Crowd-Powered Parameter Analysis for Visual Design Exploration

Yuki Koyama **Daisuke Sakamoto** **Takeo Igarashi**
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan
koyama@is.s.u-tokyo.ac.jp, {d.sakamoto, takeo}@acm.org

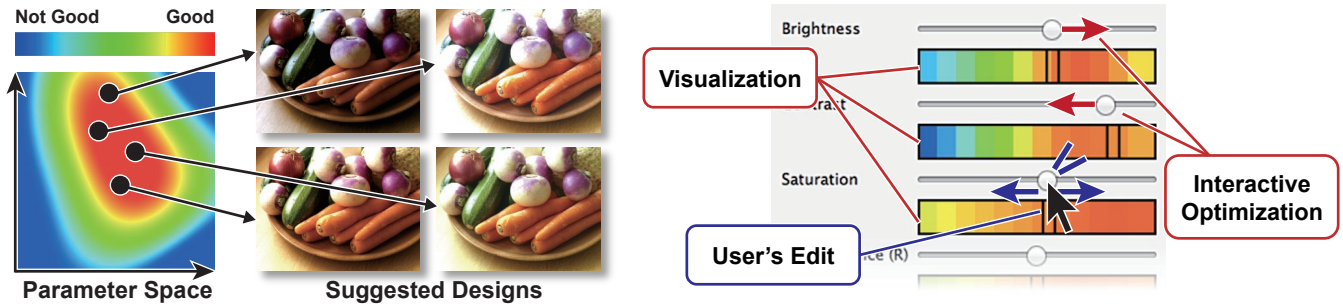

Figure 1. Two interfaces for visual design exploration that are realized by our analysis technique. (Left) *Smart Suggestion*: The user can obtain appropriate parameter sets as suggestions. (Right) *VisOpt Slider*: The user can adjust each parameter effectively by the visualization (*Vis*) near the slider and the optimization (*Opt*) that gently guides the current parameters to the optimal direction.

ABSTRACT

Parameter tweaking is one of the fundamental tasks in the editing of visual digital contents, such as correcting photo color or executing blendshape facial expression control. A problem with parameter tweaking is that it often requires much time and effort to explore a high-dimensional parameter space. We present a new technique to analyze such high-dimensional parameter space to obtain a distribution of human preference. Our method uses crowdsourcing to gather pairwise comparisons between various parameter sets. As a result of analysis, the user obtains a *goodness function* that computes the goodness value of a given parameter set. This goodness function enables two interfaces for exploration: *Smart Suggestion*, which provides suggestions of preferable parameter sets, and *VisOpt Slider*, which interactively visualizes the distribution of goodness values on sliders and gently optimizes slider values while the user is editing. We created four applications with different design parameter spaces. As a result, the system could facilitate the user's design exploration.

Author Keywords

Crowd-powered parameter analysis; parameter tweaking; design exploration; human computation;

http://dx.doi.org/10.1145/2642918.2647386

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces - Graphical user interfaces; I.3.6. Computer Graphics: Methodology and Techniques - Interaction techniques

INTRODUCTION

Exploring possible visual designs by tweaking parameters is a common practice when designing digital content. For example, if we want to clean up a photo for use at the top of a Web page, we first open the photo in Photoshop or GIMP, and then adjust the parameters—brightness, contrast, saturation, etc.—to explore which combination of parameters provides the best result. Similarly, when game developers create a 3D game scene, even for a simple scene, they have to adjust tens of hundreds of parameters—light positions, character positions, expressions, poses, colors of objects, values of shader parameters, etc.—that define the visual quality of the game.

These parameter-tweaking tasks can essentially be considered part of an iterative optimization process. In other words, we iteratively adjust the parameters of the visual design, and then evaluate the results. This process is often tedious and time-consuming, especially when there are a lot of parameters to adjust, and the high dimensionality of the space of the parameters makes it exponentially difficult to explore all possible ones. Moreover, unfamiliar parameters can be puzzling because we cannot predict their effect.

We present a method to facilitate parameter tweaking for visual design exploration. In the proposed approach, we use crowdsourcing to gather pairwise comparisons between various parameter sets. Users first specify an input parameter set and goodness criteria for a target application. The system

then analyzes the parameter space using crowd workers and obtains a *goodness function*, which is a function that takes a parameter set as an input and computes how good it is. We then use the goodness function for two interfaces: *Smart Suggestion* and *VisOpt Slider*. Smart Suggestion is an interface that provides users with appropriate parameter set choices as suggestions. VisOpt Slider is an extension of the conventional slider component, in which the distribution of goodness values is directly visualized on sliders, and it can interactively update the visualization on the basis of the currently chosen slider values. The slider values are interactively and continuously optimized to a better direction as the user is editing. These two interfaces are complementary; e.g., a user can first obtain a reasonable starting point by Smart Suggestion, and then interactively tunes it up using VisOpt Slider.

In this work we offer two primary contributions:

- A method of assisting design exploration using a goodness function derived from analyzing design parameter space via on-demand crowdsourcing.
- Two specific user interfaces, Smart Suggestion and VisOpt Slider, which utilize the derived goodness function and enable users to effectively explore the parameter space.

To evaluate our method, we performed experiments with four applications: photo color correction (6 parameters), camera and light control in a 3D scene (8 parameters), shader (8 parameters), and blendshape facial expression (53 parameters). We checked the quality of analysis, and conducted an informal user study of our interfaces.

RELATED WORK

Crowd-Powered Analysis

Crowdsourcing is an emerging paradigm whereby we can stably employ a large number of temporary workers (crowd) through the Internet. From the viewpoint of *human computation* [15], we can consider this new platform a resource of *human processors* whereby we can easily obtain large amounts of human-processed data for analysis on demand, which previously had required temporally- and monetary-high cost.

One of the notable benefits of using crowds as human processors is that we can exploit popular human perception. Gingold et al. [6] introduced a technique for applying the perception of a crowd to the analysis of 2D images. Chaudhuri et al. [5] proposed AttribIt, an interface using semantic attributes for designing visual content, where they utilized crowds to learn the semantic attributes of design components. We also utilize crowds as human processors for analysis. Our goal is to analyze a design space that has been formed by continuous design parameters, and then use the results of the analysis for interfaces.

Secord et al. [19] analyzed the viewpoint preference when viewing 3D shapes, through a large perceptual user study conducted on a crowdsourcing platform. Their approach was to create a generalized model to predict the viewpoint preference of (even) unknown content while using domain-specific features. Reinecke et al. [17, 16] presented a perceptual model of aesthetics for web design learned by crowdsourced study, where they utilized the features specific to web design for their computational model. In contrast to these works, our goal is to deal with arbitrary design spaces and arbitrary criteria, and to provide a model tailored for specific content.

Visual Parameter Tweaking

Parameter tweaking is a common process in various design situations, and researchers have been tackling how to best facilitate it for a long time. One of the most successful approaches is inverse design [29, 13, 24], in which the system automatically computes the best-suited parameters from a user-specified final desired look, considering it as an inverse problem. However, this is not useful if the user does not have any clear goal and rather wants to explore a range of possible looks. Furthermore, this approach is often difficult to generalize because solving an inverse problem requires knowledge of the heuristics of each application. In contrast, our goals are to achieve an algorithm that can be used with many applications and to facilitate user's exploration.

Showing design options as suggestions or landmarks is an effective way to facilitate design exploration. Many previous works have investigated this approach [11, 12, 7, 20, 9], and several tools, such as Photoshop's Variations interface[1] and AfterEffects' BrainStorming interface[2], include this function. One of our own interfaces, Smart Suggestion, follows this concept. However, unlike the above works/tools, Smart Suggestion considers the goodness values estimated using crowd preferences when generating suggestions. In addition to showing suggestions, VisOpt Sliders provide informed exploration (guidance) around a selected sample.

Side Views [26] and Parallel Paths [27] enable fast "visits" of possible designs by showing visual designs along with GUI widgets such as sliders, then the user can efficiently evaluate which parameter is good or not good. In contrast, our approach enables users to avoid visits of meaningless designs by showing goodness values instead of "raw" designs. Scented Widgets [28] also shows "raw" data that the user should evaluate according to their situations. In contrast, our VisOpt Slider shows the distribution of explore-worthy parameters that directly guides user's exploration.

As an interface that utilizes the preferences of many people, Talton et al. [23] presented a parametric modeling tool that "knows" a good design space (called *collaborative design space*) thanks to the collaboration of many people. The main difference is that our work is designed for "on-demand crowdsourcing," or human computation, for specific target problem instance, while their work is designed for collaborative exploration. Their system collects final modeling results by dedicated users, so it takes time (e.g. a year). On the other hand, our system only requires the workers to "compare" instances as microtasks, so it is much easier to get many contributors (workers). Xu et al. [30] also proposed a design system using collective knowledge. Their purpose of using collective knowledge is to get feedback for a certain design, while ours is to analyze the entire design space.

[1] `http://www.adobe.com/products/photoshop/`
[2] `http://www.adobe.com/products/aftereffects/`

CROWD-POWERED PARAMETER ANALYSIS

Overview of the Process

Our approach employs a crowdsourcing platform to analyze parameters to support the tweaking of visual design parameters. The definition of a *parameter set* in this paper is an n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$ that consists of n continuous parameters $x_i \in [0, 1]$ for $i = 1, \ldots, n$. Note that we do not deal with discrete parameters such as font or layout selection. We assume that the final form of a design task can be visualized as a 2D image I and that given a parameter set $\mathbf{x}$, the design software deterministically provides an image, which we describe as $I(\mathbf{x})$. In other words, our target design tasks can be completely parameterized by n parameters. This assumption is true in many cases, especially in the final steps of actual design processes. For example, when 3D game developers finish their implementation of game logic, the final step of the game creation could be the parameter tuning of the game scene visuals, such as the positions of game objects, the lighting conditions, the camera pose, and the shader parameters.

The goal of the analysis is to obtain a continuous scalar-valued function, or a goodness function, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that maps a parameter set to its estimated goodness; that is, to obtain a function f that takes a parameter set $\mathbf{x}$ as input and computes an estimated goodness value $y = f(\mathbf{x})$ as output. We define goodness as a continuous value from 0 to 1, where 1 is the most preferable and 0 is the least.

Our process to obtain a goodness function consists of four steps, as shown in Figure 2. First, the system generates sampling points on the high-dimensional parameter space (displayed here as 2-dimensional space). Second, using crowdsourcing, the system gathers pairwise comparisons of these sampling points. Third, on the basis of this comparisons, we analyze the goodness value for each (discrete) sampling point. Fourth, the system interpolates the goodness values and obtains a continuous scalar function, i.e., a goodness function, as a result of the analysis.

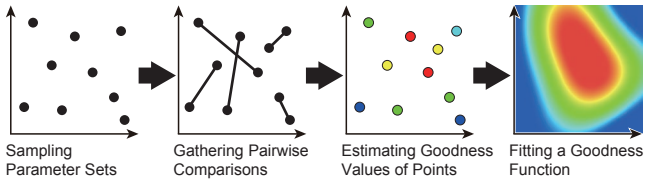

Figure 2. Overview of our algorithm of analysis.

Our method can be considered as one of the learning-from-crowd problems [25, 19, 5], but the problem we are addressing here requires some special treatments. First, the data that we deal with is a set of noisy relative scores between two sampling points, not absolute goodness values. In addition, the derived function is required to be very fast to compute its value for a given parameter set to realize our interaction techniques, and also required to be able to represent a highly nonlinear distribution to deal with various design spaces. In our understanding, any existing algorithm has not solved this specific problem yet, and cannot be applied to our problem without significant extensions.

Sampling Parameter Sets

First, the system samples M parameter sets $\mathbf{x}_1, \ldots, \mathbf{x}_M$ from the parameter space for the later process of crowdsourcing. To do this, we simply choose a random uniform sampling; the system randomly picks a parameter set up from the parameter space and then computes this process M times.

There might be smarter ways to sample parameter sets to achieve more effective crowdsourcing. For example, Secord et al. [19] took a *nearby sampling* approach, in which the system samples pairs of parameter sets that are located *near* to each other. However, nearby sampling requires empirical knowledge of the parameter space, which prohibits us from using it because we want to deal with various types of parameter spaces. Tamuz et al. [25] proposed a sophisticated adaptive sampling method for a crowd-powered analysis; however, their method cannot be directly applied for our problem because it is not designed to derive a continuous scalar function as output. It might be possible to further improve our results by using their adaptive sampling method, but the random sampling works sufficiently well for our requirements.

Gathering Pairwise Comparisons by Crowdsourcing

The next step is to gather information on the goodness of each sampling point. Because the goodness value is essentially evaluated by human preference, we use a human computation technique based on crowdsourcing. A possible naïve approach is to ask crowd workers to provide the absolute goodness values on sampling points directly; however, this is impractical because the concept of goodness is too abstract and the measure scale depends on the individual.

Thus, we take a *pairwise comparison* approach [19, 6, 5] in which crowd workers are shown a pair of designs and asked to choose the best one. As a result, relative scores instead of absolute ones are obtained. Unlike most of the previous approaches, we use the 5-pt Likert scale (1 to 5) to rate a design pair, where 1 means one design of the two is definitely better, 5 means the other design is definitely better, and 3 means neutral. Our algorithm welcomes the *neutral* score, as it is considered a constraint that the two parameter sets have the same or nearly the same goodness values.

Let P be a set of pairs of indices $\{(1, 2), \ldots, (M-1, M)\}$. For each $(i, j) \in P$, the system generates two images $I(\mathbf{x}_i), I(\mathbf{x}_j)$, shows the pair side by side to a crowd worker, and asks him/her to rate its relative score. As a whole, the system gathers $M/2$ relative scores, and each image is shown and rated once.

The instruction of the micro-task for crowd workers is important in terms of the quality of obtained data. Because the purposes and contexts of using our system differ depending on the situation, we prepare a template of instruction for users rather than providing a fixed instruction. The template that we used in our experimentation is:

> *Which of the two images of [noun] is more [adjective]? For example, [clause]. Please choose the most appro-*

priate one from the 5 options below.

In accordance with the purpose and the content, the user gives a noun, an adjective (such as "good" or "natural"), and a clause (which explains a concrete scenario to instruct crowd workers more effectively). After this instruction, two images and five options appear. These options are linked to the Likert scale; e.g., "*the left image is definitely more [adjective] than the right image*" is for option 1, and the complete opposite is option 5. Option 3 is "*these two images are equally [adjective], or are equally not [adjective].*" Figure 3 shows a screen capture of the micro task that was actually used in our experiments.

Figure 3. An example of the tasks used in our experiments. We took a pairwise comparison approach.

To control the quality of the data, we take the *duplicate* approach, as do several of the previous works [19, 6, 5]. We first asked 10 questions and then asked 10 more identical questions but with the arrangement of the two images flipped. If the answer of a question contradicted the duplicated correspondent, we simply discarded the answer. We also discarded all answers from a particular crowd worker if more than half the answers contradicted. This algorithm cannot detect if all 20 answers have been (lazily) rated as "3", and so the system checked such cases and discarded them if so.

In the remainder of this paper, we represent P' as a subset of P, each of which have passed the quality check, M' as the number of sampling points that have passed the quality check, and $Q = \{q_1, \ldots, q_{M'}\}$ as a set of indices of sampling points that have passed the quality check.

Estimating Goodness Values of Sampling Points

Given the relative scores, the next goal is to compute the absolute goodness values $\mathbf{y} = (y_{q_1}, \ldots, y_{q_{M'}})$ of the sampling points $\mathbf{x}_{q_1}, \ldots, \mathbf{x}_{q_{M'}}$. Sýkora et al. [22] dealt with a similar problem where, given the relative orders of pairs of points, the system estimates the entire consistent orders of all points. The difference between their target problem and ours is the existence of inconsistent relative orders; in our case, the solution that satisfies all the relative orders does not generally exist because the data is from unreliable crowds and is based on human preference. Gingold et al. [6] presented a robust algorithm for this problem, but it can only handle relative orders of adjacent areas, and their 2D-based algorithm cannot be extended to high-dimensional parameter space due to its high computational cost.

Thus, we present a new formulation for this problem. We consider a constraint based on the relative scores as a cost function

$$E_{\text{relative}}(\mathbf{y}) = \sum_{(i,j)\in P'} \|y_i - y_j - d_{i,j}\|^2, \tag{1}$$

where $d_{i,j}$ is an offset distance between the i-th and j-th goodness values, defined as

$$d_{i,j} = \begin{cases} 1 & (\text{relative score} = 1) \\ 0.5 & (\text{relative score} = 2) \\ 0 & (\text{relative score} = 3) \\ -0.5 & (\text{relative score} = 4) \\ -1 & (\text{relative score} = 5) \end{cases}. \tag{2}$$

Considering only the relative constraint results in a disconnected, jagged distribution of goodness values (Figure 4, left). This is undesirable because we are handling continuous parameter space, and thus we expect the goodness function to also be continuous (Figure 4, right) and smooth in most cases. We therefore add a constraint based on the assumption of continuity:

$$E_{\text{continuous}}(\mathbf{y}) = \sum_{i\in Q} \left\| y_i - \frac{1}{N_i} \sum_{j\in N_i} y_j \right\|^2, \tag{3}$$

where N_i stands for the neighborhoods of the i-th sampling points, which are defined by 20-nearests in Euclidean distance. Note that this continuous constraint is popular in Laplacian-related techniques such as smoothing [21] and thus is beneficial for reducing data noise.

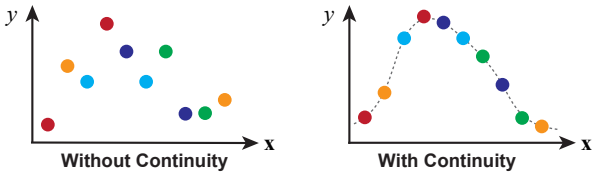

Figure 4. Constraint of continuity. This constraint ensures that the estimated goodness values are continuously distributed.

Taking these two constraints into consideration, the system solves the following minimization:

$$\min_{\mathbf{y}} \left(E_{\text{relative}}(\mathbf{y}) + \omega E_{\text{continuous}}(\mathbf{y}) \right), \tag{4}$$

where $\omega > 0$ is a parameter that defines the balance of these two constraints. In all experiments introduced in this paper, we set $\omega = 5.0$. We can solve this minimization problem as a simple linear algebraic equation. See the appendix for specific details. After solving this minimization problem, we linearly normalize the solution so that the maximum goodness value is 1 and the minimum is 0.

Fitting a Goodness Function

Our goal is now to obtain a continuous goodness function from the obtained goodness values of the sampling points. The fitted function needs to be efficient enough for use in real-time visualization and optimization for user interfaces,

so we chose the Radial Basis Function (RBF) interpolation technique for fitting, which is often used to smoothly interpolate known values at various points (RBF centers) and thus create a continuous function [3]. We use the parameter sets $\mathbf{x}_{q_1}, \ldots, \mathbf{x}_{q_{M'}}$ as the locations of the RBF centers and the obtained goodness values $y_{q_1}, \ldots, y_{q_{M'}}$ as the target values. That is, we represent the goodness function f as $f(\mathbf{x}) = \sum_{i \in Q} w_i \phi(\|\mathbf{x} - \mathbf{x}_i\|)$, where $\phi(\cdot)$ is a radial basis function, and $\mathbf{w} = (w_{q_1}, \ldots, w_{q_{M'}})$ are the RBF weights that we are going to compute. We used $\phi(r) = r$ as the basis function but found the Gaussian also works well and does not result in any significant differences.

The *exact* RBF interpolation scheme is not robust for dense, noisy data. We therefore add a regularization term when computing RBF weights [3]. Specifically, we solve the following minimization problem to obtain $\mathbf{w}$:

$$\min_{\mathbf{w}} \sum_{i \in Q} \left\| \sum_{j \in Q} w_j \phi\left(\|\mathbf{x}_i - \mathbf{x}_j\|\right) - y_i \right\|^2 + \lambda \|\mathbf{w}\|^2 \qquad (5)$$

where $\lambda \geq 0$ is the parameter for controlling regularization. We set $\lambda = 0.1$ in this work. We found that this regularization was able to avoid overfitting with our data. Optionally, we compute the reduction of RBF centers [4] to improve computational efficiency.

USER INTERFACE

We use the pararmeter analysis results to create two types of user interface to facilitate parameter tweaking tasks and design exploration.

Smart Suggestion

Smart Suggestion is a function that generates nine parameter sets having relatively high goodness values and displays the corresponding designs as suggestions. This interface facilitates design exploration by giving users a good starting point to find a better parameter set for the visual design.

Our current implementation takes a simple approach to generate quality suggestions: the system generates 2,000 parameter sets randomly and then selects the 9-best parameter sets according to their goodness values. This simple algorithm interactively provides suggestions of an adequate quality, which enables users to re-generate suggestions quickly enough for interactive use if none of the suggestions satisfy them. To generate higher-quality suggestions, other techniques such as *diversity optimization* [1] or the *dispersion* technique [11] could be useful. However, we feel that it is better to avoid spending too much time generating optimal suggestions, and instead focus on interactively providing suggestions of sufficient quality.

VisOpt Slider

The VisOpt Slider (Figure 1, left) displays bars with a "visualization" (*Vis*) of the results of parameter analysis. The distribution of goodness values is directly visualized on each slider using color mapping, which navigates the user to tweak parameters. When the "optimization" (*Opt*) is turned on, the parameters are automatically and interactively optimized while the user is dragging a slider. For example, a user drags a slider, and then the other sliders automatically move to better points corresponding to the user's operation.

A visual bar shows the distribution of goodness values along the line that passes the current parameter point and whose direction is the same as the parameter's axis (Figure 5). Note that when the user modifies a certain parameter, the visualizations of the other parameters will change dynamically. This visualization continuously tells the user which parameter should be modified and how much the parameter should be modified, thus achieving higher quality designs. This helps the user not only to find better parameter sets quickly but also to explore the design space effectively without visiting *bad* designs.

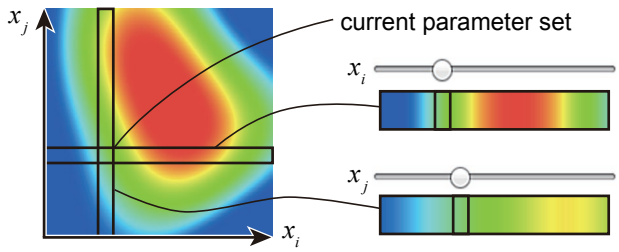

Figure 5. The bars visualize the distribution of the goodness values along each slider.

The purpose of the optimization is not to find the optimal parameter set automatically but rather to assist manual interactive exploration on the part of the user by gently guiding the current parameter set to a reasonable direction. To achieve this, we use the gradient descent method with a fixed number of iterations, following Prévost et al.'s work [14]. In this method, the system first computes the gradient of the goodness function at the current parameter set and then slightly modifies the set to move in the gradient direction instead of instantly jumping to the optimal solution. This enables the user to gradually approach to the optimal solution by continuously scribbling the slider knob back and forth. We numerically compute the gradient using forward differentiation. This process could be written as

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x}), \qquad (6)$$

where $\alpha > 0$ is a small value parameter that defines the strength of optimization. Note that, in order to avoid conflicting with the user's editing, the system does not update the currently edited parameter. This process is performed every time the system receives a slider-value-changed event. In our experimentations, we typically set $\alpha = 0.005$.

APPLICATIONS

We evaluated our method by using it with four example applications. All four applications are from different domains to

demonstrate that the proposed method is suitable for general (not domain-specific) use. We chose CrowdFlower[3] as the platform of crowdsourcing. Table 1 lists a summary of the crowdsourcing statistics. We analyzed each parameter space using all the *valid* comparisons we obtained as a result of crowdsourcing.

Color Correction

When correcting digital photos, users have to tweak many parameters, including unintuitive ones such as saturation, which is sometimes quite difficult for novices. To facilitate this task, we selected six popular parameters for this application: brightness, contrast, saturation, and color balance (color of Red, Green, and Blue (RGB)). For this experiment, we chose a photo of vegetables (Figure 6, left) from a photo sharing service (morgueFile[4]) and asked crowd workers to choose which version would be better to use in a magazine or product advertisement.

Figure 6. (Left) The input image (goodness = 0.72). (Middle and Right) Typical suggested images (goodness = 0.98, 0.95).

Figure 6 (middle and right) shows typical images suggested by Smart Suggestion. Examples of VisOpt Slider visualizations with typical parameter sets are shown in Figure 7. These visualizations provide assorted useful information; for example, the photo at left needs to have a higher contrast, the center photo can be improved by making the brightness slightly higher and the red balance slightly lower, and the right photo is already good and does not require any dramatic improvements.

Figure 7. Visualizations for typical parameter sets in the application of color correction.

Camera and Light Control

Secord et al. [19] presented a general model of view direction goodness for 3D models; however, that model is limited to the view direction and does not consider any other factors. We feel that good views will change according to other conditions such as perspective and lighting. In this scenario, we chose a camera and light control task in a simple 3D scene containing a 3D model, a perspective camera, and a point light. There are eight parameters to tweak, including camera position ($-3.0 < xyz < 3.0$), camera field of view, light position ($-3.0 < xyz < 3.0$), and intensity of light. We used the dragon model (almost symmetric), and the bunny model (asymmetric). The rotation of the camera is automatically set so that it always looks at the center of the model. We asked crowd workers to choose the better one with the same instruction.

The results indicate a highly non-linear relationship between camera and light parameters. When the camera comes to the left side (camera.$z < 0.0$, Figure 8, middle) from the right side (camera.$z > 0.0$, Figure 8, left) of the dragon model, the visualization tells us that we should also move the light to the left side (light.$z < 0.0$) so that the model is adequately lit. Figure 8 right shows the views using the bunny model, where we applied exactly the same parameter sets as to the left one of Figure 8. We found that the visualizations between the case of the dragon and the bunny are quite different even when the parameter sets are equivalent, which indicates that different models have different goodness functions. This suggests that our approach of "final look-aware crowdsourcing" is key.

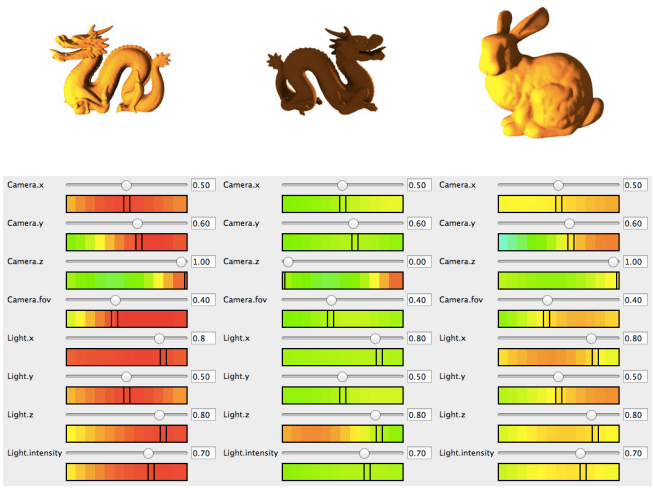

Figure 8. Designs and visualizations of the camera and light application. (Left) A parameter set is applied to the dragon scene, whose camera.z (the 3rd slider) is set to the maximum value. (Middle) Another parameter set whose values are the same as the left one except that camera.z is modified to the minimum value. (Right) The same parameter set as the left one is applied to the bunny scene.

Shader

As Talton et al. [23] discussed, shading is quite difficult for novices to understand and tweak and even game developers and computer graphics researchers struggle with it at times. The problem is that shaders often have unintuitive parameters that affect the final look in a way that is difficult for casual users to predict, such as "Fresnel Reflection" or "Metallics".

[3] `http://www.crowdflower.com`
[4] `http://www.morguefile.com`

Table 1. Statistics of crowdsourcing in our experiments. Adjective shows the words we used for the instructions. For each application, we ordered a fixed number of tasks at once, each of which contains 10 comparisons and 10 duplicated comparisons for quality control. We typically chose 200 for the number of tasks, but 600 for the facial expression application because the parameter space is quite high-dimensional. We paid a fixed amount of money for each task regardless of the quality; for example, in the case of the color correction, we paid $200 \times 0.02 = 4.00$ USD for workers in total. The number of valid comparisons indicates $|P'|$, which is the number of comparisons that passed quality check. We allowed crowd workers to perform two or more tasks if they want, so we also report the number of unique workers. Completion time denotes the passed time from the point when the first task is started to the point when the final task is finished.

	#Parameters	Adjective	#Tasks	Pay [USD / task]	#Valid comparisons	#Unique workers	Completion time [min]
Color Correction	6	good	200	0.02	1095	45	30
Camera and Light (Dragon)	8	good	200	0.02	1010	26	25
Camera and Light (Bunny)	8	good	200	0.02	922	35	40
Shader (Kitchen)	8	realistic	200	0.02	907	46	34
Shader (Window Seat)	8	realistic	200	0.02	875	47	24
Facial Expression	53	natural	600	0.02	1771	57	55

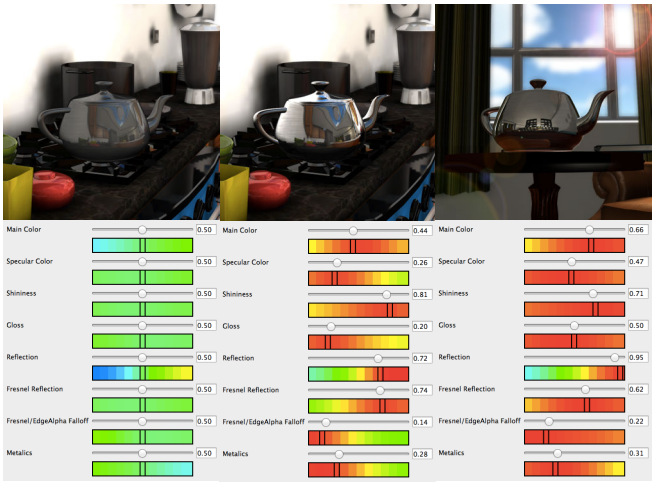

Figure 9. Typical designs and their parameter sets in the shader application. (Left and Middle) Kitchen scene. (Right) Window seat scene.

Furthermore, the final rendered images are different depending on many different factors such as lighting conditions and the geometric features of the 3D models. Thus, users have to explore the best parameters for each scene, which is time-consuming. For this experiment, we used a shader for photo-realistic metals provided in a popular shader package, Hard Surface Shaders Free[5] (Unity Asset Store). We applied this shader to a teapot model. We chose eight parameters from the shader parameters, and asked crowd workers to choose the one that was the most realistic as a stainless steel teapot. We experimented with two different scenes: a kitchen scene with standard lighting and a window seat scene with backlighting.

Figure 9 shows typical parameter sets with their visualizations. From these visualizations, we can learn, without any trial-and-error, that the "Reflection" parameter (the fifth parameter in Figure 9) performs the most important role in this application. We observed that the distributions of goodness values are different from each scene; for example, a parameter set whose goodness value was 0.67 in the kitchen scene had a goodness value of 0.93 in the window seat scene.

Blendshape Facial Expression

Blendshape is a standard approach to control the facial expressions of virtual characters [10], where a face model has a number of predefined continuous parameters and its expression is controlled by artists/designers by tweaking the parameters. Such direct control is made tedious because of the number of parameters. Furthermore, as discussed by Lewis et al. [10], the space of *valid* expressions is actually quite small in most cases, which means that extremely careful tweaking is required to ensure natural, unbroken expressions. In this scenario, we used a head model whose blendshape is defined with 53 parameters (48 based on FACS [18], and 5 for jaw control). To analyze the *validity* of this parameter space, we asked crowd workers to choose the better natural expression.

Figure 10 shows typical designs and their goodness values. We believe that the goodness function is successfully constructed and gives reasonable values for even this high-dimensional application. Unlike random suggestions (typical goodness values are 0.3–0.6), where most of expressions are broken, Smart Suggestion provides nice starting points (around 0.7) with a high enough quality to inspire users. The optimization also works well with this large number of parameters to avoid wasting time with *invalid* expressions. However, we found that the heatmap visualization was often useless because each visualization bar tends to show a similar color along its axis in this large parameter space.

EVALUATION

Quality of Analysis

Convergence with respect to Number of Comparisons

It is important to understand how many comparisons are necessary for each application. If there are too many comparisons, it takes extra time and money, and if there are too few, the obtained goodness function might be missing the important feature of the parameter space. Here, we discuss the convergence behavior graphs (Figure 11) of the shader and the facial expression applications, where the goodness values of 30 randomly selected parameter sets are plotted for each graph. Beginning with 20 comparisons, we computed the goodness function using the limited number of comparisons, and then repeatedly increased the number of used comparisons by 10. Invalid comparisons that did not pass the quality control were excluded.

As shown in the graph of the shader application, the curves are relatively stable with more than 400 comparisons (roughly more than 2 USD), though they do not converge completely. Since our goal is to assist users with interactive exploration

[5] `https://www.assetstore.unity3d.com/#/content/729`

Figure 10. Typical designs of facial expressions controlled by 53 blendshape parameters. The values shown in each picture are the goodness values computed by our method.

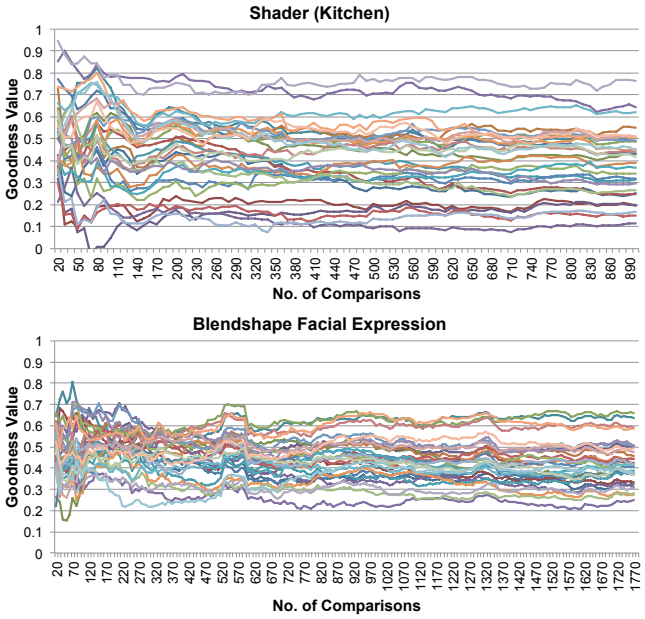

Figure 11. Convergence behavior with respect to the number of comparisons. Each curve shows the transition of the goodness value on a randomly selected parameter set.

rather than to find the exact optimal parameter set automatically, we believe our analysis is convergent enough for the purpose. We observed similar convergence graphs in the other applications, and even, somewhat surprisingly, it is true in the facial expression with 53 parameters, where more than 900 comparisons are required to be stable.

Adequacy of Estimated Goodness Functions

To check the adequacy of the estimated goodness functions, we applied a cross-validation test to the acquired data set for the color correction and the facial expression applications. We randomly selected 100 samples from the valid comparisons P' obtained via crowdsourcing as a testing set, which we denote P'_{testing}. Then, we trained the goodness function f by using the rest of the valid comparisons $P'_{\text{training}} = P' - P'_{\text{testing}}$ as a training set (the size of training data set is 995 for the color correction and 1671 for the facial expression). If the estimated goodness function f obtained from P'_{training} is adequate, it should be able to predict the distribution of the scores of the comparisons in P'_{testing}. That is, as for a paired comparison in the testing set $(i, j) \in P'_{\text{testing}}$, the difference between the estimated goodness values $f(\mathbf{x}_i) - f(\mathbf{x}_j)$

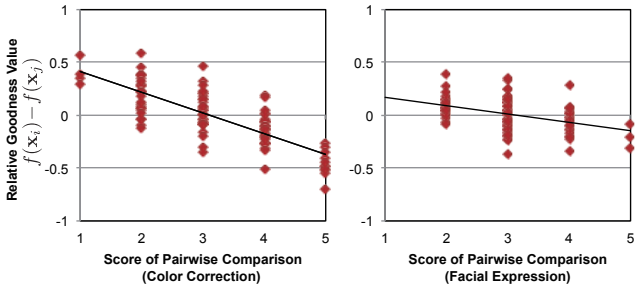

Figure 12. Relationship between the scores and their computed relative goodness values. Black lines are the fitted lines.

should be correlated to the score (1–5) given to the pair by a crowd worker. We show such plots in Figure 12. Although the data shows large variance due to diversity of human preference and noise from careless workers, we can still observe a tendency that a crowd worker rated i higher than j when the goodness value for i ($f(\mathbf{x}_i)$) is higher than j ($f(\mathbf{x}_i)$). This indicates that the trained goodness function f successfully predicted the relative goodness values of paired comparisons that do not appear in the training set.

User Study of the Interfaces

We conducted an informal user study to determine whether Smart Suggestion and VisOpt Slider interfaces are useful for novice users or not. Four computer science students participated in this study. We asked them to explore the design spaces and find the best parameter sets on the four applications, where they can use both random suggestions/standard sliders and Smart Suggestion/VisOpt Slider. After the trial, we asked the participants to fill questionnaires.

We found all the participants could use our proposed interfaces effectively. The score of System Usability Scale (SUS) [2] was 77.5 on average ($SD = 11.6$), which could be considered as "good". All the participants preferred to use the Smart Suggestion rather than random suggestions (6.75 on average with 7-pt Likert scale; 7 is the best), and also preferred VisOpt Slider rather than standard sliders (6.25 on average). Overall, a combination of the Smart Suggestion and VisOpt Slider interfaces is preferable to use (6.25 on average).

DISCUSSION

The results of the user study suggests that the Smart Suggest and VisOpt Slider interfaces provide users with a good starting point for designing visuals. For examples, modeling a facial expression is usually quite difficult, but users successfully

used Smart Suggestion to select a sample expression with which to start the precise control, thus reducing the amount of time needed to create a visual. In the case of the camera and lighting control application, the VisOpt slider helped users find better lighting parameters automatically simply by controlling one parameter for the camera. In another case of photo color correction, there are some automatic correction algorithms; however, such algorithms typically cannot consider the context of photos. For example, a user might want to make a photograph "sad" or "happy" by changing its color. This is not possible with the automatic method.

The facial expression application, for example, takes an hour and costs more than 10 USD for the crowdsourcing, which might reduce user motivation to use our interface. However, there are certain cases where the task is very critical (e.g. preparing a visual for a large-budget advertising campaign), and knowing general-public preference (customer opinion) via crowdsourcing is worth investing time and money. Another possible scenario for the facial expression application is that the 3D modeler can sell his/her 3D face model with the result of the parameter analysis. In this case, end users can use our user interface without running costly analysis by themselves.

Other Possible Representations of Goodness Function

In this work, we chose a non-parametric representation of RBF function as a representation of goodness function. The role of this function is similar to the ranking function proposed by Chaudhuri et al. [5], where they represented it as a simple weighted sum of the input vector. Secord et al. [19] proposed some representations for goodness function based on linear-K models and a quadratic model. We believe that simply applying their techniques to our problem cannot capture the non-linear distribution of goodness shown in our visualization, and also the non-linear relationships such as the relative positions of the camera and the light. Note that Secord et al. [19] also proposed a non-parametric representation based on the *K-nearest-neighbors* model in their paper. However, it cannot be used for our purpose since it cannot compute a goodness value for a single parameter set but only the preference in a pair of parameter sets; furthermore, it is computationally too expensive to use for interfaces.

Limitation

One of the most significant limitations of our approach is that the user has to make the instruction for crowdsourcing. This was not evaluated in the user study. Of course, we prepared an instruction template, but this cannot be automated.

At present, we cannot obtain the parameter analysis results in real-time because we use the crowdsourcing platform. From the statistics of the crowdsourcing we conducted, roughly half an hour seems required for around 8 dimensional design tasks and one hour for around 50 dimensional tasks. This might be a problem for users who cannot wait and want to obtain the results immediately. (Hopefully this process will become faster as the popularity of crowdsourcing expands.) In addition to the cost in terms of time, monetary cost is also a problem, as some users might feel the price of our analysis is too high. We paid 4 USD in total to crowd workers for the shader application, which we believe to be a reasonable price.

Design Implication

Learning the *personal* preference of the users (not the crowd) [20] could be an interesting future direction. Because human preferences differ depending upon the individual and the culture, considering clusters of crowds [8] could improve the quality of goodness functions. To lower the monetary and timing costs of crowdsourcing, it could be helpful to sample parameter sets in a progressive way and dynamically change the tasks [25]. Web design and presentation slide design contain many discrete parameters such as fonts so currently they are out of our scope, but considering such discrete parameters is an important direction to take in our future work. Similar to AttribIt [5], we feel that considering two or more types of goodness criteria for a design task could provide more useful interfaces. In our current approach, every different image task requires re-crowdsourcing, and this limits the scalability; to improve this, we are thinking of combining multiple crowdsourced results or reusing previous ones. Generating a small number of new sliders by using dimensionality reduction techniques is also a potential focus of our future work.

CONCLUSION

We presented a technique to analyze design parameter space via crowdsourcing. Analysis results are used to obtain a goodness function that maps a given parameter set to its goodness value. On the basis of this goodness function, we created two interfaces to facilitate design exploration: Smart Suggestion and VisOpt Slider. We applied our technique to various visual design tasks. As validation of our approach, we checked the quality of analysis and evaluated our interfaces through an informal user study.

ACKNOWLEDGMENTS

Yuki Koyama is funded by JSPS research fellowship. This work was supported by JSPS KAKENHI Grant Number 26·8574, 26240027. The face model is provided by faceshift AG under CC BY 3.0. The dragon and bunny models are provided by The Stanford 3D Scanning Repository.

REFERENCES

1. Agrawal, S., Shen, S., and van de Panne, M. Diverse motion variations for physics-based character animation. In *Proc. SCA '13*, ACM (2013), 37–44.
2. Bangor, A., Kortum, P. T., and Miller, J. T. An empirical evaluation of the system usability scale. *Int. J. Hum.-Comput. Int. 24*, 6 (2008), 574–594.
3. Bishop, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1995.
4. Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R. Reconstruction and representation of 3d objects with radial basis functions. In *Proc. SIGGRAPH '01*, ACM (2001), 67–76.
5. Chaudhuri, S., Kalogerakis, E., Giguere, S., and Funkhouser, T. Attribit: Content creation with semantic attributes. In *Proc. UIST '13*, ACM (2013), 193–202.

6. Gingold, Y., Shamir, A., and Cohen-Or, D. Micro perceptual human computation for visual tasks. *ACM Trans. Graph. 31*, 5 (2012), 119:1–119:12.

7. Hartmann, B., Yu, L., Allison, A., Yang, Y., and Klemmer, S. R. Design as exploration: Creating interface alternatives through parallel authoring and runtime tuning. In *Proc. UIST '08*, ACM (2008), 91–100.

8. Kajino, H., Tsuboi, Y., and Kashima, H. Clustering crowds. In *Proc. AAAI '13* (2013).

9. Lee, B., Srivastava, S., Kumar, R., Brafman, R., and Klemmer, S. R. Designing with interactive example galleries. In *Proc. CHI '10*, ACM (2010), 2257–2266.

10. Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F., and Deng, Z. Practice and theory of blendshape facial models. In *EG 2014 - STARs*, Eurographics Association (2014), 199–218.

11. Marks, J., Andalman, B., Beardsley, P. A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J., and Shieber, S. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proc. SIGGRAPH '97*, ACM Press/Addison-Wesley Publishing Co. (1997), 389–400.

12. Ngan, A., Durand, F., and Matusik, W. Image-driven navigation of analytical brdf models. In *Proc. EGSR '06*, Eurographics Association (2006), 399–407.

13. Okabe, M., Matsushita, Y., Shen, L., and Igarashi, T. Illumination brush: Interactive design of all-frequency lighting. In *Proc. Pacific Graphics '07* (2007), 171–180.

14. Prévost, R., Whiting, E., Lefebvre, S., and Sorkine-Hornung, O. Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph. 32*, 4 (2013), 81:1–81:10.

15. Quinn, A. J., and Bederson, B. B. Human computation: A survey and taxonomy of a growing field. In *Proc. CHI '11*, ACM (2011), 1403–1412.

16. Reinecke, K., and Gajos, K. Z. Quantifying visual preferences around the world. In *Proc. CHI '14*, ACM (2014), 11–20.

17. Reinecke, K., Yeh, T., Miratrix, L., Mardiko, R., Zhao, Y., Liu, J., and Gajos, K. Z. Predicting users' first impressions of website aesthetics with a quantification of perceived visual complexity and colorfulness. In *Proc. CHI '13*, ACM (2013), 2049–2058.

18. Sagar, M. Facial performance capture and expressive translation for king kong. In *ACM SIGGRAPH 2006 Sketches*, ACM (2006).

19. Secord, A., Lu, J., Finkelstein, A., Singh, M., and Nealen, A. Perceptual models of viewpoint preference. *ACM Trans. Graph. 30*, 5 (2011), 109:1–109:12.

20. Shapira, L., Shamir, A., and Cohen-Or, D. Image appearance exploration by model-based navigation. *Comput. Graph. Forum 28*, 2 (2009), 629–638.

21. Sorkine, O. Differential representations for mesh processing. *Comput. Graph. Forum 25*, 4 (2006), 789–807.

22. Sýkora, D., Sedlacek, D., Jinchao, S., Dingliana, J., and Collins, S. Adding depth to cartoons using sparse depth (in)equalities. *Comput. Graph. Forum 29*, 2 (2010), 615–623.

23. Talton, J. O., Gibson, D., Yang, L., Hanrahan, P., and Koltun, V. Exploratory modeling with collaborative design spaces. *ACM Trans. Graph. 28*, 5 (2009), 167:1–167:10.

24. Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., and Koltun, V. Metropolis procedural modeling. *ACM Trans. Graph. 30*, 2 (Apr. 2011), 11:1–11:14.

25. Tamuz, O., Liu, C., Belongie, S., Shamir, O., and Kalai, A. Adaptively learning the crowd kernel. In *Proc. ICML '11*, ACM (2011), 673–680.

26. Terry, M., and Mynatt, E. D. Side views: Persistent, on-demand previews for open-ended tasks. In *Proc. UIST '02*, ACM (2002), 71–80.

27. Terry, M., Mynatt, E. D., Nakakoji, K., and Yamamoto, Y. Variation in element and action: Supporting simultaneous development of alternative solutions. In *Proc. CHI '04*, ACM (2004), 711–718.

28. Willett, W., Heer, J., and Agrawala, M. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Trans. Visual. Comput. Graphics 13*, 6 (2007), 1129–1136.

29. Witkin, A., and Kass, M. Spacetime constraints. *SIGGRAPH Comput. Graph. 22*, 4 (1988), 159–168.

30. Xu, A., Huang, S.-W., and Bailey, B. Voyant: Generating structured feedback on visual designs using a crowd of non-experts. In *Proc. CSCW '14*, ACM (2014), 1433–1444.

APPENDIX

Solving Minimization as a Linear Algebra

In our analysis, the system solves a minimization problem described in (4), which can be solved as a simple linear algebra. Using matrices, we can deform the formula into

$$\min_{\mathbf{y}} \left(\|\mathbf{A}\mathbf{y} - \mathbf{d}\|^2 + \omega \|\mathbf{L}\mathbf{y}\|^2 \right), \tag{7}$$

where the first term corresponds to the relative constraint, and the second term corresponds to the continuous constraint. $\mathbf{A} \in \mathbb{R}^{M' \times M'}$ is defined as $A_{i,i} = 1, A_{i,j} = -1$ if $(i,j) \in P'$ and otherwise 0, $\mathbf{d} \in \mathbb{R}^{M'}$ is defined as $d_i = d_{i,j}$ if $(i,j) \in P'$ and otherwise 0, and $\mathbf{L}$ is the Laplacian matrix. By setting the derivative with respect to $\mathbf{y}$ to zero, we can obtain the following linear equation:

$$\left(\mathbf{A}^T\mathbf{A} + \omega \mathbf{L}^T\mathbf{L}\right)\mathbf{y} = \mathbf{A}^T\mathbf{d}. \tag{8}$$

This provides the optimal solution of the minimization problem and could easily be solved by standard techniques such as Cholesky decomposition.