One at a Time by Voice: Performing with the Voice-Controlled Interface for Digital Musical Instruments

Stefano Fasciani[*], Lonce Wyse[†]

[*]Graduate School for Integrative Sciences & Engineering

[†*]Arts and Creativity Laboratory, Interactive and Digital Media Institute

[†]Department of Communications and New Media

National University of Singapore

stefano17@nus.edu.sg

Abstract

The Voice-Controlled Interface for Digital Musical Instruments is an alternate controller conceptually designed to be an extension to traditional touch-based musical interfaces. It allows simultaneous control of an arbitrary number of instrument parameters by variation of the performer vocal sound timbre. The generative and adaptive dual-layer mapping strategy makes an extensive use of unsupervised machine learning and dimensionality reduction techniques to compute hoc voice map for musical control. The aim is to maximize the breadth of explorable perceptual sonic space of specific digital musical instruments, providing dimensionality reduction of the instrument control space and adaptation to the vocal characteristics of the performer. In this paper we discuss mainly application and user perspective of the interface, together with a high level overview of the system. We describe the procedure to train and setup the system as well as the available user options and their effects on the interface response and instrument interaction. Finally we introduce a performance exclusively based on the Voice-Controlled Interface, in which one instrument at time is driven by the performer's voice and then looped, building up a an improvised composition.

Keywords: Voice Control, Musical Interface, Mapping, Unsupervised Machine Learning.

1. Introduction

A distinguishing and intrinsic feature of Digital Musical Instruments (DMI) is the physical independence of the sound generation component from the interface, despite the specific synthesis algorithm and the user input modality. This has promoted the proliferation of novel musical interfaces, mapping strategy and instrument control techniques, described and formalized in (Arfib and Kessous, 2002), (Paradiso and O'Modhrain, 2003), (Wanderley and Depalle, 2004), and (Miranda and Wanderley, 2006). Moreover, improvements on specifications and high availability of general-purpose computers, software development tools, software libraries, communication protocols, sensors, and hardware development kits have drastically streamlined the implementation process of musical interface. These improvements have also reduced the range of necessary skills for the designer, while constantly enhancing the musical interfacing potential. As we discussed in (Fasciani and Wyse, 2012a), where we firstly introduced our Voice-Controlled Interface for Digital Musical Instruments (VCI4DMI) approach, the concurrent use of multiple interfaces in musical performance is rarely addressed in literature. The majority of interfaces require hand gesture as the input modality, hence there is a bottleneck in the flow of musical intention to realization, or rather a limit in the number of events and parameters simultaneously controllable by the user. In this context the performer's voice can often be considered as a "spare bandwidth" (Cook, 2009), hence used as an extra source of gesture to ease this limitation.

Human voice has been adopted as the input modality for DMI interfaces and in (Fasciani and Wyse, 2013a) we review the existing literature highlighting application and limitation of the proposed approaches. The VCI4DMI aims to be generic and adaptive to implement ad hoc interfaces for any sound generation or processing device. We consider it as an extension more than an alternative to traditional touch based interfaces and the control target is an arbitrary number of real-valued time-continuous instrument parameters. The VCI4DMI maps continuous gestural parameters to continuous musical parameters (Kvifte and Jensenius, 2006) that mainly affect the instrument's timbre in an high dimensional space, and accordingly with Mulder (2000) it can be classified as an alternate controller. This design choice is meant to preferably, but not mandatorily, map discrete and time-critical musical parameters, such as temporal event triggering, to traditional touch-based interfaces, because potential errors and the intrinsic delay of the voice processing may have a sensible impact to the rhythm, harmony, or melody structures.

The use of the vocal timbre as the gestural input modality introduces high variabilities in the system. At the same time adapting the mapping to the parameters-to-sound characteristic of specific instrument is not trivial. Therefore to ease the system setup procedure we use automatic and generative strategies (Hunt et al., 2000), which make extensive use of machine learning and dimensionality reduction techniques to compute the mapping (Caramiaux and Tanaka, 2013). In the VCI4DMI we employ mostly unsupervised techniques to further minimize the amount of data that the user has to provide to the learning algorithms. This implies that the user, in turn, has to learn the outcome of the automatic mapping process, which can be further manually tuned at runtime.

2. System Overview

The VCI4DMI allows interfacing arbitrary instrument to arbitrary "vocal gesture" and it automatically adapts the mapping by a prior learning stage in which it analyses and processes the specific characteristics voice and instrument. The interface is based on a dual-layer strategy that maintains separated the adaptive map for the voice and the instrument. The analysis and learning processes are kept separated as well. This approach allows the reusability of the same voice map with different instruments and vice versa, a further simplification of the system setup.

The DMI analysis and mapping approach is based on an extension of the work presented in (Fasciani and Wyse, 2012b). In this context we consider a DMI any device that generates or processes a sound signal, thus a sound synthesizers or audio effects. Any combination of these still falls in the above-mentioned categories. The relationship between DMI's parameters and output sound must be deterministic and we assume no prior knowledge the synthesis/processing algorithms. The subset i_t of instrument parameters that will be mapped to the interface must be specified. Therefore an automatic procedure generates all the possible combinations of DMI parameters and for each one it analyses the output sound computing a vector of perceptual-related sonic features. The parameters and sonic data are stored respectively in the matrices **I** and **D**, which characterize the behaviour of the DMI. In order to analyze the output of sound processors, the system provides them with a test signal that is either noise or impulses. The different mode of analysis and options are described in Section 3. **D** represents the high dimensional sonic space that describes how the DMI sound varies within the i_t set. We reduce it to a lower dimensional space $\mathbf{D}^*$, typically to two or three dimensions in the VCI4DMI context, using non-linear dimensionality reduction techniques such as ISOMAP (Tenenbaum et al., 2000). The DMI mapping is based on the inverse relationship $\mathbf{D}^*$ to **I**, therefore the instrument's parameters are retrieved browsing the reduced sonic space, similar to intermediate perceptual layer approach presented in (Arfib et al., 2002). The arbitrary shape and distribution of $\mathbf{D}^*$ may lead to a difficult exploration of the space despite the used controller. We assume that the gestural control space has the same dimensionality as $\mathbf{D}^*$, but the samples are uniformly distributed into a hypercube. We find the transformation that maps the samples in $\mathbf{D}^*$ into an uniform hypercube and maintains the neighbourhood relations by a rank transformation followed by a truss structure node relocation iterative algorithm based on the Delaunay algorithm (Persson and Strang, 2004), also used for a similar purpose in (Lallemand and Schwarz, 2011). Finally a Neural Network (NN) learns the inverse of this nonlinear transformation, to be used in the runtime mapping. Since relationship parameters-to-sound may not be one-to-one, discontinuities may appear in the DMI parameters generation flow. To reduce this implicit shortcoming we reduce the search space in $\mathbf{D}^*$ to those points that are instantaneously within a user-defined distance in the space **I**.

On the other end of the system the VCI4DMI handles the variabilities that we find in different performer's vocal characteristics and gestures, as well as voice capturing hardware. The interface is designed to drive the instrument with unchanged parameters if the performer's voice timbre doesn't vary over time ("vocal posture"), while it allows the exploration of the instrument's sonic space when the voice changes over time ("vocal gesture"). The temporal unfolding of the gesture is not considered here because it would provide only a mono-dimensional control space. The VCI4DMI considers the multi-dimensional spatial distribution of gesture in order to provide a mean to explore the DMI's sonic space. From a set of user provided "vocal postures" and several instances of the same "vocal gesture" we compute a large set of heterogeneous low-level features (LPC, MFCC, and PLP). We search for the optimal features computation configuration, as described in (Fasciani, 2012). Noisy features over the postures are discarded, while the remaining features computed over the gesture define the high dimensional gestural input space **V**. We implement a Gestural Controller (GC) (Rovan et al., 1997) from this space using the Self-Organizing Gesture (SOG) method described in (Fasciani and Wyse, 2013b). The output lattice of the Kohonen's Self-Organizing Map is used as a GC, while the network is trained using a variation of the standard algorithm. The data in **V** is pre-processed using clustering, outlier detection, and dimensionality reduction techniques (ISOMAP or multiclass LDA) generating $\mathbf{V}^*$. The SOG technique compresses and expands the dynamics in **V** remapping them into a lower dimensional uniform hypercube, with the same dimensionality of the one embedded in **V**, that typically for voice data is two or three. It also guarantees a topology distortion-free transformation and continuity in the GC intermediate signals output.

The two learning stages described above map the "vocal gesture" and the DMI's sonic space to equal dimension and uniformly distributed hypercubes. Hence the connection between the two layers of the VCI4DMI is simply a linear one-to-one mapping. This is possible despite the different dimensionality, shape, distribution and number of elements in **D** and **V**. Indirect distance weighting is used in both spaces as an interpolation technique to cope with regions where the density of samples is low. In the runtime interface the hypercube mapping is direct in the voice related layer, while it is inverse in the DMI related section. In Figure 1 is an example of the two spaces after a reduction to two dimensions.

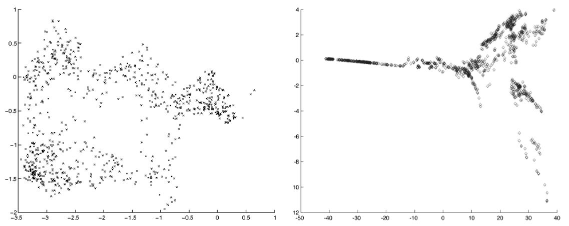


Figure 1. An example of **V** (left) and **D** (right) reduced to two dimensions by ISOMAP.

The differences between the spaces are evident and not predictable, as shown in the Figure 1 examples, therefore generative mapping techniques offer an effective solution in this context. The mapping tuning options and the DMI parameters retrieval preferences are described in the next section. Figure 2 presents simplified schemes of the VCI4DMI offline learning phase and runtime part. In the bottom section, the vector **c**, exchanged between the two layers, represents the GC output within the uniformly distributed hypercube.

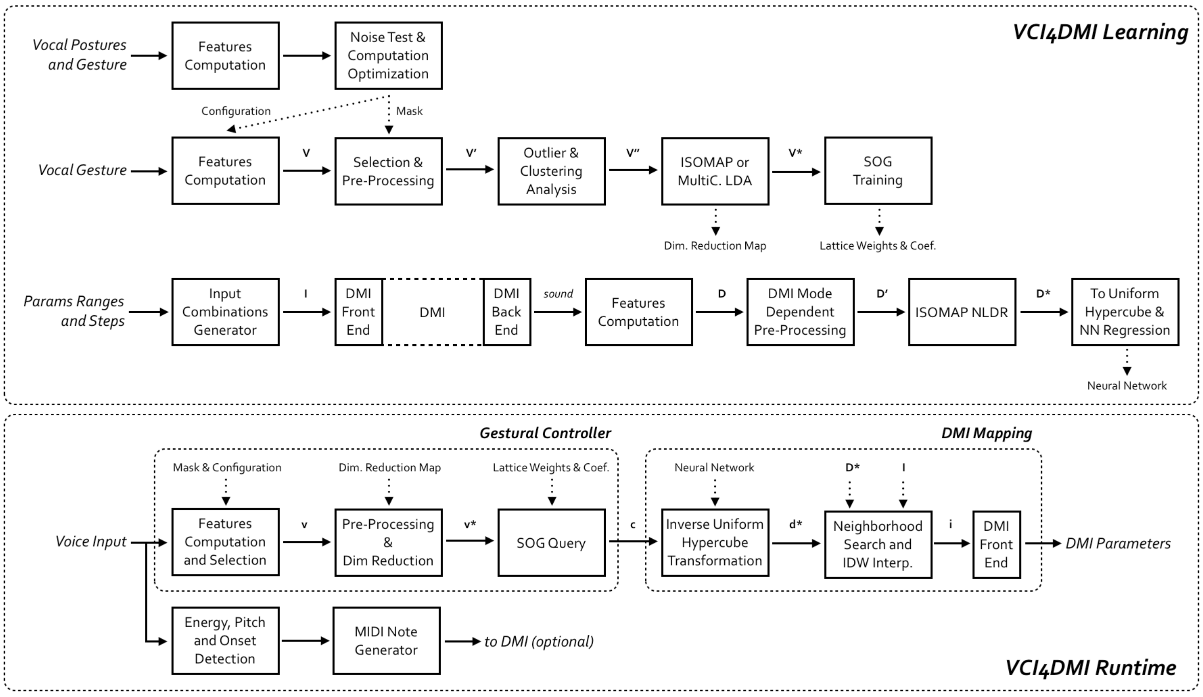


Figure 2. Simplified schemes of the VCI4DMI offline learning phase (top) and runtime part (bottom).

3. User Perspective

A functional prototype of the VCI4DMI[1] is implemented concurrently in Max/MSP and MATLAB. We developed a set of Max For Live devices to easily interface DMIs hosted in Ableton Live. In this section we describe the procedure that the user has to follow to train and use the system. Moreover we report all the available choices in the learning phase and in the runtime part. Most of these are exposed in the Max/MSP GUI, and eventually propagated to MATLAB using the OSC protocol.

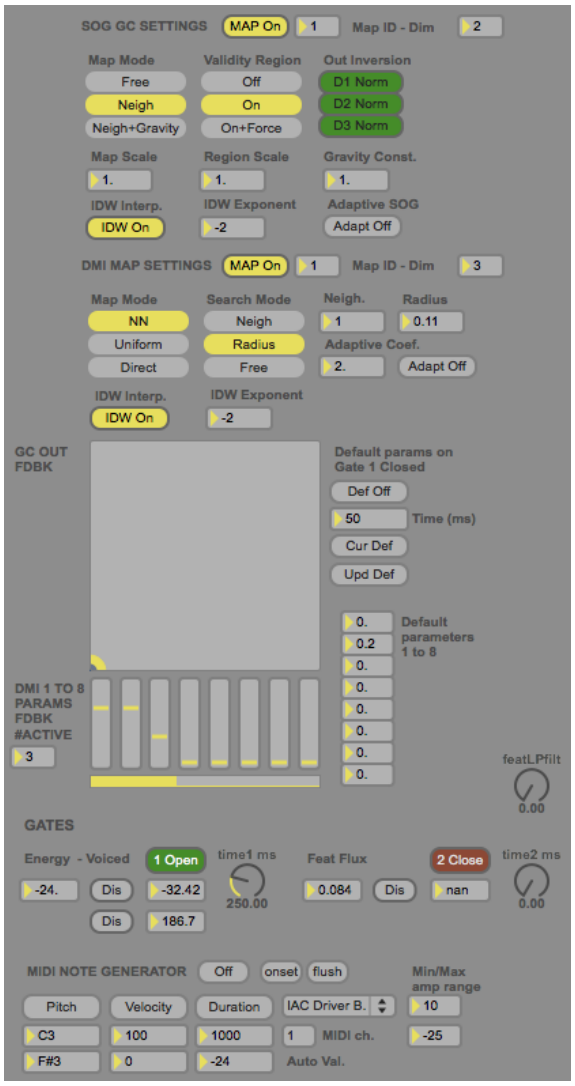
At first, the user has to identify the DMI parameters that will be controlled by the system (up to eight in the prototype but theoretically unlimited). For each one the user specifies range and analysis resolution. These directly affect the number of entries in **I**, in **D**, and in turn the overall analysis and learning time. From our experience, an excessive number of unique parameters combinations do not bring usability improvement to the system since an interpolation method is in place. For synthesizer, the analysis features available are the energy of the Bark bands, the MFC coefficients, or spectral moments. The user must also select one of the following analysis modes: sustain phase averaging, attack-decay-release envelopes, sustain phase dynamics. For effects it is possible to run the analysis in the frequency or in the time domain. In the first case we use noise (white, pink or brown) to excite the DMI and then it can be analyzed with the same synthesizers available modalities. For the time domain option, the sound processor is excited with an impulse and the response is captured. The impulse response can be down-sampled and used directly as a high dimensional feature vector or it can be

[1] http://stefanofasciani.com/vci4dmi.html

processed to compute features such as the total energy, slope, T60, number of peaks, and maximum amplitude. For both DMI categories, synthesizers and processors, the GUI allows the user to fine tune low-level analysis parameters such as: window size and overlap, number of bands, MIDI note and velocity, analysis timings, number of feature vectors per state, impulse response duration, and individual feature mask.

The SOG GC can be trained using any "vocal gesture" that comprises any sequence of voiced and unvoiced sounds. As mentioned before the temporal unfolding of the gesture is not considered by the system, but the focus is on the spatial unfolding of the low-level vocal features. The user must provide several instances of the same "vocal gesture" and the "vocal postures". A basic example of gesture is the gliding between the vowels [i] and [u], and the two related postures would be exactly the vocal sounds [i] and [u]. Gestures and postures can be captured "on the fly" or provided to the training script as audio files. In the latter case, all the feature computation parameters (sample rate, window size, window step, pre-emphasis, order of LPC, MFCC and PLP) are set automatically to optimal values found by an iterative algorithm, instead of leaving them to default values or user-defined values.

The mapping layers generated in the learning stages are saved into separate memory structures and dumped to binary files, so that they can be reutilized in different interface implementations. When starting the runtime VCI4DMI, the user has to load a pair of maps, one for the voice and another for the instrument. Since we used only unsupervised techniques in the learning process, at first the user should learn and get familiar with the mapping that the system has generated. However the GUI exposes a large set of parameters and options that advanced users can use to tune or change the default VCI4DMI response. These, visible in the GUI detail in Figure 3, can be changed while the system is running. The response can be modified between these two theoretical extremes: specific vocal timbres mapped to specific DMI's sounds, or vocal timbre variations to browse the DMI sonic space in different directions. The VCI4DMI stores all the user preferences in the map file so that is not necessary to configure the map again after a system restart.


SOG GC SETTINGS: MAP On; 1; Map ID - Dim: 2
Map Mode: Free / [✓ Neigh] / Neigh+Gravity
Validity Region: Off / [✓ On] / On+Force
Out Inversion: D1 Norm / D2 Norm / D3 Norm
Map Scale: 1.
Region Scale: 1.
Gravity Const.: 1.
IDW Interp.: IDW On
IDW Exponent: -2
Adaptive SOG: Adapt Off

DMI MAP SETTINGS: MAP On; 1; Map ID - Dim: 3
Map Mode: [✓ NN] / Uniform / Direct
Search Mode: Neigh / [✓ Radius] / Free
Neigh.: 1
Radius: 0.11
Adaptive Coef.: 2.; Adapt Off
IDW Interp.: IDW On
IDW Exponent: -2

GC OUT FDBK
Default params on Gate 1 Closed: Def Off; 50 Time (ms); Cur Def; Upd Def
Default parameters 1 to 8: 0.; 0.2; 0.; 0.; 0.; 0.; 0.; 0.
DMI 1 TO 8 PARAMS FDBK #ACTIVE: 3
featLPfilt: 0.00

GATES
Energy - Voiced: 1 Open; -24.; Dis; -32.42; Dis; 186.7
time1 ms: 250.00
Feat Flux: 2 Close; 0.084; Dis; nan
time2 ms: 0.00

MIDI NOTE GENERATOR: Off; onset; flush
Pitch: C3; F#3
Velocity: 100; 0
Duration: 1000; -24
IAC Driver B.; 1 MIDI ch.; Auto Val.
Min/Max amp range: 10; -25

Figure 3. Detail of the VCI4DMI GUI section related to the runtime parameters and options.

For the SOG based GC the user can choose to restrict the iteration movement in the lattice to the Moore neighbourhood and optionally use the gravity instead of the Euclidean distance as the search metric. He can also set how voice vectors outside the validity region are handled: considered, ignored, or projected to the boundary of the region. Scaling coefficient can be applied to the lattice and to the validity region, while the mapping of the single GC output dimensions can also be inverted. An option to adapt the map during runtime is available as well. It simply applies an automatically computed offset to every lattice dimension lattice.

In the DMI mapping section the user can select to bypass the NN and map the GC output either to the original $\mathbf{D}^*$, or to the $\mathbf{D}^*$ rearranged into the uniformly distributed hypercube, useful when the NN learning performance is poor. The following search can be across the entire sonic space, restricted to the first or second Moore neighbourhood level, within a user defined radius, or within an adaptive radius determined from to the instantaneous vocal gesture variation. In all these cases the neighbours or the points within the radius are measured in the parameters space $\mathbf{I}$ and brought back to $\mathbf{D}^*$. The indirect distance weighting interpolation for the SOG GC and for the DMI mapping can be disabled and the interpolation exponent modified. Moreover both layers can be bypassed, allowing the incoming data to pass through.

There are three available gates to temporary disable the interface, interrupting the flow of input gestural data. These are based on energy, low-level features vector distance, and voiced/unvoiced detection. Each one has an independent threshold value and opening time. Moreover it is possible to define a default DMI parameters configuration to which the system will force the instrument when the gates are closed. The user can also define the transition time to the default parameters. The VCI4DMI can optionally generate MIDI notes to trigger sound generators. Energy, pitch and onset detected are mapped one-to-one to their MIDI equivalents. The pitch, velocity and duration can be manually defined or individually set to automatic mode so that they are derived from the vocal input.

There are four real-time visual feedback provided to the user: position of the GC output in the hypercube, DMI parameters, position and transitions in the SOG lattice with validity region, and position in sonic space with active neighbourhood highlighted. The latter two are not visible in Figure 3. The visual feedbacks play a key role in getting familiar and tuning the automatically generated mapping. Moreover the system can be tuned to respond more smoothly or sharply by changing the voice signal feature computation window overlap, or the feature one-pole low pass filter. In the current implementation, partially optimized, the interface can generate a new DMI parameters vector every 8ms in worst-case scenarios (4ms typical), that is further linearly interpolated every 1ms.

4. "One at Time by Voice" Setup

The version of VCI4DMI used in this performance was specifically developed with additional features to fit the needs of this scenario. The voice is the only gestural input used to interact with the DMIs, therefore fast recalling of mapping presets, seamless interaction with the DMI hosting Digital Audio Workstation (DAW), off screen visual feedback and minimalistic hand interaction are essential. This specific version loads a bank of voice maps and a bank of instrument maps. Individual elements of the two banks can be paired and saved into a specific preset, which can be recalled with the press of a single button on an external interface or on a remote control. Each preset stores also information about DAW interaction, such as instrument track, input mode (parameters and/or MIDI), record mode, and quantization. System visual feedback are encoded and routed to a 9 by 9 LED matrix. To load a new pair of maps the system temporary disables the interface for as little as 150ms.

The pool of instruments and effects loaded in Ableton Live for this live performance was selected to demonstrate the wide range of sonic interaction capabilities offered by the interface. While the DAW loops on 4 bars, the performer builds up an improvised composition recording on each instrument's track the MIDI and instrument parameters generated by the VCI4DMI. Only one instrument at a time is under the direct control of the performer. However for each DMI the performer can control as many as eight real valued parameters simultaneously and trigger MIDI notes. The track recording and the MIDI generation is optionally enabled so that the performer-instrument interaction can be limited to just parameters modulation in certain cases.

References

Arfib, D., Couturier, J.M., Kessous, L. and Verfaille, V. (2002), "Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces", *Organized Sound*, Vol. 7 No. 2, pp. 127–144.

Arfib, D. and Kessous, L. (2002), "Gestural Control of Sound Synthesis and Processing Algorithms", *International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction*, GW '01, Springer-Verlag, pp. 285–295.

Caramiaux, B. and Tanaka, A. (2013), "Machine Learning of Musical Gestures", *Proc. of NIME 2013*.

Cook, P.R. (2009), "Re-Designing Principles for Computer Music Controllers: a Case Study of SqueezeVox Maggie", *Proc. of NIME 2009*.

Fasciani, S. (2012), "Voice Features For Control: A Vocalist Dependent Method For Noise Measurement And Independent Signals Computation", *Proc. of DAFx-12.*

Fasciani, S. and Wyse, L. (2012a), "A Voice Interface for Sound Generators: adaptive and automatic mapping of gestures to sound", *Proc. of NIME 2012*.

Fasciani, S. and Wyse, L. (2012b), "Adapting general purpose interfaces to synthesis engines using unsupervised dimensionality reduction techniques and inverse mapping from features to parameters", *Proc. ICMC 2012.*

Fasciani, S. and Wyse, L. (2013a), "Mapping the Voice for Musical Control", *Technical Report*.

Fasciani, S. and Wyse, L. (2013b), "A Self-Organizing Gesture Map for a Voice-Controlled Instrument Interface", *Proc. NIME 2013*.

Hunt, A., Wanderley, M. and Kirk, R. (2000), "Towards a model for instrumental mapping in expert musical interaction", *Proc. NIME 2000*.

Kvifte, T. and Jensenius, A.R. (2006), "Towards a coherent terminology and model of instrument description and design", *Proc. NIME 2006*.

Lallemand, I. and Schwarz, D. (2011), "Interaction-Optimized Sound Database Representation", *Proc. DAFx-11*.

Miranda, E.R. and Wanderley, M.M. (2006), New digital musical instruments: control and interaction beyond the keyboard, A-R Editions, Inc.

Mulder, A. (2000), "Towards a choice of gestural constraints for instrumental performers", *Trends in gestural control of music*, pp. 315–335.

Paradiso, J.A. and O'Modhrain, S. (2003), "Current Trends in Electronic Music Interfaces. Guest Editors' Introduction", *Journal of New Music Research*, Vol. 32 No. 4, pp. 345–349.

Persson, P.-O. and Strang, G. (2004), "A Simple Mesh Generator in MATLAB", *SIAM Review*, Vol. 46 No. 2, pp. 329–345.

Rovan, J.B., Wanderley, M.M., Dubnov, S. and Depalle, P. (1997), "Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance", *Proc. of 1997 Kansei*.

Tenenbaum, J.B., Silva, V. and Langford, J.C. (2000), "A global geometric framework for nonlinear dimensionality reduction", *Science*, Vol. 290 No. 5500, p. 2319.

Wanderley, M.M. and Depalle, P. (2004), "Gestural Control of Sound Synthesis", *Proc. of the IEEE*, Vol. 92 No. 4, pp. 632–644.