Marlon Schumacher & Marcelo M. Wanderley
Pre-publication Draft

Full article published in Journal of New Music Research
Volume 46, Issue 1: Interactive Composition, Pages 87-101, 2017
https://doi.org/10.1080/09298215.2016.1254662

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

anonymized

Abstract

In this article we discuss physical gestures from the perspective of computer-aided composition and propose solutions for the integration of gesture signals as musical materials into these environments. After presenting background and motivations we review related works and describe design guidelines for our developments. 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 case studies are presented for the use of this library for synthesis of symbolic score and spatial audio from gesture recordings of dance and instrumental performances.

1 Background and Motivation

1.1 Origins and applications of computer-aided composition

Environments for Computer-Aided Composition (CAC) are traditionally concerned with the representation and manipulation of symbolic musical materials, such as notes, rhythms, harmonies, etc. Dating back to the early experiments of Hiller in the late 1950s [1], composers like Xenakis (1962), or Koenig (1964), used specialized computer programs for the genesis of musical structures, mostly implementing combinatoric or probabilistic techniques, in the tradition of algorithmic composition [2]. 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” [3]. 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 which has led to an extended conception of CAC systems, integrating several aspects of a work in a single programming framework, such as synthesis of electronic sounds [4], spatialization [5], and orchestration [6].

Recent trends in cognitive psychology have stressed the importance of physical interaction for human expression and creativity, see. e.g. theories of *enaction* [7], *embodied cognition* [8], or *motor-mimetic cognition* [9]. Findings in cognitive linguistics, for instance, indicate that all human thought, including abstract concepts, are based on conceptual metaphors and schemes derived from physical experience with the world and that reasoning is embodied and mostly unconscious [10, 11]. 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 [12]. Cadoz, for instance, stated that sound portrays indices specifically related to physical gesture, and that gesture is both concrete and symbolic, thus music composition can be seen as “...an implicit and indirect composition of the instrumental gesture” [13]. Similar to how sound composition in CAC systems can be carried out using both abstract and sample-based approaches, we suggest to complement the generative-algorithmic possibilities for creating musical structures with the use of recorded, physical gestures.

1.2 Notions of “Gesture”

Since the term “gesture” is often used in ambiguous ways with partially overlapping and even contradictory meanings [14] it is sensible to precise our notion of gesture used in this article. Indeed, some researchers consider the term “gesture” misleading and avoid its use at all [15] — a notion that, if applied to other research terms, would likely dramatically reduce our vocabulary. Within our framework, we take a phenomenological viewpoint, considering gestures as spatio-temporal morphologies[1] of physical variables that can be sensed and represented in the form of digital signals [16]. This may include sound-producing actions (*excitation* and *modification*), ancillary movements (*phrasing* or *entrained* movements), sound-accompanying movements (*sound tracing* and *mimicking*), and communicative movements (*gesticulations*) [17, 13, 18]. Previous research on integration of sound into CAC has shown that digital signals can be represented, abstracted into

[1]The term ”morphology” is used here in the sense of forms or structures.

higher-level objects, and embedded into symbolic compositional contexts [19]. This integration has proven a fruitful direction, leading to interesting compositional approaches and applications [20, 21].

1.3 State of the Art for Gestures in CAC

Thanks to the proliferation of inexpensive sensing 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 and custom-made input devices. A closer look at their use within the field of computer music, however, reveals a curious dichotomy: although being widely used in the context 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 [22, 23, 24]. The importance of gesture for human expression seems to be underrepresented and not reflected appropriately in current computer-aided composition systems.

1.4 Extended Possibilities and Applications

The integration of gesture data within the deferred-time context of computer-aided composition allows to envisage interesting new scenarios and applications. For instance, it would allow composers to physically express and capture their musical ideas using gestural input devices (a historic example is the UPIC system, discussed in more detail in the next section). Another example would be the recording of gesture data captured from instrumental performers, for instance from performances of compositional sketches. Since gestures carry unique movement characteristics of individual performers, gesture recordings can serve as materials for the creation of personalized music materials [25], similar to the practice of composers making use of recorded sound material from individual performers. In the case of live musical performance with digital musical instruments (DMIs), it would allow composers to record gesture data and use these recordings as materials for the development of other aspects of a piece. The possibility of storage, representation and arrangement of gesture data might allow the composition of prescriptive gestural performance scores (describing performer gestures, rather than the sounding results), and might also be of interest beyond compositional applications, e.g. for performer studies or instrument design.

1.5 Structure of this article

This article is organized as follows: section 2 describes related works and points out differences between the use of gestural input devices for human-computer-interaction (HCI) versus gesture signals as materials for composition. In section 3 we discuss requirements and design guidelines for a software tool for representation, processing, and mapping of gestures. Section 4 will introduce the library OM-Geste and describe its general functionalities with a particular focus on the concept of mapping in the context of computer-aided composition. We follow with a validation of our framework with two case studies, using gesture data captured from dance and instrumental performances (section 5), before concluding the article with a discussion (section 6).

2 Related Works

To this date there have been relatively few works involving physical gestures in the context of CAC. Most developments have focused on freehand drawing and sketching applications, such as entering control data with a digital stylus. A historic example is the *UPIC*[2] system, developed at the "Centre d'Etudes de Mathématique et Automatique Musicales" (CeMaMu) with its first prototype in 1977 on a SOLAR mainframe computer [26]. 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 version was introduced which allowed controlling sound synthesis by moving the stylus on the tablet in real time, 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" [27], "HighC" [28] and "Iannix" [29]. 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 [30].

Besides these graphical input applications and idiosyncratic approaches, the field of CAC has been largely isolated from the world of physical gestures.

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

More recently, there has been a renewed interest from an HCI perspective, based on the observation that even computer-literate composers still use physical means such as pen and paper to sketch their initial creative impulses, thereby reducing cognitive load compared to the use of conventional computer interfaces [31]. 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 [32, 33]. Garcia et al. presented an artistic application, 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 [22]. Other applications in this direction include the use of drawing gestures entered via mobile devices and gaming controllers for specification of spatialization parameters [34, 24]. These works show interesting new interaction possibilities, however their focus is on specific applications rather than a generic framework.

One work which comes closer to our objectives was presented by Ramstein and Cadoz [13, 35]. The two main differences are that their system is intended exclusively for *instrumental* (i.e. *sound-producing*) gestures, and that it did not aim to integrate these data into compositional environments for producing direct music representations (such as symbolic scores or audio). Nevertheless, a number of aspects related to coding, representation and manipulation of gestures [36] are addressed, which are relevant for both our works. The authors use two alternative representations of gesture data, an *internal* one and *external* one: the former is close to the signal and intended for transmission and storage (i.e. coding). The latter is based on higher-level descriptors ("gestual attributes", e.g. sharpness of attack) and is intended for user manipulation. It is also proposed that representations should not be fixed, but extendable by the user, for instance by adding new gestual [sic] attributes. The authors further stress that for compositional work, a temporal segmentation of continuous data streams into discrete entities ("gestual events") is necessary, which we also consider an important requirement (cf. section 3.2). In terms of user interfaces, different views for different scales are proposed: a zoomed-in view for editing details of a single segment ("articulation"), and a zoomed-out view, for comparing multiple segments and their relationships ("sequence").

It can be seen that while all of the cited works allow to apply gesture in compositional contexts, they each focus either on specific types of gestures, devices or applications, and none of them offers a complete framework for integrating gestures into symbolic compositional processes. Although developed in different eras and contexts, there are conceptual similarities between the graphical-input systems in the tradition of UPIC, and the more recent developments emplyoing advanced sensing and interface technologies. 1) They employ graphical input devices as a means for entering data to the composition environment. 2) They focus exclusively on drawing gestures, which (as any other human motion) are typified by morphological characteristics resulting from physical constraints of the mechanical system performing the gesture [37]. 3) They focus on the "trace" of gestures: taking the example of a hand drawing on smart paper, the gesture producing the drawing cannot be recovered from the figure; it can thus not be considered a representation of the gesture itself, but rather its result or effect. The work of Ramstein and Cadoz is more similar to ours in that they are interested in the spatio-temporal morphology of gestures. Their application context, however, is memorization, treatment and restitution for instrumental performance, rather than its integration as materials for symbolic composition.

3 Design Guidelines

In this section we describe requirements for gesture integration in CAC and resulting design guidelines for the software library presented later on. The requirements we identified can be classified into three domains: 1) description and encoding, 2) abstraction and representation, 3) mapping and synthesis.

3.1 Description and Coding

A generic framework for composition with gesture materials should allow users to work with gesture data regardless of sensing device or method (e.g. motion capture systems, commercial controllers, custom-made input devices, etc.). According to Luciani et al. gesture data is characterized by its variable *geometric* dimensionality (i.e. the dimensionality of the space in which the gesture evolves) and *structural* dimensionality (i.e. the functional organization of degrees-of-freedom) [38]. Ideally, it should be possible to describe and encode gesture data consistently and coherently, regardless of dimensionality, type, or resolution. Since these requirements overlap with interests of the musical gesture research community, we can draw from existing work on the development of gesture description formats, see e.g. [38, 39, 40]. One of these research efforts is the GDIF (Gesture Description Interchange Format) initiative [15]. Originally intended for description of sound-motion relationships, it is based on a set of standardized descriptors of variable dimensionality[3] and puts forward the idea of structuring gesture data into multiple heterogenous layers sharing a common timeline. GDIF has been conceived as a sister format to SDIF (Sound Description Interchange Format) [41] which was originally conceived for storing and exchanging sound description data. SDIF is based on time-tagged frames containing data of variable dimensionality, organized into heterogenous streams. It has a number of useful properties for storing gesture-related data, such as the possibility of defining new descriptors on the fly, and it has already been successfully used in our previous computer-aided composition projects for coding of sound and spatialization data [42, 43]. SDIF is well-suited for the multi-layered descriptor scheme proposed by GDIF and is supported by a growing number of computer music environments, including OpenMusic [42]. Accordingly, we adopt the approach of describing gestures according to GDIF specifications and using SDIF as the data container [44].

3.2 Representation

In CAC environments, users typically manipulate symbolic representations rather than concrete low-level data, and consequently, gesture signals need to be abstracted into higher-level representations in order to be integrated in compositional processes. We propose a gesture composition system should provide functionalities for three essential tasks:

- **transcoding** of low-level sensor signals into higher-level gesture descriptors, e.g. deriving orientation or velocity from accelerometer measurements
- **segmentation** of continuous data into discrete units that can be integrated as discrete objects into symbolic compositional processes [19, 45];
- **abstraction** of gesture data into symbolic representations, e.g. reducing a time-domain waveform to a break point function consisting of a few number of points

Besides functionalities for transcoding, segmentation and abstraction, it is important to allow for conversions between alternative representations.[4] The representation of an underlying domain object (such as gesture data) determines affordances and provides an interaction context for manipulation

[3]A list of descriptors can be found online: `http://xdif.wiki.ifi.uio.no/Data_types`

[4]See also [46] for a similar discussion on representations of audio signals.

[47] which can be strongly dependent on individual composer's conceptions and needs.

3.3 Mapping and Synthesis

The ultimate goal of any compositional system is to produce a musical output, such as a symbolic score, sound or other musical media. While in CAC this musical output can be directly generated using generative-algorithmic means, 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 [48].[5] While in the design of digital musical instruments the notion of mapping typically refers to the association of gesture variables to sound synthesis parameters [50], Chadabe pointed out limitations of this model as a structural descriptive [51], which according to Downie, "deflates the awesome power of the algorithmic before it can appear" [52]. 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 into a direct music representation through a formalized process. 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)" [2]. Indirect music representations are non-musical, incomplete, or unordered structures, such as functions, images, gesture data, etc. With this extended definition, two particular differences to the instrumental model become apparent.

The first concerns representations of time. A popular notion in the context of DMI design is the *systems* point of view, conceptualizing a mapping as an "...out-of-time snapshot of input/output control potential", similar to a *flow-chart* graph which is devoid of temporal representations [53]. 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), the manipulation of time itself is a distinct property of offline-composition. Consequently, it is a requirement to integrate temporal specifications as part of a mapping and associate gestures with musical elements of arbitrary temporal scale and structure.

A second aspect concerns the gesture representation and its interface exposing variables for mapping. In DMI design it is a common approach to use multi-layered representations of gesture data, e.g. by converting raw

[5] Indeed, there are many historic examples for mapping in composition, spanning from medieval to contemporary music practices [49].

sensor measurements into a set of intermediate parameters [54]. These intermediate parameters may be derived from signal features [55], perceptual models [56], interaction metaphors [57], etc. and are typically created in a preliminary stage than the transdomain mapping. The situation in compositional contexts is different, since semantics of gesture variables might not be known in advance, and new gesture layers may be developed as part of the compositional process. This can be related to the notion of the design of mappings as a tradeoff between simplifying the mapping versus the structure of the gesture description [18]. Whether complexity is created as part of the gesture description or the mapping should not be biased through the system. Rather, it should allow for manipulating both gesture description and mapping in an integrated workflow.

4 The library OM-Geste

This section presents our implementation of the requirements and design guidelines described in section 3, as the library OM-Geste for the CAC environment OpenMusic (OM).[6]

4.1 Segmentation and Abstraction

4.1.1 Gesture-Array

In OM-Geste, gesture signals are described using the GDIF specification and imported/exported into the environment via SDIF files (cf. section 3.1). 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 of the object consists of a number of *gesture-streams*, each representing a gesture descriptor (e.g. 3D position in cartesian coordinates). A *gesture-stream* consists of a group of *substreams*, each describing the evolution of a scalar variable over time (one degree-of-freedom). Figure 1 illustrates the internal structure of a *gesture-array* containing two *gesture-streams*: 3D-position data and absolute velocity.[7]

As discussed in section 3.2 the continuous data in a *gesture-array* needs to be segmented and abstracted into entities which are closer to symbolic representations that can be integrated into compositional processes. In compositional environments – conceived as open and neutral systems – the

[6]OM-Geste is open-source software, available at `https://github.com/marleynoe/OM-Geste`

[7]The hierarchical structure of the *gesture-array* object can be compared to the "internal representation", in which *units* contain *channels* containing *lanes* [36].

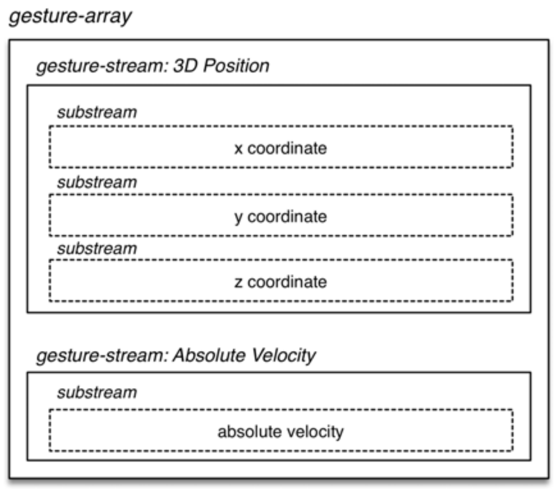

Figure 1: Structure of a *gesture-array* object. On the top level, two *gesture-streams* representing gesture descriptors (*3D Position* and *Absolute Velocity*). Each *gesture-stream* consists of a number of *substreams*.

semantics of gestures and the criteria for segmentation cannot be presupposed. For instance, criteria can be based on symbolic information and meta data which is not contained in the data itself. Moreover, the segmentation of gesture signals might itself be a compositional parameter. Therefore, rather than providing pre-defined methods for segmentation, users can determine a time structure externally (a sequence of temporal locations in seconds) through arbitrary processes or data, which can then be provided to our system for segmentation of the gesture signals. Within this framework we can distinguish three basic approaches:

- **external** segmentation: based on referential external data, for example a media file, meta data, 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 a user-defined procedure (e.g. compositional structure or process)

4.1.2 Gesture-Models

We developed a new object, titled *gesture-model*, which serves as a structured container for *gesture-streams*, segmented into temporal units. A *gesture-*

model is built from a *gesture-array* and a time-structure, i.e. a sequence of temporal locations. During this process, two operations are carried out: First, the individual *gesture-streams* are segmented into temporal units, titled *gesture-segments*. Second, the *gesture-segments* are converted into OM objects, based on the dimensionality of the descriptor.

Visually, a *gesture-model* provides a tabulated 2D-interface in which the OM objects are organized. Rows represent *gesture-streams* and columns represent temporal segments. Individual objects can be visualized and edited through an associated OM graphical editor. Note, that for each *gesture-stream* a dedicated inlet/outlet is created through which the corresponding data can be accessed or modified.

The process of creating a *gesture-model* object from gesture recordings is illustrated in Figure 2. Gesture signals are imported from an SDIF file *(a)* and converted into a *gesture-array* object *(b)*. A time-structure is derived from temporal markers which have been set on the waveform of a referential audio file *(c)*. The *gesture-array* is segmented according to the time structure and a *gesture-model* is built, containing OM objects organized into a 2D-table *(d)*. Visible on the right hand side of the Figure are two graphical editors showing 3D and 1D gesture data.[8] Note, that the transcoding process, i.e. the conversion of *gesture-segments* into OM objects is completely invertible. This is an important feature, since depending on the context it might be desirable to represent gesture materials in different ways (c.f. section 3.2). It is also possible to re-segment a *gesture-model* multiple times, for instance to introduce or extract different temporal structures from the same gesture recording.

4.2 Manipulation and Processing

Another important functionality is the possibility of associating gesture data with other musical materials and data in a common structure (cf. sections 3.1, 3.2). This can be useful for including additional streams (e.g. higher-order descriptors derived from analyses of existing streams), or external data a user wishes to associate with gesture data. OM-Geste provides functionalities for extending existing *gesture-models* wth arbitrary data, which are automatically synchronized, segmented and integrated into the tabulated structure of the *gesture-model*. Figure 3 shows an example for adding and synchronizing new streams to a *gesture-model*. First, a *gesture-stream* is extracted, its first derivative calculated and the resulting data

[8]The organization of gesture segments into a larger-scale struture can also be related to the notion of a "formula" in [36].

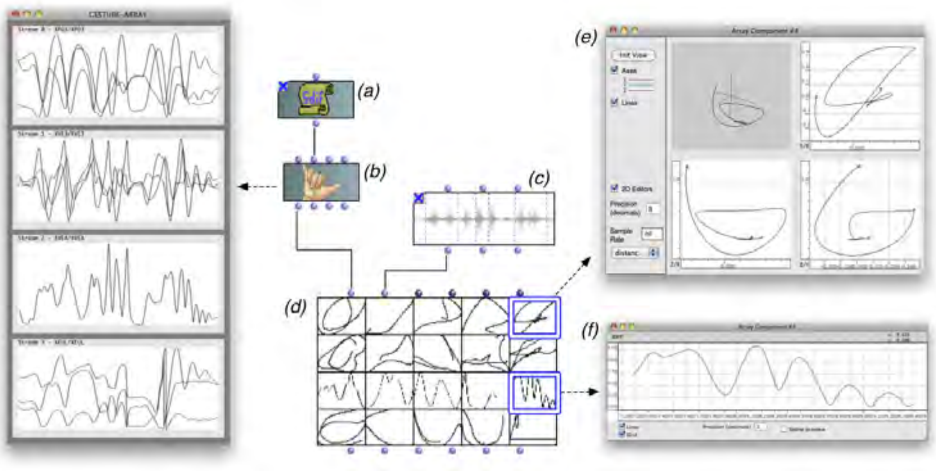

Figure 2: Segmentation, abstraction, and representation of gesture data in OM-Geste. *(a)*: gesture data in an SDIF file is extracted into a *gesture-array* object *(b)*, visible in the editor on the left. *(c)*: a temporal structure, specified via markers in an audio file, is used to segment the *gesture-array* and build a *gesture-model* object *(d)*, the individual segments are converted into OM objects, depending of the dimensionality of the respective gesture descriptor. On the right hand side: a *3D-trajectory* object displayed in its associated editor showing the gesture trajectory in 3D with its 2-dimensional projections *(e)*, a *break point function* object displayed as one-dimensional time-series data in its associated graphical editor *(f)*.

added as a new *gesture-stream (a)*. Subsequently, a number of OM objects (symbolic score, audio file, textual annotations) are supplied as a list and added as individual streams *(b)*. The resulting *gesture-model* containing the new *gesture-streams* is visible at the bottom *(c)*.

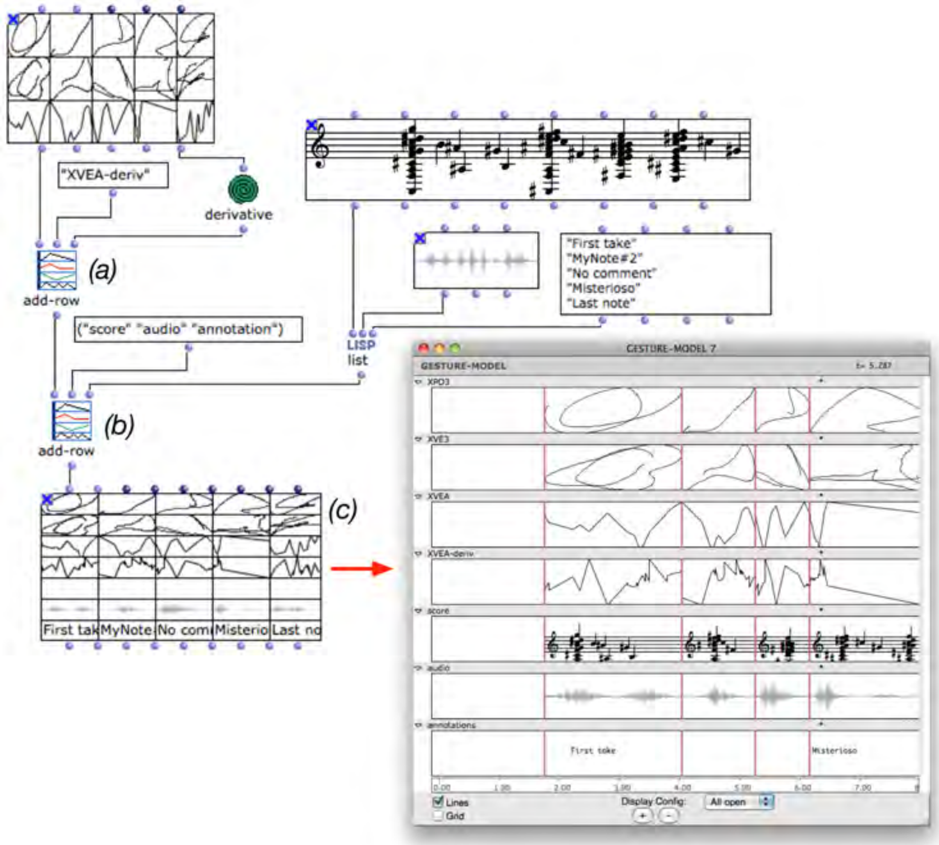

Figure 3: Associating gestures with other musical data in a *gesture-model* object. *(a)*: A stream is extracted, its derivative calculated and appended as a new stream to the *gesture-model* via the function *add-row*. *(b)*: A number of OpenMusic objects (containing a symbolic score, audio file, textual comments) are added as new streams to the *gesture-model*. *(c)*: The new streams are automatically segmented according to the temporal structure of the *gesture-model*.

Working with sensor measurements often involves carrying out signal processing operations, such as conditioning the raw data by removing unwanted noise and jitter as well as applying other types of processing, e.g. for extracting higher-order features (see e.g. [58]). To this end, the library provides a toolbox of functions, including FIR/IIR and non-causal filters, algorithms for feature extraction, statistics, data reduction and optimization. For some of these functions it can be convenient to determine parameter values empirically by manual tweaking (such as finding adequate filter settings for smoothing operations). Thanks to the recent "reactive" extension in OM [59] it is possible to design visual programs which react to user events in real time. This allows for interactive processing, e.g. to visualize the result of applying a filter to a gesture signal by moving a slider. It is also possible to manipulate *gesture-models* via higher-order functions, for instance to apply user-defined algorithms (as OM patches or LISP functions) globally or to selected data fields.

4.3 Mapping as a Program

As discussed in section 3.3 the integration of a mapping system into compositional environments requires a different approach as compared to the design of digital instruments. In CAC contexts, gesture data, generative processes, and manual specifications might be interconnected at different levels in mapping processes and used to specify parameters for objects ranging from symbolic to signal domains. Accordingly, we developed a mapping system which implements a consistent approach for setting arbitrary parameters of any musical object in the environment, independently of structure or scale.

This is realized thanks to advanced programming features in CLOS [60], which allows designing a mapping as a *program*, rather than a snapshot of connections. A mapping therefore benefits from the full expressivity of the underlying programming environment, including abstraction, recursion, control structures, etc. From a user's point of view a mapping is designed like a standard OM visual program (a patch) with the difference of using named *inputs* and *outputs* in order to access gesture data and set synthesis parameters. This design leverages existing skills by providing users with tools and interfaces they are already familiar with. Since mapping programs are standard patches, users can benefit from any feature provided with the OM programming interface, such as copy/paste actions, comments, structuring of subroutines into subpatches, persistent storage, etc. Another advantage is that the use of a visual programming interface for designing mappings offers a higher-level, more intuitive approach by de-emphasizing syntactical issues (as compared to the one-line expressions found in many current tools), making the mapping task more accessible for non-expert programmers [61].

Figure 4 shows an example for the use of the mapping system. The three

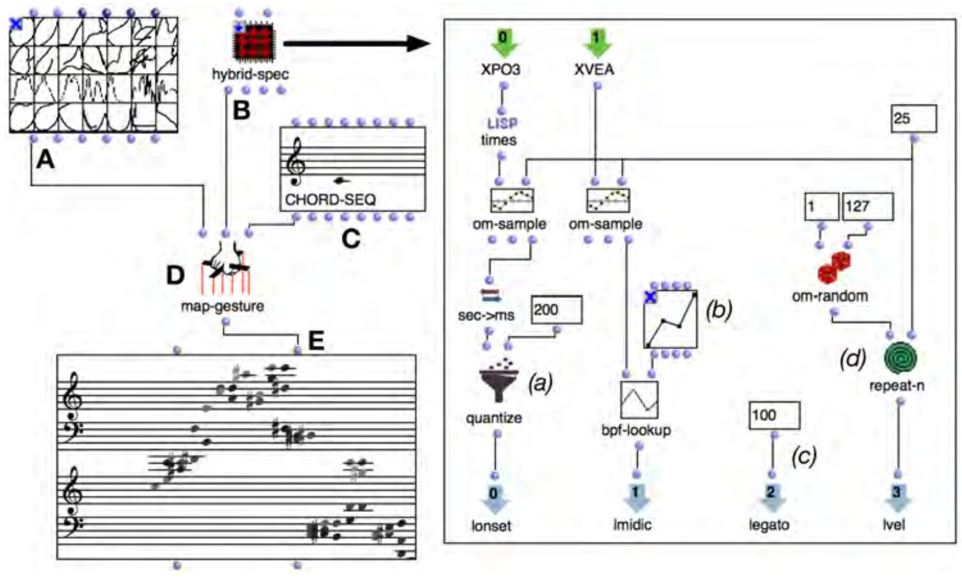

Figure 4: A mapping process as a visual program in OM-Geste. On the left: **A** a *gesture-model* object, **B** an OM patch defining a mapping program, **C** the OM class to produce (*chord-seq* object). **D** The function *map-gesture* performs the mapping for each segment of the *gesture-model* and outputs a list of *chord-seq* instances. **E** the instances gathered into a *multi-seq* object. On the right: The mapping program (OM patch) to set values for the slots of the *chord-seq* instances. Onset times are determined by downsampling the time tags of the *gesture-stream* for 3D-position (XPO3) and quantizing the result to a time grid of 200ms *(a)* . Pitch is determined by downsampling the values of the *gesture-stream* for absolute acceleration (XVEA), and using these for table lookup *(b)*. Durations are indirectly determined as legato factors, set to a constant value of 100 percent *(c)*. Dynamics are specified by an algorithm picking a random value between 1 and 127 for each note *(d)*.

main elements are shown on the left hand side of the Figure. A: a *gesture-model* object, B: an OM patch defining the mapping program, C: an OM class specifying the direct music representation to produce (*chord-seq* object, i.e. a sequence of chords in linear time notation). The function *map-gesture* (D) takes these three elements as input arguments and iterates through the temporal segments in the *gesture-model*, producing instances of the connected OM class with values for its slots specified through the mapping program. E: The resulting *chord-seq* instances are gathered together and displayed in a *multi-seq* object at the bottom of the Figure.

On the right-hand side is a detailed view of the mapping patch "hybrid specs", in which gesture data, symbolic specifications, literals, and algorithms are used for setting parameters of *chord-seq* instances (each containing 25 notes). Inputs (top) are named to specify the *gesture-streams* in the *gesture-model* to access, and outputs (bottom) are named to specify the slots (parameters) of the *chord-seq* instance to set. Onset times and pitches of notes, are derived by sampling discrete values from *gesture-streams* and using these for subsequent temporal processing (quantization) and table lookup, to determine the final values. Durations of notes are specified through a literal value (a value of 100 for *legato* means, every note will last until the onset of the following note). Finally, dynamics of the notes are determined through a random process.

4.4 Rendering of Gesture Descriptions

An alternative scenario for the use of OM-Geste is an environment for gesture visualization, editing/processing, and possibly restitution (see e.g. [13, 35]). We developed functionalities which allow to convert a *gesture-model* into an *SDIF-buffer* object, i.e. a structure of SDIF type definitions and frames according to the GDIF specification. This object can then be exported to disk as an SDIF file, using the standard tools provided in OpenMusic. These files can for instance be interchanged with other environments and possibly used for real-time streaming (see [43] for a similar approach for spatialization data).

5 Case Studies

In this section we describe two case studies for synthesis of direct music representations from real-world gesture recordings. Music-related gestures can be roughly divided into two groups: gestures of those that perceive the sound (*listener* or *dancer gestures*) versus gestures of those that produce the sound (*performer* gestures) [62]. In our first study, we will use gesture recordings of a dance performance for the creation of a symbolic musical score. The second will use gesture data from a DMI performance for rendering of a spatial sound synthesis process. Video recordings of the captured performances (dance and DMI) are available online.[9] For both case studies we used the same workflow, i.e. import of gesture data from SDIF files, processing and segmentation of *gesture-models*, and mapping for synthesis of direct music representations. We designed the mapping programs aiming at "gestural coherence", i.e. a perceivable relationship between the performer's effort and the energetic morphology in the sound [63]. The temporal scale of the produced musical outputs was kept the same as in the original gesture capture, which allows to synchronize the produced direct music representation to the video recording of the performance and validate the gestural coherence between movement and sound.

5.1 Dance Performance

5.1.1 Acquisition

For our first example we used the recording of a dance performance originally choreographed by Isabelle van Grimde as part of the collaborative research-creation project *Les Gestes* carried out by McGill University and the dance company Van Grimde Corps Secrets [64]. This project involved two dancers, two instrumental performers (Violin and Cello), and a family of custom-built physical controllers that could be attached to the body – inspired by the idea of creating a virtual quartet in which dancer gestures influence sound processing of acoustic instruments in real time. One of the controllers is a visor-like device, which transmits data from an embedded 3-axis accelerometer wirelessly to a central computer. Figure 5 shows dancer Sophie Breton wearing the *Visor* device, during the gesture capture performance.

5.1.2 Processing and Segmentation

The recorded data was imported from an SDIF file and smoothed (using b-spline interpolation) before further processing to derive additional descriptors for 3D-orientation and magnitude of the derivative of 3D-acceleration (jerk). The descriptors were gathered together with the original sensor measurements into a *gesture-model*.

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

Figure 5: Video frame showing Sophie Breton wearing the *Visor* during the captured dance performance in Pollack Hall of McGill University. Video by Audréane Beaucage. Used with permission.

Since for the dance performance no referential media or meta information was available, segmentation was determined via analysis of the gesture signals themselves ("internal segmentation", cf. section 3.2): a peak-finding function (laplacian edge detection) was used to detect troughs in the acceleration data and the respective temporal locations were used as a time-structure.[10]

5.1.3 Mapping

The mapping process is shown in Figure 6. For this example we used the same direct music representation (*chord-seq* object) as described in section 4.3. Visible at the top left of the figure is the *gesture-model* containing the segmented gesture signals from the dance performance. The right hand side of the figure shows the mapping program with two inputs for accessing data of the *gesture-streams XAA1* (jerk magnitude, *a*) and *XAA1-ema* (the same descriptor smoothed with an exponential-moving-average filter, *b*). A number of OM functions are used for sampling and scaling descriptor values into useful ranges to specify parameters for the generation of 25 discrete notes per temporal segment (see caption of the figure for more detailed description). Using this mapping program, the function *map-gesture* creates a *chord-seq* object for each segment of the *gesture-model* and outputs a list of instances, which are then merged into a single object (D). This *chord-seq* object is then converted into a *voice* object (a sequence of chords with a metric representation) thereby quantizing the temporal location of chords in linear time into a metric structure (E).[11] The final score is visible in detail

[10]Troughs in acceleration data can be indicative of points of demarcation between gestures [65].

[11]OM's default quantization algorithm groups notes within a time window of 100 ms into chords

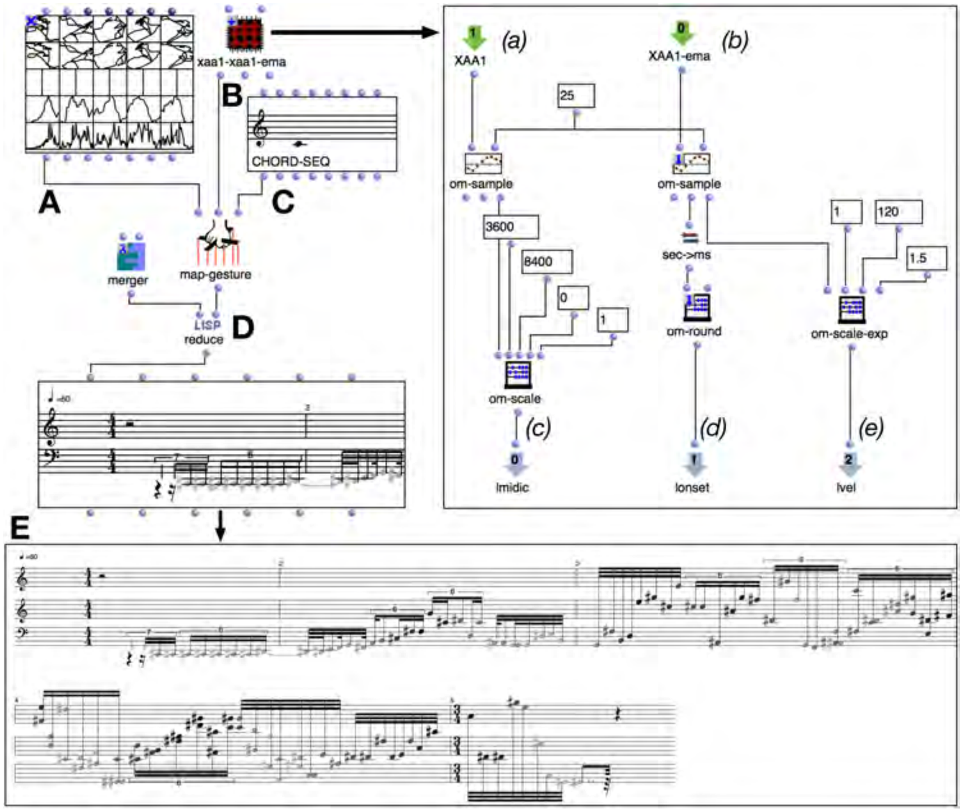

Figure 6: Mapping process for synthesis of a symbolic musical score from dance gestures. A: the *gesture-model* containing gesture data from the dance performance, B: a mapping program as an OM patch, C: the OM class to be produced for each segment (*chord-seq*). Right: Detailed view of the mapping patch, specifying parameters for 25 discrete notes per segment. Jerk magnitude (XAA1, *a*) is downsampled and rescaled into a range from 3600 to 8400 to specify pitch in midicent (*lmidic*, *c*). Lowpass-filtered jerk magnitude (XAA1-ema, *b*) is downsampled and the time tags of the values are used to specify onset times (*lonset*, *d*). The values are exponentially scaled into a range from 1 to 120 to specify dynamics as MIDI velocities (*lvel*, e). D: the resulting *chord-seq* objects are merged together into a single object and converted into a *voice* object with metric rhythmic time notation. E: a detailed view of the final symbolic score (grayscales represent dynamics) is visible at the bottom.

at the bottom of the figure (grayscales represent dynamics). We rendered a MIDI synthesis of the score and overlaid the audio to the original video recording of the dance performance. The resulting video file is available online.[12]

5.2 DMI Performance

5.2.1 Acquisition

Our second case study is based on gesture recordings of a performance with a DMI, titled *SoundSaber*. This instrument uses a Motion Capture (MoCap) system to sense the position of the tip of a 120cm long pvc tube. From this position measurement other motion descriptors are derived and used to drive a real-time sound synthesizer (see [66] for detailed description). For this example we use gesture data of an excerpt of an improvisational performance. Figure 7 shows K. Nymoen with the SoundSaber instrument at University of Oslo's *FourMS* laboratory.

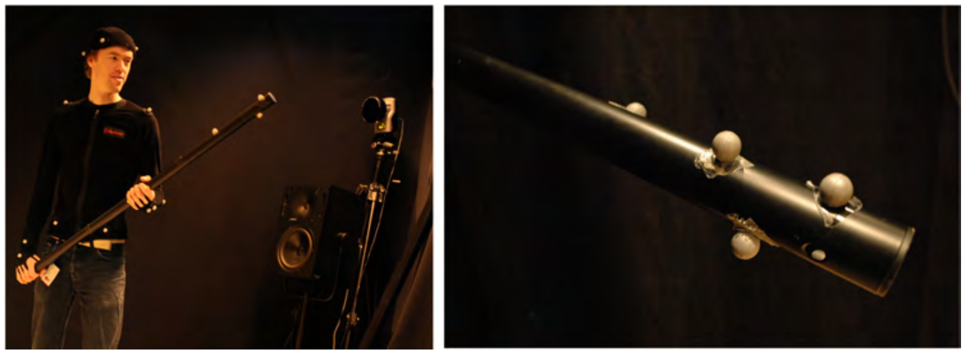

Figure 7: The *SoundSaber* DMI. Left: Kristian Nymoen wearing optical markers during a performance with the instrument. Right: close-up of the controller (tip of the pvc tube) showing marker placement. Used with permission.

5.2.2 Processing and Segmentation

Complementary to the previous case study, where an abstract score consisting of discrete note events was generated, here we will produce a concrete multichannel soundfile containing synthesized spatial audio. Since in this case the gesture data was recorded synchronously with synthesized audio, it can serve as referential data to be used for "external" segmentation of the

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

gesture-streams (cf. section 4.1). Accordingly, both audio and gesture data was imported into OM and the visual representation of the audio waveform was used to manually determine points of minimal amplitude[13] which were used to specify temporal locations for segmenting the gesture signals.

5.2.3 Mapping

For synthesis of spatial sound in OM we used the libraries OMChroma and OMPrisma. OMChroma is a framework for sound synthesis based on Marco Stroppa's *Chroma* system [67]. Classes dedicated to various sound synthesis algorithms can be instantiated to devise sound synthesis processes which are rendered via the *Csound* language [68]. OMPrisma is a library which implements the same control structures and provides an object-oriented system of classes for spatial sound rendering, sharing a consistent control interface [69]. Combining the two libraries, classes for sound synthesis, processing and spatialization can be merged into single hybrid structures and controlled in an integrated process (see [5] for more detailed description).

For illustrative purposes we based our mapping choices on the same considerations of action-sound relationships as in the original instrument design, e.g. establishing a correlation between kinetic energy and sound energy [66].[14] Figure 8 shows the visual program for the mapping and synthesis process. At the top we see the *gesture-model* containing the recorded gesture data of the DMI performance. A: The class-merging process for defining the spatial sound synthesis class is visible on the right hand side of the figure. B: combining an FM synthesizer with a resonant bandpass filter and a spatialization module for reverberated VBAP (rVBAP) [70]. C: On the left hand side of the figure we see the mapping program, applied to each segment of the *gesture-model*. Note, the combination of specifications (literal, symbolic, functional) used in the mapping program. *(a)*: global temporal parameters, such as onset time (*action-time*) and duration (*durs*) of each synthesis event correspond to the original gesture segments, extracted from the descriptor for 3D-position (XP30). *(b)*: the spatialization data for rVBAP is specified as a *many-to-one* mapping: XP30 is first multiplied by a constant (*traj-mult*), before being multiplied with the vector magnitude of acceleration (XVA0). The function *traj-separ* dispatches the

[13]In the original mapping for the recording, audio amplitude correlates with absolute velocity of the SoundSaber's tip.

[14]Note, that this choice of mapping is arbitrary and is intended to validate the faithfulness of our synthesis by overlaying the video recording DMI performance with the audio produced in OM.

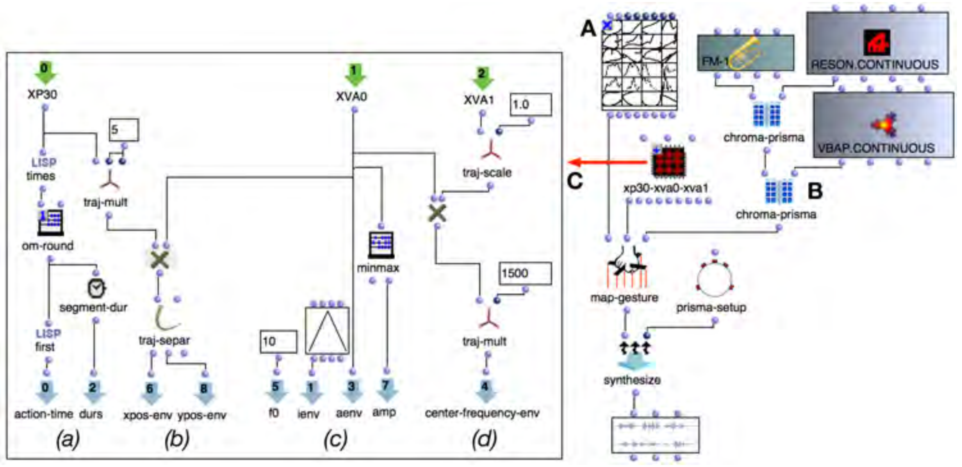

Figure 8: Controlling a spatial sound synthesis process via DMI performance data. On the right: A) *gesture-model* containing recorded MoCap data of a DMI performance. B) definition of a class for spatial sound synthesis, combining FM synthesis, resonant filtering and rVBAP spatialization. C) mapping program for driving the spatial sound synthesis process, shown in detail on the left hand side. (a): specification of global temporal parameters from gesture data, (b): specification of spatialization data combining 3D-position and absolute acceleration, (c): specification of FM synthesis parameters using constants, envelopes, gesture descriptors, (d): control of center-frequency of bandpass-filter by combining absolute acceleration and jerk.

scaled 3-dimensional data into individual envelopes for x and y coordinates, describing position in the horizontal plane. *(c)*: the first two parameters for the FM synthesis are specified as a constant value for carrier frequency (*f0*) and as a triangular envelope for modulation index (*ienv*) – note, that these are global specifications for every event, independently of the gesture data. The shape and the maximum value of the amplitude envelope (*aenv* and *amp*) corresponds to the shape of XVA0 and its maximum value. *(d)*: the center-frequency of the bandpass filter is controlled by normalizing the values describing magnitude of jerk (XVA1) into a range from 0-1 (*traj-scale*), multiplying it with XVA0, before multiplying it times 1500 into a useful range for controlling the center-frequency of the bandpass filter (*center-frequency-env)*. The audio resulting from synthesis process was rendered into a stereo file and added to the original video recording of the performance. The video file is available online.[15]

We chose these two examples as case studies in order to validate the flexibility of our framework by using examples that differ significantly in terms of types of gestures, acquisition context, gesture data, mapping approach, and direct music representation. An overview of these differences is given in table 1.

Parameter	**Dance choreography**	**DMI performance**
Sensing Device	*Visor*	*SoundSaber*
Sensing Method	IMU	MoCap
Measurement reference	egocentric	exocentric
Temporal resolution	16.7 Hz	200 Hz
Numerical resolution	10 Bit	32 Bit
Referential Data	—	audio recording
Mapping scheme	one-to-many	hybrid
Music Representation	symbolic score	spatial sound synthesis

Table 1: Comparison of the two case studies.

6 Discussion and Future Work

In this article we presented a system for integrating physical gestures as musical materials into the symbolic domain of computer-aided composition.

[15] `http://tinyurl.com/DMI-mapping`

We reviewed related works, described requirements for description, representation, and mapping, and presented an implementation of our concepts in the library OM-Geste. To validate the flexibility of our framework we described two case studies for the use of dance and instrumental gestures for synthesis of a symbolic musical score and spatial audio.

A particular focus of was given to the concept of mapping for CAC applications. While in the context of instrumental performance mapping is commonly understood as the association of gesture variables to sound synthesis parameters, the notion of mapping in composition typically has a wider scope, for example as the correlation of data sets, 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" [49]. To address this extended conception we designed a system which is seamlessly integrated into the OM interface using existing tools and interfaces, and in which mappings are conceived as visual programs which allow for hybrid specifications, combining generative-algorithmic approaches, gesture data, and manual setting.

Note, that the presented framework does not attempt to solve compositional problems pertaining to relationships between gestures and music materials. Rather, our aim is to provide open and generic tools that allow composers to develop and experiment with individual artistic concepts, and which may help reintroducing physical expression into computer-aided composition. The first author found that in particular for complex processes requiring large amounts of control data (such as *spatial sound synthesis* [5]), the organic, multi-dimensional nature of gestures combined with the extended mapping possibilities in CAC are promising directions.

Future work will investigate strategies for simultaneous, alternative representations of gestures (similar to multiple "views") as well as flexible data structures which allow for multiple simultaneous segmentations. Another direction we are exploring is the integration of machine learning techniques, both in order to develop tools for segmentation and recognition, as well as implicit mapping strategies. The integration of gesture data is part of a recent trend exposing the programming features and expressivity of CAC environments as components within larger-scale systems, towards combining algorithmic-generative approaches with interactive applications.

Acknowledgements

We would like to express our thanks to 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 like to thank Jean Bresson for support with the design and implementation of the library. This research was partially funded by an NSERC discovery grant to the second author.

References

[1] 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 page 1)*

[2] 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 1 and 8)*

[3] G. Assayag, "Computer Assisted Composition today," in *First Symposium on Music and Computers.* Corfu Greece, Oct. 1998. *(Cited on page 2)*

[4] 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 page 2)*

[5] M. Schumacher and J. Bresson, "Spatial Sound Synthesis in Computer-Aided Composition," *Organised Sound*, vol. 15, no. 03, pp. 271–289, Dec. 2010. *(Cited on pages 2, 21, and 24)*

[6] G. Carpentier and J. Bresson, "Interacting with Symbol, Sound, and Feature Spaces in Orchidée, a Computer-Aided Orchestration Environment," *Computer Music Journal*, pp. 1–18, Feb. 2010. *(Cited on page 2)*

[7] 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 page 2)*

[8] F. J. Varela, E. Thompson, and E. Rosch, *The Embodied Mind: Cognitive Science and Human Experience.* The MIT Press, 1992. *(Cited on page 2)*

[9] R. I. Godøy, “Motor-Mimetic Music Cognition,” *Leonardo*, vol. 36, no. 4, pp. 317–319, Jan. 2003. *(Cited on page 2)*

[10] D. Geeraerts and H. Cuyckens, *The Oxford handbook of cognitive linguistics*. Oxford University Press, USA, 2007. *(Cited on page 2)*

[11] G. Lakoff and M. Johnson, *Philosophy in the Flesh*. New York: Basic Books, 1999. *(Cited on page 2)*

[12] D. F. Armstrong, W. C. Stokoe, and S. E. Wilcox, *Gesture and the Nature of Language*. Cambridge University Press, 1995. *(Cited on page 2)*

[13] C. Cadoz, “Instrumental Gesture and Musical Composition,” in *International Computer Music Conference*. San Francisco, USA: Proceedings of the 1998 International Computer Music . . . , 1988, pp. 1–12. *(Cited on pages 2, 5, and 16)*

[14] C. Cadoz and M. M. Wanderley, “Gesture-Music,” in *Trends in Gestural Control of Music*. Ircam—Centre Pompidou, 2000, pp. 71–93. *(Cited on page 2)*

[15] A. Jensenius, “Action-sound: Developing Methods and Tools to Study Music-related Body Movement,” Ph.D. dissertation, Department of Musicology, University of Oslo, July 2007. *(Cited on pages 2 and 7)*

[16] M. M. Wanderley, “Interaction Musicien-Instrument: Application au contrôle gestuel de la synthèse sonore,” Ph.D. dissertation, 2001. *(Cited on page 2)*

[17] F. Delalande, “Le Geste, outil d’analyse: quelques enseignements d’une recherche sur la gestique de Glenn Gould,” 1988. *(Cited on page 2)*

[18] M. M. Wanderley and P. Depalle, “Gestural Control of Sound Synthesis,” in *Proceedings of the IEEE*, 2004, pp. 632–644. *(Cited on pages 2 and 9)*

[19] J. Bresson and C. Agon, “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 3 and 7)*

[20] C. Agon, G. Assayag, and J. Bresson, Eds., *The OM Composer’s Book: Volume 1*, ser. Collection Musique/Sciences. Éditions Delatour France, 2006. *(Cited on page 3)*

[21] J. Bresson, C. Agon, and G. Assayag, Eds., *The OM Composer's Book: Volume 2*, ser. Collection Musique/Sciences. Éditions Delatour France, 2008. *(Cited on page 3)*

[22] 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 3 and 5)*

[23] 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. *(Cited on page 3)*

[24] 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*, 2015. *(Cited on pages 3 and 5)*

[25] 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, June 2005. *(Cited on page 3)*

[26] G. Marino, M. Serra, and J. Raczinski, "The UPIC system: Origins and innovations," *Perspectives of New Music*, vol. 31, no. 1, pp. 258–269, 1993. *(Cited on page 4)*

[27] E. Wenger. (1997) Metasynth. [Online]. Available: http://www.ircam.fr/produits-real/logiciels/metasynth-e.html *(Cited on page 4)*

[28] HighC. [Online]. Available: http://highc.org/ *(Cited on page 4)*

[29] T. Coduys and G. Ferry, "IanniX. Aesthetical/symbolic visualisations for hypermedia composition," in *Sound and Music Computing Conference*, Lefkada, Greece, 2004. *(Cited on page 4)*

[30] M. Wright, A. Freed, and A. Momeni, "OpenSound Control: state of the art 2003," *Proceedings of the 2003 conference on New interfaces for musical expression*, pp. 153–160, 2003. *(Cited on page 4)*

[31] T. Coughlan and P. Johnson, "Interaction in Creative Tasks: Ideation, Representation and Evaluation in Composition," in *SIGCHI Confer-*

ence on Human Factors in Computing Systems. Montreal, Canada: CHI 2006, 2006. *(Cited on page 5)*

[32] W. Mackay, "Designing Interactive Paper: Lessons from three Augmented Reality Projects," in *Proceedings of IWAR*, 1999. *(Cited on page 5)*

[33] 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. *(Cited on page 5)*

[34] J. Garcia, J. Bresson, and T. Carpentier, "Towards interactive authoring tools for composing spatialization." *3DUI*, pp. 151–152, 2015. *(Cited on page 5)*

[35] C. Ramstein, "Analyse, Représentation et Traitement du Geste Instrumental," Ph.D. dissertation, Institut National Polytechnique de Grenoble, 1991. *(Cited on pages 5 and 16)*

[36] 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 5, 9, and 11)*

[37] F. Lacquaniti, C. Terzuolo, and P. Viviani, "The law relating the kinematic and figural aspects of drawing movements," *Acta psychologica*, 1983. *(Cited on page 6)*

[38] 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," *Conference on Computer Graphics Theory and Applications*, 2006. *(Cited on page 6)*

[39] D. McGilvray, "On The Analysis of Musical Performance by Computer," Ph.D. dissertation, University of Glasgow, 2007. *(Cited on page 6)*

[40] 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 6)*

[41] M. Wright, A. Chaudhary, A. Freed, D. Wessel, X. Rodet, D. Virolle, R. Woehrmann, and X. Serra, "New applications of the sound description interchange format," in *International Computer Music Conference*, Ann Arbor, USA, 1998. *(Cited on page 7)*

[42] 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 7)*

[43] J. Bresson and M. Schumacher, "Representation and interchange of sound spatialization data for compositional applications," in *International Computer Music Conference*, Huddersfield, UK, 2011. *(Cited on pages 7 and 16)*

[44] 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 page 7)*

[45] J. Bresson, "Sound Processing in OpenMusic," in *9th International Conference on Digital Audio Effects*, Montreal, Canada, 2006, pp. 325–330. *(Cited on page 7)*

[46] M. Schumacher, "Ab-Tasten: Atomic Sound Modeling with a Computer-Controlled Acoustic Grand Piano," in *The OM Composer's Book: Volume 3*, J. Bresson, G. Assayag, and C. Agon, Eds. Éditions Delatour, IRCAM/Centre Pompidou, 2016. *(Cited on page 7)*

[47] 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: ACM, 2000, pp. 446–453. *(Cited on page 8)*

[48] P. Doornbusch, "Composers' views on mapping in algorithmic composition," *Organised Sound*, vol. 7, no. 02, p. 12, Aug. 2002. *(Cited on page 8)*

[49] ——, "Mapping in Algorithmic Composition and Related Practices," Ph.D. dissertation, RMIT University, Victoria, 2010. *(Cited on pages 8 and 23)*

[50] 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 page 8)*

[51] J. Chadabe, "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 page 8)*

[52] M. N. Downie, "Choreographing the extended agent: performance graphics for dance theater," Ph.D. dissertation, Massachusetts Institute of Technology, 2005. *(Cited on page 8)*

[53] 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, 2014. *(Cited on page 8)*

[54] M. M. Wanderley, N. Schnell, and J. Rovan, "Escher - Modeling and Performing "Composed Instruments" in Real-Time," in *IEEE International Con- ference on Systems, Man and Cybernetics (SMC'98)*. San Diego, USA: IEEE, 1998, pp. 1080–1084. *(Cited on page 9)*

[55] 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. *(Cited on page 9)*

[56] 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 9)*

[57] D. Wessel and M. Wright, "Problems and Prospects for Intimate Musical Control of Computers," *Computer Music Journal*, pp. 11–22, Oct. 2002. *(Cited on page 9)*

[58] J. Malloch, S. Sinclair, and M. Schumacher. Digital Orchestra Toolbox for MaxMSP. [Online]. Available: http://www.idmil.org/software/digital_orchestra_toolbox *(Cited on page 14)*

[59] J. Bresson, "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 page 14)*

[60] 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, 1991. *(Cited on page 14)*

[61] 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 page 15)*

[62] 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 16)*

[63] M. Goldstein, S. Studio, and M. Park, "Gestural coherence and musical interaction design," *Systems*, 1998. *(Cited on page 17)*

[64] 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 17)*

[65] 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*. Citeseer, 2002. *(Cited on page 18)*

[66] 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. *(Cited on pages 20 and 22)*

[67] 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 page 21)*

[68] J. Fitch, V. Lazzarini, and S. Yi, "Csound6: old code renewed," in *Linux Audio Conference*, Graz, Austria, 2013, p. 69. *(Cited on page 21)*

[69] M. Schumacher and J. Bresson, "Compositional Control of Periphonic Sound Spatialization," in *2nd International Symposium on Ambisonics and Spherical Acoustics*. Paris, France: Citeseer, 2010. *(Cited on page 21)*

[70] O. Mathes. RVBAP = Reverberated VBAP. [Online]. Available: http://impala.utopia.free.fr/pd/patchs/externals_libs/vbap/rvbap.pdf *(Cited on page 22)*