Augmenting Parametric Synthesis with Learned Timbral Controllers

Jeff Gregorio
Drexel University
3401 Market Street
Philadelphia, USA
jgregorio@drexel.edu

Youngmoo E. Kim
Drexel University
3401 Market Street
Philadelphia, USA
ykim@drexel.edu

ABSTRACT

Feature-based synthesis applies machine learning and signal processing methods to the development of alternative interfaces for controlling parametric synthesis algorithms. One approach, geared toward real-time control, uses low dimensional gestural controllers and learned mappings from control spaces to parameter spaces, making use of an intermediate latent timbre distribution, such that the control space affords a spatially-intuitive arrangement of sonic possibilities. Whereas many existing systems present alternatives to the traditional parametric interfaces, the proposed system explores ways in which feature-based synthesis can augment one-to-one parameter control, made possible by fully invertible mappings between control and parameter spaces.

Author Keywords

Feature-based synthesis

CCS Concepts

•Information systems → Music retrieval; •Applied computing → Sound and music computing; •Computing methodologies → *Neural networks;*

1. BACKGROUND

Being unconstrained by mechanics of physical vibration, many parametric synthesizers are able to offer musicians and sound designers much wider spaces of sonic possibility than acoustic and electroacoustic instruments. This flexibility comes at the cost of a relatively high barrier to entry, as operating the instrument with intention requires users to learn complex and often non-intuitive relationships between the parameters and the resulting sound. Parameters often have technical names relating to the underlying synthesis algorithm, and exhibit strong interdependencies and redundancies such that mastery requires a working knowledge of signal processing concepts and extensive ear training. Moreover, these requisite skills are unlikely to transfer from other instrument families.

A number of researchers have proposed alternatives to controlling parametric synthesizers in the traditional way, namely by developing mappings from low-dimensional controllers to parameters in such a way that places perceptually similar sounds in close proximity in the control space. These techniques often borrow from modern timbre analysis and machine learning methods used in Music Information Retrieval (MIR), though the potential for musical control based on timbral arrangement was perhaps first identified by David Wessel [16] in 1979. When named, this approach has been called feature-based synthesis [2] and perceptual sound synthesis [3]. We can interpret "control spaces" as including those based on audio input, as used in sound example matching systems [6][5][9], text-based semantic spaces [10], and gestural controllers [14][3]. The proposed work pertains primarily to the latter class interfaces.

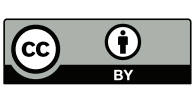
These control spaces have been proposed to simplify complex, process-based interaction, which undoubtedly benefits novices. However, care should be taken in such cases as to avoid trading flexibility and nuance for a lowered barrier to entry, in that such an interface is likely to lead to a quick exhaustion of possibility and lack of adoption by more experienced users [11]. Such bias of experienced musicans in favor of complexity and nuanced control over simplified mappings was observed by Jack et. al. [8] using a guitar-derivative digital musical instrument (DMI).

Despite the potential for over-simplification, user studies conducted by Tubb and Dixon[13][15] using a gestural mapping system (though non-timbral) have suggested a specific utility for expert users, namely in facilitating sound space exploration, which was characterized as an early, divergent stage of creation which is followed by a later, fine-tuning stage where the original parameter interface may be more appropriate. We believe this suggests that feature-based synthesis also has the potential to mitigate trade-offs, accommodating novices and experts alike.

It is worth noting that our current approach, as a starting point, is intended primarily to be of interest to researchers exploring the integration of gestural controllers with specific synthesizers and modules via timbre-based mappings, provided to end users as pre-trained models in a 'black box' system. In this respect, its motivation differs from the work of Stefano Fasciani [3], which considers use by creative practitioners as well as researchers, and more general approaches to the use of machine learning in tools for music creation, most notably the work of Rebecca Fiebrink [4], which has produced flexible and accessible toolkits for a much wider class of users.

2. MOTIVATION

In the proposed system, we detail an invertible control space to parameter space mapping system which allows the two spaces to be used in tandem, which retains the original flexibility of one-to-one mappings while adding capabilities for quick, gesture-based exploration and expressive modulation. The stated goal of augmenting, rather than supplanting traditional control is also intended to ground the



NIME'19, June 3-6, 2019, Federal University of Rio Grande do Sul, Porto Alegre, Brazil.

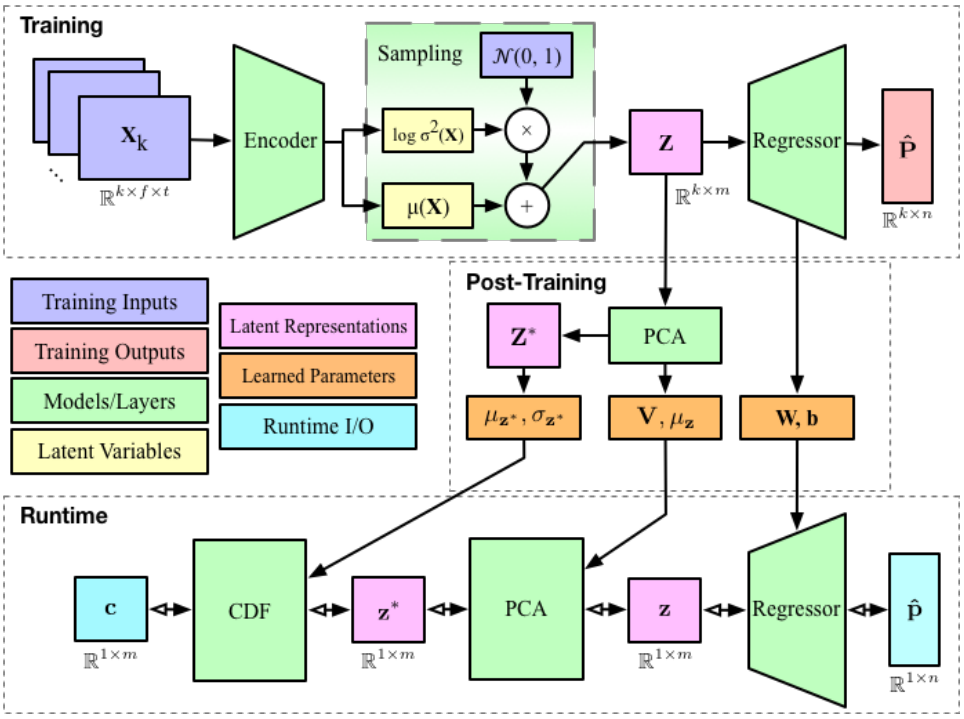


Figure 1: Overview of the system during training and runtime. Encoded training examples are used to predict known parameter values. The network minimizes the loss function given in equation 1 to obtain a normally-distributed latent encoding. Post-training, principal component analysis re-orients the latent space and parameters are exported for the bidirectional run-time model.

new interface in a familiar mode of interaction, and to task prospective users with building on existing skills rather than developing completely new ones. This serves as the guiding principle for many design choices in the proposed system.

The importance of this principle is apparent considering the mathematical intuition necessary to recognize and develop an accurate user mental model of a system built on low-dimensional projection of high-dimensional spaces. Such a mode of control is likely to be poorly understood by general users based on aural feedback alone. One strategy for conveying this underlying operation is the use of inferred parameter values for updating parameter sliders, which is trivial to implement in many software instruments. Doing so should make it visually apparent to a user exploring the control space that its dimensions do not neatly map to parameters in a one-to-one manner, as is the typically encountered use of multi-dimensional gestural controllers.

Just as important to establishing the equivalence of controller and parameter space, however may be the invertbility of control to parameter mapping, as such a mapping affords rapid switching between the two spaces and maintenance of a visual representation of the current control space location as parameter sliders are updated. In doing so, experienced users are able to see how familiar gestures and modulations relate to the novel controller, grounding any new understanding in their existing practice.

To our knowledge, the only existing system whose interface makes use of a fully invertible mapping is the aforementioned system proposed by Tubb and Dixon. However, this system uses a mapping of control space to parameters based on space-filling Hilbert curves, which uses no intermediate timbral representation. Its low-dimensional control space therefore, while more quickly navigable than a high-dimensional parameter space, is not likely to have a more perceptually-intuitive arrangement, and would not eliminate timbrally-redundant parameter settings, such as the myriad combinations of modulation parameters when have little to no effect when the modulation amount is close zero. The proposed system addresses these limitations using a straightforward, yet overlooked approach based on timbre spaces learned by neural networks.

3. PROPOSED SYSTEM

This work proposes a deep latent Gaussian architecture for obtaining invertible and computationally efficient mappings through a learned timbre space representation. At training time, an encoder projects a feature representation of sound examples into a parameterized latent space, which feeds a fully invertible regressor comprised of one or more dense layers, terminating in a continuous output layer whose activations are optimized to infer parameter values.

Following training, we export the model parameters indicated in Figure 3, which are loaded by a custom Max/MSP external which manages forward and inverse control to parameter space mapping. In this way, the encoder is allowed to be arbitrarily complex, and datasets to be arbitrarily large without increasing the computational complexity of the learned mapping. In demonstrations of the proposed system, we use a *Roli Lightpad Block M*[1] for control space coordinates, which include horizontal and vertical position, plus depth. Example patches and training/runtime source code are available on GitHub[2].

A note on notation

In this document we use P to denote an n-dimensional parameter space, C to denote an m-dimensional control space, and Z to denote an also m-dimensional latent space learned by the model. We use $\mathbf{p}$ to represent a length-n vector of

[1] `https://roli.com/products/blocks/lightpad-m`
[2] `https://github.com/JeffGregorio/TimbreMap`

parameter values originating from instrument's interface, and $\hat{\mathbf{p}}$ to represent parameter values inferred by the learned mapping. Similarly, $\mathbf{c}$ and $\hat{\mathbf{c}}$, as well as $\mathbf{z}$ and $\hat{\mathbf{z}}$ represent known and inferred control and latent coordinates, respectively. Finally, $\mathbf{X}$, $\mathbf{Z}$, $\mathbf{P}$, and $\hat{\mathbf{P}}$ represent matrices of training data, encodings, and known and inferred parameters.

4. DATA COLLECTION AND TRAINING

Here, we briefly describe a system for automatically generating datasets consisting of audio features $\mathbf{X}$ and ground-truth parameter values $\mathbf{P}$. The synthesizers used in the proposed work are implemented/hosted in Max/MSP. The synthesizer's parameter inputs are normalized to $[0, 127]$. Max/MSP's `js` objects are used to perform a grid search over n parameters with a given step size, or draw from a uniform random distribution of values. These `js` instances write their generated parameters to CSV files and control `sfrecord~` objects which write the examples to WAV using the sample rate specified in the Max/MSP audio settings.

Durations of examples are controlled by the `js` objects, and can be specified as a uniform length across all examples. Alternatively, custom synths or hosted VST/AudioUnit plugins whose amplitude envelope generators provide end-of-state signals can be used for variable example duration. For example, after the parameter generator sets parameters and sends sends a MIDI note on, it can be configured to wait for an end-of-decay message to send a MIDI note off, then wait for an end-of-release message to terminate the recording and write the example to WAV.

We have primarily tested the proposed system using Mel-scaled spectrogram images computed from each audio file, typically after downsampling to $11.025kHz$, using length-2048 FFTs computed with a hop size of 128. Mel-spectrograms are computed in Python scripts using librosa [1], and models are trained using the Keras deep learning API [3].

4.1 Encoding and Parameterization

In our preliminary experiments, we have evaluated encoders based on dense layers, long short term memory (LSTM) layers, and convolutional layers. Though we make no claims regarding the ideal encoder model in general, we've found that given our input representation, the LSTM-based encodings proved to be the best predictors for parameter accuracy, particularly concerning slowly-changing parameters, such as LFO rates, which may have cycle lengths exceeding the example length. We use a single LSTM layer with 128 units (one for each Mel-spectrogram bin), followed by a dropout layer, which feeds an L2-regularized dense layer of 64 units.

We use a latent Gaussian model inspired by variational autoencoders [12]. We assume the data to be generated by some Gaussian generative process $p(\mathbf{x}|\mathbf{z})$ with independent latent variables $\mathbf{z} \sim N(0, I)$, and model an estimated posterior $q(\mathbf{z}|\mathbf{x})$. We minimize a cost function consisting of a the estimated posterior's KL divergence from the assumed Gaussian prior, which penalizes latent encodings which deviate from the Gaussian distribution, plus a mean-squared prediction error term, which encourages encodings which yield predictive latent dimensions.

$$\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 - \frac{1}{2}\sum_{i=1}^{m}(1 + log(\sigma_i^2) - \mu_i^2 - \sigma_i^2) \qquad (1)$$

Local variational parameters μ_i^2 and $log(\sigma_i^2)$ are approximated by dense layers, and latent space encodings are produced using reparameterization.

[3] `https://keras.io/`

4.2 DNN Regression

Though dense neural network layers are not strictly invertible in all cases, and there is an active area of research on the subject of invertible neural networks, both the nature of our mapping problem and additional range constraints imposed by controller and parameter values lend well to preserving convertibility with minimal complexity. Therefore, it is appropriate to take the naïve view of neural network inversion, namely that we can learn a set of weights $\mathbf{W}$ and biases $\mathbf{b}$, and sample the latent space $\mathbf{z}$ to infer parameters as such

$$\hat{\mathbf{p}} = \sigma(\mathbf{z}\mathbf{W} + \mathbf{b}) \qquad (2)$$

and from known parameters $\mathbf{p}$, infer $\hat{\mathbf{z}}$

$$\hat{\mathbf{z}} = (\sigma^{-1}(\mathbf{p}) - \mathbf{b})\mathbf{W}^{-1} \qquad (3)$$

assuming $\mathbf{W}$ is invertible, as is the case when we have as many predictors as targets, or $m = n$. In the case where $m > n$, the inverse mapping is not guaranteed to be unique. However, the problem posed by feature-based synthesis essentially dictates that we have more targets than predictors as a condition for simplifying interactions using low-dimensional controllers. So in the typical case where $m < n$, we can replace $\mathbf{W}^{-1}$ with the pseudoinverse $\mathbf{W}^{+}$ to obtain the inverse mapping, provided activation σ is a linear or otherwise invertible function such as sigmoid, hyperbolic tangent, or leaky ReLU.

We use a single dense output layer with a sigmoidal activation function, followed by a lambda function which scales the normalized outputs of this layer to $[0, 127]$. Parameter inputs for the inverse mapping never exceed this range.

It's important to note that this mapping is invertible in the sense that we can send $\mathbf{c}$ through the mapping and back, and recover $\hat{\mathbf{c}} = \mathbf{c}$ with negligible error. However, the same is only true of $\mathbf{p}$ to the extent that the model can accurately infer parameter values from the latent encodings. This means there are locations in P which map onto C, but are unreachable from C alone.

4.3 Control to Latent Space Mapping

Real-time exploration in the encoder's learned latent space Z is accomplished by mapping a normalized control data vector $\mathbf{c}$ to $\mathbf{z}$. The control data can originate from any controller whose available degrees of freedom matches the dimensionality of the latent space, provided the output range of the controller is normalized to $(0, 1)$.

Though navigation in the latent space amounts to sampling from the joint distribution $p(\mathbf{z}, \mathbf{X})$, generative models are not strictly necessary. Whether generative or discriminative, we obtain the parameters of the mapping of $C \rightarrow Z$ by encoding a dataset $\mathbf{X}$ and computing statistics of its latent space representation.

The linear mapping, which uses the minimum coordinate and the range, is trivial, and while functional, this mapping assumes a uniform distribution in the latent space, which maps the same proportion of the control space to the latent space, regardless of the density of encoded examples in a given volume. Its also sensitive to outliers if no efforts are made to detect and remove these from the dataset.

Both of these problems are addressed by assuming that the latent space is normally distributed, and mapping $\mathbf{z}$ to the uniform control space $\mathbf{c}$ and vice versa using the normal cumulative distribution function (CDF)

$$\mathbf{c} = \mathbf{\Phi}(\mathbf{z}, \mu_\mathbf{z}, \sigma_\mathbf{z}) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{\mathbf{z} - \mu_\mathbf{z}}{\sigma_\mathbf{z}\sqrt{2}}\right)\right] \qquad (4)$$

and its inverse, the quantile function

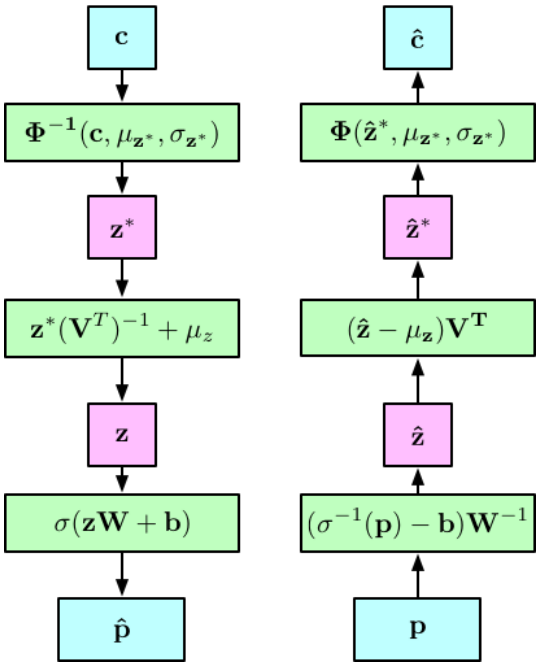


Figure 2: Left: forward runtime mapping $\mathbf{c} \rightarrow \hat{\mathbf{p}}$. Control space values are mapped onto the normally-distributed Z^* using the quantile function given in equation 5, followed by an inverse PCA transformation to the original latent space Z, and finally a dense layer with sigmoid activation function σ. Normalized parameter outputs are scaled to $[0, 127]$. Right: inverse mapping $\mathbf{p} \rightarrow \hat{\mathbf{c}}$

.

$$\mathbf{z} = \mathbf{\Phi^{-1}}(\mathbf{c}, \mu_{\mathbf{z}}, \sigma_{\mathbf{z}}) = \mu_{\mathbf{z}} + \sigma_{\mathbf{z}}\sqrt{2}\mathrm{erf}^{-1}(2\mathbf{c} - 1) \qquad (5)$$

respectively, under the condition that $c_i \in (0, 1)\ \forall i$, which is guaranteed by using normalized control space values.

4.3.1 PCA Transformation

Though the learned latent space is optimized by the end-to-end model for accurate prediction of parameters $\mathbf{p}$, the dimensions of Z are not guaranteed to align with the primary axes of variation in $\mathbf{Z}$. This would lead to poor utilization of controller area. In addition, the controller used to demonstrate this system is not equally sensitive in all its dimensions. The depth dimension in any 3D touch controller might be expected to be the most sensitive and least controllable. It is therefore preferable to map this dimension onto the axis of least variation in Z.

This is easily accomplished by performing a Principal Component Analysis (PCA) on $\mathbf{Z}$, yielding a matrix of orthogonal basis vectors $\mathbf{V}$, arranged in decreasing order of their corresponding eigenvalues. The last eigenvector, representing the flattest dimension, is used in mapping the depth dimension on the touch controller. The reoriented space Z^* is inserted between C and Z.

5. DISCUSSION

The proposed system can be trained on any synthesizer parameter with continuous value, which makes it quite difficult for user evaluations to disentangle the control model from the underlying synthesis method. Although we have not yet made such a systematic evaluation of this system, in this section we describe some of the models we have trained and offer some observations, however subjective, which may be of use in narrowing the field of synthesis methods offering promising applications of timbre-based gestural control. We also detail some caveats and assumptions that must be made in order to generate training data tractably.

5.1 Duration

Generation of training data should ideally make considerations for the durations of slowly-evolving signal components. Training on amplitude envelope parameters, for example, is contingent on the use of an end-of-release signal to truncate audio examples. Models can be trained on examples with widely varying length either by zero padding to the maximum length, or by training on batches which have been grouped by duration and standardized to a fixed length within batches.

Similarly, low modulation rates require some care. For example, learning parameters of a triangular LFO with variable duty cycle (varying the shape between descending ramp, through triangular, to ascending ramp), highlights one case of ambiguity at low modulation rates if example durations are fixed and relatively short. If we assume that the model's learned representation relates to the rate of change of such a modulator, then it becomes important for examples to extend across both the rising and falling portions of the modulator since, for example, a short window containing a relatively high slope might part of a symmetric, high-rate modulator or an asymmetric, low-rate modulator. In this case, an LFO end-of-cycle signal can control example times by sustaining the note until the LFO has completed an entire cycle.

5.2 Pitch

Our current models have been trained on datasets generated with notes of a single fixed pitch (middle C), under the assumption that pitch is largely independent from the timbral representation learned by the model. This somewhat unsafe assumption has seemed to work reasonably well in practice for controlling amplitude envelopes and LFO modulations of pulse width and filter cutoff frequency. Though it's possible that dependencies between pitch and timbral arrangement in the control space may be mitigated by generating multiple copies of the parameter grid, each used with a different note, in practice this would be highly dependent on the input features and encoder design.

Care should be taken in cases where a learned parameter has a strong interaction with pitch. Such is the case, for example, when learning a continuously-variable LFO rate used for amplitude or frequency modulations (AM/FM). While this limitation provides no indication for or against any subjective utility of a continuous AM/FM control space, it nonetheless cannot be assumed to preserve a timbral arrangement independent of pitch. Traditional 'FM synthesis' may be more appropriate for preserving timbral arrangement, where modulator frequencies are constrained to be harmonically-related to the carrier frequency. This leaves the model able to learn other parameters of modulator signals which would primarily affect timbre, such as amplitude, phase, and shape. This application of synthesis parameter learning has been explored by Horner [6] among others.

Similarly, other parameter modulations may exhibit strong dependencies on the parameter's base value and related parameters. For example, learning parameters of a modulation signal used for filter cutoff frequency modulation can assume a single fixed cutoff frequency center and quality factor when generating data, but deviations from those settings disrupt the timbral arrangement of the controller to varying degrees.

5.3 Dimensionality

Perhaps the greatest caveat to applying these methods to high-dimensional synthesizers is that possible combinations of parameter values increase exponentially as new parameters are added, so dataset sizes and training times increase drastically unless the grid resolution is reduced to a degree that may lead to undesirable sparsity in the latent space.

As noted in Section 4.2, the inverse mapping guarantees that every location in the parameter space maps onto the control space, but the same is not strictly true of the forward mapping from control to parameter space due to imperfect inference of parameter values from the learned encodings. This effect is particularly pronounced in higher-dimensional models, perhaps due to over-regulation of the latent space distribution and the underlying uni-modal assumption on the Gaussian prior $p(\mathbf{z})$. In these cases, we may consider developing latent space encodings that allow for multimodal distributions.

We have also found that models trained on sets of disparate parameters found in high-dimensional synths, especially those which affect pitch or include multiple types of modulation, tend to yield latent spaces that range quite widely, but at some cost to subtlety. Namely, such a wide latent space may support the divergent mode of creation detailed by Tubb and Dixon [15] or afford novice users quick location of sounds in a manner analogous to preset selection, it may be difficult to use for expressive modulation in keyboard or sequencer-driven performance without either further configurability of the control space, or thoughtful consideration of which parameter sets would be conducive to subtler timbre variation.

5.4 Configurability

The invertibility of the mapping layer makes it straightforward to explore the former path. Considering a user familiar with parametric synthesis, we might anticipate a mode of interaction consisting of finding familiar settings in P, which would project to a unique location in C. The user may then wish to modulate about that point, only confined to a much smaller sub-region of C, avoiding areas which differ drastically. This mode of interaction is currently supported by our runtime model (a Max/MSP external) using a 'center' toggle, which sets the center of the sub-region to the current location, and a 'scale' slider, which re-scales the bounds of the control space around the center (equally in all dimensions).

Similarly, two specific settings in P can define two vertices of a cube in a sub-region of C. Any such scaling within C is easily implemented with an additional linear mapping layer between C and Z^* which re-scales the normalized outputs of the controller to smaller volumes, using the current control space coordinate as a reference.

5.5 Choosing Parameters

While exploring the latter path may have a technical advantage of limiting timbral range and ensuring smooth latent space distributions, it may also be advantageous in that it leads to an encapsulation of function within specific modules, which fits well with our guiding principle of accessibility via building off existing skills. We might speculate that an experienced synthesist is accustomed to viewing a synthesizer as comprised of individual interacting modules (especially considering the recent re-popularization of modular and semi-modular synthesizers), and might prefer gestural control confined to a specific module as a way of integrating it with their existing systems.

As yet, we've primarily explored fairly standard additive and subtractive synthesis engines, where we've found the best modules for timbral control to be those with a number of closely-related parameters which yield subtle effects on timbre, without drastic affects on pitch. The additive example has eight harmonics and sub-harmonics configured in the manner of a 'clonewheel' organ, and has been the most stable and useful high-dimensional example, perhaps due to the lack of any significant interdependencies between its parameters.

The subtractive example models in the proposed system have been trained on modulation sources built on an a custom Attack Decay Sustain Release (ADSR) envelope with a shape parameter allowing each curve to range from exponential, through linear, to logarithmic. Another ADSR instance is used also as a low frequency oscillator (LFO), with its sustain level set to zero, and an end-of-decay signal used to retrigger attacks. The LFO offers rate and symmetry controls, provided by mapping these parameters to attack and decay times, as well as a shape control. The resulting LFO wave shape can therefore range continuously between triangular, ascending and descending sawtooth, and (nearly) square. We train models on this 3-parameter LFO, as well as 4- and 5-parameter versions which include amplitude and bias of the LFO waveform, which correspond to modulation amount and the base value of the parameter being modulated.

6. CONCLUSIONS AND FUTURE WORK

We have detailed the motivation, design, caveats, and first impressions of a system for augmenting one-to-one parametric synthesizer interfaces with timbre-based gestural control. Our future work will continue to explore a wider range of synthesis and effects engines, particularly considering the use of Max/MSP affords relatively easy integration with a wide array of VST and AudioUnit plugins.

Regarding the neural network model, it is possible that some modifications would obviate the post-training PCA step by generating a latent space oriented on its primary axes of variation in an end-to-end manner. This would further reduce the computational and memory requirements of the runtime model, though they are currently relatively lightweight. Further reduced complexity would be particularly advantageous if we were to consider the feasibility of signal-rate modulations through timbre space with external modulator signals.

Considering applicability to a limited number of parameter sets, we've noted (subjectively) that high-dimensional

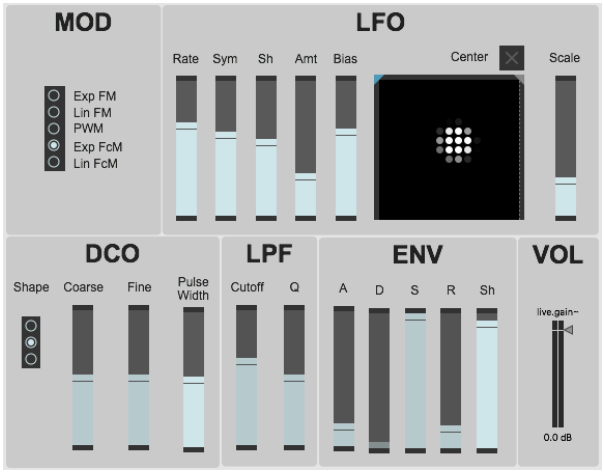


Figure 3: Example patch controlling 5-parameter LFO

models are best trained on sets of related parameters. Techniques like granular synthesis offer such an interface, in addition to an infinite space of audio sources to granulate, which can also be synthesized with parametric models. Similarly, parameterized excitation signals for physical models such as waveguides may offer a promising application.

We also have yet to consider applications that are not primarily keyboard or sequencer driven, in which it may be advantageous to explore integrating feature-based synthesis methods with higher level controller gestures, such as using strikes to trigger notes of a parametric percussion model, with the strike location determining the note's timbre.

We intend to use the proposed system to address questions regarding preferences for one-to-one and many-to-many mappings in parametric synthesis and how preferences are influenced by prior experience and evaluation task. Similar questions have been addressed by Hunt and Kirk [7] using hand-designed many-to-many mappings, yet never with a direct comparison between one-to-one and many-to-many mappings on the same controller. In doing a direct comparison between a learned 3-parameter mapping and an identity mapping, we may be able to indicate whether preference is driven by complexity of the mappings or the gestural nature of the controller itself.

We also intend to use the mapping invertibility to investigate the degree to which establishing visual equivalence of parameter and controls spaces influences the accuracy user mental models, which would have strong implications for the feasibility of adapting these techniques to commercially-produced synthesizers.

7. REFERENCES

[1] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and Music Signal Analysis in Python. In Kathryn Huff and James Bergstra, editors, *Proceedings of the 14th Python in Science Conference*, pages 18 – 24, 2015.

[2] M. D. Hoffman and P. Cook. Feature-based synthesis: A tool for evaluating, designing, and interacting with music ir systems. In *Transactions of the International Conference on Music Information Retrieval*, pages 361–362, 01 2006.

[3] S. Fasciani. Interactive computation of timbre spaces for sound synthesis control. In *Proceedings of the 2nd International Symposium on Sound and Interactivity*, pages 69–78, 2016.

[4] R. Fiebrink and B. Caramiaux. *Machine Learning and Listening in Composition and Performance*, chapter 12. Oxford University Press, 2018.

[5] S. L. Groux and P. F. Verschure. Perceptsynth: mapping perceptual musical features to sound synthesis parameters. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 125–128, March 2008.

[6] A. Horner, J. Beauchamp, and L. Haken. Machine tongues xvi: Genetic algorithms and their application to fm matching synthesis. *Computer Music Journal*, 17(4):17–29, 1993.

[7] A. Hunt, M. M. Wanderley, and M. Paradis. The importance of parameter mapping in electronic instrument design. *Journal of New Music Research*, 32(4):429–440, 2003.

[8] R. H. Jack, J. Harrison, F. Morreale, and A. P. McPherson. Democratising dmis: the relationship of expertise and control intimacy. In T. M. Luke Dahl, Douglas Bowman, editor, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 184–189, Blacksburg, Virginia, USA, June 2018. Virginia Tech.

[9] J. Janer. Voice-controlled plucked bass guitar through two synthesis techniques. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 132–135, Vancouver, BC, Canada, 2005.

[10] C. G. Johnson and A. Gounaropoulos. Timbre interfaces using adjectives and adverbs. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 101–102, Paris, France, 2006.

[11] S. Jordà. Digital instruments and players: Part i – efficiency and apprenticeship. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 59–63, Hamamatsu, Japan, 2004.

[12] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[13] R. Tubb and S. Dixon. The divergent interface: Supporting creative exploration of parameter spaces. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 227–232, London, United Kingdom, 2014. Goldsmiths, University of London.

[14] R. Tubb and S. Dixon. A zoomable mapping of a musical parameter space using hilbert curves. *Computer Music Journal*, 38:23–33, 2014.

[15] R. H. Tubb. *Creativity, Exploration and Control in Musical Parameter Spaces*. PhD thesis, Queen Mary University of London, School of Electronic Engineering and Computer Science, 2016.

[16] D. L. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, 3(2):45–52, 1979.