A Voice Interface for Sound Generators: adaptive and automatic mapping of gestures to sound

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
Sound generators and synthesis engines expose a large set of parameters, allowing run-time timbre morphing and exploration of sonic space. However, control over these high-dimensional interfaces is constrained by the physical limitations of performers. In this paper we propose the exploitation of vocal gesture as an extension or alternative to traditional physical controllers. The approach uses dynamic aspects of vocal sound to control variations in the timbre of the synthesized sound. The mapping from vocal to synthesis parameters is automatically adapted to information extracted from vocal examples as well as to the relationship between parameters and timbre within the synthesizer. The mapping strategy aims to maximize the breadth of the explorable perceptual sonic space over a set of the synthesizer's real-valued parameters, indirectly driven by the voice-controlled interface.

Keywords
Voice Control, Adaptive Interface, Automatic Mapping, Timbre Morphing, Sonic Space Exploration.

1. INTRODUCTION
The growing computational power of general-purpose and digital signal processors has enabled a dramatic increase in the complexity of sound synthesis algorithms able to execute under continuous parametric control in real time. In electronic and experimental musical genres the exploitation of instrumental timbral manipulation is becoming more pervasive along side, or as an alternative to traditional note-based control. At the same time communication protocols such as MIDI and OSC are evolving to empower performers to exploit the growing musical potential of digital musical interfaces.

Two main limitations within the current state of affairs have become apparent. The first is a lack of design consideration in interfaces integration and cooperation. The issue of how a performer can make simultaneous use of multiple interfaces is rarely addressed. A single interface can demand all of a performer's interaction bandwidth, preventing the parallel use of another control device. Most of the commonly used musical interfaces exploit interaction via the hands (tactile, haptic, gestural). As a result, the number of events and parameters under the direct control of the user at any given time remains constrained despite the rich interfacing capabilities offered by control devices.

The second limitation is common in the design of general-purpose interfaces. Although they have the benefit of flexibility and reusability, they are not tailored to work with specific instruments. Extensive manual intervention is necessary to define and implement mappings for a particular pairing of interface and instrument. The relationship between sensor signals and synthesis parameters is often programmed as a one-to-one mapping or as a one-to-many mapping. The first case puts a limitation in timbre morphing potential, due to the limited number of sensors simultaneously controllable by the performer, while the second case inherently reduces the explorable sonic spaces. The many-to-many relationship [1] can lead to more appealing mappings, but it presents challenges in their manual definition, especially if the dimensionality is high and if continuity and differentiability are required [2]. Capturing and exploiting dependencies, correlations and nonlinearities between control parameters and sound is a non-trivial balancing act between these trade-offs.

The work described in this paper addresses these limitations proposing the use of vocal gesture to control time-continuous and real-valued sound synthesis parameters, an automatically generated many-to-many mapping, and the adaptation to the relationship between synthesis parameter and perceptual sound features. In the authors' pragmatic approach, the human voice is chosen as a source of gesture because it can be considered as "spare bandwidth" [3] for performers engaged with instrument interfaces, especially in the field of electronic music. Moreover the simultaneous use of voice along with other body gestures is a natural human capability, and has the benefit of providing and additional layer of control without any hardware dependencies other than a microphone. From this perspective, the unused resources of the voice present a tremendous opportunity. The mapping strategy developed herein creates a relationship between vocal gestures and perceptually related features computed on the synthesized sound without the explicit introduction of perceptual intermediate layers [4].

2. RELATED WORK
The Singing Tree [5], part of the Brain Opera installation, was one of the first systems which extracted a large set of vocal features to drive a set of Digital Musical Instruments (DMIs), re-synthesizing the human voice with the sound of an ensemble. Despite certain limitations such as its fixed mapping, based on prior knowledge about vocal gesture and instruments, it demonstrates the potential of voice for interaction with DMIs.

The Gesture Follower [6] and the Wekinator [7] offer solutions to map generic gesture to synthesis parameters. Both flexibly define the mappings, and they specifically address the continuous generation and modulation of an output signal. However, these systems are not specifically designed to work with a vocal input signals and do not consider the effect that the generated parameters have on the output sound of the DMI.

NIME'12, May 21-23, 2011, University of Michigan, Ann Arbor.

Manual intervention to define the mapping with the DMI still plays a central role and is often a challenging task. The Gesture Follower is based on a left to right Hidden Markov Model, while the continuous parameter component of the Wekinator, is based on neural networks. The first presents causality constraints, the second requires manual intervention to generate a set of input-output pairs for the training phase, which may need to be very large in case of nonlinear mappings.

The Singing-Driven Interfaces for Sound Synthesizer [8] address the use of voice to control sound synthesis. The author proposes a system based on the imitation of the instrument's sound by the user, performing a temporal segmentation of the vocal signals based on syllables, and transforming voice-specific features into instrument features. The system generates a score and one continuous parameter for minor timbre modulation. The main limitation of this approach is a missing deep control of the instrument timbre, which is considered nearly fixed once a target instrument is selected.

In Making Music Through Real-Time Voice Timbre Analysis [9], two approaches are presented. In the first one, discrete event-based, vocal features are used for drum synthesis control by vocal beatboxing. In the second one, vocal timbral features are continuously "remapped" into the synthesizer timbral features space through a linear piecewise transformation of the principal components or with a regression tree transformation. In both cases the nearest neighbor identifies the unique correspondence between the two spaces. The choice of identical timbral features spaces for the voice and the synthesizer simplifies the system but is also a limitation. Furthermore, the rigid selection of features, especially on the voice timbre space, does not help to cope with the intra speaker variability.

3. AUTOMATIC ADAPTIVE MAPPING

The fundamental goal of this work is to dynamically modify the synthesis timbre using dynamics in the vocal sound. Since the voice-controlled interface we develop creates a many-to-many mapping relation, the dimensionality of the spaces with dependencies, correlations, and nonlinearities, can be challenging to human understanding. The automatic mapping generation and adaptation not only relieves the user from a tedious task, but it is the only practical way to proceed. The adaptation is based on deterministic and statistical information learned from an analysis of the target DMI behavior and of the user vocal gestures. With this approach we aim to maximize the coverage of the DMI's perceptual feature space given the range of vocal gestures. Figure 1 shows a functional scheme of the proposed approach. Users specify their intention by providing vocal examples and identifying the target control parameters of the DMI. The system automatically analyzes invariant and dynamic features of the voice, finds a relationship between the DMI's control parameters and the perceptual features of the resulting sound, and generates the mapping. Thereafter during run-time, the DMI parameters are driven by the live vocal input. We assume that the control parameter-to-perceptual features is deterministic and one-to-one, so that the sound descriptors generated by the mapping can be converted back to DMI parameters. If the relationship is not one-to-one, we adopt a strategy to minimize potentially noisy output that could result from the non-invertability as described in Section 3.3.

3.1 DMI parameters-to-sound analysis

We consider the DMI as a black box and use no prior knowledge about the synthesis model. To retrieve the necessary information we explore systematically of the input parameters in relation to the output sound. Equations (1-3) relate the DMI target time-continuous real-valued parameters C_{Tx} and the set of their unique combinations I_T.

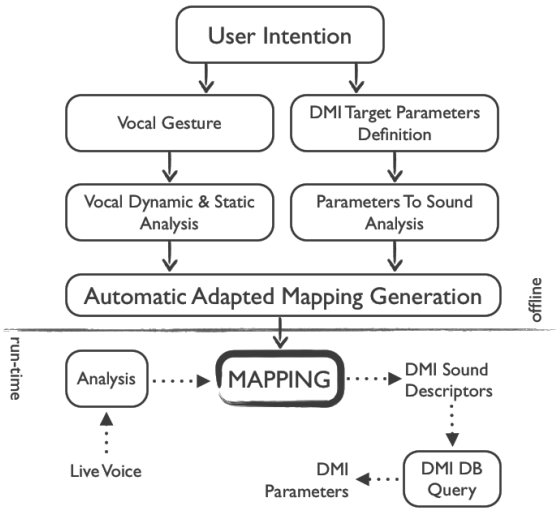

Figure 1: High abstraction functional scheme.

$$I_T = \left\{C_{T0}, C_{T1}, \ldots, C_{Tj}\right\} \text{ with } \left|I_T\right| = \left|C_{T0}\right| \cdot \left|C_{T1}\right| \ldots \cdot \left|C_{Tj}\right| = z \quad (1)$$

$$\left|C_{Tx}\right| = (\max(C_{Tx}) - \min(C_{Tx})) \,/\, \mathrm{res}(C_{Tx}) \quad (2)$$

$$\mathbf{i}_{T,w} = \left\{c_{T0,w}, c_{T1,w}, \ldots, c_{Tj,w}\right\} \qquad \text{with } w \in [0, z] \quad (3)$$

In equation (1) I_T represents the set of j unique DMI target parameter combinations. Its cardinality is given by the product of the cardinality of each C_{Tx} in (2). Once j parameters and are selected, together with the respective maximum, minimum and resolution (step value) there are z unique combinations of the target parameter vector **i** (3). For the z vectors **i** the DMI sound is analyzed, to create a vector of perceptual features **d**. The synthesizer is hence projecting a control parameter vector $\mathbf{i}_T$ with semantic meaning (e.g. cutoff frequency, modulation depth) into a perceptual feature vector **d**, usually with higher dimensionality. Vectors **d** are stored in matrix **D**, which is our representation of the DMI behavior. The vectors $\mathbf{i}_T$ are stored in matrix **I**, hence through indexing it is possible to find the unique correspondence between $\mathbf{d}_i$ and $\mathbf{i}_{T,i}$. The audio descriptors stored in **d** capture relevant features of the DMI timbre in the perceptual feature space. According to [10] timbre perception can be reduced to a three dimensional space. This reduction could be accepted in sound recognition or classification domains, but here a prior blind dimensionality reduction can lead to losses in understanding the parameters-to-sound relation. We apply a posterior dimensionality reduction through PCA (Principal Component Analysis) is applied to the entire matrix **D**. We retain a number of dimensions, called PC (Principal Components), carrying 90% of the total energy.

3.2 Vocal gesture analysis

The abstraction level of the interface is close to the timbre of the voice with the low-level features computed from the vocal gesture examples. To avoid the introduction of constraints and to allow users to explore a variety of different vocal gestures, we compute a large feature set, assuming no prior knowledge about which subset will be the most representative set in terms of variation and robustness, leaving this selection to a posterior stage. The system expects the user to provide examples of invariant and dynamic vocal sound. Although the system has no minimum requirements, robustness and reliability increase with multiple examples. We compute the features over 40ms windows with 75% overlap and we obtain an arbitrary number

of matrices $\mathbf{V}_{S,k}$ containing vectors from invariant examples and matrices $\mathbf{V}_{D,p}$ containing vectors from examples with vocal sound variation. The following steps are applied to obtain a matrix **V** as a compact representation of the vocal gesture:

1. Analysis within the $\mathbf{V}_{S,k}$ to discard noisy features, minimizing the distance from cluster centers.
2. Analysis across the $\mathbf{V}_{D,p}$ to discard inconsistent features, using a normalized standard deviation threshold.
3. Merging of all vectors **v** into a single matrix **V**.
4. Detection and removal of outlier vectors.
5. Reduction of cluster density by vector removal.
6. Normalization of each feature to zero mean and unitary standard deviation.
7. PCA to reduce dimensionality.

The number of clusters is one of the user-provided parameters, equivalent to the number of invariant sounds used in the examples. In step 4, a vector is considered an outlier and discarded only when two conditions are met: the distance from its cluster center is larger than a threshold distance, and the distance from all others cluster centers is bigger than the inter-cluster center distance. At the end of this process the matrix **V** contains the principal components carrying 90% of the total energy. Independence (or at least uncorrelated as ensured by PCA) vocal features ensure "freedom of navigation" within the multidimensional sonic space **D**, without restricting the action to sub-regions or to specific trajectories.

3.3 Mapping strategy

The mapping is based on the triplet of matrices **D**, **I** and **V**. Its role is to apply a transformation *m* on a vector **v** that generates a vector **d** from which we retreive the corresponding parameter vector **i.** The main challenges are represented by the different number of elements, different dimensionality, and embedded information between **D** and **V**. The transformation *m* is based on computed statistical information since the number of elements in **D** and **V** is not uniform. The number of PCs considered for the mapping is set to the smallest between **D** and **V** after the truncation to 90% of total energy. If **D** is truncated, the vocal gesture is not articulated enough to fully explore the sonic space **D**. If **V** is truncated, the vocal gesture is over articuleted. Only the first case presents limitations in achieving full control over the DMI sonic space. We transform each PC of **V** to a PC of **D**, respecting their variance ranking. The transformation is based on the densities of every PC of **V** and **D**, which are estimated from the histograms. Then we integrate their normalized complements to obtain the monotonic mapping functions. Equations (4) and (5) show the transformation of each principal component of **v** into a principal component of **d**, maximizing the overlap between densities of each.

$$m_{d,v} = \int_{PC_i} hist^{COMP}(pc_i) \cdot dpc_i \tag{4}$$

$$pc_i(\mathbf{d}) = m_d^{-1}(m_v(pc_i(\mathbf{v}))) \tag{5}$$

Equation (5) describes the mapping function which is applied twice in (6) to obtain the principal component in **d** from those in **v**. Figure 2, presents an example of densities, distribution and mapping functions $m_{v,d}$ for the first principal component of the voice and DMI data. The mapping process executes the following steps: every principal component of a vector **v**, represented as a point along the x axis, is mapped to the y axis through the m_v function in the left figure. It is then mapped though the inverse m_d in the right figure down to a point on the x axis which represents the principal component of **d**.

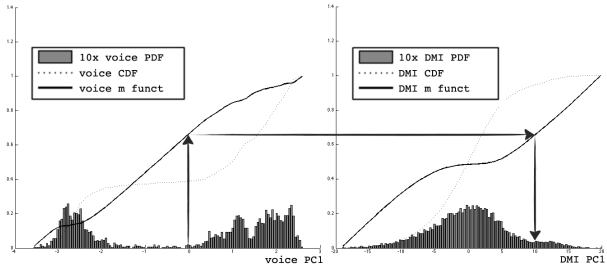

Figure 2: Examples of mapping functions, distributions, and densities of the first vocal principal component (left) and the first DMI principal component (right).

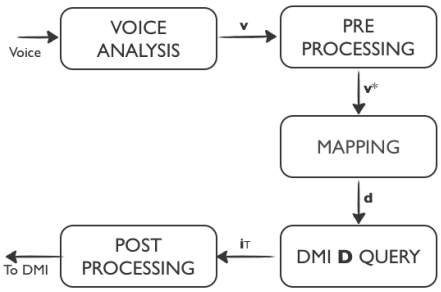

Figure 3: Run-time mapping data and processing flow.

The run-time mapping in Figure 3 shows the dataflow from voice to DMI parameters. The preprocessing stage, normalizes and projects the input to vocal vectors **v**. It also includes an optional gate operating on input energy level to inhibit output when the signal level is low. The projected vectors **v*** are mapped into a vector **d** as described above. The vector **d** is then used to find the *k* nearest neighbors (*k* set to 3 but user modifiable) in the matrix **D**, with the corresponding $\mathbf{i}_T$ vectors. Their average is used to drive the sound synthesis. Using only the nearest neighbor, as in [9], leads to discontinuity in the $\mathbf{i}_T$ stream if the parameter-to-sound relationship is not one-to-one.

4. PROTOTYPE & USAGE EXAMPLES

A proof of concept prototype[1] has been implemented in Max/MSP for the sections requiring real time computation, while MATLAB is used for the remaining offline parts. The Max/MSP prototype uses FTM [11], Gabor [12] and MnM [13] extensions from IRCAM, and it is integrated with Ableton Live using the Max For Live framework allowing interconnection with a state-of-the-art DAW. The Voice Analyzer patch computes feature vectors with a dimension of 50 including: energy, pitch, zero crossing rate, spectral flux, spectral centroid, spectral deviation, Mel spectrum centroid, Mel spectrum deviation, spectral flatness coefficients, LPC coefficients, MFC coefficients, formants frequencies and magnitude. The DMI Analyzer Front End and Back End patches cooperate to analyze the DMI behavior. Up to 6 parameters in any Live device can be chosen. For each parameters combination stored in a matrix, the second patch analyzes the audio signal of the DMI, generating a perceptual feature vector. These vectors, with dimension of 170 are stored in a second matrix, include: loudness, pitch, brightness, noisiness, sinusoidal components, the Bark scaled spectrum with 25 bins, and Mel scaled smoothed spectrum with 128 bins. The Run-time VCI patch

[1] Images of the Max For Live prototype and audio files available at http://anclab.org/downloads/fasciani_nime12.zip

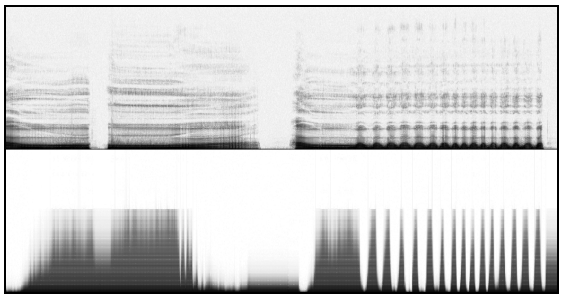

Figure 4: Spectrogram for the first example. Vocal gesture (top) and DMI output (bottom) showing how /a/ to /u/ gesture modulates the cutoff freq. The example on the left shows a slow variation /u/ to /a/ and /a/ to /u/. The example of the right shows quick and repeated /a/ to /u/ transitions.

implements the mapping described in Section 3.3. Although the system is designed to minimize user interaction, the prototype patches expose options and internal parameters allow us to perform user evaluations of different settings. Moreover if the user is not comfortable with the generated mapping, it is possible to modify it in run-time inverting the polarity of each principal component. To initialize the system it is required to identify the target parameters with their max, min and step values, and to provide the dynamic and invariant vocal examples. After this, the mapping is generated without further interaction. The sound-to-parameter analysis is the most time consuming part, proportional to the I_T cardinality (100ms per $\mathbf{i}_T$). The remaining parts, running non-optimized routines, requires about a minute on general-purpose machines.

In the first example to provide the basic proof-of-concept, the target device is a synthesizer generating a filtered square wave at 110Hz. The target DMI control parameter is the cutoff frequency of the 12dB Low Pass filter, within the range 110Hz-18.5KHz, equivalent to 0.19-1.0 when normalized within the unitary range. The scanning resolution is set to 0.007, leading to an I_T with cardinality 104. The phonemes /a/ and /u/ were the basis for the vocal gestures. Two invariant examples for each phoneme and two dynamic examples gliding between the two phonemes were provided to the systems. The automatic mapping process discovers that only a low number of dimensions is carrying the majority of the energy, so the system limits itself to work with a maximum of 5 principal components with the first 2 strongly dominating. The phoneme /a/ is mapped to descriptors corresponding to a high cutoff frequency, while /u/ corresponds to a low cutoff. Figure 4 shows the spectrogram of a dynamic vocal gesture and the DMI output, showing coherent, smooth and synchronous transitions between voice and DMI output sound. As a second more complex example, the DMI is the U-He Tyrell Nexus 6 synthesizer. Five parameters are chosen, with different ranges and a scanning resolution of 0.1. These are: pulse width depth, oscillators' shapes, the volume of first oscillator and cross modulation depth, which resulted in an I_T with cardinality 9900. The vocal gesture is a variation through the entire vowel space with 5 clusters, given by the phonemes /a/, /e/, /i/, /o/ and /u/. The number of components comprising the DMI sound descriptors is higher in this case, while for the voice only 5 components are considered, but with energy more evenly distributed among them in this case. In the audio examples file, a sequencer is executing a sequence of notes while the vocal signal drives the synthesis parameters. The parameters default values are selected when the input energy is low.

5. CONCLUSION AND FUTURE WORK

We have developed a system for mapping vocal signals to perceptual features for any synthesis algorithm. The prototypes show the validity of the approach. No end user-involvement is required beyond providing examples of control vocal gestures and the selection of DMI parameters to be controlled since the flow is completely automated. The user's time can instead be spent on vocally exploring and learning the mapping. One advantage of the system modularity is the reusability of the individual voice and DMI analysis data, further reducing the setup time. The three components of the system, described in Section 3, can be also exploited in interfaces of a non-musical or non-vocal nature. An issue that remains open arises when the DMI sound has non-static timbre for fixed parameters. The current audio descriptors here are sufficient to capture only static features. Another issue requiring further attention is a more elaborate feature rejection process to address noise affecting the vocal features, which sporadically gives instability to the system. We have also yet to address the presence of external noise at the input of the run-time system. Since much of the "noise" is generated by the musicians themselves, it may be possible to use knowledge of this signal to address this issue.

6. REFERENCES

[1] M. M. Wanderley, "Performer–Instrument Interaction: Applications to Gestural Control of Sound Synthesis," Ph.D. Thesis, University Paris, 2001.

[2] D. Van Nort, M. M. Wanderley, and P. Depalle, "On the choice of mappings based on geometric properties," in *Proc. of the NIME 2004 conference*.

[3] P. R. Cook, "Re-Designing Principles for Computer Music Controllers: a Case Study of SqueezeVox Maggie," in *Proc. of the NIME 2009 conference*.

[4] D. Arfib, J. M. Couturier, L. Kessous, and V. Verfaille, "Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces," *Org. Sound*, vol. 7, no. 2, pp. 127–144, Aug. 2002.

[5] W. Oliver, J. Yu, and E. Metois, "The Singing Tree: design of an interactive musical interface," in *Proc. of the 2nd conference on Designing interactive systems: processes, practices, methods, and techniques*, 1997.

[6] F. Bevilacqua and R. Muller, "A gesture follower for performing arts," 2005.

[7] R. A. Fiebrink, "Real-time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance," Ph.D. Thesis, Princeton, 2011.

[8] J. Janer, "Singing-driven Interfaces for Sound Synthesizers," PhD Thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[9] D. Stowell, "Making music through real-time voice timbre analysis: machine learning and timbral control," Ph.D. Thesis, Queen Mary University of London, 2010.

[10] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *Journal of the Acoustical Society of America*, vol. 61, no. 5, pp. 1270–1277, 1977.

[11] D. Schwarz, N. Schnell, R. Borghesi, F. Bevilacqua, R. Muller, "FTM: Complex Data Structure for Max," 2005.

[12] N. Schnell and D. Schwarz, "Gabor, multi-representation real-time analysis/synthesis," in *Proc. of the DAFx 2005 conference*.

[13] F. Bevilacqua, R. Müller, and N. Schnell, "MnM: a Max/MSP mapping toolbox," in *Proc. of the NIME 2005 conference*.