OSC Virtual Controller

Victor Zappi
Istituto Italiano di Tecnologia
via Morego 30
Genoa, Italy
victor.zappi@iit.it

Andrea Brogni
Istituto Italiano di Tecnologia
via Morego 30
Genoa, Italy
andrea.brogni@iit.it

Darwin Caldwell
Istituto Italiano di Tecnologia
via Morego 30
Genoa, Italy
darwin.caldwell@iit.it

ABSTRACT

The number of artists who express themselves through music in an unconventional way is constantly growing. This trend strongly depends on the high diffusion of laptops, which proved to be powerful and flexible musical devices. However laptops still lack in flexible interface, specifically designed for music creation in live and studio performances. To resolve this issue many controllers have been developed, taking into account not only the performer's needs and habits during music creation, but also the audience desire to visually understand how performer's gestures are linked to the way music is made. According to the common need of adaptable visual interface to manipulate music, in this paper we present a custom tridimensional controller, based on Open Sound Control protocol and completely designed to work inside Virtual Reality: simple geometrical shapes can be created to directly control loop triggering and parameter modification, just using free hand interaction.

Keywords

Glove device, Music controller, Virtual Reality, OSC, control mapping

1. INTRODUCTION

In the last decade digital music has become one of the most important means of artistic expression, sometimes in an unconventional way. Many contemporary artists do not make use of keyboards or other common instruments, what they strictly need to create music, in studio or at home, is just a laptop and, in some cases, they even perform live using no other devices. However surfing on the web it is very common to come across digital music communities, where people ask suggestions and share projects and expertise about musical devices: web sites like CDM[1] and MONOME[2] host different kind of people (e.g. musicians, programmers, hardware skilled people) that work together to better exploit and even build musical instruments and innovative controllers. Indeed most of the popularity of these web sites comes from the newly created devices, showed in action during stunning live performances or studio recordings.

What is the reason for all this effort in creating digital music and devices? As discussed by Jordà [6] the laptop+mouse+MAX trinomial is characterized by a very high Macro diversity and represents one of the more generic and versatile instruments; furthermore the intrinsic customizable nature of this instrument permits to have fine control of any structural variation, determining high Middle and Micro diversity too. These are wonderful features for a musical instrument, and this partly explains the world interest in laptop music; but in a previous work by Jordà [5] big relevance was given to the instrument efficiency and to the control input complexity: in music software precision and range are really cutting-edge features, but the set of gestures that permit us to control them is very scarce, and based on general purpose interfaces, not designed for music creation (e.g. mouse and laptop keyboard), and also unable to show to the audience the mechanisms by which the music was created [15]. Consequently, an ever growing number of musicians feel the necessity to create digital music in a very direct and personal way, controlling hardware and software resources with custom highly efficient interfaces; this trend is witnessed but the rich variety of custom controllers that is possible to connect to our laptops [4][11][14][1]. Some of these brand new controllers proved to be particularly well designed and ground-breaking, proposing some fresh methodologies to interpret digital music creation; today their usage is not limited to a small number of "digital fanatics", but it is wide spread into the whole underground music scene, and beyond: recently worldwide artist like Nine Inch Nails and Daft Punk performed using Monome and Lemur[3] controllers, two of the more astonishing devices coming from the new wave of digital music movement.

According to the common need of unconventional ways to express ourselves through music, in this paper we present a custom tridimensional controller, based on Open Sound Control (OSC) protocol and completely designed to work inside Virtual Reality (VR): the surrounding virtual ambient can be modified through free hand interaction, affecting music parameters according to the chosen mappings; objects can be moved and triggered according to the posture of the hand. With this project we want to compound the concept of 3D music controller with the concept of 3D visual generator, proposing a new kind of performances where audio and visuals are perceived as a unique and solid piece of art, both for musicians and for the audience.

[1]http://createdigitalmusic.com/
[2]http://monome.org/
[3]http://www.jazzmutant.com/

2. RELATED WORKS

The VR controller we developed permits to create some simple geometric shapes and to define arbitrary motion and trigger metaphors. In literature a metaphor is an association among human gesture, graphic appearance and sound variation. We could consider mappings (the links between the various streams of controller output and the input parameters of synthesis engines [7]), as the part of each metaphor that is directly related with sound. Metaphors are a key concept when dealing with instruments and controllers, as many researches can demonstrate.

In Maki-Patola et al. [8] four gesture controlled virtual instruments were developed starting from four physical music instruments. Authors decided for four fixed metaphors that granted a realistic play, using two wired data gloves. The heaviest difference from real playing was the lack of tactile feedback. More unconventional ways of interaction were presented by Rodet et al. [12], Campbell et al. [2] and Neaf and al. [9], where, according to the context, it was possible to define various metaphors, which lead to different implementations, more or less close to reality. Particularly, in Neaf et al. [9] the authors introduce the possibility to change volume and position of sound sources just pinching virtual objects.

An interesting project is presented by Voto et al. [13], where users can explore and manipulate visual elements, generating different musical outputs, constructing and deconstructing some reproduction of Kandinsky's paintings; exploiting the intrinsic synaesthetic art of the painter, colors and shapes are associated to specific instruments, chords, and rhythms. This work shares many features with our VR interface, but what a controller has to assure is the possibility to completely map control inputs onto sound parameter variations, according to the current performer's needs. In Petersen at al. [10] a hands-free gesture-based control interface is presented, describing in particular a set of virtual controllers that users can create, compound and map to control the music output. These controllers are 2D representations of common interfaces (i.e. trigger pads and faders).

3. EQUIPMENT

We designed our controller on a $4x2m^2$ powerwall, projected by two Christie Mirage S+ 4000 projectors, synchronized with StereoGraphics CrystalEyes active shutter glasses. We use a Sun workstation (dual core AMD Opteron, processors 2218, 2.60 GHz, 2.50 GB RAM) with Windows XP Professional, Nvidia Quadro FX 4600 video card and M-Audio Profire 2626 audio interface.

We use VRMedia XVR[4] as main software, to handle graphics, scene behavior and input/output data sending. Devices and software exchange data with the main application through XVR internal modules written in C++ and Python.

XVR natively supports tracking systems: we use the Intersense IS-900 inertial-ultrasonic motion tracking system to track user's head position inside the Virtual Environment (VE). In order to perform dislocated multi-point motion capture we set up an Optitrack FLEX:100 system composed by 12 infrared cameras: doing so it is possible to track hands and fingers using small light-weighted passive markers.

Our aim was to create a controller characterized by an interface that could completely involve the user and her/his senses, grounding on the high resolution and flexibility of OSC. Unfortunately the most used music software, like Logic Pro[5], MainStage[6] and Reason[7], do not still support this powerful protocol; hence we decided to exploit LiveAPI release, an un-official package but delivered by Ableton[8] itself. In this way, using Ableton Live 8 and LiveOSC, which works as an OSC server for Live, it is possible to operate a full bi-directional OSC data exchange between Live and our application (Figure 1).

[4]http://vrmedia.it/Xvr.htm

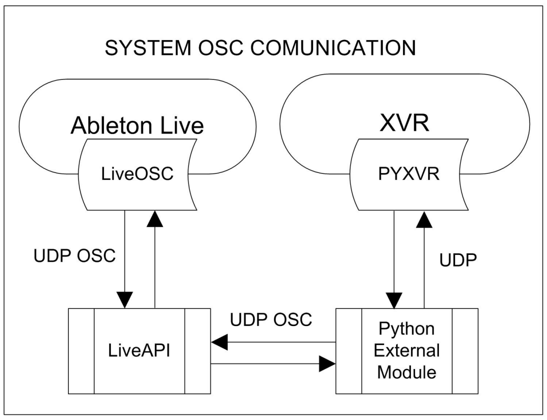

Figure 1: XVR and Ableton Live communication schema: on both sides serialized data are formatted by external modules.

4. USER INTERFACE

4.1 Hand Tracking

What is primary conveyed by the user interface we created is the possibility to quickly switch among four different commands: translation, triggering, un-triggering and de-selection of a 3D object. We decided to activate each command according to the posture of the hand that is interacting with the object. This means that each command it is not activated counting the number of fingers that are currently touching the object, but detecting instead which is the current posture as soon as interaction begins. This solution, compared to a simple finger counting algorithm, assures a more accurate interaction, permitting users to concentrate less on how many fingers are exactly on the object surface, and more on the desired music output.

4.2 System Set Up

The stereophotogrammetric motion tracking system we use is able to retrieve position of detected markers (point cloud) and position and orientation of customizable rigid bodies, composed by at least three markers.

It is very difficult to track the position of a specific marker inside a point cloud since markers are indexed according to the order of detection: this means that at each frame we may have a different id for the same marker. To overcome this issue we combined the use of rigid bodies and free markers, and we exploited some physical limitations of finger movements to obtain additional data on hand posture.

The set up of hand markers is very simple: users have been attached on the hands (using velcro straps) two small

[5]http://www.apple.com/it/logicstudio/logicpro/
[6]http://www.apple.com/it/logicstudio/mainstage/
[7]http://www.propellerheads.se/products/reason/
[8]http://www.ableton.com/

rigid bodies, a three marker rigid body for the right hand, and a four marker rigid body for the left hand. Doing so, it is always possible to univocally retrieve position and orientation of both hands. A small strip marker is attached on the last phalanx of index, middle and ring fingers of both hands. The absence of cables and heavy components permits users to move hands and fingers in an unconstrained and natural way [3].

In the VE an invisible box is positioned around each hand, with its top face adjacent to the plane passing through the back hand markers: this works as a convex hull for the fist of the user, so that postures could be distinguished counting the number of flexed fingers. Hand and finger dimensions differ from person to person, and, even if velcro strap can be easily adapted to each user, it is impossible to design a standard box that is never too large or to narrow to exactly contain the fist; to do so a calibration session is required. To avoid this annoying solution, we analyzed how phalanxes flex towards the palm in order to choose a heuristic to distinguish flexed fingers from stretched fingers, in the four postures we want to detect.

When holding a fist the last phalanx of each finger is in the middle of the palm, just below the knuckle line. This happens regardless of the dimension of the hand. Even when holding something smaller than the hand itself, fingers can be still considered flexed: there is no contact with the palm, but last phalanxes are again below the knuckle line. According to these remarks we decided to label as "stretched" each finger that is over the knuckle line or out of the volume between the back of the hand and a plane parallel to it, 8 cm far. We designed the hand hull boxes to quickly perform this double check on finger status. The dimension of the two boxes are 12cm x 8cm x 8cm; positioning the top most markers of the rigid bodies on the knuckle line, each box covers the volume where fingers move during bending. In this way, each time a finger is detected inside a box, both stretch conditions are false.

4.3 Detection Algorithm

The algorithm we are presenting permits to recognize five natural hand postures (Figure 2): single finger pointing, that is the common pointing posture with closed fist and the index finger stretched; two finger pointing, with index and middle finger stretched; three finger pointing, with index, middle and ring finger stretched; open hand, with index, middle, ring and pinkie fingers stretched; fist (detected but not used), with all fingers flexed. As thumbs are not tracked, all postures do not depend on their positions. The algorithm does not require any calibration session.

Since markers belonging to a rigid body are also included into the point cloud, the first part of the algorithm aims at distinguishing finger markers from back hand markers. The process is composed by two simple and very direct steps: in the first step the algorithm checks whether the distance from the current marker to each back hand marker is greater than a threshold (5 mm, minimum detection error of the tracking tool). In the second step we exploit physiological constraints on finger-hand distance to assign the finger: since fingers cannot move farther than about 20 cm from the respective hand, the two distances from the hand barycentres are compared, in order to define the point as a right or left hand finger. If the marker is too close or too far from both hands, the finger is not assigned. This is a drawback of the current algorithm, in fact until now it is only possible to interact with the VE if one hand is at least 20 cm far from the other.

The second part of the algorithm effectively recognizes the hand current posture, counting the number of fingers

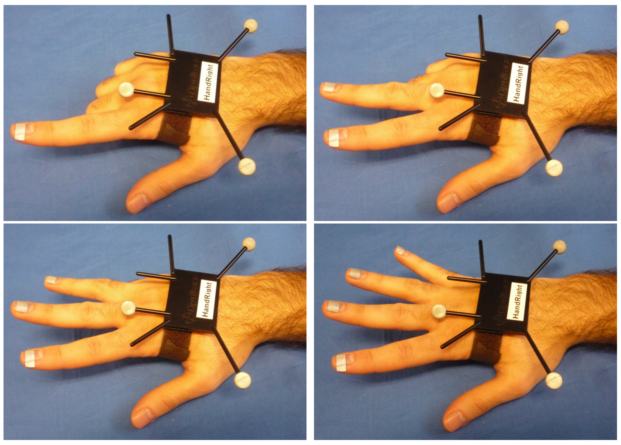

Figure 2: Marker configuration for the four hand postures. Thumb is not tracked, hence its position has no influence on the detection.

that are inside the hand hull; hence for both hands all the respective fingers are cycled. So far the algorithm does not recognize which are the stretched fingers, but only their number; this limitation produces some redundancy in posture recognition: for example the single finger pointing posture is detected if just any of the three fingers is stretched, and the other two are flexed. Although we intend to resolve this issue, it does not affect the usability of the interface, because all of the redundant postures (e.g. the middle-ring two finger pointing posture, the ring finger pointing posture) reveal to be unnatural, even difficult to adopt by most people. When detected, each of the four postures is associated to a specific interaction command among translation (one finger), triggering (two fingers), un-triggering (three fingers) and de-selection (four fingers, used to stop interaction at all).

When working on finger tracking it is important to notice that a flexed finger is hardly detected by infrared cameras, even if the scene is caught from different points of view: for example, each time we assume a pointing posture, our thumb moves over the flexed fingers, hiding the phalanxes where we use to set finger markers. We exploited this occurrence considering a non-detected finger as a flexed finger. So, a hand is assumed to be in a fist posture if all of its detected fingers are flexed, as well as if none of its fingers are detected.

5. VIRTUAL CONTROLLER

5.1 Controlling Objects and Music

The system provides full control of an Ableton Live set. When the application is launched the user is presented two panels that move together with her/him, containing interactive menus for object and mapping creation (Figure 3). Menus are structured in pages and voices. Menu voices can be highlighted just moving a finger on their surfaces, and selected using two fingers (i.e. two finger hand posture).

The object menu is showed on the left panel and can be used with the left hand only; it permits to create and delete interactive 3D shapes, and, when used together with sound menu, to edit mappings. Selecting the "Create object" voice, it is possible to choose from the "Create object" page the shape to create, among "Sphere", "Cube" and "Cylinder". Shapes thus created appear in front of the user and can be freely moved inside the VE. Objects can be

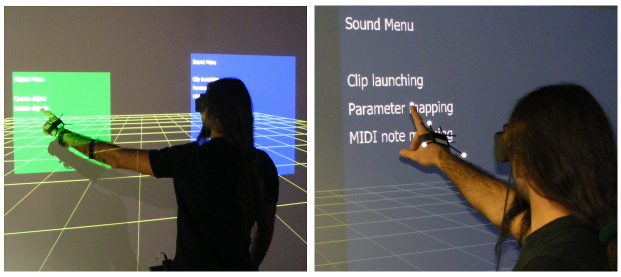

Figure 3: Interactive panels. Menu voices can be highlighted and selected moving one or two fingers over text surfaces.

permanently deleted using the "Delete object" voice in the main object menu.

The sound menu, showed on the right panel, is a much more powerful graphic user interface, structured in a complex dynamic way. However its use results to be very direct and simple. When the application is launched, all the track information of the current Live set are retrieved by the Python OSC module, and stored in a complex XVR structure (see next sub-paragraph). The VR controller permits to map the triggering of all the clips contained in Live tracks, and all the parameters contained in track devices, including general controls like volume, panning, mute and solo. The amount of stored data strictly depends on the complexity of the current Live set: some musicians work with many loops but few devices, concentrating on loop pattern combination; others, instead, prefer using a small set of loops, exploiting the various sound morphing possibilities provided by effects, synthesizers and other devices. This last approach determines a huge set of data to be retrieved by the application, so we developed the sound menu structure to permit a fast navigation and easily find the desired clip/parameter to map, even when working with hundreds of information.

The first "Sound menu" page divides clip launching from parameter mapping. In both cases the information set is divided into sub-pages: in "Clip launching" pages the list of tracks containing clips is first presented, then, selecting a track, it is possible to have access to all the contained clips. "Parameter mapping" pages contain the list of track containing devices, the list of devices in each track, and the list of all the parameters in each device. All tracks, devices, parameters and clips are identified by the name given by the user within the Live set: this feature, which really speeds navigation up, is based on OSC and LiveAPI callback functions. Once selected the desired clip or parameter, the object panel has to be used to complete mapping procedure.

General mapping procedure is summarized in Figure 4. Clip launching mapping is very simple: as described on the object panel, the user has to select which object to connect with the selected clip, just choosing it with two fingers. When mapping is successfully concluded, each time the object is touched with two fingers, the clip is triggered, while three finger selection forces clip to stop. Loop and quantization rules are the same as inside the current Live set.

Parameter mapping deals with the variation of floating point values between ranges, for this reason it is a more composite process. The key concept of the VR controller is to map the variation of a sound parameter onto the translation of an object between two coordinates (a maximum and a minimum) along a chosen axis. In this way each object can work as a tridimensional fader, free to move inside the volume of an invisible box, defined by the ranges set by X, Y, and Z axis mappings. When mapping for the first time a parameter onto a specific object, the object panel asks the user to select a reference point that works as barycentre for the tri-axial translations. Using the single finger posture it is possible to move the object to the desired position, defining a reference point that can be anyway edited in further interactions. The next steps consist in selecting an axis and a maximum or a minimum coordinate, as the two ranges are mirrored. From now on the object position is converted into a floating point value, in the range of the chosen device parameter, and sent via OSC to Live.

The final result is a maximum of 50 virtual objects that can be moved and triggered with both hands, each controlling up to 20 clips, and 20 parameters, divided onto the X, Y and Z axes (sharing reference point and ranges). The performer is clearly immersed into an interactive world, built according to her/his own preferences and needs during music creation (Figure 5); her/his gestures can be easily linked to the mechanisms by which the music is made, creating a visible tridimensional interaction with sound. The system is affected by an overall latency of approximately 10 ms, caused by the infra-red tracking tool.

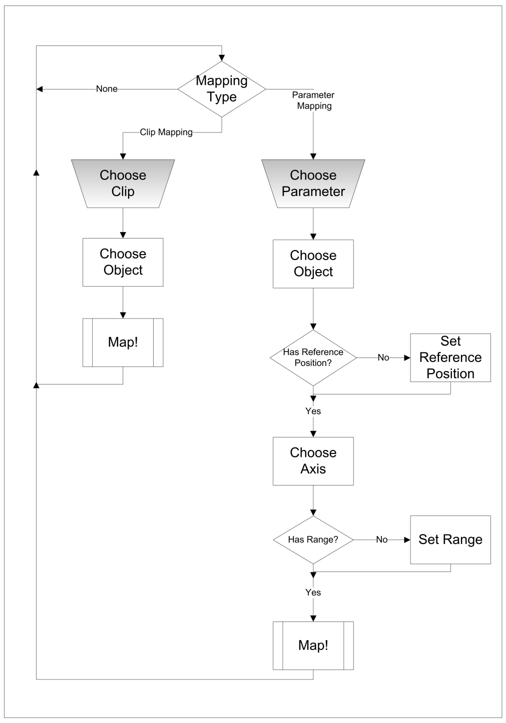

Figure 4: Clip and parameter mapping flow chart.

5.2 OSC Data

To execute Python scripts inside XVR we used an extension module called PYXVR[9]. Although XVR supports UDP communication, we decided not to format data, as requested by OSC protocol, directly within XVR scripts, exploiting instead Python as a glue language to manipulate messages to and from LiveAPI. In this way we send raw data (unformatted XVR data types) to Python module using UDP port 1246; Python module performs the four byte alignment necessary to communicate through OSC, and sends

[9]http://wiki.vrmedia.it/index.php?title=PYXVR

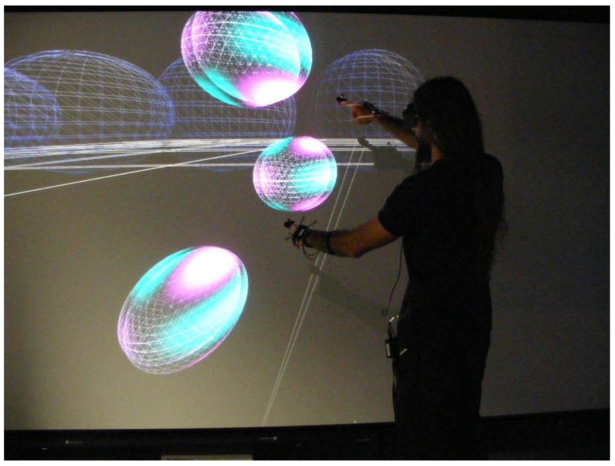

Figure 5: An user controlling two spheres as virtual 3D faders with both hands.

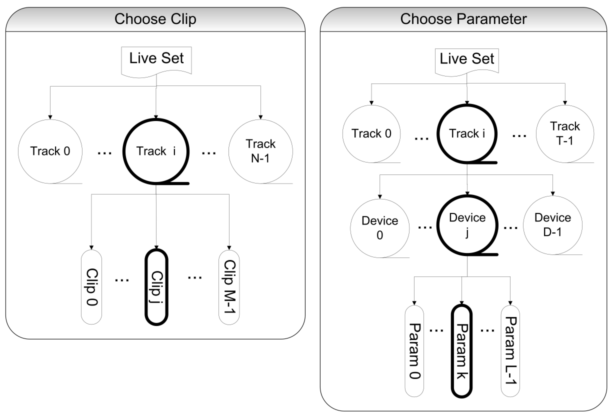

Figure 6: Using LiveOSC it is possible to navigate through the current Live set structure, directly within the Virtual Environment.

the message to Live, using UDP port 9000. When LiveAPI listeners return a controller change message from Live, data are sent to Python module on UDP port 9001, translated into XVR data types, and finally forwarded to XVR on port 1245.

The music menu of the VR controller includes much information about the current Live set, including the total number of devices and the name and the ranges of each parameter. These data are retrieved using a chain of functions that produce LiveAPI callbacks. All data sent by callbacks are received and stored in XVR by a single target function. A simple example is the string "*/live/tracks*", which forces LiveAPI callback that sends the total number of tracks in the current set. Each callback sends the requested data appended to a unique tag string: this tag is useful to understand what data has just been received, especially when many callbacks are pending.

All tracks are cycled at the application start up, in order to retrieve information about clips, devices, parameters and track status (i.e. volume, panning, mute, solo and arming). All data are stored in matrixes, indexed according to the number of the related track/clip/device/parameter. Doing so, a parameter is univocally defined by three integers: the number of the current track, the number of the current device, and finally the number of its position among all the other device parameters (Figure 6). The indexes of each mapped clip/parameter are stored within the object they have been mapped onto, so that, each time the object is triggered or moved, the correct OSC messages could be sent to Live interface.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented a virtual music controller, which supports OSC communications to map status and position of 3D objects on Live set parameters. The main characteristic of this system is a complete freedom in creating and associating virtual objects, according to the performer's preferences. The interactive VE thus created fully surrounds the user, creating a visible tridimensional interaction with sound.

Right now many features of the VR controller have to be enhanced, starting from mappings: although the mapping procedure is very direct, there is not yet the possibility to visualize all the clip/object and parameter/object connections, as showed in music software through MIDI browsers. However MIDI browsers do not manage in creating a quick reminder that links each parameter to the related controller. This happens because using hardware interfaces the only way to associate a physical knob or a fader to a software parameter is exploiting the MIDI CC message that is being currently used: there is not a direct visible connection between hardware and software, just a univocal numeric code. This kind of reminders can hardly be useful during a performance. Our system works instead with a set of completely software controller surfaces, 3D shapes that can be modified with textures and textual cues to display all the current associations: doing so all the information are always visible to the performer.

The more postures can be detected the more complex and expressive metaphors can be applied, in order to permit a very customizable and artistic music creation/control environment. For this reason, future developments also aims at tracking both thumbs; doing so many other natural postures could be recognized, and grasping, pinching and rotation could be used to manipulate meshes in a very direct and natural way.

A battery of tests is necessary to perform some important evaluations: we have already planned different sessions where digital music artists will be able to play and control their own music set (e.g. loops, effects) using our virtual interface. The VR controller was not designed as an alternative to hardware control surfaces, but as an additional device: subjects will be free to experiment cross-modality control, blending virtual and real devices into a personal live set. Analysis of numeric data is necessary to evaluate some system characteristics (e.g. efficiency, usability); furthermore questionnaires and artists' suggestions are useful to define which are the benefits and the shortcomings of virtual manipulation of music.

7. REFERENCES

[1] T. Beamish, K. Maclean, and S. Fels. Manipulating music: Multimodal interaction for djs. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2004.

[2] S. Campbell. Play+space: An ultrasonic gestural midi controller. In *Proceedings of the Australasian Computer Music Conference 2005*, 2005.

[3] K. Dorfmller-Ulhaas and D. Schmalstieg. Finger tracking for interaction in augmented environments. In *IEEE and ACM International Symposium on Augmented Reality (ISAR'01)*, 2001.

[4] A. Jensenius, R. Koehly, and M. Wanderley. Building low-cost music controllers. *Lecture Notes in Computer Science*, page 123129, 2006.

[5] S. Jordà˙Digital instruments and players: Part i - efficiency and apprenticeship. In *Proceedings of the 2004 conference on New Interfaces for Musical Expression*, 2004.

[6] S. Jordà˙Digital instruments and players: Part ii diversity, freedom and control. In *Proceedings of the International Computer Music Conference*, 2004.

[7] J. Malloch, S. Sinclair, and M. M. Wanderley. From controller to sound: Tools for collaborative development of digital music instruments. In *Proceedings of the International Computer Music Conference*, 2007.

[8] T. Mki-Patola, J. Laitinen, A. Kanerva, and T. Takala. Experiments with virtual reality instruments. In *Proceedings of the 2005 conference on New interfaces for musical expression*, 2005.

[9] M. Naef and D. Collicott. A vr interface for collaborative 3d audio performance. In *Proceedings of the 2006 conference on New Interfaces for Musical Expression*, 2006.

[10] K. Petersen, J. Solis, and A. Takanishi. Development of a real-time gestural interface for hands-free musical performance control. In *Proceedings of the International Computer Music Conference*, 2008.

[11] T. Reis, L. Carrio, and C. Duarte. Interaction design: The mobile percussionist. In *Proceedungs of the 4th International Haptic and Auditory Interaction Design Workshop Volume I*, 2009.

[12] X. Rodet, J.-P. Lambert, R. Cahen, T. Gaudy, F. Guedy, F. Gosselin, and P. Mobuchon. Study of haptic and visual interaction for sound and music control in the phase project. In *Proceedings of the 2005 conference on New interfaces for musical expression*, 2005.

[13] D. Voto, M. V. Limonchi, and U. DAuria. Multisensory interactive installation. In *Sound and music computing*, 2005.

[14] D. Wessel, R. Avizienis, A. Freed, and M. Wright. A force sensitive multi-touch array supporting multiple 2-d musical control structures. In *Proceedings of the 2007 conference on New Interfaces for Musical Expression*, 2007.

[15] M. Zadel and G. Scavone. Laptop performance: Techniques, tools, and a new interface design. In *Proceedings of the International Computer Music Conference*, 2006.