ADAPTING GENERAL PURPOSE INTERFACES TO SYNTHESIS ENGINES USING UNSUPERVISED DIMENSIONALITY REDUCTION TECHNIQUES AND INVERSE MAPPING FROM FEATURES TO PARAMETERS

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

In this paper we develop adaptive techniques for mapping generic user interfaces to synthesis engines. Upon selecting a subset of synthesis parameters, the system automatically finds the parameters-to-sound deterministic relationship in a multidimensional space. We analyze this sonic space using two different unsupervised dimensionality reduction techniques and we build the mapping using statistical information on a lower, but maximally representative, number of dimensions. The result is an adaptation of any general-purpose interface to a specific synthesis engine, providing control directly over the perceptual features with greatest variance. This approach guarantees a linear relationship between control signals and perceptual features, and at the same time, reduces the control space dimensionality maintaining the maximum explorability of the sonic space.

1. INTRODUCTION

Synthesis engines often expose a large set of parameters to the users. Runtime variation of the parameters produces modification in the sound generated as well as in its perception. The physical separation of control from synthesis has promoted the proliferation of a variety of generic control interfaces, enabling reusability of the same controller with different synthesizers and vice versa. Since controllers and synthesizers are not "co-designed" [1], some kind of manual intervention is generally required to establish the "mapping" between them.

In modern music genres the flexibility of the synthesis engine is widely exploited in such a way that notes and chords are looped or generated algorithmically rather than played with individual input gestures. The parameters that the musician modulates, usually represented by a real-valued numbers, result in timbral variation. This trend can be seen in a recurrent interface design pattern where sensors capable of capturing real-valued and time-continuous gesture are augmenting or replacing discrete ones. In addition, the evolution of the MIDI communication protocol and the introduction of OSC (Open Sound Control) are providing a more suitable communications infrastructure for this kind of control.

As synthesis networks become more complex, predicting or understanding the parameter-to-sound relationship becomes more challenging. This is especially true when multiple synthesis parameters are interfering with each other or have built-in dependencies, correlations and nonlinearities. Users build their understanding of the causal relation between synthesis parameters and their sonic effect through experimentation, practice, and heuristics strategies.

Within the majority of control devices and synthesis engines the routing of control signals to parameters is fairly basic: the range and occasionally the standard mapping functions (linear, exponential, logarithmic) are the only available options. Therefore, even assuming that it is possible to garner a heuristic understanding of the parameters-to-sound relationship, the desired mapping implementation might be impossible without the introduction of an intermediate processing layer.

In this work we propose a technique to adapt a general-purpose interface to a synthesis engine through:

- automatic analysis of the synthesis engine parameter-to-sound relationship based on perceptually related audio features;
- generation of an adaptive mapping based on the application of unsupervised dimensionality reduction techniques on the multidimensional perceptual sonic space.

Similar one-to-one, one-to-many or many-to-many mappings [2] have been developed through the introduction of an intermediate layer in the perceptual space [3]. Our work is focused on reducing the burden on the user who needs only to provide the system with information about the variable synthesis parameters. The dimensionality of the control space (number of independent signal from the control interface) can be set or modified a posteriori. Other than enabling direct control over perceptual aspects of the synthetized sound, which are user defined, this technique introduces two additional benefits:

- a linear relationship is created between the control signal and the variation in the generated sound, avoiding situations where the controllers' range leads to drastic sound variation or to null sound variation;
- an optimal mapping is created from a control signal space C, with dimensionality c, to

synthesis engine control parameters space P, with dimensionality p, with c smaller than p.

The optimal mapping is defined as the one allowing the widest a sonic exploration when projecting the control space C into the perceptual features in the sonic spaces D, directly related to the synthesizer parameter space P. The number of concurrent signals controllable through human gesture is constrained by human cognitive and physical limitations. The consequence is that the of dimensionality of C is generally much smaller than P. Figure 1 shows a generic control interface driving a synthesis engine through the perceptual sonic space. In our approach this space is retrieved automatically and analysed with unsupervised dimensionality reduction techniques in order to compute the adapted mapping between control interface and synthesized sound.

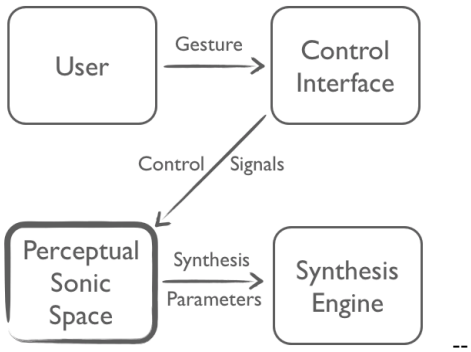

Figure 1: Synthesis engine control through the perceptually related sonic space.

1.1. Related Work

Synthesis algorithms usually involve large number of parameters. Wessel [4] showed that it is possible to describe timbre spaces with fewer degrees of freedom than those exposed by most synthesis algorithms. This principle is exploited in several works where the sound synthesis or, in general, the sound generation is driven by musically or perceptually meaningful audio descriptors. In [5] the timbre of a synthetic sound is driven by an audio stream produced by a live musician. A low dimensional vector computed in real-time from the input audio stream and representing the user intention, is transformed into a vector representing the synthetic timbre. The timbre vector is then converted into synthesis parameter based on a prior analysis and multidimensional scaling of the synthesizer output sound behaviour.

CataRT [6] is a concatenative sound synthesis system where grains are played from a large corpus of segmented and descriptor analyzed sounds and it can be seen as extension to granular synthesis. Depending on the corpus, CataRT is potentially able to generate a wide range of timbres. To facilitate the user navigation across the different sounds, the control is implemented in a low dimensional sound descriptors space.

Granular synthesis, for its high dimensional control space and the variety of generated sound, is also used to demonstrate the benefit of the Modulation Matrix [7]. The authors' approach, although substantially different from the one proposed here, provides a similar usability solution: an expressive control over the instrument in a lower dimensional space. The modulation matrix defines flexible interrelations between modulation sources and synthesis parameters, and allows modulation feedback. The entire matrix is dynamically altered through interpolation when the performer navigates gesture space.

2. SYNTHESIS ENGINE PARAMETERS-TO-SOUND ANALYSIS

Within this context we define a synthesis engine as any chain of algorithmic processes that produce audio. We consider this chain of processes as a black box that converts vectors **p** of synthesis parameters into sound. Moreover we assume a deterministic behaviour, excluding the presence of any stochastic component within the chain. Hence it is possible to state that given a vector **p**, there is one and only one associated sound generated by the synthesis engine. The opposite of this statement may not be always true since, depending on the synthesis engine, different control **p** may lead to identical or very similar sounds. This will be taken into consideration in the adaptive mapping strategy in Section 3 to avoid potential noisy or discontinuous output.

The set of unique combinations of synthesis parameters p_{se} is defined upon selecting the variable parameters, their respective maximum value, minimum value and sampling resolution. Here we assume that each parameter is in the range [0,1] (if not a simple scaling operation is applied). Choosing j parameters p_k the cardinality of p_{se} is given by the equations below.

$$|p_{se}| = \prod_{k=1}^{j} |p_k| \tag{1}$$

$$|p_i| = \frac{\max(p_i) - \min(p_i)}{\text{resolution}(p_i)} \tag{2}$$

$$p_{se} \subset \left[\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_{|p_{se}|}\right] = \mathbf{P} \tag{3}$$

Equation (1) shows the cardinality of p_{se} computed through the product of the cardinality of each p_k, which in turn depends on the individual maximum, minimum, and sampling resolution (2). The set p_{se} can be represented with a matrix **P**, where each column is a vector $\mathbf{p}_i$ (unique combination of synthesis parameters). The cardinality of p_{se} and the size of **P** grow exponentially with the number of parameters j, and linearly with the sampling resolution values. The selection of these values determines a trade-off between the level of detail in the synthesis engine analysis and the size of **P**. The size of this matrix affects not only memory but computational load as well, as discussed in Section 3.

The sound is generated for each $\mathbf{p}_i$ and analyzed to produce a corresponding vector $\mathbf{d}_i$, using a fixed note on

the chromatic scale. For each unique combination of synthesis parameters we compute not one but a sequence of vectors containing perceptually related features. For timbre that is static over time, the $\mathbf{d}_i$ corresponding to the fixed $\mathbf{p}_i$, is set to the mean of the sequence of computed feature vectors. This help to minimize the noise caused by the random position of the analysis window in relation to the generated sound. For dynamic timbres, such as those due, for example, to the presence of low frequency oscillations (LFOs) in the synthesis algorithm, the sequence of feature vectors is used also to capture extra information about the dynamic aspect of the sound. The vector $\mathbf{d}_i$ corresponding to the fixed $\boldsymbol{p}_i$ is set to the mean of the sequence of vectors, adding and an extra scalar, which represents the timbre periodicity. Autocorrelation is used to compute the periodicity of each computed feature. If different periods are detected, their mean is used instead. The size and number of the analysis windows define the minimum detectable periodicity, while window overlap affects the maximum. For a better characterization of the dynamic aspect, the vector $\mathbf{d}_i$ can be further extended adding a periodicity value and oscillation range for each perceptually related feature, tripling its size.

Vectors $\mathbf{d}_i$ are stored in a matrix **D** and together with **P** fully characterize the parameters-to-sound relationship of the synthesis engine in the perceptual sonic space. Through column indexing it is possible to associate the $|p_{se}|$ unique combinations of synthesis parameters with the relative perceptual features vector and vice versa. The adaptive mapping is based on the information embedded in these two matrices.

3. ADAPTIVE MAPPING

Here we assume that the general-purpose control interface generates a set of independent signals within the range [0,1] and with uniform distribution. Therefore the space C, with dimensionality c, can be approximated with a hypercube. To obtain an adaptive mapping we further analyze the matrix **D** to generate another hypercube in a projected perceptually related parameter space, then the mapping is simply obtained by linking the two hypercubes. The number of control signals does not affect the **D** post-processing stages to define the mapping; its posterior definition simply restricts the navigation in the sonic space to a certain number of dimensions. However this method guarantees that even with a limited control space dimensionality c, the perceptual feature space is explored along the directions corresponding to the maximum variability. This may not always correspond to the maximum variability in the pure human perception. We describe and apply two different unsupervised dimensionality reduction techniques: PCA (Principal Component Analysis) and ISOMAP, both followed by a statistical analysis from which the mapping is derived.

3.1. Principal Component Approach

PCA is an unsupervised technique that uses an orthogonal transformation to convert a set of multivariate observations of potentially correlated variables into a set of uncorrelated variables called Principal Components (PCs). Since the matrix **D** can have high dimensionality, we apply a stage of PCA to project the data into a lower dimensional space. The multivariate data in **D** is subjected to a prior whitening which scales each dimension to zero mean and unitary variance. The orthogonal and uncorrelated set of PCs is ranked by variance, representing the quantity of information carried by each. Mapping the hypercube C on to the PCs of $\mathbf{D}_{PC}$, guarantees control within the subspace where the perceptual features change the most.

Compared to other works, the number of perceptually related features can be relatively high here. It is not necessary to have prior knowledge about variations of features with synthesis engine parameter alteration. Perceptually meaningful features that are constant are automatically discarded. However, the user can compose and weight individual features in order to customize the adaptive result if desired. In this way it is possible to obtain a control focused on specific perceptual features that are not necessarily the dominants in terms of absolute variability.

To provide a response that is as linear as possible, it is necessary to analyze the data across the PCs. For each dimension the density is estimated through a histogram with a number of bins proportional to the product of the inverse of the sampling resolutions. Since PC ranges with low density should be explored with a finer step compared to those with high density we use the complement of the histogram, represented in (5) $hist^{COMP}$. For each PC the mapping function is based on its normalized integral, implemented through the cumulative sum in the discrete domain. Two examples of c_i (vertical axis) mapping over the PC_i (horizontal axis) are showed in Figure 2, where the black continuous monotonic line represents the mapping function. Equation (4) shows how the control signal c_i is transformed into a PC_i value through the inverse of the mapping function m_i (5).

$$PC_i = m^{-1}(c_i) \tag{4}$$

$$m_i(pc_i) = \int_{PC_i} hist^{COMP}(pc_i) \cdot dpc_i \tag{5}$$

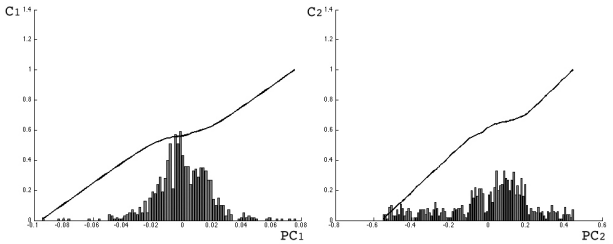

Figure 2: An example of histograms (scaled 10x) and mapping functions m_i (solid line) for the first two PC_i.

The interface signals c_i are used to generate a value of the first c PC_i of $\mathbf{D}_{PC}$ with linear interpolation, obtaining a $\mathbf{d}_i$ in the principal components space. The number of components considered in the system is limited to the number carrying 90% of the total energy. If c is smaller than the number of PC_i, the control signal mapped on the lower rank component is optionally mapped at the same time also on all the remaining ones.

3.2. ISOMAP Approach

ISOMAP is a low-dimensional embedding method [8], where geodesic distances on a weighted graph are incorporated with classical scaling. It is exploited to compute a quasi-isometric, low-dimensional embedding of a set of high dimensional data points. At the same time this algorithm provides a simple method for estimating the intrinsic geometry of a data manifold. The main difference with other multi dimensional scaling methods is in the choice of the geodesic distance metric, rather than the Euclidean one. In ISOMAP, the geodesic distance is the sum of edge weights along the shortest path between two nodes, computed using Dijkstra's algorithm. The top n eigenvectors of the geodesic distance matrix represent the coordinates in the new n-dimensional Euclidean space. ISOMAP implements a transformation of the space, while PCA projects the data into a new coordinates system in the same space.
Dimensionality reduction with ISOMAP is applied to **D** with the same method described in the previous subsection for PCA. The mapping of the c control signals on the new coordinates system, named ISO_i, is based on an estimation of densities and distributions as before. ISOMAP has a higher computational cost compared to PCA, but it detects and exploits the embedded manifold, achieving a more effective dimensionality reduction. ISOMAP is preferred when the control space C has a very low dimensionality. This difference is evident when comparing the energy in each dimension or the residual variance. Figure 3 shows an example of an energy distribution, measured in terms of variance, across the PC_i and ISO_i for the same data set.

The number of vectors **d** in $\mathbf{D}_{ISO}$ can be lower than $\mathbf{D}_{PC}$ because the ISOMAP algorithm includes an outlier removal stage. To guarantee coherence, the number of elements in $\mathbf{D}_{PC}$ and **P** must be the same. Hence the vectors **p** relative to the outliers are removed from **P.**

3.3. Synthesis engine parameters retrieval

For both approaches, after the generation of the vector $\mathbf{d}_{PC}$ (or $\mathbf{d}_{ISO}$) we search the nearest neighbour vector in the matrix $\mathbf{D}_{PC}$ (or $\mathbf{D}_{ISO}$). Through column indexing we retrieve from **P** the vector **p** used to drive the synthesis engine instantaneously. This simple approach leads to potential discontinuity in the synthesis parameters generation, because different combinations of synthesis parameters that might be far apart in the control space may lead to identical or near points in the perceptually related feature space. To guarantee continuity we propose two solutions. In the first one we retrieve K NN (nearest neighbours) in $\mathbf{D}_{PC}$ rather than one and drive the synthesis engine with the mean of the K corresponding vectors **p**. In the second one, before searching for the nearest neighbour, we append **p** to the $\mathbf{d}_{PC}$ (or $\mathbf{d}_{ISO}$) and we append the matrix **P** to $\mathbf{D}_{PC}$ (or $\mathbf{D}_{ISO}$). The first solution shows a limitation when the K **p** are very far apart, while the second can be debatable because perceptual features and synthesis parameters are merged in the same multidimensional space, hence the search is performed in a heterogeneous space. However, these methods improve a shortcoming in [5] where occasionally the system gets trapped in local minima.

As mentioned before, the sampling resolutions affect the size of **P** and **D**. The computational load required for the K NN search is thus proportional to the size of the matrix and it affects the system minimum response time.

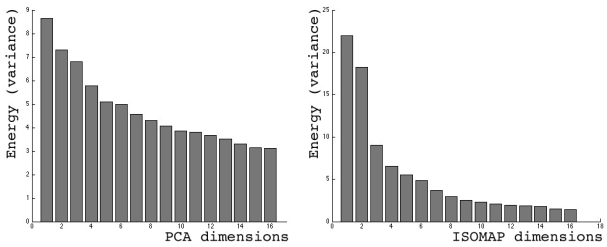

Figure 3: PCA (left) and ISOMAP (right) energy distribution across the reduced dimensions accounting for 90% of the total energy for the same dataset (note the different y axis scale).

4. PROTOTYPE AND APPLICATION

A prototype[1] has been developed and is implemented in Max/MSP and MATLAB. The prototype uses the FTM [9] and MnM [10] toolbox for vector and matrix processing in Max/MSP. The perceptually related feature set is based on Tristan Jehan's "analyzer~" (includes "fiddle~" by Miller Puckette) max external. The feature vector hence includes: loudness, pitch, brightness, noisiness, and the energies in the 25 Bark bands. Each feature can be enabled/disabled by the user and a weight vector can be defined as well to provide better customization. The adaptive approach is independent of the dimensionality and content of the feature vector, therefore a different selection is possible.

The prototype is integrated with Ableton Live using the Max For Live framework for the interfacing capabilities with state-of-the-art synthesis engines. Two Max For Live patches cooperate to analyze the synthesis engine. The front-end generates the $\mathbf{p}_i$ set and drive the synthesizer with up to 8 parameters, and the back-end analyses the audio signal, stores $\mathbf{p}_i$ and the relative multiple $\mathbf{d}_i$ in the matrices **P** and **D**. The post processing of **D** described in Section 2, and the adaptive mapping described in Section 3, are computed within MATLAB using the author's ISOMAP implementation[2]. Another

[1] Images of the Max For Live prototype patches are available at http://anclab.org/downloads/fasciani_icmc12.zip

[2] http://isomap.stanford.edu/

two Max For Live patches implement the runtime adaptive control for PCA and ISOMAP respectively, exposing up to 4 PC_i/ISO_i mapped control parameters.

Through the prototype's Max For Live patches it is possible to set and modify several system settings allowing exploration of different configurations. In the analysis patches it is possible to set the sampling resolutions, the parameters range, the note and the timing (in terms of delays) of the automatic analysis. Moreover, the number of analysis windows and the hop size are flexible, while the window size is fixed at 4096 samples. The control patches allow further reduction of the dimensionality of the PCA projection and ISOMAP transformation, modification of the K NN number, and the dimensionality of the control space C. The prototype allows also for inverting the polarity of every PC_i or ISO_i in order to flip the synthesis engine response.

4.1. Single Parameter Application

In this first application we chose a simple scenario to demonstrate the adaptation capability. The synthesis engine is the Ableton Live Operator synth, implementing a simple FM synthesis using just two oscillators. The only variable parameter is the cut-off frequency of the low pass filter. We run the analysis over the full range of the parameter and a reduced set of features, using the energy of Bark bands only. Four analysis windows per state $\mathbf{p}_i$ are computed with a hope size of 2048 samples, using C2 as fixed note. For the mapping we use only the principal dimension from the PCA and ISOMAP methods in order to have a 1D comparison metric. Both provide an identic result in terms of adapted control: most of the energy is concentrated on the first component since there is high correlation in **D**. Figure 4 shows three-dimensional scatter plot of the first three PC_i or ISO_i (note the different axis ranges), where is possible to appreciate the capability of ISOMAP to detect the manifold and organize the data almost on a line. Figure 5 illustrates how the adapted control provides a linear response over the feature with the greatest variance, while the control signal applied directly to the cut-off frequency present a non-linear response and a range with almost no effect over the generated sound.

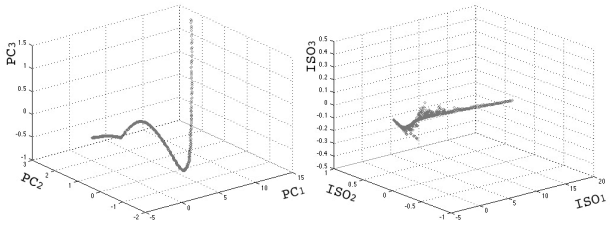

Figure 4: 3D scatters of the lower dimensional perceptually related features after PCA (left) and ISOMAP (right).

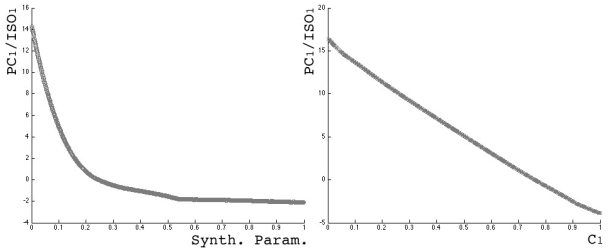

Figure 5: The synthesis engine parameter (left) and the adapted control (right) versus the principal feature.

4.2. Two Parameters Application

In a second example, we run the analysis computing the complete features set on a preset of the Ableton Live Analog synth, modifying the two "oscillator detune" parameters with a coarse sampling resolution. Ten analysis windows per state $\mathbf{p}_i$ are computed with a hope size of 1024 samples, using C3 as fixed note. Figure 6 shows how the two principal PCA and ISOMAP projected perceptual features are very noisy over the control parameter space, but in Figure 7 it is evident that these are linear and stable due to the adaptive control. The wider range obtained with the ISOMAP is due to its capacity to embed energy in a lower number of dimensions.

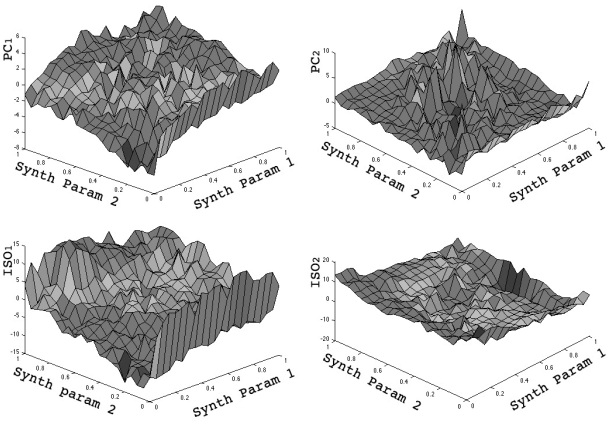

Figure 6: Two synthesis engine parameters versus: PC_1 (top left), PC_2 (top right), ISO_1 (bottom left), ISO_2 (bottom right).

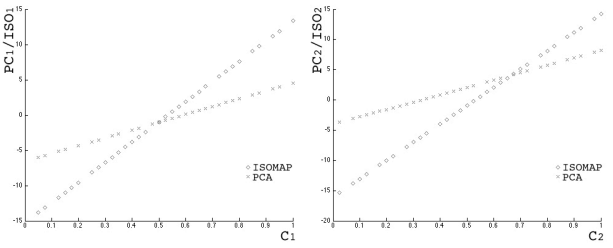

Figure 7: First adapted control versus the primary feature (left) and second adapted control versus the secondary feature (right) for PCA and ISOMAP adaptations.

4.3. Partikkel Hadron Application

In the last example we use the Partikkel Hardon[3] granular synthesizer with one of the provided pre-set. Through granular synthesis it is possible to obtain large timbre variation due to the nature of the synthesis, but often the control parameter set is large and challenging to design an interface for. This device exposes just 6 parameters for timbre manipulation thanks to the exploitation of the Modulation Matrix [7]. We analyze the generated audio with 25 Bark bands energies over the whole control space given by all possible combinations of the 6 parameters. Sixteen analysis windows per state $\mathbf{p}_i$ are computed with a hope size of 512 samples, using C2 as fixed note. In this more complex scenario the performance of ISOMAP is sensibly better than PCA. The data presented in Figure 8 shows how ISOMAP, when compared with PCA, allows the reduction of at least one dimension in the control space C without losses in the overall descriptors space energy With the ISOMAP adaptation, we obtain a further reduction of the control space. This enables the use of a simple 2D controller, while still permitting the navigation of the majority of the granular synthesis sonic space spanned by the original 6 parameters.

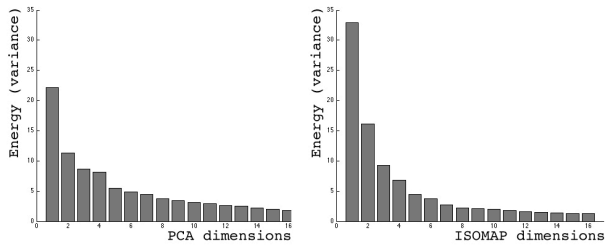

Figure 8: PCA (left) and ISOMAP (right) energy distribution across the reduced dimensions for the same Hadron granular synthesizer dataset.

5. CONCLUSION AND FUTURE WORK

We presented a generic method to adapt general-purpose interfaces to synthesis engines through unsupervised dimensionality reduction techniques and statistical analysis of the perceptually related features computed over the synthetic sound. The application of the prototype demonstrates the benefits introduced by this adaptive technique, including the linearization of the relationship controller-to-sound, and the dimensionality reduction of the control space. However some aspects can be further explored for improvements.

The exploitation of dynamic features in synthetic timbres must be explored more extensively. The computation of the dynamic aspect of the timbre has been tested, but embedding static and dynamic features in the same vector **d** may not be appropriate for all cases. Storing this information in two separate matrices and running dimensionality reduction separately on each may result in an adaptive mapping that is easier to use.

The current MATLAB implementation of the ISOMAP algorithm is computationally expensive in terms of time and memory, thus we had to limit the dimensionality of **D** and **P** to 4000, which is too small to handle large numbers of parameters sampled with a high resolution. This limitation of the resolution is reflected in the usability experience. An optimization of the algorithm implementation is thus desirable.

In Section 3 we make some assumptions about the control interface output signals. These are generally true for most of the commercial general-purpose interfaces (e.g. sets of sliders, knobs, touch surfaces, touch screen devices). For other interfaces built with large numbers of sensors, or devices capturing human gesture through image or sound, the assumptions may not hold. Through a statistical study of the interface signals it should be possible to apply a pre-processing stage that produces independent components within the desired range.

6. REFERENCES

[1] P. R. Cook, “Principles for designing computer music controllers,” in *Proceedings of the 2001 conference on New interfaces for musical expression*, 2001, pp. 1–4.

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

[3] 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.

[4] D. Wessel, “Timbre Space as a Musical Control Structure,” *Computer Music Journal*, vol. 3, no. 2, pp. 45–52, 1979.

[5] M. Puckette, “Low-dimensional parameter mapping using spectral envelopes,” in *Proceedings, International Computer Music Conference, Miami*, 2004.

[6] D. Schwarz, G. Beller, B. Verbrugghe, S. Britton, and others, “Real-time corpus-based concatenative synthesis with catart,” in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06),(Montreal, Quebec, Canada)*, 2006, pp. 279–282.

[7] U. Brandtsegg, S. Saue, and T. Johansen, “A modulation matrix for complex parameter sets,” in *Proceedings of the New Interfaces for Musical Expression Conference*, 2011.

[8] J. B. Tenenbaum, V. Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, no. 5500, p. 2319, 2000.

[9] D. Schwarz, N. Schnell, R. Borghesi, F. Bevilacqua, and R. Muller, “FTM — Complex Data Structure for Max,” 2005.

[10] F. Bevilacqua, R. Müller, and N. Schnell, “MnM: a Max/MSP mapping toolbox,” in *Proceedings of the 2005 conference on New interfaces for musical expression*, 2005, pp. 85–88.

[3] http:// http://www.partikkelaudio.com/