Sequential Gallery for Interactive Visual Design Optimization

YUKI KOYAMA, National Institute of Advanced Industrial Science and Technology (AIST), Japan
ISSEI SATO, The University of Tokyo, Japan
MASATAKA GOTO, National Institute of Advanced Industrial Science and Technology (AIST), Japan

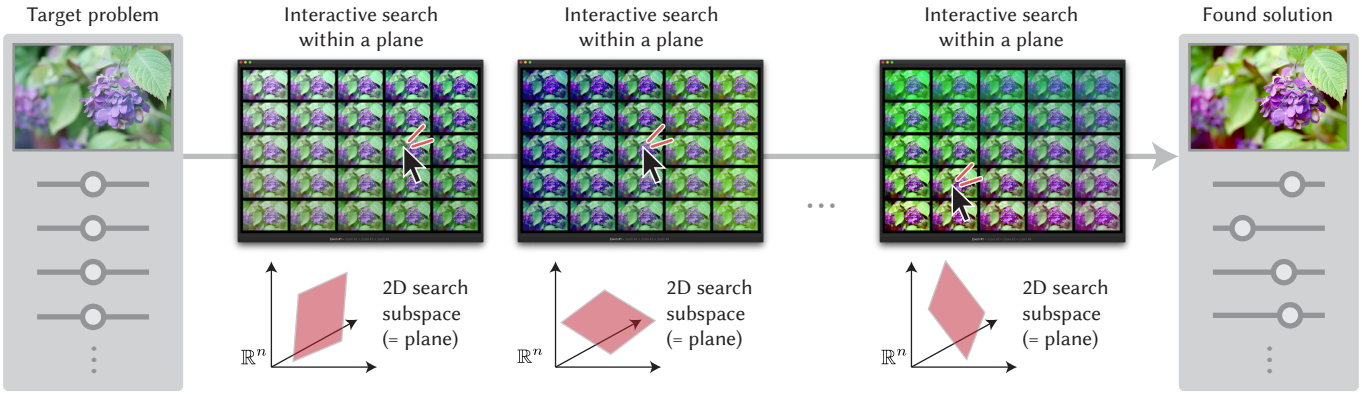

Fig. 1. ***Sequential Gallery*** is an interactive framework for exploring an n-dimensional design space formed by a set of n sliders and then finding an appropriate parameter set from that space. This framework lets the user sequentially select the most preferable option from the options displayed in a grid interface. To enable this framework, we propose a new *Bayesian optimization* method called *sequential plane search*, which decomposes the original high-dimensional search problem into a sequence of two-dimensional search (*i.e.*, plane-search) subtasks.

Visual design tasks often involve tuning many design parameters. For example, color grading of a photograph involves many parameters, some of which non-expert users might be unfamiliar with. We propose a novel user-in-the-loop optimization method that allows users to efficiently find an appropriate parameter set by exploring such a high-dimensional design space through much easier two-dimensional search subtasks. This method, called *sequential plane search*, is based on *Bayesian optimization* to keep necessary queries to users as few as possible. To help users respond to *plane-search* queries, we also propose using a gallery-based interface that provides options in the two-dimensional subspace arranged in an adaptive grid view. We call this interactive framework *Sequential Gallery* since users sequentially select the best option from the options provided by the interface. Our experiment with synthetic functions shows that our sequential plane search can find satisfactory solutions in fewer iterations than baselines. We also conducted a preliminary user study, results of which suggest that novices can effectively complete search tasks with Sequential Gallery in a photo-enhancement scenario.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; • **Human-centered computing** → *Human computer interaction (HCI)*.

Authors' addresses: Yuki Koyama, koyama.y@aist.go.jp, National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki, Japan, 305-8568; Issei Sato, sato@k.u-tokyo.ac.jp, The University of Tokyo, Hongo 7-3-1, Bunkyo, Tokyo, Japan, 113-8654; Masataka Goto, m.goto@aist.go.jp, National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki, Japan, 305-8568.

Additional Key Words and Phrases: Visual design exploration, Bayesian optimization, human-in-the-loop optimization.

ACM Reference Format:
Yuki Koyama, Issei Sato, and Masataka Goto. 2020. Sequential Gallery for Interactive Visual Design Optimization. *ACM Trans. Graph.* 39, 4, Article 88 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392444

1 INTRODUCTION

Visual design tasks often involve many parameters that should be carefully adjusted via sliders. The purpose of tweaking these parameters is, for example, to reproduce the desired design in mind or to make the design as aesthetically pleasing as possible. This process is, however, often difficult because the parameters may affect the design in combination and the space of possible parameter configurations is very broad due to the high dimensionality. Moreover, evaluating a certain parameter configuration is also difficult without actually manipulating the slider values and seeing the corresponding visual representation, which thus requires many trials and errors. All this is especially true when users are unfamiliar with the design parameters. For example, photo retouch software has many sliders for color enhancement, including advanced ones such as "shadows (red)" and "highlights (red)" [Adobe 2017a; Instagram, Inc. 2019], which both affect shades of red but in different ways and can produce various effects in combination with other parameters. This complexity requires users to try many slider configurations at the beginning to understand what effects are possible and tweak the slider values little by little alternately in the last fine-tuning step. Similar parametric design scenarios appear in many graphics

applications, including, but not limited to, material appearance design [McAuley et al. 2012; Ngan et al. 2006], procedural modeling and animation [SideFX 2019], rigged character animation [Lewis et al. 2014], personalized fabrication [Shugrina et al. 2015], digital composition [The Foundry Visionmongers Ltd. 2020], and generative design using learned models [Jin et al. 2017; Yumer et al. 2015]. Parametric design is also actively used in other domains, such as architecture and product design [Robert McNeel & Associates 2019].

We propose a user-in-the-loop optimization method, called *sequential plane search*, that allows users to tactically explore such a high-dimensional design space and efficiently find an appropriate parameter set. Its novelty is that it decomposes the original search problem into a sequence of much easier two-dimensional search subtasks, called *plane-search* subtasks. This method is based on *Bayesian optimization* (BO), which is a black-box optimization technique that has recently become popular in the machine learning community [Shahriari et al. 2016]. BO automatically balances *exploration* (*i.e.*, encouragement to visit unobserved regions) and *exploitation* (*i.e.*, encouragement to visit high expectation regions) on the basis of Bayesian inference, by which it tries to minimize the number of necessary observations to find a good solution. By taking advantage of this characteristic in determining plane-search subtasks, our sequential plane search enables users to perform a structured and efficient design exploration guided by a computational strategy.

We also propose an interactive framework, called *Sequential Gallery*, that provides a gallery-based interface to help users effectively perform plane-search subtasks; see Figure 1 for the overview. The interface is called *zoomable grid* and works as follows. It displays a finite set of clickable visual options from the search plane in a grid. At the beginning of a plane-search subtask, it provides a wide variety of options by mapping the entire region of the plane. Then, it lets users "zoom" into a relevant region by selecting the best option among displayed ones. After a few zooms, this subtask finishes with the best option in the search plane. Users repeat this coarse-to-fine selection process (*i.e.*, perform the plane-search subtasks sequentially) until they find a satisfactory design. This gallery-based interface allows users to efficiently grasp possible designs in the subtask without actively manipulating sliders even when they are unfamiliar with the target design space at the beginning of the task.

We demonstrate the generality and applicability of our approach by using two different scenarios: enhancing colors of photographs (12 dimensions) and generating human body shapes using a learned generative model (10 dimensions). To evaluate our sequential plane search, we conducted a simulated experiment using synthetic functions. This experiment revealed that this method could find good solutions in fewer iterations than baseline methods, including the recently proposed *sequential-line-search* method [Koyama et al. 2017], on top of which our method is built. We also conducted a preliminary user study with the photo enhancement scenario, which suggested that novices could effectively complete parameter tweaking tasks and produce satisfactory designs with Sequential Gallery.

In summary, our contribution is twofold.

- We propose a novel method called *sequential plane search* for user-in-the-loop visual design optimization, which requires fewer iterations to find a satisfactory design parameter set than the previous method [Koyama et al. 2017]. We evaluate its performance against baselines through simulated experiments.
- We use a *zoomable grid* interface in combination with our sequential-plane-search method, which enables users to effectively explore the design space and perform the search. We tested this interactive framework, named *Sequential Gallery*, through a small user study.

2 RELATED WORK

2.1 Interfaces for Exploration and Parameter Tweaking

Exploratory design is a design process in which the goal is only loosely specified at the beginning but becomes more and more concrete (or even changes) through exploration [Talton et al. 2009]. To facilitate this process, researchers have investigated gallery-based design interfaces. The seminal work by Marks *et al.* [1997], called *Design Gallery*, visually displays representative design options on a two-dimensional screen by low-dimensional embedding. Followers have demonstrated gallery-based interfaces for effectively exploring complex design spaces of image recoloring [Shapira et al. 2009] and material reflectance design [Ngan et al. 2006] using domain-specific formulations. The Brainstorm tool [Adobe 2017b] aimed at providing users with inspiration, especially at the beginning of exploration, by showing a gallery of randomly sampled designs. Lee *et al.* [2010] reported how a gallery of example designs could help users obtain inspiration to improve design work. Our grid interface follows these gallery-based approaches since our target scenario is similar to exploratory design (though we do not assume that users' preferences drift over time as described in Section 3).

Another common approach to facilitating parameter tweaking is to augment slider interfaces for more effective direct manipulation. *Side Views* [Terry and Mynatt 2002], a mechanism to augment graphical user interface widgets, can augment sliders with design previews. *VisOpt Slider* [Koyama et al. 2014, 2016] visualizes estimated "goodness" of slider values using a colormap so that it gently guides users to relevant regions of the target design space during slider manipulation. Desai *et al.* [2019] further extended this interface and used it for robotic motion design. In contrast to these parameter-wise editing approaches, we take the *what-you-see-is-what-you-get* (WYSIWYG) approach: users can explore the space by interacting with visual representations without caring about raw parameter values, which eliminates the need for being familiar with (or creating a mental model of) the design parameters.

To help users complete plane-search subtasks, we propose using a zoomable grid interface instead of using two sliders with a preview. This interface follows the concept of "zoom-and-pick" (*e.g.*, [Forlines et al. 2005]): users can select a point from the target space precisely by zooming around the relevant region.

2.2 Bayesian Optimization and Preference Learning

Bayesian approaches have been drawing more and more attention in building interactive systems [Kristensson et al. 2019]. Our sequential-plane-search method is based on one such approach, called Bayesian optimization (BO) [Brochu et al. 2010b; Shahriari

et al. 2016], which is a black-box global optimization technique. BO tries to minimize the number of necessary queries to obtain the optimal solution on the basis of Bayesian inference, and thus it is suitable when the target function is expensive to evaluate. For example, BO has been successful in hyperparameter tuning tasks for deep neural networks [Snoek et al. 2012] as each run of the training is expensive.

Preference learning is a category of machine learning which handles preference information (*e.g.*, A is preferred to B) [Chu and Ghahramani 2005; Koyama et al. 2014]. Brochu *et al.* [2007] combined BO and preference learning [Chu and Ghahramani 2005] to enable humans to perform optimization using their preferences. We refer to this human-in-the-loop approach as *preferential Bayesian optimization* (PBO)[1], and our sequential plane search is along this line. Researchers have proposed various interaction forms of PBO [Brochu et al. 2010a; Chong et al. 2019; Koyama et al. 2017]. Among them, our method is particularly inspired by the *sequential-line-search* method [Koyama et al. 2017], which was originally developed for crowdsourcing settings. We review these previous PBO methods in detail in Section 3. Also, our experiment shows that our sequential-plane-search method drastically outperforms the sequential-line-search method. Chong *et al.* [2019] proposed a generative image modeling method that constructs a multi-dimensional search subspace and lets the user explore it, which is similar to ours at the concept level. Whereas their method relies on domain-specific formulations, ours is formulated as a general method so that it is applicable to various domains. Moreover, whereas their method simply uses multiple sliders, our sequential plane search is formulated to be tightly coupled with the gallery-based interface to effectively facilitate users' design exploration.

2.3 Optimization-Based Design

Researchers have proposed various methods and systems for finding desired design parameters by formulating design processes as mathematical optimization problems. In the computer graphics community, this *computational design* approach has been often taken for fabrication-oriented design scenarios [Bächer et al. 2014; Bharaj et al. 2015; Li et al. 2016; Prévost et al. 2013; Umetani et al. 2014]. This approach has also been taken in the human-computer interaction community for optimizing user interface design [Bailly et al. 2013; Dudley et al. 2019; Karrenbauer and Oulasvirta 2014; Todi et al. 2016] and building advanced creativity support tools [Koyama and Goto 2018; O'Donovan et al. 2015]. Our work is along this line, but our target problem involves handling a perceptual objective function as described in the next section.

Interactive evolutionary computation (IEC) methods [Takagi 2001] also aim at involving human evaluation to produce artifacts. In contrast to the IEC-based approach, our sequential plane search mainly aims to minimize the number of necessary queries to a human by incorporating BO techniques. In addition, our sequential plane search is designed so that it can be performed effectively with a gallery-based interface. Design optimization using human evaluation has also been investigated in the mechanical engineering domain [Ren and Papalambros 2011]; our approach is applicable to this domain as long as designs can be visually evaluated through a grid interface.

[1]Note that we use the term PBO in a broader sense than Gonzáles *et al.* [2017].

3 PROBLEM DESCRIPTION AND BACKGROUND

In this section, we first describe our target problem using the terminologies in numerical optimization. Then, we review the existing methods for this problem in detail.

3.1 Problem Description

We consider a parametric visual design task that involves n design parameters. We assume that the parameters are all continuous and thus are typically adjusted by sliders and that their visual effects are also continuous (but not necessarily linear). The primary goal of this task is to search for the slider configuration that produces the "best" design (*i.e.*, the most preferable design for the user who performs the task) among all possible designs achievable via slider manipulation. We can interpret this task as an n-dimensional continuous numerical optimization problem from a mathematical viewpoint; here, we can consider that the user's preference plays the role of the *objective function* of this optimization problem and the user tries to find a maximizer of this function though exploration [Koyama and Igarashi 2018].

Let $\mathcal{X}$ be the n-dimensional design space that the user is going to explore (*i.e.*, the *search space*). We assume $\mathcal{X} = [0, 1]^n$ without loss of generality by applying normalization. An element in this space, $\mathbf{x} \in \mathcal{X}$, represents a slider configuration. The goal of this task is to find the best parameter set, $\mathbf{x}^* \in \mathcal{X}$. The *goodness* of a parameter set is evaluated by a perceptual function, which is called a *goodness function* [Koyama et al. 2014, 2016, 2017], and we represent it as $g : \mathcal{X} \to \mathbb{R}$. Then, we can write the task in the form of an optimization problem as

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} g(\mathbf{x}). \tag{1}$$

However, solving this problem is not easy. The goodness function g is infeasible to formulate as a simple mathematical function since the function g is tightly coupled with the user's preference and even the user him/herself usually does not know what it looks like before exploring the design space $\mathcal{X}$. Thus, we cannot apply standard optimization techniques, which usually expect the objective function to be executable by computers, to this problem. This has motivated researchers to develop human-in-the-loop optimization methods. Note that our target problem is similar to but different from exploratory design (*e.g.*, [Talton et al. 2009]), where users' preferences can change over time during the task. In contrast, we assume that the perceptual function g does not change over time.

Asking a user to provide feedback about goodness requires special care; it is not feasible for the user to reliably and consistently rate visual designs using *absolute* values. That is, we should not query the goodness value $g(\mathbf{x})$ directly for a certain parameter set $\mathbf{x}$. One reason is that such a value can be reliably determined only when the user is familiar with the design space $\mathcal{X}$ and can imagine other possible options in $\mathcal{X}$. However, this is not true in most cases, especially when the user does not have a clear goal vision at the beginning of the design process. Also, as discussed by Brochu *et al.* [2010a], *drift* (*i.e.*, the subjective scale may change over time)

and *anchoring* (*i.e.*, the subjective scale may be dominated by earlier experiences) effects might cause inconsistencies.

Instead, asking about *relative* preference is more promising. The simplest form of such a preference query could be *pairwise comparison*, in which the user is provided with two visual options, say $\mathbf{x}^a$ and $\mathbf{x}^b$, and then chooses his/her preferred one [Tsukida and Gupta 2011]. We can interpret this information as either $g(\mathbf{x}^a) > g(\mathbf{x}^b)$ or $g(\mathbf{x}^a) < g(\mathbf{x}^b)$ depending on the response. The important point is that the user can provide a preference without being familiar with the entire design space or imagining other possible options. Also, we can expect that the drift and anchoring effects will not critically affect responses to preference queries.

Another issue that we need to consider is that human evaluation is much more expensive than typical executable objective functions. For example, a user cannot feasibly be expected to perform a task involving 10,000 subjective evaluations, whereas a computer can do so efficiently. This is our motivation to use BO [Shahriari et al. 2016], which is specifically designed to find a solution with as few queries as possible. In the next subsection, we review previous BO-based methods for preference queries, on which our method is built.

3.2 Preferential Bayesian Optimization Methods

PBO is a variant of BO, which is based on preference (*i.e.*, relative-comparison) queries instead of function-value (*i.e.*, absolute-value) queries. By a preference query, the PBO method obtains feedback in the following form:

$$\mathbf{x}^{\text{chosen}} \succ \{\mathbf{x}^{(j)}\}_{j=1}^{m}, \tag{2}$$

where $\succ$ means that the goodness value at the left-side parameter set is likely to be larger than any goodness values at the right-side m parameter sets. We call this relational information *preference data*. The likelihood of any preference data can be modeled by the Thurstone–Mosteller model [Brochu et al. 2007; Chu and Ghahramani 2005] (for $m = 1$) or Bradley–Terry–Luce model [Koyama et al. 2017; Tsukida and Gupta 2011] (for any number m). Note that we may omit the right-hand side curly bracket when $m = 1$ for simplicity.

Brochu *et al.* [2007] proposed using a *pairwise-comparison* task (also known as *two-alternative forced choice* (2AFC)), in which the user is provided with two visual options and asked to choose one. Suppose that the method has received $i - 1$ query responses from the user and is going to determine the i-th query. Let $\mathbf{x}_i^+ \in \mathcal{X}$ be the "current-best" parameter set among the observed parameter sets and $\mathbf{x}_i^{\text{EI}} \in \mathcal{X}$ be the parameter set that is chosen by BO for the i-th iteration of the optimization. Refer to Appendix A for the exact definitions of these parameter sets, but an intuition for the latter is as follows: the point $\mathbf{x}_i^{\text{EI}}$ is defined as the point that maximizes a criterion called *expected improvement* (EI), which evaluates the effectiveness of a point as the next query (we will explain this in slightly more detail in Section 5). The i-th pairwise-comparison task is then formulated using $\mathbf{x}_i^+$ and $\mathbf{x}_i^{\text{EI}}$. As a result of user feedback, the method obtains new preference data of either

$$\mathbf{x}_i^+ \succ \mathbf{x}_i^{\text{EI}} \quad \text{or} \quad \mathbf{x}_i^{\text{EI}} \succ \mathbf{x}_i^+, \tag{3}$$

depending on the user's choice.

Koyama *et al.* [2017] proposed using a *single-slider-manipulation* task, in which the user is provided with a single slider and a preview widget that is dynamically updated in accordance with the slider value and asked to find the best slider tick position. From a mathematical viewpoint, the user is considered to solve a *line-search* query. Koyama *et al.* proposed constructing the one-dimensional subspace for the i-th iteration, $\mathcal{S}_i$, as

$$\mathcal{S}_i = \{(1-t)\mathbf{x}_i^+ + t\mathbf{x}_i^{\text{EI}} \mid t \in [0,1]\}. \tag{4}$$

Then, the task for the user is described as

$$\mathbf{x}_i^{\text{chosen}} = \operatorname*{arg\,max}_{\mathbf{x} \in \mathcal{S}_i} g(\mathbf{x}), \tag{5}$$

and the user's feedback for this task is interpreted as

$$\mathbf{x}^{\text{chosen}} \succ \{\mathbf{x}^+, \mathbf{x}^{\text{EI}}\}. \tag{6}$$

Note that more points from $\mathcal{S}_i$ can be added into the right-side set, but this may increase the computational cost unnecessarily. A notable advantage of this approach over the pairwise-comparison approach is that a single query can obtain important information on an additional parameter set chosen from the *continuous* subspace, whereas the pairwise-comparison approach can obtain only information on *discrete* parameter sets. This makes the number of iterations necessary to obtain a good solution much smaller.

4 APPROACH OVERVIEW AND USER INTERACTION

We propose a new variant of PBO called *sequential plane search*, in which *plane-search* queries are used for human evaluation to enable the optimization to be even more efficient. We also propose using a *zoomable grid* interface for the user to perform tasks involving plane-search queries. We refer to the entire interactive framework consisting of the sequential-plane-search backend and the zoomable-grid-interface frontend as *Sequential Gallery* (see Figure 1). This framework lets the user sequentially perform plane-search subtasks using the gallery-based interface to solve the target visual design optimization problem.

4.1 Plane-Search Query

We define a plane-search query as follows. Let $\mathcal{P}$ denote a two-dimensional manifold in the n-dimensional design space $\mathcal{X}$ (also simply called a *plane*) and $\mathcal{P}_i$ denote the plane for the i-th step (we will explain how $\mathcal{P}_i$ is parameterized and constructed in Section 5). For each step, our sequential plane search asks the user to search for the best parameter set on the plane $\mathcal{P}_i$. From a mathematical viewpoint, the task for the user is described as

$$\mathbf{x}_i^{\text{chosen}} = \operatorname*{arg\,max}_{\mathbf{x} \in \mathcal{P}_i} g(\mathbf{x}). \tag{7}$$

Note that this is analogous to a line-search query (Equation 5), but our method involves two-dimensional subspaces instead of one-dimensional ones.

4.2 Task Execution with Zoomable Grid Interface

A straightforward way of performing the plane-search task (Equation 7) is to use two sliders mapped to the plane with a preview of the visual representation that can be dynamically updated in accordance with the slider values. However, this approach requires the

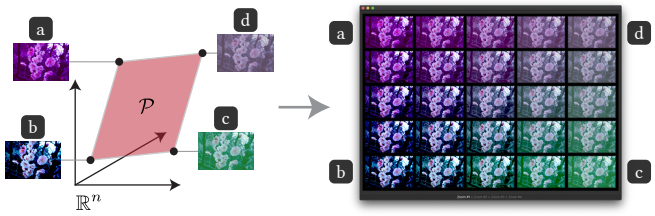

Fig. 2. **Illustration of the mapping** from a search plane $\mathcal{P}$ in the target n-dimensional design space to the zoomable grid interface.

user to actively try many combinations of those two slider values at the beginning of the task to understand the design variation in the current subspace and then to adjust the slider values alternately and little by little at the fine-tuning stage.

We instead use a zoomable grid interface to execute the plane-search task. This interface takes advantage of the fact that the subspace and display are both two-dimensional; it displays clickable visual options in a grid so that their corresponding parameter sets are spatially mapped to the plane (see Figure 2). At the initial coarsest zoom level, the entire plane is mapped to the grid. Once the user clicks an option, the interface goes to the next zoom level with a short zooming animation (around 1.5 seconds). The center element of the new grid is the one chosen in the previous zoom level. In the current implementation, we set the zooming factor as two; that is, the new grid is mapped to an area one-quarter of that in the previous zoom level. Note that extrapolation along the plane is performed when the user selects an option at an edge of the grid. After a certain number of clicks (four in our implementation), this plane-search task finishes, and the method receives the parameter set that the user selected in the finest zoom level. The procedure is illustrated in Figure 3.

This interface involves only discrete selection, and thus we need to consider the effect of discretization. The use of this interface approximates Equation 7 to

$$\mathbf{x}_i^{\text{chosen}} \approx \arg\max_{\mathbf{x}\in\mathcal{G}_i} g(\mathbf{x}), \tag{8}$$

where $\mathcal{G}_i \subset \mathcal{P}_i$ is the finite set of the parameter sets that can be accessed by the zooming procedure. Despite the discretization of $\mathcal{P}_i$, we can still consider that the resulting parameter set is virtually chosen from a continuous space because $\mathcal{G}_i$ contains a large number of samples thanks to the hierarchical zooming procedure.

Thus, in a Sequential Gallery session, the user sequentially solves Equation 8 for $i = 1, 2, \ldots$ until a satisfactory design is found.

5 METHOD: SEQUENTIAL PLANE SEARCH

This section describes the technical aspect of the sequential plane search, which is used inside Sequential Gallery. Note that we omit details that the sequential plane search shares with the previous methods [Brochu et al. 2007; Koyama et al. 2017] and are not necessary to understand its novelty. Those who want to implement the proposed method from scratch should refer to these papers as well.

5.1 Plane Construction Strategy

To make a sequential-plane-search procedure effective, search planes must be constructed appropriately. For the requirements for an effective algorithm, we consider two conditions as the basic design goals.

- The plane should be constructed such that it is likely to minimize the number of iterations necessary for finding a satisfactory parameter set. For this, we propose a new tailored measure (*i.e.*, a new *acquisition function*) to evaluate the effectiveness of search planes (Section 5.3).
- The plane should include the parameter set, $\mathbf{x}^{\text{EI}}$, as in the previous methods [Brochu et al. 2007; Koyama et al. 2017]. This is important from a theoretical viewpoint to ensure that the plane includes the optimal solution in the ideal case that Bayesian inference is perfectly correct. Additionally, this ensures that sequential plane search always performs better than (or at least equivalent to) the previous methods because the subspaces in the previous methods are always a subset of our subspace.

We also consider two additional design goals from the user experience perspective.

- The current-best parameter set, $\mathbf{x}^+$, should always be centered at the plane. This ensures that the position of the current-best design in the zoomable grid interaction is consistent.
- The plane should be *planar* in the mathematical sense. That is, we do not want the plane to be curved or folded in the design space. This ensures that any set of options aligned in a direction in the grid view exhibits linear parameter changes and should help users recognize the presented subspace in many cases. See Figure 4 for an illustration.

5.2 Plane Construction

Our method supposes a plane to always be a rhombus or diamond (*i.e.*, diagonal lines are orthogonal and crossed at their centers) to avoid constructing unnecessarily skewed quadrangles. On the basis of this, we parameterize a plane $\mathcal{P}$ by its center $\mathbf{c} \in \mathbb{R}^n$ and two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ such that the four vertices are represented as $\{\mathbf{c} \pm \mathbf{u}, \mathbf{c} \pm \mathbf{v}\}$ (see Figure 5). We represent this as $\mathcal{P}(\mathbf{c}, \mathbf{u}, \mathbf{v})$.

The basic idea to construct the plane for the i-th step, denoted by $\mathcal{P}_i$, is to solve the following optimization problem:

$$\mathcal{P}_i = \arg\max_{\mathcal{P}} \hat{a}^{\text{EI}}(\mathcal{P}; \mathcal{D}_{i-1}) \tag{9}$$

$$\text{s.t.} \begin{cases} \mathcal{P} \subset \mathcal{X} \\ \mathbf{x}^{\text{EI}} \in \mathcal{P} \end{cases}, \tag{10}$$

where $\hat{a}^{\text{EI}}$ is an acquisition function to evaluate the effectiveness of search planes, which we will define in Equation 14, and $\mathcal{D}_{i-1}$ denotes the preference data obtained by the end of the $(i-1)$-th iteration.

From a practical viewpoint, we solve the above optimization problem in a simplified manner as follows. First, on the basis of the design goal, we always set $\mathbf{c}_i = \mathbf{x}_i^+$. Then, we find the parameter set that maximizes the EI, $\mathbf{x}^{\text{EI}}$, by solving an optimization problem as in the previous methods [Brochu et al. 2007; Koyama et al. 2017];

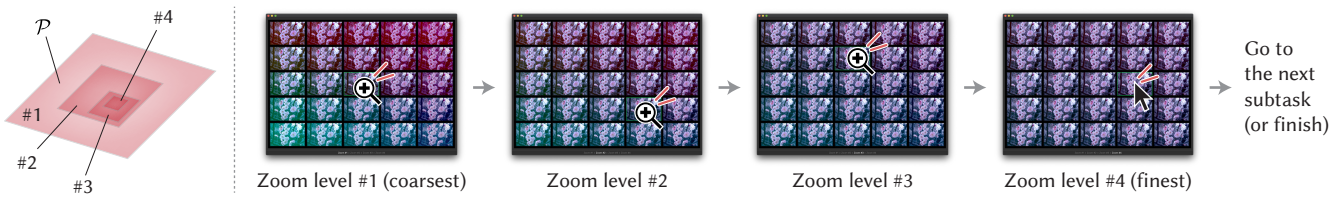

Fig. 3. **Zooming procedure in the zoomable grid interface.** The user clicks the best option displayed in the grid, and then the interface goes to the next finer zoom level. After a certain number of clicks (four in this example), this plane-search subtask ends. If the user wants to continue exploration, our method constructs a new search plane and then asks the user to start another zooming procedure.

Fig. 4. **Importance of planarity of the search subspace.** When the search subspace is planar (Left), the grid exhibits linear parameter changes across all directions. In many cases, this makes it easier for users to grasp the current plane than when the search subspace is not planar (Right).

that is,

$$\mathbf{x}^{\mathrm{EI}} = \arg\max_{\mathbf{x}\in\mathcal{X}} a^{\mathrm{EI}}(\mathbf{x}; \mathcal{D}_{i-1}), \tag{11}$$

where a^{EI} is an acquisition function to evaluate the effectiveness of search points, used in standard BO methods [Snoek et al. 2012]; see the appendix for details. Note that $\hat{a}^{\mathrm{EI}}$ in Equation 9 takes a search plane as an argument whereas a^{EI} takes a search point as an argument. Then, we set $\mathbf{u}_i = \mathbf{x}^{\mathrm{EI}} - \mathbf{x}^{+}$, which ensures that the plane satisfies the second constraint in Equation 10. The remaining one, $\mathbf{v}_i$, is then obtained by solving

$$\mathbf{v}_i = \arg\max_{\mathbf{v}} \hat{a}^{\mathrm{EI}}(\mathcal{P}(\mathbf{c}_i, \mathbf{u}_i, \mathbf{v}); \mathcal{D}_{i-1}) \tag{12}$$

$$\text{s.t.} \begin{cases} \mathbf{u}_i \cdot \mathbf{v} = 0 \\ \mathbf{c}_i \pm \mathbf{v} \in \mathcal{X} \end{cases}, \tag{13}$$

where the equality constraint in Equation 13 is added for ensuring the two vectors are orthogonal and thus the plane is a rhombus.

Although the vertex $\mathbf{x}_i^{\mathrm{EI}} = \mathbf{x}_i^{+} + \mathbf{u}_i$ is always inside the design space $\mathcal{X}$, its opposite-side vertex, $\mathbf{x}_i^{+} - \mathbf{u}_i$, might be outside of $\mathcal{X}$ (though the probability is very small). If we detect this, we correct the vertex position by simply moving it along the diagonal-line direction to the closest bound of $\mathcal{X}$. Note that it is also possible not to correct the vertex position; in this case, the grid interface does not display options whose parameter sets are outside of $\mathcal{X}$.

At the first step of a sequential-plane-search procedure, no preference data is available. For this case, our current implementation simply constructs a fixed-sized square centered at the center of the search space $\mathcal{X}$ with a random direction.

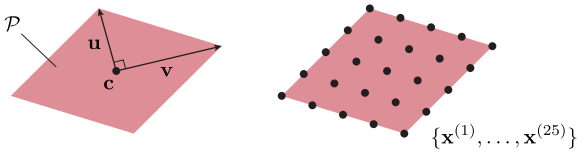

Fig. 5. (Left) **Parameterization** of a plane $\mathcal{P}$. (Right) **Sampling points** used in our implementation for approximating the surface integral (Equation 14).

5.3 Acquisition Function

An *acquisition function* evaluates the effectiveness of a search query and is used to determine the next search query so that it is as effective as possible [Shahriari et al. 2016]. To increase the likelihood of finding a good solution, a search fquery must observe a region that is (1) likely to have a higher value (*i.e.*, exploitation) and (2) less certain because of the lack of observations around the region (*i.e.*, exploration). Several different acquisition functions can be used; among them, we choose the EI since it is commonly chosen [Brochu et al. 2007; Koyama et al. 2017; Snoek et al. 2012] and it balances exploitation and exploration without needing additional hyperparameter adjustment.

In the standard BO setting, in which determining a query is equivalent to finding an appropriate sampling *point*, such a point is defined as a maximizer of the acquisition function [Shahriari et al. 2016]. However, how we should determine a query in our sequential-plane-search setting is not trivial, since the query is determined by finding a *plane*, not a point. The previous sequential-line-search method [Koyama et al. 2017] finds a *line* by simply connecting the current-best point and the maximizer of the acquisition function as explained in Equation 4. Although this simple approach was demonstrated to work well, it only considers the two ends of the line.

To overcome this limitation, we propose an acquisition function tailored for evaluating the effectiveness of a *plane* as a search subspace for the next iteration. We define it as a surface integral of the density of the EI value over the plane:

$$\hat{a}^{\mathrm{EI}}(\mathcal{P}; \mathcal{D}) = \frac{1}{A} \int_{\mathcal{P}} a^{\mathrm{EI}}(\mathbf{x}; \mathcal{D}) dS, \tag{14}$$

where A is the area of the plane $\mathcal{P}$. In practice, we approximate it by a summation at N sampling points on the plane, $\{\mathbf{x}^{(j)}\}_{j=1}^{N}$. More

specifically,

$$\hat{a}^{\mathrm{EI}}(\mathcal{P};\mathcal{D}) \approx \frac{1}{N}\sum_{j=1}^{N} a^{\mathrm{EI}}(\mathbf{x}^{(j)};\mathcal{D}). \tag{15}$$

In the current implementation, we use $N = 25$ points simply sampled in a 5-by-5 lattice pattern as shown in Figure 5 (Right).

5.4 Interpreting Query Response as Preference Data

Given a plane $\mathcal{P}_i$, the user provides the maximizer on the plane, $\mathbf{x}_i^{\mathrm{chosen}}$, as feedback to our method. Then, our method interprets this information as the following preference data:

$$\mathbf{x}_i^{\mathrm{chosen}} \succ \{\mathbf{c}_i, \mathbf{c}_i \pm \mathbf{u}_i, \mathbf{c}_i \pm \mathbf{v}_i\}. \tag{16}$$

Note that more parameter sets from $\mathcal{P}_i$ can be added to the right-side sets. However, to avoid an unnecessarily large computational cost, we use the above five parameter sets as the representatives of the plane.

5.5 Implementation Details

We implemented BO based on a *Gaussian process* [Rasmussen and Williams 2005] prior with the *automatic relevance determination (ARD) Matérn 5/2* kernel, following the suggestion by Snoek *et al.* [Snoek et al. 2012]. We determine the kernel hyperparameters in the same way as the previous paper [Koyama et al. 2017]: we use *maximum a posteriori* (MAP) estimation every time new preference data is added and assume a log-normal distribution $\mathcal{LN}(\mu, \sigma^2)$ as the prior distribution for each kernel hyperparameter. We set $\mu = 0.2$ for the *amplitude* and $\mu = 0.5$ for every *length scale*, and $\sigma^2 = 0.01$ for both; refer to Snoek *et al.* [2012] for the definitions of these hyperparameters. We handle the equality constraint in Equation 13 by simply interpreting it as a soft constraint term:

$$-(\mathbf{u}_i \cdot \mathbf{v})^2, \tag{17}$$

and adding it to the objective function. As a result, the problem can be simply considered as an unconstrained, bounded optimization problem, and the overall objective function can still be differentiable with respect to the unknown, $\mathbf{v}$. Thus, we solve it by using the limited-memory BFGS (L-BFGS) method [Nocedal and Wright 2006]. Note that, to handle the equality constraint more exactly, the *augmented Lagrangian* method [Nocedal and Wright 2006] can be used here. However, we decided to use the simplified approach as we found it sufficiently accurate and also efficient. As the problem of Equations 12 and 13 can have multiple local maxima, we perform the optimization 10 times with random initial solutions in parallel and then use the best-found solution.

6 APPLICATIONS

An advantage of our Sequential Gallery is that it does not rely on any domain-specific formulation and thus can be immediately applied to many visual design domains involving a set of continuous parameters. To demonstrate its generality, we created a photo color enhancement system and a human body shape generation system.

6.1 Photo Color Enhancement (12 Design Parameters)

Photo color enhancement involves editing colors in a photograph to make the photograph more appealing. For this task, in addition to basic parameters (brightness, contrast, and saturation), we implemented an advanced color balance functionality that independently manipulates colors in shadows, midtones, and highlights. Since this advanced color balance is important, popular photo enhancement tools [Adobe 2017a; Instagram, Inc. 2019] usually have it or similar functionality. In total, this system has 12 design parameters to be adjusted. Note that the expressiveness of our implementation is much higher than that of the previous work [Koyama et al. 2017], in which only basic 6 parameters were implemented.

Figure 6 (Left) shows a full sequence of a Sequential Gallery session for photo color enhancement. In this example, the user obtained a satisfactory result after 4 iterations. Figure 6 (Right) shows additional results, and refer to the supplemental video figure for corresponding sequences and a larger view.

6.2 Human Body Shape Design (10 Design Parameters)

It is useful for end-users to be able to easily design human body shapes for various purposes such as those for creating virtual avatars and for visually conveying body shapes in their mind to someone else. To enable this, we used the skinned multi-person linear (SMPL) model [Loper et al. 2015], which is a publicly available pre-trained human shape model based on principal component analysis. We used its first 10-dimensional latent parameters and set the bound of the design space as three times the standard deviation from the mean shape. Figure 7 shows variations of human body shapes achievable by this system.

Compared with photo color enhancement, human body shaping requires comparing more details of visual representations among options, and we noticed that cells in a 5-by-5 grid with a 13-inch display were too small for this application. Thus, we used a 3-by-3 grid for the interface of this application (see Figure 8 (Left)).

One challenge in this design domain is that the latent space of the learned model does not have *semantically meaningful* dimensions, so exploration by direct slider manipulation is difficult. For this, *Body Talk* [Streuber et al. 2016] remapped these parameters to a semantic space by crowdsourcing perceptual annotations. Our method takes a different approach: we handle the design space as a black-box and use the WYSIWYG interface.

An advantage of involving users in the loop is that users can produce designs even from vague pictures in their minds. Following the previous work [Streuber et al. 2016], we created a human body shape from a description in a novel, *The Maltese Falcon*:

> *He was of medium height, solidly built, wide in the shoulders, thick in the neck, with a jovial heavy-jawed red face . . .* [Hammett 1930]

Figure 8 (Center) shows the result obtained after 7 iterations. We also created a body shape of a famous fictional character, *Spider-Man* [Wikipedia 2019], without looking at any references. Figure 8 (Right) shows the result obtained after 10 iterations. Another interesting usage is to reproduce body shapes in photographs [Streuber et al. 2016].

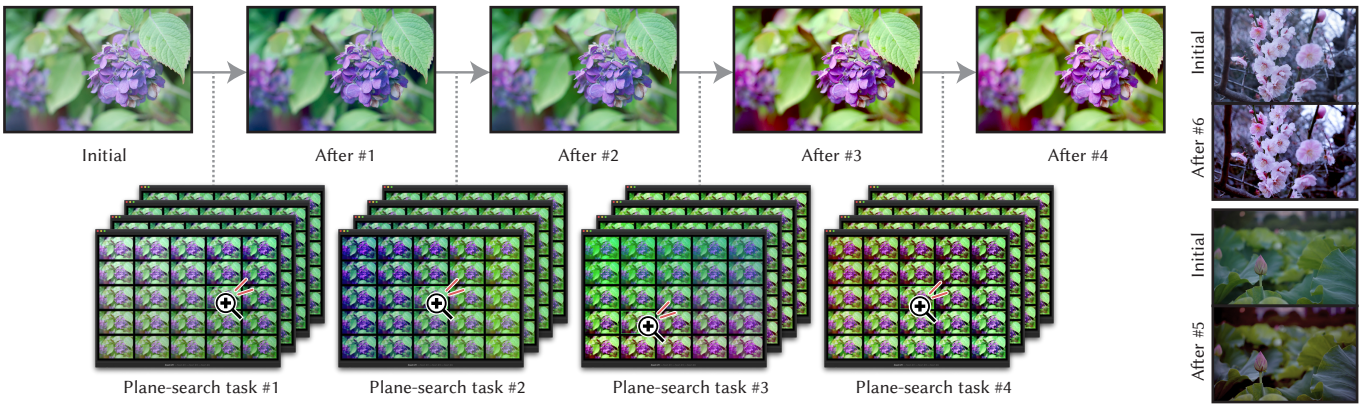

Fig. 6. **Photo color enhancement with Sequential Gallery**, in which 12 design parameters were adjusted. (Left) An entire optimization sequence, in which the user could obtain a satisfactory result after four iterations. (Right) Additional results; refer to the supplemental video figure for details.

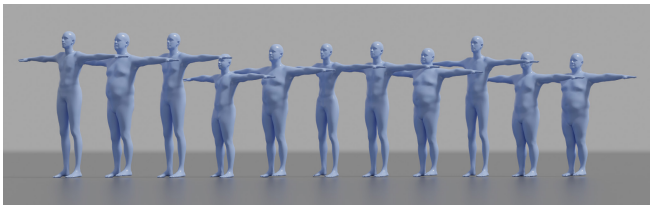

Fig. 7. **Random human body shapes** generated from the SMPL model [Loper et al. 2015]. We used the top-10 dimensions as the design space.

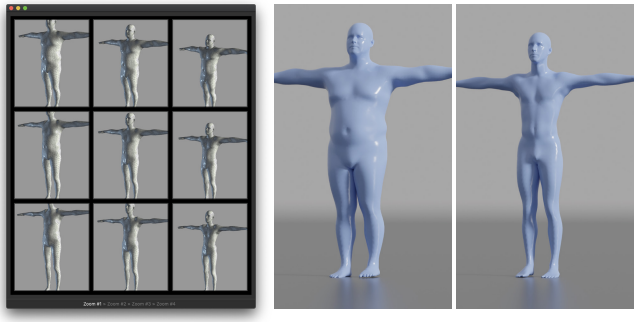

Fig. 8. (Left) **Appearance of the zoomable grid interface in the human body shape design scenario,** where we set the grid resolution as 3 by 3. (Center) **Body shaping from a description of a character in a novel**, *The Maltese Falcon* [Hammett 1930]. (Right) **Body shaping of a famous fictional character**, *Spider-Man* [Wikipedia 2019].

7 EVALUATION

7.1 Experiment Using Synthetic Functions

First, we conducted an experiment using synthetic functions to artificially simulate users' responses on the basis of their preferences. This simulation-based approach is useful to generate a large number of responses to properly understand the behavior of our sequential plane search from a technical viewpoint.

7.1.1 Goals. The specific goals of this experiment were (1) to evaluate the efficiency of the proposed sequential plane search and (2) to validate whether the use of the BO-based plane construction is effective.

7.1.2 Methods to be Compared. We compared three methods:

- **SLS**: The sequential-line-search method [Koyama et al. 2017] that constructs a one-dimensional subspace using BO. Except for the subspace construction, SLS used the same settings as the implementation of our method, such as kernel function choice and hyperparameter handling.
- **SPS (Random)**: The sequential-plane-search method as proposed in this paper, but using a random plane construction. More specifically, a plane is constructed by first setting $\mathbf{c} = \mathbf{x}^+$ like in our proposed method, and then choosing $\mathbf{u}$ and $\mathbf{v}$ randomly such that they have a constant length ($\|\mathbf{u}\| = \|\mathbf{v}\| = 1$) and are orthogonal ($\mathbf{u} \cdot \mathbf{v} = 0$).
- **SPS (Ours)**: The sequential-plane-search method using the proposed BO-based plane construction.

Note that we omitted the BO-based pairwise-comparison method proposed by Brochu *et al.* [2007] because Koyama *et al.* [2017] already reported that it consistently requires much more iterations than SLS.

7.1.3 Synthetic Functions. We used two different synthetic functions. The first consists of a single isotropic (*i.e.*, no covariance) Gaussian kernel:

$$g_1^{\text{synth}}(\mathbf{x}) = \exp\left[-(\mathbf{x}^* - \mathbf{x})^\mathsf{T}(\mathbf{x}^* - \mathbf{x})\right],$$

where we set $\mathbf{x}^* = \begin{bmatrix}0.3 & \cdots & 0.3\end{bmatrix}^\mathsf{T}$. The second is a test function called the *Rosenbrock function* [The SciPy community 2019]. We take its negative to obtain maximizers rather than minimizers since our formulation (Equation 1) uses maximization. Also, we scale it by a factor of 0.25 so that its important region lies within $\mathcal{X} = [0, 1]^n$. As a result, it is defined as

$$g_2^{\text{synth}}(\mathbf{x}) = -\sum_{i=1}^{n-1}\left[100(4x_{i+1} - 16x_i^2)^2 + (1 - 4x_i)^2\right],$$

which has its global minimum at $\begin{bmatrix}0.25 & \cdots & 0.25\end{bmatrix}^{\mathsf{T}}$. The first function is relatively easy to optimize whereas the second is much more difficult because of its complex shape. We measured the performance by using the *optimality gap*: the difference between the optimal function value and the best-found function value [Wang et al. 2016].

7.1.4 Results. Figure 9 shows the results of 50 trials for each method and for each function. We can observe that the SPS methods are consistently far superior to the SLS method in all the settings. This indicates that using plane-search queries instead of line-search queries significantly reduces the number of iterations necessary for reaching good solutions. Also, we can observe that our SPS is consistently superior to the random SPS except for the first few iterations. This indicates the importance of taking the BO-based plane construction approach; it clearly improves the performance after several iterations.

To determine whether these differences in performance are statistically significant, we performed two-sided Mann–Whitney U tests ($\alpha = 0.05$) to compare the three methods. As this test involves multiple comparisons, we adjusted the significance level α in each comparison by the Bonferroni correction. The performance of the SLS method was significantly different from those of the SPS methods in all the functions at all the iteration counts. The performances of the two SPS methods were not significantly different for the first 1, 6, 4, and 10 iterations (for Isotropic Gaussian 5D, Isotropic Gaussian 15D, Rosenbrock 10D, and Rosenbrock 20D, respectively) but became significantly different at the next iteration counts ($p \ll 0.001$ ($f = 0.803$), $p \ll 0.001$ ($f = 0.763$), $p = 0.001$($f = 0.688$), and $p = 0.001$($f = 0.694$), respectively, where f represents the common language effect size); the reason for the lack of significant differences in the first few iterations could be that the observed data points were so sparse given the high dimensionalities that the BO-based SPS performed almost pure exploration like the random method. After those iteration counts, the differences continued to be statistically significant, but there was one exception that the differences were not significant in the Isotropic Gaussian 5D function after 15 iterations; the reason of this exception could be that both methods had sufficiently approached the optimal solution and thus already converged.

7.2 Preliminary User Study

The first evaluation validated the effectiveness of our sequential plane search in goal-driven settings by using synthetic functions. However, the effectiveness of the overall interactive framework remains unevaluated. Thus, we conducted an additional small user study, where we used the photo color enhancement scenario described in Section 6.1.

7.2.1 Goals. The specific goals of this user study were (1) to evaluate whether even novice users who are not necessarily familiar with the target parameters and not likely to have clear mental images at the beginning of the task can perform the plane-search subtasks by the zoomable grid interface and find a satisfactory parameter set, (2) to evaluate whether the interface can facilitate exploration and provide inspiration, and (3) to gather qualitative feedback on our interactive framework.

7.2.2 Method. Five students and one researcher ($P_1, \ldots, P_6$) participated in this user study. This study consisted of four parts: an instruction session, the first photo enhancement task, the second photo enhancement task, and a questionnaire. We prepared three photographs and used one of them for instruction for every participant and the other two for the main tasks. For photo enhancement, we instructed the participants to imagine that they were going to upload the photographs to their Facebook or Instagram accounts and wanted to make the photographs appealing to their friends. We asked them to continue each task for 15 iterations (*i.e.*, a sequence of 15 plane-search subtasks) regardless of whether they were already satisfied with intermediate results or not. For recording purposes, we also asked the participants to push a "satisfied" button on the screen during the iterations of each task when they felt satisfied with the current design (*i.e.*, the option clicked last). We used a 13-inch MacBook Pro with a mouse and maximized the application window size. After the tasks, we asked the participants to fill in a questionnaire consisting of a question about expertise in photo color enhancement, questions arranged on a 7-pt Likert scale with 7 corresponding to "strongly agree," and an optional free comment space.

7.2.3 Results. Five participants (other than P_3) pushed the button to indicate that they were satisfied within 15 iterations in the main tasks. The participant (P_3) who did not push the button in one of the main tasks informally told us that the initial photograph was already satisfactory and it was not clear whether the button should be pushed at the beginning. Overall, these results sugget that the framework could provide satisfactory results. The mean iteration count necessary for finding satisfactory results was 5.36 with $SD = 2.69$ (the task in which the button was not pushed was excluded). One plane-search subtask took 14.8 seconds on average.

In the questionnaire, P_4 described him/herself as an expert and the other five described themselves as novices. For the statement, *"I could find satisfactory photo enhancement results,"* the mean score was 5.83 with $SD = 0.753$. For another statement, *"I could get inspiration for possible enhancement from the grid view,"* the mean score was 6.50 with $SD = 0.548$. Overall, these results support our claims.

We obtained feedback that validates our interface design. P_2 wrote that he/she selected designs *"based on criteria that I didn't have at the beginning"* thanks to the encouragement of our framework to explore designs. Also, P_3 wrote the interface *"is really inspiring in that it makes me want to try out different styles,"* which validates that our interface can facilitate exploration. P_4 appreciated that *"the system proposed some nice photos"* and so that he/she *"could get inspiration."* We also obtained feedback for improvement. P_4, who self-described as an expert, wrote that he/she was familiar with photo enhancement parameters and so they wanted a direct parameter manipulation functionality along with the gallery-based search functionality. P_3 wanted to *"have the original photo alongside to compare with during the enhancement."*

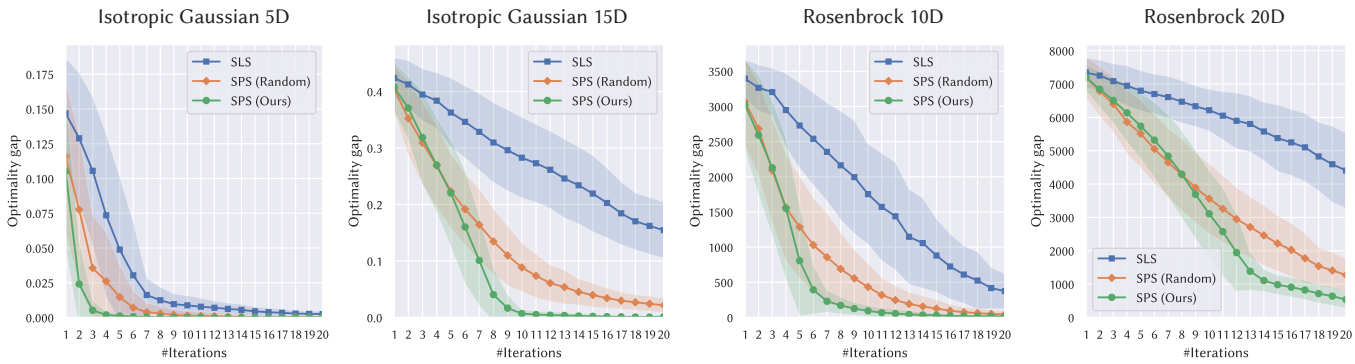

Fig. 9. **Results of the experiment with synthetic functions.** We compare the sequential line search (SLS) [Koyama et al. 2017], the sequential plane search (SPS) using random plane construction, and our SPS using Bayesian optimization-based plane construction. We run 50 trials for each condition. Each plot shows the mean value with the colored regiong showing the standard deviation. Vertical axes represent optimality gaps (lower is better).

8 DISCUSSIONS AND FUTURE WORK

Initial Plane Selection. Our current implementation randomly chooses the initial plane since no preference data is available at the beginning. Another possible initialization method is to insert a step similar to Design Gallery [Marks et al. 1997] before running the sequential-plane-search procedure as follows. First, the design system provides a diverse set of options from the entire design space by embedding them into a two-dimensional widget and then let users choose the best one. Then, the system handles this preference data (*i.e.*, the chosen option is preferred over the other options), $\mathcal{D}_0$, by the Bradley–Terry–Luce model [Tsukida and Gupta 2011] and then calculates the acquisition function (Equation 4) to construct the initial plane. Finally, the system begins sequential plane search with the initial plane and also with the additional preference data $\mathcal{D}_0$. Note that the Design Gallery approach is not for exploring a multi-dimensional space to sequentially refine a solution, though it is good at providing an overview of the entire design space and letting users quickly pick out an initial solution. Our Sequential Gallery can thus complement the Design Gallery approach.

Further Understanding of Interfaces. We adopted the grid interface to help users not only find the best parameter set within the subspace but also easily grasp the landscape of the current subspace and obtain inspiration from the interface. Thus, the goal was not to make each task execution quicker. Nonetheless, it would be interesting to perform comparative studies to investigate both qualitative and quantitative differences between interfaces (a single slider, two sliders, and ours). Also, as our user study was only preliminary, formal studies will need to be conducted with more participants and more practical settings.

Grid Resolution and Zooming Levels. We used fixed values for the grid resolution (*e.g.*, 5-by-5 for the photo enhancement) and the number of zooming levels for each query (*e.g.*, four). We empirically set these values for each application since every design domain has different appropriate values for these variables. Nonetheless, automatically determining appropriate values is important future work. For example, it is worth investigating how to dynamically adjust the number of zooming levels by analyzing the *just-noticeable difference* (JND) of the visual representations in the grid. Another direction is to enable users to interactively adjust these values during tasks to avoid unnecessarily fine zooming levels or grid resolutions.

Plane Construction Strategy. Our strategy always chooses the maximizer of the acquisition function, $\mathbf{x}^{\mathrm{EI}}$, as one of the vertices of the rhombus. While we believe this strategy is reasonable in the sense that we discussed in Section 5.1, it is worth investigating other strategies that provide better exploration-exploitation balancing or theoretical regret bounds [Srinivas et al. 2012]. Also, our strategy enforces the current-best parameter set, $\mathbf{x}^{+}$, to be the center of the next search plane, which is for ensuring consistent interaction when moving from one search plane to another. Another possibility for improving usability would be to introduce a constraint for enhancing continuity between search planes so that users can more easily understand variations in new search planes.

Non-Visual Design. One notable limitation of Sequential Gallery is that it may be ineffective with non-visual designs such as electronic timbre design for a sound synthesizer. This limitation is because the grid interface assumes that designs are visually recognizable at a glance. Another limitation is that our sequential plane search does not handle discrete parameters such as layouts, fonts, or filter types.

Even Higher Dimensionality. BO is known to perform poorly with very high dimensionality (*e.g.*, over 20 dimensions) [Wang et al. 2016]. However, many design tasks involve such high-dimensional design spaces. For example, facial expression modeling for a virtual character usually involves around 50 parameters and sometimes around 1,000 [Lewis et al. 2014]; to conduct these tasks with our sequential plane search, users need to choose a moderate number of relevant parameters beforehand.

Prior Knowledge. Our method assumes that everywhere in the search space is equally good (or bad) at the beginning of the process. To accelerate the search, incorporating prior knowledge about the target design domain would be beneficial. For example, we could build a rough approximation of the goodness function by gathering

preference data by crowdsourcing [Koyama et al. 2014] or implementing common practices in the domain and then use it as a prior of the Bayesian inference.

Time-Varying Preference. Our method handles all preference data equally to infer latent preferences and determine search planes. This assumes that users' preferences do not change over time. In practice, however, this assumption is not always valid (*c.f.*, exploratory design). We could accept such concept drift by simply allowing users to discard accumulated preference data (either entirely or partially) at any time during the search. Incorporating the time-varying property into the BO formulation is also interesting future work.

Acquisition Function Choice. Following previous work, we chose the EI as the criterion to evaluate the effectiveness of a search point and then proposed its extension for evaluating the effectiveness of a search plane (Equation 14). Other acquisition functions (*e.g.*, Gaussian process upper confidence bound [Srinivas et al. 2012]) are applicable for plane search in the same way. Note that there exist acquisition functions specifically tailored for the *discrete* pairwise-comparison setting [González et al. 2017]. However, they are not directly applicable to our problem as our goal is to evaluate the effectiveness of a *continuous* subspace. This is why we proposed an integral-based acquisition function.

Sequential Subspace Search. Line search (Equation 5) and plane search (Equation 7) are one- and two-dimensional, respectively. This suggests a generalization: letting users perform m-dimensional search subtasks sequentially to solve the original n-dimensional problem ($m < n$), which we call *sequential subspace search*. One of our findings is that the convergence with $m = 2$ is drastically faster than that with $m = 1$ (Figure 9) while plane-search tasks remain easy thanks to the zoomable grid interface. We expect that the convergence will become even faster with $m \geq 3$. However, the subtasks would become unreasonably tedious since active trials and errors are inevitable for users to understand m-dimensional subspaces. This is why we chose $m = 2$ and investigated an interface suitable for this choice.

Latent Spaces of Deep Generative Models. Recent advances in deep generative models have demonstrated a new paradigm of design, in which users obtain various designs by specifying parameter sets in latent spaces. However, these latent spaces are difficult for users to explore because they are black-boxes for users and usually intractably high dimensional (*e.g.*, 128 dimensions for geometry generation [Chen and Zhang 2019]), which poses a new problem that needs to be solved by computational techniques and interaction designs in combination. We believe that this work could be an important step in this direction.

9 CONCLUSION

We presented sequential plane search, a novel optimization method for parametric visual design tasks. This method involves the user in the loop of its procedure; it decomposes the target high-dimensional problem into a sequence of much easier two-dimensional subtasks, which can be solved by the user via a simple zoomable grid interface. Our experiment using synthetic functions revealed that using plane-search subtasks was much more effective than using line-search subtasks [Koyama et al. 2017] and that our BO-based plane construction was significantly more effective than a random plane construction. In addition, our user study confirmed that novices could perform our sequential plane search via the zoomable grid interface and find satisfactory results in the photo color enhancement scenario. The study also confirmed that the interface could facilitate exploratory design.

The overall framework, called Sequential Gallery, is quite general and does not rely on any domain-specific formulations, which makes it directly applicable to various problems. Furthermore, it provides a promising future opportunity to adapt our framework to specific problems by incorporating domain-specific considerations into the BO routine (*e.g.*, [Chong et al. 2019]) or the interface design. We plan to make our source codes accessible to encourage this direction.

ACKNOWLEDGMENTS

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan, and JST CREST Grant Number JPMJCR17A1, Japan.

REFERENCES

Adobe. 2017a. Color Balance adjustment in Photoshop. Retrieved September 16, 2019 from https://helpx.adobe.com/photoshop/using/applying-color-balance-adjustment.html#apply_the_color_balance_adjustment.

Adobe. 2017b. Using the Brainstorming tool in After Effects CS6. Retrieved September 12, 2019 from https://helpx.adobe.com/after-effects/atv/cs6-tutorials/brainstorming.html.

Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-It: Optimizing Moment of Inertia for Spinnable Objects. *ACM Trans. Graph.* 33, 4 (July 2014), 96:1–96:10. https://doi.org/10.1145/2601097.2601157

Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. 2013. MenuOptimizer: Interactive Optimization of Menu Systems. In *Proc. UIST '13*. 331–342. https://doi.org/10.1145/2501988.2502024

Gaurav Bharaj, David I. W. Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. 2015. Computational Design of Metallophone Contact Sounds. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 223:1–223:13. https://doi.org/10.1145/2816795.2818108

Eric Brochu, Tyson Brochu, and Nando de Freitas. 2010a. A Bayesian Interactive Optimization Approach to Procedural Animation Design. In *Proc. SCA '10*. 103–112. https://doi.org/10.2312/SCA/SCA10/103-112

Eric Brochu, Vlad M. Cora, and Nando de Freitas. 2010b. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. https://arxiv.org/abs/1012.2599 arXiv:1012.2599.

Eric Brochu, Nando de Freitas, and Abhijeet Ghosh. 2007. Active Preference Learning with Discrete Choice Data. In *Proc. NIPS '07*. 409–416. http://papers.nips.cc/paper/3219-active-preference-learning-with-discrete-choice-data

Zhiqin Chen and Hao Zhang. 2019. Learning Implicit Fields for Generative Shape Modeling. In *Proc. CVPR '19*. 5939–5948. http://openaccess.thecvf.com/content_CVPR_2019/html/Chen_Learning_Implicit_Fields_for_Generative_Shape_Modeling_CVPR_2019_paper.html

Toby Chong, I-Chao Shen, Issei Sato, and Takeo Igarashi. 2019. Interactive Subspace Exploration on Generative Image Modelling. arXiv:1906.09840. https://arxiv.org/abs/1906.09840

Wei Chu and Zoubin Ghahramani. 2005. Preference Learning with Gaussian Processes. In *Proc. ICML '05*. 137–144. https://doi.org/10.1145/1102351.1102369

Ruta Desai, Fraser Anderson, Justin Matejka, Stelian Coros, James McCann, George Fitzmaurice, and Tovi Grossman. 2019. Geppetto: Enabling Semantic Design of Expressive Robot Behaviors. In *Proc. CHI '19*. 369:1–369:14. https://doi.org/10.1145/3290605.3300599

John J. Dudley, Jason T. Jacques, and Per Ola Kristensson. 2019. Crowdsourcing Interface Feature Design with Bayesian Optimization. In *Proc. CHI '19*. 252:1–252:12. https://doi.org/10.1145/3290605.3300482

Clifton Forlines, Ravin Balakrishnan, Paul Beardsley, Jeroen van Baar, and Ramesh Raskar. 2005. Zoom-and-Pick: Facilitating Visual Zooming and Precision Pointing with Interactive Handheld Projectors. In *Proc. UIST '05*. 73–82. https://doi.org/10.1145/1095034.1095046

Javier González, Zhenwen Dai, Andreas C. Damianou, and Neil D. Lawrence. 2017. Preferential Bayesian Optimization. In *Proc. ICML '17*. 1282–1291. http://proceedings.mlr.press/v70/gonzalez17a.html

Dashiell Hammett. 1930. *The Maltese Falcon*. Alfred A. Knopf.

Instagram, Inc. 2019. How do I apply effects to my photo? | Instagram Help Center. Retrieved September 16, 2019 from https://help.instagram.com/1424679781131460.

Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. 2017. Towards the Automatic Anime Characters Creation with Generative Adversarial Networks. arXiv:1708.05509. https://arxiv.org/abs/1708.05509

Andreas Karrenbauer and Antti Oulasvirta. 2014. Improvements to Keyboard Optimization with Integer Programming. In *Proc. UIST '14*. 621–626. https://doi.org/10.1145/2642918.2647382

Yuki Koyama and Masataka Goto. 2018. OptiMo: Optimization-Guided Motion Editing for Keyframe Character Animation. In *Proc. CHI '18*. 161:1–161:12. https://doi.org/10.1145/3173574.3173735

Yuki Koyama and Takeo Igarashi. 2018. Computational Design with Crowds. In *Computational Interaction*, Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes (Eds.). Oxford University Press, Chapter 6, 153–184. https://doi.org/10.1093/oso/9780198799603.003.0007

Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2014. Crowd-Powered Parameter Analysis for Visual Design Exploration. In *Proc. UIST '14*. 65–74. https://doi.org/10.1145/2642918.2647386

Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2016. SelPh: Progressive Learning and Support of Manual Photo Color Enhancement. In *Proc. CHI '16*. 2520–2532. https://doi.org/10.1145/2858036.2858111

Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. 2017. Sequential Line Search for Efficient Visual Design Optimization by Crowds. *ACM Trans. Graph.* 36, 4 (July 2017), 48:1–48:11. https://doi.org/10.1145/3072959.3073598

Per Ola Kristensson, Nikola Banovic, Antti Oulasvirta, and John Williamson. 2019. Computational Interaction with Bayesian Methods. In *Proc. CHI EA '19* (Glasgow, Scotland Uk). C16:1–C16:6. https://doi.org/10.1145/3290607.3298820

Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with Interactive Example Galleries. In *Proc. CHI '10*. 2257–2266. https://doi.org/10.1145/1753326.1753667

J. P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. 2014. Practice and Theory of Blendshape Facial Models. In *Eurographics 2014—State of the Art Reports*. https://doi.org/10.2312/egst.20141042

Dingzeyu Li, David I. W. Levin, Wojciech Matusik, and Changxi Zheng. 2016. Acoustic Voxels: Computational Optimization of Modular Acoustic Filters. *ACM Trans. Graph.* 35, 4 (July 2016), 88:1–88:12. https://doi.org/10.1145/2897824.2925960

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-person Linear Model. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 248:1–248:16. https://doi.org/10.1145/2816795.2818013

Joe Marks, Brad Andalman, Paul A. Beardsley, William T. Freeman, Sarah F. Gibson, Jessica K. Hodgins, Thomas Kang, Brian Mirtich, Hanspeter Pfister, Wheeler Ruml, Kathy Ryall, Joshua E. Seims, and Stuart M. Shieber. 1997. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *Proc. SIGGRAPH '97*. 389–400. https://doi.org/10.1145/258734.258887

Stephen McAuley, Stephen Hill, Naty Hoffman, Yoshiharu Gotanda, Brian Smits, Brent Burley, and Adam Martinez. 2012. Practical Physically-based Shading in Film and Game Production. In *ACM SIGGRAPH 2012 Courses*. 10:1–10:7. https://doi.org/10.1145/2343483.2343493

Addy Ngan, Frédo Durand, and Wojciech Matusik. 2006. Image-driven Navigation of Analytical BRDF Models. In *Proc. EGSR '06*. 399–407. https://doi.org/10.2312/EGWR/EGSR06/399-407

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization* (2nd ed.). Springer Science+Business Media. https://doi.org/10.1007/978-0-387-40065-5

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proc. CHI '15*. 1221–1224. https://doi.org/10.1145/2702123.2702149

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Trans. Graph.* 32, 4 (July 2013), 81:1–81:10. https://doi.org/10.1145/2461912.2461957

Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. The MIT Press. http://www.gaussianprocess.org/gpml/

Yi Ren and Panos Y. Papalambros. 2011. A Design Preference Elicitation Query as an Optimization Process. *Journal of Mechanical Design* 133, 11 (November 2011), 111004:1–111004:11. https://doi.org/10.1115/1.4005104

Robert McNeel & Associates. 2019. Grasshopper - New in Rhino 6. Retrieved September 16, 2019 from https://www.rhino3d.com/6/new/grasshopper.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (January 2016), 148–175. https://doi.org/10.1109/JPROC.2015.2494218

Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. 2009. Image Appearance Exploration by Model-Based Navigation. *Comput. Graph. Forum* 28, 2 (2009), 629–638. https://doi.org/10.1111/j.1467-8659.2009.01403.x

Maria Shugrina, Ariel Shamir, and Wojciech Matusik. 2015. Fab Forms: Customizable Objects for Fabrication with Validity and Geometry Caching. *ACM Trans. Graph.* 34, 4 (July 2015), 100:1–100:12. https://doi.org/10.1145/2766994

SideFX. 2019. Houdini | 3D Procedural Software for Film, TV & Gamedev | SideFX. Retrieved September 16, 2019 from https://www.sidefx.com/products/houdini/.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Proc. NIPS '12*. 2951–2959. https://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2012. Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting. *IEEE Trans. Inf. Theory* 58, 5 (May 2012), 3250–3265. https://doi.org/10.1109/TIT.2011.2182033

Stephan Streuber, M. Alejandra Quiros-Ramirez, Matthew Q. Hill, Carina A. Hahn, Silvia Zuffi, Alice O'Toole, and Michael J. Black. 2016. Body Talk: Crowdshaping Realistic 3D Avatars with Words. *ACM Trans. Graph.* 35, 4 (July 2016), 54:1–54:14. https://doi.org/10.1145/2897824.2925981

Hideyuki Takagi. 2001. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. *Proc. IEEE* 89, 9 (Sep. 2001), 1275–1296. https://doi.org/10.1109/5.949485

Jerry O. Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. 2009. Exploratory Modeling with Collaborative Design Spaces. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 167:1–167:10. https://doi.org/10.1145/1618452.1618513

Michael Terry and Elizabeth D. Mynatt. 2002. Side Views: Persistent, On-demand Previews for Open-ended Tasks. In *Proc. UIST '02*. 71–80. https://doi.org/10.1145/571985.571996

The Foundry Visionmongers Ltd. 2020. Nuke, NukeX & Nuke Studio | VFX Software | Foundry. Retrieved January 20, 2020 from https://www.foundry.com/products/nuke.

The SciPy community. 2019. scipy.optimize.rosen — SciPy v1.3.0 Reference Guide. Retrieved September 13, 2019 from https://docs.scipy.org/doc/scipy-1.3.0/reference/generated/scipy.optimize.rosen.html.

Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proc. DIS '16*. 543–555. https://doi.org/10.1145/2901790.2901817

Kristi Tsukida and Maya R. Gupta. 2011. *How to Analyze Paired Comparison Data*. Technical Report UWEETR-2011-0004. University of Washington, Department of Electrical Engineering. https://vannevar.ece.uw.edu/techsite/papers/refer/UWEETR-2011-0004.html

Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: Interactive Design and Optimization of Free-Formed Free-Flight Model Airplanes. *ACM Trans. Graph.* 33, 4 (July 2014), 65:1–65:10. https://doi.org/10.1145/2601097.2601129

Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando De Freitas. 2016. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *J. Artif. Intell. Res.* 55 (February 2016), 361–387. https://doi.org/10.1613/jair.4806

Wikipedia. 2019. Spider-Man - Wikipedia. Retrieved September 20, 2019 from https://en.wikipedia.org/wiki/Spider-Man.

Mehmet Ersin Yumer, Paul Asente, Radomir Mech, and Levent Burak Kara. 2015. Procedural Modeling Using Autoencoder Networks. In *Proc. UIST '15*. 109–118. https://doi.org/10.1145/2807442.2807448

A DEFINITIONS OF $\mathbf{x}^+$ AND $\mathbf{x}^{\mathrm{EI}}$

Let $\mu : \mathcal{X} \to \mathbb{R}$ and $\sigma^2 : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be functions that return the mean and variance, respectively, of the posterior distribution of the goodness value inferred by Gaussian process regression using the currently available data. We denote by $\mathbf{x}^+ \in \mathcal{X}$ the "current-best" parameter set among the observed parameter sets. A possible criterion for the "best" here is the μ value [Brochu et al. 2007; Koyama et al. 2017] and we followed this approach in the evaluation. Another possibility is to simply use the user's last selection as $\mathbf{x}^+$, which is actually more stable sometimes. We denote by $\mathbf{x}^{\mathrm{EI}} \in \mathcal{X}$ the parameter set that maximizes the EI value, which is calculated by

$$a^{\mathrm{EI}}(\mathbf{x}) = \sigma(\mathbf{x})(\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x}))) + \phi(\gamma(\mathbf{x}))) \text{ if } \sigma(\mathbf{x}) > 0 \text{ else } 0, \quad (18)$$

where $\gamma(\mathbf{x}) = (\mu(\mathbf{x}^+) - \mu(\mathbf{x}))/\sigma(\mathbf{x})$, ϕ is the standard normal function, and Φ is the cumulative distribution function of ϕ [Snoek et al. 2012].