A Framework for Computer-Aided Composition of Space, Gesture, and Sound. Conception, Design, and Applications

Marlon Mario Schumacher

A Framework for Computer-Aided Composition of Space, Gesture, and Sound. Conception, Design, and Applications.

Mario Marlon Schumacher

Music Technology Area
Schulich School of Music
McGill University
Montreal, Canada

Jury:
Dr. Jean Bresson
Prof. Sean A. Ferguson
Prof. Marcelo M. Wanderley
Prof. Joel Chadabe (External Examiner)
Prof. Philippe Leroux (Internal Examiner)
Prof. Pierre Michaud (Oral Examiner)

Defended on July 14, 2016

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Computer-Aided Composition (CAC) research aims to provide composers with specialized programming languages and interfaces to simulate processes of composition in an interactive approach referred to as "compositional modelling". Although the increased performance and availability of computer resources over the past decade has led to a widening of technologies used in musical practice, in particular in the fields of spatial audio, physical computing and sound synthesis, these developments have remained mostly isolated from the domain of CAC and are therefore not accounted for in the corresponding models and applications.

This dissertation proposes a further evolution of CAC systems by integrating spatial-, sound-, and gesture-related data with the objective of making these technologies available and responsive to formal music composition. We present conception, design, and applications of an original software framework implemented as three libraries for the visual computer-aided composition environment OpenMusic.

The library OMPrisma allows for the design of spatialization processes of arbitrary complexity and relate them to high-level musical processes and materials in a compositional framework. Beyond conventional concepts we introduce a novel paradigm which extends the notion of sound source spatialization from pre-existing sound objects to the spatial micro-structure of sound, referred to as *spatial sound synthesis*.

The library OM-Geste proposes solutions for integrating physical gestures as materials into compositional formalisms. Coding, representation and processing of gestures as multi-dimensional signals are discussed and design guidelines are presented. A novel approach is introduced which allows to extend the scope of mapping to high-level programs for creating musical objects of arbitrary scale using hybrid specifications. Two complementary examples for validation with the use of real-world gesture recordings are described.

The library OM-Pursuit introduces a dictionary-based analysis/synthesis system which provides abstract representations of sound targets that can be integrated and manipulated within symbolic compositional processes. This model constitutes a granular counterpart to the prevalent sinusoidal representations and allows for renewed conceptions of sound and compositional directions. A professional application of the software for a composition for computer-controlled piano and electronics is described.

The presented software libraries have been designed as a coherent framework based on the same design concepts, interfaces and programming tools. All of the presented developments are freely available open-source projects which have been validated in research and professional artistic productions, and are now being taught as part of computer music curricula in different universities and research institutions.

Keywords: Computer-Aided Composition, Spatialization, Gesture, Mapping, Sound Representation

Résumé

La recherche en composition assistée par ordinateur (CAO) a pour but d'offrir aux compositeurs des langages de programmation et des interfaces spécialisés permettant de simuler des processus compositionnels dans un cadre interactif et basé sur la notion de modélisation. Bien que la disponibilité et le niveau de performance des ressources informatiques se soient développés depuis les dernières décennies et aient conduit à une prolifération de l'utilisation des technologies dans la pratique musicale, en particulier dans les domaines de l'audio spatialisé, du contrôle gestuel et de la synthèse sonore, ces développements sont demeurés relativement isolés du domaine de la CAO et n'ont donc pas été pris en compte dans les modèles et applications relatifs à celle-ci.

Cette thèse propose un développement des systèmes de composition assistée par ordinateur en y intégrant des outils concernant les données d'ordre spatial, gestuel et sonore, dans l'objectif de rendre ces technologies disponibles et réactives dans le contexte de la composition écrite ou symbolique. Nous présentons la conception, le design et des applications d'une structure logicielle (software framework) intégrée sous la forme de trois bibliothèques pour l'environnement de composition assistée par ordinateur OpenMusic.

La bibliothèque OM-Prisma permet la spécification de processus de spatialisation de complexité arbitraire et les relie à d'autres processus et matériaux musicaux de haut niveau dans un environnement compositionnel. Nous introduisons un nouveau paradigme au-delà des concepts conventionnels, qui dépasse la spatialisation d'objets sonores individuels pour atteindre la microstructure du son et ainsi réaliser une *synthèse du son spatialisée*.

La bibliothèque OM-Geste propose des solutions pour intégrer les gestes physiques comme matériaux dans les formalisations musicales. La description, la représentation et le traitement des gestes en tant que signaux multidimensionnels y sont discutés et nos pistes de solutions y sont présentées. Un système est introduit et permet de concevoir le *mapping* comme un programme pour la création d'objets musicaux d'échelle et de résolutions arbitraires en utilisant des spécifications hybrides. En guise de validation, deux exemples complémentaires, incluant des enregistrements de gestes réels y sont décrits.

La bibliothèque OM-Pursuit introduit un système d'analyse/synthèse fondé sur des dictionnaires qui fournit des représentations abstraites de son-cible qui sont susceptibles d'être intégrées et manipulées par des procédés compositionnels dans le domaine symbolique.

Ce modèle constitue une contrepartie granulaire aux modèles sinusoïdaux dominants et permet le renouvellement des manière de concevoir le son et la composition. Une application professionnelle de ce logiciel dans le cadre de la composition d'une œuvre pour piano contrôlée par ordinateur et dispositif électronique y est décrite.

Les bibliothèques logicielles présentées dans cette thèse ont été conçues comme une structure cohérente à partir des mêmes concepts, interfaces et structures de données. Tous les développements exposés sont en accès libre ; ce sont des projets *open-source* et ils ont été validés au sein de productions artistiques professionnelles, de projets de recherche et font maintenant partie des programmes d'études de musique électronique dans différentes universités et institutions de recherche.

Mots-clés : Composition assistée par ordinateur, spatialisation, geste, mapping, synthèse sonore.

Acknowledgments

First, I would like to extend my gratitude to my supervisor Marcelo Wanderley for sharing his knowledge, scientific spirit and sense of humour. I would also like to thank Sean Ferguson for his encouragement and for supporting my research with project opportunities and academic exchanges. Deepest thanks to Jean Bresson, whose personality, both as researcher and supervisor, was an inspiration to me. Thanks also to Philippe Depalle for teaching me how to think outside the box and ask the right questions. I would also like to extend my thanks to Profs. Heather Goad, France Bouthillier, and Eleanor Stubley.

Thanks to my colleagues at the Input Devices and Music Interaction Laboratory at McGill University; Johnny, Marcello, Carolina, Emma (KTH Stockholm), Deborah, and the fellow researchers and staff at the Centre for Interdisciplinary Research in Music Media and Technology: Cedric, Yves, Harold, Julien. Thanks also to my colleagues from the Music Representations team at IRCAM: Gerard Assayag, Carlos Agon, Gregoire Carpentier, Philippe Esling, Jérémie Garcia, Thibaut Carpentier (ECA). Special thanks to my friend and colleague Graham Boyes.

My gratitude to Marco Stroppa for guiding me into this, and to Thomas Troge for opening a new window of possibilities.

Thanks to the professional composers, performers, and artists I had the opportunity to work with, in particular Christopher Trapani, Aaron Einbond, Núria Giménez-Comas, Elliot Britton, Elinor Frey, Marjolaine Lambert, Sophie Breton, Soula Trougakos. I would also like to thank Isabelle van Grimde, who kindly agreed on the use of gesture recordings of her choreographies for this research. Thanks to Karine Bouchard and Jimmie Leblanc for help with translations.

Thanks to the Fonds de Recherche du Québec – Nature et Technologie (FRQNT), CIRMMT and McGill University for funding this research.

My sincere thanks to Enzo Savarino, whose support made all this possible, and above all to my family and friends for their love and faith in me.

To Pascale.

Contribution of Authors

The document is formatted as a manuscript dissertation and includes the peer-reviewed publications listed below.

Chapter 2 Schumacher, M., Bresson, J. (2010). Spatial Sound Synthesis in Computer-Aided Composition. Organised Sound, 15(03), 271–289. Cambridge University Press (UK).

Chapter 3 Schumacher, M., Wanderley, M. M. (2016). Integrating Gesture Data in Computer-Aided Composition: Paradigms for Representation, Processing and Mapping. Accepted for publication in Journal of New Music Research, Taylor and Francis Group.

Chapter 4 Schumacher, M. (2016). Ab-Tasten: Atomic Sound Modeling with a Computer-controlled Grand Piano. In Bresson, J., Assayag, G. and Agon, C. (Eds.), The OM Composer's Book: Volume 3. Éditions Delatour France / IRCAM – Centre Pompidou. In press.

I was responsible for all software developments presented in this thesis, the artistic applications, and the manuscripts for the above listed publications. Chapter 2 presents conception and design of an object-oriented system for spatial sound synthesis, integrated into the computer-aided composition environment OpenMusic. The described system was conceived and developed by myself with contributions by Jean Bresson for its integration with existing sound synthesis frameworks. Bresson also contributed to parts of the manuscript. The work presented in chapter 3 describes motivation, design and implementation of a software system for integrating gesture signals as musical objects into compositional formalisms. I conceived and developed the system myself with contributions by Jean Bresson for the mapping system. Gesture recordings of a dance performance choreographed by Isabelle van Grimde were carried out under my direction with the help of fellow students. I wrote this manuscript inspired by theoretical discussions with Marcelo Wanderley, who provided feedback and suggestions. Chapter 4 discusses abstract sound representations for musical applications and presents a corpus-based, atomic sound model, integrated into a computer-aided composition environment. Conception and development

of this system were carried out by myself, but in the context of close collaboration with Graham Boyes, who contributed signal-processing functionalities in an external executable.

Contents

List of Figures

List of Acronyms

3DC	3D Curve Object
3DC-lib	3D Curve Library
ASA	Auditory Scene Analysis
BPF	Break Point Function
BPF -lib	Break Point Function Library
BPC	Break Point Curve
BPC-lib	Break Point Curve Library
CAC	Computer-Aided Composition
CBAD	Corpus-based Atomic Decomposition
DBAP	Distance Based Amplitude Panning
DMI	Digital Musical Instrument
DSP	Digital Signal Processing
GDIF	Gesture Description Interchange Format
GMS	Gesture and Motion Signal Format
GUI	Graphical User Interface
HCI	Human Computer Interaction
HOA	Higher-Order Ambisonics
HRTF	Head-Related Transfer Function
ICLD	Inter-Channel Level Difference
ICTD	Inter-Channel Time Difference
MIDI	Musical Instrument Digital Interface
MOCAP	Motion Capture
OM	OpenMusic
OSC	OpenSoundControl

PML	Performance Markup Language
SDIF	Sound Description Interchange Format
SUG	Space Unit Generator
SpatDIF	Spatial Sound Description Interchange Format
VBAP	Vector Base Amplitude Panning
ViMiC	Virtual Microphone Control
WIMP	Windows, Icons, Menus, Pointers
WFS	Wave Field Synthesis

Chapter 1

Background and Motivation

"In go the program and some parameters, and out comes music."

Miller Puckette, *The OM Composer's Book: Volume 1.*

1.1 A Brief History of Computer-Aided Composition

1.1.1 1st Generation: Early Systems

The earliest experiments using computers for producing music date back to the early 1950s, often carried out by mathematicians or engineers, and involved not only musical structures but also generation of elementary sounds. The birth cry of computer music was probably the rendition of simple tunes by the CSIRAC computer in 1951 [77]. In 1955, Caplin and Prinz programmed Mozart's "Musikalisches Würfelspiel" (for recombination of melodic fragments) on Manchester University's Ferranti Mark 1 computer, sonified via a sound synthesis algorithm described by Turing [14]. In 1956 Bolitho and Klein developed a program for melody generation based on random numbers (representing pitches) and rule-based selection. About the same time Pinkerton developed the *banal tune maker* which performed a probabilistic analysis and resynthesis of "nursery tunes". In the same year, Cohen and Sowa developed *A Machine to Compose Music* which generated tunes using similar ideas [130]. The first original musical piece composed by a digital computer is probably the *Illiac Suite for String Quartet* by Isaacson and Hiller in 1957 [109]. The program was run on the ILLIAC 1 (a supercomputer at University of Illinois) and the output was in the form of tables that had to be manually transcribed into a symbolic

score to be played by human performers. Consisting of four movements conceived as computer music "experiments", Hiller's techniques involved the use of the Monte-Carlo algorithm and higher-order markov chains [10]. From 1955 to 1965 a surprising number of other computer composition experiments were carried out which, unfortunately, were not documented well [14]. With *Project 1* and *Project 2* (1965-1970) Koenig reintroduced human-decision-making into the computer composition process: The global form was specified deterministically by the user, and the program carried out the periodic or aperiodic distribution of materials, following serial organizational principles [124].

This notion of algorithmic composition is still alive, albeit more popular in the United States than in Europe. Cope's *Experiments in Musical Intelligence* for instance, employs databases of musical "signatures" or "genetic materials" which are recombined to simulate a certain musical style (e.g. Bach, Mozart, Beethoven). Cope refers to his method as "non-linear" and "linguistics-based" composition [64]. Other examples based on artificial intelligence-related approaches are the works of Truax, Pope and Haus [19, 108].

A different approach was taken by Xenakis in his *Stochastic Music Program* (SMP, 1962) [237], which substitutes syntactical relationships with mathematical or physical models, controlling distributions of sound events in the time-frequency plane. SMP interlinks probability functions which determine both macro- (e.g. length of sections) and meso-level (e.g. pitch, duration) aspects of a composition. It was a computer implementation of the same compositional methods that Xenakis carried out manually in his piece *Achorripsis* [134].

What characterizes these early CAC systems, is that the composition program was designed a priori, and once the input data was specified, the act of music generation was carried out by the program autonomously. It should be noted, however, that the idea of "removing" a composer's subjective choices from the musical result can also be seen in the perspective of post-war compositional aesthetics. This notion of computer composition is well-described in the following quote by Xenakis [236]:

> "Freed from tedious calculations, the composer is able to devote himself to the general problems that the new musical form poses and to explore the nooks and crannies of this form while modifying the values of the input data."

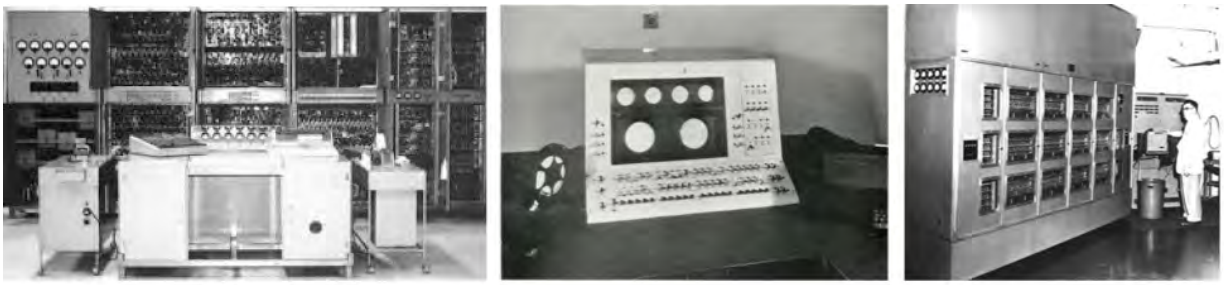

Fig. 1.1: From left to right: The CSIRAC, Ferranti Mark 1, and ILLIAC 1 Computers.

1.1.2 2nd Generation: Programming Languages and Graphical Interfaces

A significant step towards "assisted" (as opposed to "automatic") composition was taken in *Musicomp* (Music Simulator Interpreter for Compositional Procedures) developed from 1963 to 1972 mostly on the IBM 7090 computer by Baker/Hiller, intended as a tool to solve specific musical problems by providing access to subroutines for generation, selection and modification of music materials. It is considered one of the earliest examples of the use of a programming language for composition, in that it was not a program to produce a musical piece (or "output"), but a system that allowed its user to specify rules for how to produce this output [110]. The introduction of programming languages created a new paradigm, which was both freeing –now composers could design their own programs– but also intimidating, since it required to formalize musical ideas and "translate" them into a programming language. This initiated a research field interesting both for composition and computer science: the design of programming languages for expressing and modelling musical thought. The use of programming languages instead of "applications" (e.g. as used for commercial music software) also has the advantage to open the composition system to different musical and aesthetical directions. The composition system was not anymore a program designed for a specific task (implementing a certain musical viewpoint), but a generic, programmable system. As stated by Assayag [16]:

> "one cannot imagine any more a CAC environment as a rigid application offering a finished collection of procedures for generation and transformation. On the contrary, we conceive such an environment as a specialized computer language that composers will use to build their own musical universe."

Following Musicomp many composition environments have been developed as specialized programming languages using different programming paradigms. Here we should mention the *Mode* (Musical Object Development Environment [173]) and *Formes* [187] environments for object-oriented programming. The latter was intended for composition and synthesis of sound, and is probably the first environment addressing the integration of signals in symbolic composition –which is today one of the major research axes in CAC. Other notable works implementing constraint-based programming and musical knowledge-representations (of tonal harmony) are *MusES* by Pachet [162] and the *Carla* environment by Courtot [66]. *Carla* was one of the earliest developments towards "visual programming" systems (described hereafter), providing a set of types with associated heuristics, a logic programming environment and a graphical interface for formalizing relationships between these types.

Functional programming languages turned out to be favourable for compositional environments, as they presented a number of useful properties. LISP is a functional, interpreted language with a simple and powerful syntax (s-expressions) which allowed composers to interactively develop and modify programs (and even the language itself) at run-time, i.e. during the creative process [204]. This extended the interaction paradigm with the composition environment from the mere input of data, to actual designing of the programs processing this data, blurring the distinction between *making* the program and *using* it. *Crime* was probably the first LISP-based composition system to carry out general symbolic musical formalisms (rhythmic, harmonic, etc.) and visualize them in the form of a traditional score [17]. It also included psychoacoustic models and was used by a number of renown composers at the time, including Malherbe, Stroppa, Benjamin, Lindberg, and Saariaho. Another LISP-based environment which was developed partially at Stanford University (USA) and at ZKM (Germany), "as a response to the proliferation of different audio hardware, software and computers", is Taube's *Common Music* [210]. This system enables composition of musical patterns and their transformation into a variety of protocols for sound synthesis and display (MIDI, Common music notation, VRML, etc.). Together with *PWGL* and *OpenMusic* (described later) it is one of the popular LISP-based CAC systems which are in active development and use today.

The second generation of CAC systems is characterized by the use of high-level programming languages supporting modern concepts (such as constraints, objects, abstraction, etc.), graphical interfaces, and music notation. These systems also introduced

a different conception of the role of the computer, as a tool for modelling compositional processes in an interactive dialog, rather than an automaton producing an entire musical work. Gill seemed to anticipate this development, stating in 1963 [99]:

> "Perhaps in the end we shall see musical composition taking the form of a co-operative venture between the human composer and the computer [...] It would probably be done most effectively by means of a time-sharing program in a very powerful computer, since it would require spasmodic bursts of rapid computing."

We should also mention the emergence of a related branch of CAC systems intended for "interactive" or "realtime" composition, with the probably first digital system in 1977 by Chadabe [60]. In the 1980s a number of systems were developed, blurring the lines between composition, performance, and improvisation. Historic examples of this kind of software are Spiegel's *Music Mouse* [203], or the *M* and *Jam factory* softwares [239], which allowed the use of mouse and keyboard for adjusting parameters for music-generation processes (markov chains, probabilities, etc.) in real time. A more contemporary example, implemented as a set of objects for the Max software [175], is Essl's *RTC* (real-time-composition) library.[1] As we focus here on systems for composition that are not "performed" in real-time we will not discuss this branch further.

1.1.3 3rd Generation: Visual programming, Sound, and User Libraries

A novel approach, and maybe the start of a third generation of CAC systems, was taken in Laurson's *PatchWork* (PW), a LISP-based environment which implemented a "patching" interface for visual programming [125]. While sophisticated graphical interfaces for computer-aided composition had been used before, the possibility of defining LISP programs as a visual control graph was an innovation. Objects (*boxes* in PW) are placed in a patch editor and connected together, thereby defining the functional composition of a program: each box produces its output by requesting upstream objects in the graph to compute their outputs. PW provides a large collection of pre-defined boxes which represent musically "neutral" functional units (with respect to the material that is to be produced with it). The environment can also be extended with new functionalities, bundled

[1] http://www.essl.at/works/rtc.html, accessed June 25, 2016

together in the form of dynamically loaded external libraries. The data produced by boxes can be interpreted in various ways, e.g. as break point functions or musical structures. Graphical editors can be used used for visualizing and (destructive) editing of symbolic music representations, such as notes, chords, rhythms, etc. Figure 1.2 shows a Patchwork patch editor.

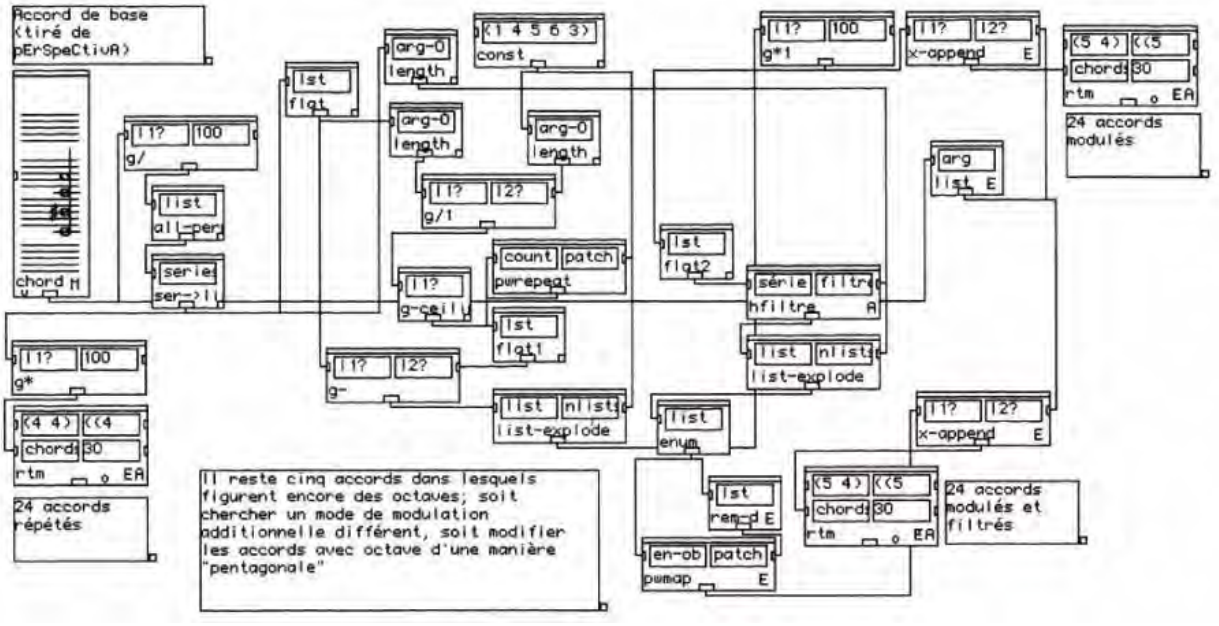

Fig. 1.2: A visual program in Patchwork as a graph of connected boxes.

Visual Programming (VP) is defined as "any system that allows the user to specify a program in a two (or more) dimensional fashion" [153]. VP tends to offer a higher-level, more intuitive description of programs by de-emphasizing syntactical issues, making the programming task more accessible for non-expert programmers [41]. Patchwork provided an interface which allowed to express musical ideas integrating both textual and visual programming in a fluidity previously impossible. This allowed users with varying degrees of programming expertise to overcome the intimidating learning curve of textual programming and progressively design more complex programs. This characteristic, combined with the graphical editors and the use of external libraries, has proven to be a significant step forward, shown in the wide acceptance by a large number of composers of different musical styles. Some pioneering users were for instance Murail, Ferneyough, Malherbe, and Grisey.

Two direct descendants of Patchwork which inherit the same visual programming paradigm and are currently among the most sophisticated systems in this domain are *PatchWork Graphical Language* (PWGL) [127] and *OpenMusic* (OM) [42]. As OM is the host environment in which our developments take place, it will be described hereafter in section 1.2. Both systems implement a visual programming interface on top of the Common LISP/CLOS language [91], however, they take a slightly different approach: while OM has a more language-oriented design in that almost all programming concepts (incl. object-oriented programming) can be carried out visually, PWGL has a scalable, OpenGL-based [160] visual interface and a sophisticated notation package (*Expressive Notation Package* [122]). Other notable features of PWGL are the integrated constraint programming system [126] and functionalities for sound analysis/synthesis (*PWGL synth* [154, 127]) which allows to analyze audio, build sound synthesis instruments and control their parameters in real time. Thus, the creation of musical scores, the analysis and synthesis of sound can be carried out in a single integrated compositional framework. This integration shows an important development step in the evolution of CAC systems, extending the scope from the domain of symbolic musical data, to the compositional representation and control of sound. Figure 1.3 shows PWGL patch editors for representation of symbolic data (left) and sound data (right).

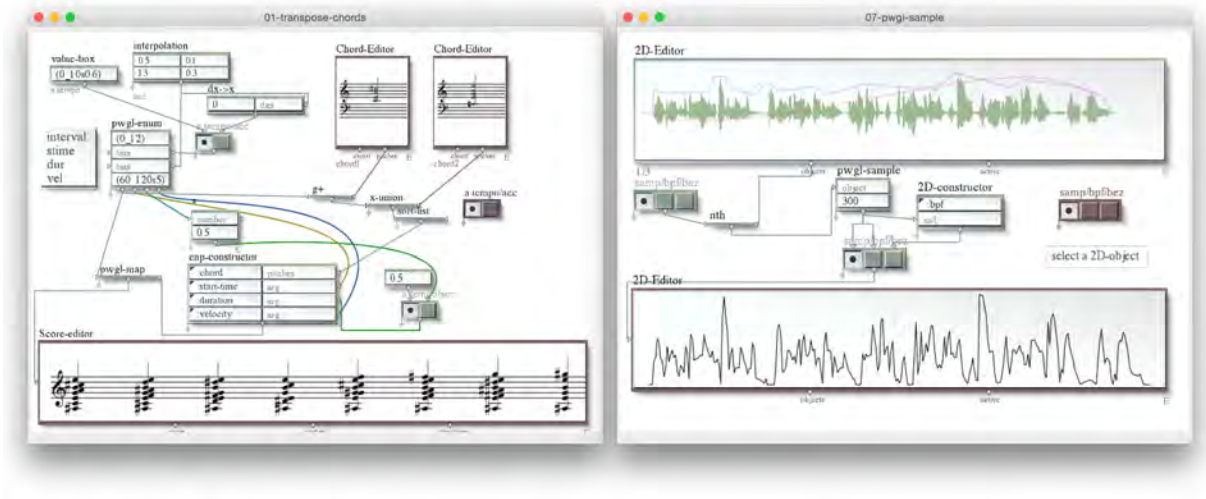

Fig. 1.3: Left: A PWGL patch for transposition of chords. Right: A break point function generated via extraction/resampling of audio data.

PWGL exists in an end-user version (PWGL Application) and as a developer version (PWGL Binaries). The latter is a pre-compiled module that can be loaded on top of the Lispworks IDE.[2] Unlike OM, PWGL is a closed-source system, therefore it is not possible for users to modify its native functionalities. PWGL and OpenMusic share a number of modern design concepts, such as:

- implemented as extensions to high-level programming languages
- open and programmable
- equivalence of visual and text-based interface
- support of communication protocols and compatibility with media standards
- graphical editors reactive to mouse and keyboard input
- multiple data representations (musical score, mathematical functions, etc.)
- running on personal computer platforms (no special hard-/software requirements)

1.1.4 Extensions and Integration of other Musical Data

The possibility of extending the CAC environment with user libraries stimulated a rich variety of developments, often dedicated to specific compositional problems or aesthetic directions. The *Chaos/Alea* libraries by Malt, for example, implemented probabilistic functions and dynamic systems, taking inspiration from Xenakis' stochastic approaches. A different direction was taken by Pachet, Rueda, and Truchet, for the modelling of musical situations as constraint-satisfaction problems, resulting in the libraries *OMsituation* [27], *PWConstraints* [126], or *OMClouds* [217] for adaptive searches. IRCAM's Music Representations team also implemented a number of libraries, such as *Repmus* for generation of harmonic structures, and *Kant* for rhythm quantification [7].

These developments reflected both the scientific interest, but also, maybe more importantly, the needs of composers who were searching for possibilities to integrate and explore their musical approaches within the composition environment. This is most evident from the numerous examples of contemporary composers who started developing their own libraries reflecting musical aesthetics/concerns. One of the earlier examples is the *Esquisse*

[2] http://www.lispworks.com/news/news34.html, accessed June 25, 2016

library by Murail, for spectral music operations, developed together with Baisnée and Duthen. Ferneyhough, was an early user of PatchWork, and the OM library *OMCombine* developed by Malt takes inspiration from his approaches. Many of these libraries were ported from one CAC system to another. They can be regarded as distinct, satellite CAC projects, dedicated to specific musical problem domains.

However, not only musical aesthetics developed, also music technologies were advancing at a fast pace and allowed for new possibilities and musical practices, such as new possibilities for sound analysis and synthesis. Although these functionalities were available as external applications or tools for real-time performance, composers were interested in including the control of these aspects into the same environment where other compositional formalisms and models were developed.

Stroppa, for example, interested in high-level sound synthesis control, co-developed the *OMChroma* [8] and *OM-Chant* [88] libraries for the OpenMusic environment. The former is a generalized system for high-level specification of control data for the "Csound" language [87], the latter, an interface for compositional control of the "Chant" synthesizer (for singing voice synthesis), developed in the 1980s at IRCAM. Lanza's interest in physical modeling synthesis led to the *OM-Modalys* library, an interface for the "Modalys" synthesizer [81]. Haddad co-developed the *OM2Csound* library, for visual programming of Csound score- and orchestra files.

1.2 OpenMusic

OpenMusic is an open-source, cross-platform, visual programming language based on Common LISP / CLOS [91], developed since the mid 1990s by the Music Representations Team at IRCAM [4]. Similar to PWGL it inherits many concepts of PatchWork (patching metaphor, graphical editors, etc.) and extends these with new programming features. OM provides visual counterparts for most of the programming concepts of CL/CLOS, such as abstraction, recursion, control structures, and object-oriented programming [41]. It implements the concept of a duality of "object" and "process" expressed through classes (corresponding to musical data) and functions for operating on them. Files, settings, and other resources are grouped into *workspaces* (conceptually similar to projects), which can be browsed using windows and menus, as known from today's popular operating systems. The principal programming interfaces in OM are *patches*, which are described hereafter.

1.2.1 Patches and Abstractions

A patch in OM is a visual program: the user is presented with an editor (a white canvas) into which graphical *boxes* are placed which represent functional units or data-structures, and which can be connected together (by drawing lines between them) to design algorithms. Boxes have a number of inlets (at the top), representing arguments, and at least one output (at the bottom), representing returned value(s). A box can be a native OM or LISP primitive function, a (graphically or textually) user-defined function, or an embedded patch (there is also a special *Lisp Function* box which allows to write a LISP lambda expression in an editor).[3] The boxes represent function calls and the interconnection of these boxes represent the structure of the program (a directed, acyclic graph). The computing paradigm is "request-driven": The evaluation of a box triggers a chain of function calls following the visual graph upwards until the terminal box is reached which generates a dataflow downwards to the initial box where the requested value is returned.[4] This "bottom-up" call of the visual program graph corresponds very much to the evaluation of a LISP expression. Note, that is also possible to selectively evaluate nodes of the graph, which facilitates incremental and explorative programming. Moreover, the results of a partial evaluation of the program can be stored by "locking" a box (meaning, that upon evaluation it will return its last computed value), which can eventually be "unlocked" again. This essentially corresponds to turning a process into data and vice versa. Parts of a visual program can be "encapsulated" into a patch (thus, turning it into a box) and persistently stored by dragging it into the workspace. This corresponds to the programming concept of *abstraction*, useful for managing complexity (by presenting information that is relevant in a given context while hiding information that is irrelevant or redundant) [18]. Abstractions can be embedded in patches just like other boxes and may themselves contain other patches or abstractions. They can also be turned into local patches, i.e. become part of a parent patch. In the visual interface, this difference between local patches and abstractions is reflected through different colors of the patch icons. Figure 1.4 shows an example of a patch containing some of the objects described above.

[3] The equivalence of textual and visual programming is an important concept; it allows users to choose between textual and visual representations based on expertise or suitability for a specific situation [18].

[4] OM has recently been extended to also support the reactive programming paradigm [34].

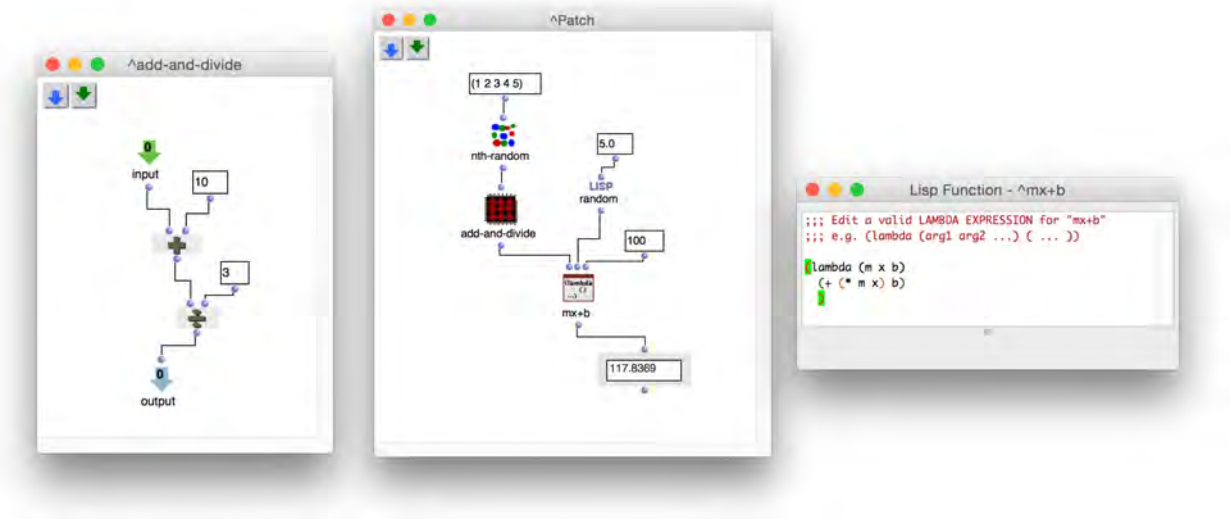

Fig. 1.4: An OpenMusic patch editor. Middle: a visual program as a graph connecting a LISP primitive function, OM native function, a subpatch and a *Lisp Function* box. Left: patch editor of the subpatch "add-and-divide". Right: The *Lisp Function* box containing textual code.

1.2.2 Factories, Editors, and Higher-order Functions

The Common LISP Object System (CLOS, [91]) offers a powerful system uniting functional and object-oriented programming which is reflected in OM in an elegant way. OM comes with various pre-defined classes representing musical or abstract objects (such as notes, chords, break point functions, etc.) Classes implement the notion of "material" (as opposed to "processes") and are integrated in the visual programming paradigm via the concept of *factories*, i.e. special functions for generating instances of the classes they represent. Factory boxes are usually associated to a graphical editor which displays the most recently created instance and exposes it to interactive editing (and/or playback). Some examples are score editors with linear or metric time notation, but OM also provides editors for other types of data and media, such as break point functions, MIDI files and audio. The iconic representation of factory boxes inside a patch can optionally be switched to display a small preview of their contents, called a "miniview". Factory boxes also have inlets and outlets which allow to set and access values of the public slots of the contained class in computational processes. The graphical editors in OM exemplify fundamental aspects of our notion of computer-aided composition environments, namely the duality of manual

intervention and interaction, possible at different stages of the calculus [5]. Figure 1.5 shows a *voice* factory box in a patch (displaying its contents as a miniview) and the corresponding graphical editors.

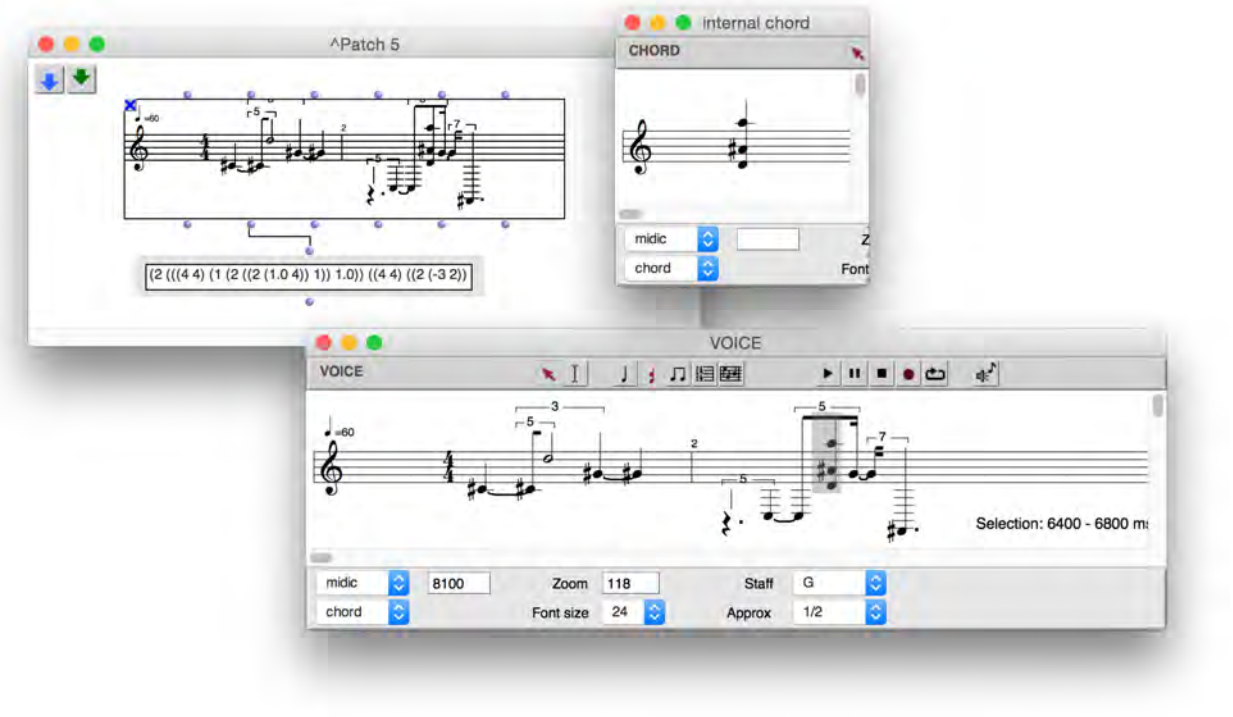

Fig. 1.5: An OM patch containing a *factory* box for a *voice* object (a symbolic musical score) displaying its contents in a *miniview*. The rhythmic specification (a *rhythm tree* is accessed through the outlet and displayed in a *text-box* object. Bottom/Right: The associated graphical editor windows for manual interaction.

An original feature of OM is the possibility of using visual editors for object-oriented programming, i.e. defining new classes or subclasses and build inheritance relationships. It is also possible to visually define new methods for polymorphic functions. Since the meta classes defining OM can be subclassed the same way, it is also possible to modify the environment itself (*reflexivity* concept in object-oriented programming).

To conclude this subsection, we should mention OM's implementation of a functional programming concept which has proven powerful and flexible for compositional applications, and which we benefit from in different parts of our tools as well (cf. sections 2.4.2 and 3.5.2): the equivalence of data and function ("first-class" objects in LISP). Functions can be constructed in computational processes and passed on as arguments (like data) to other ("higher-order") functions. This programming concept is implemented in OM's visual interface by setting a box to different evaluation modes (visible through a small character in the top left corner of the box icon). In "lambda" mode, a box returns its reference call as a functional object (anonymous function) instead of returning the value resulting from its application. When "locked", a box returns its most recent computed value (similar to a function returning a constant). When set to "Eval-once" a box will be evaluated only once during evaluation of its containing graph. Figure 1.6 shows an OM patch, a LISP primitive function and an OM-native function with different evaluation modes.

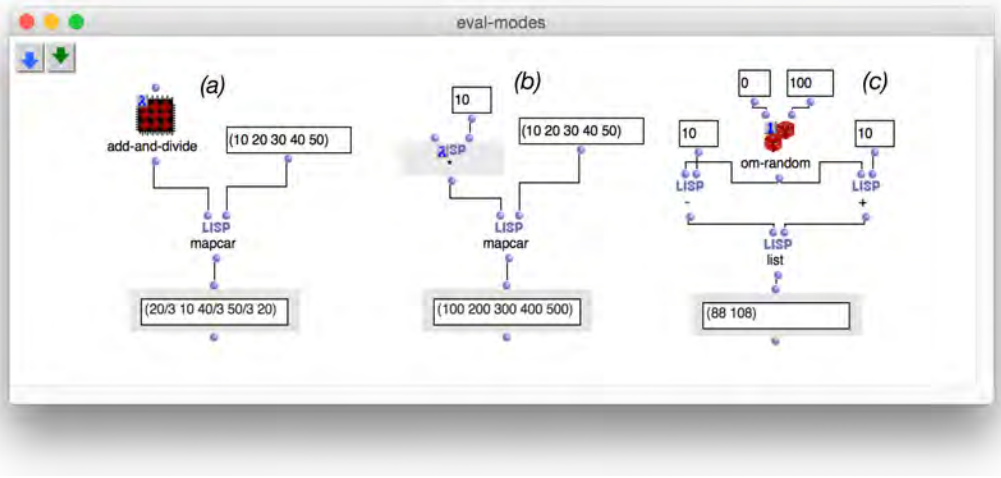

Fig. 1.6: (a): The subpatch "add+divide" as used in Figure 1.4. The box is set to "lambda" mode, and passed on as an argument to the higher-order function *mapcar*, which applies it as an anonymous function to each of the list elements connected to its right inlet. (b): The use of a LISP primite function for multiplication in "lambda" mode. Note, that one argument has been set directly (also referred to as *currying*). (c): The OM-native function *om-random* is set to "eval-once" mode and produces a random number only once during evaluation of the graph.

1.2.3 Temporal and Hierarchical Structures

The musical algorithms developed in OM patches can be embedded into temporal and hierarchical structures, expressed through a dedicated object, titled *Maquette* [4]. This object provides a special patch editor with a horizontal timeline and can be used at the same time as a temporal container and as a visual program. Musical objects with a temporal dimension (such as score objects, or even other maquettes) can be placed into a maquette and are automatically aligned and resized within the timeline. Maquettes may also contain *temporalboxes*, i.e. special patches which provide data denoting their spatial position and dimensions inside a maquette. These data can be integrated in computations defined inside the temporalbox, thus depending on its temporal/spatial properties relative to the containing maquette. Maquettes can also play back temporal musical objects (such as audio files and MIDI data) in the fashion of a sequencer. Figure 1.7 shows a *Maquette* containing several objects, including an embedded *Maquette*.

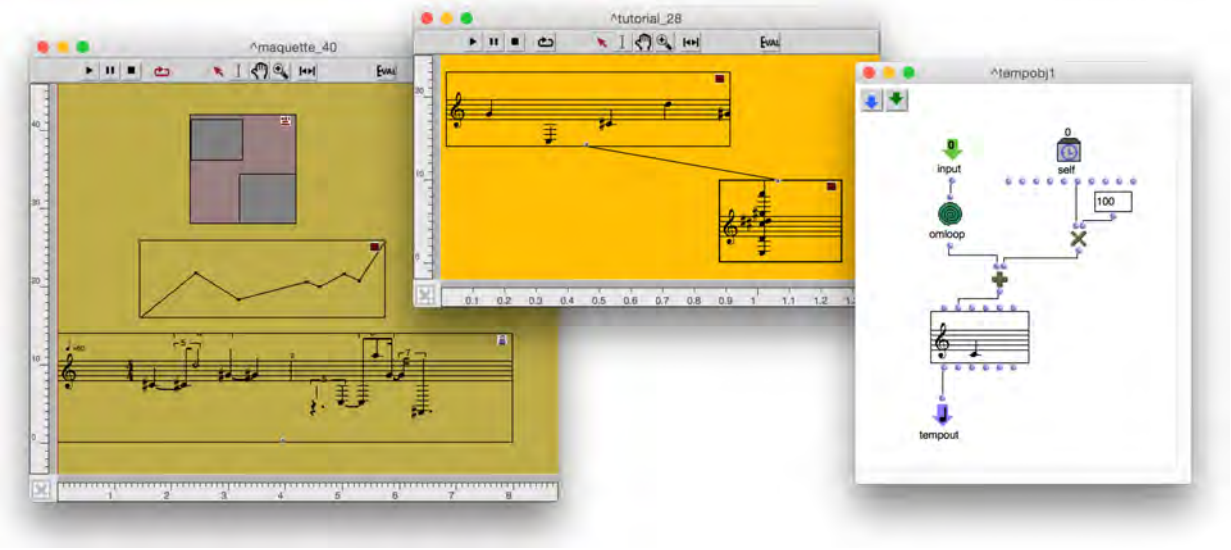

Fig. 1.7: Left: A *Maquette* object containing a *voice* object, a patch producing a *break point function* object, and a *Maquette* object. Middle: View of the embedded *Maquette* object. Bottom Right: A temporal object producing the *chord* displayed in the embedded *Maquette*.

1.2.4 Interchange Protocols and User-Libraries

OpenMusic provides a number of interfaces and protocols to interchange musical data with other applications, such as SDIF [235], OSC [233], MIDI [150], and MusicXML [102], and can be extended with dynamically loaded libraries. Many libraries dedicated to specific musical problem domains have been implemented, e.g. for constraint-solving [216], rhythm quantification [149], sound processing, harmonic functions, etc. An overview of currently maintained libraries can be found online.[5] This modularity has allowed for applications of OM in a variety of musical research fields. A number of libraries dedicated to sound analysis/synthesis and processing have been developed which are related to our works and deserve to be mentioned here. Notably, the libraries *OM-SuperVP* and *OM-pm2* provide interfaces to the corresponding sound analysis kernels for Phase-Vocoder analysis and partial modelling [32]. Dedicated libraries for the control of the sound synthesis are *OM2Csound* and *OMchroma* [8] for the Csound language and *OM-Chant* for controlling IRCAM's *Chant* synthesizer [88]. Figure 1.8 shows examples for the extraction of symbolic structures from an audio file (left), and for sound processing using symbolic specifications (right).

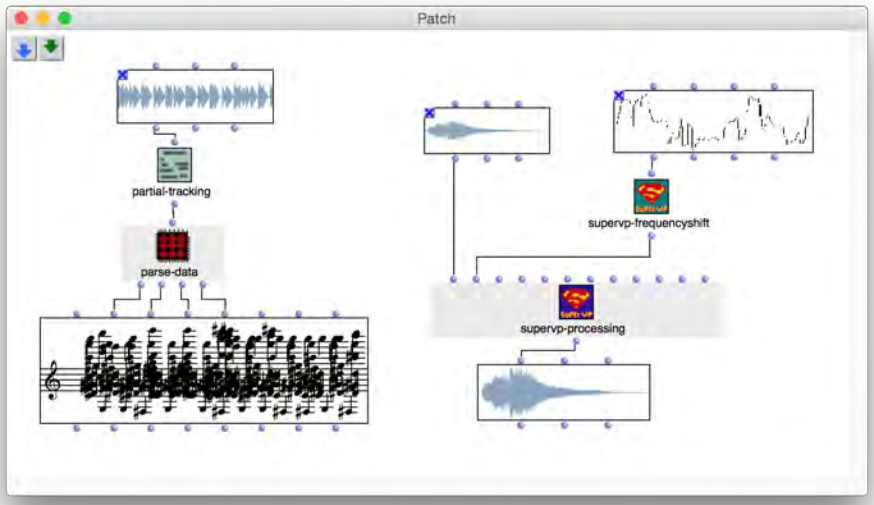

Fig. 1.8: Left: extracting symbolic data from a sound file using the *OM-pm2* library. Right: sound processing (frequency shifting) using the *OM-SuperVP* library, controlled by a break point function object in OM.

[5] http://repmus.ircam.fr/openmusic/libraries, accessed June 25, 2016

1.3 State of the Art(s)

1.3.1 Space

Spatialization has always been of great interest for music composition. Even before dedicated technologies were available, composers developed custom spatialization devices to meet their artistic needs, notable examples are Stockhausen's rotational table [26] or Schaeffer's "potentiomètre d'espace" [211]. During the past 20 years the "spatiality" in music has gained enormous momentum through the development and proliferation of advanced spatial audio technologies. Modern spatialization systems are capable of rendering spatial sound fields over extended areas (e.g. Wave Field Synthesis [21], Higher-Order Ambisonics [68]), a large number of research centres and concert venues are equipped with large-scale loudspeaker arrays, and the use of spatial elements is an increasingly important aspect of compositional practice [161]. Using state-of-the-art computer technologies it is possible to carry out sophisticated signal processing for many channels of audio, which allows for the synthesis of spatial attributes from the macro-scale of a composition down to the level of micro-level of sound [185]. Studies have revealed, that composers seem to not exploit these technologies in their musical works [161]. Maybe this is related to many tools still being based on concepts and control models introduced roughly forty years ago [62] which are less interesting for contemporary compositional practice, and far from exploiting the rich potential of musical possibilities. Indeed, composers have expressed not having adequate tools for their musical needs [170]. The discrepancy between spatialization technologies and their use in compositional practice is arguably related to two main factors.

The first concerns the lack of flexible spatialization tools and musically-relevant representations that can be associated to the symbolic domain of composition. Although a number of developments have specifically tackled the high-level control of spatial parameters, e.g. by providing tools for trajectory generation or constraint propagation [174, 213, 164], these systems lacked integration with programmable environments and abstract representations in order to accommodate different compositional applications. There have also been previous attempts to integrate the control of spatialization into the OpenMusic environment [72, 156], however these tools were restrictive in terms of complexity of control, and were bound to a specific synthesis system. While there have been interesting new approaches and control-models for real-time spatialization, e.g. based

on dynamic systems [189, 120] or gesture control [143], these developments lack musical representations that can be integrated into compositional contexts, and tend to be more interested in technological sophistication than in "a relevant thinking on the conditions of emergence of a new artistic language" [16]. Indeed, sound spatialization is often treated as an effect or decoration, rather than an integral part of the composition [155]. It is surprising, that although Stockhausen stated in the late 1980s that spatial parameters are musically as relevant as melody or harmony [205], at the time we started the works described in this dissertation (20 years later), there was yet no system or tool allowing to integrate and relate spatial sound with other musical materials in a compositional framework.

The second problem stems from a conceptual dissociation of sound- and spatial attributes. Although physically and perceptually related, they often need to be developed in different environments and at different stages of the compositional process. Most existing tools and interfaces are based on the concept of space-less sounds to which spatial attributes are added in a separate process, promoting a disconnect between sound and spatial parameters. This separation might be due to historic and pragmatic reasons (e.g. for separation of production workflows), however, it might hinder the development of new artistic directions. While a number of tools for real-time spatialization have started to experiment with new techniques, e.g. spectral and granular spatialization [121, 232], compositional tools mostly implement interfaces based on the physical metaphor of virtual objects that are placed and animated in a scene. Normandeau describes this as a historic artefact of instrumental music concepts, comparing it to early cinematic works which were essentially "filmed theatre", before cinematographic techniques (such as moving cameras, close-ups, etc.) were developed [155]. From an acoustics viewpoint, sound is inherently spatial (waves travelling through a medium), and the perceptual separation of timbral and spatial attributes is only achieved through complex processes which are not completely understood yet (this is what is studied in the field of auditory scene analysis [31]). Indeed, the psychoacoustic relationships between timbre and space offer a wide and fascinating field for artistic exploration (cf. chapter 4) which unfortunately is beyond the reach of current composition tools.

1.3.2 Gesture

While the use of gesture sensing technologies for instrumental performance has a tradition of about 100 years now, there are only sporadic historic examples for their use in compositional contexts (e.g. UPIC, cf. section 3.2), and these applications often required dedicated soft- or hardware. The digital and mobile revolutions in the past two decades have brought a plethora of new input devices as well as soft- and hardware platforms which are powerful, inexpensive, and accessible to a large community of end-users. Looking at today's research literature on gestures related to music [224] or the calls for music for the major computer music conferences (e.g. International Computer Music Conference (ICMC)[6], Sound and Music Computing (SMC)[7], or New Interfaces for Musical Expression (NIME)[8]), it becomes evident that gesture control is now an essential part of contemporary artistic practice.

Unfortunately, the term "gesture" is used in a variety of overlapping and sometimes contradicting definitions, such as in the context of "gesture control" for mobile devices (which are in fact, "actions", cf. chapter 3.2). When talking about gestures in the context of composition environments, however, we need to distinguish between alternative control methods for Human-Computer-Interaction (HCI), and gestures as compositional materials. Although there is a great body of literature on gesture-based control for music performance, the majority of this research has been carried out within the NIME community, investigating gestural interfaces from the perspective of HCI, mostly for sensor-based manipulation of real-time audio synthesis [151]. As a breakout group of the SIGCHI (Special Interest Group on Computer–Human Interaction), the main interest of the NIME community is to investigate instrumental interaction paradigms making use of (gestural) input devices which allow for more nuanced and expressive forms of computer interaction than the ubiquitous mouse, keyboard, screen interfaces [188]. While a great variety of rich and expressive tools for manipulating complex data sets have been developed in that perspective (spanning physiological data, sensing of large physical objects, computer vision, etc.), they prioritize performance over composition, and thus there is little research for the compositional use of these technologies.

[6] http://www.computermusic.org, accessed June 25, 2016
[7] http://smcnetwork.org, accessed June 25, 2016
[8] http://www.nime.org, accessed June 25, 2016

Our interests are different from the NIME developments, in that our aim is integration of gestures as musical materials, i.e. spatio-temporal (multi-dimensional) morphologies, into composition environments. Although modern CAC environments have extended the traditional scope from manipulation of symbolic music materials to the inclusion of sound signals and connectivity with real-time environments (cf. section 1.1), its main focus has mostly remained the domain of modeling abstract thought and calculus. Today the importance of physical interaction in creative processes is widely accepted, see e.g. theories of *enaction* [83], *embodied cognition* [129], *motor-mimetic cognition*, yet the intrinsic expressivity and organic nature of human gesture has so far barely been considered in the context of compositional applications. In fact, it has been argued by anthropologists, that language itself can be regarded as a form of gesture and that symbolic content and gesture are so tightly interwoven that very often they cannot be isolated [15]. Indeed, before the 19th century composers were often performing musicians and developed musical ideas via experimentation and improvisation on physical instruments. It seems there is a rich field of creative possibilities lying idle before us: while the technologies (tablets, hands-free controllers, force/haptic interfaces, etc.) are ready available, there is a lack of tools and strategies which enable to integrate the physicality of human expression into computer-aided composition models.

Some research works have presented the use of interactive pen/paper technologies (for augmented drawing) [218] or commercial gaming controllers (as a 3-dimensional mouse replacement) [94] for compositional applications. The interest of these works, however, is aligned with the NIME research, investigating the applicability of gestural controllers as alternative input devices for interaction with the CAC environment (HCI): the gestures themselves are not used as musical materials, but rather the data being manipulated with these devices (this aspect is discussed in more detail in section 3.2). A notable work which is closer to our interests in terms of representation and processing of the gesture signals themselves, is the "gesture editor" presented by Ramstein [181]. However, this system was designed for instrumental performance gestures, and thus excludes other types (e.g. freehand gestures). Moreover, it does not provide the possibility of embedding these data in compositional processes. It is surprising, that while composition environments have developed powerful frameworks for integrating data related to the *result* of physical actions, such as the velocity of pressing down a key (MIDI, [39]), composers cannot exploit the rich, spatio-temporal morphologies of physical gestures in their compositional models.

We argue that there is a need for tools which allow to integrate and represent gesture data in computer-aided composition environments. This integration requires strategies and solutions to a number of problems, pertaining to 1) acquisition of gesture, i.e. description, coding, and storage of sensor signals; 2) high-level representations and control interfaces, specifically designed for manipulation of these data in the symbolic contexts of computer-aided composition; 3) development of tools for processing and mapping between gesture signals and other music representations in the environment.

1) Embedding gesture data in offline-contexts means to work with recordings of gesture data. This naturally requires the development of tools for description, coding, and storage. Descriptions should be at the same time generic (i.e. describing the gesture independently of the device measuring it), have strong semantics (high-level descriptors), and lend themselves to efficient compositional manipulations. To account for the different and growing number of systems, coding and storage would ideally be accomplished using a format that is user-extensible, and structurally flexible in order to code the various temporalities and dimensionalities of gesture data [133]. In addition, it would be desirable to use a lightweight format due to the considerable amount of data dealt with in these contexts (e.g. motion capture) and which is supported by many environments. As description, coding and storage are also fields of interest for the general gesture research community [115], it should be possible to draw knowledge from existing research.

2) In order to be integrated into compositional models, continuous gesture data (multi-dimensional signals) need to be discretized into individual elements that can be manipulated and set in relationship to each other [37, 18]. This requires functionalities for segmentation of multi-dimensional data streams (see e.g. [23]). This is a ubiquituous topic in the literature and can be based on different models (e.g. hidden markov models [89], dictionaries [55]), however it may also be part of the compositional conception itself. The individual segments need to be converted from the concrete data into musically-relevant, higher-level representations in order to approach the symbolic representations required for integration into compositional formalisms. These representations function as models, providing an interaction context and affordances for inspection and editing [20].

3) While in instrumental applications processing of gesture data usually refers to applying conditioning or preparatory operations (such as smoothing, filtering, scaling, etc.) for mapping purposes [151], a distinct characteristic of compositional environments is the application of non-causal and temporal manipulations, impossible to perform in real-time.

While the available programming tools in a CAC environment allow composers to design a wide range of processing algorithms, it would be desirable to provide composers with a library of functions for common operations which they can readily use (see e.g. [139]). At some point in the compositional process the gesture data needs to be converted into a direct music representation (such as a symbolic score, or a sound). This conversion process can be regarded as a form of "mapping", which in the design of digital musical instruments essentially describes the association of gestural variables to synthesis parameters [136]. CAC environments present a radically different paradigm: in compositional contexts, temporal manipulations, as well as the use of generative-algorithmic processes and literal specifications can all become interdependent components of a mapping process, which thus needs to be considered from a more abstract level. The output of mapping processes might vary in terms of scale and resolution (from micro- to macro-structural levels) as well as the domain of the representation (sound synthesis, symbolic materials), and thus new approaches need to be developed which allow going beyond models and theories in the context of instrument control [74].

1.3.3 Sound

It is well known, that the low-level representation of digital sound as a sequence of numbers is not useful for composition, as the individual informational quanta do not provide meaningful information about the characteristics of the sound phenomenon as a symbolic musical object [36]. In compositional contexts, higher-level sound representations are required that allow to describe and manipulate sound via a reduced set of more meaningful parameters (e.g. based on perceptual or physical models) [33]. From the many existing libraries and sound analysis/synthesis tools in CAC environments (cf. section 1.1.4) it can be seen that there is a need for composers to be able to work with higher-level sound representations and integrate them in their symbolic compositional models and processes. This integration has turned out to be a fruitful avenue, leading to many novel compositional approaches [40]. Sound representations have not only served as control tools for sound synthesis but have also served as conceptual models and inspired musical thinking, e.g. the French spectralist school (cf. section 4.2). Virtually all of the available tools, however, implement sinusoidal models (based on the Fourier transform), representing sound as a superposition of sinusoidal frequency components. These models are probably favored on

the one hand due to their abstract nature (pure sinusoids), conceptually close to pitches in a musical score. On the other hand, due to their perceptual and historic implications, uniting symbolic musical concepts of harmony with the more concrete reality of timbre. Although often considered as a neutral sound representation, there are a number of peculiarities associated with this model. First, a frequency-domain representation depends on a number of parameters for the transform (windowsize, window function) and requires additional algorithms for creating a higher-level representation (e.g. partial tracking). Secondly, sinusoidal partials (mathematical functions) are not representative for acoustic sounds encountered in the real world and thus are quite far from the reality of, for instance, instrumental timbres. Thirdly, due to the nature of the transform being a tradeoff between frequential and temporal resolution, sinusoidal models are less faithful for representing the temporal fine-structure of sound. This makes them less adequate for modelling of non-stationary and non-harmonic sounds, such as percussion, noise and transients, which can be found in many extended instrumental techniques of the contemporary music repertoire (e.g. Lachenmann's *musique concrète instrumentale*). Lastly, (as any model) it also brings a conceptual bias, in this case, regarding sound as a vertical structure of (relatively slow evolving) frequency elements.

This discrepancy can break the conceptual link between the concrete domain of sound and its symbolic domain of music representations, and can therefore hinder composers from integrating these sounds and relate them to their compositional formalisms in similar ways that it is possible with more stationary, harmonic sounds. Indeed, timbral aspects of noise and non-pitched sounds have become an increasingly important part of contemporary compositional language, often replacing traditional concepts of harmony, see e.g. the dedicated symposia and book publications [58]. That there is a desire of composers to work with these types of timbres and morphologies can also be seen from recent publications proposing composition with alternative sound models [104] and tools to interface these models with the environment [80].

A second aspect of this problematic relates to the increased interest and practice for linking concrete sounds and instrumental writing [159]. A number of tools are available which allow for modeling or "imitating" the morphology of a referential sound (sometimes called a "target") either using electronics or via acoustic instruments (as an instrumental transcription). These approaches have appeared in the literature under different terms, incl. "audio mosaicing" [197], "mimetic instrumental synthesis" [159],

"synthrumentation" [172], "musical onomatopoeia" [59], "micro montage composition" [208], "corpus-based transcription" [80], "computer-aided orchestration" [56], "phono realism" [184]. Unfortunately, most of these tools are developed as individual applications or parts of real-time systems and thus cannot be integrated with the formal and symbolic musical materials developed in CAC. Inversely, these tools are all examples showing that there is a need for alternative sound models which are not available in current CAC environments.

To give a concrete example: In 2011 I received a commission from the Montreal-based culture initiative "Codes d'Accès" for a new work for digital percussion (a Roland TD-20 virtual drum kit with a sound-generating MIDI unit)[9] and electronics. After initial recording sessions it became clear that the noisy sounds resulting as byproducts of the sensing mechanism offered a much richer and more interesting sound universe than the synthetic sounds of the standard MIDI unit. Unfortunately, at the time there were no existing compositional tools that allowed to represent the rich, temporal morphologies of these noisy sound structures. Thus, it was impossible to find abstract representations that were able to: 1) capture and represent the perceptually and structurally salient aspects of these sounds, and 2) link them to other symbolic materials in the composition environment.

1.4 Perspectives for Improved Compositional Tools

The history of CAC systems can be seen as a development leading from the use of specialized programs written in low-level languages for dedicated hardware, to open systems based on high-level programming languages running on generic computer platforms. From a time when the only way to interact with a computer program was the use of punch cards, stems the notion of computer composition as an autonomous job carried out by the machine, generating an entire musical work according to a previously specified plan. The introduction of faster computers, graphical user interfaces and higher-level music representations has allowed a gradual shift to more interactive approaches between user and environment, extending the earlier, formal and mathematical models for *generation*, with possibilities for *experimentation* and direct *manipulation*. Exchange formats and protocols were integrated to meet the needs of composers for including sampled data (such as audio and MIDI) and develop different aspects of a musical work, such as instrumental score and electronic

[9] http://www.roland.com/products/td-20/, accessed June 25, 2016

sounds, in an integrated programming framework. Situated in the middle field between "manual composition" and "composing machine" [124], CAC systems have proven powerful platforms for exploring new musical concepts and ideas.

In the past 20 years we have experienced a historically unprecedented technological revolution and many of these developments have already found their repercussions in artistic practices (spatial audio, gesture control, signal processing). Similar to when digital sound synthesis became viable and affordable for end-users, these new technologies require corresponding efforts in the field of musical control. Our discussion of current compositional trends involving spatial, gestural, and sound modelling aspects revealed that there are many artistic directions and requirements which are not reflected in current CAC environments.

This dissertation proposes solutions for integrating these new technologies, addressing the artistic need for including them into computer-aided compositional models and formalisms. We present a coherent software framework, implemented as three libraries for the OpenMusic environment. OpenMusic was chosen as host environment for several reasons: all native code is free and open-source, which encourages developers and users to modify and extend it with new functionalities. Due to its flexible architecture we can build on the rich system of objects and editors for musical representation. This is beneficial both from a software development and maintenance perspective, but also from a usability perspective, as it does not require users to adapt and learn completely new tools and interfaces. From an academic perspective, making all code freely available fosters research by providing the possibility to learn, understand, and build on previous works. All our developments follow this academic spirit and are made available to the research community. Another motivation is the large and active user base [10] and the scope of applications for arts [6, 40], research [44], and music education [45]. It is being taught in composition and programming classes around the world [42] and is one of the few environments which have a history within an institutional artistic/research context [18], thus its design concepts have drawn from and been validated by real-world artistic experience. OpenMusic supports a variety of protocols and interchange formats and provides infrastructure for communication with external synthesis environments, making it a flexible platform for the development of our projects.

[10] http://forumnet.ircam.fr/user-groups/openmusic/, accessed June 25, 2016

1.5 Contributions

1.5.1 Publications

Peer-reviewed Journal Publications and Book Chapters

- Schumacher, M., Wanderley, M. M. (2016). Integrating Gesture Data in Computer-Aided Composition: Paradigms for Representation, Processing and Mapping. Accepted for publication in *Journal of New Music Research*, Taylor and Francis Group.

- Schumacher, M. (2016). Ab-Tasten: Atomic Sound Modeling with a Computer-controlled Grand Piano. In Bresson, J., Assayag, G. and Agon, C. (Eds.), *The OM Composer's Book: Volume 3*, Éditions Delatour France / IRCAM – Centre Pompidou. In press.

- Schumacher, M., Bresson, J. (2010). Spatial Sound Synthesis in Computer-Aided Composition. *Organised Sound*, vol. 15 (03) pp. 271–289. Cambridge University Press (UK).

Peer-reviewed Conference Proceedings

- Garcia, J., Bresson, J., Schumacher, M., Carpentier, T., Favory, X. (2015). Tools and Applications for Interactive-Algorithmic Control of Sound Spatialization in OpenMusic. Presented at the *inSONIC2015* conference, Karlsruhe, Germany.

- Bresson, J., Schumacher, M. (2011). Representation and interchange of sound spatialization data for compositional applications. Presented at the *International Computer Music Conference*, Huddersfield, UK.

- Bresson, J., Agon, C., Schumacher, M. (2010). Représentation des données de contrôle pour la spatialisation dans OpenMusic. Presented at the *Journées d'Informatique Musicale*, Rennes, France.

- Schumacher, M., Bresson, J. (2010). Compositional Control of Periphonic Sound Spatialization. Presented at the *2nd International Symposium on Ambisonics and Spherical Acoustics*, Paris, France.

1.5.2 Software development

LISP/OpenMusic libraries

- OMPrisma: Library for spatial sound synthesis. Described in chapter 2. Source code available: https://sourceforge.net/projects/omprisma/
- OM-Geste: Library for gesture composition. Described in chapter 3. Source code available: https://github.com/marleynoe
- OM-Pursuit: Library for dictionary-based sound representation. Described in chapter 4. Source code available at: https://github.com/marleynoe/OM-Pursuit

Standalone Applications

- Multiplayer.app: application for real-time decoding and diffusion of spatial audio formats. Described in [195] and in chapter 2. Requires IRCAM Forum license. Available: http://tinyurl.com/thesis-multiplayer. Native (free) version: http://tinyurl.com/thesis-multiplayer-mini
- SpatSDIF-player.app: Application for real-time streaming of spatial sound description data via OpenSoundControl [46]. Co-developed with Jean Bresson. Available: http://tinyurl.com/thesis-spatsdifplayer
- SpatDIF-viewer.app: Application for real-time 3D-visualization (OpenGL) of spatial sound description data [46]. Available: http://tinyurl.com/thesis-spat-viewer

Csound orchestras and user-defined opcodes

- Library of scriptable spatialization instruments (see Appendix B): {*ambi,babo,binaural,dbap,pan,rvbap,rvimic,spat,sug,vbap,vimic*}.*orc*
- Library of scriptable sound processing instruments for filtering {*reson,butter*}.*orc*, resonators *resonators.orc*, source decorrelation {*allpass,schroeder*}.*orc*
- Library of user-defined opcodes (UDOs) serving as components for the spatialization/processing instruments: *envelope*, *trajectory*, *xyz2aed*, *distance*, *attenuation*, *airabsorption*, *timeoftravel*.

All URLs on this page accessed June 25, 2016.

1.6 Thesis Structure

This dissertation concerns the conception and design of a structured software framework for integrating novel media and technologies for audio spatialization, gesture representation and sound modelling, into symbolic composition. While the use of these technologies has become commonplace in contemporary music practice, current computer music tools emphasize performance and production over composition, and composers can rarely include these technologies in their compositional models. We tackle this problem from the perspective of computer-aided composition, a field which is traditionally concerned with the development of musical representations and personalizable control structures, thus providing an ideal platform for the integration of these technologies into personal compositional frameworks and musical approaches. This dissertation is structured into five chapters:

The first chapter presents background and motivations: section 1.1 provided a historic overview of the development of computer-aided composition systems. Section 1.2 described the OpenMusic environment in which our developments are integrated. Section 1.3 discussed current compositional trends and the lack of corresponding tools for spatial, gestural, and sound modelling technologies.

The three manuscripts (chapters 2–4) describe conception, design and development of software libraries for integrating these new media and technologies into the OpenMusic computer-aided composition environment: Chapter 2 presents the library OMPrisma as a structured, object-oriented system which implements an abstraction layer to separate spatial authoring from rendering and reproduction, and allows to design spatialization processes related to other musical processes and materials in a compositional framework. Facilitated by high-level control structures and programming tools we introduce a novel paradigm titled *spatial sound synthesis*, which refers to the spatialization of individual components of a sound synthesis algorithm. Similar to how the development of sound synthesis techniques extended formal composition the fine structure of sound, our developments allow the compositional control of sound structures with complex spatio-timbral morphologies. In Chapter 3 we discuss related problematics in the field of gesture control. Although the importance of physical interaction for creative activities is well-known, current research prioritizes performance over composition and thus composers can rarely include physical gestures into their compositional models. While existing

developments mostly focus on gestures from an instrumental perspective, the context of composition presents a different paradigm and requires finding original solutions for coding, representation and mapping. We describe background and motivations, review related works and discuss requirements and design principles guiding our development efforts. We present the library OM-Geste and validate our concepts via two examples for synthesis of direct music representations (symbolic score and sound synthesis) from real-world gesture recordings of a dance and instrumental performance. Chapter 4 introduces a dictionary-based sound model for computer-aided composition. This development can be situated within the field of recent sound analysis/synthesis approaches for approximating sound targets via databases of sound files. We present the library OM-Pursuit which provides a complementary counterpart to the currently available sinusoidal models and allows for a wide range of applications (granular synthesis, computer-aided orchestration, audio transcription). An artistic application of a composition for computer-controlled piano and electronics is presented, which was made possible through the integration of this sound model with computer-aided composition tools.

In chapter 5 we present our conclusions. Contributions, impact, and limitations of our works are discussed, and future directions are described. Finally, we provide two appendices:

1. A list of notable artistic works realized with our software framework,

2. A list of tools and functions implemented for the respective libraries.

Chapter 2

Spatial Sound Synthesis in Computer-Aided Composition

The following chapter is based on the peer-reviewed journal publication:

Schumacher, M., Bresson, J. (2010). Spatial Sound Synthesis in Computer-Aided Composition. Organised Sound, 15 (03), pp. 271–289. Cambridge University Press (UK).

Abstract

In this article we describe our ongoing research and development efforts towards integrating the control of sound spatialisation in computer-aided composition. Most commonly, the process of sound spatialisation is separated from the world of symbolic computation. We propose a model in which spatial sound rendering is regarded as a subset of sound synthesis, and spatial parameters are treated as abstract musical materials within a global compositional framework. The library OMPrisma is presented, which implements a generic system for the control of spatial sound synthesis in the computer-aided composition environment OpenMusic.

2.1 Introduction

The digital revolution of music and media technologies in the early 1990s has stimulated an immense growth in the field of sound spatialisation. With many of todays computer music tools it is possible to render spatial sound scenes for many channels of audio and large numbers of sound sources. Many research centres and performance venues have installed large-scale multichannel systems, offering promising new possibilities for sound spatialisation applications, which require corresponding efforts in the fields of authoring and musical control. From a compositional point of view, we speak of 'sound spatialisation' as soon as the positions of sound sources, the ambience of a room, or any other spatial or acoustic element is taken into account as a musical parameter of a work. While space has probably always played an important role in music composition, the formalisation of space as a structural parameter is a rather recent phenomenon [105]. Stockhausen [205] stated that spatial configurations are as meaningful as intervals in melody or harmony, and that the consideration of spatial parameters is an integral part of the compositional process. Indeed, even prior to the advent of sound spatialisation technologies as commonly understood today, avant-garde composers in the 1950s had already begun to integrate space as a musical dimension into their pioneering electroacoustic works, taking advantage of the emerging technologies at hand, such as microphones, analogue mixing desks and loudspeakers (e.g. Karlheinz Stockhausen in *Kontakte* or *Gesang der Jünglinge*, Pierre Schaeffer in *Symphonie pour un homme seul*, or Edgar Varèse with *Poème électronique*).

Now that digital signal processing and musical acoustics are mature and well-established research fields, spatial sound scenes can be realised with a variety of rendering techniques, software tools and hardware setups. The literature reveals a broad spectrum of approaches and implementations for spatial sound rendering: perceptually informed amplitude panning techniques such as Vector Base Amplitude Panning (VBAP) [178] or Distance Based Amplitude Panning (DBAP) [132], holophonic techniques aiming at the physical reconstruction of a soundfield, such as Wave Field Synthesis (WFS) [21] or Higher-Order Ambisonics (HOA) [68], binaural/transaural techniques, and finally hybrid techniques, such as Space Unit Generator (SUG) [152] or Virtual Microphone Control (ViMiC) [30].[1]

[1]The SpatBASE project proposes an interesting and fairly documented reference of existing spatial sound rendering concepts and implementations: http://redmine.spatdif.org/wiki/spatdif/SpatBASE

Each approach, however, relies on specific assumptions about the nature of sound sources, listener and environment, and as a consequence might not be equally well-suited for different musical applications. Considering that works are often performed in multiple venues with different acoustic properties and loudspeaker arrangements, scalability and adaptability of spatialisation systems are also of major importance. To accommodate different scenarios, contexts and configurations, these systems should allow users to conceive spatialisation processes from a more abstract level. While much recent research focuses on strategies for real-time control (see for instance [143]) or the development of interchange formats [167, 118], there have been few attempts to integrate the control of spatialisation into compositional environments. In fact, sound spatialisation is often treated as a post-production technique which is unconnected to the processes dealt with in computer-aided composition, and therefore remains isolated in the corresponding compositional models and applications.

In this paper we present recent works aimed at integrating spatialisation in the computer-aided composition environment OpenMusic [4, 18]. After a brief discussion of related works (Section 2.2), we introduce a generic framework for sound synthesis and spatialisation, embedded in this environment (Section 2.3). The OMPrisma library is described as a structured system where spatialisation processes can be carried out and controlled in a flexible way, in relation to the symbolic compositional models and integrated with sound synthesis processes (Section 2.4). We present a powerful extension to the sound synthesis and spatialisation frameworks, allowing these two processes to be merged into hybrid structures implementing the concept of spatial sound synthesis (Section 2.5), and conclude with a number of example applications (Section 2.6).

2.2 Related Works

Among the most popular tools used for the compositional control of spatial sound scenes are those commonly referred to as “digital audio workstations” (DAWs). These environments are typically based on the metaphor of a multitrack tape-recorder and allow for automation and non-linear (mostly manual) editing of control parameters separated into a number of tracks. The user, however, has only limited access to the control data, and as the number of sound sources and parameters increases it becomes cumbersome to monitor and manage the complexity of the spatial sound scene. Moreover, it is difficult to link

the concrete representations (soundfiles, automation data) to more abstract compositional concepts, as this type of interface does not represent logical relationships.[2] Real-time audio processing environments, such as Max [176], PureData [177] or SuperCollider [145] provide frameworks in which control interfaces and rendering algorithms for sound spatialisation can be developed and integrated with more general sound synthesis and/or interactive processes (see for instance [190]). The IRCAM Spatialisateur [117] provides graphical user interfaces in Max (SpatViewer/SpatOper) which allow the control of numerous low-level parameters via a reduced number of perceptual descriptors such as "liveness", "presence", and the like.

Zirkonium [180] and Beastmulch [232] are examples of large-scale spatialisation systems based on the model of "live diffusion" which allow for the grouping together of sound sources and for these groups to be controlled individually. Several research projects focus specifically on higher-level control, abstracting the spatial sound scene description from the rendering techniques (see for example [97]). The Holo-Edit interface in the Holophon project [49] is an application allowing the high-level control of spatial parameters (trajectories). Conceived as an authoring tool for sound spatialisation, Holo-Edit provides complementary interfaces for viewing/editing of spatial parameters, including a top-view editor, a set of timeline controls, and 3D visualisation. Moreover, it provides a set of tools for algorithmic generation and modification of spatial trajectories [174], which is a significant step towards compositional control. Earlier projects, such as MoveInSpace [213] also provided advanced features, such as a trajectory generator, room and loudspeaker settings, and correlation of the spatialisation parameters to sound morphological features (some characteristics which will be found in different parts of our work), implemented as an independent control layer on the Ircam Musical Workstation [131]. A different approach for authoring of spatial sound scenes is taken in the MidiSpace [163] and MusicSpace [165, 71]) systems, which provide graphical interfaces allowing the design of spatial sound scenes including MIDI instruments and audio sources. Most notably, these applications include powerful constraint setting and propagation systems allowing the definition of spatial relations between the different sound sources in the scene.

Most control systems, however, focus on a specific model of sound spatialisation (such as surround-mixing, sound-diffusion, etc.). Although we noted the algorithmic functionalities

[2] An overview of DAWs in terms of surround features can be found at: http://acousmodules.free.fr/hosts.htm

(in Holo-Edit or MoveInSpace) and tools for constraint-setting and propagation (MusicSpace), these features require integration with higher-level programmable environments in order to enable more abstract representations and accommodate different compositional applications. As stated in [16], efficient compositional environments should be conceptually close to specialised programming environments: in such compositional contexts, high-level, symbolic tools and processes allow abstracting control data and processes to a set of manageable and musically meaningful representations, while remaining open enough to be used in different contexts by different composers.

OpenSpace [72] was an original attempt at integrating the MusicSpace control system in the computer-aided composition environment OpenMusic. Visual programs allowed defining a spatial setup for MusicSpace sound sources and incrementally adding constraints, while the maquette interface was used to control the unfolding of this process in time. Another project carried out in OpenMusic is OMSpat [156], a library for the control of the Spatialisateur. In OMSpat an array of sound sources, trajectories and room parameters could be created from algorithmically (or manually) defined curves and parameters. This array was then formatted as a parameter file for a specific Spatialisateur control application that could reproduce the spatial sound scene using two, four, or eight speakers, or via binaural rendering. Although the temporal resolution of the control-data and the number of simultaneous sound sources were limited, the ability to script trajectories and spatial parameters allowed the user to establish structural relationships between spatialisation and other symbolic data and processes defined in the computer-aided composition environment. This project has recently been generalised and extended, introducing new 3D-trajectory objects and tools for formatting output for external environments [43]. Some similarities can also be found in the works we present in this paper, which inherit much of the control paradigms and structures from the same type of objects (matrices: see Section 2.3).

As discussed below, we approach the control of sound spatialisation by considering spatial parameters as additional parameters in a sound synthesis framework, whether they relate to micro-level sound synthesis components (such as partials or grains) or to pre-existing sound sources. This approach, embedded in a high-level control environment, allows us to extend the common model of sound source spatialisation to the more general concept of spatial sound synthesis, and to generalise some of the techniques for time-domain or frequency-domain spatial distributions, presented for instance in [214, 121], within a symbolic and programmable compositional context.

2.3 A Generic Framework for the Control of Sound Spatialization

2.3.1 The computer-aided composition environment: OpenMusic

OpenMusic (OM) is a visual programming language for music composition based on Common Lisp/CLOS [91]. This environment allows the graphical design of programs by patching together functional components, and provides high-level musical interfaces such as scores and other graphical editors. It has been used to develop numerous musical works, constituting a powerful and efficient framework for the creation of complex musical structures related to various compositional approaches [6, 40]. Additional development in OpenMusic has involved the integration of sound processing, analysis and synthesis tools, and led to a renewed conception of sound representations in the framework of computer-aided compositional models [37]. Integrating the control of sound spatialisation into the conceptual framework of a computer-aided composition environment introduces new possibilities: spatialisation parameters, as any other musical data, can be devised and determined using algorithms and programming interfaces, hence in close relation with associated processes. OpenMusic provides a number of geometrical objects such as breakpoint- and 3D-curves (*BPC*/*3DC*) representing abstract spatial configurations defined as sequences of points. Temporal information can be explicitly specified (which turns curves into trajectories), or kept implicit and interpreted according to a given context. These objects can be generated and transformed by algorithmic processes in the programming environment or visualised and edited manually using graphical editors. Figure 2.1 shows an example for the algorithmic generation of 3D curves by visual programs.

2.3.2 Sound synthesis and spatialisation: OMChroma/OMPrisma

OMChroma [9] is a compositional framework for the control of sound synthesis in OpenMusic, based on Marco Stroppa's Chroma system [207]. This framework provides a set of classes (in terms of object-oriented programming) referring to underlying sound synthesis processes. Each class is associated with a digital signal processing (DSP) patch, currently in the form of a Csound instrument [28]. The parameters of these instruments (called *p-fields* in Csound) are detected and matched to corresponding slots of the class, which can be instantiated in OpenMusic's visual programs. Accordingly, the graphical

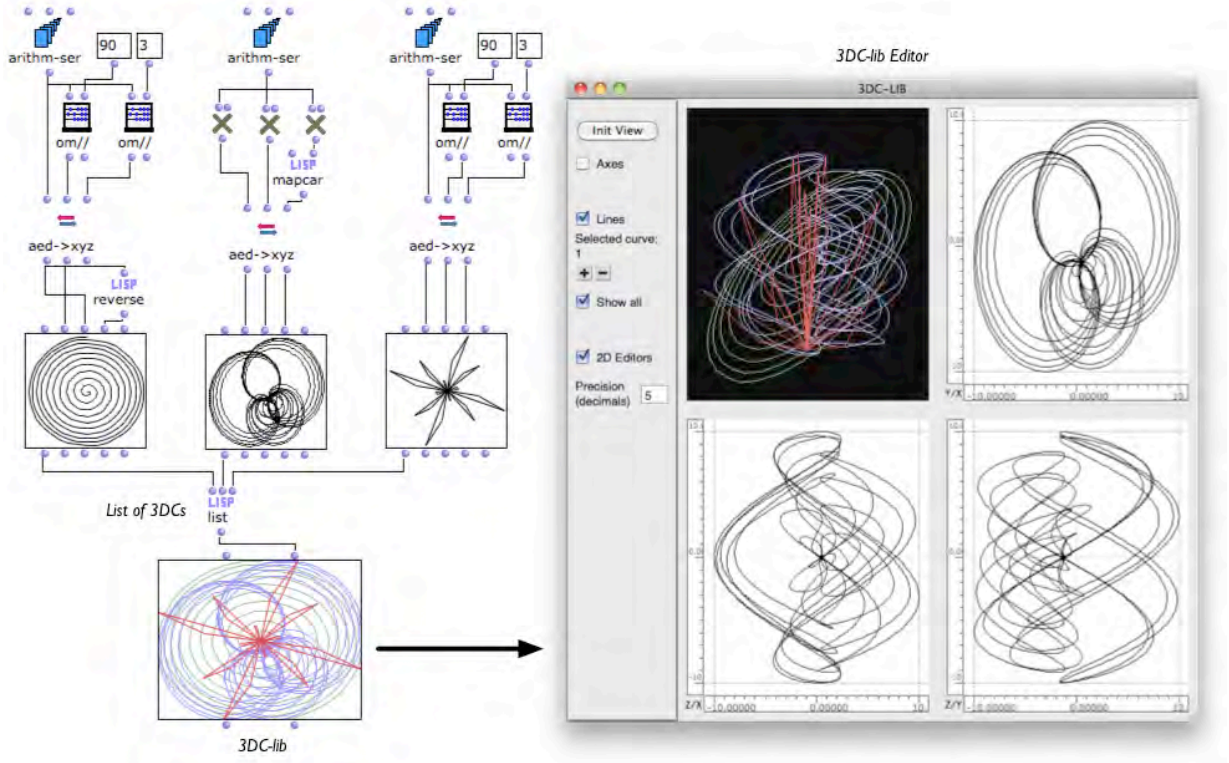

Fig. 2.1: Generation of 3D curves via visual programs in OM. The *3DC-lib* box is a set of *3DC* objects. The data can be visualised and edited in graphical editors.

representation of an OMChroma class (called a *box*) has a number of inlets corresponding to the underlying sound synthesis parameters in the Csound instrument. OMChroma includes a large library of classes, ranging from basic (e.g. additive, granular, FM, etc.) to more complex sound synthesis algorithms. This library is user- extensible, and new classes can easily be defined from existing Csound instruments.

OMChroma classes are matrix structures, instantiated with a given number of "components" (represented as columns). Each row corresponds to a slot of the class (i.e. to the related synthesis parameter in the Csound instrument). A matrix can include arbitrary numbers of components, describing vectors of parameter-values for the underlying synthesis instrument, which can be controlled via high-level and symbolic means and subjected to compositional processes. When the matrix is "synthesised" (i.e. rendered into an audio file) a Csound score is gene-rated from the 2D data structure: each component in the matrix (a column with a value for each synthesis parameter) corresponds to an event in the score (see [207] for a detailed discussion).

OMPrisma is a library providing a set of matrix classes corresponding to spatial sound rendering instruments (see Section 4). The OMPrisma classes extend the OMChroma matrix, and therefore benefit from the same expressive power and control structures used in the control of sound synthesis processes. The computed matrix contents depend on the type of the supplied data and on the number of components: a single value, for instance, means that all components have the same value for a given parameter; lists of values are taken literally or repeated cyclically until the number of elements matches the number of components; breakpoint-functions are sampled over the number of components; and mathematical/functional expressions (defined as Lisp functions or visual programs) are evaluated individually for each component. Once instantiated, the contents of a matrix can be visualised and edited manually as a 2D array using a graphical editor.

Figure 2.2 shows a basic sound spatialisation process carried out in OMPrisma. A set of monaural sound- files is spatialised and rendered into a multichannel file for quadraphonic reproduction using the class *pan.quad.discrete* from the OMPrisma class-library. The body of the instrument in the orchestra file of Figure 2.2 (from *instr1* to *endin*) is copied from the *pan.quad.discrete* class. The *synthesize* function formats the values of the components in the matrix into Csound score statements (i.e. turning the columns into rows). Most values here are numbers (except the file names used for *p4*, which are derived from the soundfiles in the OM patch). When continuously changing values are required, for example

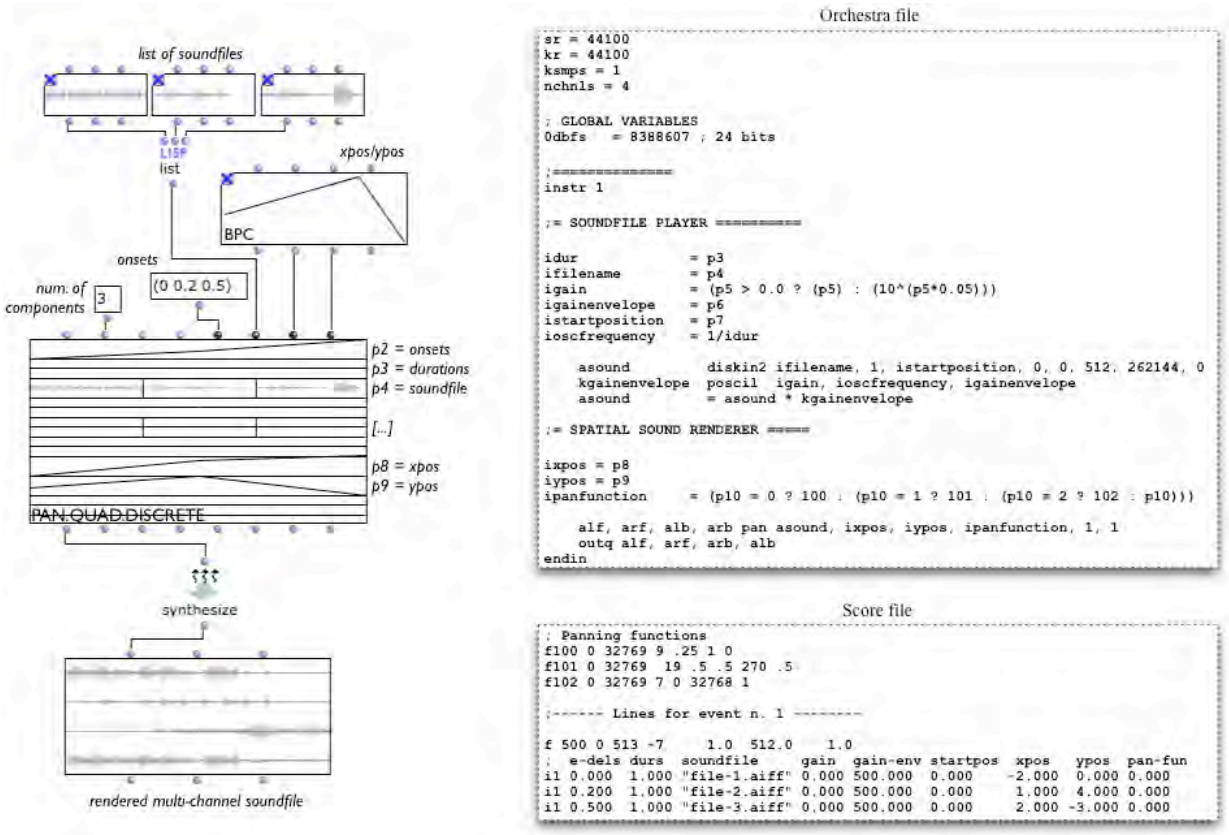

Fig. 2.2: Sound source spatialisation with OMPrisma. Left: the patch in OpenMusic. Right: the generated Csound orchestra and score files.

for describing transitions or envelopes, breakpoint-function objects can be used, which are internally converted into Csound tables.

Note that not all parameters (*p-fields* in the Csound orchestra) are explicitly specified in the OM patch. The matrix boxes allow the user to selectively display or hide the different slot inlets of a class, whereby unspecified (i.e. hidden) slots are set to default values. In figure 2.2, only the slots *onsets*, *soundfile*, *xpos* and *ypos*, corresponding to *p2*, *p4*, *p8* and *p9*, respectively, are specified to control the spatialisation process. The default value for the slot *gain-envelope*, for example, is 500 (a Csound table identifier), which is set for *p6* as no value is specified in the OM patch. Similarly, the three "panning function tables" (visible at the top of the score file in figure 2.2) are defined inside the class *pan.quad.discrete*, and function as presets, which can be referred to via numerical indices in the csound score. This way irrelevant or redundant information is hidden from the user, making for a more ergonomic and context-sensitive interface.

As in the case of sound synthesis processes, the dynamic instantiation of multiple Csound instruments (corresponding to the components of a matrix) yields a virtually unlimited polyphony for the spatial sound rendering process. In this perspective, a matrix can be regarded as a generic data structure for the description of spatial sound scenes with arbitrary numbers of sound sources, possibly controlled independently, using common rules, control data or algorithms.

It is also possible to devise a synthesis process using multiple matrices (i.e. synthesising a list of matrices instead of a single one). If the matrices cor- respond to different classes, the respective instruments are gathered in a single orchestra file and identified by instrument number (*instr1*, *instr2*, ...). Each matrix can also be assigned a global onset-time, allowing it to be considered as a temporal "event" in a larger-scale time structure.

2.4 OMPrisma

OMPrisma is implemented as an extensible framework comprising a library of classes for spatial sound rendering (Section 2.4.1), a library of tools and functions for generation and manipulation of spatialisation parameters (Section 2.4.2), and an external standalone application (titled Multiplayer) for decoding and diffusion of the rendered multichannel audio formats (Section 2.4.3). Several studies have documented a great variety of compositional approaches for sound spatialisation (see for example [105]), and it is unlikely

that a specific spatial sound rendering technique will satisfy every artist's needs. A more sensible solution is to provide a programmable abstraction layer which separates the spatial sound scene description from its rendering, and leave it to the user which spatial sound rendering approach is most suitable for a given purpose. The OMPrisma class-library provides a palette of spatial sound rendering instruments, implementing different spatial sound rendering techniques. Currently available are classes for stereo, quadraphonic and 5.0 (ITU) panning, VBAP, RVBAP, DBAP, HOA, and a mixed-order Ambisonics system with optional simulation of room-acoustics. Figure 2.3 gives an overview of implemented spatial sound rendering concepts and respective classes.[3]

Amplitude Panning	*VBAP*	*RVBAP*	*DBAP*	*Ambisonics*	*SPAT*
Pan.stereo.discrete Pan.stereo.continuous	Vbap.2D.discrete Vbap.2D.continuous	Rvbap.2D.discrete Rvbap.2D.continuous	Dbap.2D.discrete Dbap.2D.continuous	Ambi.2D.discrete Ambi.2D.continuous	Spat.2D.discrete Spat.2D.contrinuous
Pan.quad.discrete Pan.quad.continuous	Vbap.3D.discrete Vbap.3D.continuos	Rvbap.3D.discrete Rvbap.3D.continuous	Dbap.2D.discrete Dbap.3D.continuous	Ambi.3D.discrete Ambi.3D.continuous	Spat.3D.discrete Spat.3D.continuous
Pan.5.0.discrete Pan.5.0.continuous				Ambi.UHJ.discrete Ambi.UHJ.continuous	

Fig. 2.3: Spatial sound rendering concepts and classes in OMPrisma.

Dynamically changing values, such as envelopes or trajectories (i.e. "spatial glissandi") can be described both in terms of successions of discrete, "granular" positions or as a single continuous movement (consider for example the notion of a glissando on a piano vs. a fretless string instrument). In [106] the author discusses the difference between discrete or stepwise-proceeding spatial movements (dating back to the Venetian school of polychorality in the late renaissance), and continuous motion (introduced in instrumental and electronic music of the post-war avantgarde). We adopted this notion in that every OMPrisma class is available in a dedicated version for discrete and continuous control, respectively.

The separation of the spatial sound scene description from its rendering and reproduction offers many advantages [169]. For example, it allows the user to rapidly exchange a given spatial sound rendering with another one without affecting the other components. It further facilitates modifications or extensions at the renderer level (i.e. Csound instruments), since the DSP implementation can be modified independently as

[3]Note, that since the writing of this paper the library of spatialization classes has been significantly extended. A table with current spatialization techniques and their properties is given in Appendix B.

long as it provides the same interface to the environment. Moreover, the use of an external real-time application for decoding and diffusion (the Multiplayer) will provide the flexibility of adapting the reproduction of a spatial sound scene according to a given environment. Figure 2.4 shows an example of 3 OMPrisma classes rendering the same spatial sound scene using different techniques.

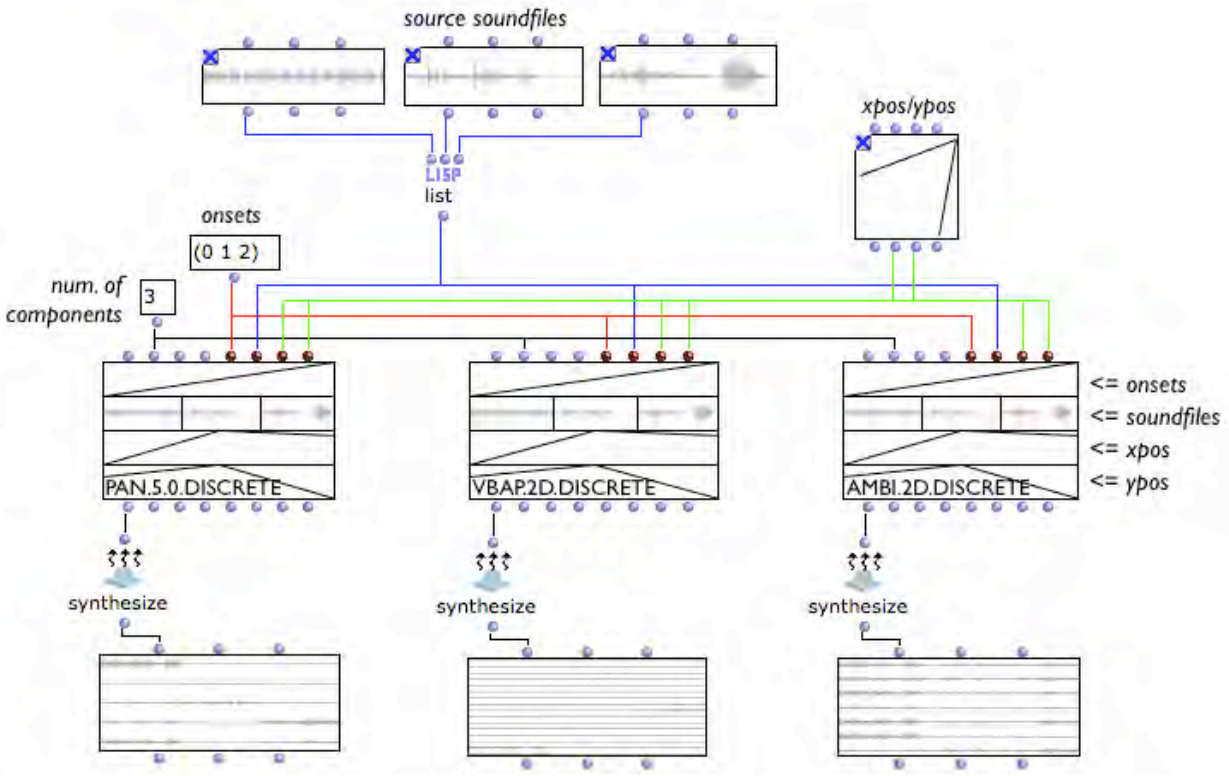

Fig. 2.4: The same spatial sound scene description realised with different spatial sound rendering techniques: 5.0 (ITU) panning, VBAP and higher-order Ambisonics.

2.4.1 Spatial sound rendering

OMPrisma employs the Csound language as spatial sound rendering engine, which allows for sample-synchronous control of all parameters, high-resolution processing and unlimited polyphony. Note that the same matrix control-structures may as well be used and formatted for another synthesis engine, or written into external interchange format files, see for example [207, 43]. In order to easily maintain, modify and extend the collection of spatial sound rendering instruments, they have been implemented following a modular design.

Common functionality is encapsulated into modules (*user-defined-opcodes*, [128]) and re-used across the different instruments, such as the soundfile-player, or source pre-processing modules. In the following section we will discuss some of the implementation-specific details.

Dynamic instrument configuration

Many spatialisation algorithms are capable of driving various loudspeaker configurations and numbers of output channels. The OMChroma system allows for the writing of global statements into Csound orchestra files before the instrument definition, which permits dynamically changing the output configuration without the need of modifying the original instrument's body. Accordingly, a single OMPrisma class (implementing a specific spatial sound rendering technique) can be used for various loudspeaker setups and output channels.

Source pre-processing

For classes implementing intensity-based panning techniques we have developed source pre-processing modules for rendering of perceptual cues to support the impression of distance and motion of a sound source. The effect of air absorption is simulated with a second order Butterworth lowpass filter with variable cut-off frequency. An attenuation-module accounts for the decrease of a sound source's amplitude as a function of distance. Doppler shifts are simulated with a moving write-head delay-line with high quality interpolation. Rather than hard-coding the equations for rendering of perceptual distance-cues into the spatialisation engines directly, we implemented a table-lookup system for greater efficiency and flexibility. Lookup tables can be generated using pre-defined (see Section 2.4.2) or user-defined functions, and manually edited using OpenMusic's graphical breakpoint-function editors. These tables can then be connected to the corresponding slot of a class to be applied to a specific sound source (see figure 2.4.2), or provided as global tables for the whole spatial sound rendering process.

Room effects

Room acoustics and reverberation are important perceptual cues for the localisation of a sound source, and provide information on a surrounding environment's size, shape and material [25]. The description of room effects is conceptually different from the description

of sound sources and therefore requires alternative control-strategies. Depending on the underlying model and implementation of a rever- beration algorithm the control interfaces can vary to a great extent (for example perceptual vs. physical models) and often require the setting of many individual parameters, which might clutter up the interface when specified as individual slots of a class. Thus, in OMPrisma room parameters are defined in tables, as a compact data-structure (provided to a single slot of a class), which can be edited graphically or algorithmi- cally directly in OM, and imported/exported as files to disk. Currently, two spatial sound rendering classes in OMPrisma include reverberation: RVBAP and SPAT. The reverberation algorithm in RVBAP is implemented as a feedback delay network based on digital waveguides, while SPAT implements a shoebox room-model based on recursive rendering of discrete reflections. Note that due to Csounds dynamic instantiation paradigm the full range of parameters of the spatial sound rendering engine is available for each individual sound source. As with any other matrix slot, room parameters can be set globally for the whole synthesis process or controlled individually for each component.

Within the loudspeaker array

The placement of virtual sound sources within the surrounding loudspeaker array is a feature often desired by composers, which is difficult or even impossible to realise with many spatial sound rendering techniques. A number of works have addressed this issue [148, 69], however these solutions are mostly idiosyncratic to a given spatial sound rendering concept and can be difficult to control for a non-expert user and without adequate technical equipment. In order to have a consistent approach for different spatial rendering classes we implemented the popular technique of decreasing the directionality of a sound source as it enters the speaker array towards the listener, approaching complete monophony (i.e. all speakers contributing equally) at its centre. For classes implementing VBAP, for example this is accom- plished through implicit control of the *spread* parameter [179]; in the case of Ambisonics, via controlled decrease of gain coefficients for higher-order components (as described in [190]). This behaviour is optional and can be tweaked or bypassed.

2.4.2 Control strategies

OMPrisma is designed to provide a higher-level abstraction layer for spatial sound scene description which is independent of the underlying rendering implementation. Accordingly, all classes share a structured interface complying with current specifications of the Spatial Sound Description Interchange Format (SpatDIF, [167]). Control parameters (i.e. class slots) are organized into conceptual groups (or namespaces), such as soundfile-player parameters, position data, renderer-specific parameters (such as the "spread" parameter for VBAP), source pre-processing settings and reverberation parameters. An overview of OMPrisma classes with respective slots is given in the Appendix. Figure shows an example of a complete spatialisation process including conversion of a *3DC-lib* into individual trajectories for position control, symbolic setting of room-parameters and rendering of perceptual distance cues. Global settings for the rendering process are provided directly to the synthesize method, independently of the spatial sound scene description.

Trajectories

Trajectories for position-control of sound sources can be defined via geometric objects, such as breakpoint-functions (*BPFs*), 2D breakpoint-curves (*BPCs*), and 3D-curves (*3DCs*, see Figure 2.1). The function *gen-trajectory* unfolds these geometric objects in time and returns the corresponding control data (envelopes for each Cartesian dimension) using two complementary strategies: In "constant speed" mode, the sound source will travel along the trajectory with constant speed, while in "constant time" mode it will respect a constant time interval between successive points in the trajec- tory. As an additional feature, the *gen-trajectory* function allows the generation of B-Spline curves; that is, polynomial interpolations between the object's initial control points. This way, a trajectory can for example be specified manually with a few breakpoints, and its curvature controlled using this function. Obviously, trajectories can be set and modified via pre-defined or user-defined algorithms. Alternatively, the new object *3D-trajectory* was implemented, which allows the assignment of time-tags to spatial points in the trajectory, either explicitly or automatically (deduced from surrounding points).

After the spatio-temporal morphology of a trajectory has been defined its absolute playback speed can be controlled via frequency envelopes (individually for each Cartesian dimension). If no frequency envelopes are specified, the speed of a trajectory is implicitly

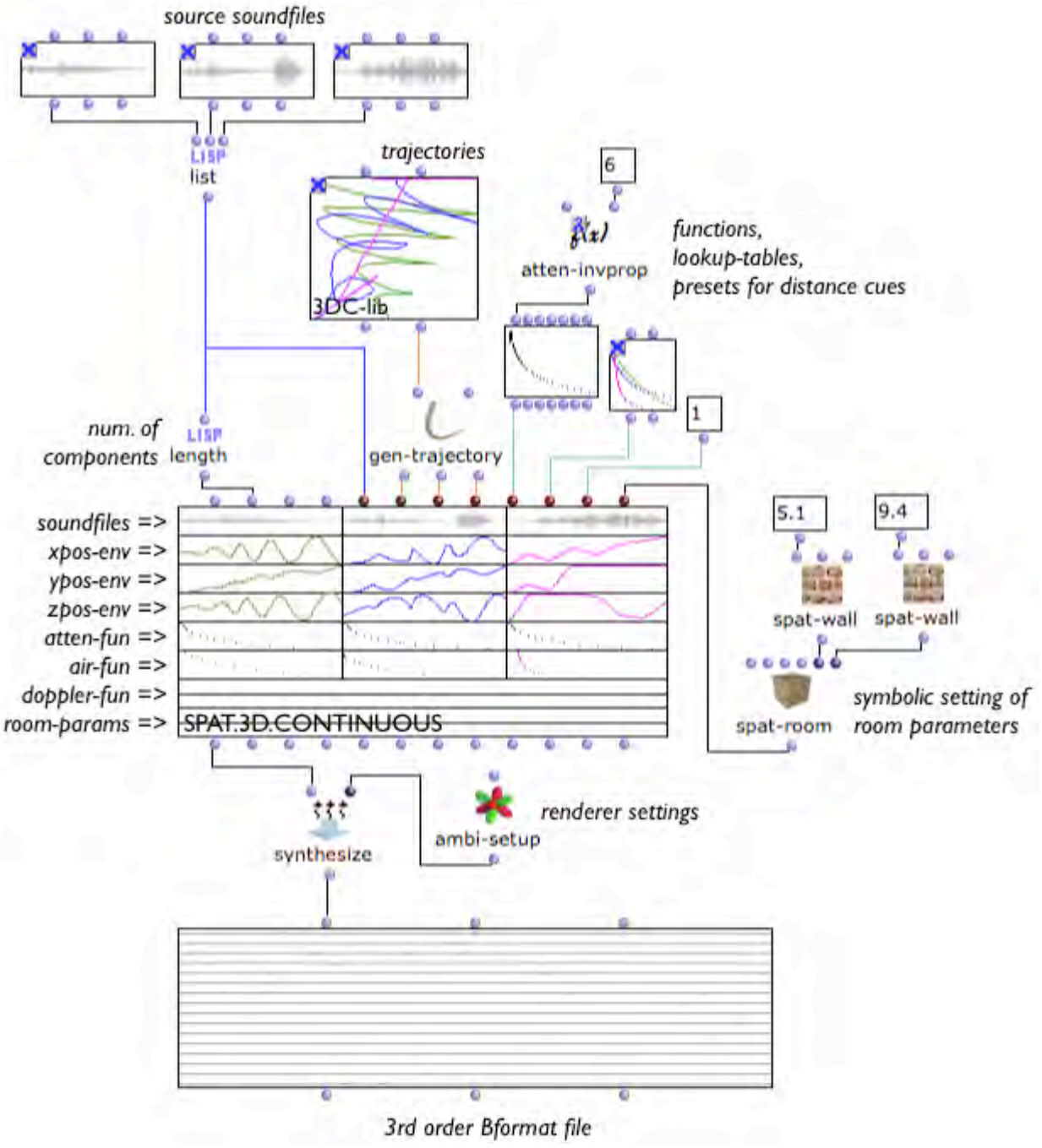

Fig. 2.5: Example for a sound spatialisation process using the OMPrisma class *spat.3D.continuous*. The *gen-trajectory* function converts a *3DC-lib* object containing 3-dimensional trajectories into envelopes for x, y, z. Functions, pre-defined lookup-tables and presets are used to control the rendering of perceptual distance cues. Room-characteristics are specified via the function *spat-room*. The function *ambi-setup* is used to set global parameters for the rendering process.

scaled to fit the duration of its corresponding synthesis event (e.g. the duration of the soundfile). The use of frequency envelopes allows for dynamic control of the speed-of-travel of a sound source (including stopping or reversing the travel direction), creating spatial patterns (e.g. spirolaterals and lissajous figures), or working with audio-rate oscillations and frequency modulations at the border between sound synthesis and spatialisation. As with any matrix parameter, trajectories can be set globally or specified independently for each component (i.e. sound source).

Function library

OMPrisma features a "compositional toolbox" of functions and utilities for generation and manipulation of spatialisation data. The function library includes tools for processing of 2D or 3D breakpoint curves, such as interpolations in Cartesian or spherical coordinates, geometric transformations (e.g. rotations, scaling, mirroring), and stochastically-driven manipulations such as perturbations and clusterings. For rendering of perceptual distance-cues a set of predefined functions are provided which implement commonly used equations to control the simulation of attenuation, air-absorption and Doppler shifts as functions of distance. Yet another category are renderer-specific functions, used for example to set reverberation/room-parameters. The *spat-room* function shown in figure 2.5, for example, allows the setting of physical properties of a shoebox room-model in a symbolic way by connecting functions describing characteristics of individual walls (*spat-wall*) to a global room-model (*spat-room*). Finally, various utilities are provided, for example for configuration of loudspeaker setups, or to perform conversions between coordinate systems. Since these tools are embedded in a programming environment, they can be easily adapted, extended and related to the extensive set of libraries and features of OpenMusic.

Algorithmic sound scene manipulation

A particularly powerful concept inherited from the OMChroma system is the *user-fun*. This function, written directly in LISP or defined graphically in an OpenMusic patch, can access and modify the contents of a matrix and defines programs in order to manipulate, possibly generate or remove elements before starting the rendering process.

User-funs can take additional arguments (provided as inlets to the matrix), which allows the user to introduce new control-parameters in the spatialisation process. This paradigm

constitutes a powerful tool for selective and global manipulations of the matrix data, such as groupings, rotations/translations of sound source positions, or arbitrary rule-based transformations of entire spatial sound scenes. One possible application is the modelling of composite sound sources emitting sound over an extended space by breaking the original sound source up into a set of independent point sources. Figure 2.6 shows a graphically-defined user-fun implementing the concept of Sound Surface Panning as described in [180]: for each sound source in the matrix a two-dimensional shape is specified, which is synthesised as a mesh of evenly distributed point sources. This process is controlled using the same data as in figure (i.e. soundfiles, trajectories, etc.) and rendered into a multichannel soundfile using VBAP. Note that, thanks to the common control-interface for OMPrisma classes, the same user-fun and global processing can be applied to any other spatial sound rendering class. Similarly, we have employed the user-fun to implement *W-panning* (described in [195]).

2.4.3 Decoding and diffusion: the Multiplayer

For any spatial sound composition tool it is of great importance that the user be able to monitor the results immediately. A tight auditory feedback loop between composition and reproduction facilitates efficient workflows and allows tweaking and fine-tuning the spatialisation processes via perceptual evaluations. Another important aspect is the ability to adjust the reproduction in real-time in order to adapt to a given environment, such as a specific studio setup or concert venue. This might include tasks such as the routing of logical output channels to physical devices, the ad- justment of gains or time-delays for specific channels, or in the case of encoded formats setting and tweaking decoder parameters in real-time.

The *Multiplayer* is a standalone application for decoding and diffusion of interleaved multichannel soundfiles in different formats. It is implemented as a set of modules complying with the Jamoma framework for Max/MSP [171]. It is intended to facilitate the work on spatial sound compositions in changing environments and for different loudspeaker setups, without requiring any expert-knowledge from the user. Figure 2.7 shows a screenshot of the Multiplayer application.

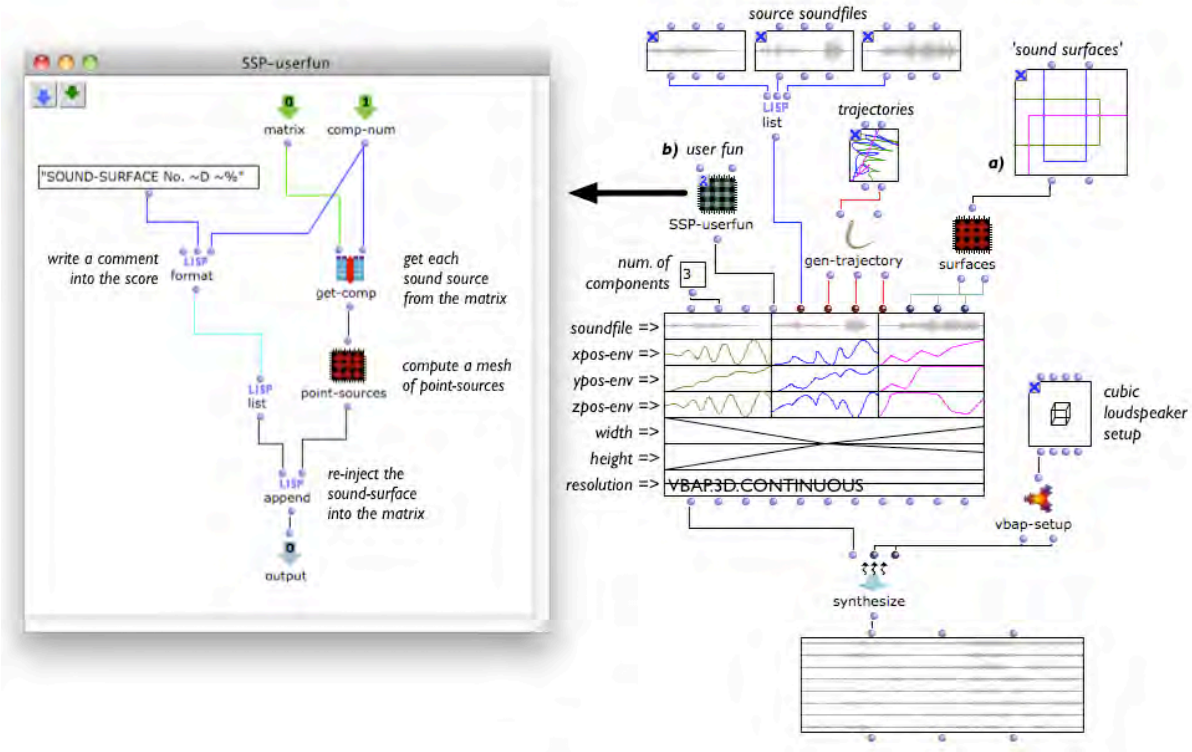

Fig. 2.6: Implementation of *Sound Surface Panning* via a user-fun applied to an OMPrisma matrix. a) Sound Surfaces are graphically defined via *BPC* objects and provided as additional parameters (width/height/resolution) to the matrix. The patch labelled SSP-userfun (b) is set as user-fun for the matrix, and therefore evaluated for each component in order to replace the sound sources with sound surfaces.

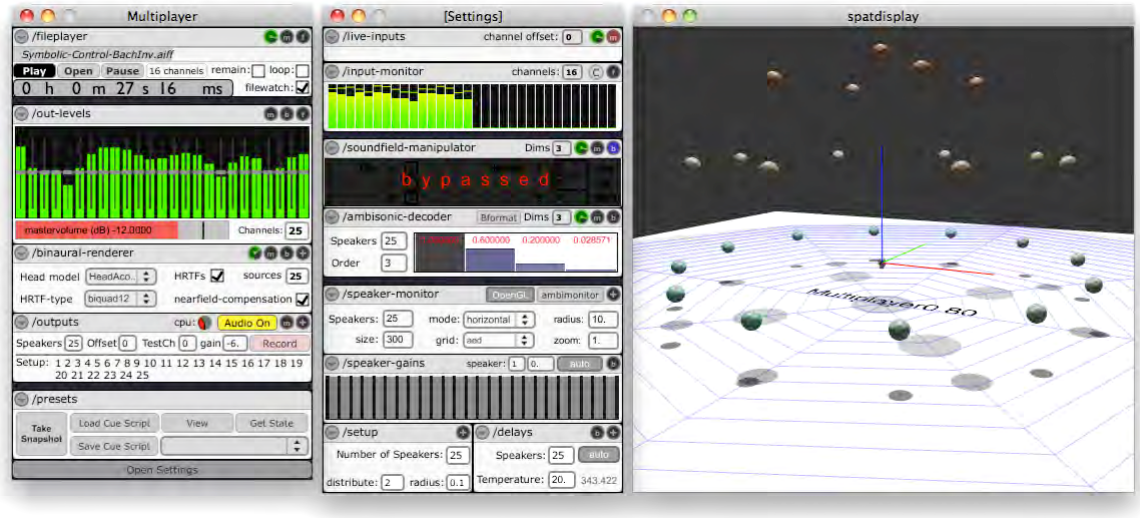

Fig. 2.7: The *Multiplayer* standalone application decoding a 3rd-order Bformat soundfile. On the right hand side is a 3D visualisation of a hemispherical loudspeaker setup.

Integration

The Multiplayer seamlessly integrates into the workflow with OMPrisma via bi-directional communication using Open Sound Control (OSC) [233]. Once the communication has been established, the Multiplayer can safely be sent to the background and entirely controlled from OM (using the same transport controls as OM's internal players). Optionally, the Multiplayer can be synchronised with rendering settings in OMPrisma, in the sense that it will automatically update its configuration accordingly, hence no switching between applications is required.

Adaptability

Other important aspects for a decoding or diffusion application are compatibility with different formats and adaptability to different reproduction environments. The Multiplayer dynamically re-configures its internal dsp structure via scripting (i.e. adds/removes channels) to match a given reproduction situation. Modules are provided for soundfield manipulations (rotations and mirroring along the principal axes), Ambisonics decoding, numerical and graphical interfaces for configuration of loudspeaker setups, and binaural rendering. Sound pressure- and time-differences in non-equidistant loudspeaker setups can be either automatically compensated (via loudspeaker positions) or manually balanced.

Auralisation

Composers are often required to elaborate musical works using loudspeaker configurations which are different from the intended reproduction setup in the performance venue. To address this issue, the Multiplayer provides a binaural rendering module for virtual loudspeaker binauralisation; that is, simulating a certain loudspeaker setup by treating the loudspeakers as virtual sound sources. Another benefit of this feature is the possibility of auditioning spatial sound scenes for various loudspeaker configurations (e.g. experimenting with irregular setups) and from different listening positions. It also allows the work with OMPrisma in the complete absence of loudspeakers. Moreover, it can be employed for rendering of binaural mixdowns. Since all parameters are accessible via OSC, external devices (such as head-trackers) can be employed for interactive control of the binaural rendering.

2.5 From Sound Source Spatialization to Spatial Sound Synthesis

The process of sound spatialisation can be carried out on multiple time-scales [185]. While traditional diffusion practices are usually based on direct manipulations that can be performed in real-time, there is no such restriction using digital signal processing techniques. Much as the development of analogue studio techniques (and, later, the digital syn- thesiser) made it possible to manipulate sound below the level of individual sound objects in the domain of sound synthesis, spatialisation processes can be applied to the microstructure of a sound in order to synthesise sounds with complex spatial morphologies.

2.5.1 Spatial sound synthesis

Within the presented framework we consider the term spatial sound synthesis most appropriate to denote the extension of sound synthesis algorithms into the spatial domain, that is as a general term to express spatialisation processes at the micro-level of sound. Several systems have been described which allow for spatial sound synthesis applications: Torchia and Lippe [215] presented a system for real-time control of spectral diffusion effects which allows to filter a sound into its spectral components and spatialise them individually. *Scatter* [147] is another original system for granular synthesis, which allows spatial positioning of individual grains using dictionary-based methods. The *Spatial Swarm*

Granulator [232] allows the control of spatial positions of individual grains based on Reynold's Boids algorithm [182]. Kim-Boyle [121] gives an overview of frequency-domain spatial distributions, mostly controlled via particle and agent systems. Interestingly, the author stresses the need for "an interface with the power to control and transform coordinates for hundreds of particles which at the same time does not overwhelm the user with massive banks of control data". In their presentation of Spatio-Operational Spectral Synthesis (which is conceptually close to our notion of spatial sound synthesis), Topper et al. (2002) describe the process as "taking an existing synthesis algorithm and breaking it apart into logical components" and then "[assembling] the components by applying spatialisation algorithms".

The OMChroma system, which builds upon an initial implicit decomposition of the sound synthesis process into "components" (see section 2.3.2), lends itself particularly well to this idea of spatial sound synthesis. The separation into logical components is given in the initial design of the OMChroma classes and generalised by the use and control of matrix structures. The same paradigm is adopted for the individual spatialisation of each synthesis component.

2.5.2 Implementation with OMChroma/OMPrisma

Generalized spatial sound synthesis processes can be achieved in OMChroma by designing Csound instruments to perform both sound synthesis and spatial sound rendering. However, given the number of pos- sible combinations of existing synthesis classes (in OMChroma) and spatialisation classes (in OMPrisma), the explicit implementation of individual spatial sound synthesis instruments would lead to an excessive amount of classes. A more sensible solution is to combine sound synthesis instruments with spatial renderers dynamically, in order to create compound instruments capable of spatial sound synthesis. In terms of signal-flow this idea can be described as an automatic detection and redirection of the output of the synthesiser to the input of the spatial sound renderer (i.e. replacing the former input sound source). This is a non-trivial task, however, which is carried out in two steps:

1. A new Csound instrument is created by merging the original synthesis and spatial rendering instruments. In the synthesis instrument, the variable bound to the signal output must be identified, while, on the spatial rendering side, the original input and its internal dependencies must be detected and replaced by the synthesis output. Variable declarations, bindings and redundancies must be handled between both instrument-parsing processes. Useless parameters must be omitted (e.g. the "soundfile-player" part in the spatialisation instruments), and the Csound *p-fields* order and indexing must be adapted accordingly.

2. The merged Csound instrument code is then used to create a new hybrid class starting from the two initial ones (synthesis and spatial rendering). Fortunately, Common Lisp and CLOS provide powerful tools for meta-object programming and dynamic class definition [91]. After inspecting the different slots and properties of the respective classes, a new one is defined by keeping the common fields (e.g. *e-dels*, *durations*) and combining the specific ones of the different instruments. The resulting class is instantiated using the corresponding slot values in the two initial objects.

From a user's perspective this merging-process is accomplished by simply connecting two objects (any synthesis class from OMChroma and any spatial rendering class from OMPrisma) to the function *chroma-prisma*, which internally creates the new merged-orchestra class and outputs an initialised instance. The resulting instance can eventually be plugged into the synthesize function in order to perform the spatial sound synthesis. This process is illustrated in Figure 2.8, in which an additive synthesis class is merged with a quadraphonic spatial sound rendering class. This system therefore constitutes a powerful synthesis framework, enabling high-level control over the full range of sound synthesis and spatialisation parameters for each individual component – partials, grains, or any other primitive components of a given synthesis algorithm.

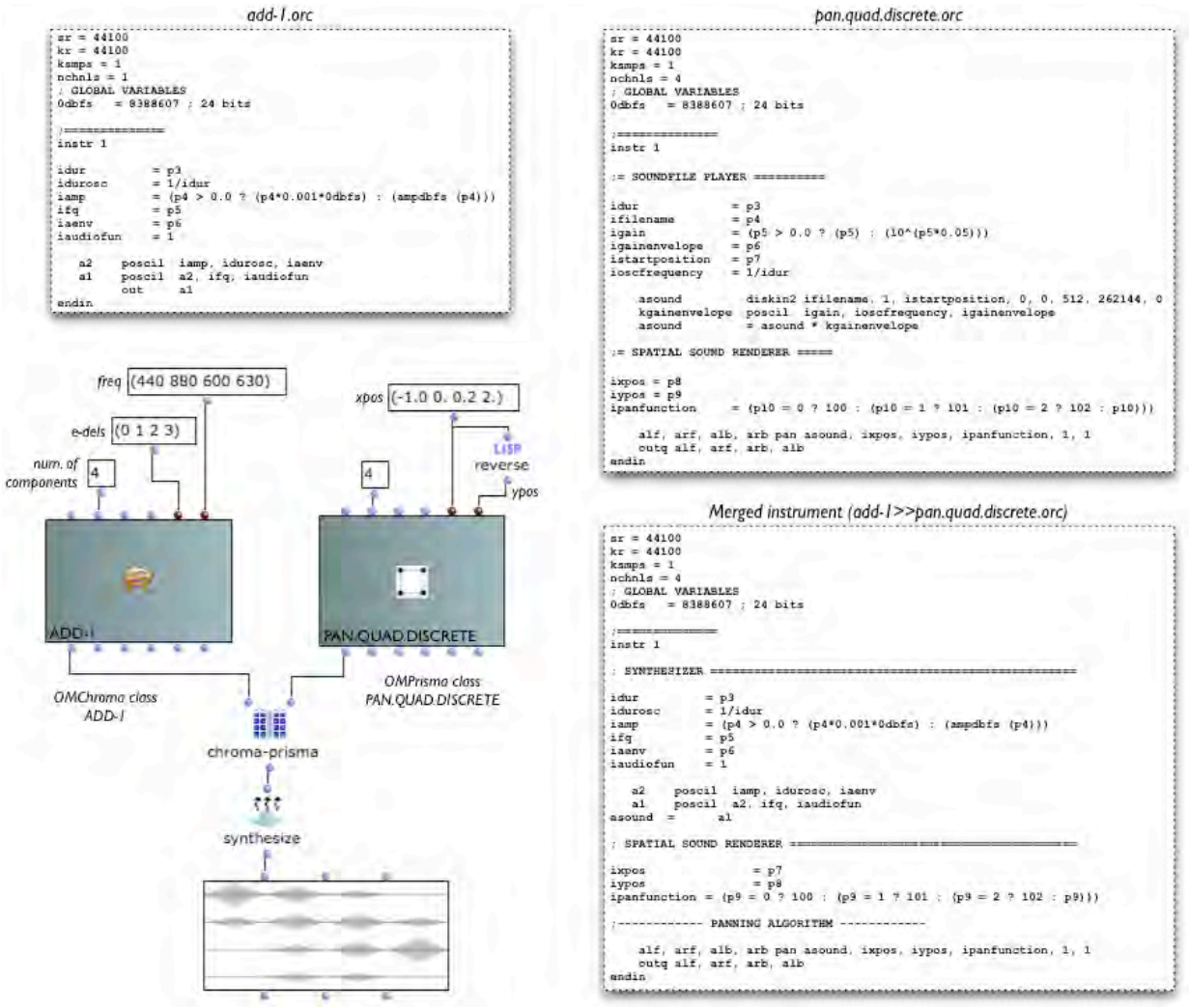

Fig. 2.8: Spatial sound synthesis: Merging synthesis and spatialisation classes in OMChroma. At the bottom right: the merged Csound instrument generated automatically from the class *add-1* (add-1.orc, top left) and the class *pan.quad.discrete* from figure 2.2 (pan.quad.discrete.orc, top right).

2.6 Example Applications

One of the main interests of the OMPrisma framework is its integration into a general compositional environment (OpenMusic). This embedding leads to novel possibilities for integrating symbolic, sound and spatialisation data into compositional frameworks and offers a degree of control difficult to achieve using conventional tools. In this section we show a number of examples of how spatial parameters can be related to other musical dimensions within a compositional framework.

In figure 2.9, symbolic musical material (a score in common music notation) is used as input data to control both sound synthesis and spatialisation parameters. The spatial positions for the synthesised sound components (plucked-string synthesis) are derived via arithmetic operations from the chords/notes contained in the score: For every chord the "virtual fundamental" is calculated; that is the highest fundamental frequency for which the notes of the chord could be thought of as harmonic partials. This fundamental frequency is first converted into a pitch-class value and then rescaled and interpreted as azimuth angle (i.e. a position on a horizontal circle). Similarly, the mean-pitch of the same chord is used to calculate an elevation angle. With these two polar coordinates a position for each chord on the surface of a unit-sphere is determined. The positions of the individual notes of each chord are specified via controlled perturbations of its center-position. The resulting "spatial clusters" (each corresponding to a chord) are visualized in the *3DC-lib* at the bottom left of the figure.

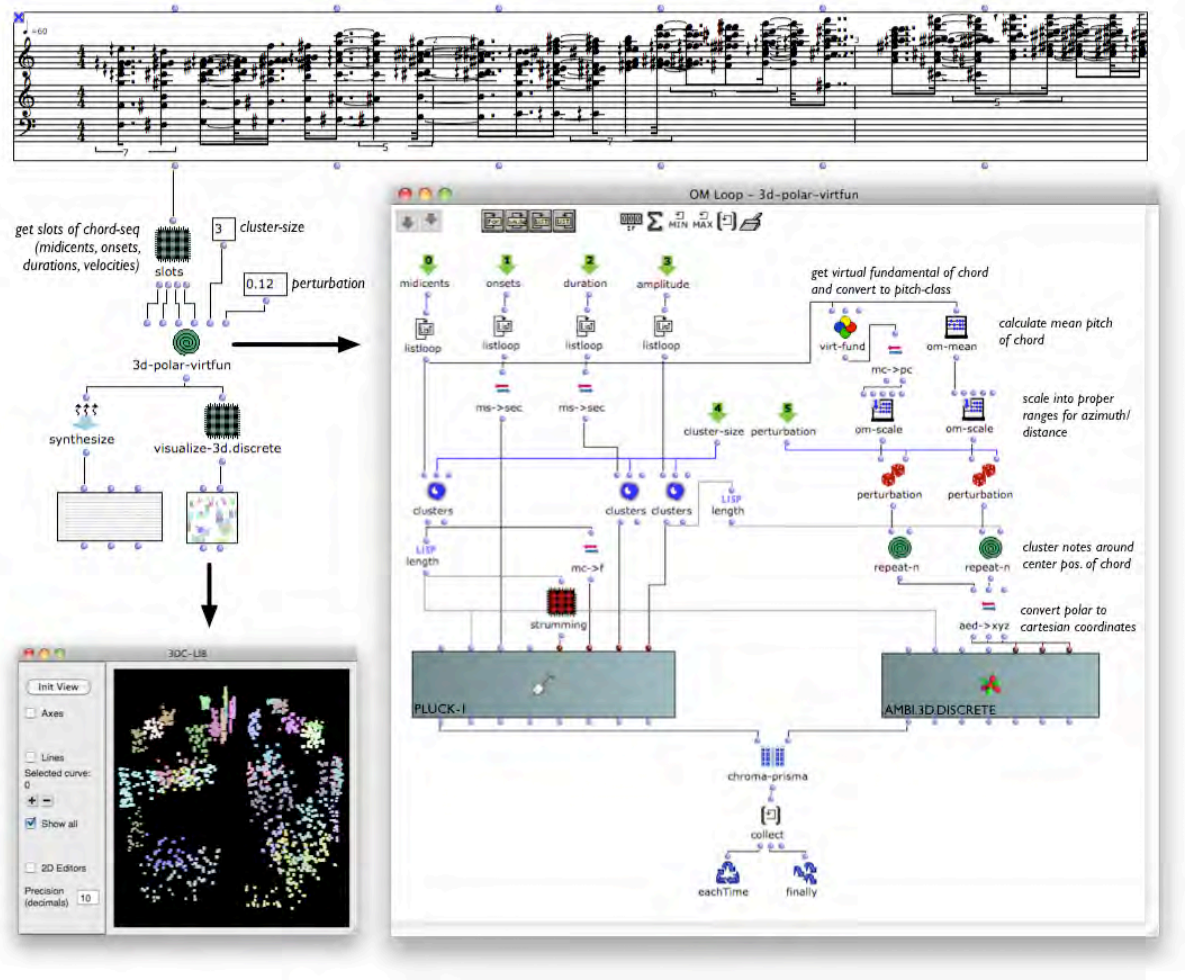

Fig. 2.9: Symbolic control of a spatial sound synthesis process in OpenMusic. The visual program on the right hand side (3d-polar-virtfun) converts the symbolic musical materials to synthesis and spatialisation parameters. The spatial positions of the synthesised score materials are shown in the *3DC-lib* at the bottom left.

Figure 2.10 shows a somewhat complementary approach: the use of concrete, external data to control the spatialisation of an existing soundfile.[4] In this example, the spatialisation parameters of a percussion (tabla) solo are controlled exclusively via data which is extracted from the soundfile itself using OMs sound analysis tools [32]. First, the source soundfile is analysed for transients in order to segment the percussion solo into individual tabla strokes. A fundamental frequency-estimation and partial-tracking is performed, from which profiles for pitch and spectral centroid are generated (a). For each sound segment (i.e. tabla stroke), its fundamental frequency and spectral centroid value is looked up and mapped to azimuth and elevation angle for spatialisation control. Since we are working in a differed-time paradigm, the duration of each segment can be used as control data to determine its distance from the centre position, and to set reverberation parameters (b). Consequently, every tabla stroke in the original soundfile will be assigned an individual spatial position and reverberation characteristics, correlated with its pitch, spectral centroid and duration.

A large variety of spatial sound synthesis applications can be realised by freely combining sound synthesis and spatialisation instruments. Granular synthesis, for instance, is a popular model for time-domain spatial sound synthesis (see for example [147, 232]). Typically, each sound grain is assigned an individual position in space, often controlled stochastically (rather than literally) in order to reduce the large number of parameters to a few manageable variables.

Another interesting model is "spatial additive synthesis": In spatial additive synthesis a complex sound is synthesised from elementary sinusoidal components which are spatialised individually. Figure 2.11 shows an example for continuous control of spatial additive synthesis in which hundreds of partials are synthesised, each with its individual set of dynamically changing sound synthesis and spatialisation parameters. It is also a nice illustration of how a complex spatial sound synthesis process – requiring large amounts of control-data – can be managed via a small number of conceptually meaningful, high-level abstractions.

[4] This technique could be considered an *auto-adaptive digital audio effect*, as the control data is derived from sound features using specific mapping functions [221].

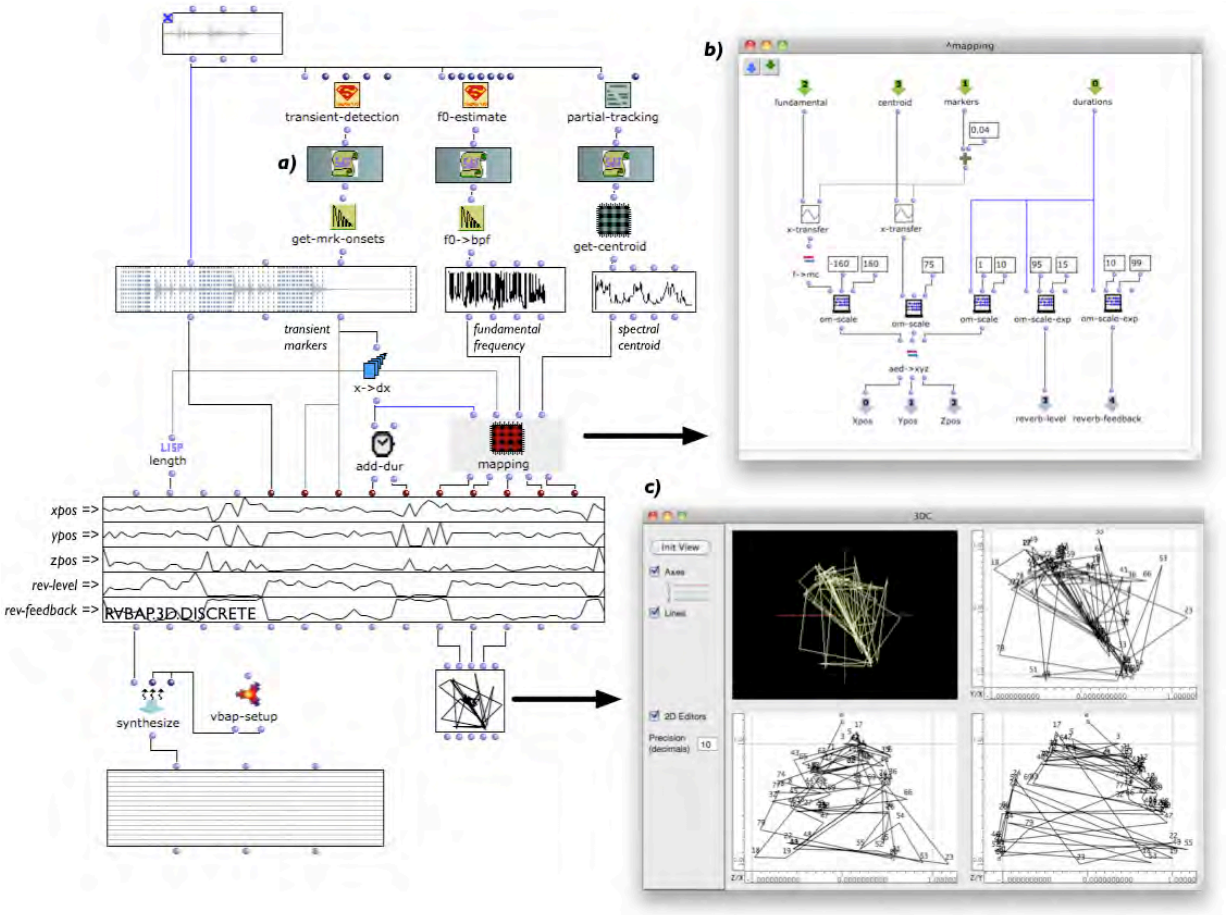

Fig. 2.10: Interfacing sound spatialisation and analysis tools in OpenMusic. Control-data is derived from sound analyses (transient detection, fundamental-frequency estimation and partial-tracking) of the original soundfile (a). A visual program (mapping) specifies mapping-functions to determine spatial positions and reverberation characteristics (b). The *3DC* visualises the spatial distribution of the sound segments, indices represent successive sound segments (c).

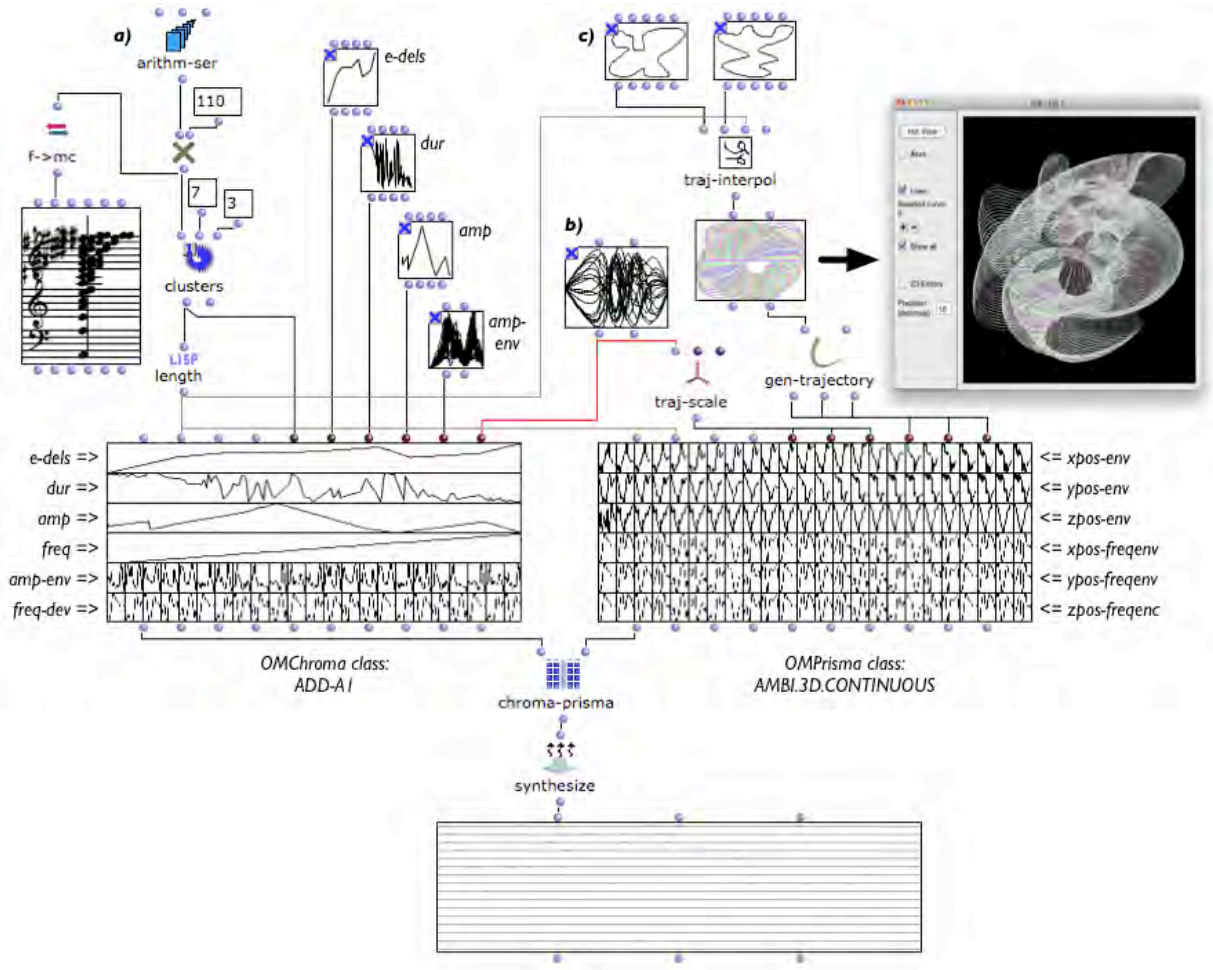

Fig. 2.11: Spatial additive synthesis using the classes *add-1* and *ambi.2D.continuous*: (a) a harmonic spectrum is generated (visualised as a chord) and additional partials (micro-clusters) added around each harmonic; (b) A set of envelopes (*BPF-lib*) is used to control both sound synthesis and spatialisation parameters; (c) Two manually-defined (i.e. hand-drawn) trajectories are interpolated over the number of partials. Each partial is assigned an individual trajectory, displayed in the *3DC-lib* on the right-hand side.

2.7 Conclusion

We have presented a system for the symbolic control of sound source spatialisation and spatial sound synthesis in compositional contexts. This system has been implemented as an extension of OMChroma in the computer-aided composition environment OpenMusic. Embedded in this framework, spatial sound scenes can be generated and manipulated via high-level structures and algorithms, which allows for explicit control of arbitrary numbers of control parameters, difficult to achieve via manual editing or other conventional approaches (e.g. real time, track-based).

The visual programming and compositional environment provides an extensive set of tools for the generation, manipulation and control of spatial parameters and processes. More importantly, the integration of spatialisation tools into environments for computer-aided composition allows spatialisation to be treated as a structural parameter, and enables complex relationships with other musical dimensions and in relation to a global compositional framework. Interesting relationships can for instance be created using external data derived from sound analyses or any other musical/extra-musical source or process.

OMPrisma separates the different stages of sound spatialisation into several layers as proposed in [169], with authoring in OM programs, description in matrices, interpretation via the *synthesize* function, rendering via Csound orchestras, and finally, the decoding and communication with physical devices using the external Multiplayer application.

Thanks to an original class-merging system between OMChroma (sound synthesis) and OMPrisma (spatial sound rendering), the concept of spatial sound synthesis is implemented in a generic way, allowing arbitrary combinations of sound synthesis and spatial sound rendering techniques. The control of sound spatialisation is tightly integrated in the compositional framework and benefits from the same flexibility and expressive power as sound synthesis and general compositional processes.

Future work is planned in several directions: The existing class-library could be extended with other spatial sound rendering concepts, such as SUG. More processor-demanding approaches such as ViMiC or WFS would be particularly promising candidates (facilitated by the offline-rendering paradigm). It would also be interesting to use the system in the context of directivity synthesis [229], for instance to synthesise artificial sounds with intricate directivity patterns. More complex processing chains could be

envisaged in this framework, including for instance spectral diffusion or other spatial audio effects. On the "control" level, different OpenMusic tools such as the *maquette*, for temporal modelling [4], or the *cr-model* for abstract sound representations based on time-frequency structures (see [47]r) form interesting contexts in which spatial sound control could be integrated.

So far OMPrisma has been used for the composition of a number of works, most notably *Cognitive Consonance* by C. Trapani, performed at the Ircam Agora Festival, Paris 2010. OMPrisma is distributed with the OpenMusic package through the IRCAM forum and is also available as an open source OM-library. More information and sound examples are available: http://www.idmil.org/software/omprisma.

Acknowledgements

This work was developed in collaboration with the IRCAM Music Representation team and the Input Devices and Music Interaction Lab of the Schulich School of Music of McGill University (IDMIL), funded by an Inter-Centre Research Grant from the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT).

Chapter 3

Integrating Gesture Data in Computer-Aided Composition: Paradigms for Representation, Processing, and Mapping

This chapter is based on the following manuscript:

Schumacher, M. M., Wanderley, M. M. (2016). Integrating Gesture Data in Computer-Aided Composition: Paradigms for Representation, Processing and Mapping. Accepted for publication in Journal of New Music Research, Taylor and Francis Group.

Abstract

In this paper we discuss physical gestures from the perspective of computer-aided composition and propose solutions for the integration of gesture signals as musical material into these environments. After presenting background and motivations we review related works and discuss requirements and design principles guiding our development efforts. A particular focus is given to the idiosyncracies of working with gestures in real-time vs differed-time paradigms. We present an implementation of our concepts as the library OM-Geste, integrated into the computer-aided composition environment OpenMusic.

Two examples are given for the use of this library for synthesis of symbolic score and audio data from real-world gesture recordings.

3.1 Background and Motivation

Environments for Computer-Aided Composition (CAC) are traditionally concerned with the representation and manipulation of symbolic musical materials, such as notes, rhythms, etc. Dating back to the early experiments of Hiller in the late 1950s [109], composers like Xenakis (1962), or Koenig (1964), used specific computer programs for the genesis of musical structures, mostly implementing combinatoric or probabilistic methods, in the tradition of algorithmic composition [13]. Today, the notion of algorithmic composition can be seen as a subset of CAC which has widened its scope in the sense of using the computer as a generic tool to develop and interact with musical ideas using programming languages and formalisms, also referred to as "compositional modelling" [16]. With the advent of more powerful computers and signal processing applications it became possible to introduce other media and data within computer-aided composition processes and integrate several aspects of a work in a single programming framework, such as synthesis of electronic sounds [47], spatialization [196], and orchestration [56]

Since these early days of CAC, recent trends in cognitive psychology have emphasized the importance of physical interaction for human expression and creativity, see. e.g. theories of *enaction* [83], *embodied cognition* [220], or *motor-mimetic cognition* [101]. Indeed before the 20th century, there was little separation between music performers and composers, and compositional work was often carried out via physical interaction and improvisation on instruments [103]. In fact, it has been argued by anthropologists, that language itself can be regarded as a form of gesture and that symbolic content and gesture are so tightly interwoven that very often they cannot be isolated [15].

What do mean when we talk about "gesture"? This term is often used in ambiguous ways and has many partially overlapping and even contradicting definitions (see e.g. [53]) Indeed, some researchers in the field consider the term too ambiguous and misleading and avoid its use at all [113], a notion that, if applicable to other research terms, would likely dramatically reduce our vocabulary. Delalande, for example classifies gestures in music into *effective*, *accompanist* and *figurative*, ranging from purely functional to purely symbolic [70]. Cadoz further distinguishes functions of the gestural channel -which is

responsible for the production of effective gesture into *ergotic*, *epistemic* and *semiotic* functions [51]. Gestures can further have communicative and cultural meanings, such as in*metaphorics* and *emblems* described by Kendon [119]. As in compositional contexts the semantics and communicative functions of gestures cannot be known in advance, we take a phenomenological viewpoint: considering gestures as multi-dimensional spatio-temporal morphologies of physical variables that can be sensed and represented in the form of digital signals [223].[1] Previous research on representation of sound signals (e.g. [37]), has demonstrated that continuous phenomena can be represented, abstracted into higher-level objects, and embedded into symbolic compositional contexts. This integration has proven fruitful and introduced new possibilities for compositional models and applications (see e.g. [6, 40]).

Despite the current trend of composers increasingly making use of gestures in their compositional processes (e.g. [85, 94]) there is a lack of strategic methods for including them in compositional frameworks and a sparsity of previous research and literature on the topic. The importance of the physicality of human expression seems to be underrepresented and not reflected appropriately in current computer-aided composition systems. Thanks to the proliferation of inexpensive sensor technologies and platforms in recent years, there is a wide range of possibilities for capturing gestures, from commercial controllers, to modular plug-and-play systems, to custom-made input devices. A closer look at their use within the field of computer music, however, reveals a curious dichotomy: while being widely used for real-time control of music performance, only recently and sparsely have these technologies been investigated for compositional purposes and thus, composers can rarely take advantages of these new technologies [96, 95]. With today's computing and sensing technologies, gestures can be captured, memorized, undergo transformations, and become a source for development of musical materials, thus become themselves objects to be operated on in computer-aided composition [181].

Indeed, the integration of gesture data within the deferred-time context of computer-aided composition allows to envisage a variety of new interesting scenarios and applications. For instance, it would allow composers to physically express and capture their musical ideas using gestural input devices. We could draw parallels here to the emphasis of

[1] This may include sound-producing actions (e.g. *excitation* and *modification*), ancillary movements (e.g. *phrasing* or *entrained* movements), sound-accompanying movements (e.g. *sound tracing* and *mimicking*), and communicative movements (e.g. *gesticulations*) [50, 70, 225].

physical expression in *action paintings* in the visual arts [90]. Another example would be the recording of gesture data captured from instrumentalists, for instance during the performances of compositional sketches (this may include *effective*, as well as *ancillary*[2] and *figurative* gestures). Since the morphology of gestures carry characteristic "signatures" of individual performers, this data can serve as materials for the creation of personalized music materials [227]. This can be related to the practice of composers making use of concrete sound material acquired through recording sessions with individual performers (see e.g. [159]). In the case of live musical performance with gestural controllers, it would allow composers to record sections and use these data as materials for the development of other parts of a piece. More generally speaking, the editing and arrangement of this gesture data might allow the composition of gestural performance scores (see e.g. [181]). The possibility of storing, representing and manipulating gestures in an integrated environment would also have benefits for related fields, such as performance studies and instrument design. As we will show in our discussion of related works, there is currently no complete framework providing the required functionalities for integrating gesture signals into environments for computer-aided composition.

In this article we present design concepts and solutions for representation, processing and mapping of gesture signals in computer-aided composition, implemented as the library OM-Geste for the OpenMusic environment [42]. This project can be regarded as part of our larger research efforts on mediating signal-symbol dialectics, following the idea of providing an integrated programming framework where various steps of compositional manipulations from signal to symbolic data, and from this symbolic data to sound and other musical data can be developed.

This article is organized as follows. The next section provides a background of related works and points out differences between he use of gestural input devices for human-computer-interaction (HCI) versus gesture signals as materials for composition. We will then discuss requirements for description, representation, manipulation and synthesis into direct music representations (section 3.3). Section 3.4 will introduce the library OM-Geste, and describe its general architecture for representing and processing gesture data in compositional contexts. In section 3.5 we will discuss the notion of mapping from a computer-aided composition perspective. This will be followed with two example

[2]The term "ancillary" is used to designate gestures which are integral part of a performance, but not required to produce the sound [227].

applications using gesture data captured from real-world dance and music performances (section 3.6), before concluding the article with a discussion (section 3.7).

3.2 Related Works

To this date there have been relatively few works involving physical gestures in the context of computer-aided composition. Most developments have focused on freehand drawing and sketching applications. Some of the earliest applications are graphical specifications of control data entered with a digital stylus. A historic example is the *UPIC*[3] system, developed at the "Centre d'Etudes de Mathématique et Automatique Musicales" (CeMaMu) with its first prototype in 1977 on a SOLAR mainframe computer [142]. Its origins date back to the early 1950s while composer Iannis Xenakis was working on the orchestral piece *Metastasis* and required graphic notation to specify continuous transitions in time-pitch space. Technically, UPIC can be described as a digitizer tablet with a graphical interface connected to a bank of wavetable oscillators. Envelopes and waveforms could be stored in memory and arranged into compositions by drawing arcs and lines on the canvas. In 1987 a real-time version was introduced which allowed controlling sound synthesis by moving the stylus on the tablet, thus biasing the system more towards a performance instrument than a compositional tool. UPIC had a number of successors running on generic hardware, most notably "Metasynth" [230], "HighC" [1] and "Iannix" [63]. The latter is not bound to a specific synthesizer and can be regarded as a graphical sequencing and scripting environment for sending control messages via OpenSoundControl [234].

At this point we would like to point out a few differences to our objectives. Common to these applications, is that although using gestural input, the interest lies not in the spatio-temporal qualities of gesture itself, but rather its trace or artifact. Taking the drawing example, the cause-and-effect relationship producing this artifact cannot be recovered from the figure; it can thus not be considered a representation of the gesture itself but rather the result of an action (see also [202] for the distinction between "gesture" and "action"). Note, that our objective is not the use of gestural controllers as alternative means for interacting with the composition environment, but rather the use of gesture data as compositional materials. It is also noteworthy that drawing gestures (as any other human motion) are

[3]UPIC is an acronym for "Unit Polyagogique Informatique du CeMaMu".

typified by morphological characteristics resulting from physical constraints of the system performing the gesture [123], and our aim is to provide a framework which permits the integration of any type of physical gesture.

Besides these graphical input applications and idiosyncratic solutions, the field of CAC has been largely isolated from the world of physical gesture. More recently, there has been a renewed interest from an HCI perspective for using alternative input devices in computer-aided composition contexts, based on the observation that even highly computer-literate composers use physical means such as pen and paper to sketch their initial creative impulses, thereby effectively reducing cognitive load compared to the use of conventional computer interfaces [65]. Like their predecessors, these developments are based on drawing gestures and employ digital styluses and interactive paper technologies as alternative input devices to the composition environment [218, 93, 92]. Garcia et al. presented an artistic application with composer P. Leroux, in which historic musical scores were used as templates which could be re-drawn, thereby providing gestural data which was streamed/imported into the CAC environment [96]. This work is close to our conceptions, however its focus was on a specific application rather than a generic framework.

Although developed in different eras and contexts, we can see a number of conceptual similarities between these early freehand drawing-based systems and the more recent approaches making use of advanced sensing and interface technologies: They employ alternative input devices for interacting with the composition environment, essentially for augmenting the WIMP-based interaction paradigm [20]. Moreover, most of these tools focus exclusivley on drawing gestures, which have specific morphological characteristics [228] (see also [65, 212] for other examples in this category). Our research interests lie in a different direction, which is finding solutions for integrating and representing any type of gesture as musical material into compositional processes.

One work which comes close to our objectives was presented by Ramstein and Cadoz [50, 181]. The two main differences are that their system is intended exclusively for sound-producing gestures (more precisely, the classical keyboard playing activity), thus excluding other types of gesture, such as freehand, gesticulations, etc. Further, since it is intended as a gesture editor, it has no means for producing direct music representations, such as symbolic scores or audio. Nevertheless, there are a number of overlapping aspects to the work presented in this paper, related to coding, representation and manipulation of gestures, which are relevant to be mentioned here [52]: the authors use two alternative

representations of gesture data, an *internal* one, which is close to the signal and intended for economic transmission and storage, and an *external*, higher-level representation, closer to gesture descriptors (called "gestual attributes", e.g. sharpness of attack) on which the user operates. They further stress that for compositional work, a discretization of continuous data streams in terms of temporal segments ("gestual events") is necessary, which we also consider an important requirement (cf. section 3.3.2). In terms of user interfaces, different views with different scales are proposed: a "zoomed-in" view for editing details of a single segment, versus a larger-scale "zoomed-out" view, for comparing multiple segments and their relationshps. The authors further state that representations should not be fixed, but extendable by the user, for instance by adding new gestual attributes. Concluding this section, it can be seen that all the cited works deal with gesture and composition in some way. Each of them, however, focuses on specific aspects, devices or applications and none of them offers a complete framework for integrating gestures into symbolic compositional processes.

3.3 Design Guidelines

In this section we describe a number of requirements we have identified during our research and which served as design guidelines for the development of the software library presented later on. These requirements can be grouped into three domains: 1) description and coding, 2) abstraction and representation, and 3) mapping and synthesis.

3.3.1 Description and Coding

When talking about gesture-data we need to distinguish "actions" which can be described at a high level, possibly by discrete symbols, from "movements", which are spatio-temporal morphologies that can be represented as multi-dimensional, temporal signals [84]. Aiming at a maximum of compatibility and flexibility for dealing with the great variety of input device and capture contexts, an ideal system should support the following features:

- different data resolutions and dimensionalities in order to not restrict the use of gesture recordings to specific devices or systems
- heterogenous temporal streams rather than fixed sampling rates

- flexible structure and grouping of data
- strong semantics via use of standardized descriptors
- possibilty of extension with user-defined descriptors
- associate and synchronize other music-related data and media, e.g. symbolic scores, audio, video, etc.
- storage in a file format which is light-weight, user-extensible and supported by many computer music applications

A number of projects in the gesture research community have tackled similar problematics, e.g. [133, 146, 54]. One collaborative research initiative which aims to fulfil the requirements outlined above is the GDIF project (Gesture Description Interchange Format) [113]. An interesting aspect for our project is that GDIF has been conceived as a sister format to SDIF (Sound Description Interchange Format) – which has been used in our previous computer-aided composition projects for coding of sound and spatialization data [35, 46]. SDIF has a number of useful properties for storing gesture-related data, such as the possibility of defining new descriptors on the fly, or the notion of time-tagged frames containing data of variable dimensionality, organized into heterogenous streams. SDIF is well-suited for the multi-layered descriptor scheme proposed by GDIF [157] and supported by a growing number of computer music environments (including OpenMusic [35]). Therefore, we chose for our developments to adopt the GDIF concepts for gesture description, and use SDIF files as container for storage and interchange.

3.3.2 Abstraction and Representation

A second requirement for the use of gestures in compositional models are possibilities for abstraction and representation. In CAC environments, users typically manipulate symbolic representations rather than concrete low-level data itself. Therefore it is necessary to abstract the continuous data streams into higher-level representations, closer to symbolic materials that can be integrated in musical thought and calculus. We propose the following main directions:

- discretizing the continuous data into smaller-scale units that can be integrated as discrete objects into symbolic compositional processes [37, 32];
- deriving symbolic representations from low-level data by means of abstraction, e.g. by reducing data through a formalized process, such as resampling a time-domain waveform to break point function; described by a few number of points
- transcoding, for instance by extracting features or higher-level descriptors from low-level sensor signals.

Besides functionalities for segmentation, abstraction and transcoding, it should be possible to build different representations (see also [192] for a similar discussion on sound representations). These representations serve as models which determine affordances provide an interaction context for manipulating the underlying domain object [20]. Choices for abstraction and representation of gesture signals are strongly dependent on individual composer's concepts and we believe a more sensible approach to imposing a one-size-fits-all solution is the development of tools and building blocks which can easily be adapted to meet individual composers' needs.

3.3.3 Mapping and Synthesis

The ultimate goal of the integration of gestures is to produce a musical output, be it in the form of a score, sound or other musical media. While in CAC synthesis parameters can be directly generated (for instance as the result of algorithmic processes), a different approach is manipulating a distinct data set (such as gesture data) and applying a process of *mapping* to convert this data set to musical structures or sounds [76]. In the field of instrument design the notion of mapping refers to the association of gesture variables to sound synthesis parameters and is a highly discussed topic [151]. But how does the notion of mapping transfer to computer-aided composition?

While Chadabe pointed out limitations of the instrument-oriented concept of "mapping" [61], which according to Downie "deflates the awesome power of the algorithmic before it can appear" [79], Doornbusch states that mapping in instrument-design and composition share a number of commonalities and that certain taxonomies can be applied to both [75].[4] For

[4]Interestingly, one of the pionerring projects in computer-aided composition implemented a mapping from text-to-pitch, strinkingly similar to the techniques described in Guido Arrezzo's *Micrologus* (appr. 1026) [14].

instance, while in instrument design the activity of *making* the instrument and *performing* it are usually carried out in separate stages, in computer-aided composition there is less of a distinction and all three involved components (source, mapping, synthesis) might be manipulated, possibly interdependently. It is also evident that the synthesis object may take different forms (e.g. symbolic score, sound synthesis), and different scales (from micro- to macro-structural elements of a piece).

One way to approach this problem is by reconsidering the notion of "mapping" for compositional contexts. Drawing from Ariza's taxonomy of CAAC (computer-aided-algorithmic-composition) systems we consider mapping in a broader sense, as the conversion of an indirect music representation through a formalized process into a direct music representation. A direct music representation is a "linear, literal, or symbolic representation of complete musical events, an event list (a score in Western notation or a MIDI file) or an ordered list of amplitude values (a digital audio file or stream)" [13]. Indirect music representations are incomplete and unordered structures or specifications, such as functions, objects, images, etc. or gesture data. According to this definition two requirements can be identified:

- a system which extends the notion of mappings to the more general concept of generating direct music representations from a source data set
- functionalities for producing this direct music representation, i.e. synthesis functionalities;

3.4 The library OM-Geste

In this section we will present our design concepts, implemented as the library OM-Geste for the OpenMusic environment. For this framework we have made developments in three main areas:

- coding and import/export of gesture descriptors via SDIF files
- objects for representation and tools for processing of gestures
- a mapping system which seamlessly integrates into OpenMusic

3.4.1 Segmentation and Abstraction

Gesture signals are descriped using the GDIF specification and imported/exported into the environment via SDIF files (cf. section 3.3.1). Once a gesture recording has been imported, we need to convert and abstract the signals into a representation which is closer to symbolic materials, useful for compositional manipulation (see also the concept of "transcoding" in [52]). We developed a new object, called *gesture-array*, which allows to extract specific descriptors from an SDIF file and organize them in a hierarchical structure: The top level consists of *gesture-streams*. Each gesture-stream represents a descriptor (e.g. 3D position in cartesian coordinates). A gesture-stream consists of a group of *substreams*. Each substream describes the evolution of a scalar variable over time (one degree-of-freedom). Figure 3.1 illustrates this structure on the example of a gesture array containing two gesture-streams: 3D position data and absolute velocity (the magnitude of the first derivative of position data).[5]

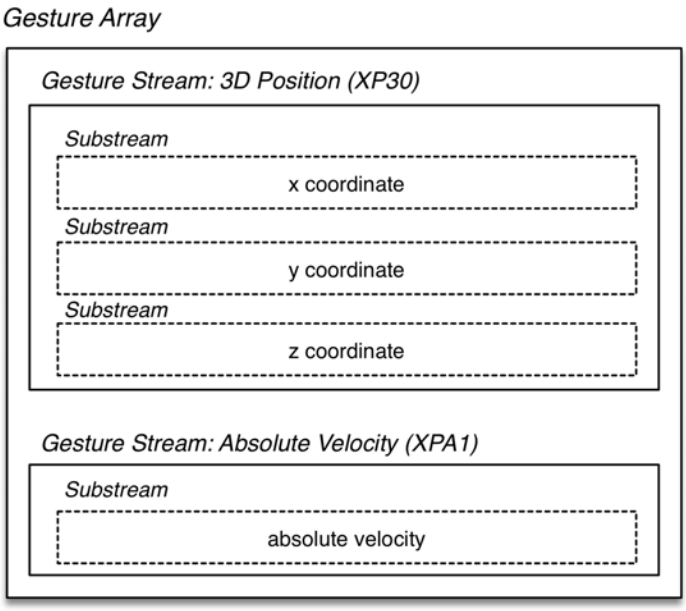

Fig. 3.1: Structure of a *gesture-array* object: *gesture-streams* representing descriptors, containing *substreams*

[5]The hierarchical structure of the *gesture-array* object can be compared to the "internal representation" described in [52], in which *units* contain *channels* containing *lanes*.

Although being organized in a hierarchical structure, the individual gesture signals (substreams) are represented as continuous, one-dimensional time-series data, which in this form is not convenient for compositional activities yet. In order to integrate gestures into compositional formalisms, higher-level representations are required, closer to the notion of musical objects and symbolic materials that can be associated with compositional processes (cf. section 3.3.2). Accordingly, the continuous data in a *gesture-array* need to be segmented and abstracted into entities which are closer to symbolic representations and can be organized into larger-scale structures.

In order to provide a higher level representation, we developed a dedicated *gesture-model* object, which serves as a container for *gesture-streams*, discretized into temporal segments (*gesture-segments*). A *gesture-model* provides a tabulated interface in which the horizontal dimension represents time. Thus, rows represent *gesture-streams* and columns represent represent temporal segments. The individual fields represent *gesture-segments*.

The literature reveals a variety of approaches for gesture segmentation which can be roughly divided into two groups: techniques requiring no prior knowledge of the data (e.g. minimum velocity of displacement [23]) versus techniques which need to be "trained" with samples of the data, such as hidden markov models [24] or dictionaries [55]. A particularity of compositional environments is that semantics of materials users work with, such as gesture signals and the criteria for their discretization, cannot be presupposed. For instance, gestural units might be based on symbolic information (such as words in spoken language) or external criteria which are not available in the data itself. Moreover, the temporal structure of the gesture data might itself become a compositional parameter. It can be seen that rather than imposing a pre-defined gesture segmentation strategy, it is more sensible to provide different tools for segmentation which can be configured based on the compositional context. Accordingly, the segmentation is specified via a temporal structure (a list of values in seconds) that is determined externally, by arbitrary user-defined processes or data. Within this framework we have classified three types of segmentation as shown in Figure 3.2.[6]

[6] A different taxonomy was presented in [181], differentiating "manual" segmentation (which would to our "expert" segmentation) from "automatic" segmentation (which could be any of our three types if carried out algorithmically).

- **External** segmentation: based on referential external data, for example if gesture data was captured synchronously to the performance of a musical score, playback of a sound file, video recording, etc.;
- **Internal** segmentation: based on the gesture data itself, for instance by extracting signal features (e.g. minima of quantity of motion);
- **Expert** segmentation: based on "expert" knowledge for specifying segmentation points manually or by a user-defined procedure.

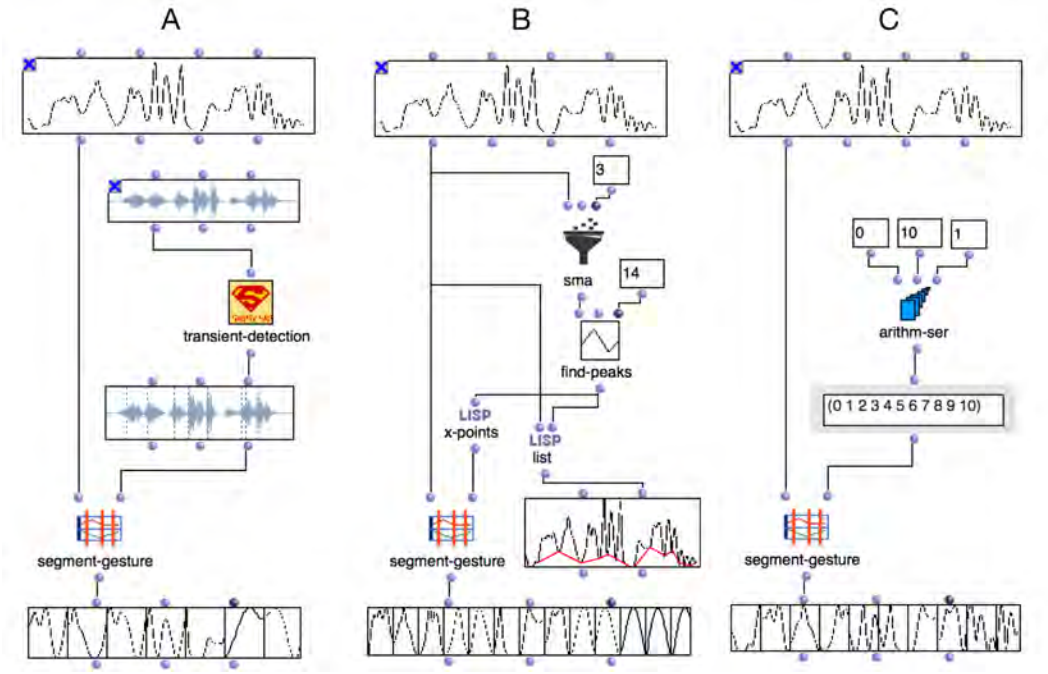

Fig. 3.2: Three types of gesture segmentation in OM-Geste. **A** External: Transient detection in a referential audio file for specification of temporal locations; **B** Internal: Gesture data is smoothed, and analyzed for troughs in the waveform to specify temporal locations; **C** Expert: A series of timepoints is generated by a user-defined process.

Once a temporal structure is defined, a *gesture-model* can be built from a *gesture-array* and a list of temporal locations. A special function *segment-gesture* takes these data as arguments and outputs an instance of a *gesture-model*. During this process, two operations are carried out (segmentation and transcoding): First, the individual *gesture-streams* are parsed and segmented into discrete units. Second, the gesture descriptor of each gesture

segment is looked up and the data is converted into a higher-level OpenMusic object on the basis of the dimensionality of the descriptor. To give a few examples:

- 3-dimensional position data is converted into *3D-trajectory* objects (3-dimensional break point curves).
- 1-dimensional descriptors (such as absolute velocity) are converted into break point functions (*BPF* objects).
- multiple instances of the same descriptor (e.g. absolute acceleration obtained from multiple inertial measurement units) are converted into a collection of superposed objects sharing the same editor/timeline (called "object library" in OpenMusic, e.g. a *BPF-lib* object).

Thus, a *gesture-model* organizes the gesture data into a 2D matrix, in which each field corresponds to a temporal segment of a gesture descriptor, represented as higher-level object with an associated editor. Note that for each *gesture-stream* an inlet/outlet is created through which the corresponding data can be easily accessed. The gesture segments can thus be inspected and manipulated using the standard editors users are already familiar with, and can be seamlessly integrated into other compositional processes. Figure 3.3 shows an example OpenMusic patch for creating a *gesture-model*. Gesture signals are imported from an SDIF file and converted into a structure of *gesture-streams* within a *gesture-array* object. The temporal specification for the segmentation is derived from from a referential audio file. The *gesture-streams* are segmented, converted into individual objects, and organized into a *gesture-model* object.[7]

Note, that the transcoding process, i.e. the conversion of gesture streams into OpenMusic objects is completely invertible. This is an important requirement as during the compositional experimentation gesture materials may be converted multiple times between different representations. It is also possible to segment gesture-models multiple times, allowing to introduce or extract different temporal structures from the same gesture data. Since the gesture segments are abstracted into standard OpenMusic objects, they can be extracted, analyzed and manipulated using algorithmic means or graphical editors.

[7]The organization of gesture segments into a larger-scale struture can also be related to the notion of a "formula" in [52].

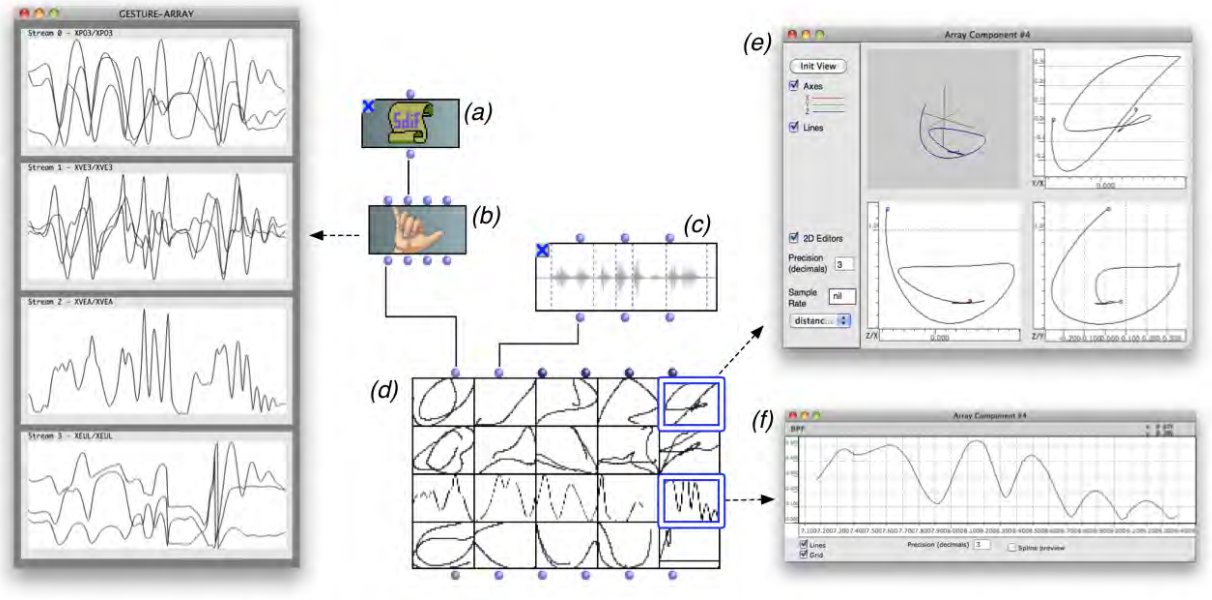

Fig. 3.3: Gesture data contained in an SDIF file (a) is extracted and converted into a structure of *gesture-streams* in a *gesture-array* object, shown in the editor on the left (b). A temporal structure, specified via markers in an audio file (c), is used to segment the streams and instantiate a *gesture-model* object (d). Different types of gesture descriptors are converted into different OpenMusic objects, visible in the editors on the right (e): *3D-trajectory*, (f): *break point function*.

3.4.2 Manipulation of Gesture-Models

Another important functionality (cf. sections 3.3.1 and 3.3.2) is the possibility of associating and synchronizing gesture data with other musical or symbolic materials in a common structure. This can be useful for including additional streams, e.g. derived from analysis of existing streams, or other data a user wishes to associate and synchronize with the contents of the *gesture-model* (e.g. textual/symbolic annotations, images, etc.). OM-Geste provides functionalities for adding and synchronizing arbitrary data to existing gesture-models. Depending on type and temporality, data are automatically segmented and integrated into the tabulated structure of the gesture-model. Figure 3.4 shows an example for adding and synchronizing new streams to the gesture-model, more precisely, data derived via analysis/processing of existing streams and arbitrary external data. Note, the segmentation of the new objects inside the *gesture-model*.

3.4.3 Conditioning and Processing

In this section we describe our approaches and developments for inspection and manipulation of gestures. When working with sensor measurements, it is common to condition the data via smoothing, scaling, clipping, etc. to remove unwanted noise and jitter, but also to carry out more sophisticated processing, e.g. for extracting higher-level descriptors. A number of toolboxes for gesture processing in the context of digital musical instrument design have been described, e.g. the "Digital-Orchestra-Toolbox",[8] part of the *McGill Digital Orchestra* project [86]. Note, however, that the types of processing that can be carried out in real time are restricted. This is unlike offline-processing where non-causal operations, e.g. searches, temporal manipulations, and optimization techniques are possible. To this end, we implemented a toolbox of over 40 functions, including filters, feature extraction, statistics and data reduction/optimization (see also [48] for similar toolboxes).

In gesture processing applications it is often convenient to determine processing parameters empirically by careful tweaking (e.g. filter coefficients for smoothing filters). Fortunately, the recent "reactive" extension allows OpenMusic programs to be executed in response to user events, which permits designing visual programs for interactive editing and processing. e.g. visualizing the result of a process in real-time while moving a slider

[8] http://www.idmil.org/software/digital_orchestra_toolbox

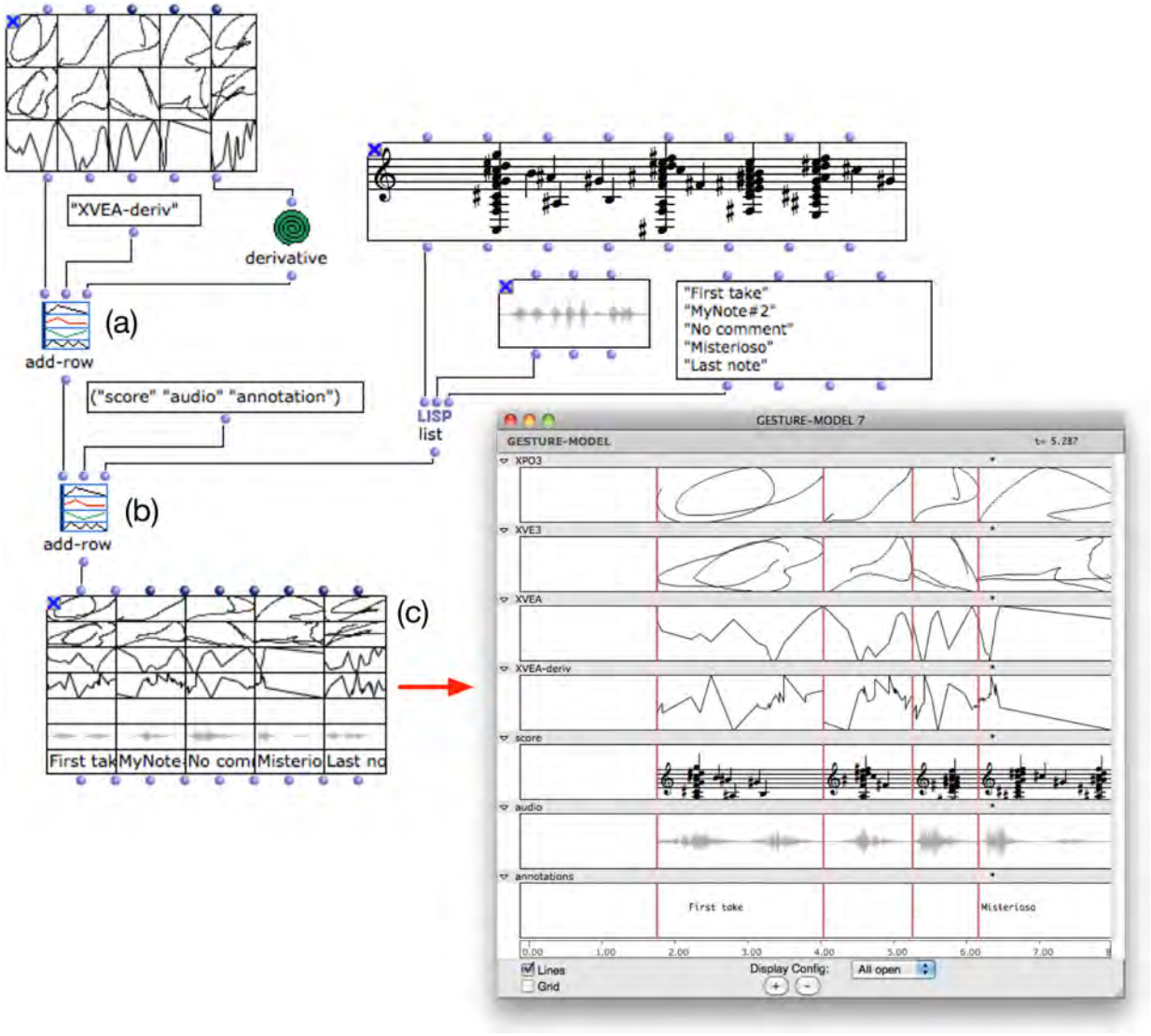

Fig. 3.4: (a): A stream is extracted, its derivative calculated and appended as a new stream to the *gesture-model* via the function *add-row*. (b): A number of OpenMusic objects (symbolic score, audio file, textual comments) are added as new streams to the *gesture-model*. (c): The original objects are automatically internally segmented and included within the *gesture-model*.

[34]. It is also possible to process gesture-models via higher-order functions, which allows to apply arbitrary user-defined algorithms (OM patches or LISP functions) globally, or to selected data fields. Figure 3.5 shows examples for interactive processing, inspection and global editing of gesture data in OM-Geste.

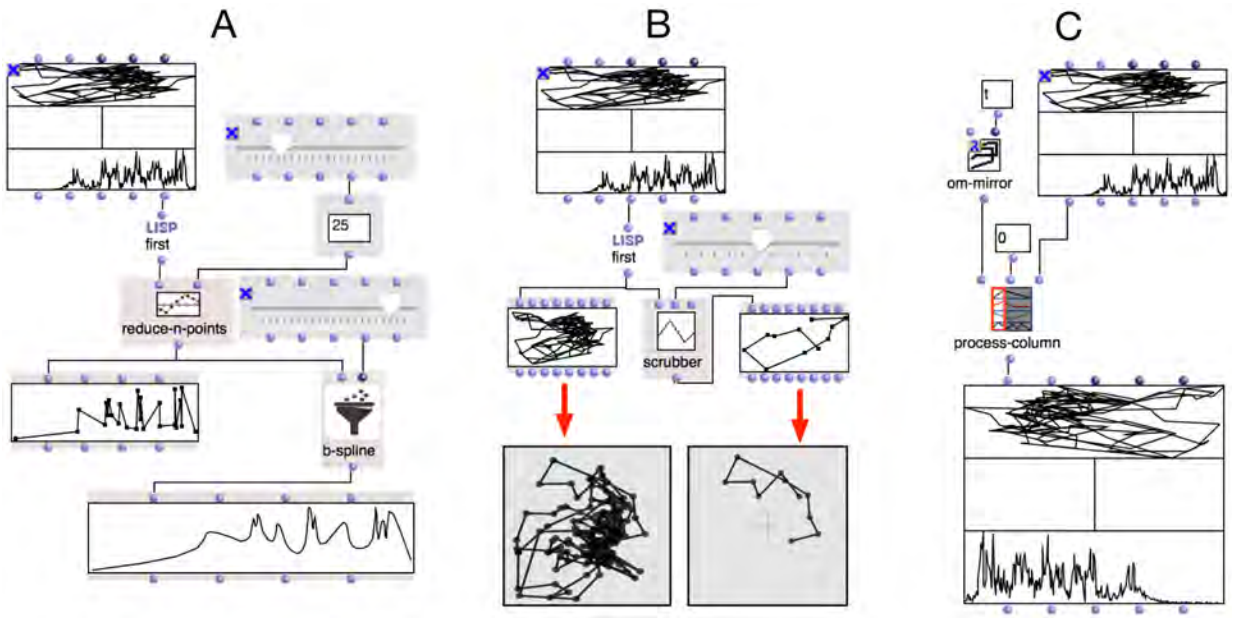

Fig. 3.5: Interactive processing of gesture data. **A**: slope-based reduction of time-series data to 25 points and smoothing via b-spline interpolation. **B**: scrubbing through segments of a *3D-trajectory* object; **C**: the higher-order function *process-column* applies the function *om-mirror* globally to all fields of the gesture-model.

3.4.4 Export of Gesture Descriptions

One possible application of OM-Geste is its use for visualization, editing and processing of gesture data without subsequent conversion into a direct music representation (e.g. for study purposes or synthesis at a later stage). In this case the library can be regarded as a gesture processor and editor, see also [50, 181] for a similar application. After processing and manipulating gesture signals in the environment, the resulting *gesture-model* can be stored in SDIF files. These files can for instance be interchanged with other environments and possibly used for real-time streaming (see [46] for a similar approach for spatialization data). We developed a function *make-gdif-buffer* which takes a *gesture-model* object and a list of SDIF frame/matrix type specifications as arguments and produces an *SDIF-buffer* object,

i.e. a structure of SDIF type definitions and frames according to the GDIF specification[9] [157]. This object can then and exported to an SDIF file using the standard SDIF tools provided in OpenMusic. Figure 3.6 shows an example for manipulating a gesture-model via a user-defined process and its subsequent storage as an SDIF file.

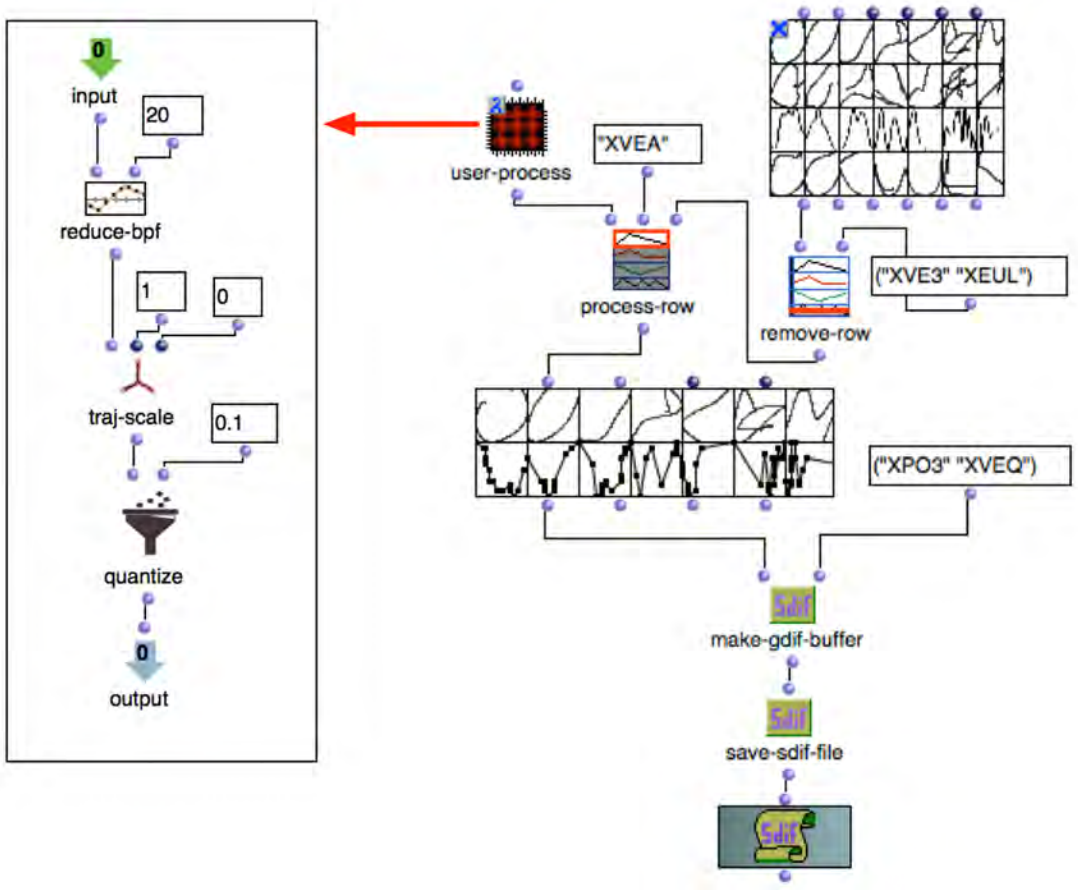

Fig. 3.6: Top right: the function *remove-row* is used to remove two descriptors (XVE3 and XEUL) from the *gesture-model*. Top middle: an OpenMusic patch "user-process" is connected to the function *process-row* for processing of the XVEA descriptor. Left: The slope-based data reduction function (*reduce-bpf*) is used to downsample each segment to 20 data points. The profile is inverted and quantized to steps of 0.1 using the *traj-scale* and *quantize* functions. The resulting two streams in the resulting *gesture-model* are then formatted using the stream/frametypes XPO3 and XVEQ, and stored as an SDIF file.

[9]The temporal structure of the *gesture-model* is described in the SDIF file as a separate stream of temporal markers (SDIF frames of type "1MRK").

3.5 Paradigms for Gesture Mapping in CAC

A particular interest of our work is reconsidering the concept of mapping for compositional contexts. While in the context of DMIs mapping is commonly understood as the association of gesture variables to sound synthesis parameters, Salter et al. demand to "re-invigorate the notion of mapping and move it beyond the simplistic input/output model" [188]. The notion of mapping in composition typically has a wider scope, for example as the correlation of data sets between domains, the "connection between structures, or gestures and audible results in a musical performance", or even more abstract, as "an idea in one domain being manifested in another" [78]. The possibility of manipulating abstract models and indirect music representations has been described as a defining characteristic of CAC systems [13, 16]. At some point these indirect music representations must be converted into a direct representation. As described in section 3.3.3 we use the term "mapping" to refer to the explicit design of functional relationships linking variables of an indirect music representation, including gesture data, to parameters of a direct music representation. Consequently, some form of mapping is inherent in the activity of composing with gestures within a CAC system. Depending on the type of this direct music representation, these parameters may correspond to sound synthesis controls, MIDI events, common music notation, etc. The question then is, how to design a mapping system that is powerful and intuitive to use for variable source data, temporal scales and synthesis domains.

3.5.1 Real-time vs Deferred-time

The notion of mapping in computer-aided composition has both similarities and differences to mapping in the design of digital instruments for real-time performance (see [136] for a comprehensive overview of approaches to the latter). A popular notion is the *systems* point of view, in which mappings are conceptualized as an "...out-of-time snapshot of input/output control potential", similar to the classical "flowchart" paradigm [219]. Many existing environments and tools for designing mappings are based on this idea (e.g. [2, 140, 171, 3]). This approach arguably has its roots in instrument design applications where the notion of mapping is often similar to a data-flow graph which is devoid of temporal representations. Two characteristics are inherent to this notion of mapping which may be limiting for compositional contexts:

The first relates to representations of time. Tools which are based on the notion of a snapshot of connections are mostly agnostic of temporal structures. Although time may be represented implicitly (data rates, recursive filters, etc.) there are few possibilities for specifying temporal parameters or structures as part of a mapping. Since in deferred-time contexts the temporal dimension can be manipulated as any other parameter (a distinctive property of offline composition) this notion can be limiting. It would be desirable to allow manipulating time within a mapping function (e.g. to match the temporality of the desired synthesis system), and/or allow combining different temporal structures for individual parameters.

A second consideration concerns the gestural variables available for mapping. One popular strategy in instrument design is the use of multi-layered representations of gesture data, by converting raw sensor measurements into a set of intermediate parameters used for mapping. These parameters can be based on extracted features [141], perceptual models [12], interaction metaphors [231], etc. Among the motivations for multi-layered approaches are abstraction [226] and generalizability [114]. Some intermediate layers are regarded as interface-specific and are created in an earlier stage than the mapping between gesture variables and synthesis parameters. The situation is different in compositional contexts, since the semantics of an abstraction can be part of compositional conceptions which are not known in advance.

Indeed, it is not obvious where to draw the line between transformations being part of the gesture description versus part of a mapping.[10] This can be related to the distinction by Hunt et al. whether a mapping is regarded as part of an instrument or part of a composition [112]. In order to obtain a satisfactory musical result both gesture data and mapping can be manipulated in an iterative and explorative approach. Indeed, there seems to be a relationship between the nature of the gesture data and the choice of mapping: if the source data is complex, composers tend to prefer simple, ratiometric mappings and vice versa [74]. This is congruent with the notion of multi-layered mappings as a tradeoff between simplifying the mapping process versus the structure of the gesture description [225]; or in terms of sound synthesis control, creating complexity as part of the score versus part of the instrument [207].

[10] A first characteristic is the recursive structure: any manipulation of the gesture data as part of a mapping can be reinserted as a new descriptor into a gesture-model.

3.5.2 Mapping as Program

The integration of a mapping system into compositional environments requires a different approach as compared to instrument design. The concept of control as a link from gesture variables to synthesis parameters might be too limited for describing the structure of generative and algorithmic music systems [61, 79]. In CAC contexts, gesture data, generative processes, and manual specifications might be interconnected at different levels in mapping processes and used to specify parameters ranging from symbolic to signal domains. Therefore, a compositional mapping system should provide a consistent approach for connecting and setting parameters of any synthesis or musical generation system available in the environment. In terms of usability and workflow it should allow for seamless integration and leverage user's existing skills by providing tools and interfaces they are already familiar with.

Thanks to advanced programming features in CLOS [91], we were able to design a system in which a mapping is conceived as a *program* (rather than a state of connections) supporting the full range of programming features available in the underlying language (data- / control structures, recursion, etc). Due to the recursive nature of the language, it can be structured into subprograms and is itself a program, embedded in a higher-level program. From a user's perspective, it is at the same time code and notation expressed in a visual language for modelling forms, structures and relations [16]. For seamless integration this program can be designed as a LISP function or a standard OpenMusic patch. Gesture variables can be selected via named *inputs*, and data associated with synthesis parameters via named *outputs*. This allows to connect and control arbitrary parameters of any musical object (OM class) with minimal effort.

Figure 3.7 illustrates the general concept of the mapping system in OM-Geste. The higher-order function *map-gesture* constitutes the core of the system. This function requires three arguments: a *gesture-model* object (a), an OpenMusic patch or LISP function (b), and an OpenMusic class (c). *Map-gesture* (d) then iterates through the gesture-segments contained in the gesture-model, and for each segment produces an instance of the connected OpenMusic class and sets the slots with values determined by the mapping program. Inside the mapping patch "hybrid-specs" (cf. figure 3.8) inputs can be created and named to extract source data from respective slots (corresponding to gesture-streams) of the gesture-model. Outputs can be created and named to set values for the respective slots of the

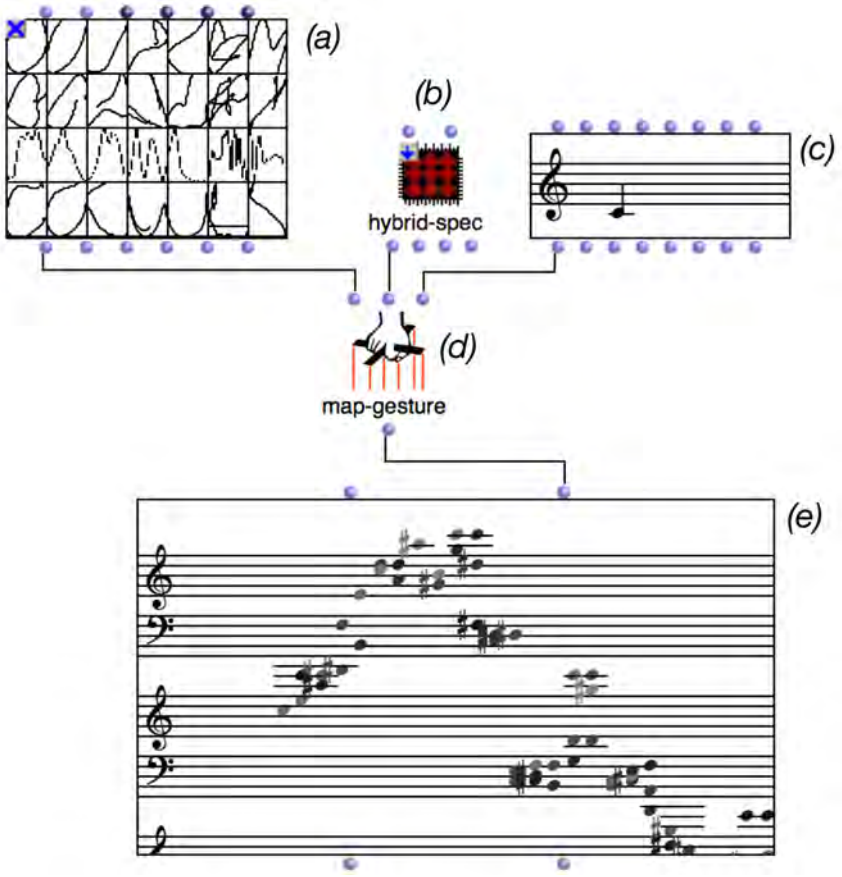

Fig. 3.7: A mapping process as a visual program in OM-Geste. From left to right: a *gesture-model* (a), a patch defining a mapping (b), and an OpenMusic class (*chord-seq* object) (c). The function *map-gesture* (d) performs the mapping for each segment of the *gesture-model* and outputs a list of *chord-seq* instances (sequences of chords) displayed as different staves in a *multi-seq* object (e). Grayscales represent dynamics.

OpenMusic class (corresponding to synthesis parameters).[11] The mapping itself can is defined as a visual program using any objects and functions available in the environment. Since mapping programs are standard OpenMusic patches, one can conveniently make use of other features of the programming interface, such as copy/paste actions, comments, structuring of subroutines into subpatches, possibly exporting parts of the program as files. Another benefit is that a visual program (cf. section 1.1.3) can quickly elucidate the important structural aspects of a mapping in the fashion of a flowchart. It should thus be more intuitive and flexible as compared to one-line expressions found in many current tools. Also note, that since classes in OpenMusic have default values, not all parameters of a synthesis process need to be specified (see also the *micro* and *macro* modes in [226]). Thus, the interface is context-sensitive in that it does not display unnecessary information (such as all available inputs and outputs at any time), but only those a user wishes to include in the mapping.

3.5.3 Hybrid Specifications and Temporalities

While in applications for real-time performance the relationship between control variables and synthesis parameters is typically instantaneous (i.e. inputs are linked to outputs at any given time), in CAC contexts time itself can become a mapping parameter, allowing for example expansion and reduction of the gesture data as part of a mapping function. This allows users to associate gesture segments to musical elements of arbitrary scale, structure, or temporality, as Doornbusch states:"from the micro scale of sound synthesis through to the macro scale of the structural model of the piece" [76]. Therefore, our notion of mapping goes beyond frameworks and control theories for digital musical instruments which are often based on temporality and scale [200, 137].

Another extension of the traditional mapping concept is the possibility of combining control paradigms within a single program, e.g. literal specification, generative algorithms, sampled data, and symbolic descriptions. Figure 3.8 shows the contents of the mapping patch "hybrid-specs" (from the previous figure): A *chord-seq* object (a sequence of chords in linear time notation) is used as the class to instantiate for each gesture segment. As visible from the names of the outputs, the specified parameters for each *chord-seq* are pitch (*lmidic*), onset time (*lonset*), duration (*legato*) and velocity (*lvels*). This data (a vector of

[11] Note, that it is possible to visually identify (from the round in-/outlets) how many connections a mapping involves without having to open the mapping patch.

25 values per chord-seq) is produced using a combination of gesture variables (3D-position and absolute velocity), literal values, symbolic objects, and generative algorithms.

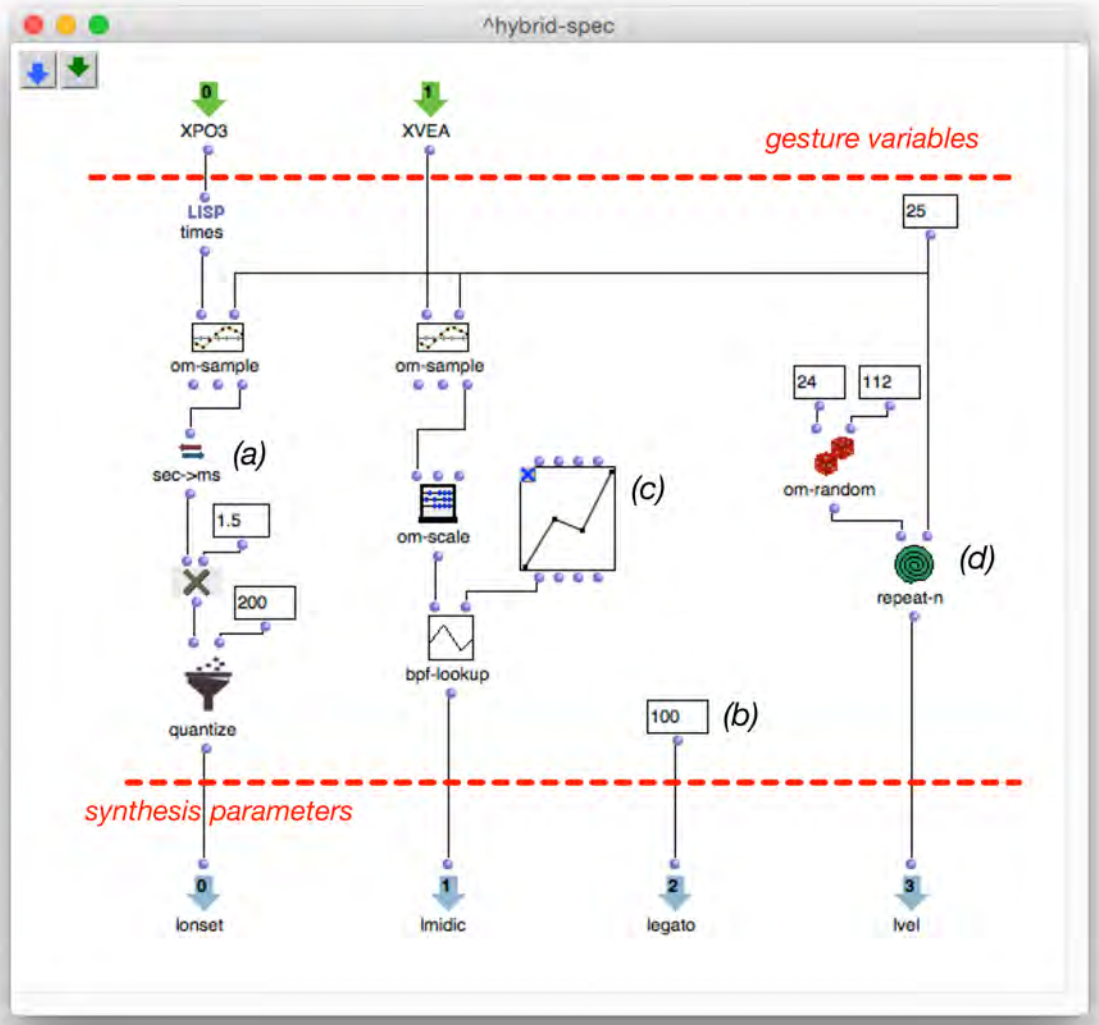

Fig. 3.8: View at the mapping function "hybrid-specs" from Figure 3.7. Synthesis parameters are determined via different types of data an processes: (a) re-sampled and time-stretched gesture variables, (b) constant, (c) *BPF* object, (d) generative algorithm.

3.6 Case Studies

In this section we discuss two examples for the use of OM-Geste for synthesis of direct music representations from real-world gesture recordings. In the first, a symbolic musical score for piano will be created based on gesture recordings of a dance performance. The second will use gesture data captured during a DMI performance for the controlling a sound synthesis and spatialization process. We chose these two examples because they exemplify two distinct types of music-related gestures (dance versus instrumental performance), and since sensing devices, gesture descriptors and resolution of recordings differ significantly. For both examples we recorded gesture data in sync with audio/video recordings. The original video recordings are available online.[12]

We will describe a full workflow starting with import of gesture data via SDIF files, abstraction into gesture streams and OpenMusic objects, application of different types of segmentation and processing, and finally synthesis into different music representations (symbolic musical score and spatial audio). We designed the mapping programs aiming at gestural coherence, i.e. a perceivable relationship between the performer's effort and energetic morphology in the sound. In order to validate our results we kept the temporal scale of the produced musical outputs the same as in the original gesture recordings. This allows to synchronize the resulting musical output to the video recording and view the sonic output in relationship to the performed gestures.

3.6.1 Dance Performance

Music-related gestures can be roughly divided into two groups: gestures of those that produce the sound (*performer* gestures) versus gestures of those that perceive the sound (*listener* or *dancer* gestures) [116]. For this example we were interested in inverting the role of dancer gestures, traditionally performed to existing music (e.g. classical ballet), to the creation of instrumental music from the physical motions of a dancer. In this way the dance is not a figurative interpretation, but becomes source and material for the composition of new music (see also [23] for similar applications using MoCap recordings).

The gesture data used in the first example is a recording of a dance performance by Sophie Breton choreographed by Isabelle van Grimde. This choreography was originally created as part of the artistic research-creation project *Les Gestes* whose principal

[12] http://tinyurl.com/Dance-DMI-original

investigators were Sean Ferguson and Marcelo Wanderley at McGill University and Isabelle van Grimde and the dance company Van Grimde Corps Secrets [138]. We recorded inertial measurement data of a 3-axis accelerometer embedded in a visor-like device worn on the head of the dancer (see [107] for descriptions of the device). Figure 3.9 shows dancer Sophie Breton wearing the *Visor* during her performance in Pollack Hall of McGill University.

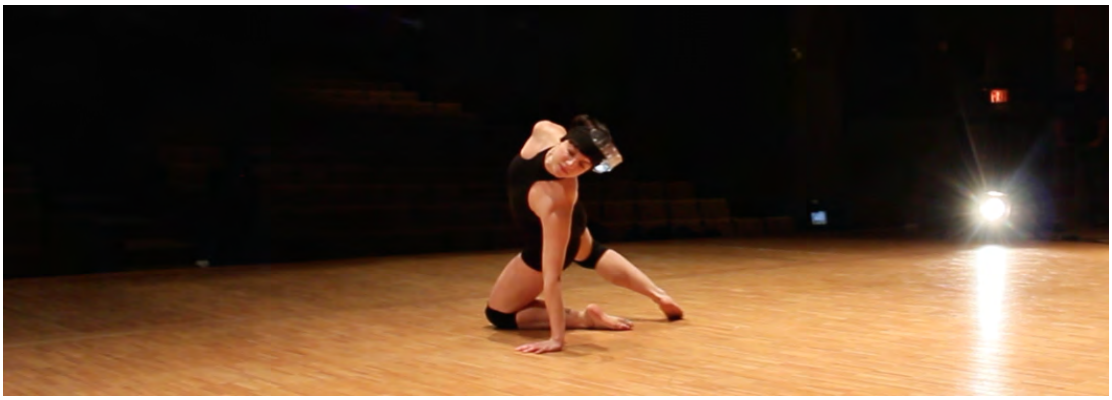

Fig. 3.9: Video frame showing Sophie Breton wearing the *Visor* during a recording of her dance performance. Video by Audréane Beaucage. Used with permission.

The sensor data was transmitted wirelessly from the visor device using the 2.4 GHz XBee implementation of the ZigBee IEEE standard to a central computer, which recorded the data in the Max software [175] at a sampling interval of 60 milliseconds with 10bit resolution. The raw accelerometer data was preprocessed prior to recording using the *digital-orchestra-toolbox* [139], to derive descriptors for 3D-acceleration, 3D-orientation, and vector magnitude of the first finite difference of 3D-acceleration (jerk).[13] This recording is an example case for inertial sensing without absolute reference points and without external referential data or media. As the temporal rate and numerical precision of the recorded gesture signals are comparatively low, additional descriptors were derived using functions in OM-Geste for smoothing and upsampling of the recorded data. This process is illustrated in Figure 3.10.

On the left hand side visible at the top is a gesture-model containing about 12 seconds of the original recorded gesture signals. The 3-dimensional acceleration data ("XA30") is extracted and upsampled using b-spline interpolation. Additionally, the magnitude vector of the first finite difference of acceleration ("XAA1") is extracted and smoothed using a

[13]The corresponding SDIF frame and matrix types are XA30, XRE0, XAA1.

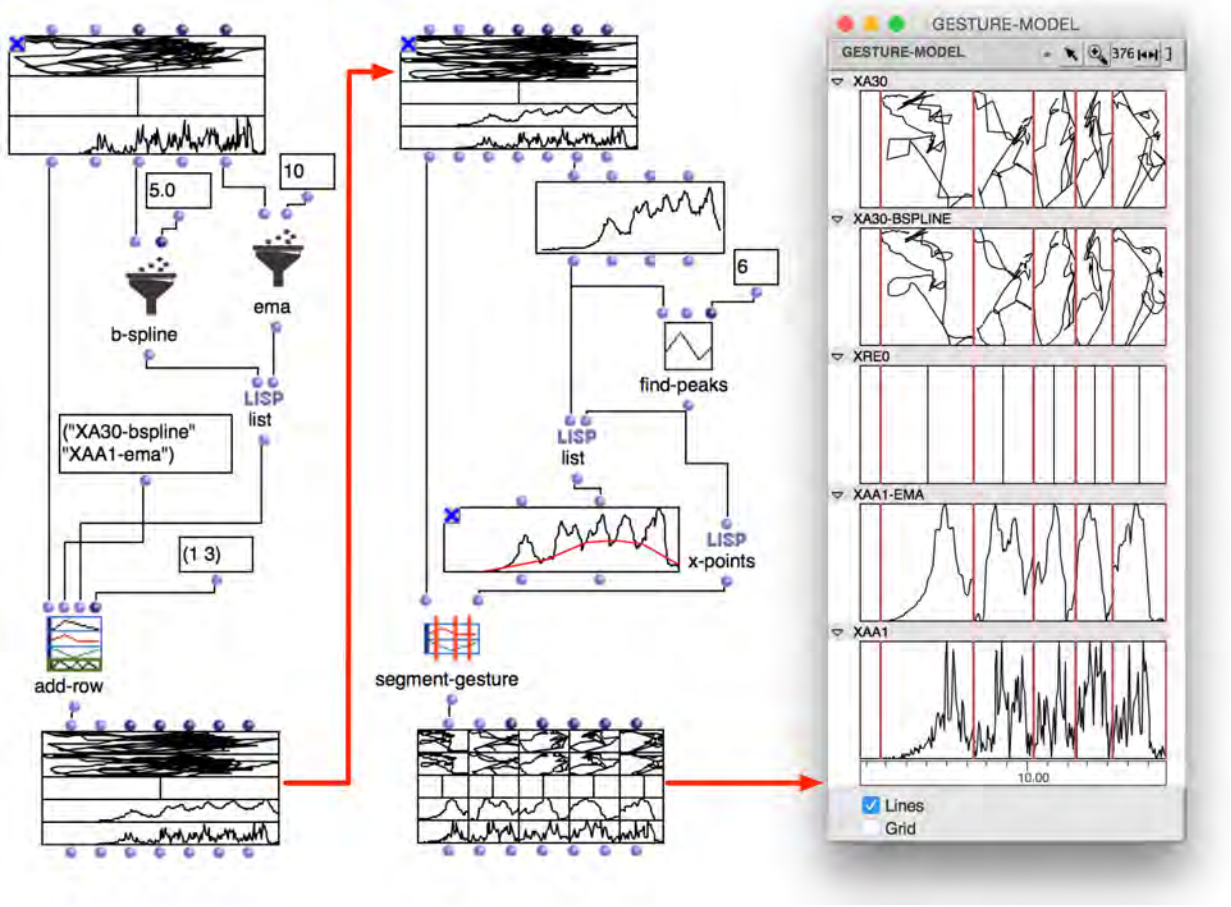

Fig. 3.10: Top left: Original gesture data in a *gesture-model* object. The functions *b-spline* and *ema* are used for smoothing and interpolation. The function *add-row* add these descriptors to the new *gesture-model* visible at the bottom left. Middle: The function *find-peaks* determines troughs in the smoothed data. The temporal locations of the troughs are used for segmentation with the function *segment-gesture*. Right: editor displaying contents of the resulting, segmented *gesture-model*.

recursive filter (*ema*). These two new streams are added as "XA30-bspline" and "XAA1-ema" to the *gesture-model* via the function *add-row* (see section 3.4.2). In the middle of the figure we see at the top the resulting *gesture-model*. Since we recorded the gesture data without external referential data here we determine the segmentation based on analysis of the gesture streams themselves ("internal segmentation", cf. section 3.3.2). The "XAA1-ema" descriptor is extracted (displayed in the BPF object) and the function *find-peaks* is used to determine the temporal locations of troughs in the jerk data (via Laplacian edge detection). These troughs are known to be indicative of points of demarcation between gestures (see also [129, 23] for similar approaches). Note that smoothing and interpolation of the raw data was essential to determine segmentation points, since the recorded signals were too noisy (due to small inaccuracies and jitter in the measurement and the motions of the dancer) for edge detection.[14] The *BPF-lib* object at the bottom displays the descriptor data superposed with line segments connecting the troughs. The temporal locations of the throughs are used for segmentation of the *gesture-model*. Visible on the right side of the figure is the editor displaying the contents of the segmented *gesture-model*.

As in the mapping example in section3.5.3 we used again a *chord-seq* as the musical object to instantiate for each gesture segment (cf. section 3.5.3). For sake of simplicity the mapping function is a relatively simple and direct mapping of 2 gesture variables to 3 synthesis parameters. Visible at the top of the mapping program on the right-hand side of the figure, are two inputs for the two descriptors vector magnitude of acceleration (XAA1) and the smoothed version (XAA1-ema) (B). The function *om-sample* is used to linearly sample 25 discrete values for each gesture-segment. The XAA1 values are scaled linearly from an input range of 0 to 1 to an output range of 3600 to 8400, providing values for pitch in midicent (*lmidic*). The XAA1-ema values are normalized for each segment into a range between 1 and 120, and used to specify the dynamic values of the notes as MIDI velocities (*lvel*). 25 temporal locations are sampled and converted from seconds into milliseconds, to serve as onset times (*lonset*). Note, that depending on the duration of the segment this translates into greater or smaller time intervals between individual notes. With these specifications, the function *map-gesture* creates and outputs a list of *chord-seq* objects (one for each segment). The function *merger* concatenates this list and creates a single *chord-seq* object. This object is then converted into a *poly* object, which quantizes

[14] OpenMusic's reactive extension proved very useful as it allowed us to tweak parameters of our processing functions in real-time for obtaining the desired results (cf. section 3.4.3).

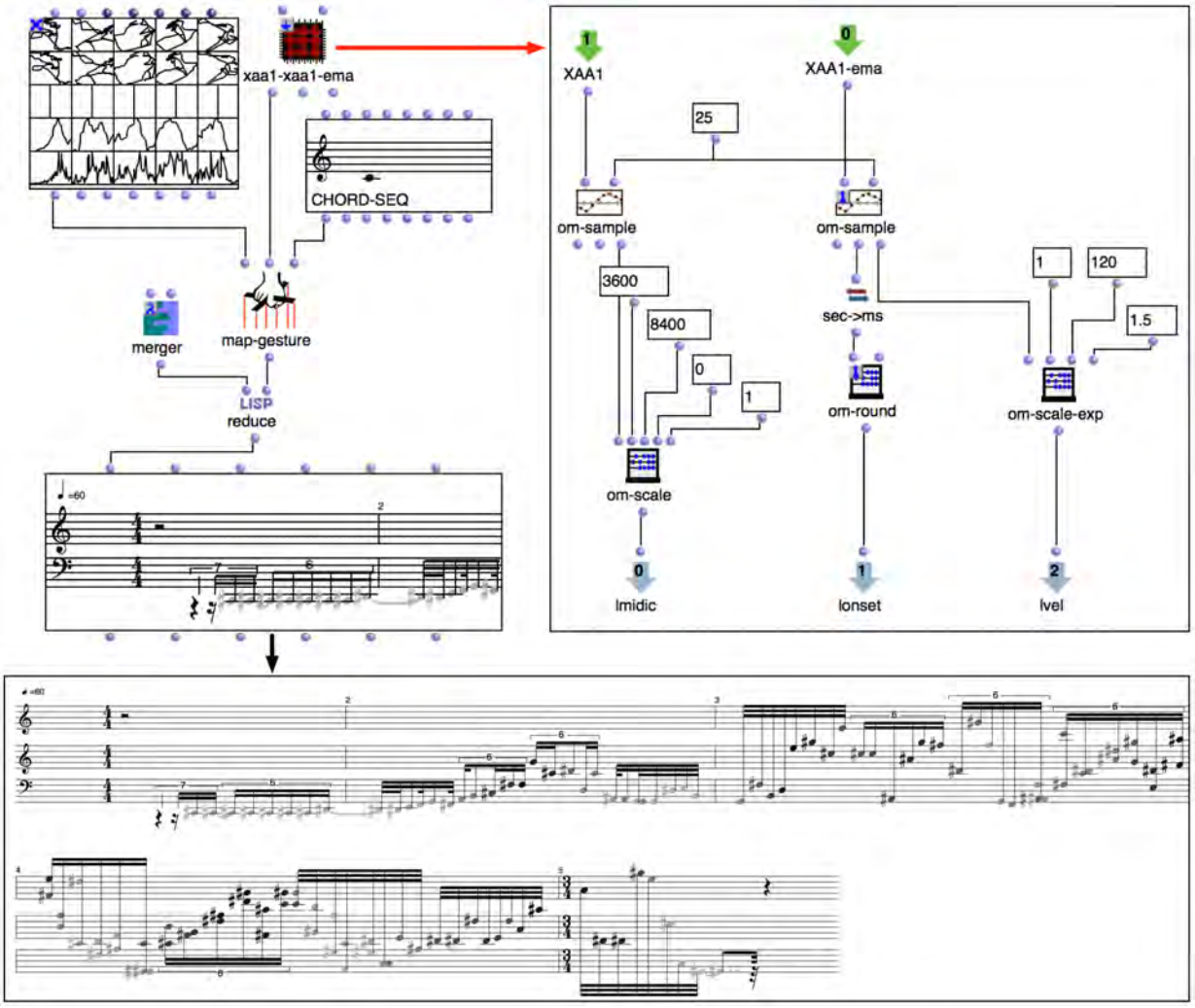

Fig. 3.11: Left: Mapping process for synthesis of a symbolic musical score. Right: Two gesture descriptors are resampled to set values for pitch, onset time and velocity of 25 discrete notes per gesture segment. The resulting musical structures are concatenated and quantized into a metric representation (bottom).

the linear (continuous) time representation in ms into a metric time structure at 60BPM. The resulting score is visible at the bottom of the figure (grayscales represent dynamics). This piano score was rendered to audio using a MIDI synthesizer and and added to the original video recording. The resulting video file is available online.[15]

3.6.2 DMI Performance

Our second example is based on a performance by Kristian Nymoen with a DMI titled *SoundSaber*. This instrument is shaped as a 120cm long rod and uses a MoCap system for sensing absolute position of the tip relative to the room. From this 3D position reading other motion descriptors were derived and used to drive a real-time sound synthesizer via explicit mapping. The author's motivations for development of this MoCap-based instrument were the high accuracy and temporal resolution which allows sensing subtle changes in motion, see [158] for a description of this instrument. In our example, we will control a sound synthesis and spatialization process using the recorded motion descriptor data. The data of the performance was recorded synchronuously to audio and video data into SDIF files using a number of modules implementing the GDIF concepts in Max [157]. Position data was tracked through a Qualisys Motion Capture system with nine Oqus 300 cameras at a sampling rate of 200 Hz. The raw position data was conditioned (smoothed, normalized) and preprocessed in the Max software to derive additional descriptors: 3D-velocity, vector magnitude of velocity, and orientation in Euler angles.[16] Note the differences (both quantitative and qualitative) between this motion-capture-based DMI performance versus the intertial measurements of the dancer: The MoCap data was obtained with external reference, i.e. absolute position relative to the room as compared to inertial measurements on the body of a dancer. The MoCap data is of much higher temporal and numerical resolution (200Hz vs. 16.7Hz and 32Bit vs. 10Bit). For the DMI performance, synchronized referential data in the form of an audio recording is available. Figure 3.12 shows K. Nymoen with the SoundSaber instrument.

This recording is an example of a scenario in which gesture signals were recorded synchronously with other media or data - which can serve a template for "external" segmentation (cf. section 3.4.1). Since in this case an audio file of the sound synthesis is available, we can import it into OpenMusic's sound editor and use the displayed time-

[15] http://tinyurl.com/Dance-mapping

[16] In accordance with GDIF, the SDIF matrix and frametypes are XP30, XV30, XVA0, XOE0.

Fig. 3.12: Left: Kristian Nymoen performing with the *SoundSaber* DMI. Right: close-up of optical markers on the pvc tube. Used with permission.

domain waveform to manually set temporal markers at points of minimum amplitude - which due to the mapping design in the original performance correlates with moments of minimum velocity in the motion of the performer. Figure 3.13 shows on the left hand side the extraction and conversion of 3 gesture signals from an SDIF file to a gesture array and subsequently to an (unsegmented) *gesture-model* (A). We can see the waveform of the original audio and the manually-set temporal markers in the sound editor which are used to specify a list of temporal locations for the function segment-gesture (B). Visible at the bottom left is the segmented gesture-model. On the right-hand side we can see how two additional descriptors are derived via differentiation and smoothing of 3D-velocity (3D-jerk), and calculation of the vector magnitude of this new descriptor. These streams are added as XV31 and XVA1 (C). The resulting *gesture-model* is shown in the editor on the right.

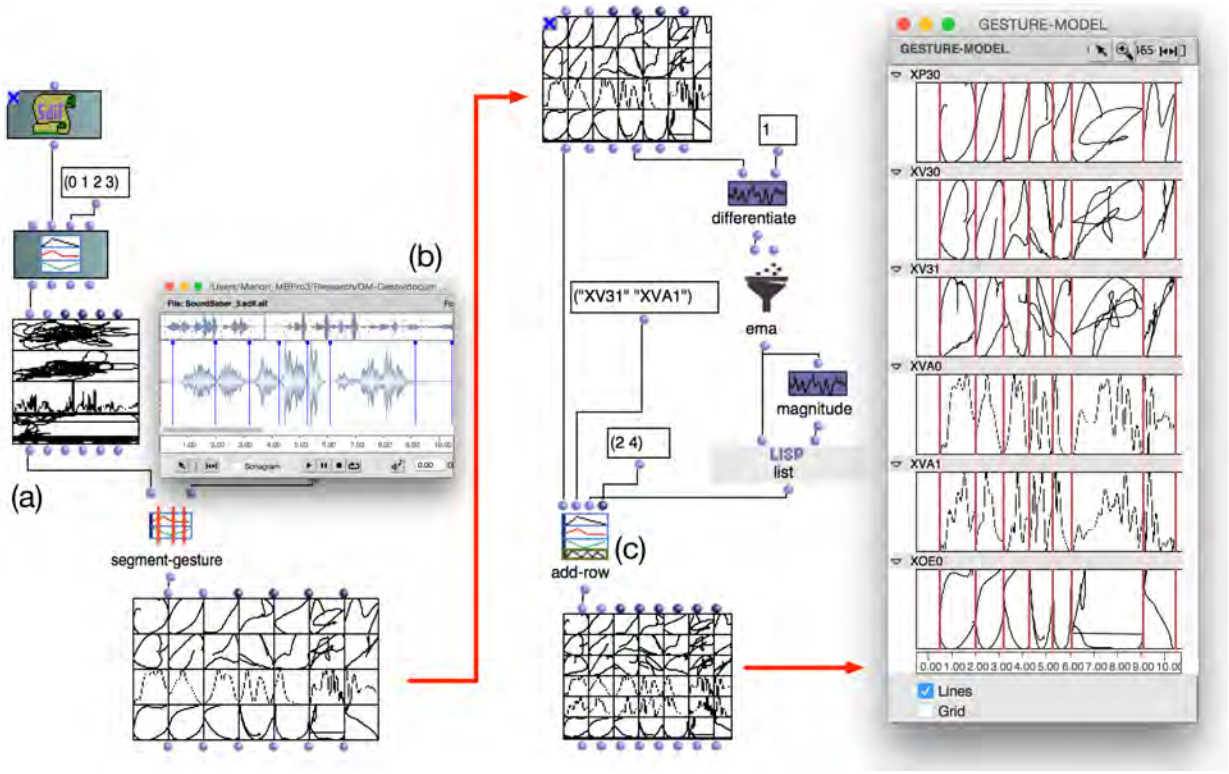

Fig. 3.13: Import and Processing of Gesture Data from the *SoundSaber* DMI. Extraction of gesture descriptors and construction of *gesture-model* (a). Manual specification of temporal location for segmentation based on external data (audio recording of sound synthesis) (b). Processing of gesture signals for adding additional streams (c). Right: resulting *gesture-model*.

We designed the mapping based on the same considerations of action-sound relationships as in the original (real-time) instrument [158]. For sound synthesis and spatialization in OpenMusic we used the libraries OMChroma and OMPrisma, which are briefly described here. OMChroma is a framework for sound synthesis adapted from composer Marco Stroppa's *Chroma* system [8]. Similar to the gesture-models discussed in this article, control data is represented via matrix structures. Classes dedicated to various sound synthesis algorithms can be instantiated to devise sound synthesis processes which are rendered via the Csound language [87]. OMPrisma is a library which adopts the same high-level control structures and provides an object-oriented system of classes for spatial sound rendering, sharing a consistent control interface in compliance with SpatDIF specifications [195, 168]. An original class-merging system is provided, which allows the combination of classes for

sound synthesis, processing and spatialization into single hybrid structures, in which these aspects can be controlled in an integrated process (see [196] for more detailed description). For our example we constructed a merged class, combining an FM synthesizer with a resonant bandpass filter and a spatialization module for reverberated VBAP [144].

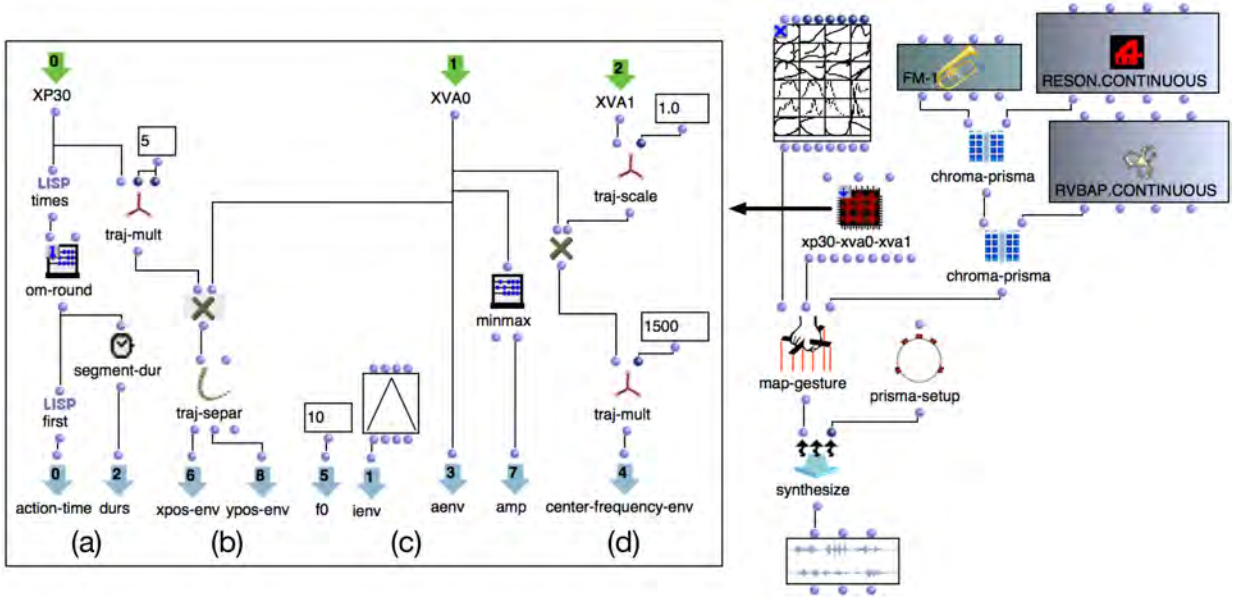

Fig. 3.14: A many-to-many mapping process for control of sound synthesis, processing and spatialization. Specification of global temporal parameters extracted from 3D-position data (a). Specification of spatial trajectory as *many-to-one* mapping (b). Hybrid specification of FM synthesis parameters using constants, envelopes, gesture data (c). Control of filter center-frequency by combining gesture descriptors for absolute acceleration and jerk (d).

Figure 3.14 shows the visual program for the mapping and synthesis process. We will use the topology by Hunt and Wanderley for describing the structure of this mapping [111]. The class-merging process for building the spatial sound synthesizer is visible on the right hand side of the figure, using two invocations of the function *chroma-prisma*. On the left hand side of the figure we see the mapping function (OM patch) which is applied to each segment of the *gesture-model* (see also section 3.5). Note, the different types of specification (literal, symbolic, functional) used in the mapping program for setting parameters for sound synthesis, processing and spatialization: Global temporal parameters of the synthesis process, such as the onset time (*action-time*) and the duration (*durs*) are determined from the temporalities of the original gesture segments, extracted from the descriptor for 3D-position XP30 (a). The parameters controlling the spatialization

(sound source position) are controlled as a *many-to-one* mapping: The 3D coordinates are first scaled by a constant, before being multiplied with the vector magnitude of acceleration (XVA0) and then separated into individual envelopes describing the two horizontal cartesian dimensions (b). The fundamental frequency (*f0*) for the FM synthesis is specified as literal value (10) and the modulation index (*ienv*) as a symbolic representation (a triangular envelope) which is scaled to the duration of each segment. The amplitude envelope (*aenv*) and its scale (*amp*) are controlled via the vector magnitude of 3D-acceleration (XVA0) as a one-to-many mapping (c). The center-frequency of the bandpass filter is controlled as another many-to-one mapping: the magnitude of jerk (XVA1) is first normalized into a range from 0–1 and then multiplied with XVA0. The resulting values are multiplied times 1500 for controlling the center-frequency of the bandpass filter in Hz (*center-frequency-env*, d). The synthesis process was rendered into stereo audio and added to the original video recording of the performance. The video file is available online.[17]

3.7 Conclusion

In this paper we presented a system for integrating physical gestures as musical materials into the symbolic domain of computer-aided composition. We outlined requirements related to coding, representation and manipulation. We discussed the notion of mapping from a computer-aided composition perspective with reference to real-time gesture control for music performance. In this context it was particularly interesting to revisit the idea of multiple abstraction layers, temporal scales, and hybrid specifications. Indeed, in compositional workflows the boundaries between generation, processing and mapping are blurry and can be applied interdependently and at multiple stages. We introduced the library OM-Geste, compared our approaches to related works, and showed two examples for real-world applications. Our system is intended to provide an open and generic framework which may constitute a first step for reintroducing physical expression into computer-aided composition.

Future work will investigate strategies for conversion between alternative representations of gesture segments as well as flexible data structures which allow for multiple simultaneous segmentations of gesture signals. Another direction we are exploring is the integration of machine learning techniques, both in order to develop tools for segmentation, and for

[17] http://tinyurl.com/DMI-mapping

compositional applications of implicit mapping strategies . The integration of gesture data is part of a recent trend exposing the computational power and expressivity of CAC environments as components within larger-scale systems, towards combining algorithmic-generative approaches with data-driven and real-time applications. In particular for complex synthesis processes requiring large amounts of control data (such as Spatial Sound Synthesis [196]), the organic, multi-dimensional nature of gestures combined with the extended mapping possibilities in CAC are promising directions.

Acknowledgements

We would like to thank Kristian Nymoen and Sophie Breton for sharing data of their performances. Thanks to Sean Ferguson for insightful discussions and to Isabelle van Grimde for permission to use excerpts of her choreography. The first author would also like to thank Jean Bresson for advice with conceptual and implementation-specific details of the library. This research was partially funded by an NSERC discovery grant to the second author.

Chapter 4

Ab-Tasten: Atomic Sound Modeling with a Computer-controlled Grand Piano

The following chapter is in press as:

Schumacher, M. (2016). Ab-Tasten: Atomic Sound Modeling with a Computer-controlled Grand Piano. In Bresson, J., Assayag, G. and Agon, C. (Eds.), The OM Composer's Book: Volume 3. Éditions Delatour France / IRCAM – Centre Pompidou.

We observe a fraction of the process, like hearing the vibration of a single string in an orchestra of supergiants. We know, but cannot grasp, that above and below, beyond the limits of perception or imagination, thousands and millions of simultaneous transformations are at work, interlinked like a musical score by mathematical counterpoint. It has been described as a symphony in geometry, but we lack the ears to hear it. Stanislav Lem, *Solaris*.

Abstract

This chapter describes concepts and techniques for the composition of the piece *Ab-Tasten* for computer-controlled grand piano and electronics. It will discuss some of the conceptual implications of sound representations for music creation and introduce a model for corpus-based atomic decomposition, which served for the composition of both acoustic and electronic

materials. The accurate control of timing and dynamics on the computer-controlled piano allowed me to compose in a continuum between instrumental writing and sound modeling, and exploit principles of auditory organization to create the illusion of spatial sound synthesis with an acoustic instrument.

4.1 Introduction

The piece *Ab-Tasten* for computer-controlled piano and electronics was commissioned in 2011 for a *live*@CIRMMT concert, dedicated to works reflecting the historic, cultural, and technological transformations of the *clavier*.[1] With a history of over four hundred years of development, its latest embodiment, the modern grand piano, is one of today's most versatile instruments. In an organological sense, it stands out as a hybrid instrument associated with different instrument families, based on its sound producing medium (vibrating string), excitation mechanism (striking action of hammers), or playing technique (keyboard interface). Indeed, composers have interpreted the piano in various ways, e.g. as percussion instrument (Béla Bartók, *Szabadban*), string instrument (Helmut Lachenmann, *Klangschatten – Mein Saitenspiel*), resonator (Luciano Berio, *Sequenza X*), and even as a predecessor of the electronic synthesizer (Karlheinz Stockhausen, *Klavierstücke XV–XIX*).

The piano's harmonic and polyphonic capabilities, together with its timbral richness (see e.g. [22]), enable it to conjure up sonorities evoking extramusical sounds, which has inspired generations of composers. Maurice Ravel and Claude Debussy, for instance, used idiomatic pianistic gestures (*arpeggi, glissandi, tremoli*) to describe the fluid, amorphous movements of water, e.g. in *Jeu d'eau* or *Reflets dans l'eau* [166]. Other notable examples are Olivier Messiaen's transcriptions of birdsongs (*Catalogue d'oiseaux*), and more recently, Peter Ablinger's resynthesis of human speech in *Quadraturen* [184].

The tradition of recreating the sonorities of concrete sources inspired me to develop a method that transcends the notions of sound (timbre) and symbolic organization, and to extend piano writing to the fine structure of sound itself. While digital sound synthesis has allowed us to craft virtually any sound material on a medium, the constraints given by the physics of instruments and human performance have made it more challenging to apply similar techniques in acoustic composition. Implementing such a concept requires a sound model that considers the timbral and physical characteristics of the instrument, as well as a degree of control and accuracy that exceeds the limits of human performance. The concert took place in the Multimedia Room (MMR) of the

[1] http://www.cirmmt.org/activities/live-cirmmt/clavisphere/

Schulich School of Music of McGill University, which is equipped with a Yamaha DCFX Mark IV Disklavier (a MIDI-compatible concert grand piano). The precise polyphonic control of pitch, dynamics, and timing possible with this instrument seemed well-suited to pursuing the idea of sound composition with a physical instrument. Accordingly, the piece has no human performer (hence no symbolic score) and can be considered a fixed media piece for robotic instrument and electronics.

This chapter will describe two compositional concepts developed for the piece, realized in OM. The first is a corpus-based, atomic sound model that allows writing music in a continuum from abstract musical materials to modeling of concrete sound. The second is an approach for transferring concepts of spatial sound synthesis to an electroacoustic setting using principles of auditory organization.

4.2 Abstract Sound Representations

Using computer technologies, any sound can be captured as a series of samples on a digital medium. Data and structures can be extracted via analysis techniques to build higher-level sound descriptions, which can then be integrated into symbolic compositional processes. Such descriptions may serve for the creation of musical materials, but also inspire compositional thinking and lead to a rich dialectic between symbolic and sonic processes (see for instance the many examples in [40]). We should remind ourselves, however, that extracting meaningful information from the modulations of a sound wave is a non-trivial task. Indeed, a given description of a sound is conditioned by the assumptions of the underlying model used for analysis. This choice determines which aspects of sound are considered meaningful and exposed to the foreground versus aspects that are regarded as less relevant (and rendered implicitly or possibly not at all) [37]. Consequently, the information retrieved from a sound is subject to different interpretations, each deriving from the structure imposed by the underlying assumptions.

In musical contexts, representations based on the short-time Fourier transform (STFT) or wavelet transform are popular examples, using time and frequency/scale as two dimensions (reminiscent of a musical score). These representations are typically agnostic with regards to the content of the sound to be modeled, presuming that the salient aspects of a sound can be modeled via expansions of a single frame of functions. When the frame of functions fits the structure of the sound well, a meaningful representation results; inversely, discrepancies can obfuscate or distort the information (e.g. representing noise or transients via sinusoidal components). In addition to the consequences related to the fidelity of describing different sound characteristics, every model puts forward its structural characteristics. An additive model, for example, describes sound as

a superposition of homogenous sinusoidal partials, whereas a granular model describes sound as a succession of short-duration sound quanta. In an abstract sense, this could also be seen as supporting a compositional preference for simultaneous (vertical) vs. sequential (horizontal) organization. Thus, each model provides a particular viewpoint defining a framework of possible operations. A further consideration in the context of instrumental composition and transcription concerns the abstract nature of these sound representations. Their smallest structural elements (i.e. the frame functions) are typically based on mathematical objects (sinusoids, wavelets, etc.), which are on the one hand not a perfect match for acoustic sounds encountered in the physical world (such as instrumental timbres), and on the other not universal or objective, since they depend on many analysis parameters (resolution, windowing, etc.) and emphasize specific characteristics and structures, independently of and possibly far from the nature of the sound or compositional context.

An alternative category of sound representations, that aim to adapt to different sound characteristics using heterogeneous sound elements, are *dictionary-based models*. Widely used in signal processing applications (e.g. compression, in-painting, denoising) they decompose sound into a linear combination of elementary waveforms (called "atoms") contained in an analysis dictionary. Rather than using a single frame function, a dictionary may contain atoms of variable duration, bandwidth, spectral content, etc. and is thus capable of associating atoms that ideally match different sound characteristics. Once a dictionary has been defined, techniques for finding a sparse combination of atoms (i.e. with the least number of elements) can be used, such as *Matching Pursuit* (MP [135]). MP is an iterative algorithm that aims at finding a combination of atoms best to approximate a sound using a greedy search strategy (i.e. choosing at each iteration the atom that best matches the residual part of the sound). The temporal structure is specified via temporal locations in the target sound which correspond to possible positions of atoms in the search procedure. In simplified terms, the structure of the algorithm can be described as follows:

- For each temporal location, calculate a correlation coefficient of each atom with the corresponding segment in the signal (i.e. starting at the temporal location and lasting for the duration of the atom);
- Select the atom with the highest global coefficient, store it as part of the model and subtract it from the signal (leaving a "residual");
- Repeat this process on the residual until a breaking condition is met.

This process results in a model, i.e. a linear combination of atoms over time, and the remaining residual (the difference between the target sound and the model). Since for each iteration the

globally best matching atom (over all possible temporal locations) is selected, a model is built from the most significant to least significant element (in terms of energy), rather than from beginning to end of the target sound (in terms of time). Consequently, the final selection of atoms as well as their temporal positions in the model are determined in the matching process. A dictionary-based model can be further analyzed to derive higher-level structures (similar to partials derived from STFTs). For instance, atoms can be grouped based on their own parameters (e.g. amplitude or duration) or depending on their context (e.g. proximity of atoms in time or frequency). See also [209] for an overview of dictionary-based methods for analysis, visualization, and transformation of audio signals.

4.3 Corpus-Based Atomic Decomposition

The composition of *Ab-Tasten* was the incentive for my software developments in sound representations based on Corpus-based Atomic Decomposition (CBAD). Rather than using abstract signals as in signal processing applications (short-duration waveforms), in CBAD a dictionary is built from a collection of arbitrary sound files, also referred to as a *corpus*. These sound files are typically concrete sound objects themselves (i.e. with a complex spectral morphology) and constitute the smallest structural elements of the sound representation. As in the case of dictionaries containing abstract signals, a matching pursuit algorithm is used to find a combination of atoms that best approximates a target sound. This approach establishes a direct relationship between the sounds in the dictionary and the sound to be modeled, and can be thought of as representing a sound as a polyphonic organization of other sounds. Because in musical contexts we are not necessarily aiming to find a sparse representation that eventually converges with the target sound, there are no constraints on the temporal structure, contents of the dictionary, or cardinality (number of atoms) of the model. Instead, these specifications become a compositional choice. Atoms can be indexed and tagged, and can thus be assigned arbitrary meanings, such as a note, instrumental gesture, or abstract symbol – which allows us to think of the dictionary as a kind of musical vocabulary. Compared to other sound representations, CBAD has a number of interesting characteristics for compositional applications:

- It extracts polyphonic, temporal structures;
- It is a non-uniform representation using arbitrary collections of sounds;
- It is an iterative approximation of variable resolution, i.e. it is possible to control the perceptual similarity of target and model;

- It leaves a residual sound that is complementary to the model (mixing the model and the residual perfectly reconstructs the target sound);
- It permits creating multiple models from the same target using different dictionaries (parallel), or consecutive decompositions by using the residual sound as a new target which can be modeled with a different dictionary and so forth (serial);
- The selection of atoms from the dictionary, as well as their horizontal and vertical organization in the model, are determined by the matching pursuit algorithm, which can be an interesting creative resource.

The use of a target sound as a template for sound synthesis based on collections of concrete sound material relates to a number of techniques, such as audio mosaicing [240], adaptive micromontage [208], or data-driven concatenative sound synthesis [104].[2] Although these systems offer powerful sound analysis/synthesis techniques, few of them provide possibilities for linking the models to other compositional processes and symbolic representations. One notable example is the work by Einbond *et al.* using the CataRT system for feature-based transcription of audio targets in OM [80]. This representation is based on a Euclidean vector space of audio descriptors populated with atoms (or *units*), which are selected based on their proximity to a target position. One difference between this approach and CBAD is that its k-nearest-neighbour matching does not iterate over a residual in an out-of-time context and thus has no model for polyphony or temporality.

4.3.1 Analogies to Visual Arts

The concept of modeling larger-scale forms as a combination of concrete, smaller-scale objects can be seen in various manifestations in the visual arts: an early example is the work of 16th century painter Giuseppe Arcimboldo, who created portraits that were made entirely of recognizable smaller-scale objects, such as fruits, flowers, fish, etc. These objects, often symbolizing an underlying theme, are combined in such a way that their visual features (colour, shape, etc.) create the emerging perception of a larger-scale form. While artists in the pre-digital age carried this work out manually, today there are computer technologies for approximating image targets. Robert Silver, for instance, patented a computer algorithm that selects and combines images from a database for creating "photomosaics" [201]. The artistic interest in these techniques lies not in creating an exact reproduction (*facsimile*), but rather in the addition of a semantic layer, by which the characteristics of both the object to be modeled and the smaller elements used for the

[2] In another perspective, the possibility of modeling a sound in several stages with different sound elements shares similarities with spectral modeling synthesis [199].

modeling are retained. The emerging appearance of the larger-scale form can also be considered an intended illusion, exploiting principles of perceptual organization described in the theories of Gestalt psychology [82]. Figure 4.1 shows a number of works by Arcimboldo, Dalí, Pras, and Silver, each using a distinct technique.

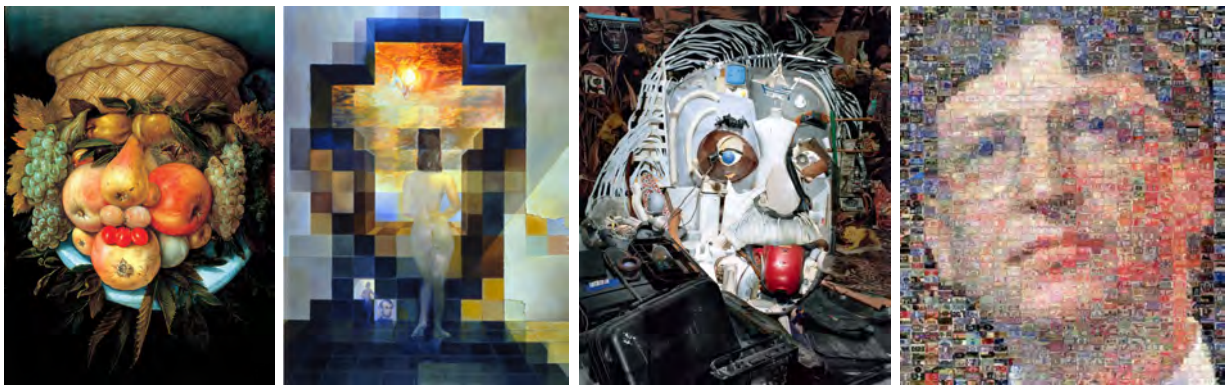

Fig. 4.1: Visual artworks by Arcimboldo, Dalí, Pras, Silver. Note the simultaneous perception of local objects and larger-scale forms.

The visual analogy lends itself well to illustrating the notion of different modes of perception as a function of context and resolution: in the case of a target image that is modelled with only a few smaller-scale images, it will hardly be possible to recognize the underlying larger-scale form. Instead, the characteristics of the smaller-scale images themselves and the relationships between them (e.g. relative position, organization) are perceived as the content of the image. In the opposite case, with a high number and density of smaller-scale images (as in a photomosaic), their cumulative features become perceptually more salient and perception shifts to a more holistic mode, focusing on macro-scale characteristics. At this resolution, details of the smaller-scale elements will be barely visible, shifting the mode of perception from recognition of individual objects to qualities of the "fabrics" of the modeled image (think of individual threads in a carpet).

A well-known example for a similar effect in the musical domain is György Ligeti's orchestral work *Atmosphères*, in which the dense superposition of instrumental parts shifts perception from recognition of individual notes and patterns to the textural qualities of a larger-scale morphology. It is hardly possible to follow the characteristics of individual instruments, which in this context do not carry individual musical meaning themselves but rather become properties of a global timbre (a whole that is other than the sum of its parts).

4.3.2 Building a Dictionary of Piano Sounds

In *Ab-Tasten*, the first step for modeling of target sounds was the creation of a dictionary that is representative of the grand piano. Simply stated, this means recording individual piano notes from the acoustic instrument to be used as atoms, which represent the smallest indivisible elements of the model. Sampling a keyboard instrument such as the piano is comparatively straightforward due to its percussive sound production and structure of discrete keys with orthogonal dimensions of pitch and dynamics. Since the sound production of the Disklavier can be controlled via MIDI, this sampling process could be automated via a computer program that triggered individual notes and recorded the sounds with a microphone. Sampling the entire combinatoric space of possible MIDI key numbers and velocities would require recording 11176 (88×127) samples, which at an average duration of 10 seconds would correspond to a corpus of about 31 hours of sound. Atomic decomposition with such a large dictionary would be impractical in terms of computing time. Initial tests with the Disklavier revealed that sampling the keys at 127 individual MIDI velocities was unnecessary, as the JND (just-noticeable difference) for dynamics was at a value of about 4. Moreover, MIDI velocities greater than 120 sounded unnaturally harsh, while values below 20 would not always accelerate the piano hammers enough to reach the string and produce a sound. Thus, the 88 keys of the piano were sampled at 25 distinct MIDI velocities (in the range of 20 to 120 in steps of 4), resulting in a total of 2200 individual recordings. Since the decay of a piano note varies with its pitch the duration of the notes was fixed to 15 seconds and the recording trimmed off earlier if the rms amplitude fell below a threshold of -24dB. The program for carrying out this sampling process was realized in OpenMusic using the library OM-SoX, which provides a suite of objects and functions for audio recording and processing [194]. Figure 4.2 shows the OpenMusic patch sending MIDI events to the grand piano for triggering individual notes and recording the resulting acoustic sounds as audio files.

On the top left of this figure we see the generation of the lists for MIDI key numbers and velocities, a number specifying the maximum duration of the recording in seconds, and a string used as prefix for naming the files (a). Visible on the top right is the outer loop (*sample-robot*), which iterates over the list of key numbers (b). On the bottom left we see the inner loop, which iterates over the list of MIDI velocities and generates unique filenames (c). The abstraction *sox-samplebot* (d) performs the sampling: the function *sox-process* starts the audio recording. After 0.4 seconds a MIDI "note on" event is sent, and the program sleeps for 15 seconds. In the meantime, the function *sox-trimsilence* trims off "silence" (i.e. audio with an RMS amplitude value below -24dB) from the beginning and end of the recording, before the "note off" event is sent. The resulting sound is written to disk as an audio file with a unique name.

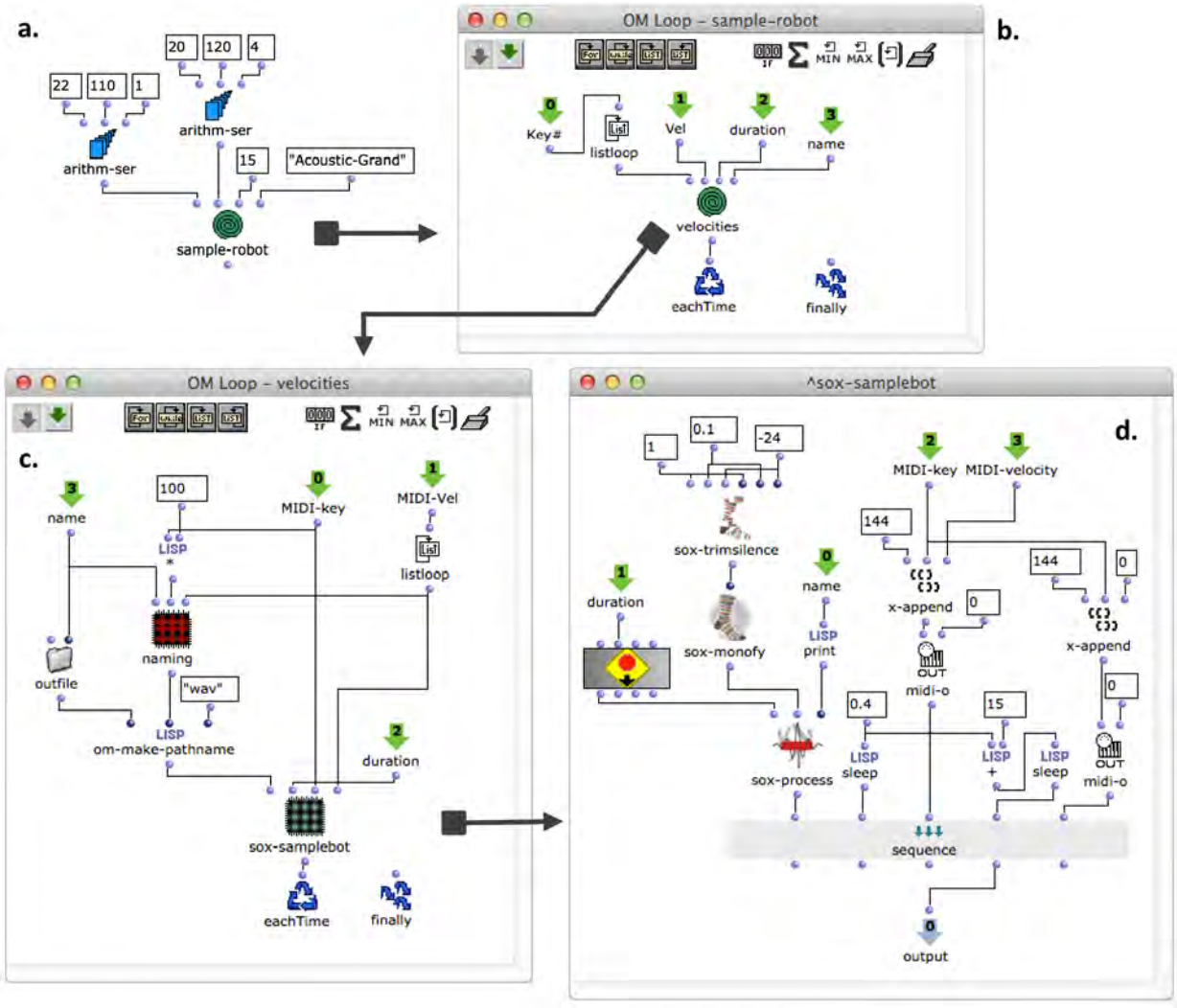

Fig. 4.2: The patch used for automated sampling of acoustic piano sounds.

4.3.3 The library OM-Pursuit

The functionalities for CBAD were implemented as an external OpenMusic library, titled OM-Pursuit[3] [193]. This library wraps signal processing functionalities in pydbm [29] and uses the Sound Description Interchange Format (SDIF) as a container for storage and interchange of data describing the model and the dictionary. OM-Pursuit implements a number of classes for representing atomic sound models and a set of functions for processing them. These models can

[3]This text describes the library at the time of composing *Ab-Tasten* (2011). It has since then been further developed and extended with spectral audio descriptors and a constraint programming system.

be used for audio synthesis or converted into different structures to be manipulated and eventually transcribed into a symbolic score. Figure 4.3 shows an example of an OpenMusic patch used for creating a corpus-based atomic model of a sound target.

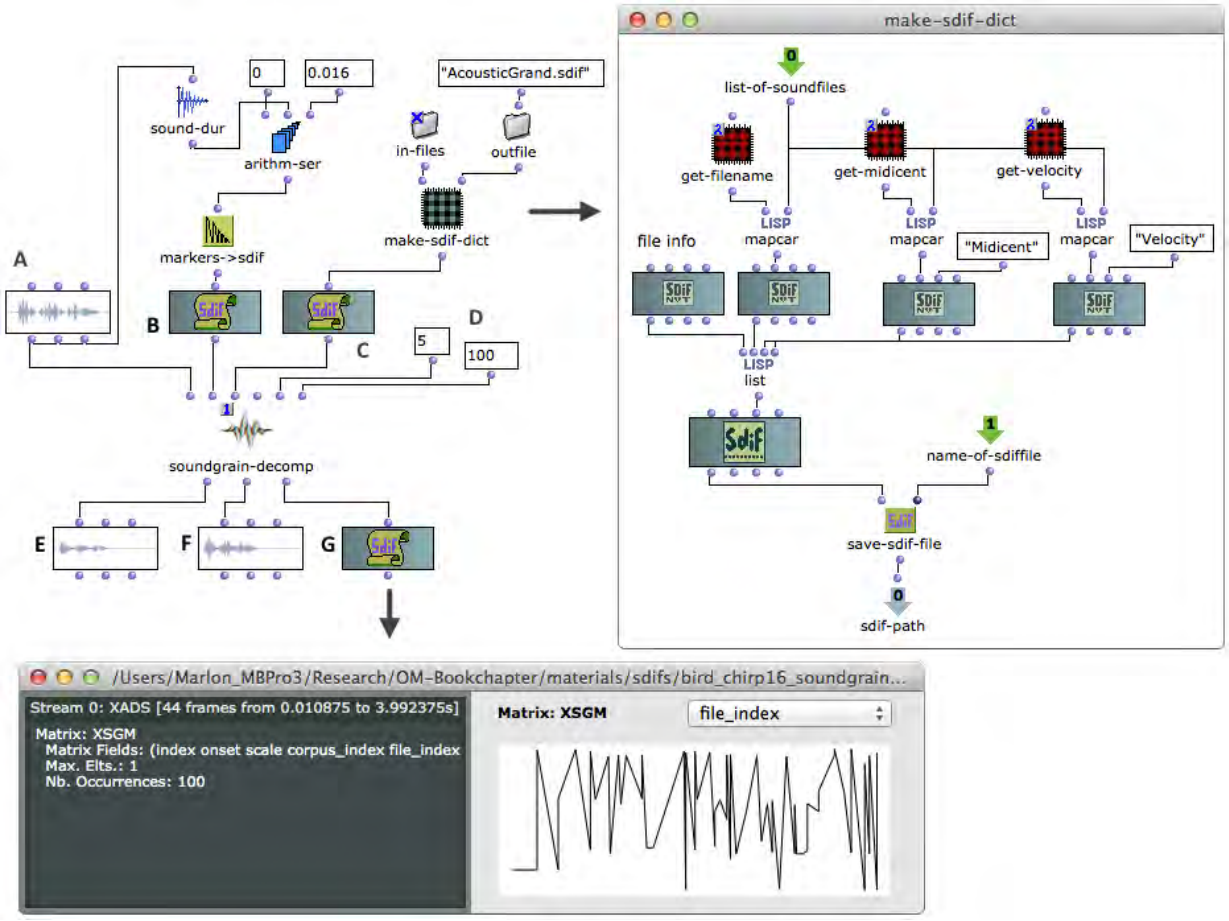

Fig. 4.3: A patch illustrating atomic sound modeling with the library OM-Pursuit.

The patch contains a target sound (*sound* object) (A), the temporal locations calculated as an arithmetic series with an interval of 16 milliseconds (stored in an *SDIFfile* object) (B), the dictionary generated via the abstraction *make-sdif-dict* (stored in another *SDIFfile* object) (C), and parameters for the decomposition (maximum number of simultaneous atoms and total number of atoms) (D). Three types of information are stored in the dictionary for each sound file (top right in the figure): the file path, pitch in midicents, and MIDI velocity. The function *soundgrain-decomp* then takes these data, carries out the matching process, and returns three values: a sound file of the audio synthesis of the model (E), a sound file of the residual (F), and an SDIF file containing a parametric description of the model (G). At the bottom we can see

OpenMusic's SDIF editor, which displays information about the model; each atom is represented as an array with fields for onset time, duration, magnitude and norm (amplitude), corpus index, file index, and file path. This parametric description (G) can be converted into OpenMusic objects and integrated into compositional processes like any other musical data. Figure 4.4 shows an OpenMusic patch containing objects and functions for representing and manipulating a model.

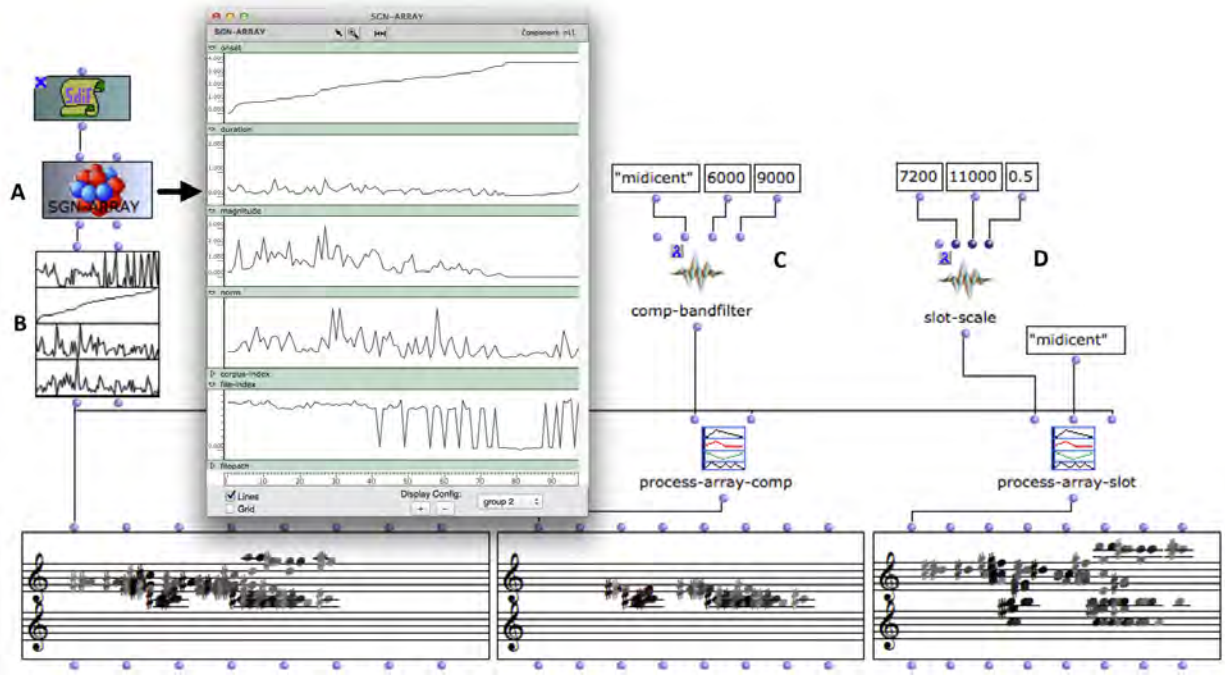

Fig. 4.4: Left (A, B): Conversion from *SDIFfile* to *sgn-array* and *score-array*. Right (C, D): two examples for vertical and horizontal processing before converting to *chord-seq* objects.

The *SDIFfile* object at the top left (containing the model) is converted to a *sgn-array* object, a tabulated structure in which columns represent individual atoms (in OM-Pursuit referred to as "soundgrains") and rows represent different parameters of the atoms (A). This object can be directly used to drive audio synthesis or spatialization processes, e.g. using the libraries OM-SoX or OMPrisma [196]. Since the *sgn-array* object internally stores pitch and velocity information for the individual soundgrains, it can also be converted into a *score-array* object, a similar structure, in which columns represent MIDI notes (B). This *score-array* object can be directly converted into a *chord-seq* object (visible at the bottom left of the figure), and eventually exported to a MIDI file: a "score" which can be performed by the computer-controlled piano. OM-Pursuit includes two higher-order functions which allow connecting a LISP function or

patch in lambda mode for vertical and horizontal manipulation of these objects: *process-array-comp* iterates over columns, e.g. for filtering soundgrains that fall outside a specified pitch range (C), and *process-array-slot* allows selecting a row to perform global processing, such as rescaling of a parameter for the entire object (D). Figure 4.5 illustrates the process of CBAD in OM-PURSUIT and the possible representations and renderings of the model.

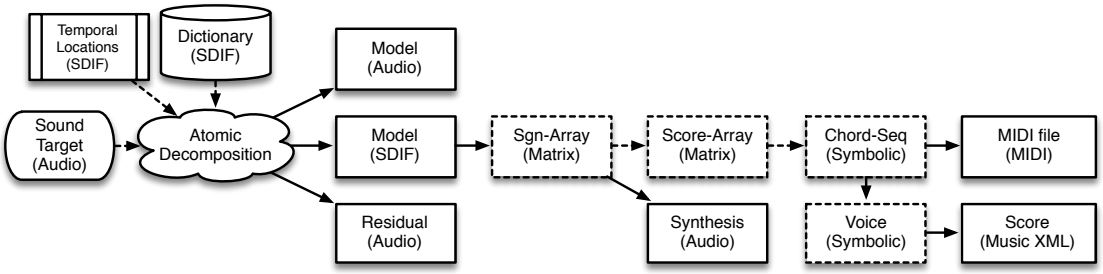

Fig. 4.5: Corpus-based Atomic Decomposition in OM-PURSUIT.

4.3.4 Modeling a Birdsong with an Acoustic Grand Piano

Let us look at a concrete example of the creation of musical materials for the piece. Figure 4.6 shows a sonogram (LPC analysis) of an excerpt of the birdsong that served as a source for developing most of the materials. Besides being an hommage to Messiaen, the sound was chosen because of its inherent musical qualities including its rhythmic and motivic structure. Visible as vertical lines in the sonogram are the markers used to specify possible temporal locations for atoms, defined as an evenly spaced grid with an interval of 16 milliseconds (256 audio samples at 16 kHz sampling rate). Below the sonogram we can see three horizontally aligned models whose parameters for the decomposition are identical except for the number of iterations (10, 50, and 850, respectively). The grayscale values in the *chord-seq* objects represent dynamics (as in the sonogram). This example demonstrates the perceptual shift discussed before as a function of density and resolution; for the first model consisting of 10 iterations (10 piano notes) it will be hardly possible to recognize an underlying target sound. Instead, we perceive an abstract musical phrase in which each of the individual piano notes are clearly identifiable. At 850 iterations, in contrast, the same piano notes become micro-scale elements, integrated into a cloud of sounds whose global features (granularity, pitch contour, etc.) become perceptually salient. Instead of the relationships between individual notes we perceive an emergent form on a larger scale.

Once converted into a symbolic representation, models of the birdsong were then further processed (e.g. quantized, filtered) to create musical materials for the piece. For the development of these materials I often switched between alternative representations (e.g. continuous vs. metric

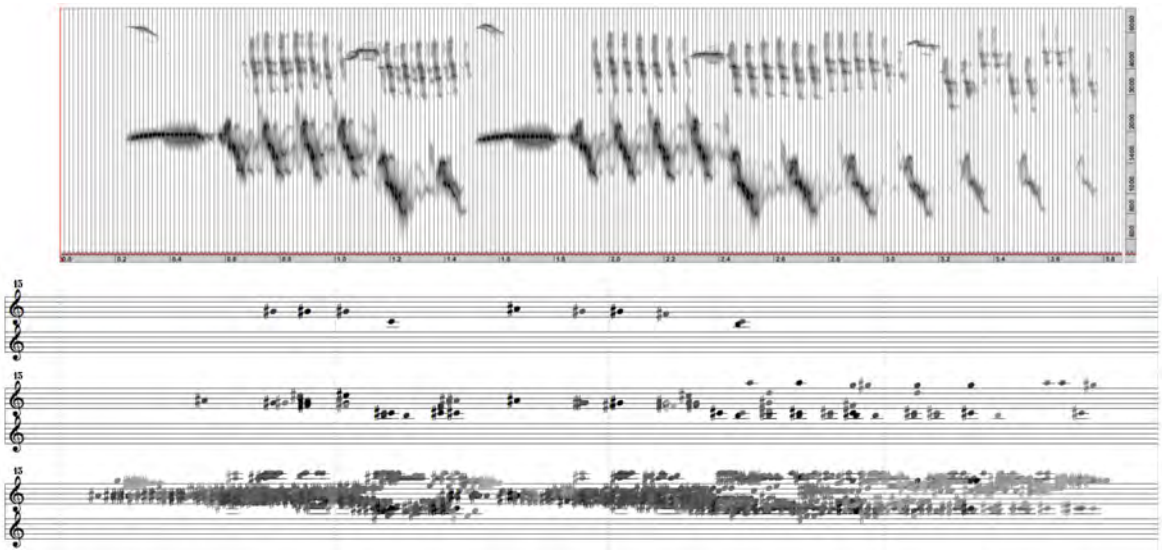

Fig. 4.6: Top: Sonogram of birdsong. Bottom: Three models with 10, 50, 850 atoms, respectively. Grayscale represents dynamics. Note how the matching process determines polyphony and temporal structure.

time), as each offers different possibilities for manipulation. For instance, in some situations the model was treated as a continuous signal using sound-processing techniques (filtering, stretching, resampling), while in other situations the model was treated as a symbolic musical structure and developed based on intervallic relationships, harmony, rhythm, etc. Figure 4.7 shows a model of the complete birdsong (4 phrases) in the *chord-seq* object and its quantification via the function *omquantify* into a metric representation in a *voice* object. This is the main theme of the piece as it appears on the acoustic piano at 1'20".

4.4 From Virtual Ensemble to Meta-Instrument

The second part of this chapter is dedicated to the relationship between acoustic instrument and electronics, as well as the approach to spatialization in this piece. I was interested in using electronics as spectral and spatial augmentations of the piano, almost indiscernible from the acoustic instrument, to immerse the listener in a soundscape of electroacoustic piano sonorities. Inspired by the notion of different perceptual modes discussed before, I was aiming to develop a dialectic between the fusion of acoustic and electronic elements into single sound source (akin to a "meta-instrument") and the splitting apart into individual musical identities, similar to a chamber music setting ("virtual ensemble").

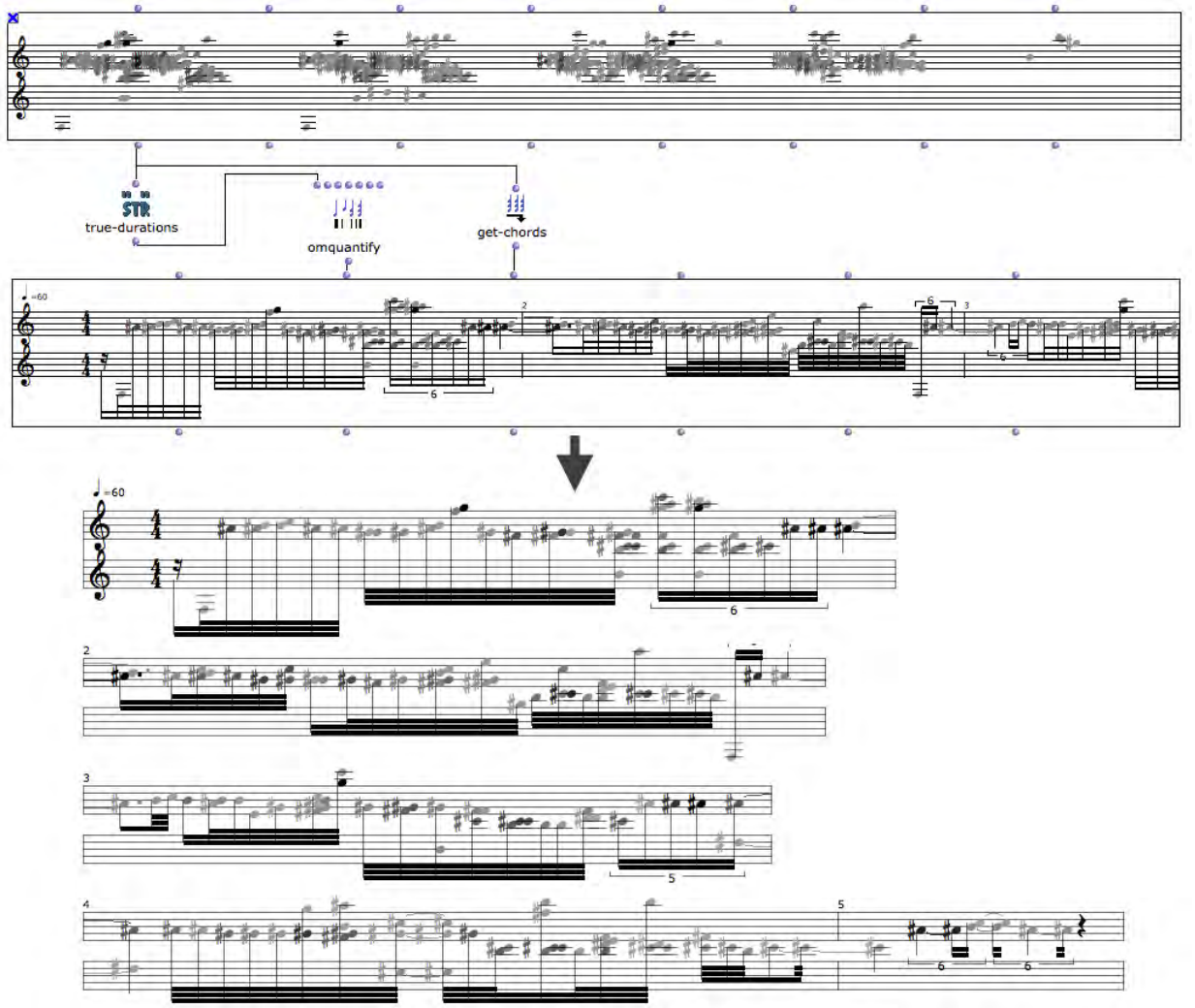

Fig. 4.7: Piano model of the birdsong as a *chord-seq* object (top) and its metric representation in a *voice* object (bottom).

An interesting feature of CBAD is the possibility of decomposing a sound using a mixed dictionary containing several *corpora* (sound collections). From the resulting model, the elements from the respective corpora can be extracted and organized into individual structures, similar to instrumental parts in an orchestral score. Using a dictionary containing corpora for instrumental and electronic elements (e.g. processed or synthetic sounds), it is possible to develop both types of materials as complementary parts in an integrated approach, through the same formalism.

4.4.1 Electronics as Microtonal Augmentation

The idea of fusion and segregation of instrumental and electronic parts was realized by conceiving the electronics as an ensemble of four "virtual pianos", represented as microtonal transpositions of the acoustic grand. To that end, four new sound corpora were created, representing each of the virtual pianos. The original recordings of the acoustic piano were copied and transposed upwards by 20, 40, 60, and 80 midicents respectively, resulting in a total of five sound corpora in 12-tone equal temperament (12TET), each "tuned" 1/10th tone higher. These sound corpora could then be combined together into a mixed dictionary with a resolution of 60 pitches per octave (60TET). From the model created with this mixed dictionary, the pitches corresponding to the acoustic and virtual pianos can be extracted and assigned to the respective parts. Depending on the perceptual properties of the resulting musical structure, these individual parts (corresponding to individual pianos) may be integrated into a single auditory stream (comparable to a musical "voice"), or segregated into individual voices, according to principles of auditory organization [31]. Figure 4.8 shows three examples to illustrate this effect.

The *chord-seq* object on the top left shows an ascending scale over one octave in 60TET (A). The function *micro=> multi* parses the corresponding pitches into five microtonal parts (staves) in semitone resolution. In this example, the notes are closer in pitch and time between adjacent parts than within the same part, which creates the perception of a single auditory stream that is alternating between the parts. The middle example (B) shows a chord with pitches corresponding to the frequencies of the first 30 partials of a harmonic spectrum (quantized to 60TET resolution), which are distributed to the five microtonal parts. Here, the harmonic structure of the pitches and their simultaneous onset create the perceptual fusion of the individual parts into a single auditory object. On the right (C) we see eight chords in rapid succession consisting of three to eight random pitches between 3600 and 8600 midicents that are quantized to 60TET and distributed to the individual piano parts. This creates an ambiguous situation in which cues for sequential and simultaneous grouping compete with each other, and where other cognitive processes –such as attention and expectation– can influence the forming of auditory streams.

4.4.2 Fusing Performance and Listening Spaces

An important aspect of the relationship between instrument and electronics in this piece is the development of the approach to spatialization and the transfer of concepts of spatial sound synthesis to an electroacoustic setting. Historically, piano music was performed in intimate settings, often for a small audience located around the instrument. Each of the listeners had an individual listening position (see e.g. Josef Danhauser's famous painting

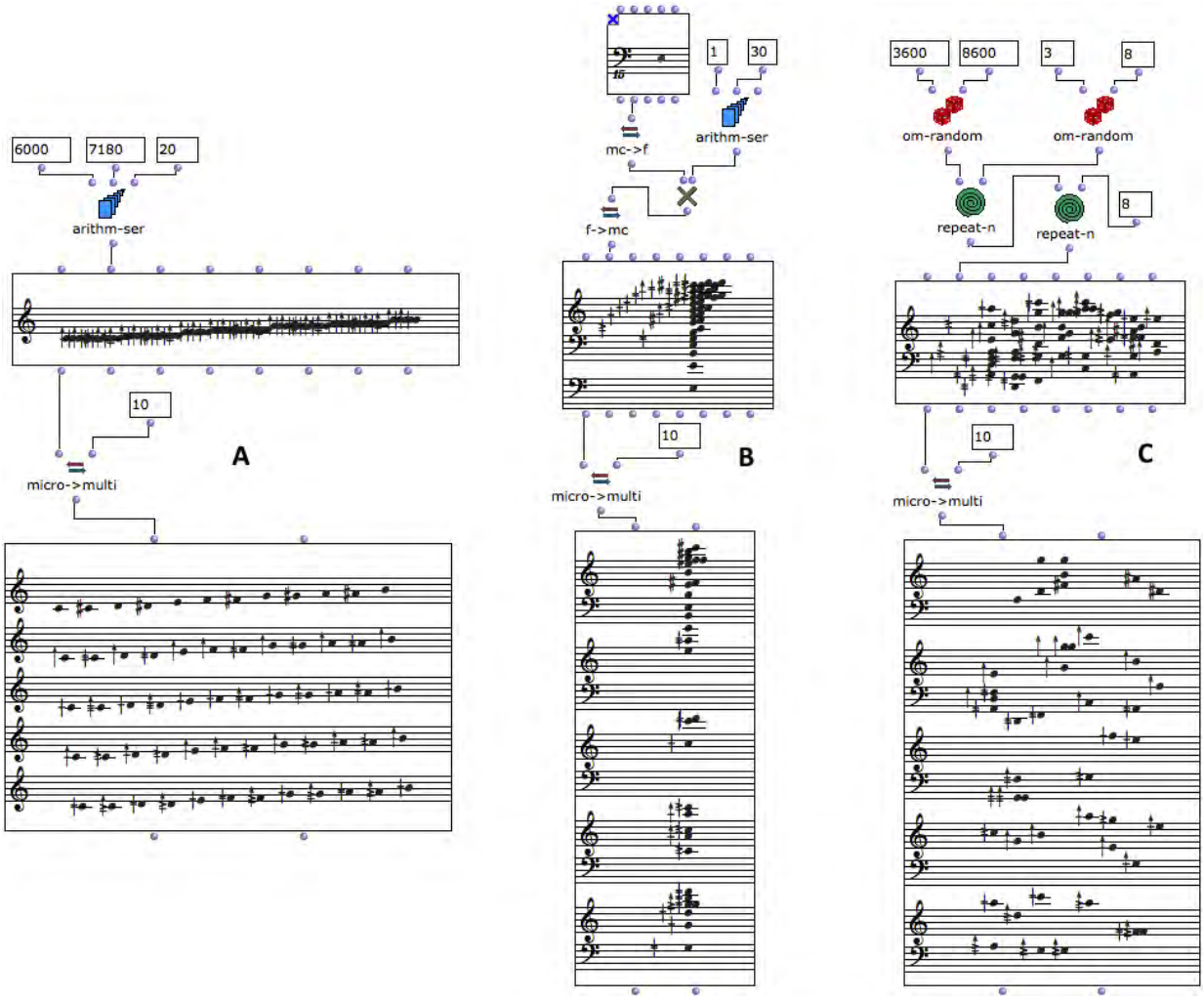

Fig. 4.8: Examples of musical structures in 60TET (*chord-seq* objects) dispatched into five distinct parts in 12TET (*multi-seq* objects).

Liszt am Flügel phantasierend). Stockhausen described this situation as "comparable with people today who wear headphones and completely immerse themselves in the music" [206]. Inspired by these reflections, my aim was to recreate such an immersive listening situation in an electroacoustic context: each audience member would have her personal perspective and experience of the piece, comparable to viewing a physical artwork from different angles. This required the instrument to be positioned close to the audience and developing an approach that offers heterogeneous, equally privileged listening positions, rather than an idealized "projection" of the music which is degraded for the majority of listeners.

To realize this idea, an unconventional setup was used: the acoustic grand piano was positioned at the centre of the hall. The audience was seated around it, turned towards the instrument. Four loudspeakers were placed around the audience, at the corners of the hall. Each virtual piano part was projected from a dedicated loudspeaker, which avoided the use of phantom sources and thus the distortion of the spatial image depending on listener position. The result was an intimate listening situation with individual auditory perspectives, fusing performance space and listening space. This setup is shown in Figure 4.9.

The spatial disposition and the assignment of micro-tuned chromatic scales to different loudspeakers result in a morphological correlation between pitch and spatial position. If we recall the microtonal structures from Figure 4.8, it can be seen how adjacent microtonal pitches translate to adjacent positions in space, whereas vertical pitch structures (chords/harmonies) result in spatial constellations. Each listening position provides an individual auditory perspective for an acoustic situation in which instrumental sounds emanating from the centre are complemented by microtonal and spatial extensions in the electronics.

4.4.3 Spatial Sound Synthesis as Perceptual Experience

The spatialization of individual piano notes, which correspond to the components of a sound model, can be related to the concept of *spatial sound synthesis*, first described in [196]. In simplified terms, the idea of spatial sound synthesis is to consider spatialization as a parameter of a sound synthesis process. Rather than spatializing pre-existing sounds and conceptualizing musical space in terms of spatial sound scenes (based on the model of sound sources in a physical space), spatial sound synthesis is an approach in which spatial perceptions are created through composition and synthesis of sound components, such as frequency-domain partials or time-domain grains, spatialized individually. When cleverly controlled, this allows for the creation of physically impossible sound sources and auditory illusions. Similar to how the synthesis of frequency structures, such as the individual partials of a tone complex, can be used to create the sensation of pitch [183], in spatial sound synthesis the individual spatialization of sound

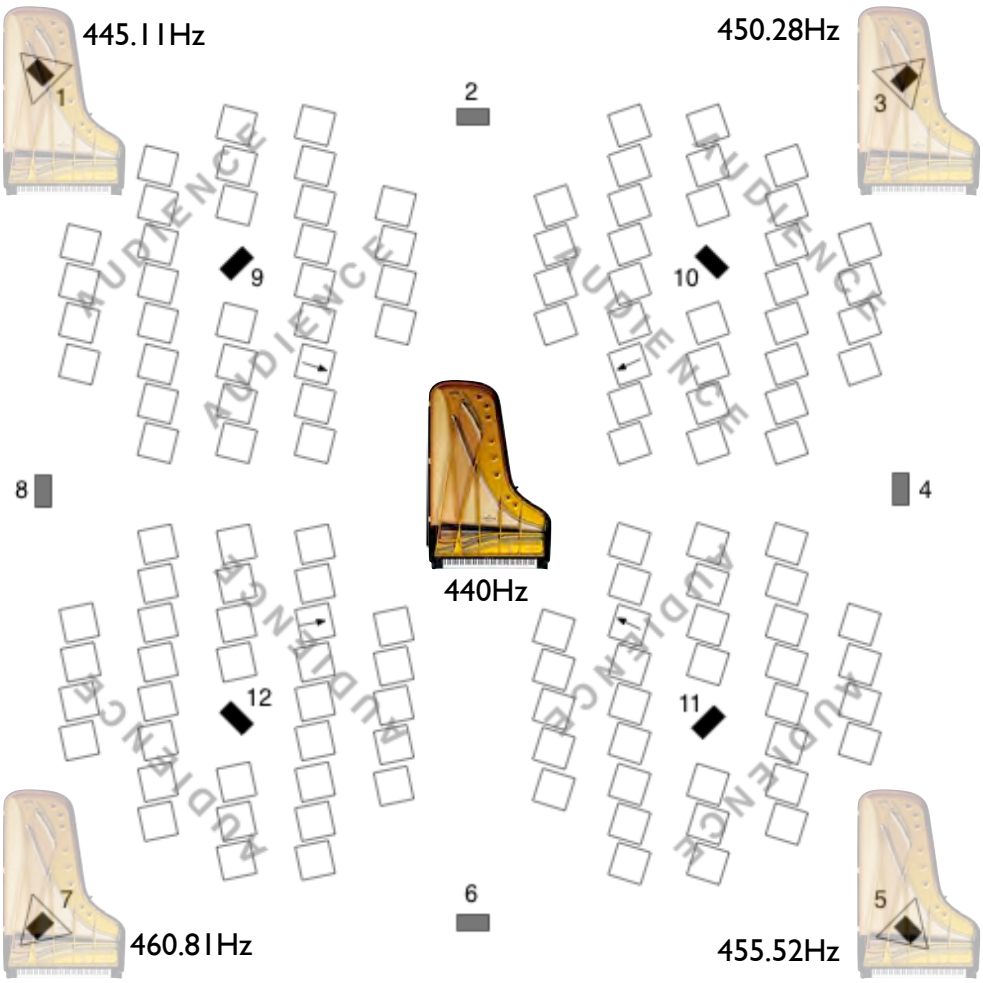

Fig. 4.9: Performance setup for the piece. Note the physical piano at the centre of the audience and the four loudspeakers at the corners of the hall representing the virtual pianos. The numbers next to the pianos indicate their diapason in Hertz. (The speakers placed in between the audience members were used for other electronic parts which are not discussed in this article.)

components can give rise to the perception of spatial auditory objects.

Indeed, auditory perception can be described as a heuristic process evaluating perceptual criteria to build a meaningful mental representation of an auditory scene [31]. Spatial cues, such as time- and level-differences between ear signals, are part of these criteria, but can be overridden by other perceptual cues. In the case of incomplete or paradoxical auditory stimuli the perceptual system follows a "best guess" strategy in which certain cues can dominate others in favour of the most plausible interpretation [191]. This can result in auditory conflicts and illusions,

such as the "*Glissando* Illusion" described by Deutsch [73]. In this experiment, a continuously upwards/downwards gliding sine tone is abruptly switched between left and right loudspeakers in a stereo setup. Despite the spatial discontinuity, most people perceive an uninterrupted single *glissando*, in which the spatial movement is correlated with the pitch movement. This illusion demonstrates that the Gestalt principles of continuity and proximity in pitch can override localization cues, producing the auditory illusion of a single source that is continuously moving in space.

Let us consider an excerpt from *Ab-Tasten* in which a similar auditory illusion is produced in a musical context. The sonogram of the birdsong from Figure 4.6 shows a number of rapidly descending *glissandi* (chirps). At 6'10" in the piece this birdsong appears in a dilated and resampled form; it was time-stretched by a factor of 15 (using a phase-vocoder) and the resulting sound file was then decomposed using a mixed dictionary of the five corpora of the different pianos. For the temporal locations, the same evenly spaced temporal grid (16 milliseconds) was used as in the examples shown in Figure 4.6. The birdsong was decomposed into 2500 atoms (piano notes), which were distributed into five parts for the respective pianos. This is another example of how resolution and scale changes the mode of perception: the short-time morphological features of the birdsong (e.g. pitch contour, amplitude envelope) are changing so slowly that they shift into the macro-scale of the score, while the decomposition at 15 times higher temporal resolution brings the micro-scale details in the fine structure of the sound to the foreground, akin to a "zoom" effect. Figure 4.10 shows an excerpt from the piece (6'47"–7'00") displayed in five *chord-seq* objects (representing the physical piano and the four virtual pianos). The rapid chirps from the original birdsong now resemble slowly modulating, granular *glissandi*, traversing the individual piano parts.

This perceptual situation produces a striking effect: although the individual piano sounds emanate from discrete spatial locations, their high repetition rate, as well as the proximity and continuity of pitch, create the perception of distinct streams of piano sounds that seem to move continuously through the hall and in between the acoustic and virtual pianos. The global impression of this sonority can be described as an amorphous granular sound texture in which individual sound sources seem to emerge, move through space and disappear again. This auditory illusion can be explained by the general preference of the perceptual system for organizing sound events in a way that yields the simplest interpretation of an auditory scene [191]. As in the *Glissando* Illusion, the sounds produced at the different physical locations are timbrally quasi identical, making it ambiguous whether sound events have been produced by distinct, static sound sources, or by a single source that has changed its position. As a result, the perceptual system prioritizes other cues for forming auditory streams. Rather than parsing the acoustic information

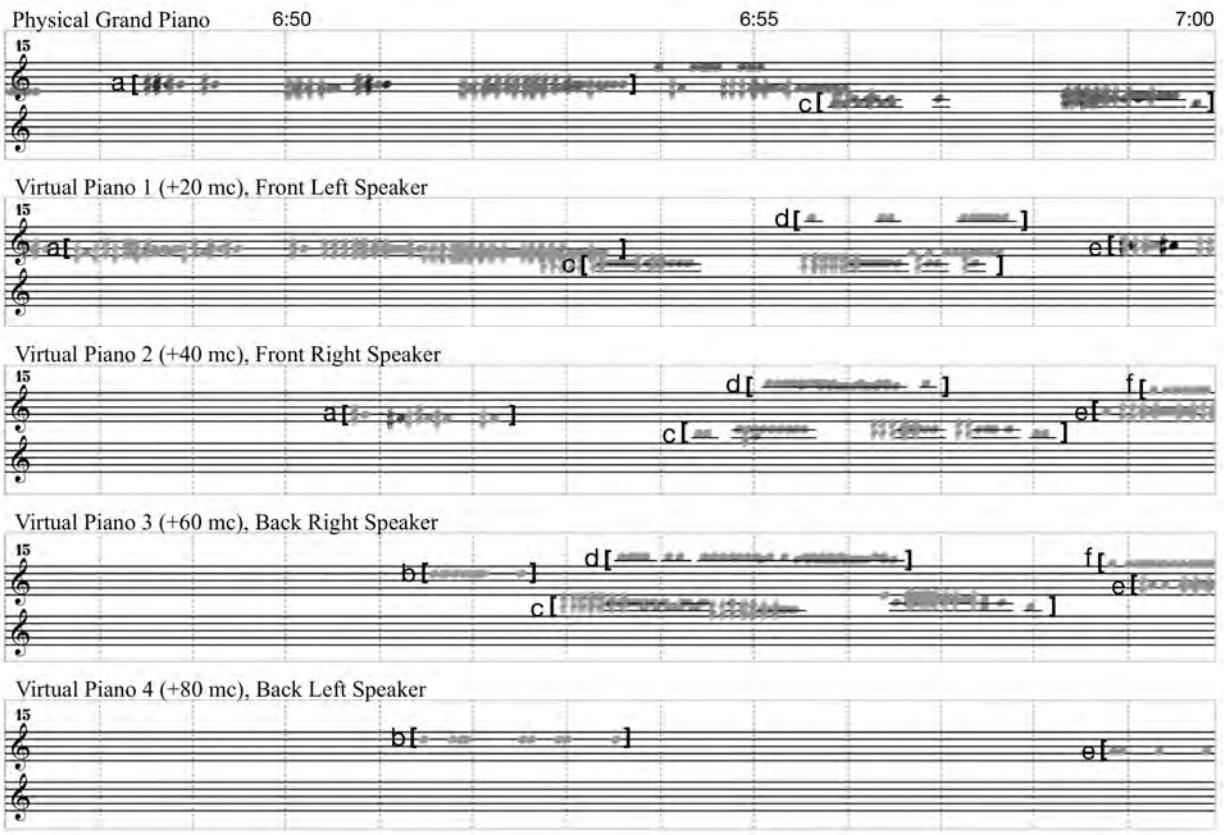

Fig. 4.10: Atomic model of a time-stretched birdsong displayed in 5 staves (*chord-seq* objects) representing the physical and virtual piano parts. The labelled brackets indicate individual *glissandi*.

in a consistent way that would create complicated temporal/harmonic patterns for each physical sound source, it trades off the misinterpretation of spatial cues in favor of forming more plausible, simple streams based on continuity and proximity in pitch and time. The resulting effect is an illusion of spatial movement correlated with pitch.

The perception of spatial depth and distance might be attributed to the similarity between the characteristic changes of sound features as a function of distance to the listener, and as an effect of dynamics on the piano. Indeed, studies of piano timbre have revealed semantic congruencies and similar pianistic performance strategies to express the labels "distant" (a spatial attribute), "dark", and "muddled" (timbral attributes) [22]. Two important perceptual cues to determine the distance of a sound source are its overall level (which decreases with distance due to the spreading of the wavefront) and relative high-frequency energy (which decreases with distance due to atmospheric absorption by water molecules) [238]. A similar change in sound characteristics

can be observed as a function of dynamics on the piano: the lower the velocity when striking a key, the lower will be the overall level as well as the relative high-frequency energy (spectral centroid) of the sound. These correlated spectral and level differences of piano tones might be interpreted by the perceptual system as distance cues, such as a *pp* note on the piano sounding more distant as compared with a *ff* note.[4]

4.5 Closing Remarks

In this chapter I discussed some of the conceptual implications of abstract sound representations as compositional models and introduced a corpus-based, atomic representation that establishes a direct link between a concrete sound phenomenon and a collection of sounds. Combined with the possibilities offered by the robotic grand piano, this model allows an integrated approach for the composition of materials for both acoustic and electronic parts by extracting structures at different resolutions and time scales, and with different sound corpora.

The electronics were conceived as virtual microtonal copies of the acoustic instrument, creating a perceptual ambiguity between acoustic and electronic sounds, between cumulative whole and constituent parts, and between reality and imagination. Using a specific spatial disposition of loudspeakers, acoustic instrument, and listeners, together with a morphological correlation between pitch and spatial location, it was possible to exploit principles of auditory organization to create the illusion of depth and spatial movement and transfer concepts of spatial sound synthesis to an electroacoustic setting. The libraries OM-Pursuit and OM-SoX provided the functionalities to realize artistic ideas that would have otherwise been difficult to achieve.

Although it might seem that CBAD is merely another analysis/synthesis technique, I believe its true potential lies in its conceptual and compositional implications. The process of atomic modeling raises interesting questions related to context, resolution, and scale as constructors of modes of perception and, consequently, semantics of materials. Similar to how spectral representations paved the way for new approaches, the atomic model offers an alternative paradigm that may inspire new directions for compositional thinking.

The German title *Ab-Tasten* has multiple meanings: its literal translation means “sampling”, as in sampling an audio signal. In this spelling, it also refers to the piano’s performance interface, meaning “removing the keys”, in the sense of a renewed interpretation of the piano as an acoustic synthesizer. Lastly, it can be interpreted as “touching”, “sensing”, or “exploring”, as in exploring unknown lands.

[4] A binaural recording of the premiere of *Ab-Tasten* is available online:
http://soundcloud.com/marleynoe/sets/ab-tasten.

Chapter 5

Conclusion

Technologies for sound spatialization, gesture control, and sound modelling provide powerful possibilities and potentials for artistic creation. Current tools, however, do not address the need of composers for high-level, musically-relevant interfaces and abstractions allowing to integrate these aspects into symbolic compositional formalisms and models. At the beginning of this research it was not practical or even feasible to formally compose with these aspects and develop ideas in a complexity similar to symbolic music materials. This research aims at overcoming the dissociation of powerful signal processing technologies on the one hand, and flexible programming environments for music compositon on the other, addressing the need for musical composition in terms of *écriture* (writing and musical thought) including spatial, gestural, and sound-related aspects. We argue that this integration is both relevant from a scientific perspective, for research in computer tools for modelling of compositional processes, and from an artistic one, by providing expanded possibilities, fostering the development of rich and novel compositional directions.

Our works are situated in the field of computer-aided composition which was originally conceived for the manipulation of symbolic musical materials. We have shown that the history of CAC environments can be seen as a development from closed systems implementing specific musical algorithms, to open and programmable systems, following a trend towards integration of other music-related data and media (cf. section 1.1). Notably, the develoment of frameworks for composition with sound signals has extended the scope of these environments from the symbolic level of music representations to the concrete level of digital sound, and has stimulated many original artistic approaches [40]. Unfortunately, the musical dimensions of space and gesture, have not yet been accounted for, and the available sound representations are not flexible enough to account for current compositional trends (cf. section 1.3). This thesis proposes solutions for computer-aided composition with space, gesture, and sound. The integration of these media

into composition environments is an ambitious task which demands a systematic approach and careful consideration of the particularities of each of these musical dimensions to adapt them to the context of "compositional modelling" [16]. Our design concepts and implementations for this integration have been described in the three manuscripts forming the core of this dissertation:

In Chapter 2 we described a library for the symbolic control of sound spatialization. The library is designed as an object-oriented system which provides an extendable collection of spatialization classes, separating rendering from description in accordance with the hierarchical model proposed by the SpatDIF standard [168]. This system overcomes the limitations of many current tools, by providing complete flexibility in terms of numbers of sources, spatialization method, and loudspeaker configuration. It thus enables composers to unite symbolic composition and sound spatialization in a complexity previously impossible. We presented an original system which allows to merge sound synthesis with spatialization classes into hybrid structures that can be controlled using high-level interfaces and algorithmic specifications. We introduced the concept of *spatial sound synthesis*, i.e. the composition with spatial attributes at arbitrary scales of the musical structure. The system thus enables composers to develop sophisticated spatialization processes, such as spectral [121] or granular spatialization [232], integrated with other high-level musical processes and materials in a compositional framework. In our recent works in this field we have presented solutions for generation and streaming of spatial sound description data [46] and new tools for *channel-based* spatialization approaches [95].

In Chapter 3 we presented a system for the integration of gesture data into CAC environments. This system represents gestures as multidimensional spatio-temporal morphologies of high-level descriptors in accordance with the GDIF concepts [114]. We described requirements for their integration as musical materials and presented tools for segmentation, representation, and processing. We discussed compositional limitations of the instrumental notion of "mapping" as an out-of-time connection between gesture variables and synthesis parameters and presented a flexible system which allows to generate musical objects of arbitrary scale through a mapping process expressed as a visual program. The system was validated through two example applications making use of real-world gesture recordings of a dance performance and a performance with a digital musical instrument.

In Chapter 4 we described a dictionary-based sound model which can be regarded as the complementary counterpart to existing sinusoidal models [209]. This model offers an alternative conception of sound, as a granular, mosaic-like structure, and enables composers to establish

a direct link between a target sound and the timbral vocabulary or instrumentation related to a specific work. The model is able to build compositional representations of noise-like and percussive sounds, which are of increasing relevance to contemporary compositional practice [58]. Our work allows to integrate these sound worlds into computer-aided composition models, which was difficult or impossible to achieve with existing tools. We showed that our system allows for flexible transcription of temporal morphologies of target sounds into instrumental scores, establishing a link between the domain of symbolic music (musique même) and the domain of concrete sounds (musique concrète). We discussed conceptual implications of this model and presented a professional artistic application for the composition of a piece for computer-controlled grand piano and electronics which would not have been possible without these tools.

5.1 Contribution

This thesis presented the conception, the design and applications of a structured software framework for the integration of spatial, gestural, and sound data into the computer-aided composition environment OpenMusic. This framework is implemented as three open-source libraries which employ the same data structures and interfaces (matrices). Aligned with design concepts of modern CAC systems, these libraries are implemented as programmable and extendable tools, rather than closed applications, and emphasize the interactive aspects of compositional modelling (duality of algorithmic and manual specification): for instance, the examples we have shown in chapter 2 for algorithmic vs. manual specification of trajectory data (figs. 2.1 and 2.11); in chapter 3 the alternative types of segmentation (fig. 3.2), and algorithmic vs. interactive processing of gesture data (fig 3.5); in chapter 4 the user-defind basis of a sound analysis system (dictionaries, fig. 4.2) and the possibilities for specification of temporal structures for the decomposition using algorithmic or manual specifications.

Our software framework is seamlessly embedded in OpenMusic, leveraging existing programming tools and interfaces. This provides composers with tools and concepts they are already familiar with which encourages their quick adaptation and integration into personal compositional frameworks. Each of the presented libraries provides a collection of readily-available functions, to facilitate processing and setting of these data. A listing of these functions can be found in Appendix B.

All three libraries are “open” systems, allowing for interchange of data using standardized descriptions (e.g. GDIF [114], SpatDIF [168]) and employ extensible formats (SDIF [198]), which are supported by an increasing number of computer music applications. Beyond the individual possibilities offered by each of these tools, our software design enables composers to relate spatial-,

gestural-, and sound-related aspects in a unified and coherent approach. This framework should provide composers with flexible tools and building blocks that can be easily integrated into individual compositional processes. To our knowledge there is currently no other system which allows for high-level control of spatialization, gesture, and sound description data embedded in a symbolic compositional framework.

5.2 Impact

Our software framework has already found numerous applications for research, artistic, and pedagogical purposes. Since OpenMusic is one of the most popular environments for CAC in use today,[1] the integration of the tools allows a large number of users to benefit from our developments. The library OMPrisma, for instance, is regarded as an essential part of the software package distributed through the IRCAM Forum, which gathers over 2000 users (artists, sound engineers, researchers, teachers) worldwide.[2] All of the software developments presented in this dissertation are cross-platform, free and open source, and can thus be built upon and extended by interested artists and researchers.

The results of our research have so far been published in a scientific journal [196], a book chapter [192], and several conference papers [195, 43, 46, 95]. In 2012, a project based on the OM-Pursuit library was awarded the CIRMMT prize for interdisciplinary excellence.[3] The software framework has been used internationally for a number of artistic/research projects at McGill University (Canada), IRCAM (France), and the Royal Institute of Technology (RITM) in Melbourne (Australia).

The usability and impact of this work to the artistic community is attested by the realization of numerous works by professional composers and artists around the globe, including acousmatic pieces, vocal works, ensemble pieces, mixed pieces involving interactive media, interdisciplinary art projects, and an opera. A list of notable compositions is provided in Appendix A.

The developments have also shown strong pedagogical relevance: at the time of writing the libraries have been taught within workshops and as part of courses in various professional and academic institutions both locally (McGill University, CIRMMT, Université de Montréal), and internationally (IRCAM, Université Paris-Sorbonne, University of Music, Stuttgart, Experimentalstudio Freiburg). Most notably, since 2015 the software is part of the Music

[1] The recent version (6.0) has a download count of over 100.000

[2] http://forumnet.ircam.fr, accessed June 25, 2016

[3] http://www.cirmmt.org/activities/newsletter/past/february2012, accessed June 25, 2016

Informatics course program (undergraduate and graduate levels) of the University of Music, Karlsruhe, Germany. A number of graduate students have already started to extend the libraries as part of individual projects.

5.3 Limitations and Future Work

Although the presented libraries offer a complete and functional framework, there is potential for future developments in multiple directions. The OMPrisma library is currently limited to the combination of sound synthesis and spatial sound rendering classes. It would be interesting to extend this system with a collection of sound-processing classes, for filtering or modulation of source signals, prior to spatialization. In this context we have successfully carried out a number of proof-of-concept experiments, for daisy-chaining of multiple classes to build more complex processing topologies. While our developments are primarily concerned with the "writing" of music (écriture), it is clear that a tight feedback loop between the creation of materials and their auditioning is desirable. The current workflow requires sound files to be written in order to be able to audition created materials. Since the Csound language allows for real-time audio rendering, it is conceivable to extend the class-library in OMPrisma with modules for real-time audio input and explore the possibility of using OSC [233] for real-time control. In a parallel direction, we have published first works on the integration of our spatialization tools with more interactive applications for real-time processing and streaming of spatialization data using mobile devices [95].

In terms of gesture composition, we are investigating more flexible data structures and representations, for instance to provide possibilities for multiple, alternative temporal segmentations. On a larger scale, OpenMusic's *Maquette* [4] and *Sheet* [38] objects would be interesting contexts for embedding and manipulating gesture data. Another direction we are exploring is the use of advanced analysis algorithms and machine learning techniques for gesture segmentation and mapping. We have started implementing functions for k-nearest-neighbour classifaction and principal component analysis [23]. These possibilites would enable a different class of applications, such as machine learning of relationships between performed gestures and symbolic music materials or compositional sketches. These relationships could then serve for the creation of new materials. Another promising avenue relates to the recent extension of OpenMusic for uniting demand-driven (typical for deferred-time environments) and reactive (typical for real-time environments) computing paradigms [34]. In this context, it would be interesting to explore the use of OM-Geste as a soft-real-time gesture processor, similar to current works merging compositional and interactive control of spatialization data [95].

In our works on dictionary-based sound representations we are investigating possibilities for control of the target approximation (i.e. "mosaicing") process. Although not explicitly presented in this dissertation, recent developments in OM-Pursuit allow the use of multiple dictionaries, and have integrated a prototype constraint system based on symbolic meta-data and audio descriptors [222]. Logical constraints and weights can be used for controlling or biasing both the selection of sound atoms (e.g. allowing only atoms falling into a certain range of descriptor values), as well as their arrangement in the sound modelling process (e.g. how many simultaneous atoms are allowed at any time). This would enable the dynamic selection of sound materials, as well as controlling temporal and polyphonic aspects of the modelling process, which is a promising step towards more sophisticated applications, such as instrumental transciption or computer-aided orchestration [11].

Appendix A

List of Works

- *Lontananze Fragili II* by Marco Bidin (2016). For video and electronics (video by Nataliya Gurevich). Commissioned for "next generation 6.0" concert, Zentrum für Kunst und Medientechnologie, Karlsruhe, Germany, 2016. http://zkm.de/en/event/2015/06/nextgeneration-60-concert-iii

- *Inside Out* by Marlon Schumacher (2015), in collab. with Graham Boyes and John Sullivan. Networked Intermedia Performance. For Montréal en Lumière/Nuit Blanche, 2015. www.insideout-project.com Video documentation: https://vimeo.com/135278341

- *Naissance des mots* by Núria Giménez-Comas (2015). For cello and electronics. Commissioned by Marie Ythier for "Le geste augmenté", Evidence Classics label. Premiered at Festival Exhibitronic, Strasbourg, France 2015. http://exhibitronic.eu/a-propos/

- *(Dé)jouer le son / Ungraving Sound* by Marlon Schumacher and Karine Bouchard (2015). Interactive Art Installation. Commissioned by 8th International Biennial of Engraving, Trois-Rivières, Canada. Premiered at Ateliér Silex, TR, Canada, 2015. http://www.biectr.ca Video recording available: https://vimeo.com/135280391

- *Three Variations on a Synthetic Sound* by Marlon Schumacher (2015). For brass quartet. Commissioned by Montréal New Music Festival (MNM) for Ensemble Klangfabrik. Montral New Music Festival (MNM). Premiered at Goethe Institut, Montreal, Canda, 2015. http://festivalmnm.ca/mnm/en/2015/prog/concert/33099/

- *No more words* by Núria Giménez-Comas (2015). For soprano and electronics. Premiered by Carla Huhtanen at Conservatory Theater, Toronto, Canada, 2015.

- *Writing against time* by Christopher Trapani (2014). For two pianists, two percussionists, and electronics. Written in conjunction with the Julius F. Jezek prize and premiered by Yarn/Wire at Issue Project Room in Brooklyn, NY, USA, 2014. http://christophertrapani.com/wordpresssite/writing-against-time/. Video recording available: https://www.youtube.com/watch?v=9m3gynTVrDo

- *Urlicht* by Richard Barrett (2014). For three vibraphones, percussion and eight channels electroacoustics. Premiered by Speak Percussion at the Design Hub, RMIT University, Melbourne, Australia, 2014. http://speakpercussion.com/?page_id=1349&year=37#3826

- *Machines for Making (MfM)*: *Delicacy, Zephyrs, Grace, Insight, Faces, Simplicity.* By Lawrence Harvey and Peter Downton. Auditory imagining of architectural models. Produced for *The Portal* (SIAL Sound Studios' speaker orchestra). RMIT University, Melbourne, Australia, 2014.

- *Hyle* by Núria Giménez-Comas (2014). For string quartet (Diotima Quartet). Written for IRCAM's Composition Cursus 2. Premiered at Royaumont Festival, Paris, France, 2014. http://www.nuriagimenezcomas.com/#!blank/c22mu. Audio recording available: https://soundcloud.com/nuriagimenez/hyle-les-g-ants-de-vico/s-nNFGI.

- *Convergence Lines* by Christopher Trapani (2013). For ten instruments (flute, clarinet, horn, trumpet, trombone, piano, harp, percussion, violin, bass) and electronics. Commissioned by GRAME in conjunction with the New Forum Jeune Création. Premiered by Ensemble Orchestral Contemporain at Ultraschall Festival, Berlin, Germany, 2014. http://christophertrapani.com/wordpresssite/convergence-lines Audio recording available: https://soundcloud.com/christophertrapani/convergence-lines

- *Spielraum* by Marlon Schumacher (2013). For violin, cello, digital musical instruments and live electronics. Commissioned by VGCS/CIRMMT for the Research-Creation Project "Les Gestes", Montreal, Canada, 2013. http://www.idmil.org/projects/gestes Video recording: https://vimeo.com/148773944

- *Continuous Snapshots* by Sébastien Gaxie (2013). For piano and electronics. Commissioned by the Ircam ManiFeste Festival 2013. Premiered by David Lively on June 6th, 2013 at Centre Pompidou, Paris. http://brahms.ircam.fr/works/work/34275/

- *Codex XIII* by Richard Barrett (2013). Improvisational Structure for Large Ensemble and Electronics. Commissioned by the Symposium Composing Spaces 2013, Royal Conservatory of Den Haag, Netherlands. Premiered April 12, 2013.

- *6 Fragments on One Act of Cleaning a Piano* by Marlon Schumacher (2013). For two pianists, vibrational transducers, and electronics. Commissioned by Edition Fink label. Premiered by Christopher Goddard and Charles Zoll at McGill Association of Student Composers (MASC/ACEM) workshop, McGill University, Canada, 2014. Video recording: https://www.youtube.com/watch?v=B-VUxA_nVUk

- *Widening Circles* by Christopher Trapani (2012). For shô, sheng, clarinet, accordion, harp, zheng, qanûn, and santur and electronics. Commissioned by Atlas Ensemble. Premiered at Muziekgebouw aan't IJ, Amsterdam, Netherlands, 2012. http://christophertrapani.com/wordpresssite/widening-circles Audio recording: https://soundcloud.com/christophertrapani/widening-circles

- *Spin* by Marlon Schumacher and Graham Boyes (2012) . For digial percussion, interactive video and live electronics. Commissioned by Codes D'Acces. Premiered at the "Prisma et Les Messagers" concert, Usine-C, Montreal, Canada, April 2012. http://www.codesdacces.org

- *Re Orso* by Marco Stroppa (? - 2011). Opera, Libretto by Catherine Ailloud-Nicolas and Giordano Ferrari after Arrigo Boito. Commissioned by Opéra Comique, Thtre Royal de la Monnaie, Ensemble InterContemporain, Françoise and Jean-Philippe Billarant, IRCAM-Centre Pompidou, Paris. Premiere: May 19, 2011. Video documentation: https://www.youtube.com/watch?v=EACKCP-oZ8c

- *Ab-Tasten* by Marlon Schumacher (2011). For computer-controlled piano, 4 virtual pianos, and spatial sound synthesis. Commissioned by CIRMMT. Premiered at the live@CIRMMT concert Clavisph'ere, Montreal, Canada, 2011. Audio recording: https://soundcloud.com/marleynoe/ab-tasten-for-disklavier-4?in=marleynoe/sets/ab-tasten

- *Construction* by Richard Barrett (2005-2011). For three vocalists, ensemble, and electronics. Commissioned by Liverpool Capital of Culture. Supported by Sound and Music, ELISION Ensemble, Ernst von Siemens music foundation; British Council, Australian Research Council. Premiered at the Huddersfield Contemporary Music Festival, UK, 2011. Video documentation: https://vimeo.com/35322232

- *Cognitive Consonance* by Christopher Trapani (2010). For qanûn, hexaphonic electric guitar, ensemble (flute, clarinet, harp, mandolin, percussion, violin, cello, bass), and electronics. Written for IRCAM's Composition Cursus 2. Premiered at Agora Festival, Paris, France, 2010. http://christophertrapani.com/wordpresssite/cognitive-consonance Audio recording: https://soundcloud.com/christophertrapani/cognitive-consonance

Appendix B

List of Software Tools and Functions

This appendix gives an overview of the various software components constituting the software framework presented in this thesis.

B.1 OMPrisma

Table B.1: Overview of OMPrisma Spatial Rendering Classes

Classname	Technique	2D/3D	Sweetspot	Local/Global	LS Config	ICLD	ICTD	Directivity	Room Model
AMBI	Higher-order ambisonics [68]	3D	Y	global	spherical	X			
BABO	Ball-in-a-Box [186]	3D	N	global	arbitrary	X	X		Physical Resonator Model
BINAURAL	HRTF, Reverb Model (dev) [57]	3D	na	na	na	X	X	X	Image-Source, Feedback-Delay-Network
DBAP+	Distance-Based Amplitude Panning [132]	3D	N	global	arbitrary	X	X		
SURROUND	Amplitude Panning, ITU 5.0 [67]	2D	Y	local	fixed	X			
RVBAP	Reverberated VBAP [144]	3D	Y	local	spherical	X			Feedback-Delay-Network
RVIMIC	Reverberated ViMiC (dev) [30, 144]	3D	N	global	arbitrary	X	X		Feedback-Delay-Network
SPAT	1st-Order Ambisonics, Room model [98]	3D	Y	global	spherical	X			Image-Source
SUG	Room-in-a-Room (dev) [152]	3D	N	global	fixed	X	X	X	Image-Source
TRANSAURAL	HRTF, Reverb Model, RACE (dev) [100]	3D	Y	na	fixed	X	X	X	Image-Source, Feedback-Delay-Network
VBAP	Vector-Base Amplitude Panning [178]	3D	Y	local	spherical	X			
VIMIC	Virtual Microphone Control [30]	3D	N	global	arbitrary	X	X	X	

Trajectory Functions	
`traj-dur`	Converts a duration in secs to a *BPF* with the corresponding frequency value in Hz as a straight line.
`traj-interpol`	Interpolates two trajectories of same type (*BPF*, *BPC*, *3DCs*, *3D-trajectory*)
`traj-mirror`	Mirrors (calculates a symmetric curve) of *BPC*, *3DC*, *3D-trajectory* along principal axes.
`traj-mult`	Scales individual dimensions of a *BPF*, *BPC*, *3DC*, *3D-trajectory* by multiplication factor.
`traj-perturb`	Perturbates a *BPF*, *BPC*, *3DC*, *BPF/BPC/3DC-lib*s statically (number) or dynamically (list, *BPF*).
`traj-reverse`	Reverses the sequence of spatial points describing a trajectory in (*BPF*, *BPC*, *3DC*, *3D-trajectory*)
`traj-rotate`	Rotation in 3D using Euler angles for *BPC*, *3DC*, *3D-trajectory*, *3DC-lib*.
`traj-scale`	Scales individual dimensions of a BPF, *BPC*, *3DC*, *3D-trajectory* according to min max values.
`traj-translate`	Translates a *BPC*, *3DC*, *3D-trajectory* in 3 cartesian dimensions.
`trajectory-lib`	Trajectory generation function for 2D or 3D based on parametric functions (lissajous, hyperbola, helix, etc.).
Perceptual Cues	
`air-furse`	Air-absorption function proposed by R. Furse.
`air-icst`	Arbitrary high frequency cutoff as a function of distance
`air-valente`	3rd-order polynomial best-fit air-absorption function (D. Valente).
`atten-exp`	Exponential distance attenuation.
`atten-invprop`	Inverse proportional distance attenuation (in *dB* per doubling of distance).
`time-of-flight`	Calculates time delay as a function of distance and speed of sound.
Parsing Functions	
`2D-mesh`	Calculates a 2D-mesh of points with variable size and resolution (uniform spreading e.g. for *Sound Surface Panning*).
`2D-refs_is`	User-Fun for calculating image-sources in 2D
`2D-refs_perc`	User-Fun for calculating image-sources in 2D
`3D-refs_is`	User-Fun for calculating image-sources in 3D
`rescale-env`	Rescales envelopes (trajectories) in 3D.
`w-distance`	Computes gain factor as a function of width and distance
`w-panning`	Returns ambisonic order as a function of width an distance of a sound source. Implements W-panning [148] function.

Geometry	
`ad->xy`	Converts 2D polar coordinates [azimuth, distance] to cartesian coordinates [x,y].
`aed->xyz`	Converts 3D spherical coordinates [azimuth, elevation, distance] to cartesian coordinates [x,y,z].
`xy->ad`	Converts 2D cartesian coordinates [x,y] to polar coordinates [azimuth, distance].
`xyz->aed`	Converts 3D cartesian coordinates [x,y,z] to spherical coordinates [azimuth, elevation, distance].
`platonic-solids`	Calculates platonic solids as *3DC* objects and optionally sends Open Sound Control messages to set the MULTIPLAYER.
Image-Source Reflections	
`image-source-2D`	Calculates n-th order image-source reflections for arbitrary room geometries defined in 2D
`image-source-3D`	Calculates n-th order image-source reflections for arbitrary room geometries defined in 3D
`make-walls`	Create point-symmetric walls around the coordinate origins for defining a rectangular room
Loudspeaker Configurations	
`prisma-setup`	Configures OMPRISMA renderers for loudspeaker configuration via preset (menu), *BPC* or *3DC* object.
`speaker-presets`	Returns loudspeaker-setups as *BPC*/*3DC* and optionally sends OpenSoundControl messages to set the Multiplayer
OM-Spat/SpatDIF Interface	
`objfromobjs (SDIFfile)`	Initializes an OMPRISMA class from an *SDIFfile* containing SpatDIF descriptors.
`objfromobjs (Spat-Matrix)`	Initializes an OMPRISMA class from a *Spat-Matrix*.
Class Merging	
`chroma-prisma`	Merges an OMCHROMA synthesis class with an OMPRISMA spatialization class
Utilities	
`3DClib->3DC`	Concatenates the individual *3DC*s in a *3DC-lib* to form a single *3DC*.
`traj->BPFs`	Converts a *BPC*, *3DC*, *3D-trajectory* into individual *BPF* objects.
`BPF-interpol`	Interpolates between two *BPF*s or two *BPC*s
`BPF-perturb`	Perturbates a *BPF*, *BPF-lib*, or list of *BPF*s, within boundaries given by min and max.

B.2 OM-Geste

Gesture-Models	
`add-column`	Adds a temporal segment to a *gesture-model* object.
`add-row`	Adds a new stream or list of streams to a *gesture-model* object.
`get-column`	Extracts a temporal segment from a *gesture-model* object.
`get-field`	Extracts a temporal segment of a stream from a *gesture-model* object.
`get-row`	Extracts a stream from a gesture-model object.
`make-gdif-buffer`	Formats data in a *gesture-model* for storage in an SDIF file.
`process-column`	Higher-order function for processing of a temporal segment in a *gesture-model* object.
`process-field`	Higher-order function for processing of a temporal segment of a stream in a *gesture-model* object.
`process-row`	Higher-order function for processing of a stream or list of streams in a *gesture-model* object.
`remove-column`	Removes a temporal segment from a *gesture-model* object.
`remove-row`	Removes a stream or list of streams from a *gesture-model* object.
`segment-gesture`	Segments a *gesture-model* according to temporal structure.
Mapping	
`get-times`	Retrieves time points from a *3D-trajectory* or *BPF* object.
`map-gesture`	Applies a mapping from *gesture-model* to an OpenMusic class defined in a visual program.
`segment-dur`	Calculates duration of segment from timepoints.
Utilities	
`scrubber`	Allows scrubbing through a function using a sample window of n points.
Classes	
`gesture-array`	A structure of *gesture-streams* containing *substreams*.
`gesture-model`	A matrix structure of temporal segments.

Statistics	
`differentiate`	n-th-order (recursive) differentiator.
`integrate`	n-th-order (recursive) integrator.
`extrema`	Returns extrema of a time-domain function.
`rms`	Root-mean-square of a sample window.
`magnitude`	Calculates magnitude of n-dimensional vector.
`euc-distance`	Calculates euclidean distance between 2 vectors.
`minmax`	Returns extrema of a list.
`arith-mean`	Calculates the arithmetic mean of a series of samples.
`variance`	Calculates the variance of a series of samples.
`stdev`	Calculates the sample/standard deviation.
`dot-product`	Calculates dot-product of two vectors.
`covariance`	Calculates covariance of two vectors over a window.
`correlation`	Calculates correlation of two vectors over a window.
`centroid`	Calculates the center-of-gravity of a series of samples.
`find-peaks`	Finds peaks/troughs in a time-domain function.
`KNN`	Finds k-nearest neighbours in n-dimensional space.
Filters	
`EMA-HP`	Implements an exponentiall-weighted moving-average filter (highpass).
`EMA-LP`	Implements an exponentiall-weighted moving-average filter (lowpass).
`EMD`	Exponential-moving-difference. First-order difference of an EMA-filtered signal.
`SMA-HP`	Implements the simple-moving-average: the arithmetic mean of a list of numbers in a sliding window (highpass).
`SMA-LP`	Implements the simple-moving-average: the arithmetic mean of a list of numbers in a sliding window (lowpass).
`SMM-HP`	Implements a simple-moving-median: the median of a list of numbers in a sliding window (highpass).
`SMM-LP`	Implements a simple-moving-median: the median of a list of numbers in a sliding window (lowpass).
`WMA-HP`	Calculates the linear-weighted-moving-average of a list of numbers in a sliding window (highpass).
`WMA-LP`	Calculates the linear-weighted-moving-average of a list of numbers in a sliding window (lowpass).

B.3 OM-Pursuit

Constraints	
`ctr-combine`	Function combining constraints using logical connective
`ctr-compile`	Compiles a hierarchy of nested sgn-constraint constraint into an SDIF file.
`ctr-conditional`	Defines and returns an instance of a constraint specification.
`ctr-define`	Defines and returns an instance of sgn-constraint
`ctr-weight`	Function for weighting constraints
`mpctr-define`	Defines and returns an instance of mp-constraint.
SDIF Interface	
`find-sid`	Returns source/treeway for StreamIDs from a list of StreamIDtable *sdifsid* objects.
`get-pursuit-data`	Gets the model-data from an SDIFfile
`getpursuitSIDlist`	Returns the list of Stream ID descriptions in an SID table.
`model-to-matrix`	Converts an OM-PURSUIT model represented in an SDIF into a *soundgrain-matrix* class
`pursuit-sid-paths`	Returns source/treeway for a list of StreamIDs in a StreamIDtable *sdifsid*.
`pursuit-sids`	Returns source/treeway for a list of StreamIDs in a StreamIDtable *sdifsid* .
Matching Pursuit	
`pursuit-dictionary`	The OM-Pursuit dictionary building function
`soundgrain-decomp`	Main function for atomic decomposition of target sound.

References

[1] HighC. [Online]. Available: http://highc.org/ *(Cited on page 65)*

[2] junXion. [Online]. Available: http://steim.org/product/junxion/ *(Cited on page 80)*

[3] OSCulator. [Online]. Available: http://www.osculator.net/ *(Cited on page 80)*

[4] C. Agon, "OpenMusic: Un langage visuel pour la composition musicale assistée par ordinateur," Ph.D. dissertation, Université Pierre et Marie Curie, Paris VI, 1998. *(Cited on pages 9, 14, 31, 59, and 123)*

[5] C. Agon and G. Assayag, "Programmation visuelle et editeurs musicaux pour la composition assistée par ordinateur," in *14ème Conférence Francophone sur l'Interaction Homme-Machine*, Poitier, France, 2002, pp. 205–206. *(Cited on page 12)*

[6] C. Agon, G. Assayag, and J. Bresson, Eds., *The OM Composer's Book: Volume 1*, ser. Collection Musique/Sciences. Éditions Delatour France / IRCAM—Centre Pompidou, 2006. *(Cited on pages 24, 34, and 63)*

[7] C. Agon, G. Assayag, J. Fineberg, and C. Rueda, "Kant: a Critique of Pure Quantification." in *International Computer Music Conference*, Aarhus, Denmark, 1994, pp. 52–59. *(Cited on page 8)*

[8] C. Agon, J. Bresson, and M. Stroppa, "OMChroma: Compositional Control of Sound Synthesis," *Computer Music Journal*, vol. 35, no. 2, pp. 67–83, May 2011. *(Cited on pages 9, 15, and 93)*

[9] C. Agon, M. Stroppa, and G. Assayag, "High Level Musical Control of Sound Synthesis in OpenMusic," in *International Computer Music Conference*, Berlin, Germany, 2000, pp. 332–335. *(Cited on page 34)*

[10] C. Ames, "The Markov process as a compositional model: a survey and tutorial," *Leonardo*, vol. 22, no. 2, pp. 175–187, Jan. 1989. *(Cited on page 2)*

[11] A. Antoine and E. R. Miranda, "Towards Intelligent Orchestration Systems," in *11th International Symposium on Computer Music Multidisciplinary Research*, Plymouth, UK, 2015. *(Cited on page 124)*

[12] D. Arfib, J. M. Couturier, L. Kessous, and V. Verfaille, "Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces," *Organised Sound*, vol. 7, no. 02, pp. 127–144, Aug. 2002. *(Cited on page 81)*

[13] C. Ariza, "Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors and a Lexicon of Systems and Research," in *International Computer Music Conference*, Barcelona, Spain, 2005, pp. 765–772. *(Cited on pages 62, 70, and 80)*

[14] ——, "Two pioneering projects from the early history of computer-aided algorithmic composition," *Computer Music Journal*, vol. 35, no. 3, pp. 40–56, Sep. 2011. *(Cited on pages 1, 2, and 69)*

[15] D. F. Armstrong, W. C. Stokoe, and S. E. Wilcox, *Gesture and the Nature of Language*. Cambridge University Press, 1995. *(Cited on pages 19 and 62)*

[16] G. Assayag, "Computer Assisted Composition today," in *First Symposium on Music and Computers*, Corfu, Greece, 1998. *(Cited on pages 3, 17, 33, 62, 80, 82, and 120)*

[17] G. Assayag, M. Castellengo, and C. Malherbe, "Functional Integration of Complex Instrumental Sounds in Musical Writing," in *International Computer Music Conference*, Burnaby, Canada, 1985. *(Cited on page 4)*

[18] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, "Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic," *Computer Music Journal*, vol. 23, no. 3, pp. 59–72, Oct. 1999. *(Cited on pages 10, 20, 24, and 31)*

[19] M. Balaban, K. Ebcioglu, and O. E. Laske, *Understanding music with AI : perspectives on music cognition*. MIT Press, 1992. *(Cited on page 2)*

[20] M. Beaudouin-Lafon, "Instrumental interaction: an interaction model for designing post-WIMP user interfaces," in *SIGCHI Conference on Human Factors in Computing Systems*, The Hague, Netherlands, 2000, pp. 446–453. *(Cited on pages 20, 66, and 69)*

[21] A. J. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *Journal of the Acoustical Society of America*, vol. 93, no. 5, pp. 2764–2778, May 1993. *(Cited on pages 16 and 30)*

[22] M. Bernays, "The expression and production of piano timbre: gestural control and technique, perception and verbalisation in the context of piano performance and practice," Ph.D. dissertation, Université de Montréal, Jan. 2013. *(Cited on pages 98 and 116)*

[23] F. Bevilacqua, J. Ridenour, and D. J. Cuccia, "3D Motion Capture Data: Motion Analysis and Mapping to Music," in *Workshop/Symposium on Sensing and Input for Media-centric Systems*, Santa Barbara, USA, 2002. *(Cited on pages 20, 72, 86, 89, and 123)*

[24] F. Bevilacqua, N. Schnell, N. Rasamimanana, B. Zamborlin, and F. Guedy, "Online Gesture Analysis and Control of Audio Processing," in *Musical Robots and Interactive Multimodal Systems*, J. Solis and K. C. Ng, Eds. Springer Berlin Heidelberg, 2011, pp. 127–142. *(Cited on page 72)*

[25] J. Blauert, *Spatial Hearing: The Psychophysics of Human Sound Localization.* MIT Press, 1983. *(Cited on page 41)*

[26] M. Böhlandt, ""Kontakte" - Reflexionen naturwissenschaftlich-technischer Innovationsprozesse in der frühen Elektronischen Musik Karlheinz Stockhausens (1952-1960)," *Berichte zur Wissenschaftsgeschichte*, vol. 31, no. 3, pp. 226–248, Sep. 2008. *(Cited on page 16)*

[27] A. Bonnet and C. Rueda, *Un langage visuel basé sur les contraintes pour la composition musicale*, ser. Recherches et applications en informatique musicale. Hermes, 1998. *(Cited on page 8)*

[28] R. Boulanger, *The Csound Book.* The MIT Press, 2000. *(Cited on page 34)*

[29] G. Boyes, "Dictionary-Based Analysis/Synthesis and Structured Representations of Musical Audio," Master's thesis, McGill University, Dec. 2011. *(Cited on page 105)*

[30] J. Braasch, "A Loudspeaker-based 3D Sound Projection using Virtual Microphone Control (ViMiC)," in *118th Convention of the Audio Engineering Society*, Barcelona, Spain, 2005. *(Cited on pages 30 and 130)*

[31] A. S. Bregman, *Auditory Scene Analysis - The Perceptual Organization of Sound.* MIT Press, 1994. *(Cited on pages 17, 111, and 114)*

[32] J. Bresson, "Sound Processing in OpenMusic," in *International Conference on Digital Audio Effects*, Montreal, Canada, 2006, pp. 325–330. *(Cited on pages 15, 55, and 69)*

[33] ——, "La synthèse sonore en composition musicale assistée par ordinateur," Ph.D. dissertation, École Doctorale d'Informatique, Telecommunications et Électronique de Paris, Jan. 2008. *(Cited on page 21)*

[34] ——, "Reactive Visual Programs for Computer-Aided Music Composition," in *IEEE Symposium on Visual Languages and Human-Centric Computing*, Melbourne, Australia, 2014, pp. 141–144. *(Cited on pages 10, 78, and 123)*

[35] J. Bresson and C. Agon, "SDIF Sound Description Data Representation and Manipulation in Computer Assisted Composition," in *International Computer Music Conference*, Miami, USA, Nov. 2004, pp. 520–527. *(Cited on page 68)*

[36] ——, "Sound Writing and Representation in a Visual Programming Framework," in *Digital Music Research Network Doctoral Research Conference*, London, United Kingdom, 2006. *(Cited on page 21)*

[37] ——, "Musical Representation of Sound in Computer-Aided Composition: A Visual Programming Framework," *Journal of New Music Research*, vol. 36, no. 4, pp. 251–266, Dec. 2007. *(Cited on pages 20, 34, 63, 69, and 99)*

[38] ——, "Scores, Programs, and Time Representation: The Sheet Object in OpenMusic," *Computer Music Journal*, vol. 32, no. 4, pp. 31–47, Nov. 2008. *(Cited on page 123)*

[39] ——, "Processing Sound And Music Description Data Using OpenMusic." in *International Computer Music Conference*, New York, USA, 2010, pp. 546–549. *(Cited on page 19)*

[40] J. Bresson, C. Agon, and G. Assayag, Eds., *The OM Composer's Book: Volume 2*, ser. Collection Musique/Sciences. Éditions Delatour France / IRCAM—Centre Pompidou, 2008. *(Cited on pages 21, 24, 34, 63, 99, and 119)*

[41] J. Bresson, C. Agon, and G. Assayag, "Visual Lisp/CLOS programming in OpenMusic," *Higher-Order and Symbolic Computation*, vol. 22, no. 1, pp. 81–111, Dec. 2009. *(Cited on pages 6 and 9)*

[42] ——, "OpenMusic: visual programming environment for music composition, analysis and research," in *ACM International Conference on Multimedia*, Scottsdale, USA, 2011, pp. 743–746. *(Cited on pages 7, 24, and 64)*

[43] J. Bresson, C. Agon, and M. Schumacher, "Représentation des données de contrôle pour la spatialisation dans OpenMusic," in *Journées d'Informatique Musicale*, Rennes, France, 2010. *(Cited on pages 33, 40, and 122)*

[44] J. Bresson, D. Bouche, J. Garcia, T. Carpentier, F. Jacquemard, J. MacCallum, and D. Schwarz, "Projet EFFICACE: Développements et perspectives en composition assistée par ordinateur," in *Journées d'Informatique Musicale*, Montreal, Canada, Mar. 2015. *(Cited on page 24)*

[45] J. Bresson, F. Guedy, and G. Assayag, "Musique Lab 2: From computer-aided composition to music education," *Journal of Music, Technology and Education*, vol. 5, no. 3, pp. 273–291, Jan. 2013. *(Cited on page 24)*

[46] J. Bresson and M. Schumacher, "Representation and interchange of sound spatialization data for compositional applications," in *International Computer Music Conference*, Huddersfield, United Kingdom, 2011. *(Cited on pages 26, 68, 78, 120, and 122)*

[47] J. Bresson, M. Stroppa, and C. Agon, "Generation and Representation of Data and Events for the Control of Sound Synthesis," in *Sound and Music Computing Conference*, Lefkada, Greece, 2007. *(Cited on pages 59 and 62)*

[48] B. Burger and P. Toiviainen, "MoCap Toolbox – A Matlab toolbox for computational analysis of movement data," in *Sound and Music Computing Conference*, Stockholm, Sweden, 2013, pp. 172–178. *(Cited on page 76)*

[49] B. Cabaud and L. Pottier, "Le contrôle de la spatialisation multi-sources. Nouvelles fonctionnalités dans Holophon version 2.2," in *Journées d'Informatique Musicale*, Marseille, France, 2002, pp. 269–271. *(Cited on page 32)*

[50] C. Cadoz, "Instrumental Gesture and Musical Composition," in *International Computer Music Conference*, San Francisco, USA, 1988, pp. 1–12. *(Cited on pages 63, 66, and 78)*

[51] ——, "Le geste canal de communication homme/machine: la communication instrumentale," *TSI. Technique et science informatiques*, vol. 13, no. 1, pp. 31–61, 1994. *(Cited on page 63)*

[52] C. Cadoz and C. Ramstein, "Capture, representation, and 'composition' of the instrumental gesture," in *International Computer Music Conference*, Glasgow, Scotland, 1990, pp. 53–56. *(Cited on pages 66, 71, and 74)*

[53] C. Cadoz and M. M. Wanderley, "Gesture-Music," in *Trends in Gestural Control of Music*, M. M. Wanderley and M. Battier, Eds. IRCAM—Centre Pompidou, 2000, pp. 71–93. *(Cited on page 62)*

[54] A. Camurri, P. Coletta, G. Varni, and S. Ghisio, "Developing Multimodal Interactive Systems with EyesWeb XMI." in *Conference on New Interfaces for Musical Expression*, New York, USA, 2007, pp. 305–308. *(Cited on page 68)*

[55] B. Caramiaux, M. M. Wanderley, and F. Bevilacqua, "Segmenting and Parsing Instrumentalists' Gestures," *Journal of New Music Research*, vol. 41, no. 1, pp. 13–29, Mar. 2012. *(Cited on pages 20 and 72)*

[56] G. Carpentier and J. Bresson, "Interacting with Symbol, Sound, and Feature Spaces in Orchidée, a Computer-Aided Orchestration Environment," *Computer Music Journal*, vol. 34, no. 1, pp. 10–27, Feb. 2010. *(Cited on pages 23 and 62)*

[57] B. Carty, "hrtfmove, hrtfstat, hrtfmove2: Using the New HRTF Opcodes," *Csound Journal*, no. 9, pp. 1–8, Feb. 2014. *(Cited on page 130)*

[58] A. Cassidy and A. Einbond, Eds., *Noise in and as Music*. University of Huddersfield, 2013. *(Cited on pages 22 and 121)*

[59] L. E. Castelões, "A Catalogue of Music Onomatopoeia," *International Review of the Aesthetics and Sociology of Music*, vol. 40, no. 2, Dec. 2009. *(Cited on page 23)*

[60] J. Chadabe, "Interactive Composing: An Overview," *Computer Music Journal*, vol. 8, no. 1, pp. 22–27, Apr. 1984. *(Cited on page 5)*

[61] ——, "The Limitations of Mapping as a Structural Descriptive in Electronic Instruments," in *Conference on New Interfaces for Musical Expression*, Dublin, Ireland, 2002, pp. 197–201. *(Cited on pages 69 and 82)*

[62] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Computer Music Journal*, vol. 1, no. 2, pp. 46–54, Apr. 1977. *(Cited on page 16)*

[63] T. Coduys and G. Ferry, "IanniX. Aesthetical/symbolic visualisations for hypermedia composition," in *Sound and Music Computing Conference*, Paris, France, 2004, pp. 18–23. *(Cited on page 65)*

[64] D. Cope, "Experiments in Musical Intelligence (EMI): Non-linear Linguistic-based Composition," *Journal of New Music Research*, vol. 18, no. 1-2, pp. 117–139, Jan. 1989. *(Cited on page 2)*

[65] T. Coughlan and P. Johnson, "Interaction in Creative Tasks: Ideation, Representation and Evaluation in Composition," in *SIGCHI Conference on Human Factors in Computing Systems*, Montreal, Canada, 2006, pp. 531–540. *(Cited on page 66)*

[66] F. Courtot, "CARLA: Knowledge Acquisition and Induction for Computer Assisted Composition," *Journal of New Music Research*, vol. 21, no. 3-4, pp. 191–217, Jan. 1992. *(Cited on page 4)*

[67] P. Craven, "Continuous Surround Panning for 5-speaker Reproduction," in *24th International Conference of the Audio Engineering Society*, Banff, Canada, 2003. *(Cited on page 130)*

[68] J. Daniel, "Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia," Ph.D. dissertation, Université Pierre et Marie Curie, Paris VI, Oct. 2001. *(Cited on pages 16, 30, and 130)*

[69] ——, "Spatial Sound Encoding Including Near Field Effect: Introducing Distance Coding Filters and a Viable, New Ambisonic Format," in *23rd International Conference of the Audio Engineering Society*, 2003. *(Cited on page 42)*

[70] F. Delalande, "Le Geste, outil d'analyse: quelques enseignements d'une recherche sur la gestique de Glenn Gould," *Analyse Musicale*, 1988. *(Cited on pages 62 and 63)*

[71] O. Delerue, "Spatialisation du son et programmation par contraintes : le système MusicSpace," Ph.D. dissertation, Université Pierre et Marie Curie, Paris VI, Apr. 2004. *(Cited on page 32)*

[72] O. Delerue and C. Agon, "Open Music + Music Space = Open Space," in *Journées d'Informatique Musicale*, Issy-les-Moulineaux, France, 1999, pp. 89–96. *(Cited on pages 16 and 33)*

[73] D. Deutsch, "Grouping mechanisms in music," in *The psychology of music*, D. Deutsch, Ed. Academic Press, San Diego, 1999. *(Cited on page 115)*

[74] P. Doornbusch, "A brief survey of mapping in algorithmic composition," in *International Computer Music Conference*, Gothenburg, Sweden, 2002. *(Cited on pages 21 and 81)*

[75] ——, "The Application of Mapping in Composition and Design," in *Form, space, time: the Australasian Computer Music Conference*, Melbourne, Australia, 2002. *(Cited on page 69)*

[76] ——, "Composers' views on mapping in algorithmic composition," *Organised Sound*, vol. 7, no. 02, pp. 145–156, Aug. 2002. *(Cited on pages 69 and 84)*

[77] ——, "Computer Sound Synthesis in 1951: The Music of CSIRAC," *Computer Music Journal*, vol. 28, no. 1, pp. 10–25, 2004. *(Cited on page 1)*

[78] ——, "Mapping in Algorithmic Composition and Related Practices," Ph.D. dissertation, RMIT University, Victoria, 2010. *(Cited on page 80)*

[79] M. N. Downie, "Choreographing the extended agent: performance graphics for dance theater," Ph.D. dissertation, Massachusetts Institute of Technology, 2005. *(Cited on pages 69 and 82)*

[80] A. Einbond, D. Schwarz, and J. Bresson, "Corpus-based transcription as an approach to the compositional control of timbre," in *International Computer Music Conference*, Montreal, Canada, 2009, pp. 223–226. *(Cited on pages 22, 23, and 102)*

[81] N. Ellis, J. Bensoam, and R. Caussé, "Modalys demonstration," in *International Computer Music Conference*, Barcelona, Spain, 2005, pp. 101–102. *(Cited on page 9)*

[82] W. D. Ellis, Ed., *A source book of Gestalt Psychology.* Psychology Press, 1999, vol. 2. *(Cited on page 103)*

[83] G. Essl and S. O'Modhrain, "Enaction in the Context of Musical Performance," *Interdisciplines virtual workshop (by participants in Enactive interfaces Network)*, 2004. *(Cited on pages 19 and 62)*

[84] M. Evrard, D. Couroussé, N. Castagné, C. Cadoz, J.-L. Florens, and A. Luciani, "The GMS File Format Version 0.1," Sep. 2006. *(Cited on page 67)*

[85] A. Farhang, "Modelling a gesture: Tak-Sim for string quartet and live electronics," in *The OM Composer's Book: Volume 3.* Éditions Delatour France / IRCAM—Centre Pompidou, in press. *(Cited on page 63)*

[86] S. Ferguson and M. M. Wanderley, "The McGill Digital Orchestra: An Interdisciplinary Project on Digital Musical Instruments," *Journal of Interdisciplinary Music Studies*, vol. 4, no. 2, pp. 17–35, Nov. 2010. *(Cited on page 76)*

[87] J. Fitch, V. Lazzarini, and S. Yi, "Csound6: old code renewed," in *Linux Audio Conference*, Graz, Austria, 2013, pp. 69–75. *(Cited on pages 9 and 93)*

[88] R. Foulon and J. Bresson, "Un modèle de contrôle pour la synthèse par fonctions d'ondes formantiques avec OM-Chant," in *Journées d'Informatique Musicale*, Paris, France, 2013. *(Cited on pages 9 and 15)*

[89] J. Françoise, "Realtime Segmentation and Recognition of Gestures using Hierarchical Markov Models," Master's thesis, Université Pierre et Marie Curie, Paris VI, Sep. 2011. *(Cited on page 20)*

[90] W. J. Froman, "Action Painting and the World-as-Picture," *The Journal of Aesthetics and Art Criticism*, vol. 46, no. 4, pp. 469–475, Jul. 1988. *(Cited on page 64)*

[91] R. Gabriel, J. White, and D. Bobrow, "CLOS: Integrating Object-Oriented and Functional Programming," *Communications of the ACM*, vol. 34, no. 9, pp. 29–38, Sep. 1991. *(Cited on pages 7, 9, 11, 34, 51, and 82)*

[92] J. Garcia, T. Tsandilas, and C. Agon, "Interactive Paper Substrates to Support Musical Creation," in *SIGCHI Conference on Human Factors in Computing Systems*, Austin, USA, 2012, pp. 1825–1828. *(Cited on page 66)*

[93] J. Garcia, T. Tsandilas, C. Agon, and W. Mackay, "InkSplorer: Exploring Musical Ideas on Paper and Computer," in *Conference on New Interfaces for Musical Expression*, Oslo, Norway, 2011. *(Cited on page 66)*

[94] J. Garcia, J. Bresson, and T. Carpentier, "Towards interactive authoring tools for composing spatialization." in *IEEE 10th Symposium on 3D User Interfaces*, Arles, France, Mar. 2015, pp. 151–152. *(Cited on pages 19 and 63)*

[95] J. Garcia, J. Bresson, M. Schumacher, T. Carpentier, and X. Favory, "Tools and Applications for Interactive-Algorithmic Control of Sound Spatialization in OpenMusic," in *inSONIC2015, Aesthetics of Spatial Audio in Sound, Music and Sound Art*, Karlsruhe, Germany, 2015. *(Cited on pages 63, 120, 122, and 123)*

[96] J. Garcia, P. Leroux, and J. Bresson, "pOM: Linking Pen Gestures to Computer-Aided Composition Processes," in *40th International Computer Music Conference (ICMC) joint with the 11th Sound & Music Computing conference (SMC)*, Athens, Greece, 2014. *(Cited on pages 63 and 66)*

[97] M. Geier, A. Jens, and S. Spors, "ASDF: Ein XML Format zur Beschreibung von virtuellen 3D-Audioszenen," in *German Annual Conference on Acoustics*, Dresden, Germany, Apr. 2008. *(Cited on page 32)*

[98] M. A. Gerzon, "Periphony: With-height sound reproduction," *Journal of the Audio Engineering Society*, vol. 21, no. 1, pp. 2–10, Feb. 1973. *(Cited on page 130)*

[99] S. Gill, "A Technique for the Composition of Music in a Computer," *The Computer Journal*, vol. 6, no. 2, pp. 129–133, Aug. 1963. *(Cited on page 5)*

[100] R. Glasgal, "360 ° Localization via 4.x RACE Processing," in *123rd Convention of the Audio Engineering Society*, New York, USA, 2007, pp. 1–11. *(Cited on page 130)*

[101] R. I. Godøy, "Motor-Mimetic Music Cognition," *Leonardo*, vol. 36, no. 4, pp. 317–319, Jan. 2003. *(Cited on page 62)*

[102] M. Good, "MusicXML: An Internet-Friendly Format for Sheet Music," in *XML Conference and Expo*, Boston, USA, 2001, pp. 1–12. *(Cited on page 15)*

[103] J. G. Greeno, "Gibson's Affordances," *Psychological Review*, vol. 101, no. 2, pp. 336–342, 1994. *(Cited on page 62)*

[104] B. Hackbarth, N. Schnell, and P. Esling, "Composing Morphology: Concatenative Synthesis as an Intuitive Medium for Prescribing Sound in Time," *Contemporary Music Review*, vol. 32, no. 1, pp. 49–59, Feb. 2013. *(Cited on pages 22 and 102)*

[105] M. A. Harley, "Space and Spatialization in Contemporary Music: History, Analysis, Ideas and Implementations," Ph.D. dissertation, Faculty of Music of McGill University, Montreal, Apr. 1994. *(Cited on pages 30 and 38)*

[106] ——, "Spatiality of sound and stream segregation in twentieth century instrumental music," *Organised Sound*, vol. 3, no. 2, pp. 147–166, Aug. 1998. *(Cited on page 39)*

[107] I. Hattwick, J. Malloch, and M. M. Wanderley, "Forming Shapes to Bodies: Design for Manufacturing in the Prosthetic Instruments." in *Conference on New Interfaces for Musical Expression*, London, United Kingdom, 2014, pp. 443–448. *(Cited on page 87)*

[108] G. Haus and A. Sametti, "Scoresynth: A System for the Synthesis of Music Scores Based on Petri Nets and a Music Algebra," *IEEE Computer*, vol. 24, no. 7, pp. 56–60, 1991. *(Cited on page 2)*

[109] L. Hiller and L. Isaacson, "Musical composition with a high-speed digital computer," *Journal of the Audio Engineering Society*, vol. 6, no. 3, pp. 154–160, 1958. *(Cited on pages 1 and 62)*

[110] L. Hiller, A. Leal, and R. A. Baker, "Revised MUSICOMP manual," University of Illinois, Experimental Music Studio, Tech. Rep., 1966. *(Cited on page 3)*

[111] A. Hunt and M. M. Wanderley, "Mapping performer parameters to synthesis engines," *Organised Sound*, vol. 7, no. 02, pp. 97–108, Aug. 2003. *(Cited on page 94)*

[112] A. Hunt, M. M. Wanderley, and R. Kirk, "Towards a model for instrumental mapping in expert musical interaction," in *International Computer Music Conference*, Berlin, Germany, 2000, pp. 209–212. *(Cited on page 81)*

[113] A. Jensenius, "Action-sound: Developing Methods and Tools to Study Music-related Body Movement," Ph.D. dissertation, Department of Musicology, University of Oslo, Jul. 2007. *(Cited on pages 62 and 68)*

[114] A. R. Jensenius, T. Kvifte, and R. I. Godøy, "Towards a gesture description interchange format," in *Conference on New Interfaces for Musical Expression*, Paris, France, 2006, pp. 176–179. *(Cited on pages 81, 120, and 121)*

[115] A. R. Jensenius, A. Camurri, N. Castagné, E. Maestre, J. Malloch, D. McGilvray, D. Schwarz, and M. Wright, "Panel: the need of formats for streaming and storing music-related movement and gesture data," in *International Computer Music Conference*, Copenhagen, Denmark, 2007, pp. 13–16. *(Cited on page 20)*

[116] A. R. Jensenius, M. M. Wanderley, R. I. Godøy, and M. Leman, "Musical Gestures: Concepts and Methods in Research," in *Musical gestures: Sound, movement, and meaning*, R. I. Godøy and M. Leman, Eds. Routledge, 2010. *(Cited on page 86)*

[117] J.-M. Jot and O. Warusfel, "A real-time spatial sound processor for music and virtual reality applications," in *International Computer Music Conference*, Banff, Canada, 1995, pp. 294–295. *(Cited on page 32)*

[118] G. Kendall, N. Peters, and M. Geier, "Towards an Interchange Format for Spatial Audio Scenes," in *International Computer Music Conference*, Belfast, Ireland, 2008, pp. 295–296. *(Cited on page 31)*

[119] A. Kendon, *Gesture: Visible Action As Utterance*. Cambridge University Press, 2004. *(Cited on page 63)*

[120] D. Kim-Boyle, "Sound Spatialization with Particle Systems," in *International Conference on Digital Audio Effects*, Naples, Italy, 2004, pp. 65–68. *(Cited on page 17)*

[121] ——, "Spectral Spatialization - an Overview," in *International Computer Music Conference*, Belfast, Ireland, 2008. *(Cited on pages 17, 33, 50, and 120)*

[122] M. Kuuskankare and M. Laurson, "Expressive Notation Package - an Overview," in *5th International Symposium on Music Information Retrieval*, Barcelona, Spain, 2004. *(Cited on page 7)*

[123] F. Lacquaniti, C. Terzuolo, and P. Viviani, "The law relating the kinematic and figural aspects of drawing movements," *Acta psychologica*, vol. 54, no. 1-3, pp. 115–130, Oct. 1983. *(Cited on page 66)*

[124] O. Laske, "Composition theory in Koenig's project one and project two," *Computer Music Journal*, vol. 5, no. 4, pp. 54–65, Dec. 1981. *(Cited on pages 2 and 24)*

[125] M. Laurson, "Patchwork : A Visual Programming Language and some Musical Applications," Ph.D. dissertation, Sibelius Academy, Helsinki, 1996. *(Cited on page 5)*

[126] M. Laurson and M. Kuuskankare, "Extensible Constraint Syntax Through Score Accessors," in *Journées d'Informatique Musicale*, Paris, France, 2005, pp. 27–32. *(Cited on pages 7 and 8)*

[127] M. Laurson, M. Kuuskankare, and V. Norilo, "An Overview of PWGL, a Visual Programming Environment for Music," *Computer Music Journal*, vol. 33, no. 1, pp. 19–31, Mar. 2009. *(Cited on page 7)*

[128] V. Lazzarini, "Extensions to the Csound Language: from User-Defined to Plugin Opcodes and Beyond." in *Linux Audio Conference*, Karlsruhe, Germany, Nov. 2005, pp. 13–20. *(Cited on page 41)*

[129] M. Leman, *Embodied Music Cognition and Mediation Technology.* The MIT Press, 2007. *(Cited on pages 19 and 89)*

[130] H. B. Lincoln, "Uses of the Computer in Music Composition and Research," *Advances in Computers*, vol. 12, pp. 73–114, Dec. 1972. *(Cited on page 1)*

[131] E. Lindemann, M. Starkier, and F. Dechelle, "The IRCAM Musical Workstation: Hardware Overview and Signal Processing Features." in *International Computer Music Conference*, Glasgow, Scotland, 1990, pp. 132–135. *(Cited on page 32)*

[132] T. Lossius, "Sound - Space - Body: Reflections on Artistic Practice," Ph.D. dissertation, Bergen National Academy of the Arts, Bergen, Norway, Feb. 2007. *(Cited on pages 30 and 130)*

[133] A. Luciani, M. Evrard, D. Couroussé, N. Castagné, C. Cadoz, and J.-L. Florens, "A basic gesture and motion format for virtual reality multisensory applications," in *Conference on Computer Graphics Theory and Applications*, Setubal, Portugal, 2006, pp. 349–356. *(Cited on pages 20 and 68)*

[134] S. Luque, "The Stochastic Synthesis of Iannis Xenakis," *Leonardo Music Journal*, vol. 19, no. 4, pp. 77–84, Dec. 2009. *(Cited on page 2)*

[135] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries." *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993. *(Cited on page 100)*

[136] J. Malloch, "A Framework and Tools for Mapping of Digital Musical Instruments," Ph.D. dissertation, McGill University, Dec. 2013. *(Cited on pages 21 and 80)*

[137] J. Malloch, D. Birnbaum, E. Sinyor, and M. M. Wanderley, "Towards a new conceptual framework for digital musical instruments," in *International Conference on Digital Audio Effects*, Montreal, Canada, 2006, pp. 49–52. *(Cited on page 84)*

[138] J. Malloch, I. Hattwick, M. Schumacher, A. Picciacchia, M. M. Wanderley, S. Ferguson, I. Van Grimde, S. Breton, S. Trougakos, and P. Bassani. Les Gestes / Gestures. [Online]. Available: http://www.idmil.org/projects/gestes *(Cited on page 87)*

[139] J. Malloch, S. Sinclair, and M. Schumacher. Digital Orchestra Toolbox for MaxMSP. [Online]. Available: http://www.idmil.org/software/digital_orchestra_toolbox *(Cited on pages 21 and 87)*

[140] J. Malloch, S. Sinclair, and M. M. Wanderley, "Distributed tools for interactive design of heterogeneous signal networks," *Multimedia Tools and Applications*, vol. 74, no. 15, pp. 5683–5707, Jul. 2015. *(Cited on page 80)*

[141] J. Malloch and M. M. Wanderley, "The T-Stick: From musical interface to musical instrument," in *Conference on New Interfaces for Musical Expression*, New York, USA, 2007, pp. 66–70. *(Cited on page 81)*

[142] G. Marino, M. Serra, and J. Raczinski, "The UPIC system: Origins and innovations," *Perspectives of New Music*, vol. 31, no. 1, pp. 258–269, Jan. 1993. *(Cited on page 65)*

[143] M. Marshall, J. Malloch, and M. M. Wanderley, "Gesture Control of Sound Spatialization for Live Musical Performance," in *Gesture-Based Human-Computer Interaction and Simulation, Revised Selected Papers*. Springer Berlin Heidelberg, 2007, pp. 227–238. *(Cited on pages 17 and 31)*

[144] O. Mathes. RVBAP = Reverberated VBAP. [Online]. Available: http://impala.utopia.free.fr/pd/patchs/externals_libs/vbap/rvbap.pdf *(Cited on pages 94 and 130)*

[145] J. McCartney, "Rethinking the Computer Music Language: SuperCollider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, Dec. 2002. *(Cited on page 32)*

[146] D. McGilvray, "On The Analysis of Musical Performance by Computer," Ph.D. dissertation, University of Glasgow, 2007. *(Cited on page 68)*

[147] A. McLeran, C. Roads, B. Sturm, and J. Shynk, "Granular Sound Spatialization Using Dictionary-Based Methods," in *Sound and Music Computing Conference*, Berlin, Germany, 2008. *(Cited on pages 49 and 55)*

[148] D. Menzies, "W-panning and O-format, tools for object spatialization," in *22nd International Conference of the Audio Engineering Society*, Espoo, Finland, 2002. *(Cited on pages 42 and 131)*

[149] B. Meudic, "Modélisation de structures rythmiques," DEA Atiam, Ircam - Université d'Aix-Marseille II, Tech. Rep., 2001. *(Cited on page 15)*

[150] MIDI Manufacturers Association, "Complete MIDI 1.0 Detailed Specification Version 96.1 (Second Edition)," Nov. 2001. *(Cited on page 15)*

[151] E. R. Miranda and M. M. Wanderley, *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. AR Editions, Inc, 2006, vol. 21. *(Cited on pages 18, 20, and 69)*

[152] F. R. Moore, "A General Model for spatial processing of sounds," *Computer Music Journal*, vol. 7, no. 3, pp. 6–15, Oct. 1983. *(Cited on pages 30 and 130)*

[153] B. A. Myers, "Visual programming, programming by example, and program visualization: a taxonomy," *ACM SIGCHI Bulletin*, vol. 17, no. 4, pp. 59–66, Apr. 1986. *(Cited on page 6)*

[154] V. Norilo and M. Laurson, "Audio analysis in PWGLSynth," in *International Conference on Digital Audio Effects*, Espoo, Finland, 2008, pp. 47–50. *(Cited on page 7)*

[155] R. Normandeau, "Timbre Spatialisation: The medium is the space," *Organised Sound*, vol. 14, no. 03, pp. 277–285, Dec. 2009. *(Cited on page 17)*

[156] G. Nuono and C. Agon, "Contrôle de la spatialisation comme paramètre musical," in *Journées d'Informatique Musicale*, Marseille, France, 2002, pp. 115–120. *(Cited on pages 16 and 33)*

[157] K. Nymoen and A. R. Jensenius, "A Toolbox for Storing and Streaming Music-Related Data," in *Sound and Music Computing Conference*, Padova, Italy, 2011, pp. 1–4. *(Cited on pages 68, 79, and 91)*

[158] K. Nymoen, S. A. v. D. Skogstad, and A. R. Jensenius, "Soundsaber - a Motion Capture Instrument," in *Conference on New Interfaces for Musical Expression*, Oslo, Norway, 2011, pp. 312–315. *(Cited on pages 91 and 93)*

[159] J. O'Callaghan, "Mimetic Instrumental Resynthesis," *Organised Sound*, vol. 20, no. 2, pp. 231–240, Aug. 2015. *(Cited on pages 22 and 64)*

[160] OpenGL Architecture Review Board, *OpenGL reference manual: the official reference document for OpenGL, release 1.* Addison Wesley Publishing Company, 1992. *(Cited on page 7)*

[161] F. Otondo, "Contemporary trends in the use of space in electroacoustic music," *Organised Sound*, vol. 13, no. 01, pp. 1–5, Apr. 2008. *(Cited on page 16)*

[162] F. Pachet, "The MusES system: an environment for experimenting with knowledge representation techniques in tonal harmony," in *First Brazilian Symposium on Computer Music*, Caxambu, Brasil, 1994. *(Cited on page 4)*

[163] F. Pachet and O. Delerue, "MidiSpace: a temporal constraint-based music spatializer," in *ACM International Conference on Multimedia*, Bristol, United Kingdom, 1998, pp. 351–359. *(Cited on page 32)*

[164] ——, "MusicSpace: a Constraint-Based control system for music spatialization," in *International Computer Music Conference*, Beijing, China, 1999, pp. 272–275. *(Cited on page 16)*

[165] ——, "On-the-Fly Multi Track Mixing," in *109th Convention of the Audio Engineering Society*, Los Angeles, USA, Sep. 2000. *(Cited on page 32)*

[166] S. H. Park, "Elements of Impressionism evoked in Debussy and Ravel's Reflets dans l'eau and Jeux d'eau: The theme of water," Ph.D. dissertation, University of Washington, 2012. *(Cited on page 98)*

[167] N. Peters, S. Ferguson, and S. McAdams, "Towards a Spatial Sound Description Interchange Format (SpatDIF)," *Canadian Acoustics*, vol. 35, no. 3, pp. 64–65, Sep. 2007. *(Cited on pages 31 and 43)*

[168] N. Peters, T. Lossius, and J. C. Schacher, "The Spatial Sound Description Interchange Format: Principles, Specification, and Examples," *Computer Music Journal*, vol. 37, no. 1, pp. 11–22, May 2013. *(Cited on pages 93, 120, and 121)*

[169] ——, "A stratified approach for sound spatialization," in *Sound and Music Computing Conference*, Porto, Portugal, 2009, pp. 219–224. *(Cited on pages 39 and 58)*

[170] N. Peters, G. Marentakis, and S. McAdams, "Current Technologies and Compositional Practices for Spatialization: A Qualitative and Quantitative Analysis," *Computer Music Journal*, vol. 35, no. 1, pp. 10–27, Mar. 2011. *(Cited on page 16)*

[171] T. Place and T. Lossius, "Jamoma: A modular standard for structuring patches in Max," in *International Computer Music Conference*, New Orleans, USA, 2006, pp. 143–146. *(Cited on pages 46 and 80)*

[172] T. R. Poller, "Clarence Barlow's Technique of 'synthrumentation' and its use in Im Januar am Nil," *Tempo*, vol. 69, no. 271, pp. 7–23, Jan. 2015. *(Cited on page 23)*

[173] S. T. Pope, "The Musical Object Development Environment: MODE (Ten years of music software in Smalltalk)." in *International Computer Music Conference*, Aarhus, Denmark, 1994, pp. 241–242. *(Cited on page 4)*

[174] L. Pottier, "Dynamical spatialisation of sound. HOLOPHON: a graphical and algorithmical editor for Sigma 1," in *International Conference on Digital Audio Effects*, Barcelona, Spain, 1998. *(Cited on pages 16 and 32)*

[175] M. Puckette, "Max at Seventeen," *Computer Music Journal*, vol. 26, no. 4, pp. 31–43, 2002. *(Cited on pages 5 and 87)*

[176] ——, "Combining Event and Signal Processing in the MAX Graphical Programming Environment," *Computer Music Journal*, vol. 15, no. 3, pp. 68–77, Oct. 1991. *(Cited on page 32)*

[177] ——, "Pure Data: another integrated computer music environment," in *Second Intercollege Computer Music Concerts*, Tachikawa, Japan, 1996, pp. 37–41. *(Cited on page 32)*

[178] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456–466, Jun. 1997. *(Cited on pages 30 and 130)*

[179] ——, "Uniform spreading of amplitude panned virtual sources," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, USA, 1999, pp. 1–4. *(Cited on page 42)*

[180] C. Ramakrishnan, J. Goßmann, and L. Brümmer, "The ZKM Klangdom," in *Conference on New Interfaces for Musical Expression*, Paris, France, 2006, pp. 140–143. *(Cited on pages 32 and 46)*

[181] C. Ramstein, "Analyse, Représentation et Traitement du Geste Instrumental," Ph.D. dissertation, Institut National Polytechnique de Grenoble, 1991. *(Cited on pages 19, 63, 64, 66, 72, and 78)*

[182] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, Aug. 1987. *(Cited on page 50)*

[183] J. C. Risset, "Pitch Control and Pitch Paradoxes Demonstrated with Computer-Synthesized Sounds," *Journal of the Acoustical Society of America*, vol. 46, no. 1A, p. 88, Jul. 1969. *(Cited on page 113)*

[184] W. Ritsch, "Robotic Piano Player Making Pianos Talk," in *Sound and Music Computing Conference*, Padova, Italy, 2011, pp. 1–6. *(Cited on pages 23 and 98)*

[185] C. Roads, *Microsound*. The MIT Press, Sep. 2001. *(Cited on pages 16 and 49)*

[186] D. Rocchesso, "The Ball within the Box: a sound-processing metaphor," *Computer Music Journal*, vol. 19, no. 4, pp. 47–57, Dec. 1995. *(Cited on page 130)*

[187] X. Rodet, P. Cointe, and E. Starkier, "Formes: composition et ordonnancement de processus," IRCAM—Centre Pompidou, Tech. Rep., 1985. *(Cited on page 4)*

[188] C. L. Salter, M. Baalman, and D. Moody-Grigsby, "Between Mapping, Sonification and Composition: Responsive Audio Environments in Live Performance," in *Computer Music Modeling and Retrieval*. Springer Berlin Heidelberg, Aug. 2007, pp. 246–262. *(Cited on pages 18 and 80)*

[189] J. C. Schacher, "Gesture control of sounds in 3D space," in *Conference on New Interfaces for Musical Expression*, New York, USA, 2007, pp. 358–362. *(Cited on page 17)*

[190] J. C. Schacher and P. Kocher, "Ambisonics Spatialization Tools for Max/MSP." in *International Computer Music Conference*, New Orleans, USA, 2006, pp. 274–277. *(Cited on pages 32 and 42)*

[191] A. A. Scharine and T. R. Letowski, "Auditory Conflicts and Illusions," in *Helmet-mounted displays: Sensation, perception and cognition issues*, C. E. Rash, M. B. Russo, T. R. Letowski, and E. T. Schmeisser, Eds. US Army aeromedical Research Laboratory, 2009. *(Cited on pages 114 and 115)*

[192] M. Schumacher, "Ab-Tasten: Atomic sound modeling with a computer-controlled grand piano," in *The OM Composer's Book: Volume 3*, J. Bresson, G. Assayag, and C. Agon, Eds. Éditions Delatour France / IRCAM—Centre Pompidou, in press. *(Cited on pages 69 and 122)*

[193] ——. OM-Pursuit: Dictionary-Based Sound Modelling in Computer-Aided Composition. [Online]. Available: http://www.idmil.org/software/OM-Pursuit *(Cited on page 105)*

[194] ——. OM-SoX: Multichannel Audio Manipulation and Functional Batch Processing in Computer-Aided Composition. [Online]. Available: http://www.idmil.org/software/OM-SoX *(Cited on page 104)*

[195] M. Schumacher and J. Bresson, "Compositional Control of Periphonic Sound Spatialization," in *2nd International Symposium on Ambisonics and Spherical Acoustics*, Paris, France, 2010. *(Cited on pages 26, 46, 93, and 122)*

[196] ——, "Spatial Sound Synthesis in Computer-Aided Composition," *Organised Sound*, vol. 15, no. 03, pp. 271–289, Dec. 2010. *(Cited on pages 62, 94, 96, 107, 113, and 122)*

[197] D. Schwarz and B. Hackbarth, "Navigating Variation: Composing for Audio Mosaicing," in *International Computer Music Conference*, Ljubljana, Slovenia, 2012. *(Cited on page 22)*

[198] D. Schwarz and M. Wright, "Extensions and Applications of the SDIF Sound Description Interchange Format," in *International Computer Music Conference*, Berlin, Germany, 2000, pp. 481–484. *(Cited on page 121)*

[199] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, Dec. 1990. *(Cited on page 102)*

[200] T. B. Sheridan, "Musings on Music Making and Listening: Supervisory Control and Virtual Reality," *Proceedings of the IEEE*, vol. 92, no. 4, pp. 601–605, Apr. 2004. *(Cited on page 84)*

[201] R. Silver, "Digital composition of a mosaic image," Patent, 2000. *(Cited on page 102)*

[202] M. M. Smyth and A. W. Wing, *Psychology of Human Movement*. Academy Press, 1984. *(Cited on page 65)*

[203] L. Spiegel. The Music Mouse Manual. [Online]. Available: http://retiary.org/ls/progs/mm_manual/mouse_manual.html *(Cited on page 5)*

[204] G. L. Steele, Jr and R. P. Gabriel, "The Evolution of Lisp." *ACM Sigplan Notices*, vol. 28, no. 3, pp. 231–270, Mar. 1993. *(Cited on page 4)*

[205] K. Stockhausen, *Stockhausen on Music: Lectures and Interviews*, R. Maconie, Ed. London: Marion Boyars, 1989. *(Cited on pages 17 and 30)*

[206] K. Stockhausen and J. Kohl, "Clavier Music 1992," *Perspectives of New Music*, vol. 31, no. 2, pp. 136–149, Jul. 1993. *(Cited on page 113)*

[207] M. Stroppa, "Paradigms for the high level musical control of digital signal processing," in *International Conference on Digital Audio Effects*, Verona, Italy, 2000. *(Cited on pages 34, 36, 40, and 81)*

[208] B. L. Sturm, "Adaptive Concatenative Sound Synthesis and Its Application to Micromontage Composition." *Computer Music Journal*, vol. 30, no. 4, pp. 46–66, Dec. 2006. *(Cited on pages 23 and 102)*

[209] B. L. Sturm, C. Roads, A. McLeran, and J. J. Shynk, "Analysis, Visualization, and Transformation of Audio Signals Using Dictionary-Based Methods," *Journal of New Music Research*, vol. 38, no. 4, pp. 325–341, Dec. 2009. *(Cited on pages 101 and 120)*

[210] H. Taube, "Common Music: A music composition language in Common Lisp and CLOS," *Computer Music Journal*, vol. 15, no. 2, pp. 21–32, Jul. 1991. *(Cited on page 4)*

[211] D. Teruggi, "Technology and musique concrète: the technical developments of the Groupe de Recherches Musicales and their implication in musical composition," *Organised Sound*, vol. 12, no. 03, pp. 213–231, Dec. 2007. *(Cited on page 16)*

[212] J.-B. Thiebaut, P. G. T. Healey, and N. B. Kinns, "Drawing Electroacoustic Music," in *International Computer Music Conference*, Belfast, Ireland, 2008. *(Cited on page 66)*

[213] T. Todoroff, C. Traube, and J. Ledent, "NeXTStep Graphical Interfaces to Control Sound Processing and Spatialization Instruments," in *International Computer Music Conference*, Thessaloniki, Greece, 1997, pp. 325–328. *(Cited on pages 16 and 32)*

[214] D. Topper, M. Burtner, and S. Serafin, "Spatio-Operational Spectral (S.O.S.) Synthesis," in *International Conference on Digital Audio Effects*, Hamburg, Germany, 2002. *(Cited on page 33)*

[215] R. Torchia and C. Lippe, "Techniques for Multi-Channel Real-Time Spatial Distribution Using Frequency-Domain Processing," in *Conference on New Interfaces for Musical Expression*, Hamamatsu, Japan, 2004, pp. 116–119. *(Cited on page 49)*

[216] C. Truchet, "Contraintes, recherche locale et composition assistée par ordinateur," Ph.D. dissertation, Université Paris 7, 2004. *(Cited on page 15)*

[217] C. Truchet, G. Assayag, and P. Codognet, "OMClouds, a heuristic solver for musical constraints," in *5th Metaheuristics International Conference*, Kyoto, Japan, 2003, pp. 1–6. *(Cited on page 8)*

[218] T. Tsandilas, C. Letondal, and W. E. Mackay, "Musink: Composing Music through Augmented Drawing," in *SIGCHI Conference on Human Factors in Computing Systems*, Boston, USA, 2009, pp. 819–828. *(Cited on pages 19 and 66)*

[219] D. Van Nort, M. M. Wanderley, and P. Depalle, "Mapping Control Structures for Sound Synthesis: Functional and Topological Perspectives." *Computer Music Journal*, vol. 38, no. 3, pp. 6–22, Sep. 2014. *(Cited on page 80)*

[220] F. J. Varela, E. Thompson, and E. Rosch, *The Embodied Mind: Cognitive Science and Human Experience.* The MIT Press, 1992. *(Cited on page 62)*

[221] V. Verfaille, U. Zölzer, and D. Arfib, "Adaptive digital audio effects (A-DAFx): a new class of sound transformations," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 5, pp. 1817–1831, Sep. 2006. *(Cited on page 55)*

[222] H. Vinet, P. Herrera, and F. Pachet, "The cuidado project," in *ISMIR conference*, Paris, France, 2002, pp. 197–203. *(Cited on page 124)*

[223] M. M. Wanderley, "Interaction Musicien-Instrument: Application au contrôle gestuel de la synthèse sonore," Ph.D. dissertation, 2001. *(Cited on page 63)*

[224] M. M. Wanderley and M. Battier, Eds., *Trends in gestural control of music.* IRCAM—Centre Pompidou, 2000. *(Cited on page 18)*

[225] M. M. Wanderley and P. Depalle, "Gestural Control of Sound Synthesis," *Proceedings of the IEEE*, vol. 92, no. 4, pp. 632–644, Apr. 2004. *(Cited on pages 63 and 81)*

[226] M. M. Wanderley, N. Schnell, and J. Rovan, "Escher - Modeling and Performing 'Composed Instruments' in Real-Time," in *IEEE International Conference on Systems, Man and Cybernetics (SMC'98)*, San Diego, USA, 1998, pp. 1080–1084. *(Cited on pages 81 and 84)*

[227] M. M. Wanderley, B. Vines, N. Middleton, C. Mckay, and W. Hatch, "The Musical Significance of Clarinetists' Ancillary Gestures: An Exploration of the Field," *Journal of New Music Research*, vol. 34, no. 1, pp. 97–113, Jun. 2005. *(Cited on page 64)*

[228] J. Wann, I. Nimmo-Smith, and A. M. Wing, "Relation between velocity and curvature in movement: Equivalence and divergence between a power law and a minimum-jerk model." *Journal of Experimental Psychology: Human Perception and Performance*, vol. 14, no. 4, p. 622, Nov. 1988. *(Cited on page 66)*

[229] O. Warusfel and N. Misdariis, "Directivity synthesis with a 3D array of loudspeakers: application for stage performance," in *International Conference on Digital Audio Effects*, Limerick, Ireland, 2001. *(Cited on page 58)*

[230] E. Wenger. (1997) Metasynth. [Online]. Available: http://www.ircam.fr/produits-real/logiciels/metasynth-e.html *(Cited on page 65)*

[231] D. Wessel and M. Wright, "Problems and Prospects for Intimate Musical Control of Computers," *Computer Music Journal*, vol. 26, no. 3, pp. 11–22, Oct. 2002. *(Cited on page 81)*

[232] S. Wilson, "Spatial Swarm Granulation," in *International Computer Music Conference*, Belfast, Ireland, 2008. *(Cited on pages 17, 32, 50, 55, and 120)*

[233] M. Wright, "Open Sound Control: an enabling technology for musical networking," *Organised Sound*, vol. 10, no. 3, pp. 193–200, Dec. 2005. *(Cited on pages 15, 48, and 123)*

[234] M. Wright, A. Freed, and A. Momeni, "OpenSound Control: State of the Art 2003," in *Conference on New Interfaces for Musical Expression*, Montreal, Canada, 2003, pp. 153–160. *(Cited on page 65)*

[235] M. Wright, A. Chaudhary, A. Freed, S. Khoury, and D. Wessel, "Audio Applications of the Sound Description Interchange Format Standard," *Audio Engineering Society Convention 107*, 1999. *(Cited on page 15)*

[236] I. Xenakis, Formalized Music: Thought and Mathematics in Composition. Bloomington and London: Indiana University Press, 1971. *(Cited on page 2)*

[237] ——, "Free Stochastic Music by Computer," in *Formalized Music: Thought and Mathematics in Music (Revised Edition)*. New York: Pendragon Press, 1992, pp. 131–154. *(Cited on page 2)*

[238] P. Zahorik, D. Brungart, and A. Bronkhorst, "Auditory distance perception in humans: A summary of past and present research," *Acta Acustica united with Acustica*, vol. 91, no. 3, pp. 409–420, 2005. *(Cited on page 116)*

[239] D. Zicarelli, "M and Jam Factory," *Computer Music Journal*, vol. 11, no. 4, pp. 13–29, Dec. 1987. *(Cited on page 5)*

[240] A. Zils and F. Pachet, "Musical mosaicing," in *International Conference on Digital Audio Effects*, Limerick, Ireland, 2001. *(Cited on page 102)*