A Self-Organizing Gesture Map for a Voice-Controlled Instrument Interface

Stefano Fasciani[1,2] Lonce Wyse[2,3]
[1]Graduate School for Integrative Sciences & Engineering
[2]Arts and Creativity Laboratory, Interactive and Digital Media Institute
[3]Department of Communications and New Media
National University of Singapore
{stefano17, lonce.wyse}@nus.edu.sg

ABSTRACT
Mapping gestures to digital musical instrument parameters is not trivial when the dimensionality of the sensor-captured data is high and the model relating the gesture to sensor data is unknown. In these cases, a front-end processing system for extracting gestural information embedded in the sensor data is essential. In this paper we propose an unsupervised offline method that learns how to reduce and map the gestural data to a generic instrument parameter control space. We make an unconventional use of the Self-Organizing Maps to obtain only a geometrical transformation of the gestural data, while dimensionality reduction is handled separately. We introduce a novel training procedure to overcome two main Self-Organizing Maps limitations which otherwise corrupt the interface usability. As evaluation, we apply this method to our existing Voice-Controlled Interface for musical instruments, obtaining sensible usability improvements.

Keywords
Self-Organizing Maps, Gestural Controller, Multi Dimensional Control, Unsupervised Gesture Mapping, Voice Control.

1. INTRODUCTION
In Digital Musical Instruments (DMI) design, the Gestural Controller (GC) [1] plays an essential role: it converts the input gestural data into intermediate signals that are mapped to sound synthesis or processing parameters. The GC, together with the mapping, defines the relationship between the performer's gesture and DMI sonic response. Its design is considered a specialized branch of HCI where the simultaneous and continuous control of multiple parameters, the instantaneous response, and the necessity of user practice are key aspects [2] [3]. The algorithm to interpret the gestural data, performed by the GC, depends on the nature of the sensors employed in the interface and also on designer choices. Hunt, Wanderley, and Kirk [4] classify systems for gestural acquisition into three categories: direct, indirect and physiological. For the first category, each sensor captures a single feature. Correlations, dependencies, redundancy and constraints across the different sensor data can be derived and handled directly by the physical characteristic of the performer and the sensors. Hence knowing how the performer's gesture is represented in the gestural data domain allows the implementation of explicit strategies within the GC. For the other two categories, finding the relationship between a gesture and the captured gestural data may be challenging, therefore generative mechanisms, such as learning algorithms, are often used for the model estimation [5].

In this paper we propose a method to obtain a GC through unsupervised learning on a set of gesture examples. We assume that the gestural data is high dimensional, continuous, and contains potential correlations, cross-dependencies, and it is not uniformly distributed. The GC we propose here has output dimensionality generally lower than the input, its output signals are continuous and have no cross-constraints across the individual dimensions. Considering the gestural examples provided as a sequence of instantaneous postures snapshots, the GC relates its unique outputs combinations to unique postures. A performer gesture would therefore produce a continuous modification of the DMI parameters, changing the properties of the sound synthesis or processing algorithm. The non-linear transformation performed in the GC outputs also if the input gesture differs from the provided examples, but respecting the lower dimensional spatial bounds, topology, and distribution found in the example data set.

This method is particularly suited for "alternate controllers" [4] with indirect and physiological gestural data acquisition, such as those with a large set of features extracted from an audio or video signal, or from a network of sensors, but it can be applied to high dimensional direct acquisition as well where the implementation of explicit strategies can be challenging. From a broad range of potential application scenarios, in this paper we present and discuss the integration of this GC technique with our Voice-Controlled Interface (VCI) [6]. The VCI is an alternate controller for DMIs with gestural acquisition lying between the categories of indirect and physiological which estimates pseudo-physiological characteristics of the vocal tract through the analysis of the vocal audio signal. The VCI is meant to provide performers with vocal control over a multidimensional instrument real-valued parameters space. Therefore the GC we present in this work does not perform temporal gesture recognition, but it samples instantaneous postures from a gesture and maps these in an intermediate space with a bijective correspondence with DMI parameters. The key aspect of this work is the utilization of a multidimensional extension of the Kohonen Self-Organizing Maps (SOM) [7] lattice to achieve a geometrical transformation of the gestural data space into the mapping space. We perform the gestural data dimensionality reduction using non-linear techniques before the SOM training to overcome some limitations and shortcoming of the SOM lattice, which are evident when using the transformation mentioned above for GCs. Moreover, we introduce and motivate some variation in the SOM training algorithm as well.

2. RELATED WORK

Artificial Neural Networks (ANN) have been used in the design of new interfaces for musical purposes for more than two decades. From the pioneering work of Lee and Wessel [8] to Fiebrink's more recent Wekinator [9], history shows how these generative algorithms have been successfully used to learn gesture maps for DMIs, particularly for time-continuous real-valued parameters. Typically an ANN is trained with supervised techniques. On the other hand, the Kohonen SOM ANN is trained with an unsupervised technique. SOMs are commonly used to produce a two-dimensional discrete representation, called a map or lattice, of the input space. SOMs, contrary to traditional ANNs, tend to preserve the topological properties of the input space. Therefore, SOMs have been mainly used for classification, visualization, organization or retrieval of audio, exploiting the dimensionality reduction and topological organization capabilities. Ness and Tzanetakis [10] present a comprehensive survey of SOM applications in music and instrument interface related projects. Some instrument interfaces using SOMs are limited to a one-to-one linear mapping of a two or three-dimensional sensor to an equal dimensional trained map. Then the music or audio chunks previously used to train the map are retrieved and reproduced in real time, as in Odowichuk [10], which represents one of the few musical examples using an SOM output lattice with dimensionality larger than two.

Stowell [11] describes an attempt to use the SOM for remapping vocal timbre representing the gestural data to sound synthesis. He explains the relatively poor performance of this solution with the intrinsic shortcomings and limitations of the SOM. The selection of appropriate training settings, output lattice resolution, and dimensionality may vary drastically case by case. There may also be differences in map orientation for different training over the same data set, and errors in topology preservation such as output lattice twisting or folding. The latter two are harmless side effect in classification tasks but are lethal when the output lattice must represent a continuous and non-linear discretization of the input data set, as we wish to obtain here. The SOM performs a dimensionality reduction of the input data and at the same time it tries to maintain its topology. As described in [12], this dualistic role of the SOM is one of the sources of topology corruption. If the embedded dimensionality of the input data is higher than the output lattice dimensionality, topology corruption is highly probable. Moreover there is a tradeoff between continuity and resolution of the map. The SOM output, represented by the lattice node position, is intrinsically discrete. Therefore when we mention map continuity within this context we mean that vectors very close in the input manifold are mapped either to the same or adjacent nodes. These issues, their effect on a SOM-based GC, and proposed solutions are covered in the next section.

3. SELF-ORGANIZING GESTURES

In this section we describe the procedure to train the SOM-based GC and different operative modalities which may fit different DMI interface requirements. As mentioned above, our goal is to drive time-continuous and real-valued DMI parameters. The set of gestures used for the training represents a temporal sequence of instantaneous postures, defining an arbitrary shape in an N-dimensional space with arbitrary and generally non-uniform density. On the output side, the GC generates data in an M-dimensional space, therefore it applies a bijective transformation $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$. It is desirable that the transformation f maintains the local and global topological structure of the training gestures and spreads them evenly over the output dimensions. The training data subject to the transformation f must uniformly fill in a hypercube with dimensionality M. An empty or low-density sub-volume in the output hypercube results in a GC constraint that makes it impossible to obtain certain combinations of the M output parameters with any gesture similar to the training ones. Independently of the strategy used to map the GC output to the DMI (one-to-one, divergent or convergent), the transformation f described above implicitly requires that M is equal to or smaller than N. In high-dimensional gestural acquisition we often find an embedded dimensionality N' smaller than N, therefore if we want non-redundant GC outputs, M has to be equal or smaller than N'.

In principle, the SOM seems to be a suitable solution in learning the transformation f from the gestural examples. The GC can be implemented interpolating the discrete output map of the trained ANN, which has N input neurons and an M dimensional output lattice. However, due to the shortcomings in topology preservations described in the previous section, the GC can be affected by the following problems detrimental to musical instrument design:

- inverted GC response to the gestural input belonging to different subsections of the input space due to SOM lattice twisting;
- discontinuous GC response due to excessive SOM folding, (resulting in the proximity of two or more edges in the output lattice), or due to SOM curled edges;
- minor inconsistency of the GC behavior due to local topology distortion of individual output nodes relative to its neighbors.

Our approach aims to avoid or minimize topology corruptions by introducing a prior dimensionality reduction step, proper data preprocessing, and some expedients in the training algorithm. We free the SOM from the dimensionality reduction task, while using it to find a non-linear geometrical transformation between the two iso-dimensional spaces, and to compresses/expands the dynamic of the input. A similar approach for a different application domain is taken in the Isometric SOM [13], where a 3D hand posture estimation, is achieved using the SOM with a prior dimensionality reduction stage performed using the ISOMAP method.

3.1 Learning Process

The preparation of a proper training data set **G** is essential for the success of the learning algorithm. The larger the data set, the better the final system will likely respond. However, consistency and compliance of the data set are fundamental. Most sensor output data are sampled at regular intervals. Maintaining a posture over time during the recording of **G** generates inappropriate high-density cluster in the N-dimensional space, due to the sampling of identical or very similar postures. This can bias and corrupt our training process because the data density is not representative of the gesture only. Strategies to avoid the presence of identical postural data may vary with the characteristics of the gestural data acquisition system, or alternatively, a more general step of post processing over the acquired data **G** can be used. For our VCI we adopt a gate operated by the spectral flux value to filter out undesired postures while collecting the gestural data vectors in **G**. The spectral flux threshold value is automatically measured from a set of postures, used as well for noisy features rejection.

3.1.1 Gestural Data Pre Processing

A single dimension here represents a sensor signal for direct gestural acquisition, or a single feature for indirect and physiological gestural acquisition. The N-dimensional gestural training data **G** is centered at the origin removing the mean

$\mathbf{g}_{mean}$. Then we estimate N', the embedded dimensionality in the N-dimensional gestural training data. This step is essential because after the dimensionality reduction we keep at most N' dimensions, possibly fewer depending on user choice. For this estimation we round the results of the correlation dimension method [15] to the closest integer. For the dimensionality reduction there are several possibilities and experiments (e.g. [16]) that demonstrate a specific technique outperforms others only if certain specific characteristics are present in the data, while for real data scenarios often the most advanced techniques gives just a small improvement over basic techniques such as the Principal Component Analysis. Since we cannot make any assumption about the gestural data, choosing a Nonlinear Dimensionality Reduction (NLDR) technique would handle complex cases and non-linear manifolds. We chose a convex NLDR technique because it produced consistent results over different experimental iterations. We tested several NLDR techniques on real gestural data sets, and those that produced the best results were ISOMAP [17] followed by LLE [18]. The results were measured in terms of uniformity of data distribution across single dimensions, as well as the overall GC user experience. However, the tests were performed integrating the Self Organizing Gestures method with our VCI covered in the next section. It is certainly possible that for different gestural data acquisition systems the optimal NLDR might differ from those used here. After the NLDR, the data on each dimension are normalized to the range *[-1;1]*.

3.1.2 SOM Training

As mentioned above, the SOM output lattice dimensionality M will be at most equal to N'. However the output dimensionality of the GC M is user-configurable. The NLDR technique ranks the output dimensions, hence we simply discard the lowest ranked (N'- M) components to achieve the further reduction. An excessive number of output nodes, often referred to as output lattice resolution, is one of the causes for topology distortion, especially of the local type. We derive the resolution r from M and from the number of entries g in the gestural training data $\mathbf{G}$ using a nonlinear relation as in (1). A high-resolution implies a more complex model to be estimated, and the requirement of more training data is a direct consequence. Similarly, the number of SOM training iterations t_{MAX} is related to the model complexity as in (2).

$$r = \mathrm{round}(1.5 \cdot \log_M(g)) \tag{1}$$

$$t_{MAX} = M \cdot g \cdot r \tag{2}$$

The SOM output lattice represents an M-dimensional hypercube with r^M output nodes O_j distributed uniformly (forming a grid), each associated to an M-dimensional weight vector $\mathbf{w}_j$. Before initializing the weight vector $\mathbf{w}_j$ and training the SOM, we search the gestural training data for 2^M points a_k representing the extrema of the performer's gestures that best encompass the gestural space. The data is centered on the axis origin, hence we perform the search with the following steps:

1. for each quadrant, set an equal-value boundary on each axis, and increase it progressively reducing the quadrant extension until only one point remains, and compute the sum of the distances of the 2^M a_k points from the origin and between themselves;
2. repeat the previous step performing a sequence of fine angular full data rotation steps around the origin;
3. pick the rotation angle α_{opt} and the related a_k that maximizes the sum of the distances.

The weights $\mathbf{w}_j$ are initialized distributing them uniformly through the hypercube inscribed into the dataset, while the weights related to the O_j located at the 2^M vertices are initialized at the position of the a_k. The SOM is trained for t_{MAX} iterations picking a random point $\mathbf{x}_{rand}$ from the dataset and updating all the $\mathbf{w}_j$, as in equations (3) and (4)

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \mu(t) \cdot \Theta(t,j,z) \cdot (\mathbf{x}_{rand} - \mathbf{w}_j(t)) \tag{3}$$

$$\Theta(t,j,z) = \exp\left(-\left|O_z - O_j\right|^2 \big/ 2\cdot\sigma(t)^2\right) \tag{4}$$

where $\mu(t)$ is the linearly decreasing learning rate, $\Theta(t,j,z)$ is the neighborhood function described in (4), z is the index of the output node O_z with the related weight closest to $\mathbf{x}_{rand}$. In (4) the nominator of the exponential is the squared Euclidean distance of the output nodes in the output M-dimensional grid, $\sigma(t)$ is the linearly decreasing neighborhood parameter representing the attraction between the output nodes. We apply the following modification to the standard training algorithm:

- the O_j of the 2^M vertices are updated more slowly than the rest of the points, using the half of the learning rate $\mu(t)$;
- the random point selection is biased in favor of the a_k, forcing the weights update using the 2^M a_k every time a 10% of t_{MAX} has elapsed, and using half of the attraction $\sigma(t)$ for the O_j of the 2^M vertices.

At the end of the training process each output node O_j is associated with a number C_j counting how many entries in the training gestural data are associated with it (for which it is the "nearest node"). Additionally we embed information about the temporal unfolding of $\mathbf{G}$ in the lattice output nodes. This is used in one of the operative modes described in the next subsection. Since we are running a large number of training iterations, we choose a relatively small value for the initial learning rate $\mu(t_0)$. To avoid local topology distortions, we set the final value of the attraction $\sigma(t_{MAX})$ to a relatively large value, and a $\mu(t_{MAX})$ at least one order of magnitude smaller than the initial one. In particular, values producing good results for our vocal gestural data cases are: $\mu=[0.5,0.01]$ and $\sigma=[1.5,0.5]$. Other gestural data acquisition systems may require different parameter values to obtain a well-trained system, but the selection strategy is still valid.

The idea behind the modification of the conventional training is to avoid the topology distortions mentioned above, and at the same time obtain a better overlap between the weights of the SOM output nodes and the gestural data. In particular, we focused in pulling the lattice from its vertices, stretching it toward the gestural extrema, achieving at the same time lattice edges enclosing the M-dimensional gestural data subspace with higher fidelity. Moreover, the SOM training data is also used to find the M-dimensional convex hull that encloses all the vectors in $\mathbf{G}$. The convex hull defines the valid region for new incoming vectors during the real-time operation, so that the GC does not produce an output if the vector is outside the region. In Figure 1 we present two examples of SOM training on gestural data, with $M=2$ and $M=3$, showing the training data (after preprocessing), the weights of the SOM nodes at initialization and after training, node mass C_j and node neighborhood based on gestural data temporal unfolding. The weights of the SOM nodes are represented in the normalized M-dimensional training data space.

The preprocessing and training processes presented are completely unsupervised. In case the user wants to explicitly define the 2^M gestural extrema to be associated with the SOM lattice vertices, we use a supervised dimensionality reduction technique in the preprocessing stage. This requires an additional training data set $\mathbf{E}$ that contains several instances of labeled gestural extrema. We use a multiclass Linear Discriminant Analysis (LDA) for the N to M dimensionality reduction, which maximizes the between-class variance while minimizing the within-class variance. The LDA transformation is learned from the labeled data in **E**, and it replaces ISOMAP. The LDA dimensionality reduction is then applied to **G** and to the 2^M class centroids in **E**, representing the gestural extrema. The rest of the preprocessing and SOM training procedure is unchanged. With LDA, when projecting the gestural data in to the lower dimensional space, the extrema are located somewhere along the boundaries of the enclosing convex hull, insuring SOM topological preservation when associating the extrema with the vertices of the output lattice.

3.2 GC Operational Modes

The SOM-based GC is implemented applying the bijective transformation *f*, composed by a series of step in which we apply the processing learned from **G**. For a new gestural data vector **d**, we obtain the output vector **p** as follows:

1. subtract the mean $\mathbf{g}_{mean}$ from **d**;
2. apply the NDLR and truncate the vector to the top *M* dimensions if *M* is smaller than *N'*;
3. normalize each dimension to the range *[-1;1];*
4. rotate the vector by the angle α_{opt};
5. verify if the vector is within the valid convex hull;
6. find the closest $\mathbf{w}_j$ to the obtained $\mathbf{d}^*$ and output the related O_j normalized grid indexes vector **p**.

In Figure 2 we summarize the learning process and the GC processing steps. The operations in the square boxes within the learning process, learn and apply a transformation from **G**. Those in the rounded boxes within the GC simply apply the transformation based on the information learned.

The GC output **p** contains the indexes of the output node, which is its position in the *M*-dimensional lattice. Usually the indexes are integers numbers, but we normalize them in the range *[0,1]*. The resolution *r* in (1) directly defines the resolution of the *M* dimensions of **p**, which are based upon a user defined mapping strategy and used to drive DMI parameters. If *r* is small, we use the Inverse Distance Weighting (IDW) interpolation between the 2^M closest $\mathbf{w}_j$ to provide virtually infinite output resolution. The SOM output lattice can be used in more complex manners to implement variations of the GC which may suit different instrument interface philosophies.

To obtain a more smooth response **d**-to-**p**, it is possible to force the output to follow a connecting path on the lattice Moore neighborhood. The search for the $\mathbf{w}_j$ closest to $\mathbf{d}^*(t)$ is performed only on the 3^M neighbors of the $\mathbf{d}^*(t\text{-}1)$ closest O_j, including itself. If $\mathbf{d}(t\text{-}1)$ and $\mathbf{d}(t)$ are far apart due to a quick performer gesture, the GC responds more slowly. This can also help to compensate for potential performer error in actuating the interface, or noise in the gestural acquisition system. If $\mathbf{d}(t+1)$ is again close to $\mathbf{d}(t\text{-}1)$, such big variations in the gestural data space will produce only a small perturbation in the related temporal sequence of **p**.

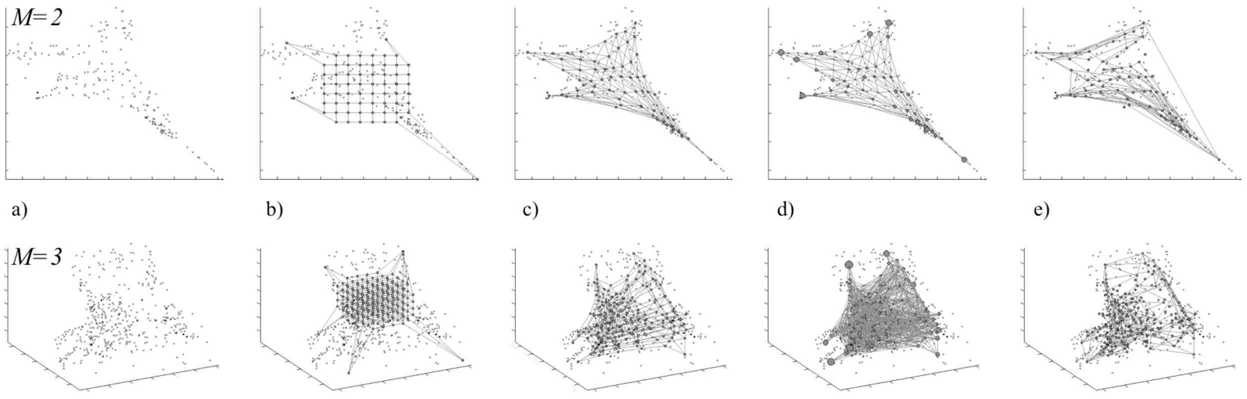

Figure 1. 2D and 3D lattice examples of an SOM trained with the proposed algorithm with different gestural data; a) gestural training data; b) weight initialization; c) trained lattice output weights; d) trained lattice output weights with mass mapped to node diameter; e) trained lattice with output node neighborhood based on gestural data temporal unfolding.

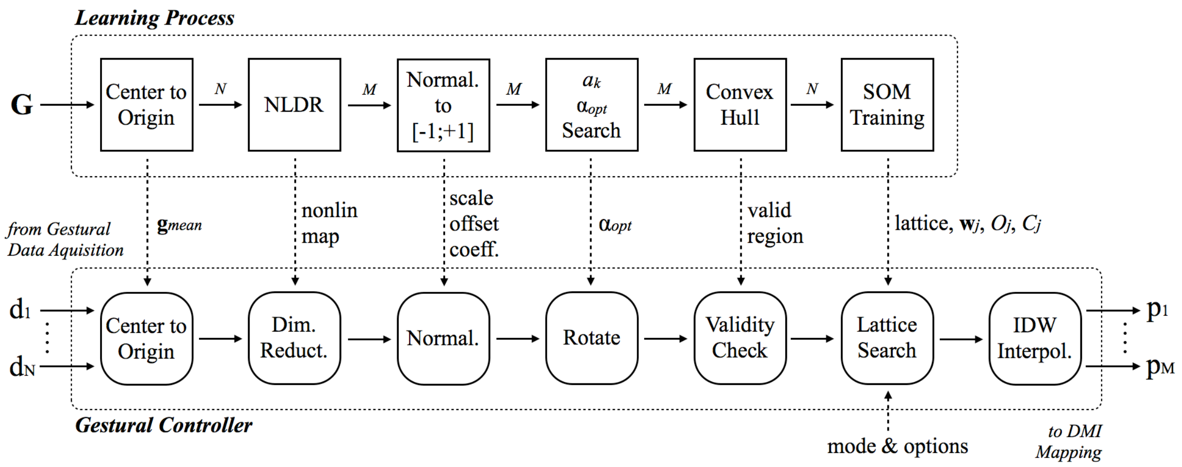

Figure 2. Learning Process and Gestural Controller functional schemes.

The SOM algorithm is supposed to organize the output node weights also according to the training data density. More dense areas will have more output nodes and vice versa, which ideally means equal C_j across all the output nodes. However in real cases some minor difference may still be present. We can consider the C_j as a mass value associated with O_j and replace the Euclidean distance with a "gravity force" in the neighborhood search as in (5)

$$F_j = \left(G_{const} \cdot C_j \Big/ \left| \mathbf{w}_i - \mathbf{d}^* \right|^2 \right) \qquad (5)$$

where G_{const} represents a user-defined gravitational constant, and we can omit the mass of $\mathbf{d}^*$ because it will just scale equally across all the F_j. With this approach, independently of the neighborhood search space definition, the winner O_j is the one giving the strongest gravitational attraction. The GC response perturbed by the mass of the output nodes is non-linear, presenting "hills" and "valleys" where a different amount of gestural energy is required to change position.

So far, we have used the SOM-based GC using the global gestural information learned from **G**. Therefore the system will also respond to gestures different from those used for training, but in terms of their multidimensional shape and density in **G**. At the end of the learning process we found the set of possible next nodes for each of the lattice output nodes, analyzing forward and backward the sequence of entries in **G**. We can use the possible next nodes associated to each O_j as a neighborhood search subspace. In this way, we obtain a different operational modality where the GC responds only to gestures consistent with those used in training. This modality introduces strong constraints between the GC input and output, and it is not used in our VCI system.

4. VCI INTEGRATION & EVALUATION

In this section we discuss the Self-Organizing Gestures user perspective. We briefly describe the integration with our VCI system, and we present some numerical results. The proposed method is completely unsupervised and allows the user to obtain the GC transformation *f* by providing only some gestural examples. Because the method is unsupervised, the implication is that users have to learn the resulting gestural organization in relation to the GC output. We provide two computationally inexpensive possibilities for allowing the user to modify the system response in real-time. The first is the range *[0,1]* inversion for every single GC output to *[1,0]* and the second is the application of a global scaling value to the $\mathbf{w}_j$, to expand or shrink the output lattice in relation to the GC input space, while maintaining the same topology.

We have implemented the learning process of Section 3.1 using a set of MATLAB functions with a simple interface. The gestural training data is read from a file, and imported into a single matrix. During the learning process it provides some intermediate text information and it shows graphically the SOM learning process plotting and updating the $\mathbf{w}_j$ grid over the **G** data after the dimensionality reduction. Plotting is supported for *M* less than *4*. The learning function returns a single structure including all the data necessary for the various operational modes described in 3.2. For the NLDR we use the Dimensionality Reduction Toolbox[1]. The real-time GC operative modes are implemented in a separate MATLAB function, which exchanges data though the Open Sound Control (OSC) protocol, receiving the gestural data **d** and sending out **p**. The real-time GC is able to change its behavior in runtime in response to specific OSC messages. It is possible to change the operative modality, invert the output ranges, and modify the global scale factor for the $\mathbf{w}_j$.

The integration with our VCI MAX/Msp system [6] is established through the OSC communication for the real time part, while the gestural training data is collected into a matrix in MAX/Msp and exported to a text file using FTM [19]. In the VCI prototype we implemented a larger vocal feature computation and a system for noisy features rejection based on vocal postures as described in [20]. A mixture of MFCC and PLP features represent the system's gestural data. Even though we tested the different operative mode described in 3.2, in the evaluation experiments we work using a search space limited to the Moore neighborhood of the previous output node, which is the modality that better suits voice driven GC.

The evaluation data presented here considers only the GC performances. We probe the measurement data before our DMI mapping system [21], which is integrated in the VCI. We ran two categories of tests using 10 different vocal gestural training data **G**. In the first category we verified the performances of the learning process. For each **G** we trained the two equivalent SOMs 20 times, one using our proposed method and the other one using the standard SOM method. We compared the two by measuring the frequency of global topology (twisting, folding, curling) and local topology distortions. These results are presented in Table 1, reporting the average over the 10 cases.

In the second category of test we gave the user 3 minutes to learn and fine-tune the SOM-based GC, and a second GC obtained with our previous VCI approach having equal dimensionality and output resolution The user was then asked to perform two tasks. The first was to cover the highest number of GC output combinations possible (each lattice output node represents one combination) within 1 minute; the other was to maintain the output at 2^M+1 key positions (vertices and center) for 5 seconds. We then measured the output stability in terms of standard deviation. The results are presented in Table 2, showing the average over the 10 cases. In these tests we refrained from mapping the GC output to any DMI in order to focus attention on the output signal itself that will be used for mapping. The presence of sonic feedback can ease some of the tasks thereby biasing the measurements. Performance might also have been influenced by the nonlinearities in the DMI parameters-to-sound response. Thus the only feedback provided in the user testing were the *M* on-screen continuous sliders and an *M*-dimensional grid. For these experiments, the **G** dimensionality *N* varies case by case between *7* and *28*, while the *N'* is usually either *2* or *3*, typical for vocal gestural data.

Table 1. Topology distortion performances comparison between standard and proposed learning process.

Test	Standard Training SOM	Modified Training SOM
% Global Topology Distortion	63.5%	2.5%
% Local Topology Distortion	33%	6%

Table 2. Use task comparison between the VCI GC and the Self-Organizing Gestures (SOG) based VCI GC.

Task	VCI GC	SOG-VCI GC
% Covered Output Combination	74.3%	96.5%
Key Positions Output Stability (standard deviation)	0.14	0.06

[1]http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html