10] http://www.vicon.com/products/
[11] http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVA.html
[12] http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html
[13] http://www.biovis.com/
[14] http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ASF-AMC.html
[15] http://www.acclaim.com
[16] http://www.autodesk.com/3dsmax/
[17] http://www.chiariglione.org/mpeg/

tion Signal (GMS)[18] format is interesting. GMS is a binary format based on the Interchange File Format (IFF) standard (Morrison, 1985), and is being developed by the ACROE group in Grenoble and partners in the EU Enactive Network of Excellence.[19] The goal of GMS is to provide a solution for structuring, storing and streaming of basic movement data (Evrard et al., 2006; Luciani et al., 2006). I got to know about the GMS initiative in May 2006, and have contributed to parts of a deliverable on the needs of the GMS standard (Luciani et al., 2006). GMS focuses on coding basic movement descriptors, and may therefore be seen more as a companion format than a competitor to GDIF. For this reason I have found it more interesting to focus GDIF development on mid- and higher level features.

Another relevant format is the *Performance Markup Language* (PML),[20] which is currently being developed by Douglas McGilvray at the University of Glasgow (MacRitchie et al., 2006). PML is an extension to the Music Encoding Initiative (MEI)[21] (Roland, 2002), and the main idea is to facilitate the analysis and representation of performance in the context of the musical score. PML allows the description of performance events and analytical structures with reference to the score and external sources such as audio files. The current version uses MusicXML as a basis for score representation (Good, 2001), though PML can be used with any XML based score representation. I got to know about the PML project in December 2006, and am looking forward to seeing how it develops, and how it can work together with GDIF.

Finally, we should not forget the *Sound Description Interchange Format* (SDIF), the inspiration for GDIF. SDIF was developed jointly by CNMAT and IRCAM in the late 1990s, as a solution to standardise the way audio analysis is stored (Wright et al., 1998). Just as I have found it difficult to store and work with movement data, the SDIF developers found the need for a format to store audio analysis data in a structured way. SDIF is supported in a number of audio programs and programming environments (Schwarz and Wright, 2000), although its usage seems to be somewhat limited. Nevertheless, the SDIF specification and software implementations have tackled a number of challenges relating to the synchronisation of multiple streams of high-speed data, and should therefore be studied closely in further GDIF development. In fact, GDIF may even be implemented as a part of SDIF.

9.2 Needs

We have a number of different needs when creating solutions to stream and store music-related movement data, some of which will be presented in this section. First, we need a structured and coherent approach to handling a large number of different types of raw movement data and associated analytical results, including:

[18] http://acroe.imag.fr/gms
[19] http://www.enactivenetwork.org
[20] http://www.n-ism.org/Projects/pml.php
[21] http://www.lib.virginia.edu/digital/resndev/mei/

- *raw data*: unprocessed data from sensing devices, for example motion capture systems, commercial game controllers, MIDI devices, and homemade music controllers.

- *pre-processed data*: filtered, scaled and normalised data that are the basis for further analysis.

- *analytical results*: the data resulting from both quantitative and qualitative analysis of the above mentioned data.

The idea is that all these data may be streamed and stored next to each other, as illustrated in Figure 9.1. A GDIF file may therefore contain a number of different time-synchronised streams so that it is possible to retrieve all the data from any point in time. It should also be possible to add new data streams resulting from various types of analyses, something which will allow different researchers to work on the same material.

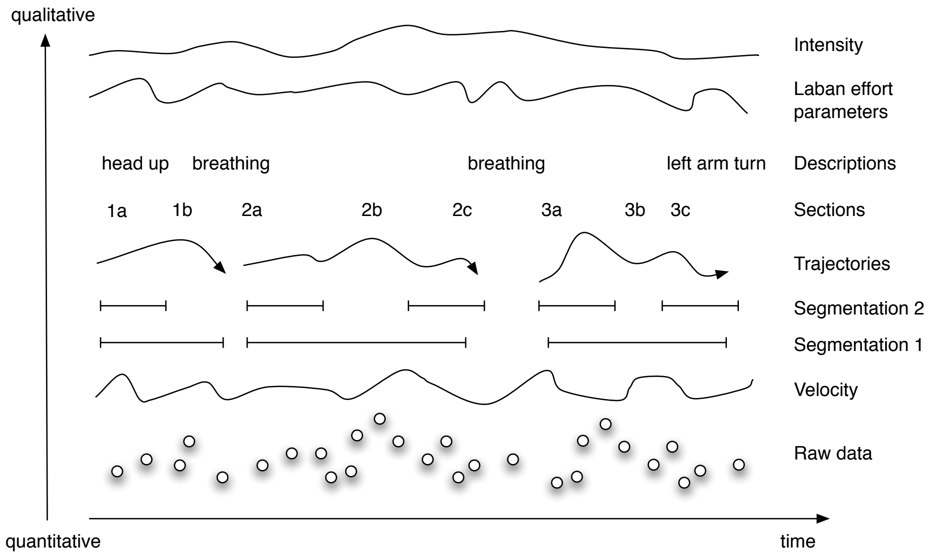


Figure 9.1: *A sketch of the different types of raw data and analytical results that could be streamed and stored in GDIF.*

9.2.1 Different Types of Raw Data

The first step in GDIF development is to find solutions to handle raw data. We typically work with many different types of systems and devices which all have different features and properties:

- *motion capture systems*: such systems usually output data at high speeds (up to 4000 Hz) for a number of multidimensional markers (anything from 2 to 50, sometimes even more) with different degrees of freedom (often 3 or 6). Most motion capture systems use their own proprietary formats to store the data from the device, although several seem to be able to export the C3D format.

- *MIDI devices*: most commercial music controllers and instruments output MIDI. As discussed in Chapter 6, MIDI is an event-based protocol which does not contain any information about the movements associated with the control messages. As such, MIDI is not particularly suited for streaming or storing movement related data, but due to its widespread use, we need to find solutions to include data from such devices. This should not be too problematic since the MIDI specification is clear, and the messages are well defined. MIDI streams are typically output at fairly low sampling rates (up to a 100 Hz), and low bit-rates (7-bit).

- *commercial controllers*: game controllers, graphical tablets, mouses, and other commercial devices usually comply to well-known protocols (for example HID) and use more or less well-defined ranges and resolutions. As is the case for MIDI devices, such general input devices mainly focus on control aspects, and not the body movements associated with these control messages.

- *custom-made controllers and instruments*: such devices are more problematic to make general specifications for, since they often only exist as a single prototype. That said, custom-made controllers are often built with standard sensors, and use sensor interfaces based on a well-known protocol for data transfer (for example MIDI or OSC), which may be used as a starting point for a general specification.

Most of these devices, except for the custom-made controllers, have native formats and messaging structures that should be preserved in GDIF for backwards compatibility. This is particularly important in experiments, so that it is possible to check the raw data in case there are problems or uncertainties resulting from the GDIF encoding.

9.2.2 Time and Synchronisation

One of the major challenges when it comes to finding a solution to code data from various devices, is that of *time-coding*[22] and *synchronisation*.[23] Such issues will have to be addressed in GDIF development, including time-coding *within* streams, and synchronisation *between* streams of data.

First, when time-coding inside streams, many of the devices we work with operate with a fixed sampling rate which may be used as the basis for time-coding the material. However, I have experienced that such "fixed" sampling rates are not as accurate as they claim to be, and may drift over time. This is particularly evident when recording data from high-speed systems which demand a lot of computer memory, disk access and CPU

[22]A *time-code* is a numeric sequence used to keep track of time in a system.

[23]*Synchronisation* is used here to denote the organisation and coordination of events in time.

processing. Another challenge related to time-coding inside streams is that several devices (for example MIDI) operate with *delta* time, based on specifying events relative to the preceding and/or succeeding event. The messages from such devices are typically sent at an uneven rate, transmitting only when they are active. It is necessary to find a solution to handling such a time-coding consistently.

Several challenges arise when synchronising data streams from devices using relative time-coding with data sets using a fixed sample rate. Here an important question is whether all streams should be time-coded and sampled with the same time-code. This would allow for easier synchronisation, but it would probably also lead to redundant information being stored. For example, if data from a MIDI device is to be synchronised with data from a high speed motion capture system, the MIDI data would have to be recorded at a much faster rate than they are output from the MIDI device. The best solution is probably to allow each stream to operate with its own time-coding, and create a system to secure synchronisation between the streams.

A third challenge is related to synchronising the various data streams with simultaneously recorded audio and video. Fortunately, there are several industry standards for both audio and video time-coding and synchronisation, one of the most commonly used being the SMPTE time-code.[24] Using a video time-code as the basis for time-coding data in GDIF may not be the best option, though, since several of these time-codes are based on a combination of time and video frames. Also, the question of time-code is related to the media formats and codecs being used. I find it important that GDIF should not rely on specific audio and video formats/codecs (e.g. QuickTime, Windows Media Audio/Video, etc.), so we need to find a more generic solution to synchronise such media with the movement data.

Finally, we also have to tackle challenges related to synchronising data and media that are not based on a specific time-code. Music is largely based on relative timing, often defined by a musical score, and different performances of the same piece often vary considerably in duration and timing. It is therefore important to develop solutions to synchronise the absolute or delta time-codes of movement data, with the relative time-codes of musical scores. This will probably have to be based on solutions to "time warp" data sets, or set musical "keyframes" that define synchronisation points.

9.2.3 Requirements

When it comes to the format itself, there are several criteria that should be met to make GDIF versatile and useful in music-related movement research. The format should be:

- *open*: the specification and the implementations should be open (source) so that everyone can use them and expand them as they see fit.
- *human-readable*: it should be possible to easily understand what the data means by looking at the data stream or file, or there should be converters readily available if storing in a binary format.

[24] Developed and used by the *Society of Motion Picture and Television Engineers* (SMPTE).

- *multiplatform*: cross-platform and cross-software support is essential to ensure compatibility with systems past, present and future.
- *compatible*: it is important to ensure that the format will work with other relevant formats and standards. As far as possible it should also be based on other well-known standards to allow for easy adaptation.
- *simple*: the basic set of descriptors should be as small and simple as possible so that it is easy to get started using the format.
- *extendable*: the format should allow for a great deal of flexibility when extending it to cover new areas that were not thought of in the original specification.
- *efficient*: I believe the above mentioned elements are the most important, and many of these contradict efficiency. GDIF is intended as a research format, and we are therefore not limited to the same type of restrictions that may be found in industrial standards. That said, efficiency will also be a goal as long as it does not come in conflict with the other aims.

These requirements have been the basis for our choice of solutions and implementations which will be described in the following sections.

9.3 A Multilayered Approach

While many data formats often focus only on storage, I believe it is important to create a solution that allows for both streaming and storage. This will make GDIF useful not only in experiments and analysis, but also as a tool for realtime control. In fact, even though streaming and storage may seem to be two widely different domains, they are often closely connected when working with movement and music. Streaming data from a controller and mapping these data to features in a music engine involves many of the same types of processing and analyses as does recording data to a file. Also, since GDIF development is as much about conceptualisation as it is about computer formats, both streaming and storage should be included in the process.

9.3.1 Streaming and Storage

GDIF is currently being developed in multiple directions: an implementation for real-time control of sound synthesis, and two different storage solutions. A sketch of this is shown in Figure 9.2. The idea is that data from the sensor devices will be converted to GDIF OSC streams as quickly as possible, and that these streams will be used both internally and in network communication.[25] This makes GDIF immediately useful for OSC-enabled devices and systems, and allows us to focus on conceptual rather than technical development.

[25]Typically using UDP/IP for communication, but other protocols could also be possible.

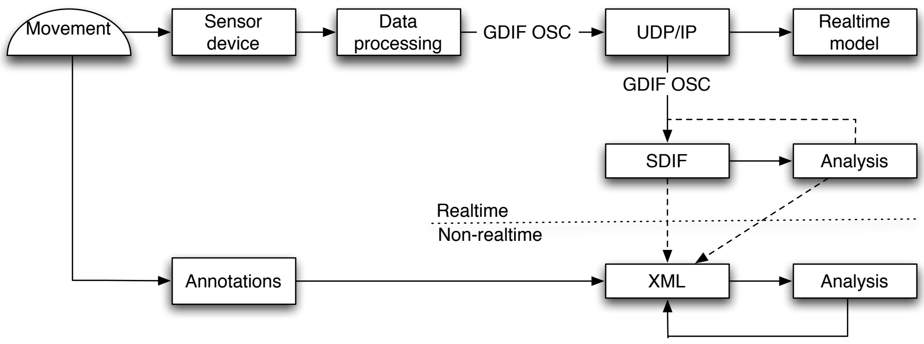


Figure 9.2: *Sketch of a setup using a GDIF OSC namespace for streaming, and storing these GDIF streams in SDIF files or structured XML files.*

The GDIF OSC streams can also be parsed and stored as streams in SDIF files, for example using IRCAM's FTM library[26] for Max (Schnell et al., 2005). This allows for synchronisation with audio, audio analysis and MIDI, and makes it possible to use the analytical tools available to SDIF to analyse GDIF data. For a more structured approach, and to be compatible with many other software solutions, we have conducted some tests to implement GDIF data in structured XML files. An example of this, in the context of analysing the movements in violin performance is discussed in Maestre et al. (2007).

The solution suggested in Figure 9.2 is but one of many ways to stream and store the data. In fact, my idea behind GDIF development is not so much about specifying *how* to stream and store data, but rather to create a framework and structure for *what* to store. This is the reason why I have mainly focused on understanding how different types of data are related, and how we can create solutions to structure them. A suggestion for a multilayered GDIF OSC namespace is illustrated in Figure 9.3, in which raw, pre-processed and analysed data are separated into different streams. The structure and properties of the different layers will be discussed in the following sections.

9.3.2 Acquisition Layers

The *acquisition* layers in Figure 9.3 refer to pre-analysed data: raw data coming from the device (*raw*) and pre-processed data (*cooked*).

Raw Layer

The *raw* layer is used to pass on raw data coming from the device. Here the idea is that the namespace should be constructed so that it reflects the way the data was received from the device. Game controllers, MIDI devices and other types of commercial controllers all have more or less well defined messages that can easily be turned into raw GDIF messages. For example, the namespace for a simple controller could be:

[26]http://freesoftware.ircam.fr/wiki/index.php/FTM

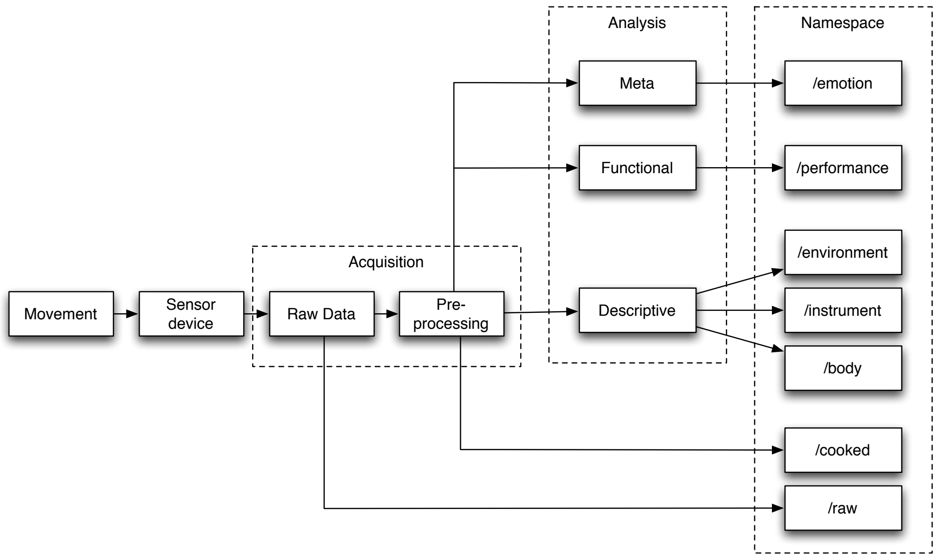


Figure 9.3: *Sketch of the multilayered approach suggested for the GDIF OSC namespace.*

```
/raw/name/<controller number> <value>
```

Similarly, data from a MIDI keyboard could be coded as:

```
/raw/keyboard/<note> <velocity> <channel>
```

It is more difficult when it comes to data from custom-built controllers, in which the values often depend on the sensor interface being used. Here it might be relevant to code the strings from either the device, the sensor interface or the sensor:

```
/raw
    /device/<sensor number> <value>
    /sensor-interface/<sensor number> <value>
    /sensor/<sensor name> <value>
```

This could result in messages like:

```
/raw
    /cheapstick/2 32
    /teabox/4 255
    /sensor/accelerometer 24 54 40
```

Such messages may not be particularly useful without some extra information about the device, but, as will be described later, the idea is to allow metadata to be available in the system to provide information about what the data mean.

Cooked Layer

The structure of the *cooked* layer is similar to the raw layer, except that the data may have been filtered (for example using a low-pass filter to remove noise in the signal), and/or scaled to some useful range. For example, data from a joystick may be coded as:

```
/cooked/joystick/<controller numer> <0. - 1.>
```

where <0.–1.> denotes that the values have been normalised to a decimal range. As such, the cooked data layer may not be particularly different from the raw layer, but I still find it important to separate the two. This is because data in the cooked layer is often what is used as the basis for further analysis. On the other hand, it might be important to have access to the raw data layer, in case there are problems relating to the pre-processing in the cooked layer.

An important thing to remember, is that all GDIF layers are optional and may be skipped if there is no need for them. The idea of having multiple layers is not to create large sets of redundant messaging, but rather suggest a structure for the way different types and levels of data could be communicated.

9.3.3 Descriptive Layers

The *descriptive* layers are used for coding data resulting from the analysis of the raw and cooked data layers. Here I find it important to clearly define which analytical perspective is used when describing such analytical data. This is why I suggest to split the descriptive layer into three separate layers: *device*, *body* and *environment*.

Device Layer

The *device* layer represents data analysed with respect to the sensor, instrument or controller used, totally independent of the performer and the performer's movements. The idea is that the device layer should be fairly generic, in the sense that devices with similar properties should be coded in the same way. This could be accomplished by creating a "device library", a collection of standardised namespaces for devices that share similar properties. For example, even though most commercial joysticks have a different number of buttons and look different, they still behave more or less similarly. Thus it is possible to create generic device specifications for various popular devices, for examples joysticks, gamepads, mouses, graphical tablets, etc. These specifications can then be used as the basis for all devices that have a similar (but not necessarily identical) functionality. This will allow for more flexibility and the possibility to interchange devices without having to worry about whether the raw data they output are identical.

One question in the device layer is how the values should be grouped. For example, the raw data from a joystick is usually transmitted as separate strings containing a control number and the associated value. But sometimes several such control numbers may be linked and dependent on each other, such as the two control numbers used to describe the tilt (x,y) of a joystick. And there is also the question about the rotation of the stick.

Should all these messages be passed together in one message, or be split into multiple messages? Wright et al. (2001) argued for the former, and suggested that data from a joystick could be represented as:

```
/joystick/b xtilt ytilt rotation ...
```

when button "b" is pressed. Here they even included information about the buttons in a long string containing a number of different messages. This is a computationally efficient way of passing the data, but the problem is that the end user will not necessarily know what the different values mean. It may also complicate working with the data, as it will require an **unpack** object in Max to retrieve the different values after the OSC string has been parsed. Even though it is less computationally efficient, I would prefer to only group messages that are clearly related, for example:

```
/device
       /joystick/tilt <x, y>
       /joystick/rotation <-1. - 1.>
```

An even more structured approach would be to pass on all messages as separate strings:

```
/device
       /joystick/button/b <1/0>
       /joystick/tilt/x <-1. - 1.>
       /joystick/tilt/y <-1. - 1.>
       /joystick/rotation <-1. - 1.>
```

This makes for more readable messages, and also allows for the whole message to be parsed using only OSC tools (for example **osc-route** or **jmod.oscroute** in Max). The downside is that it will increase the traffic and computational load in the system, since the number of communicated bits is much higher than for the combined messages.

Notice that the range used in the above example is -1.–1. This is because a joystick will typically have a centre position around which it can be tilted and rotated in either direction. Using a -1.–1. range will secure that the centre position is at (0, 0), which seems more intuitive than having a centre position at (0.5, 0.5). Notice also that the above example is hierarchically organised around the functional aspect (tilt) of the device, rather than the dimensional (x,y). In some cases, however, it may be better to do it the other way around, for example:

```
/device
       /joystick/x/tilt <-1. - 1.>
       /joystick/y/tilt <-1. - 1.>
```

Such structural differences in the namespaces will have to be discussed further, but I imagine that both solutions (and probably others) will be useful for different applications. However, we should be careful about allowing for too many different solutions, since one of the main goals of GDIF is to create a specification that will standardise the way messages are communicated.

As discussed in Chapter 7.6.1, an important issue in handling music-related movement data is the way units are used and defined. In general, I prefer to normalise values to a 0.–1. range, or -1.–1. where that is appropriate, but for some data it could be more relevant to use other units. For example, it may be practical to define the tilt or rotation of a joystick in degrees:

```
/device/joystick/rotation <0 - 360>
```

I think it is important to allow for the use of different types of units, but the question is whether the unit type should also be sent in the message:

```
/device/joystick/rotation <0 - 360> degrees
```

This is clearer, but as discussed earlier it may also lead to problems if several values are passed together.

Body Layer

The *body* layer describes the body and the body's movement in relation to the device, as seen "through the eyes" of the performer. While the device layer describes activity in terms of the different elements of the device (e.g. joystick and buttons), the body layer focuses on movement of parts of the body (e.g. hands and fingers). Such body movement may sometimes be directly inferred from the raw sensor data (e.g. using a joystick), while other times they will have to be more loosely described. An example namespace of the movements associated with a joystick could be:

```
/body
     /hand/right/finger/2/press <1/0>
     /hand/right/location/horizontal <-1. - 1.>
```

Notice that the namespace only refers to the body, and no information is available about the device. This opens for a more intuitive mapping process, and may allow for devices with similar movement properties to be exchanged unobtrusively. An example of this will be shown later in this chapter.

There are a number of issues that will need to be addressed when developing the specification of the body layer. First, what type of coordinate system(s) should be used to describe the body? What type of naming convention(s) should be used? Should values be defined absolute or relative to a specific point in the body? If so, where should the reference point be? I do not have answers to these questions yet, but for a prototype setup of control of spatialisation with hand movements (as described in Marshall et al. (2006)), we used a mirrored body model as shown in Figure 9.4. Here a positive movement space was defined for each hand in the forward, upward, outward *quadrant* of the body, which is typically where most of the movement takes place. The result was positive values when subjects moved their arms outwards, to the front, and upwards, something which simplified the mapping process in the setup.

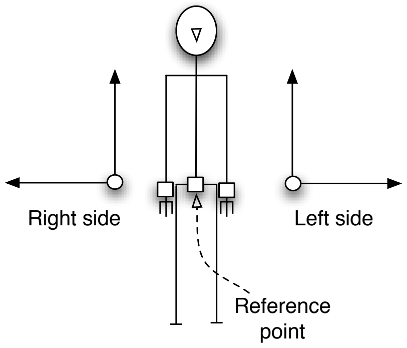


Figure 9.4: *Sketch of the mirrored and human-focused body model used in a prototype setup to control sound spatialisation with hand movements (Marshall et al., 2006). Notice that the body is split in two, and two positive quadrants are defined on either side.*

One reason why the mirrored body model turned out to be so easy to work with is probably because it reflects the way we think and talk about our body. For example, we usually refer to an arm movement as "outwards", and not "left" or "right". Thus creating such perceptually relevant namespaces is, in my opinion, important to allow for a more intuitive mapping process. That said, further development of the body layer should probably also look at various types of biomechanical models used in motion capture systems, and see how they can fit into such a perceptual and descriptive paradigm.

Environment Layer

The *environment* layer is intended to code information about the relationship between various bodies and devices in a space. For example, in the prototype spatialisation setup described above, the orientation of the person was used to control the location of sound in the space. Further development of this layer will be done in close collaboration with Nils Peters at McGill, who has recently begun developing the *Spatialisation Description Interchange Format* (SpatDIF).

9.3.4 Functional Layers

The *functional* layers will be independent of both the technical and body-related descriptors presented above, and focus on describing the functional aspects of the movement. This can be based on the taxonomy of music-related movements described in Chapter 4: sound-producing, sound-facilitating, sound-accompanying and communicative. It can also be based on traditional music performance vocabulary, such as describing the dynamics of the performance (e.g. *ppp* – *fff*) or the playing style (e.g. *staccato* or *legato*). There is still a lot of conceptual work to be done before these layers can be specified, and it will be interesting to collaborate with the developers of the *Performance Markup Language* (PML) to create suitable namespaces.

9.3.5 Meta Layers

The *meta* layers are intended to store data relating to higher level features, for example abstract representations, metaphors and emotional states. I believe it is important to create namespaces that allow for the representation of such descriptors, since this is the level at which we often talk about music. For example, it is common to refer to the "intensity" or "colour" of a sound, or describe the emotional response to the music. There is a growing body of research on these topics that could form the basis for creating relevant descriptors, for example by Bresin and Friberg (2000) and Camurri et al. (2003). They have been working on systems to analyse and model "expressive gestures" based on emotion parameters (such as anger, fear, grief and joy).

9.4 A Prototype Implementation

Several prototype GDIF setups have been implemented over the course of the project, but many of them have used special sensing devices such as the Polhemus trackers. I was therefore interested in creating a prototype that could be tested by people without such equipment, and turned to the simplest and most readily available device I could think of, the computer mouse. Here I will present how the **jmod.mouse** Jamoma module was transformed into a GDIF-compliant module called **jmod.mouse.gdif**.

9.4.1 jmod.mouse.gdif

The help file of **jmod.mouse.gdif** can be seen in Figure 9.5, and shows that the new GDIF-compliant module looks very similar to the original **jmod.mouse** module. The most notable difference is that the new module contains an inspector button (i) to open a window in which it is possible to choose which GDIF streams to output. If all the GDIF layers are to be streamed simultaneously there might be a lot of traffic and unnecessary parsing of data going on. To avoid such redundant information and processing, the idea is that the end user can choose which streams to receive.

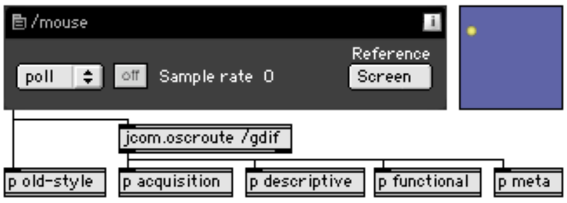


Figure 9.5: *The **jmod.mouse.gdif** module is similar to the original **jmod.mouse** module except for the inspector button (i) which opens a dialogue box in which it is possible to turn on and off GDIF streams.*

As the help patch shows, all data are outputted from the left outlet, formatted as OSC streams. I have chosen to leave the data output from the original module untouched to not break compatibility with the previous version of the module. The GDIF streams are routed to four subpatches showing the content of the streams (acquisition, description, functional, meta). The rest of this section will present these different streams.

Acquisition Layers

Figure 9.6 shows the subpatch parsing the acquisition layers. The *raw* layer contains the data as output from the **mousestate** object (which is at the heart of the module):

```
/gdif/raw
         /button <1/0>
         /location/horizontal <pixels>
         /location/vertical <pixels>
```

Using `/horizontal` and `/vertical` in this namespace is because these are the names of the outlets of **mousestate**. In this case these messages are meaningful, but there may be other cases, for example using a joystick, where the parameters in the raw layer are only numbered.

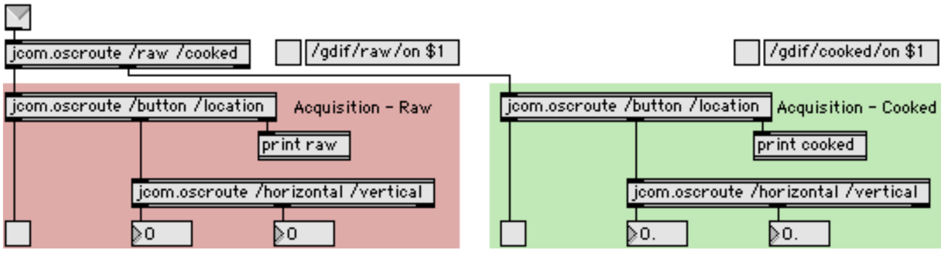

jcom.oscroute /raw /cooked
/gdif/raw/on $1
/gdif/cooked/on $1
jcom.oscroute /button /location
Acquisition - Raw
print raw
jcom.oscroute /horizontal /vertical
0
0
jcom.oscroute /button /location
Acquisition - Cooked
print cooked
jcom.oscroute /horizontal /vertical
0.
0.

Figure 9.6: *Subpatch parsing the raw and cooked data layers. Notice how the two streams can be turned on and off through OSC messages. The returned values use a namespace based on the names of the values returned from the* ***mousestate*** *object.*

The other half of the help patch shows the pre-processed data in the *cooked* layer, which are coded with the following namespace:

```
/gdif/cooked
            /button <1/0>
            /location/horizontal <0. - 1.>
            /location/vertical <0. - 1.>
```

Notice that the structure of these messages is similar to that of the raw layer. The only difference is that values have been normalised to a 0.–1. range based on dividing the location of the mouse (in pixels) by the dimensions of the computer screen (in pixels). The values have also been changed so that origo is now in the bottom left corner of the screen, as opposed to the upper left corner which is the origo of the raw data. This seems to be more in line with the way we would think about the position of the mouse.

Descriptive Layers

The *descriptive* data layers could be considered to be transformations of the cooked layer with respect to either the body of the performer, the device or the environment.

The *device* layer is focused around the functionality and elements of the device. The device layer is here implemented with the following namespace:

```
/gdif/device
            /mouse/button/1/press <1/0>
            /mouse/location/horizontal <-1. - 1.>
            /mouse/location/vertical <-1. - 1.>
```

Except for the different naming, this layer is fairly similar to both the raw and cooked layers. One important difference is the ranges used. As opposed to the cooked layer where values were normalised to a general 0.–1. range, I find it more practical to use a -1.–1. range for this device. This is because I imagine the origo being in the middle, and then movements going in either direction from that position.

The *body* descriptive layer, which describes the values from the perspective of the performer using the device, has the following namespace:

```
/gdif/body
          /hand/right/finger/2/press <1/0>
          /hand/right/location/horizontal <-1. - 1.>
          /hand/right/location/vertical <-1. - 1.>
          /hand/right/motion/quantity <-1. - 1.>
          /hand/right/motion/direction <0. - 360.>
```

Figure 9.7 shows how the data in the body layer can be parsed. Notice how the namespace is built progressively from larger to smaller features (body, hand, finger), and that the specifications of the hand (right) and finger (2) are parts of the namespace. Here numbers are used to indicate the fingers, but we should work towards a better and more uniform terminology of the various parts of the body. As the mapping example in the next section will show, this allows for *wildcards*[27] when parsing the values. This body layer will necessarily have to be subjective and context-dependent, so here the hand and finger that I use with the mouse have been chosen. Regarding the device layer, I also find it practical to use a -1.–1. range with the *home position* in the middle (0,0) of the *movement space*. In addition to the location, the body layer also displays the direction and quantity of motion, since it is often more interesting to use such movement data for control purposes. This is yet another way of making the returned values less dependent on the specific device being used, and rather focus on properties of the movement.

The *environment* layer does not contain any values in this example, as there are no defined relationships between the mouse, the performer or the environment in this setup.

[27] A *wildcard* character can be used to substitute other character(s) in a string. In this example */right* could be substituted to also allow for the left hand to be used.

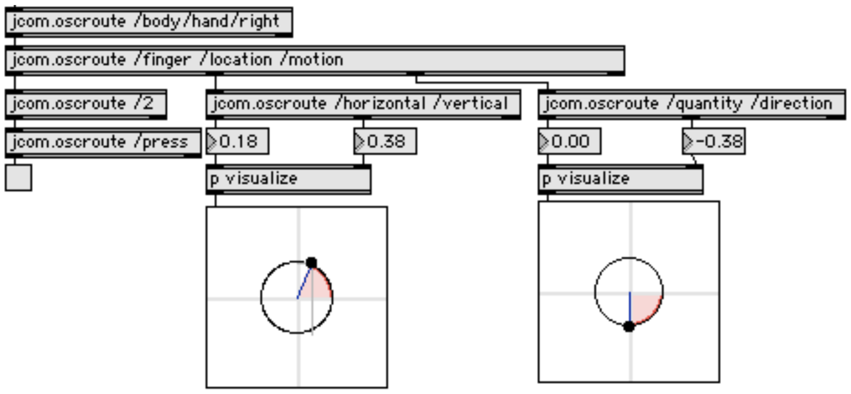

jcom.oscroute /body/hand/right
jcom.oscroute /finger /location /motion
jcom.oscroute /2
jcom.oscroute /horizontal /vertical
jcom.oscroute /quantity /direction
jcom.oscroute /press
0.18
0.38
0.00
-0.38
p visualize
p visualize

Figure 9.7: *A subpatch parsing the* body *descriptive layer.*

Functional and Meta Layers

The structure and descriptors of the functional and meta layers have yet to be defined. An important question in implementing these layers is whether the processing should be handled inside the module, or if it would be better to do it outside the module, possibly implemented in a separate module. The acquisition and description layers all mainly deal with various types of transformations of the same data, but require no particular analysis (except for when setting up the namespaces). The functional and meta layers, on the other hand, are based on various types of analyses that might be better to handle in specialised analysis modules.

9.4.2 Header Information

All of the above mentioned data types and streams refer to dynamic data. But it will also be necessary to have a solution for handling "static" information that can be stored as header information at the beginning of a file, or be queried for in realtime implementations using a `/get` message. First, general information about the location and the experimental setup should be defined, for example:

```
/general
        /date 2007.01.01
        /time 12:44:00 CET
        /location "Musical Gestures Lab, University of Oslo"
        /author "Alexander Refsum Jensenius"
        /experiment "Mouse-Joystick"
        /session 1
        /devices "Trust GM-4200 and Saitek ST290"
        /description "Testing if users prefer a mouse
                      over a joystick in the control of
                      a VST soft synth running in Max."
```

It may also be necessary to include more specific information about the devices listed in the general information. They could be defined based on the generic device type they represent, such as:

```
/device
       /type mouse
       /name GM-4200
       /manufacturer Trust
       /id S12116847
       /dof 3
       /buttons 5
       /description "A right handed computer mouse"
```

The idea is that all relevant information should be available at any time. In realtime situations this may allow for networked setups in which a new user could browse the network for available devices, find their properties and set them up for control. These data could also be written in the headers of the stored files, so that it is possible to identify what the data mean, how they were recorded, and which devices they were recorded from. Creating a clear structure for such extra information may be one of the greatest strengths of the GDIF specification.

9.4.3 Flexible Mapping

One of the main ideas behind the realtime implementation of GDIF is to allow for the quicker and easier creation of mappings between controllers and sound engines. It might be hard to see how this is the case considering all the time and effort spent creating the mouse module presented above. In this example it would clearly have been easier and quicker to create mappings manually from the outputs of a **mousestate** object. However, in larger and more complex setups I believe the structure of GDIF will, in fact, facilitate the creation of more flexible systems that may also feel liberating to work with.

I have been working at several different music technology laboratories over the last years, and one recurring problem seems to be connecting various input devices to different sound engines. A typical laboratory contains a large number of controllers, synthesisers and computers, and ideally GDIF would allow for setups to be created easily where anything can be connected to anything in any way, anytime.

The mapping modules available in Jamoma greatly simplify creating mappings, but an even more flexible approach is suggested by Malloch et al. (2007). Here the idea is to use a mapping patch that will automatically list all the available devices and their namespaces, and allow for mapping between controllers and sound engines by dragging lines between the various parameters. With the flexibility of the many layers in GDIF, many of which will be based on multidimensional transformations, this will allow for a flexible mapping process.

When creating a mapper that works in a multi-device/computer setup, it is important to figure out how to send data easily and efficiently between computers. As shown in the prototype implementation of **jmod.mouse.gdif**, the different GDIF streams can be turned on and off at will. This is particularly important when working with many devices that communicate on a computer network. In our prototype setups, we have tested using *multicast* UDP for communication, which allows for multiple *clients* to connect to a *server* from which the GDIF streams are transmitted. This is more flexible than having to set up separate connections between IP ports, and it avoids flooding the network with data using *broadcast* messaging.[28]

To simplify creating network connections, we have tested using *Zero Configuration Networking* (*ZeroConf*)[29] techniques for automatic discovery of addresses and namespaces on the network. ZeroConf handles port allocation automatically and makes it easy to distribute information about clients that are available on the network without having to know anything about the network settings (IP addresses etc.). On Mac OS X, ZeroConf is implemented in *Bonjour*, and we have had success using Rémy Mueller's **OSCBonjour** external[30] to use ZeroConf in communication between Max patches. ZeroConf and other networking solutions is something we will have to explore in future development of GDIF.

9.4.4 Meaningful Mapping

The possibility to create flexible mappings is important, but even more important I believe it is to work towards solutions for creating *meaningful mappings*. This is something which was explored in ESCHER, a modular system where the descriptions of the movements carried out with a controller are separate from the parameters of the synthesis engine, and where mappings are created with an intermediate abstract parameter level between the movement and the sound parameter levels (Wanderley et al., 1998). These ideas have later been developed into models focusing on using several perceptual mapping layers: one layer for movements, one for sound and one between movement and sound (Hunt and Wanderley, 2002; Arfib et al., 2002). This makes it possible to create mappings between actions and sounds based on perceptually relevant features, rather than technical parameters.

Creating mappings based on meaningful parameters would be closer to how composers and performers usually think about movement and sound, for example in relation to the body: "I want a forward movement of my right hand to control the brightness of the sound" or "I want to control granularity in the sound with this gentle finger tapping". Rather than forcing everyone into the constraints of the technical system, we should try to create setups that make it possible for mappings to be created at such a meaningful level. If the idea is to have a forward movement in the right hand to control the brightness of the sound, it might not be important whether that movement is done with a bow, a joystick or a novel controller, since the quality of the movement will be the same. If the mappings are created at this level, it would also be possible to retain the mappings even though either the controller or the sound system (or both) were changed.

[28]Sending data to all network ports on the local network.
[29]http://www.zeroconf.org
[30]http://recherche.ircam.fr/equipes/temps-reel/movement/muller/

A very simple example of such a meaningful mapping process is implemented in the *mouse-joystick-mapping.mxt* patch shown in Figure 9.8. This example patch retrieves data from two devices, a computer mouse and a joystick, and codes the data into *body* layer GDIF streams. The point of the patch is to show that it is possible to use either device to control a sound synthesis model, and to switch between them without having to create any new mappings between the control values and the parameters in the sound synthesis model. This is because the mappings are created at a body level, and only refer to the movement of the hand used in the interaction. Here both the mouse and joystick are controlled by the right hand, and they both allow for horizontal and vertical movement of the hand. It is therefore possible to use these body movement features to control the sound model. This is but a simple example, but I believe it can be possible to create similar meaningful mappings between multi-dimensional controllers and sound engines.

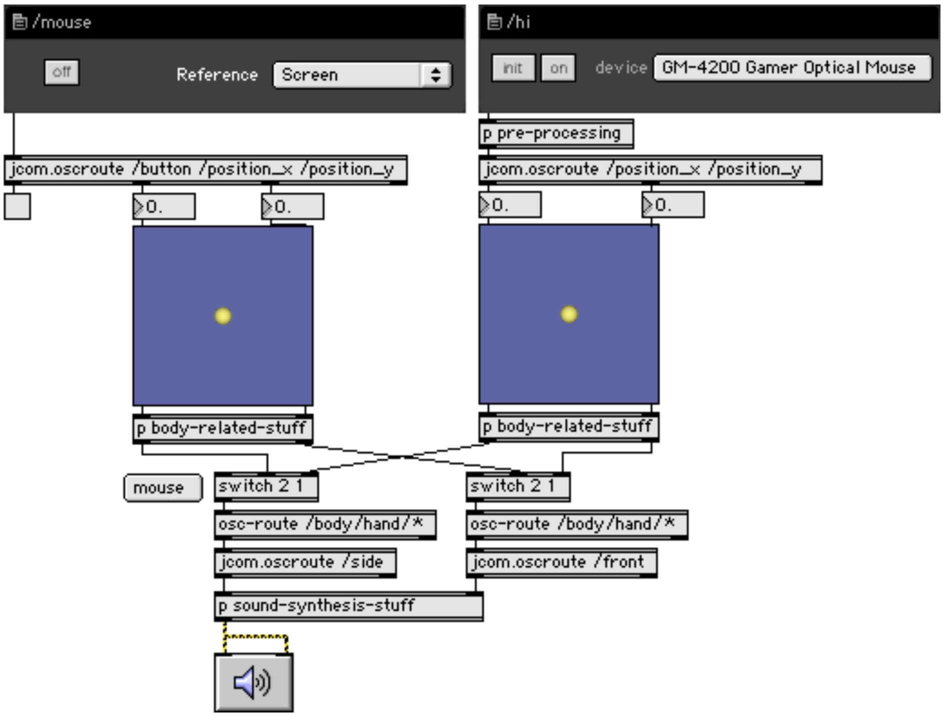


Figure 9.8: *An example of mapping movements from either mouse or joystick to a simple sound synthesis using a GDIF namespace. Since the values from both devices are coded using the same body-related descriptors is is possible to switch between the devices without having to create new mappings. Note that a star (*) is used as a wildcard for choosing either left or right hand movements.*

9.5 Summary

GDIF development started as an attempt to create a solution to store data from our observation studies, but has ended up as a more general purpose specification for handling both streaming and storage of all sorts of music-related movement data. The current implementation of GDIF has mainly focused on creating OSC namespaces and testing them in realtime control. Here there are several challenges that will have to be addressed in future development, including:

- *namespace*: the namespaces presented here are still under continuous revision, and will hopefully slowly develop into a more solid and general structure.
- *device library*: creating a library of various types of devices will open for creating generic namespaces for similar devices.
- *units*: units should ideally be handled through a unit library, which could also be used for automatic conversion between the units.
- *handshaking*: we need to develop a robust system to make devices "talk" to each other and interchange information (i.e. "handshaking"). This can be done by including enough static information that describe the features of the devices.
- *efficiency*: the long OSC namespaces used in the current GDIF implementation make a GDIF-compliant system fairly inefficient in terms of computer processing. We will therefore have to investigate various solutions to optimise the data flow, one possibility being to use *OSC aliasing*, something which was suggested in a draft OSC 2.0 specification during NIME 2006 (Jazzmutant, 2006).
- *tactility*: the current GDIF specification only focuses on control and movement aspects, but we also need to find solutions to handle information about feedback in haptic devices. Here it is possible to build on a suggestion for an OSC namespace for haptic devices presented by Sinclair and Wanderley (2007).

In addition to these challenges, there are several problems related to storage and synchronisation. Obviously, all of this is far beyond what I can accomplish alone, and I am not sure if GDIF will ever succeed in becoming anything more than a proposal and a prototype implementation. Nevertheless, the process has so far led to a number of conceptual and practical questions and challenges that in themselves are worth the effort. As such, GDIF development is not only about creating a format and tools, but is also about trying to define a vocabulary for music-related movement in general. Fortunately, in recent years I have experienced that many researchers are interested in GDIF, something which makes me believe that it may, indeed, be possible to succeed in creating a common format to stream and store music-related movement data.

CHAPTER 10 Postlude

You have to have an idea of what you are going to do, but it should be a vague idea.

Pablo Picasso

This chapter summarises the dissertation, comments on the relevance of the results, reflects on the research process, and presents some possibilities for future research and development.

10.1 Summary

The background for this dissertation project was the observation that body movement is important for both performance and perception of music. Compared to many other topics, music-related movement has received surprisingly little attention in musicology and other music research disciplines. For this reason I found it necessary to have an open perspective, and carry out research of a more exploratory character. The main goal has been to understand more about music-related movements and how such movements are important for the performance and perception of music. This has been achieved through theoretical studies, observation studies and technological development.

The three theory chapters (2, 3, 4) laid the ground for further exploration and development. Chapter 2 presented an overview of the emerging field of *embodied music cognition*, including an account of the importance of a *multimodal* approach to music perception and cognition. This overview included research suggesting that human perception and cognition is an active process, something which is supported by neurophysiological findings of *mirror neurons* in the brain. Research on *audiovisual mirror neurons* are particularly interesting from a musical point of view, since they suggest that motor areas of the brain are activated when only imagining an action, and that a sound stimulus may evoke the same type of motor activation in the brain as if the *sound-producing action* had been observed.

Chapter 2 formed the basis for my discussion of *action-sound relationships* in Chapter 3. Here I claimed that our life-long experience with *action-sound couplings* (i.e. *natural* action-sound relationships) guide our perception of both actions and sounds. My ideas of action-sound couplings are based on several of Godøy's models of relationships between actions and sounds (Godøy, 1997, 2001, 2006). He suggested that we can hardly see a sound-producing action without mentally "hearing" some kind of resultant sound, and similarly that hearing only a sound will usually evoke a multimodal sensation of how the sound was produced. I claimed that ecological knowledge of action-sound couplings also guide our perception of *artificial* action-sound relationships as found in *electronic devices* and *virtual realities*. Thus, if we want to create *practical* (i.e. intuitive) artificial action-sound relationships, they should be modelled after action-sound couplings that have similar properties to that of the action-sound relationship being designed.

Chapter 4 started with a discussion of the term *gesture*, and the many different gesture definitions used in various disciplines. My conclusion here was to discontinue using the term gesture, since it often leads to more confusion than clarification. Rather, I have decided to use the term *movement* to denote the continuous stream of displacement of an object or limb, and the term *action* to denote goal-directed and/or coherent chunks of movement. The rest of Chapter 4 discussed terminology and a classification of music-related movements, exemplified with excerpts from real performances taken from commercially available music DVDs.

Ideas from the theoretical chapters were exemplified in the chapters on exploratory studies. Chapter 5 presented three observation studies of *music-movement correspondences*: *air piano performance*, *free dance* to music, and *sound-tracing*. These studies confirmed our assumption that body movement is, indeed, an integral part of the perception of music, for novices and experts alike. We found that even people who claimed themselves to be "unmusical" performed well when it came to general movement-music correspondences, though they did not show the ability to follow more detailed features of the musical sound such as found in the performances of expert subjects. In the sound-tracing studies similar phenomena were seen in how experts tended to identify separate sonorous objects in composite sounds with more details than novices.

Chapter 6 explored the concept of action-sound couplings and relationships through the design and construction of electronic *music controllers* and *instruments*. The underlying idea was to create low-cost and human-friendly devices for musical expression. Several different types of systems were presented, starting with *MultiControl*, a software for using game controllers for musical applications. Then followed a presentation of *Cheapstick*, a low-cost digital music controller built from a modified game controller and homemade, paper and latex sensors. The concept of low-cost and homemade controllers was further explored in the development of various acoustic, electronic and electroacoustic *music balls*. Here I found that the electroacoustic music balls were the most interesting, considering that they allowed for both natural and artificial action-sound relationships. This made them both intuitive, and creatively interesting at the same time. Ideas from the development of the music balls were continued in the *Music Troll* project, an installation piece exploring how it is possible to create a large music controller that is intuitive and easy to use.

The three last chapters (7, 8, 9) introduced technologies that have been developed to meet the needs presented in the previous chapters. The *Musical Gestures Toolbox* (MGT) presented in Chapter 7, is a collection of software tools for working with music-related movement in various ways. MGT is developed for and with the graphical programming environment Max/MSP/Jitter, and is currently included in the open source *Jamoma* distribution. The main focus of MGT has been on developing solutions for qualitative video analysis, but I have also developed modules for working with different input devices. Developing these tools has involved working on basic issues of data flow, communication, and mapping in music programming, something which I have carried out in close collaboration with the other Jamoma developers.

Chapter 8 focused on exploring movement visualisation techniques for the video material from our observation studies. Inspired by experiments from early photography, I have explored techniques for creating *motion history images* that allow for seeing the trajectories of short movement sequences (up to 10 seconds). For longer sequences (from minutes to hours) I have developed a technique for creating *motiongrams* and *videograms* based on plotting reduced matrices over time. Motiongrams and videograms can simplify navigating in large collections of video material, and for comparative analysis of various types of video material.

Chapter 9 presented the current state of the *Gesture Description Interchange Format* (GDIF). GDIF is being developed to solve problems of streaming and storing music-related movement data. Throughout the project I have constantly had problems with synchronising data from various devices, and to store associated analytical data. Having a common format for handling music-related movement data may simplify working with various types of devices, and may also open for more collaborative projects. While still at an early stage of development, GDIF has attracted the interest of several researchers and institutions, and is currently a joint development project. Regardless of whether GDIF succeeds in becoming a widespread format or not, the development process has been, and hopefully will continue to be, valuable for the developers. This is because the development process makes it necessary to formalise descriptors, and raises awareness of important challenges for future research on music-related movement.

10.2 Relevance

The ideas, models, methods and tools presented in this dissertation may be relevant for a number of different fields. First, the discussions of action-sound couplings and relationships are important for a better understanding of music perception and cognition in general, and more specifically as a contribution to the emerging field of embodied music cognition. The ideas of action-sound couplings should also be of great interest to fields such as computer music, interaction design, and human-computer interaction. These fields are heavily reliant on both actions and sounds, and may benefit from more theoretical and practical knowledge of action-sound couplings and movement-music correspondences. Even a field like *music information retrieval*[1] could benefit from exploiting knowledge about action-sound couplings in data mining and retrieval.

The software applications and tools created in the project are still in development, but are already in use by a number of people. MultiControl has been, and continues to be, used in a number of workshops, installations and concerts, and is regularly downloaded from our web page.[2] MGT and Jamoma have started to receive widespread attention, and are used for both research and performance throughout the world. My movement visualisation techniques are used in various contexts, ranging from the projection of motion history images in interactive video performances, to using motiongrams for studying the movements of rats in medical experiments.

Finally, GDIF has caught the interest of several researchers and institutions, including the EU Cost Action 287 ConGAS. Hopefully, GDIF development can also become the starting point for creating shared research databases of music-related movement data, data that can form the basis for more collaborative projects in the future. GDIF may also potentially become, or inspire, a general solution for communication between music controllers and sound engines.

10.3 Reflections

This project has been highly interdisciplinary, covering theories and methods from a number of different fields, including musicology, psychology, neuroscience, informatics, and human-computer interaction. This interdisciplinary approach has been a great inspiration, since I have had the opportunity to draw on a large body of established methods and tools. Working at the intersection of several disciplines has also been a challenge, and I have constantly seen my limitations and lack of in-depth knowledge of many of the topics I have been dealing with. Sometimes I have even wondered in which field this dissertation actually belongs. Officially, my project has been carried out in a department of musicology and within a faculty of humanities, but in many ways the outcome is closer to what could be found in psychology or computer science.

One question I have often received during my project is why I am spending time making hardware and software tools rather than using existing solutions? The reason is simple: there are no tools available that adequately fulfil my needs. It is important to remember that technology is never neutral, and that the ideas behind the development of a tool will always control the way the tool can be used. I would rather turn the question around: why are not more music researchers developing their own tools rather than relying on other people's solutions? For me, development does not only result in a tool, it may also be seen as an answer to a question, and it usually involves rephrasing the question along the way. This may be seen as similar to how Trond Lossius (2007, 8) summarised his research fellowship in the arts:

[1]Music information retrieval is concerned with finding specific information in music collections.
[2]http://musicalgestures.uio.no

> It is not only a question of developing tools. [..] Programming code becomes a meta-medium, and creating the program is creating the art work.

As such, developing research tools can be seen as developing both theories and methods. That said, considering the development carried out in this project it is clear that my dissertation is in *music*, and not one in computer science. My development has never been made for the sake of technology, but has always been driven by the musical questions at hand, either scientifically or creatively. That is the reason why my software and hardware all carry the tag "in development" or "prototype". The tools have been developed for exploring musical concepts and ideas, and have not been intended for being efficient or ready for production.

Throughout the project I have also been interested in exploring new methods for documentation and "writing". Academic publication traditions are strongly text-based, something which I have found problematic when working with musical sound and movement. Describing sounds and movements with words do not do justice to the richness of the original material. This is one of the reasons I have spent a great deal of time and effort on developing solutions for creating visual displays of movements, displays that can either be shown on screen or printed on paper.

I have also tried to explore new forms of communication during the project. My research has been carried out on two continents, and this made it necessary to find solutions for efficiently disseminating ideas with colleagues in both places. I have found it practical to use my *blog*[3] for quickly presenting ideas I have been working on, as well as commenting on other people's ideas. This has been an effective way of storing my own (random) thoughts chronologically, and has made me a participant in a global network of other research bloggers from various fields. I have also spent a great deal of time on reading and posting to various mailing lists. This daily international and interdisciplinary research collaboration has been an important platform from which I have been able to discuss ideas, and to get immediate feedback from others. Such quick and informal, yet scientific, communication may very well change the way research is carried and presented out in the years to come.

10.4 Future Research

This dissertation has only scratched the surface of a number of topics that can all be explored further in future research. First, there are very many open questions related to our perception and cognition of action-sound couplings and movement-music correspondences. One important question here is how we are able to chunk continuous streams of movement and sound into units. Another burning question is how the combination of actions and sounds mutually influence our perception of music. It is also essential to continue the development of conceptual frameworks and classification systems for sounds and movements, and formalise theories that can be tested in future observation studies and experiments.

[3] http://www.arj.no

Specifically, it would be interesting to carry out more observation studies of movement-music correspondences along the lines of the studies presented in this dissertation, but using for example a full-body motion capture system. More controlled action-sound experiments could also be carried out, for example using haptic devices to investigate the influence of haptic feedback on our perception of action-sound relationships. For all such studies and experiments, I think we should work towards creating shared collections of music-related movement data (coded with GDIF) that can be used as reference material in the research community.

A controlled laboratory environment is practical for observation studies and experiments, but I also find it important to develop solutions to carry out studies of real-life situations. For example, motion capture recordings of stage performances could facilitate the study of musicians in their "real" environment. Similarly, I am interested in carrying out studies of movement-music correspondences in everyday life, for example by recording the movements of people when they are listening to their mobile music players. This could reveal many interesting aspects of how we move to music.

When it comes to software development, there are many ways of taking the results from this dissertation further. MGT can easily be extended in several directions, for example by adding more computer vision tools. It will also be interesting to see whether implementing the video modules in OpenGL will alleviate the processing load. In the future development of Jamoma I will focus on implementing a unit library that can allow for more elegant and flexible mappings between controllers and sound engines. Another important topic is to work towards a clearer separation between interface and algorithm, something that can open for more flexible multi-computer setups. This also requires further GDIF development to create better communication between various systems.

Developing better movement visualisation techniques is important to analyse the data from observation studies. In this field I will continue to explore visual displays that focus on perceptually relevant features, and try to create solutions for extracting actions from continuous streams of movement. I will also develop more advanced motiongram techniques, for example creating three-dimensional motiongrams, and improve the efficiency of the algorithms for calculating the motiongrams.

Finally, I believe that many of the topics presented in this dissertation may be useful for music creation. I am particularly interested in exploring how movements may be used to generate and control musical sound in new ways. This may lead to music systems where the musical sound is continuously adapting to the listener's preferences, based on the movements of the listener. That way music-related movement may be used to generate body-controlled musical sound, which brings me back to the opening of the dissertation: music is movement.

BIBLIOGRAPHY

Agre, P. (1997). *Computation and Human Experience*. Cambridge: Cambridge University Press. 135

Allison, J. and T. Place (2003). Sensorbox: practical audio interface for gestural performance. In *NIME '03: Proceedings of the 2003 International Conference on New Interfaces for Musical Expression*, Montreal, Canada, pp. 208–210. Montreal: McGill University. 112

Allison, J. T. and T. A. Place (2005). Teabox: a sensor data interface system. In *NIME '05: Proceedings of the 2005 International Conference on New Interfaces for Musical Expression*, Vancouver, BC, pp. 56–59. Vancouver: University of British Columbia. 112

Allport, D., B. Antonis, and P. Reynolds (1972). On the division of attention: A disproof of the single channel hypothesis. *Quarterly Journal of Experimental Psychology 24*, 225–235. 15

American Psychiatric Association (1994). *Diagnostic and statistical manual of mental disorders. DSM-IV* (Fourth ed.). Washington, DC: American Psychiatric Publishing. 200

Applegate, B., B. B. Lahey, E. L. Hart, J. Biederman, G. W. Hynd, R. Barkley, T. Ollendick, P. Frick, L. Greenhill, K. McBurnett, J. H. Newcorn, L. Kerdyk, B. Garfinkel, I. Waldman, and D. Shaffer (1997). Validity of the age-of-onset criterion for ADHD: a report from the DSM-IV field trials. *Journal of the American Academy of Child and Adolescent Psychiatry 36*(9), 1211–1221. 199

Arfib, D., J.-M. Couturier, L. Kessours, and V. Verfaille (2002). Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound 7*(2), 135–152. 224

Aschoff, J. (1979). Circadian rhythms: general features and endocrinological aspects. In D. T. Krieger (Ed.), *Endocrine rhythms*, pp. 1–61. New York: Raven Press. 50

Austbø, H. (1989). Scriabin: Piano sonatas vol. 1. [Audio-CD] Simax PSC 1055. 65

Austbø, H. (1993). Messiaen: Vingt regards sur l'enfant-Jésus. [Audio-CD] Naxos 8.550829-30. 65

Austvoll, A. (2005, December). Personal communication. 80

Bach, J. S. (1717). Violin Concerto In A Minor, BWV 1041. 195, 196

Ballas, J. A. (1993). Common factors in the identification of an assortment of brief everyday sounds. *Journal of Experimental Psychology: Human Perception and Performance 19*(2), 250–267. 25

Barlow, W. (1975). *Alexander-princippet*. Borgen forlag. 79

Beard, S. and D. Reid (2002). MetaFace and VHML: A first implementation of the virtual human markup language. In *AAMAS workshop on Embodied Conversational Agents-let's specify and evaluate them*. 207

Berthoz, A. (1997). *Le sens du mouvement*. Paris: Odile Jacob. 13

Bevilacqua, F., R. Müller, and N. Schnell (2005). MnM: a Max/MSP mapping toolbox. In *NIME '05: Proceedings of the 2005 International Conference on New Interfaces for Musical Expression*, Vancouver, BC, pp. 85–88. Vancouver: University of British Columbia. 101

Birnbaum, D., R. Fiebrink, J. Malloch, and M. M. Wanderley (2005). Towards a dimension space for musical artifacts. In *NIME '05: Proceedings of the 2005 International Conference on New Interfaces for Musical Expression*, Vancouver, BC, pp. 192–195. Vancouver: University of British Columbia. 47

Bjørkvold, J.-R. (1989). *Det musiske menneske*. Oslo: Freidig forlag. 12

Blake, R. and M. Shiffrar (2007). Perception of human motion. *Annual Review of Psychology 58*, 47–73. 175

Bobick, A. F. and J. W. Davis (2001, March). The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*(3), 257–267. 183

Boersma, P. and D. Weenink (2001). Praat, a system for doing phonetics by computer. *Glot International 5*(9/10), 341–345. 75

Boethius, A. M. S. (ca 500/1989). *Fundamentals of Music: Anicius Manlius Severinus Boethius*. Palisca, Claude V. (Editor) and Bower, Calvin M. (Translator), New Haven: Yale University Press. 61

Bongers, B. (2000). Physical interfaces in the electronic arts : Interaction theory and interfacing techniques for real-time performance. In M. M. Wanderley and M. Battier (Eds.), *Trends in Gestural Control of Music*, pp. 41–70. Paris: IRCAM – Centre Pompidou. 100

Boulez, P. (2000). Musik Trienniale Koln 2000: Berg Lulu Suite, Debussy Le Jet D'Eau, Stravinsky Firebird (Pierre Boulez, Chicago Symphony Orchestra). [DVD] Image Entertainment. 197, 198

Bradsky, G. and J. Davis (2002, July). Motion segmentation and pose recognition with motion history gradients. *Machine Vision and Applications 13*(3), 174–184. 183

Brahms, J. (1894). Clarinet sonata op.120, no.1 in f minor. 194

Braun, M. (1992). *Picturing Time. The work of Etienne-Jules Marey (1830-1904)*. Chicago, IL: The University of Chicago Press. 175

Bregman, A. S. (1990). *Auditory Scene Analysis. The Perceptual Organization of Sound*. Cambridge, MA. & London: The MIT Press. 12

Bresin, R. and A. Friberg (2000). Emotional coloring of computer-controlled music performances. *Computer Music Journal 24*(4), 44–63. 219

Browman, C. P. and L. Goldstein (1989). Articulatory gestures as phonological units. *Phonology 6*, 201–251. 18

Cabe, P. A. and J. B. Pittenger (2000). Human sensitivity to acoustic information from vessel filling. *Journal of Experimental Psychology: Human Perception and Performance 26*(1), 313–324. 26

Cadoz, C. (1988). Instrumental gesture and musical composition. In *Proceedings of the 1998 International Computer Music Conference*, Den Haag, Netherlands, pp. 60–73. 24, 39, 42, 46, 47, 49

Cadoz, C., A. Luciani, and J. Florens (1984). Responsive input devices and sound synthesis by simulation of instrumental mechanisms: The cordis system. *Computer Music Journal 8*(3), 60–73. 102

Cadoz, C. and M. M. Wanderley (2000). Gesture – Music. In M. M. Wanderley and M. Battier (Eds.), *Trends in Gestural Control of Music [CD-ROM]*, pp. 71–94. Paris: IRCAM – Centre Pompidou. 41, 47

Campbell, L., M.-J. Chagnon, and M. M. Wanderley (2005). On the use of Laban-Bartenieff techniques to describe ancillary gestures of clarinetists. Research report, Input Devices and Music Interaction Laboratory, McGill University. 50

Camurri, A., P. Coletta, A. Massari, B. Mazzarino, M. Peri, M. Ricchetti, A. Ricci, and G. Volpe (2004). Toward real-time multimodal processing: Eyesweb 4.0. In *AISB 2004 Convention: Motion, Emotion and Cognition*, Leeds, UK. 136

Camurri, A., G. De Poli, M. Leman, and G. Volpe (2001). A multi-layered conceptual framework for expressive gesture applications. In *Proceedings of the International MOSART Workshop, November 2001*, Barcelona, Spain. 39

Camurri, A., I. Lagerlof, and G. Volpe (2003, July). Recognizing emotion from dance movement: comparison of spectator recognition and automated techniques. *International Journal of Human-Computer Studies 59*(1-2), 213–225. 183, 219

Camurri, A., B. Mazzarino, and G. Volpe (2004). Expressive gestural control of sound and visual output in multimodal interactive systems. In *Proceedings of the 2004 International Conference Sound and Music Computing*, Paris, France, pp. 38–44. Paris: IRCAM – Centre Pompidou. 68

Camurri, A., M. Ricchetti, and R. Trocca (1999). Eyesweb - toward gesture and affect recognition in dance/music interactive systems. In *IEEE Multimedia Sustems*, Firenze, Italy. 66, 136

Camurri, A., R. Trocca, and G. Volpe (2002). Interactive systems design: A kansei-based approach. In *NIME '02: Proceedings of the 2002 International Conference on New Interfaces for Musical Expression*, Dublin, Ireland, pp. 1–8. Dublin: Media Lab Europe. 66

Cantwell, D. P. (1985). Hyperactive children have grown up. What have we learned about what happens to them? *Archives of General Psychiatry 42*(10), 1026–1028. 199

Carello, C., K. L. Anderson, and A. J. Kunkler-Peck (1998). Perception of object length by sound. *Psychological Science 9*(3), 211–214. 26

Carello, C., J. B. Wagman, and M. T. Turvey (2005). Acoustic specification of object properties. In J. Anderson and B. Anderson (Eds.), *Moving image theory: Ecological considerations*, pp. 79–104. Carbondale, IL: Southern Illinois University Press. 26

Casciato, C., A. R. Jensenius, and M. M. Wanderley (2005). Studying free dance movement to music. In *Proceedings of ESCOM 2005 Performance Matters! Conference*, Porto, Portugal. 77

Chung, H. and Y. Lee (2004). MCML: motion capture markup language for integration of heterogeneous motion capture data. *Computer Standards & Interfaces 26*(2), 113–130. 207

Clapton, E. (1999). Eric Clapton & Friends in Concert: A Benefit for the Crossroads Centre in Antigua. [DVD] Warner Bros (1999). 55, 56

Clark, M. R. (2002). *Singing, Acting, and Movement in Opera : A Guide to Singer-getics*. Bloomington, IN: Indiana University Press. 35

Clarke, E. (1993). Generativity, mimesis and the human body in music performance. *Contemporary Music Review 9*(1), 207–219. 42

Clarke, E. (1999). Rhythm and timing in music. In D. Deutsch (Ed.), *The Psychology of Music* (2nd ed.)., pp. 473–500. San Diego: Academic Press. 50

Clarke, E. F. (2005). *Ways of Listening: An Ecological Approach to the Perception of Musical Meaning*. Oxford: Oxford University Press. 12

Clayton, M., R. Sager, and U. Will (2005). In time with the music: the concept of entrainment and its significance for ethnomusicology. In *European Meetings in Ethnomusicology (ESEM Counterpoint 1)*, pp. 3–75. 50

Clynes, M. (Ed.) (1982). *Music, Mind, and Brain: The Neuropsychology of Music*. Berlin Heidelberg: Springer-Verlag. 2

Coker, W. (1972). *Music & Meaning: A Theoretical Introduction to Musical Aesthetics*. New York: Free Press. 40

Collins, J. (2006a). Musical Symbolism in Ghana (Lecture). University of Oslo, 16 March 2006. 11

Collins, N. (2006b). *Handmade Electronic Music: The Art of Hardware Hacking*. New York: Routledge. 113

Cook, P. (1997). Physically Informed Sonic Modeling (PhISM): Synthesis of Percussive Sounds. *Computer Music Journal 21*(3), 38–49. 120

Cook, P. (2001). Principles for designing computer music controllers. In *NIME '01: Proceedings of the 2001 International Conference on New Interfaces for Musical Expression*, Seattle, WA. New York: ACM Press. 102, 120

Cook, P. and G. Scavone (1999). The Synthesis ToolKit (STK). In *Proceedings of the 1999 International Computer Music Conference*, Beijing, China, pp. 164–166. San Francisco: ICMA. 120

Coppola, F. F. (1979). Apocalypse now. [DVD] Paramount. 31

Dahl, S. (2005). *On the beat: Human movement and timing in the production and perception of music*. Ph. D. thesis, Royal Institute of Technology, Stockholm, Sweden. 24

Dahl, S. and A. Friberg (2004). Expressiveness of musician's body movements in performances on marimba. In A. Camurri and G. Volpe (Eds.), *Gesture-Based Communication in Human-Computer Interaction, 5th International Gesture Workshop, GW 2003, Genova, Italy, April 15-17, 2003, Selected Revised Papers*, Volume LNAI 2915, pp. 479–486. Berlin Heidelberg: Springer-Verlag. 17

Dahlstedt, P. (2001). Creating and exploring huge parameter spaces: Interactive evolution as a tool for sound generation. In *Proceedings of the 2001 International Computer Music Conference*, Habana, Cuba, pp. 235–242. San Francisco: ICMA. 101

Darwin, C. (1872). *The Expression of the Emotions in Man and Animals*. London: John Murray. 2

Davidson, J. (1993). Visual perception and performance manner in the movements of solo musicians. *Psychology of Music 21*, 103–113. 42

Davidson, J. (1994). Expressive movements in musical performance. In *Proceedings of the Third International Conference on Music Cognition (ESCOM)*, Liege, Belgium, pp. 327–329. 42

De Carolis, B., C. Pelachaud, I. Poggi, and M. Steedman (2004). APML, a mark-up language for believable behavior generation. In H. Prendinger and M. Ishizuka (Eds.), *Life-like Characters. Tools, Affective Functions and Applications*, pp. 65–85. Berlin Heidelberg: Springer-Verlag. 207

DeGroot, D. (2005). Modeling and simulating the brain as a system. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, Washington, DC, pp. 3. IEEE Computer Society. 15

Delalande, F. (1988). La gestique de gould: Élements pour une sémiologie du geste musical. In G. Guertin (Ed.), *Glenn Gould Pluriel*, pp. 85–111. Québec: Louise Courteau. 40, 41, 46, 47, 53

Delalande, F., M. Formosa, M. Frémiot, P. Gobin, P. Malbosc, J. Mandelbrojt, and E. Pedler (1996). Les Unités Sémiotiques Temporelles: Éléments nouveaux d'analyse musicale. [Audio-CD] Marseille: Éditions MIM – Documents Musurgia. 84

Deutsch, D. (Ed.) (1999). *The Psychology of Music* (2nd ed.). San Diego: Academic Press. 17

Ding, W., G. Marchionini, and D. Soergel (1999). Multimodal surrogates for video browsing. In *DL '99: Proceedings of the fourth ACM conference on Digital libraries*, pp. 85–93. New York: ACM Press. 179

Dobrian, C. (2001). Aesthetic considerations in the use of 'virtual' music instruments. In *Proceedings of the Workshop on Current Research Directions in Computer Music, Institut Universitari de l'Audiovisual, Universitat Pompeu Fabra*, Barcelona, Spain. 22

Dourish, P. (2001). *Where the Action is: The Foundations of Embodied Interaction*. Cambridge, MA: The MIT Press. 11

Duchable, F.-R. (2003). Beethoven Concertos pour piano 1 and 3. A la decouverte des Concertos. François-René Duchable, piano, John Nelson, conductor, Ensemble Orchestral de Paris. [DVD] Harmonia Mundi. 50, 65

Eitan, Z. (1997). *Highpoints. A Study of Melodic Peaks*. Philadelphia, PA: University of Pennsylvania Press. 14

Ekman, P. and W. Friesen (1969). The repertoire of nonverbal behavioral categories. *Semiotica 1*, 49–98. 36, 54

Elliott, R., J. R. W. Glauert, J. R. Kennaway, and I. Marshall (2000). The development of language processing support for the visicast project. In *Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies*, pp. 101–108. New York: ACM Press. 207

Engel, K., M. Flanders, and J. Soechting (1997). Anticipatory and sequential motor control in piano playing. *Experimental Brain Research 113*(2), 189–199. 24

Escobedo, T. (2005). The synthstick. http://www.geocities.com/tpe123/folkurban/ synthstick/synthstick.html (Accessed 1 June 2007). 115

Evrard, M., D. Couroussé, N. Castagné, C. Cadoz, J.-L. Florens, and A. Luciani (2006, September). The GMS file format: Specifications of the version 0.1 of the format. Technical report, INPG, ACROE/ICA, Grenoble, France. 208

Feitis, R. (1978). *Ida Rolf Talks about Rolfing and Physical Reality*. New York: Harper and Row. 79

Feyereisen, P. and J.-D. de Lannoy (1991). *Gestures and Speech: Psychological Investigations*. Cambridge: Cambridge University Press. 38

Findlay, E. (1971). *Rhythm and Movement – Applications of Dalcroze Eurhythmics*. Miamia, FL: Summy-Birchard Inc. 79

Foote, J., M. Cooper, and L. Wilcox (2002). Enhanced video browsing using automatically extracted audio excerpts. In *Proceedings IEEE International Conference on Multimedia and Expo, August 2002*, Lausanne, Switzerland. 179

Fowler, C. (1986). An event approach to the study of speech perception from a direct-realist perspective. *Journal of Phonetics 14*(3), 28. 18

Fowler, C. A. (1996). Listeners do hear sounds, not tongues. *The Journal of the Acoustical Society of America 99*(3), 1730–1741. 18

Frawley, W. J., G. Piatetsky-Shapiro, and C. J. Matheus (1992). Knowledge discovery in databases. *Artificial Intelligence Magazine 13*(3), 57–70. 205

Freed, D. J. (1990). Auditory correlates of perceived mallet hardness for a set of recorded percussive sound events. *The Journal of the Acoustical Society of America 87*(1), 311–322. 26

Freeman, J., K. Varnik, C. Ramakrishnan, M. Neuhaus, P. Burk, and D. Birchfield (2005). Auracle: a voice-controlled, networked sound instrument. *Organised Sound 10*(03), 221–231. 131

Friberg, A., V. Colombo, L. Frydén, and J. Sundberg (2000). Generating Musical Performances with Director Musices. *Computer Music Journal 24*(3), 23–29. 17

Frick, R. (1984). Using both an auditory and a visual short-term store to increase digit span. *Memory Cognition 12*, 507–514. 15

Gabrielsson, A. (1985). Interplay between analysis and synthesis in the studies of music performance and music experience. *Music Perception 3*(1), 59–86. 42

Gabrielsson, A. (1999). Music performance. In D. Deutsch (Ed.), *The Psychology of Music* (2nd ed.)., pp. 501–602. San Diego: Academic Press. 17

Gabrielsson, A. (2003). Music performance research at the millennium. *Psychology of Music 31*(3), 221–272. 17

Gabrielsson, A. and P. N. Juslin (1996). Emotional expression in music performance: Between the performer's intention and the listener's experience. *Psychology of Music 24*, 68–91. 17

Gallese, V., L. Fadiga, L. Fogassi, and G. Rizzolatti (1996). Action recognition in the premotor cortex. *Brain 119*(2), 593–609. 18

Gambetta, C. L. (2005). *Conducting Outside the Box: Creating a Fresh Approach to Conducting Gesture Through the Principles of Laban Movement Analysis*. Ph. D. thesis, The University of North Carolina at Greensboro, Greensboro, NC. 43

Garnett, G. (2001). The Aesthetics of Interactive Computer Music. *Computer Music Journal 25*(1), 21–33. 100

Gaver, W. W. (1986). Auditory icons: Using sound in computer interfaces. *Human-Computer Interaction 2*, 167–177. 30

Gaver, W. W. (1989). The SonicFinder: An interface that uses auditory icons. *Human-Computer Interaction 4*(1), 67–94. 31

Gaver, W. W. (1993a). How do we hear in the world? An ecological approach to auditory event perception. *Ecological Psychology 5*(4), 285–313. 25

Gaver, W. W. (1993b). What in the world do we hear? An ecological approach to auditory event perception. *Ecological Psychology 5*(1), 1–29. 25

Gibson, J. J. (1966). *The senses considered as perceptual systems*. Boston, MA: Houghton Mifflin. 18

Gibson, J. J. (1977). The theory of affordances. In R. Shaw and J. Bransford (Eds.), *Perceiving, acting, and knowing: Toward an ecological psychology*, pp. 67–82. Hillsdale, NJ: Erlbaum. 16, 22

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. New York: Houghton-Mifflin. 4, 12, 18

Giordano, B. L. (2005). *Sound source perception in impact sounds*. Ph. D. thesis, University of Padova, Padova, Italy. 25, 26

Giordano, B. L. and S. McAdams (2006). Material identification of real impact sounds: Effects of size variation in steel, glass, wood, and plexiglass plates. *The Journal of the Acoustical Society of America 119*(2), 1171–1181. 26

Girgensohn, A. (2003). A fast layout algorithm for visual video summaries. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, Washington, DC, pp. 77–80. IEEE Computer Society. 178, 179

Girgensohn, A., J. Boreczky, and L. Wilcox (2001). Keyframe-based user interfaces for digital video. *Computer 34*(9), 61–67. 179

Glass, L. (2001). Synchronization and rhythmic processes in physiology. *Nature* (410), 277–284. 50

Godøy, R. I. (1997). Knowledge in music theory by shapes of musical objects and sound-producing actions. In M. Leman (Ed.), *Music, Gestalt, and Computing: Studies in Cognitive and Systematic Musicology*, pp. 89–102. Berlin Heidelberg: Springer-Verlag. 228

Godøy, R. I. (2001). Imagined action, excitation, and resonance. In R. I. Godøy and H. Jørgensen (Eds.), *Musical Imagery*, pp. 237–250. Lisse: Swets and Zeitlinger. 27, 91, 228

Godøy, R. I. (2003, August). Motor-mimetic music cognition. *Leonardo 36*(4), 317–319. 19

Godøy, R. I. (2004). Gestural imagery in the service of musical imagery. In A. Camurri and G. Volpe (Eds.), *Gesture-Based Communication in Human-Computer Interaction: 5th In-ternational Gesture Workshop, GW 2003, Genova, Italy, April 15-17, 2003, Selected Revised Papers*, Volume LNAI 2915, pp. 55–62. Berlin Heidelberg: Springer-Verlag. 22

Godøy, R. I. (2006). Gestural-sonorous objects: embodied extensions of schaeffer's conceptual apparatus. *Organised Sound 11*(2), 149–157. 4, 22, 24, 27, 28, 228

Godøy, R. I. (forthcoming 2008). Gesture affordances of musical sound. In R. I. Godøy and M. Leman (Eds.), *Sound Gestures*. 23

Godøy, R. I., E. Haga, and A. R. Jensenius (2006a). Exploring music-related gestures by sound-tracing. - a preliminary study. In *2nd ConGAS International Symposium on Gesture Interfaces for Multimedia Systems, May 9-10 2006*, Leeds, UK. 69, 71, 82, 83

Godøy, R. I., E. Haga, and A. R. Jensenius (2006b). Playing "air instruments": Mimicry of sound-producing gestures by novices and experts. In S. Gibet, N. Courty, and J.-F. Kamp (Eds.), *Gesture in Human-Computer Interaction and Simulation: 6th International Gesture Workshop, GW 2005, Berder Island, France, May 18-20, 2005, Revised Selected Papers*, Volume 3881/2006, pp. 256–267. Berlin Heidelberg: Springer-Verlag. 22, 62, 63

Godøy, R. I. and H. Jørgensen (Eds.) (2001). *Musical Imagery*. Lisse: Swets and Zeitlinger. 27

Goldin-Meadow, S. (2003). *Hearing Gesture: How Our Hands Help Us Think*. Cambridge, MA: The Belknap Press of Harvard University Press. 37

Goldman, D. B., B. Curless, D. Salesin, and S. M. Seitz (2006). Schematic storyboarding for video visualization and editing. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pp. 862–871. New York, ACM Press. 180

Good, M. (2001). Musicxml for notation and analysis. In W. B. Hewlett and E. Selfridge-Field (Eds.), *Computing in Musicology 12*, pp. 113–124. Cambridge, MA: The MIT Press. 208

Göschl, A. (1873, 6 April). Caricatures of Liszt. Borsszem Jankó, 6 April 1873. http://www.cph.rcm.ac.uk/Tour/Pages/LIszt.htm (Accessed 9 March 2007). xiv

Gould, G. (1974). The Alchemist. [DVD] EMI Classics (2003). 53

Graham, T. C. N., L. A. Watts, G. Calvary, J. Coutaz, E. Dubois, and L. Nigay (2000). A dimension space for the design of interactive systems within their physical environments. In *DIS '00: Proceedings of the conference on Designing interactive systems*, pp. 406–416. New York, ACM Press. 47

Gritten, A. and E. King (Eds.) (2006). *Music and Gesture*. Hampshire: Ashgate. 2, 41

Guedes, C. (2005). *Mapping Movement to Musical Rhythm: A Study in Interactive Dance*. Ph. D. thesis, New York University. 163

Guest, A. H. (2004). *Labanotation*. New York: Routledge. 79

Haga, E. (fortchoming 2007). *Correspondences between music and human body movement*. Ph. D. thesis, University of Oslo. 77

Hall, E. T. (1984, March). *The Dance of Life: The Other Dimension of Time*. New York: Anchor. 50

Hartmann, B., M. Mancini, and C. Pelachaud (2002). Formational parameters and adaptive prototype instantiation for MPEG-4 compliant gesture synthesis. In *Proceedings of Computer Animation, 2002*, pp. 111–119. 207

Hatten, R. S. (2003). Musical Gesture: Theory and Interpretation. Course notes, Indiana University. http://www.indiana.edu/~deanfac/blfal03/mus/mus_t561_9824.html (Accessed 1 June 2007). 40

Hatten, R. S. (2004). *Interpreting musical gestures, topics, and tropes : Mozart, Beethoven, Schubert*. Bloomington, IN: Indiana University Press. 40, 80

Haueisen, J. and T. R. Knösche (2001). Involuntary motor activity in pianists evoked by music perception. *Journal of Cognitive Neuroscience 13*(6), 786–792. 19

Hendrix, J. (1973). Jimi Hendrix. [DVD] Warner Bros (2005). 54, 56

Hewlett, W. B. and E. Selfridge-Field (Eds.) (1998). *Melodic Similarity. Concepts, Procedures, and Applications*, Volume 11 of *Computing in Musicology*. Cambridge, MA: The MIT Press. 14

Hickok, G., B. Buchsbaum, C. Humphries, and T. Muftuler (2003). Auditory-motor interaction revealed by fmri: Speech, music, and working memory. *Area Spt. Journal of Cognitive Neuroscience 15*(5), 673–682. 19

Hodgins, P. (1992). *Relationships Between Score and Choreography in Twentieth Century Dance: Music, Movement and Metaphor*. Lewiston, NY: Edwin Mellen Press. 80, 81

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences 79*(8), 2554–2558. 14

Hunt, A. (1999). *Radical User Interfaces for Real-time Musical Control*. D. phil. thesis, University of York, UK. 102

Hunt, A. and M. M. Wanderley (2002). Mapping performer parameters to synthesis engines. *Organised Sound 7*(2), 97–108. 224

Hurley, S. (1989). *Natural Reasons*. New York: Oxford University Press. 18

Hurley, S. (1998). *Consciousness in Action*. Cambridge, MA: Harvard University Press. 18

Hurley, S. (2001). Perception And Action: Alternative Views. *Synthese 129*(1), 3–40. 18

Huron, D. (2006, May). *Sweet Anticipation: Music and the Psychology of Expectation (Bradford Books)*. Cambridge, MA: The MIT Press. 14

Il Giardino Armonico (1994). Vivaldi: The Four Seasons. [Audio-CD] Teldec. 78

Jarrett, K. (1984). The Last Solo. [DVD] Image Entertainment. 65

Jazzmutant (2006). Extension and enchancement of the OSC protocol. Draft Presented at the OSC-meeting at NIME 2006, IRCAM, Paris. 226

Jehan, T. (2005). *Creating Music by Listening*. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA. 138

Jensenius, A. R. (2002). How do we recognize a song in one second? the importance of salience and sound in music perception. Cand. philol. thesis, University of Oslo, Oslo, Norway. 4, 14

Jensenius, A. R. (2007, February). GDIF Development at McGill. Short Term Scientific Mission Report, COST 287 Action ConGAS. http://www.cost287.org/documentation/stsms/report/jensenius_report.pdf. 206

Jensenius, A. R., A. Camurri, N. Castagne, E. Maestre, J. Malloch, D. McGilvray, D. Schwarz, and M. Wright (forthcoming 2007). Panel: the need of formats for streaming and storing music-related movement and gesture data. In *Proceedings of the 2007 International Computer Music Conference*, Copenhagen, Denmark. San Francisco: ICMA. 206

Jensenius, A. R., R. I. Godøy, and M. M. Wanderley (2005). Developing tools for studying musical gestures within the Max/MSP/Jitter environment. In *Proceedings of the International Computer Music Conference, 4-10 September, 2005*, Barcelona, Spain, pp. 282–285. San Francisco: ICMA. 138

Jensenius, A. R., R. Koehly, and M. M. Wanderley (2006). Building low-cost music controllers. In R. Kronland-Martinet, T. Voinier, and S. Ystad (Eds.), *CMMR 2005, LNCS 3902*, pp. 123–129. Berlin Heidelberg: Springer-Verlag. 110

Jensenius, A. R., T. Kvifte, and R. I. Godøy (2006). Towards a gesture description interchange format. In *NIME '06: Proceedings of the 2006 International Conference on New Interfaces for Musical Expression*, Paris, France, pp. 176–179. Paris: IRCAM – Centre Pompidou. 206

Johansson, G. (1973). Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics 14*(2), 201–211. 175

Jordà, S., M. Kaltenbrunner, G. Geiger, and R. Bencina (2005). The reacTable*. In *Proceedings of the 2005 International Computer Music Conference*, Barcelona, Spain, pp. 579–582. San Francisco: ICMA. 131

Jungleib, S. (1996). *General MIDI*. Middleton, WI: A-R Editions. 104

Juslin, P. N. (2003). Five facets of musical expression: A psychologist's perspective on music performance. *Psychology of Music 31*(3), 273–302. 17

Juslin, P. N. and K. R. Scherer (2005). Vocal expression of affect. In J. Harrigan, R. Rosenthal, and K. Scherer (Eds.), *The New Handbook of Methods in Nonverbal Behavior Research (Series in Affective Science)*, pp. 65–135. New York: Oxford University Press. 26

Juslin, P. N. and J. Sloboda (2001). Music and emotion: Introduction. In P. Juslin and J. Sloboda (Eds.), *Music and Emotion: Theory and Research*, Series in Affective Science. Oxford: Oxford University Press. 17

Kalbacher, G. (1982). Profile: Brian eno. *Modern Recording Music* (October). 173

Karplus, K. and A. Strong (1983). Digital synthesis of plucked-string and drum timbres. *Computer Music Journal 7*(2), 43–55. 102

Kendon, A. (1972). Some relationships between body motion and speech. In A. Siegman and E. Pope (Eds.), *Studies in dyadic communication*, pp. 177–210. New York: Pergamon Press. 36

Kendon, A. (1980). Gesticulation and speech: Two aspects of the process of utterance. In *The Relationship between Verbal and Nonverbal Communication*, pp. 207–227. The Hague: Mouton. 36

Kendon, A. (1982). The study of gesture: some remarks on its history. *Recherches Sémiotiques/Semiotic Inquiry 2*, 45–62. 36, 37

Kendon, A. (2004). *Gesture: Visible Action as Utterance*. Cambridge: Cambridge University Press. 36, 41, 46, 47

Keysers, C., E. Kohler, M. A. Umiltá, L. Nanetti, L. Fogassi, and V. Gallese (2003). Audiovisual mirror neurons and action recognition. *Experimental Brain Research 153*(4), 628–636. 19

Kipp, M. (2001). Anvil - a generic annotation tool for multimodal dialogue. In *the 7th European Conference on Speech Communication and Technology (Eurospeech)*, Ålborg, Denmark, pp. 1367–1370. 136

Kipp, M. (2003). *Gesture Generation by Imitation: From Human Behavior to Computer Character Animation*. Ph. D. thesis, Saarland University, Saarland, Germany. 136

Koehly, R., D. Curtil, and M. M. Wanderley (2006). Paper FSRs and latex/fabric traction sensors: methods for the development of home-made touch sensors. In *NIME '06: Proceedings of the 2006 International Conference on New Interfaces for Musical Expression, June 4–8*, Paris, France, pp. 230–233. Paris: IRCAM – Centre Pompidou. 114, 115

Kohler, E., C. Keysers, M. A. Umilta, L. Fogassi, V. Gallese, and G. Rizzolatti (2002). Hearing sounds, understanding actions: Action representation in mirror neurons. *Science 297*(5582), 846–848. 19

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE 78*(9), 1464–1480. 15

Kohonen, T. (2001). *Self-Organizing Maps* (3. ed.). Berlin Heidelberg: Springer-Verlag. 15

Kolesnik, P. and M. M. Wanderley (2004). Recognition, analysis and performance with expressive conducting gestures. In *Proceedings of the 2004 International Computer Music Conference*, Miami, FL. San Francisco: ICMA. 43

Kraftwerk (2005). Minimum–Maximum. [DVD] Astralwerks. 197, 198

Kranstedt, A., S. Kopp, and I. Wachsmuth (2002). MURML: A multimodal utterance representation markup language for conversational agents. In *Proceedings of the AAMAS Workshop on 'Embodied conversational agents – Let's specify and evaluate them'*. 207

Krumhansl, C., P. Toivanen, T. Eerola, P. Toiviainen, T. Jarvinen, and J. Louhivuori (2000). Cross-cultural music cognition: cognitive methodology applied to North Sami yoiks. *Cognition 76*(1), 13–58. 15

Kshirsagar, S., N. Magnenat-Thalmann, A. Guye-Vuillome, D. Thalmann, K. Kamyab, and E. Mamdani (2002). Avatar markup language. In *EGVE '02: Proceedings of the workshop on Virtual environments 2002*, Aire-la-Ville, Switzerland, Switzerland, pp. 169–177. Eurographics Association. 207

Kunkler-Peck, A. J. and M. T. Turvey (2000). Hearing shape. *Journal of Experimental Psychology: Human Perception and Performance 26*(1), 279–294. 26

Kurtenbach, G. and E. A. Hulteen (1990). The art of human-computer interface design. In B. Laurel (Ed.), *Gestures in Human-Computer Communication*, pp. 309–317. Reading, PA: Addison Wesley. 38

Kvifte, T. and A. R. Jensenius (2006). Towards a coherent terminology and model of instrument description and design. In *NIME '06: Proceedings of the 2006 International Conference on New Interfaces for Musical Expression*, Paris, France, pp. 220–225. Paris: IRCAM – Centre Pompidou. 102, 103

Laban, R. v. (1963). *Modern Educational Dance*. London: Macdonald & Evans Ltd. 44

Laban, R. v. (1980). *Mastery of Movement* (Fourth ed.). Plymouth: MacDonald & Evans Ltd. 79

Laban, R. v. and F. Lawrence (1947). *Effort*. London: Macdonald & Evans Ltd. 79

Lahav, A., E. Saltzman, and G. Schlaug (2007). Action representation of sound: Audiomotor recognition network while listening to newly acquired actions. *Journal of Neuroscience 27*(2), 308–314. 19, 20

Lakatos, S., S. McAdams, and R. Caussé (1997). The representation of auditory source characteristics: Simple geometric form. *Perception & Psychophysics 59*(8), 1180–1190. 26

Lander, J. (1998). Working with motion capture file formats. *Game Developer* (January), 30–37. 207

Lass, N. J., S. K. Eastman, W. C. Parrish, K. A. Scherbick, and D. Ralph (1982). Listeners' identification of environmental sounds. *Perceptual and Motor Skills 55*, 75–78. 25

Laukka, P. (2004). Instrumental music teachers' views on expressivity: a report from music conservatoires. *Music Education Research 6*(1), 45–56. 17

Leman, M. (1995). *Music and Schema Theory: Cognitive Foundations of Systematic Musicology*. Berlin Heidelberg: Springer-Verlag. 15

Leman, M. (forthcoming 2007). *Embodied Music Cognition and Mediation Technology*. Cambridge, MA: The MIT Press. 3, 42

Leppert, R. (1993). *The Sight of Sound: Music, Representation, and the History of the Body*. Berkeley, CA: University of California Press. 1

Lerdahl, F. and R. Jackendoff (1983). *A Generative Theory of Tonal Music*. Cambridge, MA: The MIT Press. 14

Levenson, T. (1994). *Measure for Measure: A Musical History of Science*. New York: Simon and Schuster. 95

Lewis, M. and J. M. Haviland-Jones (Eds.) (2000). *Handbook of Emotions* (2nd ed.). New York: Guildford Press. 17

Li, X., R. Logan, and R. Pastore (1991). Perception of acoustic source characteristics: Walking sounds. *Journal of the Acoustical Society of America 90*(6), 3036–3049. 26

Liberman, A. M. and I. G. Mattingly (1985). The motor theory of speech perception revised. *Cognition 21*, 1–36. 18, 42

Ligeti, G. (1998). György Ligeti Edition 7: Chamber Music: Trio for Violin, Horn & Piano; Ten Pieces; Six Bagatelles for Wind Quintet; Sonata for Solo Viola: Saschko Gawriloff, Marie-Luise Neunecker, Pierre-Laurent Aimard, London Winds, Tabea Zimmermann. [Audio-CD]: Sony Classical. 78

Lindblom, B. (1991). The status of phonetic gestures. In I. M. Mattingly and M. Studdert-Kennedy (Eds.), *Modularity and the Motor Theory of Speech Perception. Proceedings of a Conference to Honor Alvin M. Liberman*, pp. 7–24. Florence, KY: Lawrence Erlbaum Associates. 18

Lossius, T. (2007). *Sound - Space - Body : Reflections on Artistic Practice*. Research fellowship in the arts thesis, Bergen National Academy of the Arts, Bergen, Norway. 105, 230

Loy, G. (1985). Musicians make a standard: The MIDI phenomenon. *Computer Music Journal 9*(4), 8–26. 105

Luciani, A., M. Evrard, N. Castagné, D. Couroussé, J.-L. Florens, and C. Cadoz (2006). A basic gesture and motion format for virtual reality multisensory applications. In *Proceedings of the 1st international Conference on Computer Graphics Theory and Applications, Setubal, Portugal, March 2006*, Setubal, Portugal. 208

Luciani, A., M. Evrard, D. Courousse, N. Castagne, I. Summers, A. Brady, P. Villella, F. Salsedo, O. Portillo, C. A. Avizzano, M. Raspolli, M. Bergamasco, G. Volpe, B. Mazzarino, M. M. Wanderley, A. R. Jensenius, R. I. Godøy, B. Bardy, T. Stoffregen, G. De Poli, A. Degotzen, F. Avanzini, A. Roda, L. Mion, D'Inca, C. Trestino, and D. Pirro (2006). Report on Gesture Format. State of the Art. Partners' propositions. Deliverable 1 D.RD3.3.1, IST-2004-002114-ENACTIVE Network of Excellence. 208

Luciani, A., J.-L. Florens, and N. Castagne (2005). From action to sound: a challenging perspective for haptics. In *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Pisa, Italy, pp. 592–595. 22

Lutfi, R. A. (2001). Auditory detection of hollowness. *The Journal of the Acoustical Society of America 110*(2), 1010–1019. 26

Machover, T. (2004, October). Shaping minds musically. *BT Technology Journal 22*(4), 171–179. 104, 131

MacRitchie, J., N. J. Bailey, and G. Hair (2006). Multi-modal aquisition of performance parameters for analysis of chopin's b flat minor piano sonata finale op.35. In *DMRN+1: Digital Music Research Network One-day Workshop 2006, Queen Mary, University of London, 20 December 2006*, London, UK. 208

Maeda, J. (2006). *The Laws of Simplicity: Design, Technology, Business, Life*. Cambridge, MA: The MIT Press. 96

Maestre, E., J. Janer, M. Blaauw, A. Pérez, and E. Guaus (forthcoming 2007). Acquisition of violin instrumental gestures using a commercial EMF tracking device. In *Proceedings of the 2007 International Computer Music Conference*, Copenhagen, Denmark. San Francisco: ICMA. 206, 213

Malloch, J., S. Sinclair, and M. M. Wanderley (fortchoming 2007). From controller to sound: Tools for collaborative development of digital musical instruments. In *Proceedings of the 2007 International Computer Music Conference*, Copenhagen, Denmark. San Francisco: ICMA. 223

Manjunath, B., P. Salembier, and T. Sikora (2002). *Introduction to MPEG-7: Multimedia Content Description Interface*. New York: John Wiley and Sons. 207

Marey, E.-J. (1884). Analyse cinématique de la marche. cras, t. xcviii, séance du 19 mai 1884. http://www.bium.univ-paris5.fr/histmed/medica/cote?extcdf003 (Accessed 1 June 2007). 176

Marey, E.-J. (circa 1882). Vol du pélican. http://en.wikipedia.org/wiki/Image:Marey_-_birds.jpg (Accessed 1 June 2007). 176

Marshall, M. T. (2005). Building a usb sensor interface using the atmel atmega16 and the human interface device standard. Technical report, McGill University. http://www.music.mcgill.ca/∼marshall/projects/avr-hid. 117

Marshall, M. T., N. Peters, A. R. Jensenius, J. Boissinot, M. M. Wanderley, and J. Braasch (2006). On the development of a system for gesture control of spatialization. In *Proceedings of the International Computer Music Conference*, New Orleans, LA, pp. 360–366. San Francisco: ICMA. 167, 206, 217, 218

Marshall, M. T. and M. M. Wanderley (2006). Evaluation of sensors as input devices for computer music applications. In R. Kronland-Martinet, T. Voinier, and S. Ystad (Eds.), *CMMR 2005, LNCS 3902*, pp. 130–139. Springer-Verlag Berlin Heidelberg. 100

Martinez, J., R. Koenen, and F. Pereira (2002). MPEG-7: the generic multimedia content description standard, part. *Multimedia, IEEE 9*(2), 78–87. 207

McAdams, S. (2000). The psychomechanics of real and simulated sound sources. *The Journal of the Acoustical Society of America 107*, 2792. 25

McFerrin, B. and N. Kennedy (2005). Spirits of Music, Part 1. [DVD] Euroarts. 54

McGurk, H. and J. MacDonald (1976). Hearing lips and seeing voices. *Nature* (264), 746–748. 16

McNeill, D. (1992). *Hand and Mind: What Gestures Reveal About Thought*. Chicago, IL: University of Chicago Press. 36, 37, 47

McNeill, D. (Ed.) (2000). *Language and Gesture*. Cambridge: Cambridge University Press. 36

McNeill, D. (2005). *Gesture and Thought*. Chicago, IL: University of Chicago Press. 36, 37, 38

Merriam, A. P. (1964). *The Anthropology of Music*. Evanston, IL: Northwestern University Press. 12

Métois, E. (1997). *Musical Sound Information: Musical Gestures and Embedding Synthesis*. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA. 40

Meyer, L. B. (1956). *Emotion and Meaning in Music*. Chicago, IL: University of Chicago Press. 17

Middleton, R. (1993, May). Popular music analysis and musicology: Bridging the gap. *Popular Music 12*(2), 177–190. 40

Miranda, E. R. and M. M. Wanderley (2006). *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*, Volume 21 of *The Computer Music and Digital Audio Series*. Middleton, WI: A-R Editions, Inc. 39, 100

Mitchell, O., C. Ross, and G. Yates (1969). Signal processing for a cocktail party effect. *The Journal of the Acoustical Society of America 45*, 315. 15

Momeni, A. and D. Wessel (2003). Characterizing and controlling musical material intuitively with graphical models. In *NIME '03: Proceedings of the 2003 International Conference on New Interfaces for Musical Expression*, Montreal, Canada, pp. 54–62. Montreal: McGill University. 101

Moore, F. R. (1988). The dysfunctions of MIDI. *Computer Music Journal 12*(1), 19–28. 105

Morris, D., P. Collett, and P. Marsh (1979). *Gestures: Their Origins and Distribution*. London: Jonathan Cape. 2

Morrison, J. (1985). EA IFF 85: Standard for interchange format files. Technical report, Electronic Arts. 208

Mozley, A. V. (Ed.) (1972). *Eadweard Muybridge: The Stanford Years, 1872-1882*. Palo Alto, CA: Stanford University Museum of Art. 174

Muybridge, E. J. (1887). *Animal Locomotion. An Electro-Photographic Investigation of Consecutive Phases of Animal Movements, 1872-1885. 11 vol* (Philadelphia, PA: University of Pennsylvania. Reprint: Muybridge's Complete Human and Animal Locomotion, 3 vol. ed.). New York: Dover Publications, 1979. 175

Nakra, T. M. (2000, February). *Inside the Conductor's Jacket: Analysis, Interpretation and Musical Synthesis of Expressive Gesture*. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA. 43

Narmour, E. (1990). *The Analysis and Cognition of Basic Melodic Structures. The Implication-Realization Model*. Chicago, IL: University of Chicago Press. 14

Neuhoff, J. G. (2001). Perceveiving acoustic source orientation in three-dimensional space. In *Proceedings of the 2001 International Conference on Auditory Display, July 29-August 1, 2001*, Espoo, Finland, pp. 111–116. 26

Newlove, J. and J. Dalby (2004). *Laban for All*. New York: Routledge. 79

Ng, K. C. (2004). Music via motion: transdomain mapping of motion and sound for interactive performances. *Proceedings of the IEEE 92*(4), 645–655. 165

Niikura, Y., Y. Taniguchi, A. Akutsu, and Y. Tonomura (1999). Valbum: Album-oriented video storyboard for editing and viewing video. In S. Nishio and F. Kishino (Eds.), *AMCP'98*, Volume LNCS 1554, pp. 17–29. Berlin Heidelberg: Springer-Verlag. 180

Ohala, J. J. (1996). Speech perception is hearing sounds, not tongues. *The Journal of the Acoustical Society of America 99*(3), 1718–1725. 18

Overholt, D. (2006). Musical interaction design with the create usb interface teaching hci with cuis instead of guis. In *Proceedings of the 2006 International Computer Music Conference*, New Orleans, LA, pp. 425–430. San Francisco: ICMA. 117

Palmer, C. (1997). Music performance. *Annual Review of Psychology 48*, 115–38. 58

Paradiso, J. and E. Hu (1997). Expressive footwear for computer-augmented dance performance. In *Proceedings of the First International Symposium on Wearable Computers, Cambridge, MA*, pp. 165–166. IEEE Computer Society Press. 115

Parsons, D. (1992). The Parsons Dance Company. [DVD] Image Entertainment. 84

Pierce, A. and R. Pierce (1989). *Expressive Movement: Posture and Action in Daily Life, Sports, and the Performing Arts*. Cambridge, MA: Perseus Publishing. 42

Piringer, J. (2001). Elektronische musik und interaktivität: Prinzipien, konzepte, anwendungen. Master's thesis, Technical University of Vienna, Vienna, Austria. 97

Place, T. and T. Lossius (2006). Jamoma: A modular standard for structuring patches in max. In *Proceedings of the 2006 International Computer Music Conference*, New Orleans, LA, pp. 143–146. San Francisco: ICMA. 141

Pogorelich, I. (1999). Chopin: 4 Scherzi. [Audio-CD] Deutsche Grammophon 439 947-2. 65

Pollick, A. S. and F. B. M. de Waal (2007). Ape gestures and language evolution. *Proceedings of the National Academy of Sciences of the United States of America 104*(19), 8184–8189. 2

Prendinger, H., S. Descamps, and M. Ishizuka (2004). MPML: A markup language for controlling the behavior of life-like characters. *Journal of Visual Languages and Computing 15*(2), 183–203. 207

Prodger, P. (2003). *Time Stands Still – Muybridge and the Instantaneous Photography Movement*. Oxford: Oxford University Press. 174

Puckette, M. (1985). A real-time music performance system. Technical report, The MIT Department of Electrical Engineering, Cambridge, MA. 137

Puckette, M. (1988). The patcher. In C. Lischka and J. Fritsch (Eds.), *The International Computer Music Conference*, Cologne, Germany, pp. 420–429. San Francisco: ICMA. 137

Puckette, M. (1996). Pure data: another integrated computer music environment. In *Proceedings of the Second Intercollege Computer Music Concerts*, Tachikawa, Japan, pp. 37–41. 137

Purves, D., G. Augustine, D. Fitzpatrick, L. Katz, A. LaMantia, and J. McNamara (2001). *Neuroscience* (Second ed.). Sunderland, MA: Sinauer Associates. 19

Quek, F., D. McNeill, R. Bryll, S. Duncan, X.-F. Ma, C. Kirbas, K. E. McCullough, and R. Ansari (2002). Multimodal human discourse: gesture and speech. *ACM Transactions on Computer-Human Interaction (TOCHI) 9*(3), 171–193. 39

Quek, O., V. Verfaille, and M. M. Wanderley (2006). Sonification of musician's ancillary gesutres. In *Proceedings of the Proceedings of the 12th International Conference on Auditory Display*, London, UK. 50

Ramstein, C. (1991). *Analyse, représentation et traitement du geste instrumental.* Thèse de docteur ingénieur spécialité informatique, Institut National Polytechnique de Grenoble, France. 43

Repp, B. H. (1987). The sound of two hands clapping: An exploratory study. *The Journal of the Acoustical Society of America 81*(4), 1100–1109. 26

Rizzolatti, G. and M. A. Arbib (1998). Language within our grasp. *Trends in Neuroscience 21*, 188–194. 19

Rocchesso, D. and F. Fontana (2003). *The Sounding Object*. Florence: Edizioni di Mondo Estremo. 26

Rocchesso, D. and L. Ottaviani (2001). Can one hear the volume of a shape? In *2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, pp. 115–118. 26

Roland, P. (2002). The Music Encoding Initiative (MEI). In *Proceedings of the First International Conference on Musical Applications Using XML*, pp. 55–59. 208

Rosch, E., C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem (1976). Basic Objects in Natural Categories. *Cognitive Psychology 8*(3), 382–439. 16

Rosenbaum, D. A. (1991). *Human Motor Control*. San Diego: Academic Press. 83

Rowe, R. (2001). *Machine Musicianship*. Cambridge, MA: The MIT Press. 104

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP research group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1.: Foundations*, pp. 318–362. Cambridge, MA: The MIT Press. 14

Sacks, H. and E. Schegloff (2002). Home position. *Gesture* (2), 133–146. 44

Sagvolden, T. (2006). The alpha-2 A adrenoceptor agonist guanfacine improves sustained attention and reduces overactivity and impulsiveness in an animal model of attention-deficit/hyperactivity disorder (ADHD). *Behavioral and Brain Functions 2*(1), 41. 200

Sagvolden, T., E. B. Johansen, H. Aase, and V. A. Russell (2005). A dynamic developmental theory of attention-deficit/hyperactivity disorder (ADHD) predominantly hyperactive/impulsive and combined subtypes. *Behavioral and Brain Sciences 28*(03), 397–419. 200

Scavone, G. and P. R. Cook (2005). Rtmidi, Rtaudio, and a Synthesis Toolkit (STK) update. In *Proceedings of the 2005 International Computer Music Conference*, Barcelona, Spain, pp. 327–330. San Francisco: ICMA. 120

Schaeffer, P. (1966). *Traité des objets musicaux*. Paris: Editions du Seuil. 4, 11, 24

Schaeffer, P. (1967/1998). Solfege de l'objet sonore. [Audio-CD] Paris: INA/GRM. 84

Schneider, A. and R. I. Godøy (2001). Perspectives and challenges of musical imagery. In R. I. Godøy and H. Jørgensen (Eds.), *Musical Imagery*, pp. 5–26. Lisse: Swets and Zeitlinger. 4

Schnell, N., R. Borghesi, D. Schwarz, F. Bevilacqua, and R. Müller (2005). FTM – complex data structures for Max. In *Proceedings of the 2005 International Computer Music Conference*, Barcelona, Spain, pp. 9–12. San Francisco: ICMA. 213

Schomaker, L., J. Nijtmans, A. Camurri, F. Lavagetto, P. Morasso, C. Benoît, T. Guiard-Marigny, B. L. Goff, J. Robert-Ribes, A. Adjoudani, I. Defée, S. M"unch, K. Hartung, and J. Blauert (1995). A taxonomy of multimodal interaction in the human information processing system. Technical report, ESPRIT Project 8579. http://hwr.nici.kun.nl/ miami/taxonomy/taxonomy.html (Accessed 1 June 2007). 13

Schrader, C. A. (2004). *A Sense of Dance: Exploring Your Movement Potential*. Champaign, IL: Human Kinetics. 80

Schwarz, D. and M. Wright (2000). Extensions and applications of the SDIF sound description interchange format. In *Proceedings of the 2000 International Computer Music Conference*, Berlin, Germany, pp. 481–484. San Francisco: ICMA. 208

Serafin, S., A. de Götzen, N. Böttcher, and S. Gelineck (2006). Synthesis and control of everyday sounds reconstructing russolo's intonarumori. In *NIME '06: Proceedings of the 2006 International Conference on New Interfaces for Musical Expression*, Paris, France, pp. 240–245. IRCAM – Centre Pompidou. 102

Shaffer, L. H. (1980). Analyzing piano performance: a study of concert pianists. In G. E. Stelmach and P. A. Vroon (Eds.), *Tutorial in Motor Behaviour*, pp. 443–456. Amsterdam: North-Holland. 42

Sinclair, S. and M. M. Wanderley (2007). Defining a control standard for easily integrating haptic virtual environments with existing audio/visual systems. In *NIME '07: Proceedings of the 2007 Conference on New Interfaces for Musical Expression*, New York. 226

Small, C. (1998). *Musicking. The Meanings of Performing and Listening*. Hanover, New Hampshire: Wesleyan University Press. 12, 43

Smith, J. (1992). Physical modeling using digital waveguides. *Computer Music Journal 16*(4), 74–91. 102

Smith, R. (1980, December). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers C-29*(12), 1104–1113. 14

Solso, R. L., M. K. MacLin, and O. H. Mclin (2005). *Cognitive Psychology* (7th ed.). Boston, MA: Allyn and Bacon. 19

Sting (1985). Bring on the Night. [DVD] A & M (2005). 51, 53, 196, 197

Strogatz, S. and I. Stewart (1993). Coupled Oscillators and Biological Synchronization. *Scientific American Magazine 269*(6), 102–109. 50

Swanson, J. M., J. A. Sergeant, E. Taylor, E. J. Sonuga-Barke, P. S. Jensen, and D. P. Cantwell (1998). Attention-deficit hyperactivity disorder and hyperkinetic disorder. *Lancet 351*(9100), 429–33. 199

Sweller, J. (2002). Visualisation and instructional design. In *Proceedings of the International Workshop on Dynamic Visualizations and Learning*, pp. 1501–1510. 15

Tanzer, I. O. (2006). *Numerical Modeling in Electro- and Magnetoencephalography*. Ph. D. thesis, Helsinku University of Technology, Helsinki, Finland. 19

Taylor, E., J. Sergeant, M. Doepfner, B. Gunning, S. Overmeyer, H. J. Möbius, and H. G. Eisert (1998). Clinical guidelines for hyperkinetic disorder. *European Child & Adolescent Psychiatry 7*(4), 184–200. 199

Teodosio, L. and W. Bender (1993). Salient video stills: content and context preserved. In *MULTIMEDIA '93: Proceedings of the first ACM international conference on Multimedia*, pp. 39–46. New York: ACM Press. 181

The Manhattan Project (1991). The Manhattan Project: Featuring Wayne Shorter, Michel Petrucciani, Stanley Clarke, Lenny White. [DVD] Blue Note Records (2005). 51, 52

The MIDI Association (1983). MIDI - Musical Instrument Digital Interface. 104

Thelen, E. (1995). Time-scale dynamics in the development of an embodied cognition. In R. Port and T. van Gelder. (Eds.), *In Mind In Motion*. Cambridge, MA: The MIT Press. 11

Thomas, N. J. (2007). Mental imagery. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, Stanford University. http://plato.stanford.edu/archives/win2003/entries/davidson/ (Accessed 1 June 2007). 27

Thompson, W. and F. Russo (2006). Preattentive integration of visual and auditory dimensions of music. In *Second International Conference on Music and Gesture*, Royal Northern College of Music, Manchester, UK. 16

Thompson, W. F., P. Graham, and F. A. Russo (2005). Seeing music performance: Visual influences on perception and experience. *Semiotica 156*(1–4), 203–227. 16

Todd, N. P. M. (1992). The dynamics of dynamics: A model of musical expression. *The Journal of the Acoustical Society of America 91*(6), 3540–3550. 17

Todd, N. P. M. (1995). The kinematics of musical expression. *The Journal of the Acoustical Society of America 97*(3), 1940–1949. 40

Todd, P. M. and E. R. Miranda (2004). Putting some (artificial) life into models of musical creativity. In I. Deliege and G. Wiggins (Eds.), *Musical creativity: Current research in theory and practice*. London, UK: Psychology Press. 17

Trevarthen, C. (1999-2000). Musicality and the intrinsic motive pulse: Evidence from human psychobiology and infant communication. *Musicae Scientiae Special Issue: Rhythm, Musical Narrative, and Origin of Human Communication*, 155–211. 13

Trueman, D., R. L. DuBois, and C. Bahn (2001). Discovering expressive realtime parameters: the performing laptop artist as effect designer. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01)*, Limerick, Ireland. 32

Uchihashi, S., J. Foote, A. Girgensohn, and J. Boreczky (1999). Video manga: generating semantically meaningful video summaries. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, Orlando, FL, pp. 383–392. New York, ACM Press. 178

van der Meer, W. (2004). Praat tutorial for musicologists. http://www.musicology.nl/WM/research/praat_musicologists.htm (Accessed 1 June 2007). 75

van der Veer, N. J. (1979). *Ecological Acoustics: Human Perception of Environmental Sounds*. Ph. D. thesis, Cornell University, Ithaca, NY. 25

van Nort, D. and M. M. Wanderley (2006). Exploring the effect of mapping trajectories on musical performance. In *Proceedings of Sound and Music Computing*, Marseille, France. 101

Varese, E. (1993). L'oeuvre De Edgar Varese Vol 1. [Audio-CD] Erato Disques. 84

Vines, B., C. Krumhansl, M. Wanderley, and D. Levitin (2005). Cross-modal interactions in the perception of musical performance. *Cognition 101*, 80–113. 16

Wanderley, M. M. (1999). Non-obvious performer gestures in instrumental music. In A. Braffort, R. Gherbi, S. Gibet, J. Richardson, and D. Teil (Eds.), *Gesture-Based Communication in Human-Computer Interaction: International Gesture Workshop, GW'99, Gif-sur-Yvette, France, March 1999. Proceedings*, pp. 37–48. Berlin Heidelberg: Springer-Verlag. 47, 49

Wanderley, M. M. (2001). *Performer-Instrument Interaction: Applications to Gestural Control of Sound Synthesis*. Phd-thesis, Université Pierre et Marie Curie, Paris VI, Paris, France. 42, 101

Wanderley, M. M. (2002). Quantitative analysis of non-obvious performer gestures. In I. Wachsmuth and T. Sowa (Eds.), *Gesture and Sign Language in Human-Computer Interaction: International Gesture Workshop, GW 2001, London, UK, April 18-20, 2001. Revised Papers*, Volume LNAI 2298, pp. 241–253. Berlin Heidelberg: Springer-Verlag. 49

Wanderley, M. M. and P. Depalle (2004, April). Gestural control of sound synthesis. *Proceedings of the IEEE 92*(4), 632–644. 46, 47

Wanderley, M. M., N. Schnell, and J. B. Rovan (1998). Escher-modeling and performing composed instruments in real-time. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, Volume 2, San Diego, CA, pp. 1080–1084. 224

Wanderley, M. M., B. W. Vines, N. Middleton, C. McKay, and W. Hatch (2005). The musical significance of clarinetists' ancillary gestures: An exploration of the field. *Journal of New Music Research 34*(1), 97–113. 51, 194

Warren, W. H. and R. R. Verbrugge (1984). Auditory perception of breaking and bouncing events: A case study in ecological acoustics. *Journal of Experimental Psychology: Human Perception and Performance 10*, 704–712. 26

Wechlser, R., F. Weiss, and P. Dowling (2004). EyeCon – a motion sensing tool for creating interactive dance, music, and video projections. In *Proceedings of the Society for the Study of Artificial Intelligence and the Simulation of Behavior (SSAISB)'s convention: Motion, Emotion and Cognition*, University of Leeds, UK. 165

Wessel, D. L., C. Drame, and M. Wright (1998). Removing the time axis from spectral model analysis-based additive synthesis: Neural networks versus memory-based machine learning. In *Proceedings of the 1998 International Computer Music Conference*, Ann Arbor, MI. San Francisco: ICMA. 14

Wessel, D. L. and M. Wright (2001). Problems and prospects for intimate musical control of computers. In *NIME '01: Proceedings of the 2001 International Conference on New Interfaces for Musical Expression*, Seattle, WA. New York: ACM Press. 103

Widmer, G. (2002). Machine Discoveries: A Few Simple, Robust Local Expression Principles. *Journal of New Music Research 31*(1), 37–50. 17

Wilson, M. and G. Knoblich (2005). The case for motor involvement in perceiving conspecifics. *Psychological Bulletin 1*(3), 460–473. 18

Winkler, T. (1995). Making motion musical: Gesture mapping strategies for interactive computer music. In *Proceedings of the 1995 International Computer Music Conference*, Banff, Canada. San Francisco: ICMA. 97

Wohlschläger, A., M. Gattis, and H. Bekkering (2003). Action generation and action perception in imitation: an instance of the ideomotor principle. *Philosophical Transactions of the Royal Society B: Biological Sciences 358*(1431), 501–515. 62

Wright, M., A. Chaudhary, A. Freed, D. Wessel, X. Rodet, D. Virolle, R. Woehrmann, and X. Serra (1998). New applications of the sound description interchange format. In *Proceedings of the 1998 International Computer Music Conference*, Ann Arbor, MI, pp. 276–279. San Francisco: ICMA. 208

Wright, M. and A. Freed (1997). Open Sound Control: A new protocol for communicating with sound synthesizers. In *Proceedings of the 1997 International Computer Music Conference*, Thessaloniki, Greece, pp. 101–104. San Francisco: ICMA. 106

Wright, M., A. Freed, A. Lee, T. Madden, and A. Momeni (2001). Managing complexity with explicit mapping of gestures to sound control with OSC. In *Proceedings of the 2001 International Computer Music Conference*, La Habana, Cuba, pp. 314–317. San Francisco: ICMA. 106, 146, 216

Zbyszynski, M. and A. Freed (2005). Control of VST plug-ins using OSC. In *Proceedings of the 2005 International Computer Music Conference*, Barcelona, Spain, pp. 263–266. San Francisco: ICMA. 146

Zhao, L. (2001). *Synthesis and Acquisition of Laban Movement Analysis Qualitative Parameters for Communicative Gestures*. Ph. D. thesis, CIS, University of Pennsylvania, Philadelphia, PA. 36

Zicarelli, D. (1991). Communicationg with meaningless numbers. *Computer Music Journal 15*(4), 74–77. 100

Zicarelli, D. (1998). An extensible real-time signal processing environment for max. In *Proceedings of the 1998 International Computer Music Conference*, Beijing, China, pp. 463–466. San Francisco: ICMA. 137

APPENDIX A Musical Gestures Toolbox

The Musical Gestures Toolbox is implemented as a number of components and modules in the Jamoma distribution. The following lists give an overview of my contributions to the Jamoma project, and all my commits to the project can be found in Jamoma's repository[1] (my username is *alexarje*).

Components

jcom.absdiff calculates the absolute difference between two numbers.

jcom.autocrop% automatically crops an incoming video based on the colour difference in the frame. This component works well with motion images.

jcom.autoscale automatically scales incoming values to a desired output range. Different modes allow for scaling based on a running window or general maximum and mininimum values.

jcom.autosize% automatically resizes a **jit.pwindow** based on the size of the incoming matrix. This component is practical to use together with **jcom.autocrop%** or other objects/modules where the size of the matrix is changing dynamically.

jcom.binary% converts the incoming video to a binary image. It has positive and negative modes.

jcom.char2float% converts a matrix from Jitter type *char* to *float32*, keeping the planecount and dimensions of the matrix.

[1] http://sourceforge.net/projects/jamoma/

jcom.checkplanes% adjusts subsequent modules if the plane number of a matrix changes. This component might help save some CPU.

jcom.float2char% converts a matrix from Jitter type *float32* to *char*, keeping planecount and dimensions of the matrix.

jcom.fullscreen makes a patch fill the screen.

jcom.fullscreen% makes a **jit.window** fill the screen.

jcom.luma2rgb% converts a matrix from grayscale to RGB, but only if necessary. This component might save some CPU over **jit.luma2rgb**.

jcom.mean% calculates the mean of the rows or columns in a matrix.

jcom.motion% finds the motion image based on simple frame-differencing.

jcom.passmatrixinfo% passes on information about the size and planes of a matrix.

jcom.op similar to jit.op% but for floats.

jcom.pi returns the value of pi.

jcom.random returns a random float value. It is more flexible than **random** since it allows for setting the range of the random values.

jcom.rgb2luma% converts a matrix from RGB to grayscale, but only if necessary. This component might save some CPU over **jit.rgb2luma**.

jcom.spray% distributes a video matrix to multiple (8) windows on **bang**.

Modules

jmod.analyzer∼ outputs various types of sound analysis. Based on **analyzer∼** by Tristan Jehan.

jmod.background% does a simple, and a little more advanced, background subtraction.

jmod.blur% spatial blurring of the video.

jmod.brcosa% changes brightness, contrast and saturation of the video.

jmod.crop allows for cropping a video file by clicking and dragging in the video image.

jmod.file_browser makes it possible to browse a list of files and pass on the file name to the audio and video players.

jmod.fluidsynth∼ is a synthesiser module using soundfonts.

jmod.hi outputs data from human interface devices (joysticks, etc.).

jmod.input% is the general video input module. Allows for using a camera, video file or a simple video synth.

jmod.midiin outputs incoming midi data.

jmod.motion% is the general motion module with lots of features for both quantitative and qualitative analysis.

jmod.motiongram% creates motiongrams and videograms, both horizontal and vertical.

jmod.mouse outputs data from the computer mouse.

jmod.mouse.gdif outputs data from the computer mouse, formatted using GDIF.

jmod.multidelay∼ is a multiple delay line audio effect.

jmod.noise∼ plays various types of video noise.

jmod.op% is a wrapper around **jit.op**.

jmod.orsize% changes the orientation, size and rotation of the incoming video matrix.

jmod.oscnet is a networking module using Open Sound Control for communication. Developed together with Tim Place.

jmod.output% outputs video to an OpenGL window.

jmod.phidgets.accelerometer gets data from the USB accelerometers from Phidgets Inc.

jmod.phidgets.interfacekit gets data from the 8/8/8 USB sensor interfaces from Phidgets Inc.

jmod.polhemus gets data from a Polhemus electromagnetic tracker connected through the serial port. The module has only been tested with the Polhemus Patriot.

jmod.similarity% creates similarity matrices of the incoming video.

jmod.sine∼ plays various simple waveforms, including sine, square and sawtooth.

jmod.wacom gets data from a Wacom graphical tablet. Based on **wacom** by Jean-Michel Couturier. I made the first version of this module, and it has later been maintained by Pascal Baltazar.