Active learning of intuitive control knobs for synthesizers using gaussian processes

Active Learning of Intuitive Control Knobs for Synthesizers Using Gaussian Processes

Cheng-Zhi Anna Huang
Harvard University
czhuang@fas.harvard.edu

David Duvenaud
University of Cambridge
dkd23@cam.ac.uk

Kenneth C. Arnold
Harvard University
kcarnold@seas.harvard.edu

Brenton Partridge
Harvard University
bapartridge@seas.harvard.edu

Josiah W. Oberholtzer
Harvard University
josiah.oberholtzer@gmail.com

Krzysztof Z. Gajos
Harvard University
kgajos@seas.harvard.edu

ABSTRACT

Typical synthesizers only provide controls to the low-level parameters of sound-synthesis, such as wave-shapes or filter envelopes. In contrast, composers often want to adjust and express higher-level qualities, such as how 'scary' or 'steady' sounds are perceived to be.

We develop a system which allows users to directly control abstract, high-level qualities of sounds. To do this, our system learns functions that map from synthesizer control settings to perceived levels of high-level qualities. Given these functions, our system can generate high-level knobs that directly adjust sounds to have more or less of those qualities. We model the functions mapping from control-parameters to the degree of each high-level quality using Gaussian processes, a nonparametric Bayesian model. These models can adjust to the complexity of the function being learned, account for nonlinear interaction between control-parameters, and allow us to characterize the uncertainty about the functions being learned.

By tracking uncertainty about the functions being learned, we can use active learning to quickly calibrate the tool, by querying the user about the sounds the system expects to most improve its performance. We show through simulations that this model-based active learning approach learns high-level knobs on certain classes of target concepts faster than several baselines, and give examples of the resulting automatically-constructed knobs which adjust levels of non-linear, high-level concepts.

Author Keywords

Intuitive Control Knobs; Synthesizers; Sound Synthesis; Sound Design; Intelligent Interactive Systems; User Interfaces; Active Learning; Preference Learning; Gaussian Processes.

ACM Classification Keywords

H.5.5. [Sound and Music Computing]: Methodologies and Techniques; Signal Analysis, Synthesis, and Processing

INTRODUCTION

Composers and sound designers use synthesizers to create sounds with spectral and temporal dynamics beyond what is possible through acoustic instruments or recorded sounds. These users generally seek not just to create a certain sound, but rather a collection of related sounds, by transforming a set of sounds in semantically meaningful directions. The ability to intuitively adjust only the relevant qualities of a set of sounds is paramount to the artist's workflow. However, synthesizers are particularly difficult for artists to interact with, since their many parameters must often be adjusted in complex ways in order to change just a single semantically-meaningful aspect of a sound. This often forces a shift in the user's attention from expressing their high-level goals to the low-level trial-and-error tweaking of control parameters.

Several strategies have been developed to address this problem: First, providing the user with pre-set sounds having various qualities, such as "bright acoustic piano" or "dark acoustic piano". Second, interpolating between existing sounds, by averaging the low-level control settings which generate those sounds. Third, expert-engineered high-level knobs for directly controlling relatively intuitive aspects of sounds, such as Waves' OneKnobs, mostly used in the context of production to "brighten" or "phatten" tracks and mixes, and which under the hood controls sound-treatment modules such as equalizers, compressors, and resonators.

However, synthesizers can support more complex concepts, such as those that are multi-modal. For example there are different kinds of "scary" sounds, such as low rumbling, cold chilly wind-like sounds. And we might want different ways to adjust the "scariness" of a sound depending on what kind it is. For example, if a sound is directional, we might make it more "scary" by making it more aggressive. If a sound is more of an ambient pulsating sound, we can make it warble in more irregular ways or more high pitched, or perhaps add some eerie metallic "clicks" to its foreground.

In this paper, we explicitly model high-level sound qualities in order to allow users to automatically adjust arbitrary sounds to have desired levels of those qualities. We define a high-level concept as a function that maps from a high-dimensional control space to the perceived level of that quality. With such mappings, we can generate a high-level knob which directly adjusts sounds to have more or less of a given quality. These knobs are generated for each sound by constructing a path through control space which increases (or decreases) the level of the learned function. Such paths allow composers to move towards and away from sounds along perceptually meaningful dimensions. To learn such functions for a high-level concept, we learn from user ratings of sounds generated by the synthesizer.

In order to learn from a small number of user interactions, we take an active learning approach to assisting a user in finding points in the control-parameter space to demonstrate a high-level concept. The user first informs the system which sorts of sounds she wants to be able to modify and to which ranges. At any point in teaching the system a high-level concept, the user can ask the system where it wants to learn about the high-level concept the most. The system queries the user on the sound that it believes can most improve its ability to adjust the high-level quality for the given set of sounds. The user can listen to the sound and respond by rating how much they think the sound has the high-level quality. We have prototyped a user interface to allow users to express this rating by allowing them to organize sounds on a continuous one-dimensional space to indicate how much they think the sounds carry a particular high-level quality.

Our formulation can be applied to other domains where a user wants to build a layer of richer, personalized controls on top of the parameters provided by the original system.

RELATED WORK

We describe related work in computer-assisted sound design and music composition that uses the metaphor of knobs (one-dimensional, continuous controls), and also other kinds of interfaces and interactions that assist users in working with synthesizers. Our active-learning approach is informed by work in active learning in general, and active data selection, and Bayesian optimization.

High-level knobs

There have been a number of works on learning high-level "knobs" for audio editing tools such as equalizers and reverberators. [21,24,25] treat a knob as a fixed linear mapping between a low-level control and a high-level quality. They adopt a weighting-function procedure widely used in psychoacoustics to tune hearing aids, which operates by correlating gain in each frequency bin to a user's perceptual ratings.

Instead of mapping directly between parameters of a specific reverberator and user ratings, [22] derived nonlinear mappings from low-level controls to reverberation measures such as "echoness", which can be generalized onto different reverberators with similar parameterizations. Their system then learns linear mappings from reverberation measures to perceptual qualities such as "boomy". As sound modification on synthesizers is more complex and their parameterizations vary drastically across different synthesizers, it is more difficult to engineer a set of high-level functions *a priori*. Instead, we use the expressiveness of nonparametric methods to model the covariance structure between controls and to adjust to the complexity of functions being learned.

As user ratings for different high-level qualities may be correlated, [22] uses transfer learning to incorporate concepts taught by prior users. In this setting, each user rates only a subset of example sounds to train a user-concept, and the system fills in the rest of the ratings by weighting ratings of other user-concepts by their distances to the current user-concept. To choose the subset of sounds to query a user, the work uses active learning by selecting sounds that most differentiate between previously learned user-concepts. [3] crowdsources many more user-concepts to study which equalizer descriptors are widely-agreed-upon and which are true audio synonyms. Our method takes an active-learning approach to help a user more quickly teach a concept from scratch, with the objective of maximizing test-time performance, defined as the ability to construct high-level knobs that can adjust desired qualities in sounds with high certainty.

In additional to the "knobs" metaphor for interaction, [17] uses a 2D self-organizing map to place concepts with similar ratings close to each other, so that users can directly explore the semantic space instead of control space. [8] allows users to interact with different supervised learning algorithms to define mappings from arbitrary controllers to synthesis parameters. [7] describes a play-along mode where the user first composes a sequence of synthesis parameters and then gestures along as it is being played back. The parameter-gesture pairs become training data for learning a mapping that supports the reversed interaction. Instead of relying on user data, [14] applies data mining to existing presets to provide autocompletion. For example, as a user adjusts the controls, other statistically related controls adjusts itself accordingly.

A complementary interaction is when the user is not searching for a desired sound, but already has an audio recording of a target sound at hand, and wants the machine to automatically generate a synthesizer setting that approximates that sound. With such a setting, the sound is no longer canned, but can be manipulated in real-time using the synthesizer controllers. This is often achieved by minimizing the difference between the timbral trajectories of the synthesized and the target sound. As synthesizer controls are high dimensional and highly nonlinear, optimization techniques such as genetic algorithms or particle swarm optimization are often used to search through different synthesis structures and to perform parameter estimation, as in [10, 11, 16, 28]. These techniques also allow users to control a synthesizer by directly specifying values for acoustic features [12].

The "knobs" metaphor is also used in computer-assisted composition for helping users adjust high-level qualities of symbolic representations of music. For example, [18] exposes the log weighting between transition matrices in hidden Markov models trained on major ("happy") and minor ("sad") songs, as an intuitive knob for adjusting how "happy" an accompaniment sounds. [20] adjusts melodies to be more or less 'tonal', ‘serial’, or ’brown’ by moving closer or further away from that concept’s decision boundary.

Active learning

A number of systems have explored using active learning for assisting users in defining personalized concepts [1,9,21]. For example, [9] supports users in interactively defining concepts for re-ranking web images search results, and presents users the option of classifying images that are closest to the decision boundary between positive and negative classes, using a nearest-neighbor approach. They also use a heuristic to actively select images that explore new weightings in their distance metrics on low-level features used in computer vision.

More generally, one of the goals of active data selection is to select data that reduces the uncertainty the most on some parameters of interest, given past observations. For example, we can aim to maximize the expected information gain on model parameters by selecting data where the the posterior predictive variance is the highest [15]. Alternatively, the goal could be to minimize generalization error by choosing data that minimizes the overall variance of the predictor [4]. Both of these criteria have been employed in active Gaussian process GP regression, by leveraging the variance on the posterior predictive marginal distribution of GPs [26]. [13] adapts this approach into a classification setting of visual object category recognition, defines a covariance function that reflects local features for object and image representations, and then actively selects data closest to the decision boundary.

However, for settings where we are not given a set of examples a priori to select from, we need to first propose a set of queries from a continuous space, which is the case for our work. Furthermore, the objective may be more specific than reducing uncertainty on parameters. More generally, active learning is the setting where a model and an associated goodness (or loss) is first defined and then the goal is to query a next point that is predicted to result in a future model that produces the highest goodness (or equivalently lowest loss).

In our work, we define a new utility function for our domain by formalizing what the user would desire during test time. Our utility function captures how confident a model is in producing the desired knob paths for adjusting a given set of starting sounds. As in [19] and [2], our system learns in an iterative manner, and at each iteration we query the user with the point that maximizes the tool’s expected utility. While [2] uses Bayesian optimization to search globally to help users in procedural animation to find the best parameter setting, our goal is to focus learning in regions that are expected to later allow us to best fulfill user requests. As in [27], we compute the expected utility of a point by sampling its values from the current distribution, and evaluate the model’s expected confidence in adjusting sounds, conditioned on fantasized ratings.

OVERVIEW OF METHOD

We treat a high-level concept as a function that maps from synthesizer control parameter space $\mathbf{x} \in \mathbb{R}^D$ to perceived $y \in \mathbb{R}$ levels of that quality. If we knew this function $f(\mathbf{x})$, we could automatically adjust that concept on a preset $\mathbf{x}_s$ to a desired level y_d of that quality by moving $\mathbf{x}_s$ to a nearby location that has the desired quality level. A knob would move $\mathbf{x}$ along a continuous path through control-parameter space corresponding to increasing and decreasing levels of $f(\mathbf{x})$.

Since we do not know $f(\mathbf{x})$, we start with a probabilistic model of $f(\mathbf{x})$ and refine it with user feedback. Because the domain of f will be high-dimensional, learning the function over its whole domain would require an impractical amount of user feedback. Instead, we focus our learning in high-level knob space by asking for user feedback only in the parts of the control parameter space that is relevant to improving the paths for the high-level knobs.

To guide the acquisition of user feedback, we first specify how we expect the system will be used (which sorts of sounds will be modified, and the expected range of modifications). We can then estimate how well the tool can be expected to fulfill the user’s request, given the different queries made to the user during training time. We can then ask the user for the feedback which is most expected to improve the test-time performance of the system. In short, we will use active learning to select the queries made to the user in order to determine f in places which will be expected to improve the utility of the tool the most.

Determining good user queries requires two steps: first, proposing a set of possible user queries, then ranking those queries by how much they are expected to improve the utility of the tool. For the former, we develop a heuristic which proposes points nearby points visited during path optimizations. Second, to estimate a candidate’s expected impact of the utility of the system, we sample from the current posterior marginal distribution of f and, for each sample, evaluate the utility of the system as if we had actually made that observation. The candidate point that maximizes for expected utility of the system is chosen for user feedback.

MODELING HIGH-LEVEL KNOBS

We model the function f mapping from control parameters to high-level concepts probabilistically with Gaussian process (GP) priors [23]. In contrast to previous approaches which learned linear mappings [21], the non-parametric nature of GPs allows us to learn a function whose complexity can grow to match that of the function being learned. We use a zero-mean prior and the standard squared-exponential kernel:

$$f \sim GP(0, k)$$

$$k(\mathbf{x}, \mathbf{x}') = \sigma_y^2 \exp(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2\ell^2}) + \sigma_n^2 \delta_{ii'}$$

The kernel is parameterized by lengthscales ℓ, which specify the typical distances along which each dimension of f varies, and by the amplitude and noise variances σ_y^2 andσ_n^2, respectively. We denote these parameters collectively as θ.

GP regression is an appropriate tool for this problem for two reasons: First, it provides everywhere a closed-form estimate of the remaining uncertainty about the function being learned, which is necessary for estimating the expected performance of the model. Second, the marginal likelihood gives us a principled way to choose the parameters of the kernel.

Integrating over hyperparameters

Typically, the parameters of the kernel are estimated by maximum likelihood. However, when there are very few datapoints, these point estimates are known to drastically overestimate lengthscales, leading to an under-estimate of uncertainty about the function itself. As we need a useful estimate of uncertainty even conditioned on only a few data points, we must also characterize our uncertainty about the lengthscales of the GP. Therefore we take a fully-Bayesian approach by approximately integrating over hyperparameters using a Sobol sequence.

Our approximate integration over kernel parameters means that our predictive marginal posterior of f at a given point $\mathbf{x}^\star$ is a weighted sum of GP posteriors:

$$\begin{aligned} P(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) &= \int P(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}, \theta) P(\theta|\mathbf{X}, \mathbf{y}) d\theta \\ P(\theta|\mathbf{X}, \mathbf{y}) &= \frac{P(\mathbf{y}|\mathbf{X}, \theta) P(\theta)}{\int P(\mathbf{y}|\mathbf{X}, \theta') P(\theta') d\theta'} \end{aligned}$$

The mean and variance of the posterior marginal at $\mathbf{x}^*$ is given below. $\mu_i(\mathbf{x}^*)$ and $\sigma_i^2(\mathbf{x}^*)$ are the posterior mean and variance of the GP with a single set of hyperparameters θ_i.

$$\begin{aligned} \mu_i(\mathbf{x}^*) &= \mathbf{k}_{\theta_i}(\mathbf{X}, \mathbf{x})^T \mathbf{K}_{\theta_i}^{-1}(\mathbf{X}, \mathbf{X}) \mathbf{y} \\ \sigma_i^2(\mathbf{x}^*) &= \mathbf{k}_{\theta_i}(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{\theta_i}(\mathbf{X}, \mathbf{x})^T \mathbf{K}_{\theta_i}^{-1} \mathbf{k}_{\theta_i}(\mathbf{X}, \mathbf{x}) \\ \mu(\mathbf{x}^*) &= \sum_{i=1}^{M} P(\theta_i|\mathbf{X}, \mathbf{y}) \mu_i(\mathbf{x}^*) \\ \sigma^2(\mathbf{x}^*) &= \sum_{i=1}^{M} P(\theta_i|\mathbf{X}, \mathbf{y}) \left([\mu_i(\mathbf{x}^*) - \mu(\mathbf{x}^*)]^2 + \sigma_i^2(\mathbf{x}^*) \right) \end{aligned}$$

To sample a y^* given x^*, we first sample which set of hyperparameters to use, and then sample from the predictive marginal posterior given just that set of hyperparameters.

Knob paths

Once we have an estimate of the function defining a high-level quality, we can generate a knob which varies that quality for a preset sound by finding a path from the preset through control-parameter space corresponding to increasing and decreasing levels of the learned function. We generate such paths by first locally optimizing for each of a set of equally-spaced desired quality levels $\mathbf{y}_d$ spanning from the lowest user rating to the highest, then linearly interpolate between these points. In our experiments, we used 8 equally spaced levels. The objective of this optimization is to minimize the squared difference between the posterior predictive mean and a desired knob level y_d when starting from a control-parameter setting $\mathbf{x}_s$. We denote the control-parameter setting returned by the optimizer as $\mathbf{x}_{sd}$.

$$\mathbf{x}_{sd} = \arg\min_{\mathbf{x}^*} \left(y_d - \mu(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) \right)^2 \tag{1}$$

In future work, we plan to take uncertainty into account while optimizing for desired knob levels, and explicitly optimizing for desirable properties for knobs such as smoothness.

ACTIVE LEARNING

Our system learns in an iterative manner: At each iteration, we choose a point at which to query the user, based on the current model of f. We then update the model of f, and re-query the user. To determine which point to query the user at, we first propose a set of *candidate points*, whose expected impact on the utility of the system we will estimate. After estimating these utilities, we then query the user at the best candidate point.

Evaluating the expected utility of a point

Given a set of *candidate points*, we need to decide which one can most help us improve the utility of our model. Formally, we define *utility* $\mathcal{U}$ as the total mass within ϵ-wide bins centered at the desired knob levels $\mathbf{y}_d$ for all presets $\mathbf{X}_s$.

$$\begin{aligned} \mathcal{U}(\mathbf{X}, \mathbf{y}) = &\sum_{s=1}^{S} \sum_{d=1}^{L} [P(y_{sd} > (y_d + \epsilon)|\mathbf{x}_{sd}, \mathbf{X}, \mathbf{y}) \\ &- P(y_{sd} > (y_d - \epsilon)|\mathbf{x}_{sd}, \mathbf{X}, \mathbf{y})] \end{aligned} \tag{2}$$

This expression can be intuitively thought of as the probability that we will be able to give the user a sound with the desired high-level quality by moving along the learned paths.

Reoptimizing hyperparameters under fantasies

To evaluate the expected impact of a candidate $\mathbf{x}_c$ on our system, we sample the current posterior predictive marginal distribution $P(y_c|\mathbf{x}_c, \mathbf{X}, \mathbf{y})$ for what the possible perceived y_c could be. In order to more accurately evaluate expected utilities, we retrain our model for each sample by re-optimizing the hyperparameters on the GP prior as if we had seen this additional observation, resulting in a fantasized posterior predictive marginal distribution of $P(y^*|\mathbf{x}^*, \mathbf{x}_c, y_c, \mathbf{X}, \mathbf{y})$ on which the desired knob levels are optimized. Intuitively, the hyperparameter re-optimization is helpful because it accounts for the fact that new queries can potentially change our estimates of the lengthscales, which can drastically change our model's predictions. This is essential early on when we have not yet seen much data, and are very uncertain about the lengthscales.

$$\mathbf{E}(\mathcal{U}|\mathbf{x}_c, \mathbf{X}, \mathbf{y}) = \int \mathcal{U}(\mathbf{x}_c, y_c, \mathbf{X}, \mathbf{y}) P(y_c|\mathbf{x}_c, \mathbf{X}, \mathbf{y}) \mathrm{d}y_c \tag{3}$$

Proposing candidate points

Intuitively, we want to learn more about points that can potentially improve the knob paths. We use a heuristic for generating candidate points which are likely to be useful: We sample points nearby those visited by the truncated-Newton optimizer while determining the knob paths.

Because we expect f to vary slowly in directions which have long lengthscales, we heuristically add gaussian noise to candidates $\mathbf{X}_c$ with variances proportional to the length scales of that dimension. Hence, to determine the noise for a particular candidate x_c, we first sample a set of length scales according to their marginal likelihood. Then, for each dimension d, we sample from a gaussian with variance proportional to the lengthscale l^d of that dimension. We reject any points that fall outside the domain of the synthesizer.

$$\epsilon_c^d \sim \mathcal{N}(0, \frac{l^d}{2})$$
$$\mathbf{x}_c^d \leftarrow \mathbf{x}_c^d + \epsilon_c^d$$

To further increase the potential information gain, we also heuristically include as candidate points the peaks on the posterior variance of f that are reachable through the truncated optimizer from the given set of presets $\mathbf{X}_s$.

The active-learning algorithm and its variations

Algorithm 1 outlines our procedure for how to decide where the query the user next for feedback. Both `for` loops can be parallelized. Figure 1 shows a synthetic 1D visualization of some the steps involved.

Algorithm 1 Choosing the next point $\mathbf{x_r}$ for user to rate

```
Input: $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, $\mathbf{X}_s$, $\mathbf{y}_d$, $n_c$: num of candidates, $n_m$: num of monte carlo samples
Train full-bayes GP with $\mathcal{D}$
Optimize (1) for $\mathbf{X}_{sd}^{\text{path}}$ on GP given $\mathbf{y}_d$ and $\mathbf{X}_s$
Randomly choose $n_c$ candidates $\mathbf{X}_c$ from points
    visited during the previous optimization step
for $i = 1$ to $n_c$ do
    for $j = 1$ to $n_m$ do
        Sample $y_{c_{ij}} \sim \mathcal{GP}(y|\mathbf{X}_{c_i}, \mathcal{D})$
        Fantasize $GP_{c_i}$ by training it with $y_{c_{ij}}$, $\mathbf{X}_{c_i}$, $\mathcal{D}$
        Optimize (1) for $\mathbf{X}_{sd_{ij}}^{\text{path}}$ on $GP_{c_i}$ given $\mathbf{y}_d$ and $\mathbf{X}_s$
        Compute $\mathcal{U}_{ij}$ by (2)
    end for
    $\mathbf{E}(\mathcal{U}_i) = \frac{1}{n_m}\sum_{j=1}^{n_m}\mathcal{U}_{ij}$
end for
minIndex $= \arg\min \mathbf{E}(\mathcal{U}_i)$
$\mathbf{x}_r = \mathbf{X}_{c_{\text{minIndex}}}$
```

We later refer to our full-fledged path-informed model-based active-learning procedure as "active learning (re-opt)". We also experimented with a number of simplifications, both as baselines to compare against, and also because they can have shorter run times and so can be more naturally integrated into interactive systems. "Active learning" takes out the step of reoptimizing hyperparameters after each fantasized outcome. "Path entropy" skips the fantasizing step all together, and chooses the point that has the highest variance from the set of proposed candidate points, while "Path random" simply chooses a point at random from the proposed points.

USER INTERFACE AND INTERACTION

In order to support users in defining their own high-level concepts, we have prototyped a user interface that allows users to demonstrate high-level concepts with examples. The user can communicate how much a sound carries a high-level concept by placing it accordingly into a box, as shown in Figure 2, where the horizontal axis indicates an increasing level of a high-level concept from left to right. This allows users to rate sounds in reference to each other, to correct previous ratings, and also to re-adjust their overall scales.

If the user has some prior knowledge of which control parameters might give rise to a high-level concept, she can help the machine reduce the dimensionality of the problem by selecting a subset of parameters, as illustrated by labels 1 and 2 in Figure 2. The user can also make such decisions later on in the process as she gains more experience with the synthesizer.

As the user has accumulated a number of points in the box, the user can ask the machine to learn the concept and apply it to a starting sound in order to check how well the machine is understanding the concept. This can be performed at the interface by first hitting the "play" button, at which point the system trains the model and then optimizes for a path through the sound to increasing and decreasing levels of the learned concept. The system has been set to optimize for three discrete levels on the learned function. Both these four points and the original sound are added to the box and placed horizontally at where the model believes it should be. These points are the "darker" row of dots labeled as 7 in Figure 2, where the second dot from the left is the original sound. The user then has the option of correcting these points by moving them left or right, and the moved points become a new data point for training the high-level concept. The user can now move the slider above the box to adjust her sound according to the learned concept, and see how it controls the synth control parameters.

Assisting the user with active learning

The user can specify how she expects to use the system by first choosing a set of presets she wishes to modify. This can be indicated in the interface by suppressing those presets, labeled as 3 in Figure 2. Then at each iteration, the user can ask for different kinds of assistance by clicking on the corresponding helper buttons. For example, the "next filler random" button labeled as 5 calls for the "path random" variation of our active-learning algorithm to propose the next point to evaluate. A new control parameter setting is then added as a dot to the box, placed at a horizontal position where the model currently believes how much of the high-level concept it carries. The user can help refine the model by correcting these points.

EXPERIMENTS AND EVALUATIONS

We ran a preliminary pilot study with two composers to collect high-level concepts. We evaluate our method in two parts: how well our model captures these high-level concepts, and how quickly our active-learning approach is able to learn these high-level knobs compared to other baselines. For each concept, we show a few examples of knob paths.

Pilot study with two composers

The task was for users to build a knob for a high-level quality such as "scariness" that she wishes to more directly control during sound synthesis. First, the user was prompted to choose a set of preset sounds on which she would test the high-level knob. The user was then asked to identify a set of examples on the synthesizer that carries varying degrees of that high-level quality. Users interacted with our interface shown in Figure 2, with the option of asking for suggested examples from the lightweight variation "path random" of

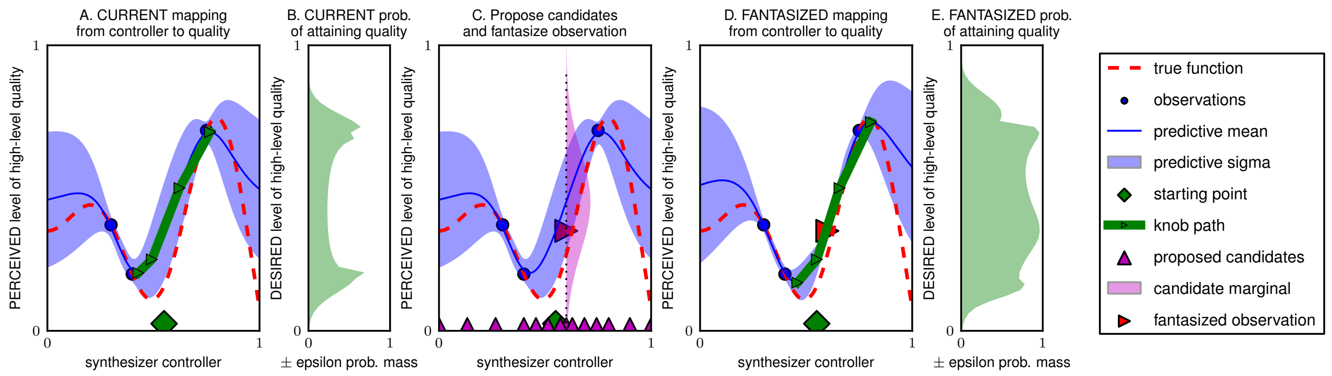


Figure 1. Synthetic 1D example of an 1-sample evaluation of the expected utility of a proposed candidate, where the model's utility increased from (B) to (E). Plots A, C, D refer to mappings from control-parameter (x-axis) to a high-level quality (y-axis). Plots B, E refer to the model's confidence in attaining desired levels of a high-level quality (y-axis) measured by the$\pm\epsilon$ probability mass (x-axis) from the mean of the predictive marginal for the corresponding control-parameter on the knob path (green line in preceding plot). (A) Current mapping from synth controller to high-level quality, given three ratings of perceived levels of high-level quality. The green diamond is a sound in the synth control space that is to be adjusted. (C) Proposed candidates shown as magenta triangles on x-axis. The Gaussian predictive marginal of one of the candidates is shown and the red-triangle shows a sample from that distribution. (D) The mapping after fitting the model with the addition fantasized observation (red-triangle).

our active-learning. The interface allows users to specify the amount of a perceived quality in a sound by moving the dot that represents it on a one-dimensional axis. The synthesizer used for the user studies is a representative software synth named FreeAlpha, from Linplug.

Modeling high-level concepts

From these two pilot studies, we collected three examples of nonlinear concepts on synthesizers, "pulsation", "guitar-like" and "scary". The first and third are multimodal, while the second is unimodal. They range from concrete to abstract. The "pulsation" concept was obtained by querying a single user uniformly on a 8x8 grid in the control-parameter space of 2 synth controls. As we wanted to use this concept for our later simulations, we did not want to bias the input distribution with any of the methods that were being compared in the simulations. For the "guitar-like" and "scary", the user was free to interact with our interface and ask for suggested queries from the "path-random" variation of our active-learning method.

We first illustrate these concepts and then show cross-validation results on how our model performs when trying to predict how users would rate a sound.

Example concept 1: "pulsation"

We want to learn about how two control parameters, the rate of low frequency modulation FM and amplitude modulation AM, interact to produce different degrees of perceived "pulsation". Turning either of these controls up increases pulsation up to a point where modulation becomes so fast that we can not perceive distinctive vibratos anymore, but instead we hear a timbral color change. Moreover, they interact in how they give rise to perceived "pulsation". When FM is low, it dominates our perception, and turning up AM does not give any effect. However when FM is so high that we cannot hear the pulsation anymore, for example the diamond in the lower right corner of Figure 7, turning AM up in the lower ranges allows us to add more "pulsation" to our colored sound. We can observe this effect on the response surface of the mean of the GP conditioned on a grid of 8x8 user ratings on these two control dimensions, shown in Figure 7.

Example concept 2: "guitar-like"

Figure 3, left, shows the learned model of the "guitar-like" quality, over 2 of the the 4 control dimensions. The GP characterizes this quality as peaking when the attack time on the amplitude envelope is short, the "cut-off" on the bandpass filter is low, the frequency of a low-frequency oscillator (LFO) is medium, and the detune on the chorus is high. Colloquially, a guitar-like has a "sharp attack", with some warble in the release, and is coloured with some "detuned harmony". In the space of these four control-parameters, this function is unimodal, which means all paths will be maximized at the same point, as shown in figure 3, left.

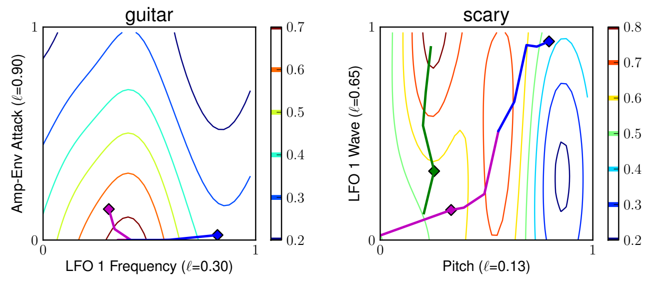
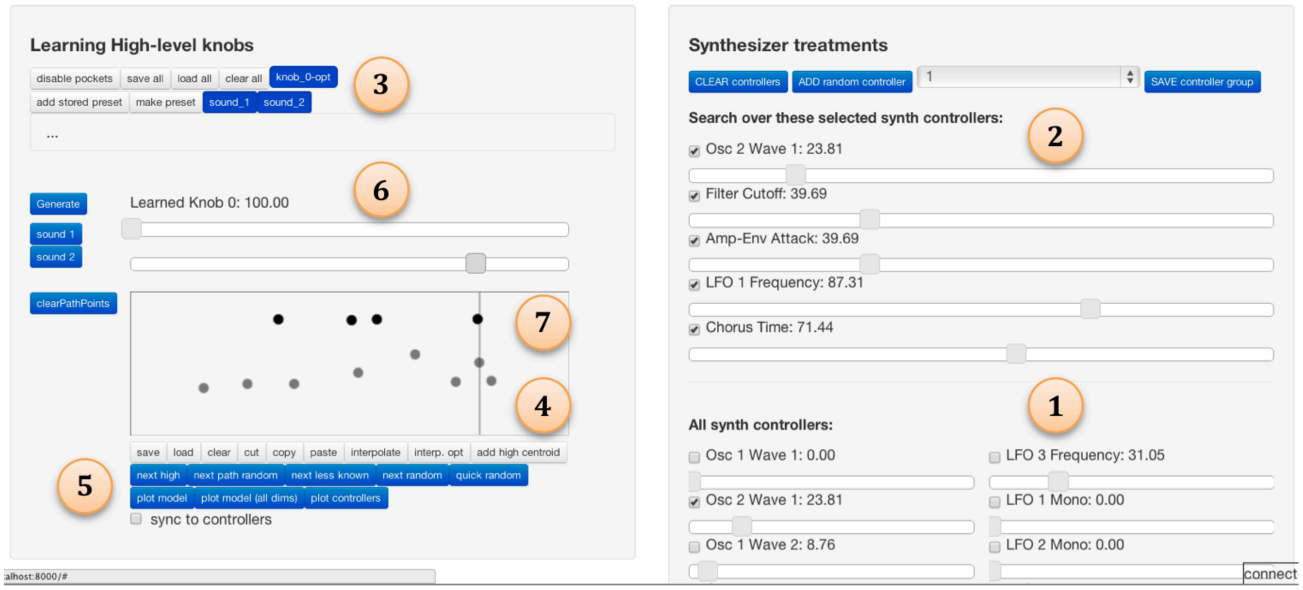


Figure 3. Visualizations of high-level concepts. Contours represent the full-Bayes GP posterior mean of the degree of the high-level quality as a function of synthesizer controls. Red, blue and green lines represent paths through control-space which vary the high-level quality, starting from sounds denoted by black diamonds. Left: 2D slice of posterior mean of "guitar-like" function, and knob paths. Right: 2D slice of posterior of "scary" function.

Example concept 3: "scary"

We model this quality as a function of four different control-parameters: the depth of a filter envelope, the waveshape and the frequency of the LFO, and the main pitch of the synth. In figure 3, right, we show a slice of the mean of a GP conditioned on user ratings. In this slice, both the LFO frequency and the depth of the filter envelope are fixed at 0.7 which



Learned Knob 0: 100.00

Search over these selected synth controllers:

- ☑ Osc 2 Wave 1: 23.81
- ☑ Filter Cutoff: 39.69
- ☑ Amp-Env Attack: 39.69
- ☑ LFO 1 Frequency: 87.31
- ☑ Chorus Time: 71.44

All synth controllers:

- ☐ Osc 1 Wave 1: 0.00
- ☐ LFO 3 Frequency: 31.05
- ☑ Osc 2 Wave 1: 23.81
- ☐ LFO 1 Mono: 0.00
- ☐ Osc 1 Wave 2: 8.76
- ☐ LFO 2 Mono: 0.00

Figure 2. User interface for learning and applying high-level knobs to sounds. (1) Sliders for adjusting the low-level control-parameters on the synthesizer directly. There are a total of 64, many are cropped in this screenshot for space. Users can choose a subset of low-level controls to optimize over by checking the checkboxes, and these controls will pop up in region (2). (3) Preset buttons for users to choose as starting sounds. Users can also instantiate new sounds as buttons. (4) Box for users to give ratings to sounds. The horizontal axis of the box reflects how much a sound carries a high-level concept, increasing from left to right. Each training example is represented as a dot and corresponds to a particular synthesizer control-parameter setting. Users can click on buttons in region (5) to ask the tool to suggest sounds to add as training examples. (6) Sliders for adjusting high-level qualities in sounds directly once the model is trained. They are like macro knobs that control the low-level synthesizer parameters in region (2). Each slider corresponds to a knob path that is specific to a starting preset in region (3). (7) The row of darker dots correspond to optimized points on the high-level knob path.

corresponds to a steady pulsating rumbling base sound. As the LFO parameter increases, the waveshape becomes more complex, for example going from sine waves, to sawtooth, and finally to a much noisier wave, introducing hisses into the foreground making the sound more scary. As the main pitch of the synth increases, the volume of the "scary" sound body also increases. However, if the pitch is increased too much, the sound becomes much thinner and loses its force. In between the two modes, the foreground hissing and the background rumbling merge, and the sound becomes slightly less scary.

Predicting user ratings

We compare predictive performance of our GP-based model to other models on held-out user ratings. The results are shown in Table 1, where we see that for multimodal qualities "pulsating" and "scary", our model predicts user ratings better than support vector regression (SVR) with a radial basis kernel, decision tree regression (DTR), and linear regression. Although the predictive performance of the GP is not uniformly better than other regression techniques, the probabilistic nature of the GP enables active learning, and the smooth function estimates it provides enable us to compute continuous adjustment paths. Future work will revisit the covariance function used in our model, in order to allow it to capture more structure.

Evaluating active learning

To evaluate our path-informed model-based active-learning algorithm, we run simulations on two functions to compare

quality \model	GP	SVR	DTR	LINEAR
pulsating (n=64, d=2)	**0.036**	0.051	0.051	0.073
guitar-like (n=31, d=4)	0.042	0.029	**0.026**	0.040
scary (n=64, d=4)	**0.065**	0.068	0.116	0.071

Table 1. Comparing the 10-fold cross-validation mean-squared error of different models on user ratings of different high-level qualities. n is the number of ratings, and d is the number of control parameters being varied. Each quality's ratings were gathered from a single user. Users rate sounds by placing sounds on a one-dimensional interface.

their performances to several baselines. We used IPython's parallel framework to evaluate the expected utility of each candidate in parallel, corresponding to the outer for loop in Algorithm 1. We first started with a simple synthetic function, and then ran another set of simulations on a function learned from user ratings. We show that our method in the former case is able to learn target concepts faster according the metric we define below in Eqn. (4) by focusing on regions most relevant to the knob paths. For the latter function, even though our method did not learn faster than some baselines according to our current metric, it performed qualitatively better by being able to identify multiple modes, and to route different starting points to their nearby peaks. In future work, we need to define an error metric that captures the multimodal nature of synthesizer concepts. We also wish to devise a metric that depends on relative perceptions. For example, a user may be less concerned about a knob giving a sound that is exactly 0.5 "scary", if there even exists such a notion, but instead she might be more concerned if moving a knob in one part of its range changes the sound a lot more than some other parts.

As a start, we define *error* as the sum of absolute differences between the desired levels $\mathbf{y}_d$ and the actual levels of the points returned when optimizing for the desired levels, summing over the knob paths for all starting points $\mathbf{X}_s$. $\mathbf{X_i}$ and $\mathbf{y_i}$ denotes the accumulated points the models have evaluated up to iteration i plus all the starting points$\mathbf{X}_s$. f here is the function that simulates the human rating. In the first experiment, this function is a synthetic function, while in the second experiment, this function is the mean of aGP conditioned on the 8x8 grid of user-ratings of the "pulsation" concept.

$$err(\mathbf{X_i}, \mathbf{y_i}, \mathbf{X_s}, \mathbf{y_d}) = \sum_{\mathbf{s=1}}^{\mathbf{S}} \sum_{\mathbf{d=1}}^{\mathbf{L}} |\mathbf{f}(\mathbf{x_{sd}}) - \mu(\mathbf{y_{sd}}|\mathbf{x_{sd}}, \mathbf{X_i}, \mathbf{y_i})| \tag{4}$$

We compare our method to several baselines, including proposing points according to the latin hypercubes, and an "entropy" baseline that proposes as the next point the mode on the variance of the posterior predictive marginal of the GP, by running local optimizers from the initial presets. We also compare between several variations of our path-informed active-learning method. "Active learning (re-opt)" re-optimizes the hyperparameters after each fantasized outcome, while "active learning" does not. The "path random" variation skips the fantasizing step, and choose at random a point from the set of proposed candidate points, which were originally randomly sampled from points nearby those visited by the truncated-Newton optimizer while determining the knobs paths. The "path entropy" variation also skips the fantasizing step, and chooses the point that has the highest variance among the proposed candidate points.

In both of the experiments, the number of candidates proposed was 15 and the number of Monte Carlo samples taken for each candidate was 30. The simulations were initialized with simulated ratings of 2 to 3 preset control-parameter settings, which corresponds to the sounds to be adjusted by the knob paths. The experiments were terminated when the performance started to plateau Figure 4 or when the model began to pick up the multimodality of the rating function Figure 4. For all our active-learning variations, the means and standard errors are averaged across five runs.

Experiment 1: Synthetic function

The synthetic function simulates a concept where most of the space does not give rise to its quality. The function was originally two-dimensional. In order to experiment with how the algorithm would perform in higher dimensions, we artificially added two more dummy dimensions, whose values do not affect the function. Figure 4 shows the performance of our method and other baselines as the models iteratively adds more data observations.

Figure 4 shows that our active-learning methods are able to learn high-level knobs faster under our error metric Eqn. (4). Furthermore, the downward curves of the active-learning methods show that they are able to iteratively improve and refine the model, while for other techniques the lines rise and fall with much higher error. Figure 5 shows the different response surfaces at the ninth iteration. We can see that our active learning approach was able to focus the evaluations in the region that is most relevant to the knob paths.

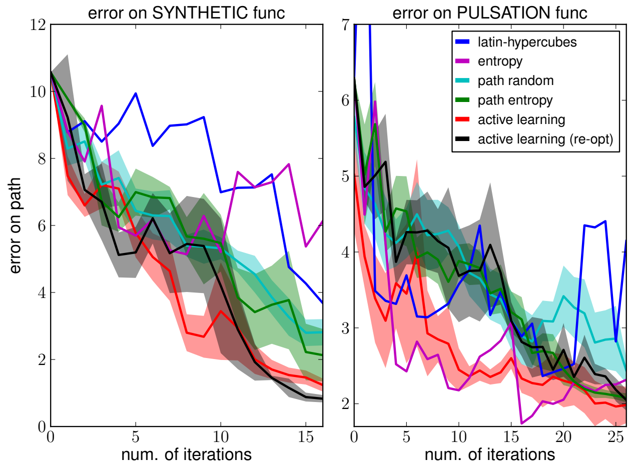


Figure 4. *Error* for active-learning variations and baselines on predicting knob paths on synthetic function (left), and "pulsation" function (right). Solid lines represent the mean, and shaded colors show the standard error over 5 runs.

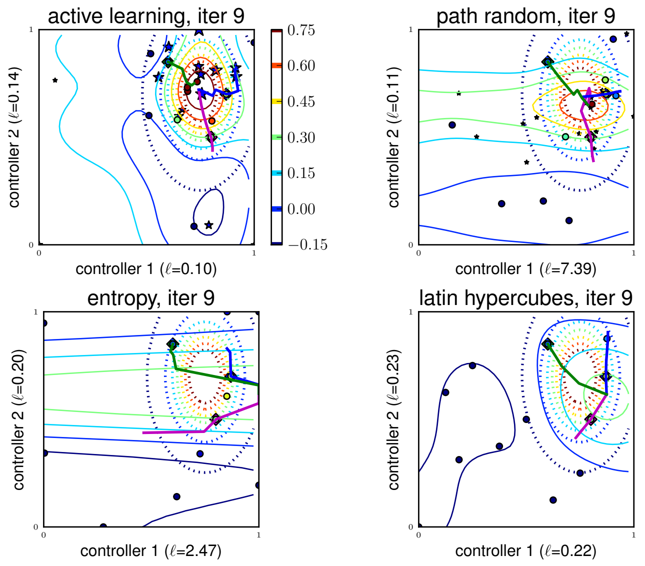


Figure 5. In the synthetic experiment, the full-Bayes GP posterior mean for different methods after 9 more observations beyond initialization. ℓs on the X and Y axes give the lengthscales of the corresponding low-level controllers under MLE. Dotted contour lines represent the target function. Dots indicate observations, and stars correspond to proposed candidates, which only applies to the active-learning methods.

The objective of our active-learning approach is to reduce the uncertainty on the knob paths, as opposed to learning the function everywhere. Figure 6 shows how well our "active-learning (re-opt)" method and the "entropy" baseline predicts the true function over the entire space. We evaluate the two corresponding GPs at 1000 locations given by the Sobol sequence, and plot these predicted values against those of the true function. This true function has a lot of low regions around zero, corresponding to the whitespace in Figure 5 without dotted contours. Only about a quarter of the space is occupied by a Gaussian bump. This shape is typical of mappings on synthesizers where most of the sounds in a space have none of a certain quality, and only a small part of the space gives rise to that quality. However, a model would have a low mean-squared error if it simply predicted low values everywhere. This is undesirable for constructing knobs, because the knob would be very flat and would not be able to make much adjustments to sounds. We see that in this case, our active-learning method is still able to predict well over a wider range of levels. Hence, if a user is able to find a sound that begins to have some of her desired qualities, then our tool will be able to give her knobs that can help her increase or decrease that degree, and this adjustment is often challenging for users to perform manually because low-level controls interact.

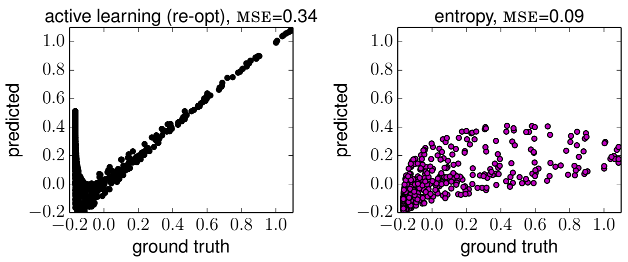


Figure 6. In the synthetic experiment, a comparison of how active learning and the "entropy" baseline predict the target function over the entire space. MSE **corresponds to the mean-squared error.**

Experiment 2: "Pulsation" function

The function learned from perceptual data is based on user ratings of sounds in terms of their "pulsation" - a high-level concept where amplitude and frequency modulation interact to give a non-trivial response surface. The user was queried at an 8x8 grid on two synth control parameters, and the mean of a GP conditioned on these ratings was used during comparison of learning procedures. We also artificially added two more dummy dimensions to this function, whose values do not affect the function.

Figure 4 shows the performance of our method and other baselines as the models iteratively adds more observations. According to our current error metric, the "entropy" baseline reaches the lowest error the quickest, at iteration 16. However, it was not able to discover that the concept is actually multimodal, as shown in Figure 7. In contrast, the best iteration from our active learning (with hyperparameter re-optimization) was able to learn that there was more than one mode in the concept, and routed the starting points to their nearest peak, as shown in Figure 7. Figure 7 shows the functions learned by the different techniques when we terminated the experiments at iteration 27.

DISCUSSION

We now discuss a few observations and lessons learned from our experiments. Currently, we assume users are able to identify a small subset of relevant low-level controls, and our

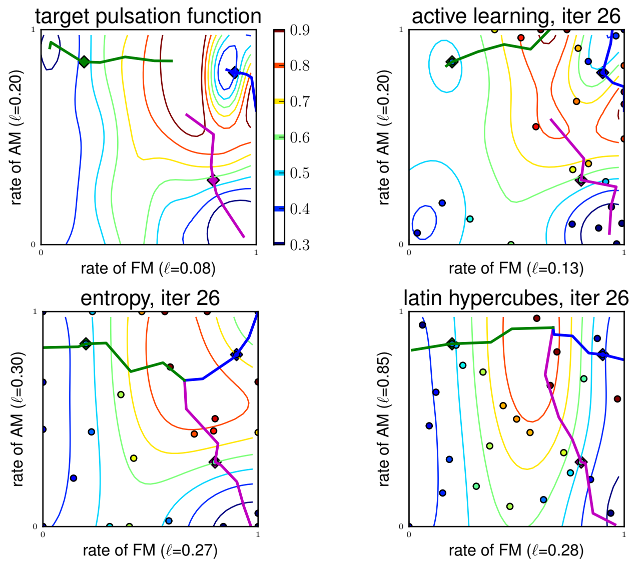


Figure 7. In the "pulsation" experiment, the full-Bayes GP **posterior means for different methods after 26 more observations beyond initialization. Note the upper-left subplot shows the target function.**

method focuses on learning the nonlinear interaction between these controls. To scale to higher-dimensional settings, we will have to encode domain knowledge into the covariance structure of our priors. For example on equalizers, adjusting gains for neighboring frequency bins vary perceptually in similar ways, we can parameterize the distance metric in our kernel by the Malahanobis distance to explicitly model covariance between dimensions. Alternatively, if we know the kinds of interaction between controls vary in different parts of the control space, we can call for a non-stationary kernel. See [6] for insights on how to choose the structural form of the kernel.

Moreover, our method for constructing knob paths is still naive, as it only optimizes separately for different knob values and there is no guarantee that the resultant path is coherent. For example, a path could jump around in the control space due to the many-to-one mapping from control space to musical concept. In the future, we plan to regularize our paths and optimize for a path as a whole so that we can control its perceived smoothness.

Furthermore, our user studies are still preliminary. More human-in-the-loop experiments are needed to better understand how users use our procedure to help them build personal control knobs. Casual observations show that users construct preferences on-the-fly, and they alternate between phases of exploration and refinement, analogous to phases of divergent and convergent thinking [5]. For example, one composer requested sounds from our lightweight "path random" active-learning procedure to refine his "guitar-like" knob, after a dozen of sounds he felt the need to explore whether different kinds of "guitar-like" qualities existed. So he switched to using our entropy-based procedure to try to touch the boundaries of the space, and was pleased that it gave him sounds that he had never heard before. Second, the listening mode of the user often evolves. Initially, users tend to react intuitively to sounds, but later on, they begin to listen more analytically, and explicitly reason and weigh sonic attributes when rating sounds. For example, a sound may have a "sharp attack" like a guitar but may "warble" too quickly to be one, and a composer may still rate it above average because it possess at least one of the main characters of a guitar sound. Third, as user preferences may change and in light of new possibilities, the re-rating of existing sounds becomes necessary, and one composer requested for a feature to "shave off" similar sounds, so that the new ratings and new sounds can have more influence.

CONCLUSION

Composers seek to explore sounds along intuitive and personal sonic dimensions, but synthesizers can often only be controlled through low-level controls that interact in complex ways. Inspired by this mismatch, we proposed a novel formulation of high-level knobs that treats a knob as a dynamic mapping that allows us to adjust different sounds along different paths in the control space. In this paper, we presented two building blocks towards realizing such knobs. We adopted the expressiveness of a fully-Bayesian nonparametric model, Gaussian processes, to model the rich perceptual world of synthesizers, where there are many ways to achieve a certain musical quality.

Second, to assist users in finding the set of examples to demonstrate a high-level concept, we derived a model-based active learning algorithm that queries the user in order to improve knob paths, and we showed in simulation that it is more effective for learning high-level knobs. Our procedure is modular, allowing us in the future to swap in different knob path formulations. As there are often multiple ways of achieving a certain desired effect, as shown by the multimodality of our example concepts, we are considering a variation that we call "branching" knobs, which would provide the user with multiple paths for adjusting a sound, some with endpoints at modes further away or higher than others. This allows composers to explore a wider palette of related sounds, and to trade off between achieving a desired adjustment and preserving different aspects of the original sound. Last by not least, our formulation is general, and can be applied to any other domain for users to construct and calibrate a layer of richer, personalized controls on top of the parameters provided by the original system.

REFERENCES

1. Amershi, S., Fogarty, J., Kapoor, A., and Tan, D. Effective End-User Interaction with Machine Learning. *AAAI* (2011).
2. Brochu, E., Brochu, T., and de Freitas, N. A bayesian interactive optimization approach to procedural animation design. In *Proceedings of the ACM SIGGRAPH* (2010).
3. Cartwright, M., and Pardo, B. Social-eq: Crowdsourcing an equalization descriptor map. In *Proceedings of the International Society for Music Information Retreival Conference* (2013).
4. Cohn, D. Neural network exploration using optimal experiment design. *Neural Networks 9*, 6 (1996).
5. Cropley, A. In praise of convergent thinking. *Creativity Research Journal 18*, 3 (2006).
6. Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of International Conference on Machine Learning* (2013).
7. Fiebrink, R., Cook, P. R., and Trueman, D. Play-along mapping of musical controllers. In *Proceedings of the International Computer Music Conference* (2009).
8. Fiebrink, R., Trueman, D., and Cook, P. R. A meta-instrument for interactive, on-the-fly machine learning. In *Proceedings of the New Interfaces for Musical Expression* (2009).
9. Fogarty, J., Tan, D., Kapoor, A., and Winder, S. Cueflik: Interactive concept learning in image search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2008).
10. Garcia, R. A. Automatic design of sound synthesis techniques by means of genetic programming. In *Audio Engineering Society Convention 113* (2002).
11. Heise, S., Hlatky, M., and Loviscach, J. Automatic cloning of recorded sounds by software synthesizers. In *Audio Engineering Society Convention 127* (2009).
12. Hoffman, M., and Cook, P. R. Feature-based synthesis: mapping acoustic and perceptual features onto synthesis parameters. In *Proceedings of the International Computer Music Conference* (2006).
13. Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. Active learning with gaussian processes for object categorization. In *Proceedings of the IEEE International Conference on Computer Vision* (2007).
14. Loviscach, J. Programming a music synthesizer through data mining. In *Proceedings of New Interfaces for Musical Expression* (2008).
15. MacKay, D. J. C. Information-based objective functions for active data selection. *Neural Computation 4* (1992).
16. Macret, M., Pasquier, P., and Smyth, T. Automatic calibration of modified fm synthesis to harmonic sounds using genetic algorithms. In *Proceedings of the 9th Sound and Music Computing Conference* (2012).
17. Mecklenburg, S., and Loviscach, J. subjeqt: Controlling an equalizer through subjective terms. In *CHI Extended Abstracts on Human Factors in Computing Systems* (2006).
18. Morris, D., Simon, I., and Basu, S. Exposing parameters of a trained dynamic model for interactive music creation. In *Proceedings of the National Conference on Artificial intelligence*, AAAI (2008).
19. Osborne, M. A., Garnett, R., and Roberts, S. J. Active data selection for sensor networks with faults and changepoints. *International Conference on Advanced Information Networking and Applications* (2010).
20. Pachet, F. Description-based design of melodies. *Computer Music Journal 33*, 4 (2009).
21. Pardo, B., Little, D., and Gergle, D. Building a personalized audio equalizer interface with transfer learning and active learning. In *Proceedings of the ACM workshop on Music information retrieval with user-centered and multimodal strategies* (2012).
22. Rafii, Z., and Pardo, B. Learning to control a reverberator using subjective perceptual descriptors. In *Proceedings of the International Society for Music Information Retrieval Conference* (2009).
23. Rasmussen, C. E., and Williams, C. K. I. *Gaussian processes for machine learning*. MIT Press, 2006.
24. Sabin, A. T., and Pardo, B. A method for rapid personalization of audio equalization parameters. In *Proceedings of the ACM International Conference on Multimedia* (2009).
25. Sabin, A. T., Rafii, Z., and Pardo, B. Weighting-function-based rapid mapping of descriptors to audio processing parameters. *Journal of the Audio Engineering Society 59*, 6 (2011).
26. Seo, S., Wallat, M., and Graepel, T. Gaussian process regression: Active data selection and test point rejection. *International Joint Conference on Neural Networks IJCNN* (2000).
27. Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems* (2012).
28. Yee-King, M. J. An autonomous timbre matching improviser. In *Proceedings of the International Computer Music Conference* (2011).