On the Development of a System for Gesture Control of Spatialization

M. T. Marshall♮, N. Peters♮, A. R. Jensenius♮♯, J. Boissinot♮, M. M. Wanderley♮ and J. Braasch♭
♮ IDMIL[1] and CIRMMT[2], Schulich School of Music, McGill University
♯Musical Gestures Group, Department of Musicology, University of Oslo
♭School of Architecture, Rensselaer Polytechnic Institute (RPI)
mark.marshall@mail.mcgill.ca, nils.peters@mcgill.ca, a.r.jensenius@imv.uio.no

Abstract

This paper presents our current approach to the development of a system for controlling spatialization in a performance setup for small ensemble. We are developing a Gesture Description Interchange Format (GDIF) to standardize the way gesture-related information is stored and shared in a networked computer setup. Examples are given of our current GDIF namespace, the gesture tracking subsystem developed to use this namespace and patches written to control spatialization and mapping using gesture data.

1 Introduction

The spatialization of sound has played an important roll in electroacoustic music performance from the very beginning. Electronic systems for control of spatialization in sound have been demonstrated as early as 1951, with the creation of the *pupitre d'espace* (Chadabe 1997). However, to date there has been little research into aspects of gesture controlled spatialization of sound. Recently a number of projects have begun to deal with this and have developed systems which allow control of spatialization using gestures.

Systems such as that described in (Naef and Collicott 2006) allow a performer to control the spatialization of a number of sound sources through the use of a virtual reality (VR) display and a custom built dataglove. Others, such as the *ZKM Klangdom* (Ramakrishnan, Grossmann, and Brümmer 2006), allow for control of spatialization from a number of different software interfaces. There has also been some work on the development of systems which allow for the use of a number of different user interface devices, such as presented in (Wozniewski, Settel, and Cooperstock 2006), which can be controlled by devices such as head trackers, datagloves or camera-based tracking systems.

The major difference between many of these systems and the one discussed in this paper, is that they make use of a seperate interface and a specific performer to control the spatialized sound. Our system is being developed to allow real-time gesture control of spatialization in a live performance setup, by the performers themselves. This gives performers control over aspects of the spatialization of the sound of their own instrument during the performance of the piece.

This paper presents our current approach to this task and describes the spatialization software we are using and the gesture control interface which we have developed to control it. The following sections of this paper describe the ViMiC spatialization system which forms the core of our implementation, the gesture control system and our attempts at developing a standardized means of communicating gesture information to the spatialization system.

2 Spatialization System

2.1 The ViMiC approach

We are using the ViMiC *(Virtual Microphone Control)* spatialization system which was presented in (Braasch 2005), and implemented in *Pure Data (PD)*.

Rather than conventional spatialization techniques such as *Wavefield Synthesis* (WFS) (de Vries, Start, and Valstar 1994), *Higher Order Ambisonics* (HOA) (Nicol and Emerit 1999) or panning law based techniques such as *Vector Based Amplitude Panning* (VBAP) (Pulkki 1997), the ViMiC-System, which has recently undergone improvement as part of this project, combines a *tonmeister's* know-how of various sound recording techniques with the knowledge of room acoustics, sound propagation and spatial perception.

ViMiC is a computer-generated virtual environment,

[1]Input Devices and Music Interaction Lab
[2]Centre for Interdisciplinary Research in Music Media and Technology

where gains and delays between a virtual sound source and virtual microphones are calculated according to their distances and the axis orientations of the microphone directivity patterns. Besides the direct sound wave, a virtual microphone signal also contains early reflections and an adequate reverberation tail. This depends both on the sound absorbing and reflecting properties of the virtual surfaces and on the geometry of the virtual enclosed space. Sound sources and microphones can be spatially placed and moved in 3D as desired.

The system architecture was designed to comply with the expectations of audio engineers, and to create sound imagery similar to those associated with standard sound recording practice.

There are a number of adjustable parameters, all of which are normalized between zero and one, which can affect this auditory virtual environment, including:

- Position of the sound sources [X,Y,Z]
- Radiation patterns of the sound sources
- Ratio of direct and reverberant sound
- Absorption properties of the walls
- Room size [X,Y,Z]
- Position of the microphones [X,Y,Z]
- Directivity patterns of the microphones (continuously adjustable from omni via cardio to figure of 8)

In addition to these low-level parameters, we have implemented a set of "higher level" parameters which are translated into low level operations. These make it possible to control the ViMiC system in a more intuitive and musical way, and include:

- Movements of the sound sources
  - Rotation [centrepoint, radius, speed, direction]
  - Orientation of radiation patters
  - Stereo spread of a 2 channel sound source
  - Ballistic curves [mass, gravitation, initial energy, ignition angle
  - Boomerang curves
  - Pendular movements [mass, gravitation, pendular length, starting & centre point]
  - Artificial life algorithms
- Movements of grouped microphones
  - Rotation
  - Radius of the grouped microphone array

Figure 1 shows the ViMiC Visualization software in a setup of eight microphones and two sound sources.

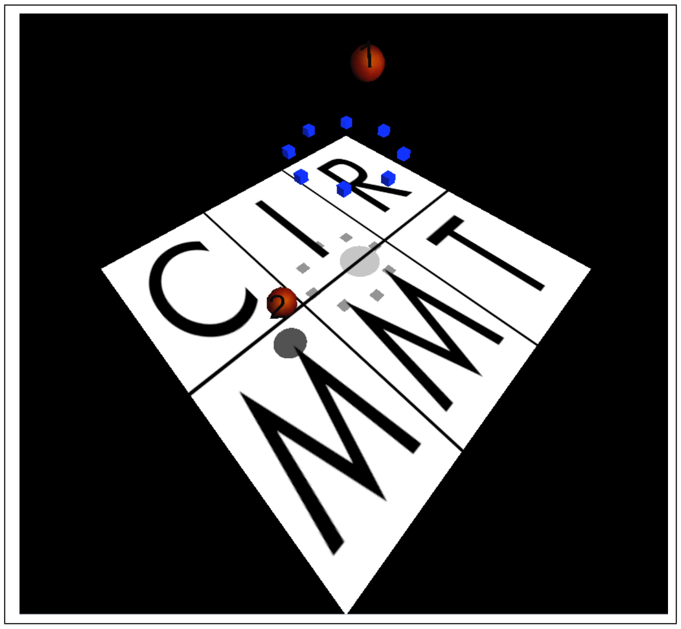

Figure 1: ViMiC Visualization: two sound sources *(red spheres)* and a virtual microphone array *(smaller, blue squares)*

2.2 ViMiC-remote

To ensure system stability it is advisable to seperate the user interface from the audio rendering algorithms in low latency applications. For this reason a convenient GUI was designed which allows control of all the parameters through a VST plugin. Together with a digital audio workstation (e.g., Steinberg Nuendo) it allows easy synchronization and storage of spatialization data along with the unprocessed, anechoic audio material (e.g. allowing storage of the trajectories of sound sources). Communication between the VST plugin and the ViMiC System is done using Open Sound Control (OSC) messages over UDP. This facilitates better temporal accuracy when controlling many parameters simultaneously and it makes it possible to communicate with other OSC-compatible interfaces.

3 Description of the Gesture Tracking System

In our current setup we are using a Polhemus Liberty electromagnetic tracker with a speed of up to 240 Hz per sensor for 8 sensors and an accuracy of 0.03 mm for X, Y, Z position and 0.15° for orientation. In this current system the user has to wear wired sensors on the body points that are to be tracked. Figure 2 shows the Polhemus Liberty system, including base station, transmitter and receivers.

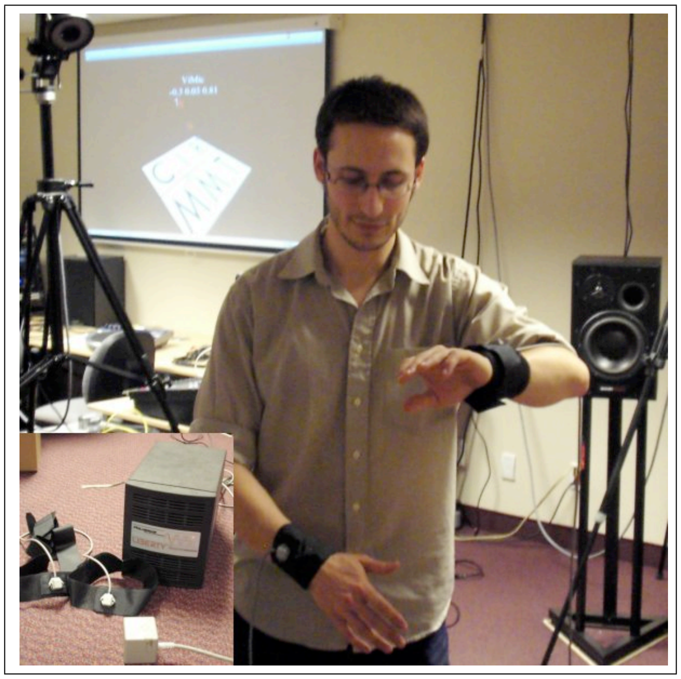

Figure 2: A user in the process of using the spatialization system, the visualization system (background) and the tracker hardware (inset)

3.1 System architecture

Figure 3 shows the overall system architecture for our implementation. The host computer communicates with the Liberty over a USB connection. Raw data is received from the Liberty at a rate of 240 updates per second for each sensor. This data gives the position and orientation of each of the sensors with 6 degrees-of-freedom. This raw data is then used in the calculation of further levels of data, such as the body-related data and the metadata described in section 4. Data is also read by the host system from other sensors, such as pressure tiles and accelerometers, which may also be used to track the users' motion.

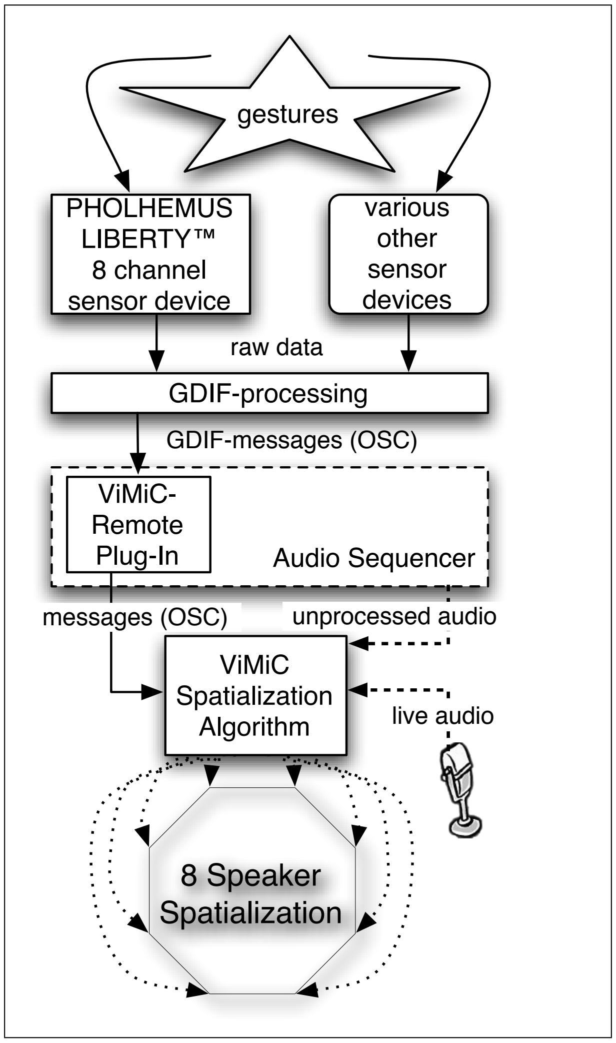

Figure 3: Sketch of the data flow inside the system

Once all calculations have been made, the data is formatted into a number of different OSC messages and broadcast using IP multicast, which is a method whereby a message can be sent simultaneously to several computers, instead of singly to one computer.

Each level of data is sent to a different multicast group. Client machines may subscribe to one or more of these groups, allowing them to receive only the data which they require. Each machine then uses this data to perform its specific task. In our particular implementation the messages are currently all being used by the spatialization and visualization system, but the capability exists for other systems to access the data.

3.2 Host implementation

The program to communicate with the Liberty, format and send the messages was originally implemented in *Python*, but is currently being ported to *C++*. Along with receiving the data and formatting the messages for transmission, this program performs a number of other useful functions, including:

Scaling To allow for easier use of the data and to dispense with any issues which might arise from the units of measurement in use, all data is scaled to a [0 1] range.

Clipping When a Liberty sensor moves outside of the range of the transmitter, the position data received from the system inverts. To avoid this, once a sensor is detected as having left the range of the transmitter, its previous valid position is resent until the sensor re-enters the transmitter's range.

Smoothing The Liberty system updates all the position and orientation data of each sensor 240 times per second. Coupled with the system's high resolution (down to 0.038 mm) this can lead to a high rate of change for the data even when the sensors are perceptually static. In order to reduce some of the load on the client systems, some smoothing is applied to the data so that only significant changes in the data are recorded.

After the neccessary calculations and other functions have been performed, the data is formatted into a number of OSC messages and sent to a specific multicast group and port combination so that they may be read by the client systems.

3.3 Client implementation

Client systems have been implemented in *PureData (PD)* and *Max/MSP*. The latter natively supports subscription to an IP multicast group, but this was not the case in PD. We therefore extended the *dumpOSC* object to take a multicast group and port number as its parameters, so that it is possible to subscribe to the specified group and allow the client system to receive the messages from the Liberty system. The client system discussed in section 2 makes use of this object to receive information from the host system.

Once the client system has subscribed to the necessary multicast groups, it will begin to receive the data from the host system. This data can then be used to control aspects of the client system. As all the data sent by the host system is normalized, it can easily be scaled and mapped to control parameters of the client system.

3.4 Host control system

A host control system is currently being implemented to allow users to specify which data they would like their client system to receive from the host. This system is being implemented in *C++*, with a HTML-based front end. Apart from the three main streams of GDIF-data, this will allow users to create additional streams of data, made up of a combination of the various data gathered from the Liberty or calculated by the system. They may then specify which multicast group and port this data will be sent to. This allows applications to receive only the exact data that they require from the host system in order to operate correctly.

The control system communicates with the host system over the network. This allows control of the host system from the client machine or other machines. Requests to the host consist of a list of parameters, a multicast group address and a port number. On receiving this data the host system will create a new socket and subscribe it to the specified multicast group address and port. Once this is completed it will begin sending the requested data to this group, where it can be accessed by the client systems.

Currently the aim of this system is to allow client systems to select to receive a subset of the processed data, which would help reduce bandwidth requirements. However, in future versions it might be possible to allow the client systems to define and request new forms of processed data, based on manipulation of some of the existing data which is extracted by the host system.

4 Towards a GDIF Namespace

As presented in (Jensenius, Kvifte, and Godøy 2006), we are developing a Gesture Description Interchange Format (GDIF) for standardizing gesture related information. In our research we typically use a number of different devices, tools and methods when studying gestures, everything from motion capture with infrared and electromagnetic tracking systems, to commercial MIDI controllers and new interfaces for musical expression. At the moment there are no standards for storing and sharing such information, and this makes it difficult when working on collaborative projects both within research groups and between institutions.

The Sound Description Interchange Format (SDIF) has emerged as a standard for storing the results of audio analysis (Wright, Chaudhary, Freed, Wessel, Rodet, Virolle, Woehrmann, and Serra 1998; Schwarz and Wright 2000), and we see the need for a similar standardized way of storing movement-related information. Working with a uniform set of descriptors could greatly facilitate the process of developing performance systems, since different types of sensors and motion capture systems could be swapped without having to change the gesture-sound mappings. This, of course, requires a set of movement descriptors that are intuitive, general and consistent.

Attempts have been made to move towards uniform namespaces using Open Sound Control (OSC) for gesture information (such as (Wright, Freed, Lee, Madden, and Momeni 2001) and recent discussions in the OSC community[1]), but there does not seem to be any consensus on how to actually describe such information. One of the reasons for this might be the focus on technical parameters (e.g. raw input from sensor systems) rather than motion *qualities*.

In the spatialization project, we separate the movement related data into three categories:

Raw data Unprocessed data coming from sensing devices (Polhemus Liberty electromagnetic tracker, accelerometers and floor pressure sensors)

Body data Information about orientation and motion of the body, and limb motion in relation to the body

Meta data Information about general motion qualities, such as perceptual intensity

The following sections discuss how these data can be implemented as a GDIF OSC namespace.

4.1 Raw data

Even though we never use raw data from the various sensor devices directly without doing some kind of preprocessing, they are kept as is for reference and later playback. The raw data coming from the Polhemus is formatted as a string containing information about the absolute position (x, y, z) and azimuth, elevation and roll (az, el, rl) of the sensor in relation to the transmitter. The general format is:

```
/device/sensor <x y z az el rl>
```

which would give messages like:

```
/liberty1/s1 0.4 0.1 0.8 -1.2 0.2 0.7
```

Notice that each device is given a unique name to allow for using multiple devices at the same time. Using lists of data rather than separate messages (e.g. /liberty1/s1/x 0.4) reduces readability of the stream, but saves bandwidth.

Similarly, data from other types of sensors are formatted as:

```
/interface/sensor/type <value>
```

which would give the following message for a sensor tile (foot pressure) connected to our AVR-HID[2] sensor interface:

```
/avr_hid1/tile1/pressure 0.9
```

The namespace can be expanded to include any other type of device, sensor interface and sensor. The idea is that all the raw data from all the devices used in a setup should be available everywhere on the network at any time. Since the raw data messages are communicated on a separate port in the multicast setup, the messages will not have to be parsed by the computers that are only interested in for example the body-centred data.

4.2 Body-centred data

Various devices and sensor interfaces all have different ranges, resolution and reference points, and this makes it necessary to do a lot of pre-processing on the data before they can be useful for mapping purposes. To allow for a more intuitive mapping process, we are interested in formatting the data with a focus on the body, since this is usually the point of departure when talking about gestures with performers. We are here exploring a combination of biomechanical properties, and concept's developed from Laban's *effort* theory (Laban and Lawrence 1947) and Bartenieff fundamentals (Hackney 2000), which have been successfully applied to musical gesture analysis (Campbell, Chagnon, and Wanderley 2005).

The first part of a body-centred system, is to define general body parameters like orientation in relation to the speaker setup, and the person's weight transfer and overall quantity of motion. These can be calculated from a Polhemus sensor positioned in the lower back (*lumbar*) of the person, and can be represented as:

```
/body1/position <angle>
/body1/weight/horizontal <value>
/body1/weight/sagittal <value>
/body1/motion <value>
```

All other values in the system will be calculated in relation to the reference point and the values representing the main body. Thus performed gestures will be independent of the body's orientation and position in the physical space, which resembles how we generally think about gestures (except pointing gestures).

In the current setup we are only using two other sensors, on the outside of each hand. Since we often think about the left and right halves of our body as two similar and mirrored sides, we have chosen to use two mirrored coordinate systems for the two sides (see Figure 4). A positive *movement space* is defined for each hand in the forward, upward, outward quadrant, where most of the movement typically takes place. So both arms will give positive values when moving outwards, to the front and upwards and negative values when moving inwards, to the back and downwards. The namespace is formatted like:

```
/body/part/side <x y z az el rl>
```

which will give messages like:

```
/body1/arm/left 0.4 0.5 0.4 0.5 0.5 0.5
```

[1]http://www.opensoundcontrol.org
[2]see http://www.music.mcgill.ca/ marshall/projects/avr-hid/

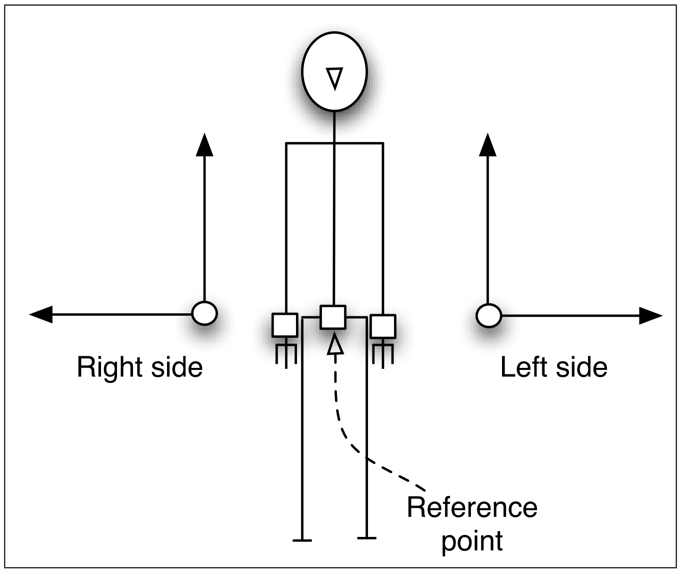

Figure 4: Sketch of our human-focused namespace, where the body is split in two, and two positive quadrants are defined on either side.

Each person is given a unique identifier (e.g. body1 or Paula), so that it is possible to use multiple performers in a setup.

The system will be expanded with more sensors (e.g. on head and feet) and other types of motion descriptors will be added, like quantity of motion for each hand and distance between hands, as well as other body parts, e.g. head. These can be used to find the *kinesphere* (i.e. maximum possible expansion) of the person by storing information about maximum positions of the hands. This can again be used to calculate a running measure for *expansion/contraction index* based on the relation between the current position of the sensors in relation to the kinesphere.

4.3 Metadata

Both the raw data and body-centred data described above are focusing on specific sensors or body parts of a performer. However, musicians, composers and dancers often describe motion in broad terms like *intensity*, or they use metaphors or refer to emotional qualities (Camurri, De Poli, Leman, and Volpe 2005). We are interested in also describing some of these types of data. Currently we have only implemented intensity, by looking at the overall quantity of motion of hands in relation to the reference point:

```
/meta/intensity <value>
```

Other meta level descriptors will be added during the testing and evaluation sessions with performers.

5 Conclusion

We have described our current approach for gesture control of sound spatialization. The idea is to create a flexible setup where various sensing devices can communicate with different sound processing modules running in a large networked computer setup.

The general GDIF OSC namespace we are currently developing makes it easier to map (and remap) gesture data to parameters in the ViMiC Spatialization system. From the technical point of view there are no limits regarding how we can link gesture data to spatialization parameters. The question is what gesture makes sense to control what parameter. As this system is being designed to allow instrumental performers to control some aspects of spatialization of their own acoustic sound during performance we must determine what aspects of the spatialization they should control and what gestures they should use to control them.

As the act of performing with acoustic instuments makes use of a large number of gestures, the possible gestural vocabulary available to control the spatialization system is limited. These gestures must be chosen to allow the performer to control the spatialization system without interfering with the performance of their instrument. Careful analysis of the performers gestures will be required to allow us to determine the optimum mappings for the system. Possible control gestures might include measurement of the performers weight balance (or centre of gravity) to allow them to control parameters through slight swaying of the body. Other possiblities might include the movement of the feet, or for certain instruments the movement of the performer's head.

5.1 Future work

Future work includes:

- Improvements to the host and client implementations and the host control system
- Further development and testing of the current GDIF namespace and inclusion of other sensor devices
- Further development of ViMiC higher level control features
- Examining the available gestures for a number of different instrumental performers to determine suitable control gestures
- Testing and evaluation of the setup in musical performance

Acknowledgments

This work has been partly funded by grants from the Canadian Natural Science and Engineering Research Council (NSERC), the Norwegian Research Council and the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT).

References

Braasch, J. (2005). A loudspeaker-based 3D sound projection using virtual microphone control (ViMiC). In *Convention of the AudioEng. Soc. 118, Barcelona, Spain, Preprint 6430*.

Campbell, L., M.-J. Chagnon, and M. M. Wanderley (2005). On the use of Laban-Bartenieff techniques to describe ancillary gestures of clarinetists. Research report, Input Devices and Music Interaction Laboratory, McGill University.

Camurri, A., G. De Poli, M. Leman, and G. Volpe (2005). Toward communicating expressiveness and affect in multimodal interactive systems for performing arts and cultural applications. In *IEEE Multimedia*, Volume 12.

Chadabe, J. (1997). *Electric Sound: The Past and Promise of Electronic Music*. Prentice Hall.

de Vries, D., E. Start, and V. Valstar (1994). The wave field synthesis concept applied to sound reinforcement: Restrictions and solutions. In *96th AES Convention*, Amsterdam, Netherlands.

Hackney, P. (2000). *Making Connections: Total Body Integration Through Bartenieff Fundamentals*. New York: Routledge.

Jensenius, A. R., T. Kvifte, and R. I. Godøy (2006). Towards a gesture description interchange format. In *Proceedings of New Interfaces for Musical Expression, NIME 06, IRCAM - Centre Pompidou, Paris, France, June 4-8*, pp. 176–179. Paris: IRCAM - Centre Pompidou.

Laban, R. and F. Lawrence (1947). *Effort*. London: Macdonald & Evans Ltd.

Naef, M. and D. Collicott (2006). A vr interface for collaborative 3d audio performance. In *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06), Paris, France*, pp. 57–60.

Nicol, R. and M. Emerit (1999). 3D-sound reproduction over an extensive listening area: a hybrid method derived from holophony and ambisonic. In *AES 16th International Conference: Spatial Sound Reproduction*, pp. 436 – 453.

Pulkki, V. (1997). Virtual sound source positioning using vector base amplitude panning. *Journal of the Audio Engineering Society*, 456–466.

Ramakrishnan, C., J. Grossmann, and L. Brümmer (2006). The ZKM klangdom. In *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06), Paris, France*, pp. 140–143.

Schwarz, D. and M. Wright (2000). Extensions and applications of the sdif sound description interchange format. In *Proceedings of the International Computer Music Conference, Berlin, Germany*, pp. 481–484.

Wozniewski, M., Z. Settel, and J. R. Cooperstock (2006). A framework for immersive spatial audio performance. In *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06), Paris, France*, pp. 144–149.

Wright, M., A. Chaudhary, A. Freed, D. Wessel, X. Rodet, D. Virolle, R. Woehrmann, and X. Serra (1998). New applications of the sound description interchange format. In *Proceedings of the International Computer Music Conference, Ann Arbor, Michigan*, pp. 276–279.

Wright, M., A. Freed, A. Lee, T. Madden, and A. Momeni (2001). Managing complexity with explicit mapping of gestures to sound control with osc. In *Proceedings of the 2001 International Computer Music Conference, Habana, Cuba*, pp. 314–317.