Computational Design Driven by Visual Aesthetic Preference
(見た目の審美的好ましさを指標とするコンピュテーショナルデザイン)

by

Yuki Koyama
小山 裕己

A Doctor Thesis
博士論文

Submitted to
the Graduate School of the University of Tokyo
on December 9, 2016
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Information Science and
Technology
in Computer Science

Thesis Supervisor: Takeo Igarashi　五十嵐 健夫
Professor of Computer Science

ABSTRACT

Tweaking parameter settings is one of the most fundamental tasks in many design domains, including 2-dimensional graphic design and 3-dimensional product design. The purpose of such parameter tweaking tasks is to maximize the quality of designed objects based on some criteria. Especially in visual design domains, *aesthetic preference*, *i.e.,* how aesthetically preferable the designed object looks, is often used as the criterion. For example, in photo color enhancement, a designer tweaks several parameters such as "brightness" or "contrast", to make the color of the target photograph aesthetically best. However, aesthetic preference is tied with human perception, and thus it is difficult to mathematically quantify this criterion using simple rules or equations.

In this thesis, we seek computational design support methods for parameter tweaking tasks in which aesthetic preference is used as a criterion. First, we investigate methods for estimating a preference distribution in the target design space using computational techniques. The estimated preference distribution can be then used for facilitating manual design exploration. Second, we investigate methods for directly finding the best parameter set from the target design space using computational techniques. These two approaches collect necessary data about human preference exploiting two difference sources: *crowdsourced human computation* and *editing history*. Crowdsourced human computation techniques provide "general" preference data generated by a large number of undefined crowds in an on-demand manner, while editing history of a single target user provides "personal" preference data of the user.

Specifically, we propose the following three computational design methods.

1. The first method estimates a preference distribution in the target design space using crowdsourced human computation. The estimated preference distribution is then used in a novel design interface to facilitate manual design exploration.

2. The second method also estimates a preference distribution and uses it for facilitating manual exploration, but the estimation is based on the editing history of the target user. Along with this history-based preference estimation technique, we also propose a workflow to effectively gather and utilize the user's editing history in practical scenarios.

3. The third method directly searches the target design space for the best parameter set that maximizes aesthetic preference, without requiring the user of this method to manually tweak parameters. This is enabled by constructing an optimization framework using crowdsourced human computation.

We evaluated these three methods mainly in the scenario of photo color enhancement, but we also demonstrate applications to other various design domains, including lighting design for 3-dimensional computer graphics and facial expression modeling of a virtual avatar. The results showed that every proposed method was able to computationally handle either general or personal aesthetic preference, and worked in meaningful ways to support design activities. We envision that these methods and the lessons learned through this study will become fundamentals of future research on computational design methods for more complex design scenarios beyond parameter tweaking.

論文要旨

デザインパラメタを調整することは，二次元のグラフィックデザインや三次元プロダクトデザインなどを含む多くのデザイン領域において，最も基礎的な作業の一つである．そのようなパラメタ調整の目的は，デザインの成果物の質を何らかの指標に基づいて最大化することである．特に視覚的デザインに関する領域においては，**審美的好ましさ** (aesthetic preference)，すなわちデザインの成果物の見た目が審美的に好ましいかどうかが，デザインの指標としてよく用いられる．例えば写真の色調編集においては，デザイナは「明るさ」や「コントラスト」などのパラメタを，写真の色調が審美的に最も好ましい状態になるように調整する．しかしながら，審美的好ましさは人間の認知と深く関連したものであるため，審美的好ましさの指標を簡潔なルールや数式を用いて数学的に定量化することは困難である．

本博士論文では，審美的好ましさを指標とするパラメタ調整作業を支援するためのコンピュテーショナルデザイン手法の可能性を模索する．第一に，コンピュテーショナルな技術を使って対象のデザイン空間中における好ましさの分布を推定する手法を追究する．推定された好ましさの分布は，手作業による対話的なデザイン探索を支援する目的で使用できる．第二に，コンピュテーショナルな技術を使って，対象のデザイン空間から最も好ましいパラメタを直接発見する手法を追究する．これら二つのアプローチでは，**クラウドソーシングに基づくヒューマンコンピュテーション** (crowdsourced human computation) と**編集履歴** (editing history) の二つの異なるデータ源から，計算に必要な人間の好みに関するデータを得る．クラウドソーシングに基づくヒューマンコンピュテーションを用いることで，不特定多数の群衆によって生成される「一般的な」好みのデータをオンデマンドに得ることができる．それに対し，対象のユーザの編集履歴を用いることで，そのユーザの「個人的な」好みのデータを得ることができる．

本博士論文では，特に以下の三種類のコンピュテーショナルデザイン手法を提案する．

1. 一つ目の手法では，対象とするデザイン空間中における好ましさの分布をクラウドソーシングに基づくヒューマンコンピュテーションを用いて推定する．推定された好ましさの分布は，手作業による対話的なデザイン探索を支援するための新しいデザインインタフェースに活用される．

2. 二つ目の手法では，一つ目の手法と同様に好ましさの分布を推定してそれを対話的なデザイン探索に活用するが，その分布の推定のためのデータとして対象とするユーザの編集履歴を用いる．更に，この履歴に基づく好みの推定技術と共に，実践的なシナリオで編集履歴を効果的に収集し活用するためのワークフローについても提案する．

3. 三つ目の手法では，この手法のユーザによる手作業のパラメタ調整を必要とすることなく，審美的な好ましさを最大化するような最も好ましいパラメタを対象のデザイン空間から自動的に発見する．これは，クラウドソーシングに基づくヒューマンコンピュテーションを用いて最適化計算の枠組みを構築することによって実現される．

我々は主に写真の色調編集のシナリオを用いてこれら三つの手法の有効性の評価を行ったが，例えば三次元コンピュータグラフィクスにおける照明デザインや仮想アバターの顔の表情のモデリングなど，その他の多様なデザイン領域への適用例も同時に提示する．結果として，それぞれの提案手法が一般的または個人的な審美的好ましさをコンピュテーショナルに扱うことができ，またデザイン活動を支援するために意味のある方法で機能することが示唆された．本研究で提案するコンピュテーショナルデザイン手法とそれによって得られた知見が，パラメタ調整作業の枠を超えてより複雑なデザインシナリオを扱っていく将来研究へと発展していくことを期待している．

Acknowledgements

My deepest gratitude goes to my supervisor, Takeo Igarashi, whose advice is always very smart and intellectually stimulating for me. I have learned really a lot from him for this five years, from ways to conduct successful research projects to ways to be a polite researcher. I would also like to thank him for providing me with many opportunities.

I thank Issei Sato for his professional advice as an expert of machine learning. I am grateful to all the Thesis Committee members: Masashi Sugiyama (Chair), Hiroshi Imai, Takayuki Itoh, Hideki Nakayama, Toshihiko Yamasaki, and Koji Yatani, for their thoughtful comments for improving this thesis.

I really like our laboratory, User Interface Research Group, and I would like to thank all the lab members. Especially, I greatly thank Daisuke Sakamoto, who always encourages me. Without his encouragement, I could not have done this thesis writing and could not maintained my motivation for this five years. I was very happy that I could invite him to the award ceremony for JSPS Ikushi Prize, which I could not have awarded if he were not in my research life. I also thank Naoki Sasaki, Genki Furumi, and Noah F. Wang for very exciting daily discussions in the first two years of my graduate school life. Collaborations with Kazutaka Nakashima, Morihiro Nakamura, and Lasse Farnung Laursen for side projects were really enjoyable experiences for me. I have learned how to write a good paper from the research collaborations with intelligent lab alumni: Nobuyuki Umetani, Kenshi Takayama, and Takashi Ijiri. Also, Jun Kato advises me a lot about not only particular research projects but also general research activities.

I was very fortunate to have the chance to participate in internship at Disney Research. I thank Shinjiro Sueda, who invited me for the internship, for spending a lot of his time for me and for getting our paper into SIGGRAPH Asia. I really enjoyed discussions with him. I also thank Ariel Shamir and Wojiech Matusik for supervising the project at Disney Research. Through the internship and the follow-up research period, I have learned a lot of things, such as how they conduct SIGGRAPH-quality researches and how they write SIGGRAPH papers.

I also thank IPA MITOH community and my project manager, Tatsunori Harada. All the experiences in my MITOH project were exciting for me. Especially, I have been really motivated by my MITOH colleagues, Yoshida Shigeo and Taisuke Ohshima. Also, Yoichi Ochiai has been always inspiring me by his unlimited passion since we first met at a MITOH meeting. I was also very fortunate to get acquainted with Masataka Goto and I would like to thank him for giving many discussion opportunities at various places.

It was really beneficial for me to have the experience of collaboration with researchers in the medical field: Kazuo Nakazawa and Shin Inada. Through the collaboration projects, I was able to expand my perspective on research. Also, discussions with Hirofumi Seo broadened my horizons and highly motivated me.

Additionally, I would like to give special thanks to Tatsunori Taniai, Keita Higuchi, Tatsunori Hirai, Hiroshi Kajino, Masaki Saito, and many others for both serious and casual discussions about what research is and what researchers should do.

I have been funded by a fellowship from the Japan Society for the Promotion of Science. The works included in this thesis have been partially supported by JSPS KAKENHI Grant Number 26·8574 and 26240027.

Finally and most importantly, a thank you to my fiancée, Minami Koto, for always encouraging me and being with me as my peerless partner.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem Formulation . . . 2
1.2.1 Challenges . . . 3
1.2.2 Assumptions and Scope . . . 4
1.3 Our Approach . . . 4
1.4 Organization of the Thesis . . . 6

2 Related Work **8**
2.1 Computational Design . . . 8
2.1.1 Functional Criteria . . . 9
2.1.2 Aesthetic Criteria . . . 10
2.2 Computational Perceptual Models . . . 10
2.3 Parameter Tweaking Interface . . . 12
2.4 Human Computation and Crowdsourcing . . . 12
2.4.1 Human Computation . . . 13
2.4.2 Crowdsourcing . . . 13
2.4.3 Crowdsourced Human Computation . . . 16
2.5 Parametric Spaces in Visual Design . . . 16
2.6 Summary . . . 18

3 Crowd-Powered Parameter Preference Estimation **19**
3.1 Introduction . . . 19
3.2 Crowd-Powered Parameter Analysis . . . 21
3.2.1 Overview of the Process . . . 21
3.2.2 Sampling Parameter Sets . . . 22
3.2.3 Gathering Pairwise Comparisons by Crowdsourcing . . . 22
3.2.4 Estimating Goodness Values of Sampling Points . . . 23
3.2.5 Fitting a Goodness Function . . . 25
3.3 User Interface . . . 26
3.3.1 Smart Suggestion . . . 26
3.3.2 VisOpt Slider . . . 26
3.4 Applications . . . 27
3.4.1 Photo Color Enhancement . . . 29
3.4.2 Camera and Light Control . . . 29
3.4.3 Shader (Material BRDF) . . . 30
3.4.4 Blendshape Facial Expression . . . 31
3.5 Evaluation . . . 32
3.5.1 Quality of Analysis . . . 32

3.5.2 User Study of the Interfaces 34
3.6 Discussion 35
3.6.1 Other Possible Representations of Goodness Function . . . 36
3.6.2 Limitation 36
3.6.3 Design Implication 36

4 History-Based Parameter Preference Estimation 38
4.1 Introduction 38
4.2 Related Work 41
4.2.1 Manual Photo Color Enhancement 41
4.2.2 Automatic Photo Color Enhancement 41
4.2.3 Demonstration-Based Techniques 42
4.3 Self-Reinforcing Photo Enhancement System 42
4.3.1 User Support Functions 43
4.4 Algorithms 44
4.4.1 Overview of Self-Reinforcement Procedure 44
4.4.2 Distance Metric Learning of Photos 46
4.4.3 Photo Feature Space Computation 47
4.4.4 Enhancement Preference Model 48
4.4.5 Implementation of User Support Functions 49
4.5 User Study 50
4.5.1 Participants 50
4.5.2 Procedure 50
4.6 Results 51
4.6.1 Preliminary and Post-Task Questionnaires 51
4.6.2 Feedback in Interview 52
4.6.3 Quantitative Results 55
4.7 Discussion 56
4.7.1 Summary of User Study 56
4.7.2 Design Implications 57
4.7.3 Limitations 57
4.7.4 Future Work 58

5 Crowd-Powered Parameter Preference Maximization 60
5.1 Introduction 60
5.1.1 Existing Methods 61
5.1.2 Contributions 62
5.2 Background: Bayesian Optimization 63
5.2.1 Overview 63
5.2.2 Gaussian Process Prior 64
5.2.3 Covariance Hyperparameters 65
5.2.4 Acquisition Function 66
5.2.5 Example Optimization Sequences 66
5.3 Bayesian Optimization Based on Line Search Oracle 68
5.3.1 Slider Space 68
5.3.2 Likelihood of Single-Slider Manipulation Responses 69
5.3.3 Data Representation 70
5.3.4 Inference from Single-Slider Manipulation Data 70
5.3.5 Example Optimization Sequence 71

5.4 Crowd-Powered Visual Design Optimizer 71
5.5 Example Scenarios and Results 75
5.5.1 Photo Color Enhancement 75
5.5.2 Material BRDF Editing 76
5.6 Evaluation 80
5.6.1 Experiment 1: Synthetic Setting 80
5.6.2 Experiment 2: Crowdsourcing Setting 82
5.7 Limitations and Future Work 84

6 Conclusion **86**
6.1 Summary 86
6.2 Discussions 87
6.2.1 Estimation of $g(\cdot)$ vs. Maximization of $g(\cdot)$ 87
6.2.2 Crowdsourced Human Computation vs. Editing History . . 88
6.3 Remaining Challenges 89
6.3.1 Design Space Parametrization 89
6.3.2 Discrete Parameters 90
6.3.3 Locally Optimal Design Alternatives 90
6.3.4 Evaluation Methodology 91
6.3.5 More Complex Models of Crowd Behaviors 91
6.3.6 Incorporating Domain-Specific Heuristics 92
6.3.7 Combining Crowd and Personal Preference 92
6.4 Future Directions 92
6.4.1 Free-Form Design from Scratch 92
6.4.2 More Complex Design Criteria 93
6.4.3 Computational Creative Design 93

References **95**

Chapter 1

Introduction

1.1 Motivation

Quality of designed objects could be assessed using various criteria according to their usage contexts. Especially in visual design domains, *aesthetic preference*—how aesthetically preferable the design looks—is an important criterion. For example, photo color enhancement (also referred to as tonal adjustment, color grading, or color correction) is one of such design scenarios where aesthetic preference performs the role of a criterion based on which the quality is determined (see Figure 1.1). When a designer enhances the color of a photograph, he or she has to tweak multiple design parameters such as "brightness" or "contrast" via a slider interface to find the most aesthetically preferable enhancement for the target photograph.

In general, finding the best parameter combination is not an easy task. It may be easily found by a few mouse drags in case that the designer is very

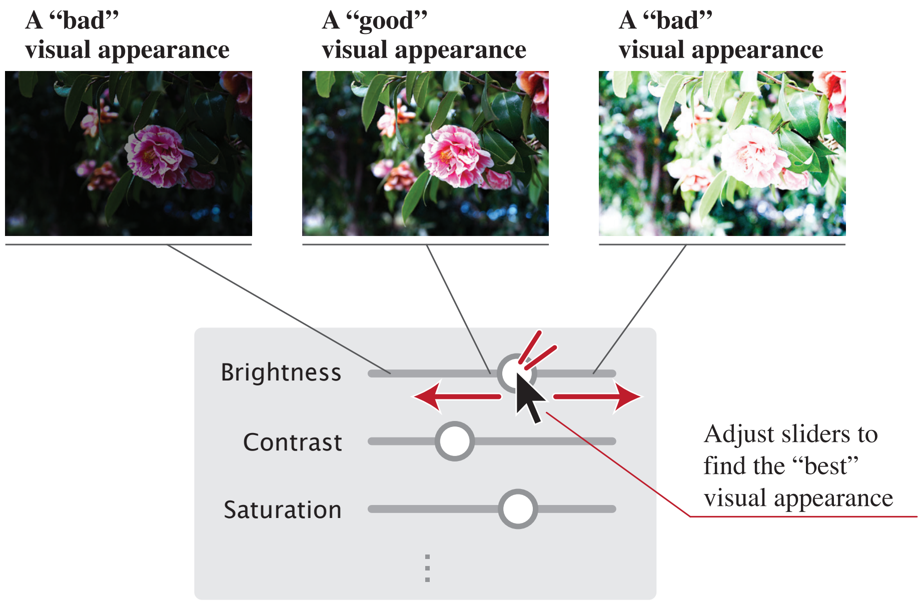
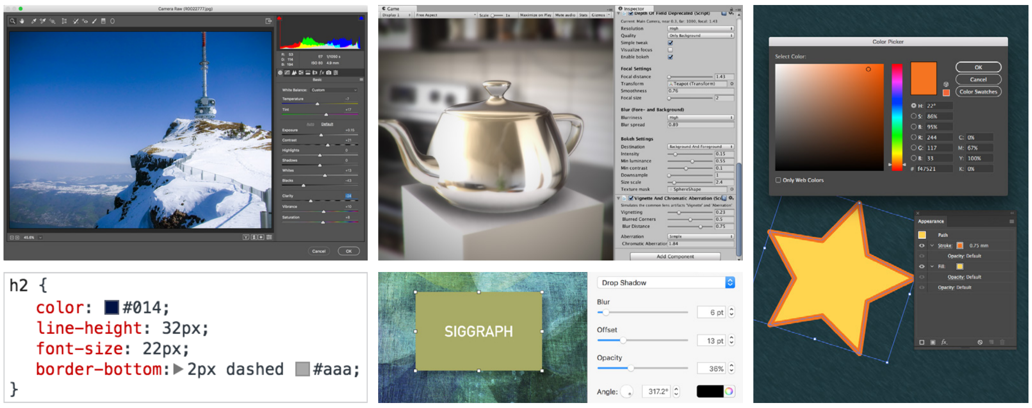

A "bad" visual appearance
A "good" visual appearance
A "bad" visual appearance
Brightness
Contrast
Saturation
Adjust sliders to find the "best" visual appearance

Figure 1.1: An example of design parameter tweaking where aesthetic preference is used as a criterion. Photo color enhancement is one of such design scenarios, in which designers tweak sliders such as "brightness" so that they eventually find the parameter set that provides the best preferable photo enhancement.



```
h2 {
   color: ■#014;
   line-height: 32px;
   font-size: 22px;
   border-bottom:▶2px dashed ■#aaa;
}
```

Drop Shadow
Blur 6 pt
Offset 13 pt
Opacity 36%
Angle: 317.2°

Figure 1.2: Example scenarios of parameter tweaking for visual design, including photo color enhancement, image effects for 3D graphics, and 2D graphic designs such as web pages, presentation slide, *etc.*

familiar with how each parameter affects the visual content and is very good at predicting the effects without actually manipulating sliders. However, this is unrealistic in most practical cases; several sliders mutually affect the resulting visuals in complex ways, and also each slider affects differently when contents are different, which make the prediction very difficult. Thus, in practice, it is inevitable that a designer explores the *design space*—the set of all the possible design alternatives—in a trials-and-errors manner, to find the parameter set that he or she believes the best for the target content. This requires many slider manipulations, as well as the designer to construct a mental model of the design space. Furthermore, as the number of design parameters increases, the design space expands exponentially, which makes this exploration very tedious. In the example of photo color enhancement, there are eleven "basic" sliders in Adobe Photoshop Lightroom CC [10] that need to be tweaked, which is already time-consuming to explore all the possibilities.

Though difficult, parameter tweaking is very common task, and similar situations can be observed in many design scenarios. Figure 1.2 illustrates example scenarios of such parameter tweaking where parameters are tweaked so that the visual content is aesthetically the best. For example, in Unity (a computer game authoring tool) [150] and Maya (a 3-dimensional computer animation authoring tool) [16], there are many sliders in the control panes, which need to be tweaked to adjust the visual of contents. In this thesis, we aim to support (or possibly automate) this general design task of preference-based parameter tweaking.

1.2 Problem Formulation

In this thesis, we investigate computational design support methods for facilitating design exploration in which aesthetic preference is used as a criterion. Specifically, we consider the design exploration where multiple design parameters have to be tweaked such that the aesthetic preference criterion is maximized. This can be mathematically described as follows. Suppose that there are n real-valued design variables

$$\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \in \mathcal{X}, \tag{1.1}$$

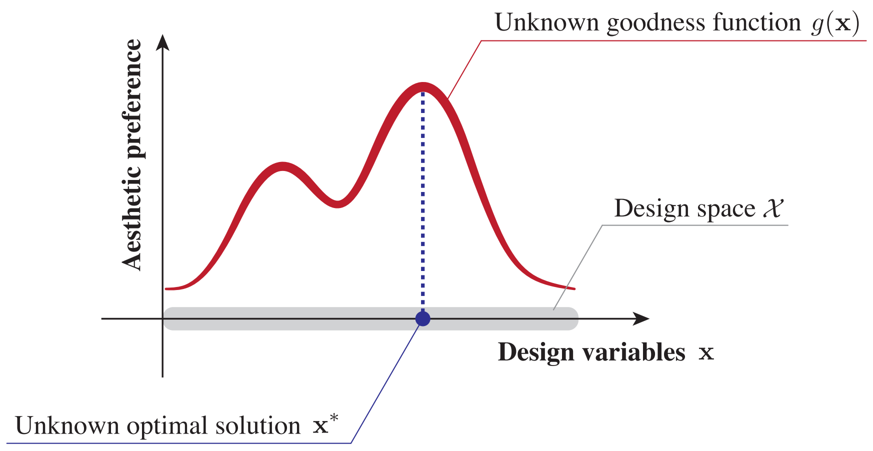

Unknown goodness function $g(\mathbf{x})$
Aesthetic preference
Design space $\mathcal{X}$
Design variables $\mathbf{x}$
Unknown optimal solution $\mathbf{x}^*$

Figure 1.3: Our problem setting. We seek computational design methods to solve the optimization problem described in Equation 1.3, or to find the optimal solution $\mathbf{x}^*$ that maximizes the aesthetic preference of the target design.

where $\mathcal{X}$ represents an n-dimensional design space. We assume that

$$\mathcal{X} = [0, 1]^n \tag{1.2}$$

for simplicity, *i.e.*, each variable takes a continuous value and its interval is regularized into $[0, 1]$ in advance. In case that a variable is tweaked by a slider, we suppose that the slider's lowest and highest values correspond to 0 and 1, respectively. Using these notations, a parameter tweaking task is described as an optimization problem:

$$\mathbf{x}^* = \mathop{\arg\max}_{\mathbf{x} \in \mathcal{X}} g(\mathbf{x}), \tag{1.3}$$

where the objective function $g : \mathcal{X} \to \mathbb{R}$, which we call *goodness function*, returns a scalar value representing how aesthetically preferable the design corresponding to the argument design variables is. A designer usually tries to solve this optimization problem by exploring $\mathcal{X}$ manipulating sliders in a trials-and-errors way without any computational support. This exploration ends when the designer believes that he or she finds the optimal argument value $\mathbf{x}^*$ that gives the best preferable design. Figure 1.3 illustrates this problem setting. Our goal is to seek methods for solving this optimization problem either automatically or semi-automatically using computational tools.

1.2.1 Challenges

Compared to typical optimization problems in computer science and engineering, the optimization problem that we are going to solve has several irregular features. The objective function $g(\cdot)$ is constructed based on human perception, and exists only in brains of designers. Thus, it is generally difficult to represent $g(\cdot)$ as a simple equation that can be calculated using machine processors alone, meaning that it is inevitable to somehow involve humans. Furthermore, even a designer themself cannot answer the goodness value for a design before exploring and learning the entire design space. This means that we cannot directly use "function-value" queries like typical optimization settings. Although some previous works in machine learning and computer graphics communities (*e.g.*, [41, 33, 139]) have tackled problems similar to ours, this direction has not been investigated sufficiently yet, and there remains many possibilities to be investigated especially from the viewpoint of human-computer interaction.

1.2.2 Assumptions and Scope

As there are a variety of possible design scenarios, we put several assumptions on this problem setting to clarify our scope. First, we assume that the target parameters are continuous, and thus discrete ones, *e.g.,* font type selection, are out of the scope. We also assume that, when a design parameter changes smoothly, the corresponding visual also changes smoothly; we do not handle discontinuous changes, *e.g.,* line breaking in text-box size adjustment. From these assumptions, the goodness function $g(\cdot)$ is considered a continuous, smooth function. Note that the goodness function is allowed to have multiple local maximums, ridges, or locally flat regions around maximums. Also, we assume that the goodness function is constant with respective to time. The design space is expected to be parameterized by a reasonable number of parameters as in most commercial softwares; parametrization itself is out of our scope. We handle the design domains where even novices can assess relative goodness of designs (for example, given two designs, they are expected to be able to answer which design looks better given two designs); but importantly, they do not need to know how a design can be improved.

Though we narrow down the target problem as discussed above, it still covers a wide range of practical design scenarios: photo color enhancement, material design by editing the bidirectional reflectance distribution function (BRDF), facial expression modeling via blendshape, 2-dimensional graphic design such as posters, procedural texture and modeling, post-rendering image effect, *etc.*. In this thesis, we use photo color enhancement as the main example to validate our methods.

1.3 Our Approach

To provide computational methods for solving the design problem described as Equation 1.3, we investigate two approaches with respect to the usages of computational tools:

Estimation of $g(\cdot)$: The first approach is to estimate the shape of the goodness function $g(\cdot)$ by using computational tools. In other words, it is to compute regression of $g(\cdot)$. Once $g(\cdot)$ is estimated, it can be used for "guided" exploration: supporting users' free exploration of the design space $\mathcal{X}$ for finding their best favorite parameter set $\mathbf{x}^*$ through some user interfaces. One of the advantages of this approach is that, even if the estimation is rough or its quality is not perfect, it can still be effective for supporting users to find $\mathbf{x}^*$. To implement this approach, there are several challenges: how to compute this unusual regression problem (in which the algorithm cannot query function values directly), and how to support the users' manual exploration using the estimated $g(\cdot)$.

Maximization of $g(\cdot)$: The second approach is to compute maximization of the goodness function $g(\cdot)$ by using computational optimization tools so that the system can directly find the optimal solution $\mathbf{x}^*$. That is, the system searches the design space $\mathcal{X}$ for the maximum of $g(\cdot)$ automatically. While the first approach equally handles the entire design space $\mathcal{X}$ everywhere, this approach focuses on only maximums and exploring paths for finding them. Thus, while it is also possible to estimate $\mathbf{x}^*$ by using the estimated $g(\cdot)$ by the first approach, the computational cost is expected to be less in this approach, especially when the dimension of $\mathcal{X}$ is high. The found solution $\mathbf{x}^*$ can be used as either a final design or a starting point that will be further refined by the user. Implementing this approach requires several non-trivial considerations, *e.g.,* which optimization algorithm can be used, and how it should be adapted for our special problem setting.

For both the approaches, human-generated preference data is necessary for enabling computation. In this thesis, we investigate the following two sources to obtain such data.

Crowdsourced human computation: First, we investigate the use of *human computation* to generate necessary preference data. As we will review in Section 2.4, human computation is a paradigm where humans are explicitly considered as processing power and integrated within systems. Such *human processors* can be obtained via *crowdsourcing*. By this approach, systems can obtain crowd-generated data on demand as function calls. Here we put an additional assumption: there exists a common "general" preference shared among crowds; though there might be small individual variation, we can observe such a general preference by involving many crowds.

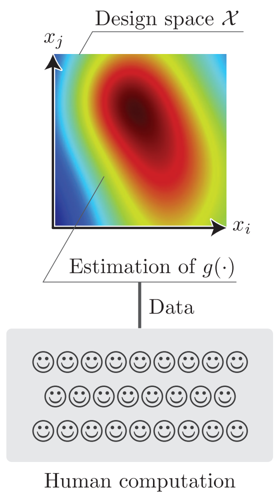
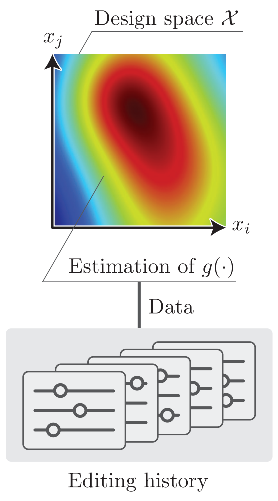
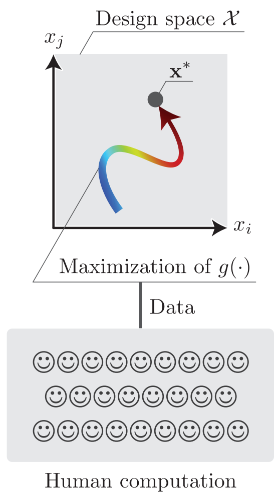
Editing history: Second, we investigate the use of *editing history* of a single user as the preference data source. While crowdsourced human computation handles "general" preference, this approach can handle "personal" preference of the target user. However, as the editing history cannot be generated on demand, some appropriate workflow may be necessary so that the system implicitly gathers available data and effectively utilizes them.

Based on the above discussions, in this thesis, we propose the following three specific methods.

Crowd-powered parameter preference estimation. This method estimates the goodness function $g(\cdot)$ using crowdsourced human computation. Specifically, we present a computational technique to infer $g(\cdot)$ from pairwise-comparison data generated by crowds. Along with this technique, we also propose a new slider interface called *VisOpt Slider* that facilitates users' interactive design exploration using the estimated $g(\cdot)$.

History-based parameter preference estimation. This method also estimates the goodness function $g(\cdot)$, but from personal editing history of a target user. As the data form is different, we propose a different computational techniques for estimating $g(\cdot)$. We also present a new workflow for effectively gathering and utilizing editing history. Our prototype system, named *SelPh*, supports users' manual design exploration through several interface functions including an extended version of VisOpt Slider. In this method, we especially focus on photo color enhancement application to validate the effectiveness of this method in professional scenarios.

Table 1.1: Overview of the three methods presented in this thesis. In the first and the second methods, we utilize computational techniques for estimating the goodness function $g(\cdot)$, while, in the third method, we do for directly searching the design space $\mathcal{X}$ for the optimal parameter set $\mathbf{x}^*$. As the data sources for computation, we seek to use crowdsourced human computation in the first and third methods, and editing history in the second method.

Crowd-powered estimation (Chapter 3)	History-based estimation (Chapter 4)	Crowd-powered maximization (Chapter 5)
 Design space $\mathcal{X}$ x_j x_i Estimation of $g(\cdot)$ Data Human computation	 Design space $\mathcal{X}$ x_j x_i Estimation of $g(\cdot)$ Data Editing history	 Design space $\mathcal{X}$ x_j $\mathbf{x}^*$ x_i Maximization of $g(\cdot)$ Data Human computation

Crowd-powered parameter preference maximization. Instead of estimating the goodness function $g(\cdot)$, this method directly explores the design space $\mathcal{X}$ and tries to find the maximum $\mathbf{x}^*$, by computing optimization process using crowdsourced human computation. We call this new paradigm as *crowd-powered visual design optimization*. While existing related methods use pairwise-comparison queries for crowdsourcing, we propose a novel query design which requires less iterations for finding $\mathbf{x}^*$, meaning that less computational cost is required.

Table 1.1 provides an overview of these three methods.

1.4 Organization of the Thesis

First, we review the related research areas in computer graphics and human-computer interaction in Chapter 2, and clarify the position of our overall attempt and each contribution. At the same time, we also try to clarify terminologies used in this thesis.

In Chapter 3, we describe the crowd-powered estimation method in detail and demonstrate its potential by applying it to four different design scenarios. The contributions presented in this chapter has been published as *Crowd-Powered Parameter Analysis for Visual Design Exploration* [75] at the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14) in Honolulu, USA. This work was conducted in collaboration with Daisuke Sakamoto and

Takeo Igarashi from the University of Tokyo.

In Chapter 4, we describe the history-based estimation method in detail and validate its effectiveness in professional photo color enhancement scenarios through a user study with eight expert designers. The contributions presented in this chapter has been published as *SelPh: Progressive Learning and Support of Manual Photo Color Enhancement* [76] at the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16) in San Jose, USA. This work was conducted in collaboration with Daisuke Sakamoto and Takeo Igarashi from the University of Tokyo.

In Chapter 5, we describe the crowd-powered maximization method in detail, demonstrate several usage scenarios, and evaluate its efficiency by comparing existing approaches. The contributions presented in this chapter is currently under review as a paper entitled *Sequential Line Search for Visual Design Optimization by Crowds* [77]. This work was conducted in collaboration with Issei Sato, Daisuke Sakamoto, and Takeo Igarashi from the University of Tokyo.

Finally, we summarize our contributions and the lessens learned from our investigation in Chapter 6. We then revisit the assumptions made in this thesis, and discuss the limitations and the future directions.

Chapter 2

Related Work

Our problem setting described in Chapter 1 is general and can be observed in many design scenarios. However, the same problem setting has not been investigated very intensively; there are only several previous projects (*e.g.,* [33, 139]), which we will discuss in details in the latter chapters. In this chapter, rather than describing specific technical differences from previous work, we aim at clarifying the position of our overall investigation by reviewing several sub-fields in computer graphics and human-computer interaction that are partially overlapped with our attempt, with respect to target contexts and techniques.

2.1 Computational Design

In this thesis, we use the term, *computational design*, as the emerging form of design activities that are enabled by computational techniques. In this concept, design activities are often formulated as mathematical optimization problems; design criteria perform the roles of either objective functions or constraints, design space is considered as the search space (or the choice set), and design exploration, which can be performed by either systems, users, or their combination, is considered as the process of searching for solutions. This viewpoint provides an opportunity to devise new ways of utilizing computational techniques (*i.e.,* mathematical tools developed in computer science that can bring out machine processing power). The main goal of researches on computational design is to enable efficient design workflow or complex design outcomes that are impossible in traditional approaches relying heavily on human thinking capacity.

Computer-aided design (CAD) is a related concept in which computational tools, such as geometric techniques (*e.g.,* [63]) and physical simulation (*e.g.,* [148]), may be utilized to increase the productivity of designers. We consider that computational design and CAD are partially overlapped; however, CAD is designed mainly for accelerating traditional design activities, and not necessarily being formulated as optimization.

Designed objects are assessed using various criteria depending on usage contexts. Some criteria might work as conditions that should be at least satisfied (*i.e.,* constraints); other criteria might work as values that is desired to be maximized (*i.e.,* objectives). We classify these design criteria into two groups: *functional* criteria and *aesthetic* criteria. Functional criteria are the criteria about how well the designed object serves in the expected contexts. For example, a chair is expected to be "durable" when someone is setting on it; in this case, durability

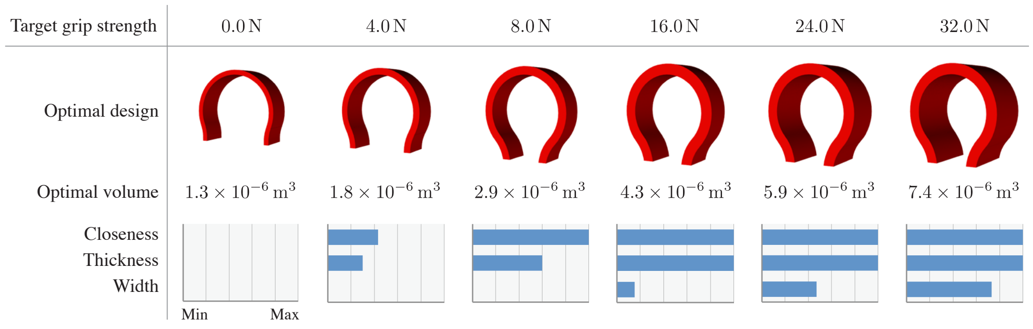


Figure 2.1: An example of computational design driven by functional criteria. In this example from [78], to design a 3D-printable functional "pipe clamp" that can hold objects rigidly, the *grip strength* is considered to be a user-specified constraint, and the *material consumption* is considered to be an objective function to be minimized. The system solves such a constrained optimization problem and provides the optimal design parameters (in this case, design parameters consist of closeness, thickness, and width of the pipe clamp).

can be a functional criterion that should be satisfied. On the other hand, aesthetic criteria are the criteria about how perceptually pleasing (or preferable) the designed object looks. A chair might look "more beautiful", for example, if its shape is smooth and the width and height follow the golden ratio rule; in this case, beauty in shape performs the role of an aesthetic criterion that is desired to be maximized. Note that these two criteria are orthogonal; in practical design scenarios, these two criteria may be simultaneously considered by designers. In the following subsections, we review existing computational design methods for each type of design criteria.

2.1.1 Functional Criteria

Recently, many computational design methods for designing functional objects have been intensively investigated, especially for digital fabrication applications. So far, a variety of functional criteria have been handled and formulated by researchers; Umetani *et al.* [149] formulated the functional criterion of paper airplane designs, *i.e., fly-ability*, and used it for optimizing airplane designs by maximizing the fly-ability criterion. Their followers extended the fly-ability formulation for kites [98] and bamboo-copters [103]. Koyama *et al.* [78] formulated *hold-ability* and *grip strength* of 3D-printed connectors and then presented an automatic method named AutoConnect (see Figure 2.1) for designing functional connectors. Several computational design methods consider functionalities about objects' mass properties, *e.g., standing stability* [115, 164], *spinning stability*, [18, 103] and *floating stability* [156]. *Structural strength* of objects is also an important factor, and researchers have developed computational design methods taking this criterion into consideration [134, 93, 99].

Another notable domain of computational functional design is automatic graphical user interface (GUI) generation. For example, Gajos *et al.* [52] presented an automatic GUI design method, in which they formulated *required user efforts* for manipulating GUI elements as the objective function to be minimized. Laursen *et al.* [82] presented a method of choosing an optimal set of icons for GUI taking icons' *identifiability* and *comprehensibility* into consideration.

2.1.2 Aesthetic Criteria

Computational design using aesthetic criteria is achieved by not only predicting (as discussed in Section 2.2) but also maximizing perceptual aesthetic quality. Compared to functionalities of things, aesthetic preference is closely tied to human perception and thus it is more difficult to quantify using simple rules. Yet, by focusing on very specific design domains, it is possible to handle and optimize aesthetic criteria by rule-based approaches. For example, Liu *et al.* [90] presented a computational photo cropping method, where they consider several heuristic rules (or guidelines) for aesthetically pleasing photo composition such as *rule of thirds* and *visual balance*, in the objective function. In general, rule-based approaches require careful implementation of heuristic rules and manual tuning of models parameters for rules.

Data-driven approaches can ease the limitations in rule-based approaches. Most of data-driven methods yet rely on heuristic rules, but can derive optimal weights or model parameters for the rules by learning them from training data. For example, O'Donovan *et al.* [109, 110] presented a data-driven method of predicting and optimizing aesthetic quality of layouts of 2-dimensional graphic designs. Their aesthetic criterion is formulated by combining several heuristic rules, *e.g., alignment* and *white space*, and machine learning techniques are used to learn the weights and the model parameters. Other examples handle color palette aesthetics [108, 73], 3D viewpoint preference [124], and photo color enhancement [35].

In this thesis, we focus on general parameter tweaking tasks, and investigate as general methods as possible. Thus, we decide not to rely on domain-specific rules or database. In this sense, Talton *et al.*' method [139] is similar to ours. Their method constructs a so-called *collaborative design space*, which is a subset of the target design parameter space consisting of only aesthetically acceptable designs, based on the design history of many volunteer users. Then, the collaborative design space supports new users' design exploration. While their method takes roughly one year to obtain the necessary design history and needs many volunteers to engage exactly the same design space, our methods are designed to obtain necessary data on demand (the crowd-powered methods) or during a repetitive design session (the history-based method).

2.2 Computational Perceptual Models

Techniques for quantifying aesthetic preference have been investigated in various design domains. This attempt is sometimes referred as *computational aesthetics*, and some of the aesthetic preference models are used for building computational design frameworks as we discussed in Subsection 2.1.2. For example, Secord *et al.* [124] showed how the aesthetic preference of viewing direction for 3-dimensional models can be computationally assessed (Figure 2.2 (Top left)). Automatic assessment of photograph quality has been intensively investigated by many researchers [48, 72, 95, 96, 107]. Most of these methods are based on domain-specific knowledge or heuristic techniques; usually, a visual content that will be assessed is translated as a domain-specific feature vector, and then converted to a scalar value that represents its goodness in terms of aesthetics. Our methods also deal with aesthetic preference, but basically do not heavily

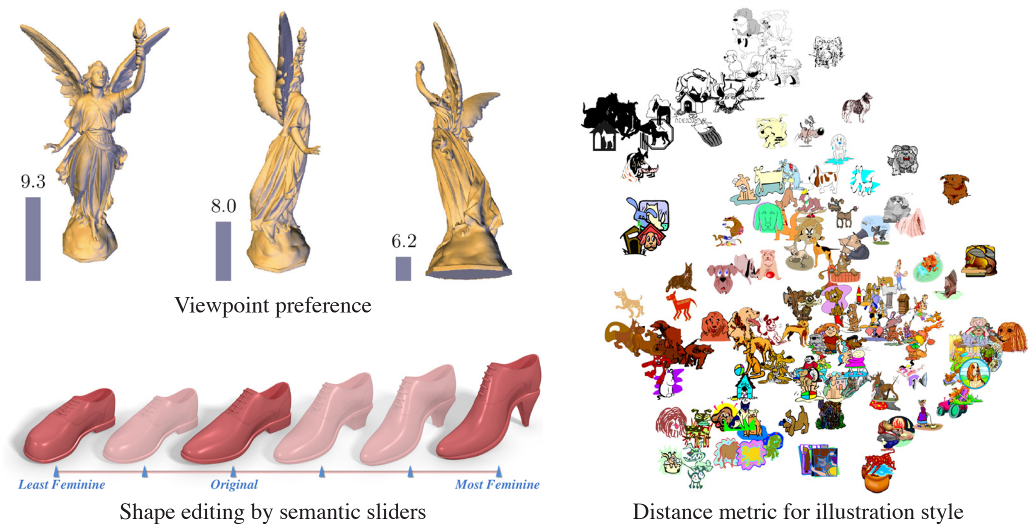

9.3
8.0
6.2
Viewpoint preference
Least Feminine
Original
Most Feminine
Shape editing by semantic sliders
Distance metric for illustration style

Figure 2.2: Examples of computational perceptual models. (Top left) Predicted viewpoint preferences [124]. (Bottom left) 3D shape editing using semantic sliders [167]. (Right) A 2D visualization of distance metric for illustration style [54], learned based on human perception.

rely on domain-specific formulations, which allows our methods to be applicable to various domains. Also, our methods estimate aesthetic preference on design parameter spaces, rather than on feature spaces.

Another popular domain of computational perceptual models are modeling *semantic attributes* of visual contents. By transforming original design spaces into semantic design spaces, researchers have provided various tools for intuitive design exploration. For example, Yumer *et al.* [167] proposed a 3-dimensional shape editing tool that has semantic sliders such as “compact”, “comfortable”, and “fashionable”, and users can explore possible deformations of the target shape by manipulating the sliders (Figure 2.2 (Bottom left)). Similarly, semantics of fonts [111], discrete design components [39], BRDFs [125], and human body shapes [135] are analyzed for supporting design exploration. Our goal is to support users to find aesthetically best designs, so that building semantic design spaces is out of our scope.

Modeling perceptual distance (or dissimilarity) metrics is also an active research domain. Measuring distances between visual contents in raw feature spaces might be perceived non-uniform. For example, l^2-norm between colors in the RGB space is known to be perceptually non-uniform; to measure perceptual distance between colors, several metrics, *e.g.,* CIEDE2000 [129], have been proposed. Recently, researchers have utilized machine learning techniques (see [80] for technical details) to computationally model perceptual distances in various design domains, *e.g.,* fonts [111], illustration style [54], and shape style [94, 91]. Figure 2.2 (Right) shows an example of illustration style. In Method B, we learn a perceptual distance metric for photographs, but unlike other methods, we learn the metric so that it reflects users’ preference and it supports design exploration.

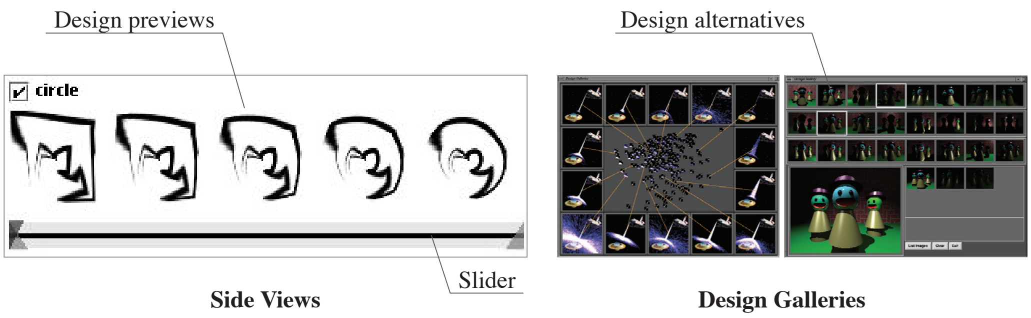


Figure 2.3: Interfaces for design parameter tweaking. (Left) Side Views [143] visualizes design previews along with the target slider. (Right) Design Galleries [97] provides users with design alternatives in a gallery style.

2.3 Parameter Tweaking Interface

One of the most popular interfaces for tweaking parameters is the slider widget, and it has been widely used for tweaking continuous parameters. Researchers have augmented slider widgets in several ways. *Side Views* [143] shows design previews along with widgets so that users can efficiently learn design alternatives achieved by the slider adjustment (see Figure 2.3 (Left)). *Scented Widgets* [159] embeds useful visual cues directly into widgets for facilitating data exploration. *Parallel Paths* [144] and *Juxtapose* [60] allow users to tweak designs in a parallel manner by simultaneously maintaining several alternatives in design interfaces. In this thesis, we present a new slider interface called *VisOpt Slider*, which effectively utilizes the information of the estimated goodness function for facilitating users' design exploration. Note that several recent works adopted similar slider interfaces for support design exploration in different situations [130, 167, 135].

Parameters can be tweaked by approaches other than sliders. *Design Galleries* [97] is a gallery-based approach for exploring high-dimensional design space, where the system shows many design alternatives generated by various candidate parameter sets and users can obtain a good parameter set by simply picking up the best favorite design (see Figure 2.3 (Right)). Our *Smart Suggestion* interface proposed with the crowd-powered estimation method (Chapter 3) is inspired by this approach, and takes the goodness values of design alternatives into account. Inverse design methods (*e.g.,* [160, 112, 140]) finds the best parameter set from users' specifications by solving inverse problems. This approach is effective when users have very concrete goal visions in advance. In contrast, we deal with exploratory design scenarios, where the final design goal is formed through exploration.

2.4 Human Computation and Crowdsourcing

Human computation and crowdsourcing are often used for gathering human-generated data that is difficult for machine to generate (*e.g.,* perceptual or semantic labels for images). We utilize this approach for formulating our crowd-powered methods for gathering perceptual preference data. We encourage readers to refer the comprehensive survey and discussions on these terms by Quinn and Bederson [116]. In this section, we review these two terms from the viewpoint of our attempt.

2.4.1 Human Computation

Human computation is a concept of enabling computations by exploiting humans as processors. This term was described by von Ahn [152] as below:

> *". . . a paradigm for utilizing human processing power to solve problems that computers cannot yet solve."*

For example, human processors are much better at perceiving semantic meanings of visual contents than machine processors; thus, for building a system that requires perceptive abilities, it may be effective to incorporate human processors as well as machine processors. Such problems that are difficult for machine processors but easy for human processors, including ours, are observed in many situations. However, human processors also have critical limitations such as that they are extremely slow and expensive to execute compared to machine processors. So, it is important to carefully choose where and how to employ such human processors.

It is possible to design a human computation system so that a user of the system themself behaves as a human processor. Interactive evolutionary computation (see [138] for the comprehensive survey) is one of such examples, where the user interactively specifies evaluations required in evolutionary computation (or, behaves as the *fitness function* in evolutionary computation). Brochu *et al.* [33] presented a visual design optimization method where the user is asked by the system to evaluate visual designs. We consider these methods as human computation although they are not always described as human computation explicitly.

For employing many human processors, one possible solution is to implicitly embed human computation tasks in already existing tasks. reCAPTCHA [155] takes such an approach where optical character recognition (OCR) tasks are embedded to web security measures. A challenge of this approach is that most of human computation tasks are difficult to naturally embed in existing tasks, and it needs very careful interaction design to be successful.

To motivate many ordinary people to voluntarily participate in human computation tasks, von Ahn proposed to do "gamification" of tasks so that they do tasks purely for entertainment purpose [154]. These games are called *games with a purpose* (GWAPs). For example, the ESP game [153] is a game in which players provide semantic labels for images (that can be used for machine learning) without realizing it. The Foldit game [45] lets players explore and find protein structures, which is a hard scientific problem.

Very recently, since the emergence of large-scale crowdsourcing markets, it has become increasingly popular to employ human processors using crowdsourcing. As we take this approach, we detail it in the following subsections.

2.4.2 Crowdsourcing

The term "crowdsourcing" was firstly introduced by Howe [62] in 2006, and later explicitly defined in [61] as

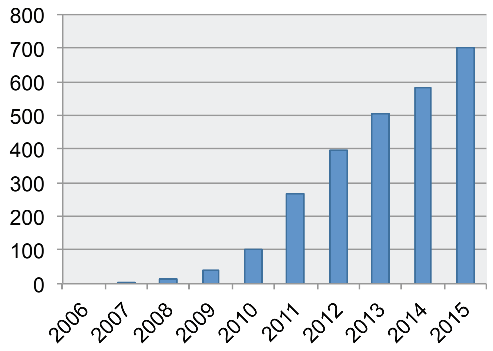


Figure 2.4: The number of newly published records in computer science literature using the term of "crowdsourcing". Search results were counted in the ACM Guide to Computing Literature on October 23, 2016.

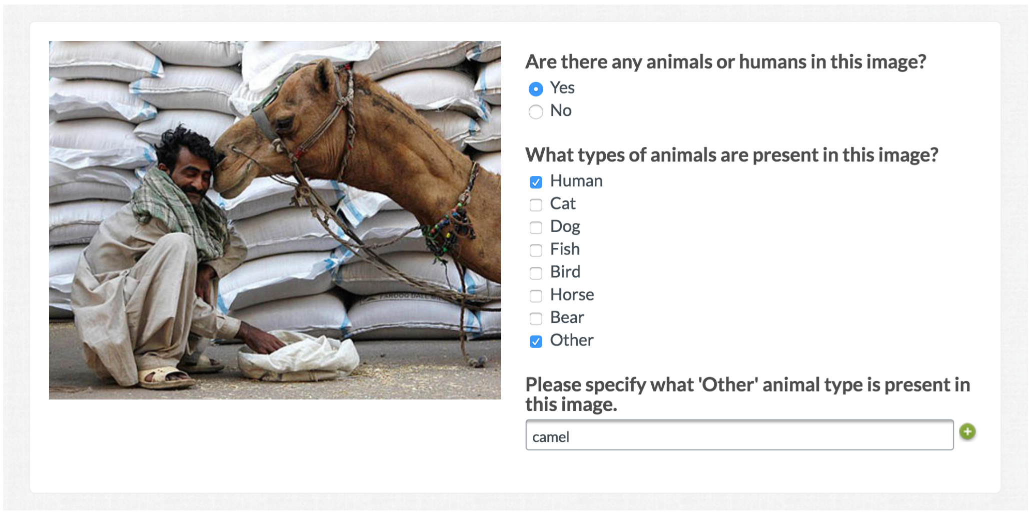
> *"Crowdsourcing is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call."*

Today, many online marketplaces for crowdsourcing are available for researchers, such as Upwork [6], Amazon Mechanical Turk [1], and CrowdFlower [2]. Since 2006, crowdsourcing has been a more and more popular research topic in computer science (see Figure 2.4).

Microtask-Based Crowdsourcing

One of the most attractive features in crowdsourcing is that anyone can stably hire a large number of crowd workers even for very small tasks on demand without any communication cost for hiring. This is impossible without recent crowdsourcing marketplaces [1, 2]. Amazon Mechanical Turk [1] is one of the popular platforms suitable for outsourcing such a large number of small tasks. We define *microtasks* as the tasks for crowdsourcing that is very small (usually completed in a minute) and that can be conducted anyone who do not have any special skills or domain knowledge. We would call this type of crowdsourcing as *microtask-based crowdsourcing*. Figure 2.5 shows an example of such microtasks. Although crowd workers in these platforms are usually non-experts, they do have full human intelligence, which enables many emerging applications.

One of the popular usages of microtask-based crowdsourcing is to outsource data-annotation tasks (*e.g.*, [21, 20]) that could be useful as training data for machine learning. Another popular usage is to conduct large-scale perceptual user studies (*e.g.*, [74]); by using microtask-based crowdsourcing, it is easy to gather thousands of participants on demand, which were traditionally difficult. The results of studies might be used to build computational perceptual models as discussed in Section 2.2. Microtask-based crowdsourcing is also used for implementing *crowd-powered user interfaces*, interfaces that query crowd workers to use their human intelligence in run time. For example, Soylent [25] is a crowd-powered word processing system that utilizes human intelligence to edit text documents. VizWiz [27] and VizLens [57] are crowd-powered mobile systems for blind people to understand the real world texts and interfaces where users take


Are there any animals or humans in this image?
[✓ Yes] No

What types of animals are present in this image?
[✓ Human] Cat Dog Fish Bird Horse Bear [✓ Other]

Please specify what 'Other' animal type is present in this image.
camel

Figure 2.5: An example of typical microtasks, taken from the tutorial in CrowdFlower [2].

photos of them and the systems ask crowd workers to read the contents appeared in the photos. Voyant [162] is a design interface that gathers structured feedback for a visual design from crowd workers.

Our systems described in Chapter 3 and Chapter 5 can be considered as crowd-powered user interfaces to utilize human perception obtained through microtasks. Compared to the crowd-powered systems discussed above, ours do not explicitly indicate the presence of crowd workers to users; rather, our methods use crowd workers as abstracted processing powers in the way of human computation. We discuss this approach in Subsection 2.4.3.

One issue that should be cared in microtask-based crowdsourcing is the *quality control* of workers' responses [64]. Crowd workers might make poor-quality responses because of cheating, misunderstanding of tasks, or simply mistakes. Various quality control methods are investigated in machine learning community [83]. In this thesis, we adopt a simple quality control method based on redundancy called *duplication* (used in many works, *e.g.,* [55, 39, 111, 54]), and leave it future work to investigate better quality control strategies suitable for our methods.

Expert Sourcing

Another attractive feature in crowdsourcing is that we can easily hire skilled experts (*e.g.,* web developers, designers, and writers) for professional tasks, which is called *expert sourcing*. Some online marketplaces, *e.g.,* Upwork [6] (formerly Elance-oDesk), provide such an opportunity for researchers to reach experts. For organizing expert crowds in a computational manner, Retelny *et al.* [119] presented techniques called *flash teams*. Suzuki *et al.* [136] presented an idea of *micro-internships* for expert sourcing, where relatively unskilled crowds can develop their skills via internship tasks working with more skilled (mentor) workers.

However, asking professional designers takes significant communication costs and large variances between individuals' skills. Compared to microtask-based crowdsourcing, expert sourcing is less suitable for employing human processors and for designing human computation algorithms that work stably, on demand, like a cloud computing.

Optimizing Crowdsourcing Performance

One of the issues in crowdsourcing is the latency for obtaining responses from crowds. This is not very critical when, for example, researchers use crowdsourcing for gathering training data for machine learning (*e.g.,* [124, 39, 54, 111]), because this latency is experienced only once. In contrast, the latency can be critical when, for example, end-users use crowd-powered systems (*e.g.,* [27, 25, 24, 162]), because long latency badly affects user experience. To reduce such latency, Bigham *et al.* [27] proposed *quikTurKit*, which enables nearly real-time responses (*e.g.,* 2 minutes from the generation of task queries), and Bernstein *et al.* [24] proposed the *retainer model*, which further reduces the latency to a few seconds. In this thesis, although our Method A and C are crowd-powered systems and thus small latency is desirable, efforts in this direction is out of our scope.

Another issue in crowdsourcing is monetary cost, and crowd-powered systems might be desirable to minimize this. However, simply paying less rewards for the same tasks might cause issues in fairness regarding the minimum wage; see the discussions in Dynamo [121, 49], which is a crowdsourcing guideline for academic requesters. While we avoid to query unnecessary tasks to reduce monetary (and timing) cost, currently we do not try to optimize the rewards setting, which is out of the scope of this thesis.

2.4.3 Crowdsourced Human Computation

We define *crowdsourced human computation* as a form of human computation where human processors are employed via microtask-based crowdsourcing. This means that programmers can embed "oracles" requiring human intelligence into their codes like as typical function calls. Little *et al.* [87] presented *TurKit Script*, a programming API for developing algorithms using crowdsourced human computation (which they call *human computation algorithms*). See Figure 2.6 (Left) for an example.

Gingold *et al.* [55] proposed several methods for solving long-standing visual perceptual problems using crowdsourced human computation, including extraction of depth and normal maps for images and detection of bilateral symmetries in photographs (see Figure 2.6 (Right)). Their image-understanding algorithms are designed to *decompose* the original difficult problem into a set of easy perceptual microtasks, *solve* the perceptual microtasks using crowdsourcing, and then *recompose* the responses from crowds using some computational techniques. In our Method A and C, we want to solve computationally complex problems (*i.e.,* regression and optimization of goodness functions), so that we seek how to decompose the problems into easily solvable microtasks, and how to recompose the crowds responses for our problem settings.

2.5 Parametric Spaces in Visual Design

Our target application domain can be considered as *parametric design.* We use the term, parametric design, as the design paradigm where visual contents are

```
ideas = []

for (var i = 0; i < 5; i++) {
    idea = mturk.prompt(
        "What's fun to see in New York City?
        Ideas so far: " + ideas.join(", "))
    ideas.push(idea)
}

ideas.sort(function (a, b) {
   v = mturk.vote("Which is better?", [a, b])
   return v == a ? -1 : 1
})
```

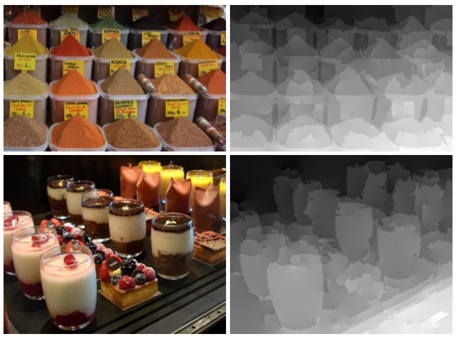


Figure 2.6: Examples of crowdsourced human computation. (Left) An example of TurKit Script [87] for organizing ideas using crowdsourced human computation. (Right) Results of the crowd-powered algorithm of extracting a depth map from an image [55].

solely controlled by a set of (either continuous or discrete) parameters. That is, given a set of parameters, a visual content is solely determined. Also, in parametric design, the number of parameters is often reasonably small so that designers can manually tweak them. For example, to adjust the tone of a photograph, designers tweak several sliders such as "brightness" and "contrast", not tweaking RGB values of every pixel one by one; this is considered that the design space here is parametrized by several degrees of freedom (*i.e.,* sliders), and thus this is considered a parametric design.

Parametric design can be found almost everywhere in visual design production. In Adobe Photoshop Lightroom CC [9, 10], which is used for photo enhancement, there are tens of sliders that control color enhancement. After Effects [7], which is a motion graphics software, has many visual effect options each of which is usually controlled by a few sliders. 3D graphics tools, such as Unity [150] and Maya [16], require designers to adjust sliders for, for example, camera effects, material BRDFs, positions of game objects such as point lights, to obtain visually pleasing designs. In the digital fabrication context, parametric 3D models, also called customizable models, are popular among maker communities [5, 78, 130].

Computer graphics researchers have investigated methods for defining better parametric design spaces. In 3D modeling contexts, human face [29] and body [14, 92] are parametrized using data-driven approaches. Facial expression of virtual characters is often parametrized using *blendshape* techniques [120, 84]. For material BRDF design, Matusik *et al.* [100] proposed a parametric space based on measured data, and Nielsen *et al.* [106] applied dimensionality reduction to the space. Procedural modeling of 3D shapes, *e.g.,* botany [157], is also considered as parametric design in that the shapes are determined by a set of tweak-able parameters. Recently, Yumer *et al.* [166] presented a method for re-designing parametric spaces for procedural modeling using autoencoder neural networks; their method produces lower-dimensional, more intuitive design spaces. Some 2D texture images can be designed by adjusting a few sliders, using procedural texture techniques such as Perlin noise [113, 114]. One of the recent trends is to define parametric spaces based on semantic attributes for facilitating intuitive exploration; this direction has been investigated for, *e.g.,* shape deformation [167], cloth simulation [131], and human body shape [135].

Although we test our methods mainly using the example of photo color enhancement, we envision to support (or automate) various parameter tweaking scenarios in visual design such as ones discussed above. Thus, we try to make our methods as general as possible by not relying on domain-specific formulations. Note that it is possible to extend our methods to be combined with domain-specific heuristics, which we leave as future work.

2.6 Summary

Our computational design methods are distinct from many of such existing methods in that ours consider perceptual aesthetic preference as the design criterion, and also in that we intend not to rely on domain-specific heuristics. Our solutions are based on novel parameter tweaking interfaces utilizing estimated preference (the methods described in Chapter 3 and Chapter 4), and effective usages of crowdsourced human computation (the methods described in Chapter 3 and Chapter 5). As parametric design is popular in many visual design domains, our potential applications can be broader than those demonstrated in this thesis.

Chapter 3

Crowd-Powered Parameter Preference Estimation

In this chapter, we describe the crowd-powered estimation method, a new method to construct a *goodness function* that computes the goodness value of a given parameter set in the target design space. In other words, this method analyzes a high-dimensional design parameter space to estimate a distribution of human preference. This method uses crowdsourced human computation to gather pairwise comparisons between various parameter sets. The estimated goodness function enables two interfaces for facilitating design 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 tested this method in four applications with different design parameter spaces.

3.1 Introduction

Exploring possible visual designs by tweaking parameters is a common practice when designing digital contents, as discussed in Chapter 1 and Section 2.5. This parameter tweaking task can essentially be considered an iterative optimization process performed manually. However, this process is often tedious and time-consuming, especially when there are many parameters to adjust, because the high dimensionality of the design space makes it exponentially difficult to visit all possible ones. We aim at providing computational supports for this situation.

We present a new method to facilitate manual parameter tweaking for visual design, by utilizing computational techniques for estimating parameter preference. In the proposed approach, we use *crowdsourced human computation* to gather human-generated preference data, *i.e.,* pairwise comparisons between various parameter sets. Users first specify a set of target sliders to be tweaked. The system then analyzes the parameter space using crowdsourced human computation and constructs a *goodness function*, which is a function that takes a parameter set as input and quantifies how aesthetically good it is. We then use the goodness function to enable two novel interfaces: *Smart Suggestion* (Figure 3.1) and *VisOpt Slider* (Figure 3.2). Smart Suggestion is an interface that can provide users with appropriate parameter set choices as suggestions, based on their estimated goodnesses. VisOpt Slider is an extension of the conventional slider component, in which the distribution of goodness values is directly visualized on sliders, and it dynamically updates the visualization on the basis of the currently chosen

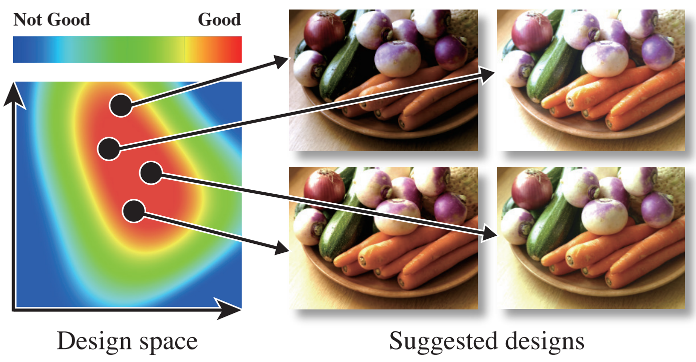


Figure 3.1: Smart Suggestion. The user can obtain appropriate parameter sets as suggestions, which are generated considering goodness of designs.

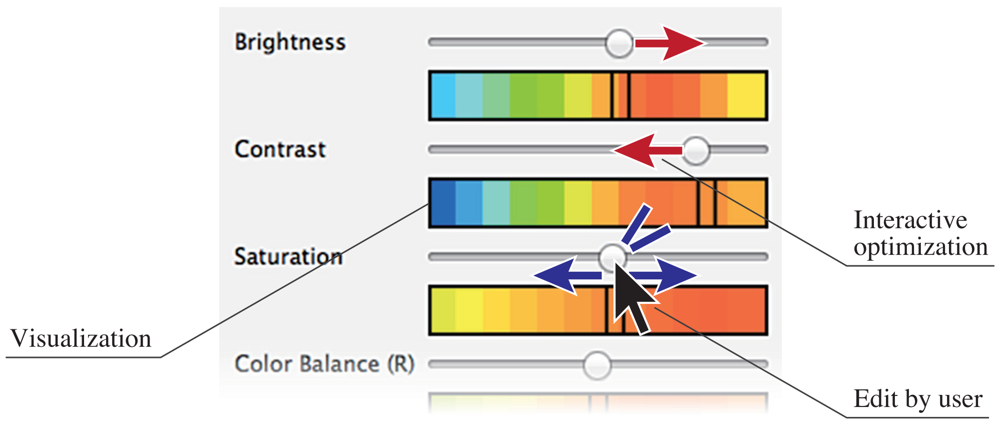


Figure 3.2: VisOpt Slider. The user can adjust each parameter effectively by the visualization (Vis) near the slider and the optimization (Opt) that gently guides the current parameters to the optimal direction.

slider values. In addition, the slider values can be 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 chapter, we offer two specific contributions:

- A framework and techniques to estimate a preference distribution in the target design space, *i.e.,* a goodness function. Here we utilize crowdsourced human computation to gather necessary preference data, which is a set of pairwise comparisons of possible designs.
- Two specific user interfaces for facilitating users' manual exploration of the design space, Smart Suggestion and VisOpt Slider, both of which are enabled by the estimated goodness function.

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

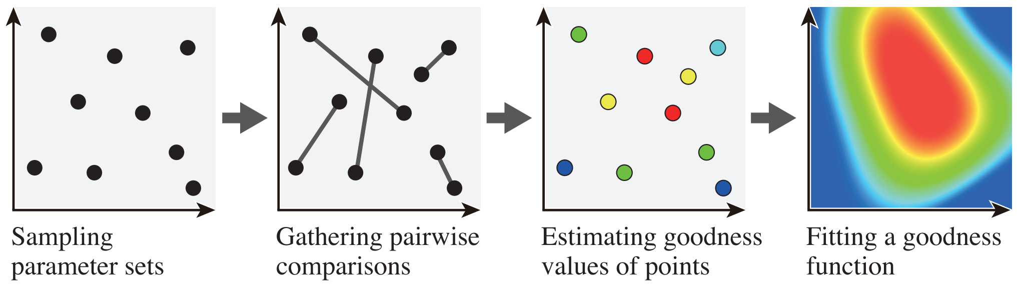

Sampling parameter sets
Gathering pairwise comparisons
Estimating goodness values of points
Fitting a goodness function

Figure 3.3: Overview of our crowd-powered analysis algorithm.

3.2 Crowd-Powered Parameter Analysis

3.2.1 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 thesis is an n-dimensional vector $\mathbf{x} \in \mathcal{X} = [0, 1]^n$ 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 are 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, $g : \mathcal{X} \rightarrow \mathbb{R}$ that maps a parameter set to its estimated goodness; that is, to obtain a function $g(\cdot)$ that takes a parameter set $\mathbf{x}$ as input and computes an estimated goodness value $y = g(\mathbf{x})$ as output. In this chapter, 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 3.3. First, the system generates sampling points on the high-dimensional parameter space (displayed here as a 2-dimensional space for simplicity). Second, using crowdsourced human computation, 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.

Our method can be considered as one of the learning-from-crowd problems [141, 124, 39], 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.

3.2.2 Sampling Parameter Sets

First, the system samples M parameter sets $\mathbf{x}_1, \dots, \mathbf{x}_M$ from the parameter space $\mathcal{X}$ 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 repeats this process M times.

There might be smarter ways to sample parameter sets to achieve more effective crowdsourcing. For example, Secord *et al.* [124] 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.* [141] 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 serves sufficiently well for our requirements.

3.2.3 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 [124, 55, 39] 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 (from 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 $\mathcal{P}$ be a set of pairs of indices $\{(1, 2), \dots, (M-1, M)\}$. For each $(i, j) \in \mathcal{P}$, 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 or her to rate its relative score. Eventually, it gathers $M/2$ scores, and each image is shown and rated once.

The instruction of the microtask 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 appropriate one from the 5 options below.*"

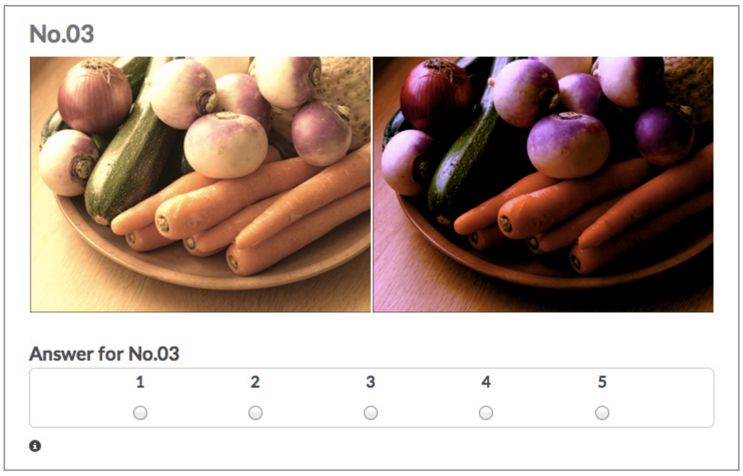


Figure 3.4: A screen capture of the microtasks used in our method. We took a pairwise comparison approach with 5-pt Likert scale rating.

In accordance with the purpose and the content, the user gives a noun, an adjective such as "good" or "natural", and a clause that 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 3.4 shows a screen capture of a microtask that was actually used in our experiments.

To control the quality of the data, we take the *duplicate* approach, as do several of the previous works [124, 55, 39]. 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 chapter, we represent $\mathcal{P}'$ as a subset of $\mathcal{P}$, each of which have passed the quality check, M' as the number of sampling points that have passed the quality check, and $\mathcal{Q} = \{q_1, \ldots, q_{M'}\}$ as a set of indices of sampling points that have passed the quality check.

3.2.4 Estimating Goodness Values of Sampling Points

Given the relative scores, the next goal is to compute the absolute goodness values $\mathbf{y} = [y_{q_1} \ \cdots \ y_{q_{M'}}]^T$ at the sampling points $\mathbf{x}_{q_1}, \ldots, \mathbf{x}_{q_{M'}}$. Sýkora *et al.* [137] 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.* [55] presented a robust algorithm for this problem, but it can only handle relative orders of adjacent areas, and their 2D-based algorithm cannot

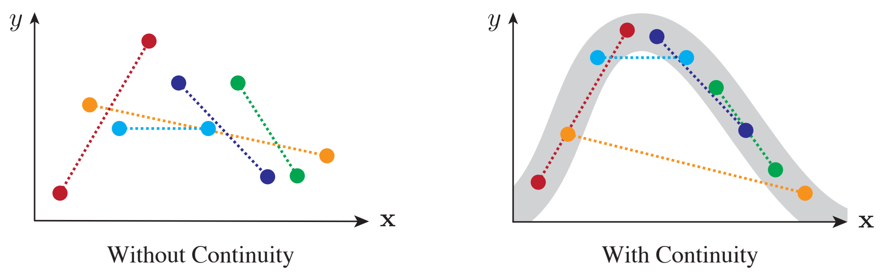


Figure 3.5: Effect of the continuity constraint. This constraint ensures that the estimated goodness values are continuously distributed.

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

$$E_{\text{relative}}(\mathbf{y}) = \sum_{(i,j)\in\mathcal{P}'} \|y_i - y_j - d_{i,j}\|^2 , \tag{3.1}$$

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

$$d_{i,j} = \begin{cases} 1 & (\text{relative score} = 1) \\ 0.5 & (\text{relative score} = 2) \\ 0 & (\text{relative score} = 3) \\ -0.5 & (\text{relative score} = 4) \\ -1 & (\text{relative score} = 5) \end{cases} . \tag{3.2}$$

Considering only the relative constraint results in a disconnected, jagged distribution of goodness values (Figure 3.5 (Left)). This is undesirable because we are handling continuous parameter space, and thus we expect the goodness function to also be continuous (Figure 3.5 (Right)) and smooth in most cases. We therefore add a constraint based on the assumption of continuity:

$$E_{\text{continuous}}(\mathbf{y}) = \sum_{i\in\mathcal{Q}} \left\| y_i - \frac{1}{|\mathcal{N}_i|} \sum_{j\in\mathcal{N}_i} y_j \right\|^2 , \tag{3.3}$$

where $\mathcal{N}_i$ stands for the set of neighborhoods of the i-th sampling points. In our implementation, $\mathcal{N}_i$ is defined by k-nearests in Euclidean distance ($k = 20$). Note that this continuous constraint is popular in many Laplacian frameworks such as mesh smoothing [133, 104], and is beneficial for ensuring that the distribution of the reconstructed values is spatially smooth.

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

$$\min_{\mathbf{y}} \{E_{\text{relative}}(\mathbf{y}) + \omega E_{\text{continuous}}(\mathbf{y})\} , \tag{3.4}$$

where $\omega > 0$ is a parameter that defines the balance of these two constraints. In all experiments introduced in this chapter, we set $\omega = 5.0$.

Using matrix form, the minimization problem can be written as

$$\min_{\mathbf{y}} \left\{ \|\mathbf{A}\mathbf{y} - \mathbf{d}\|^2 + \omega \|\mathbf{L}\mathbf{y}\|^2 \right\}, \tag{3.5}$$

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 such that $d_i = d_{i,j}$ if $(i,j) \in P'$ and otherwise 0, and $\mathbf{L}$ is the graph Laplacian matrix. By setting the derivative with respect to $\mathbf{y}$ to zero, the following linear equation is obtained:

$$\left(\mathbf{A}^T \mathbf{A} + \omega \mathbf{L}^T \mathbf{L} \right) \mathbf{y} = \mathbf{A}^T \mathbf{d}. \tag{3.6}$$

This provides the optimal solution of the minimization problem and could easily be solved by applying standard techniques such as Cholesky decomposition. Here we can also apply *sparse* Cholesky decomposition as $\mathbf{A}^T\mathbf{A} + \omega\mathbf{L}^T\mathbf{L}$ is sparse.

After solving this minimization problem, we linearly normalize the solution so that the maximum goodness value is 1 and the minimum is 0.

3.2.5 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 [28, 15]. 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 $g(\cdot)$ as

$$g(\mathbf{x}) = \sum_{i \in \mathcal{Q}} w_i \phi(\|\mathbf{x} - \mathbf{x}_i\|), \tag{3.7}$$

where $\phi(\cdot)$ is a radial basis function, and $\mathbf{w} = \begin{bmatrix} w_{q_1} & \cdots & w_{q_{M'}} \end{bmatrix}^T$ are the RBF weights that we are going to compute. We used $\phi(r) = r$ as the basis function but found the Gaussian kernel 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 [28, 15]. Specifically, we solve the following minimization problem to obtain $\mathbf{w}$:

$$\min_{\mathbf{w}} \left\{ \sum_{i \in \mathcal{Q}} \left\| \sum_{j \in \mathcal{Q}} w_j \phi\left(\|\mathbf{x}_i - \mathbf{x}_j\|\right) - y_i \right\|^2 + \lambda \|\mathbf{w}\|^2 \right\} \tag{3.8}$$

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 [37] to improve computational efficiency in interactive use.

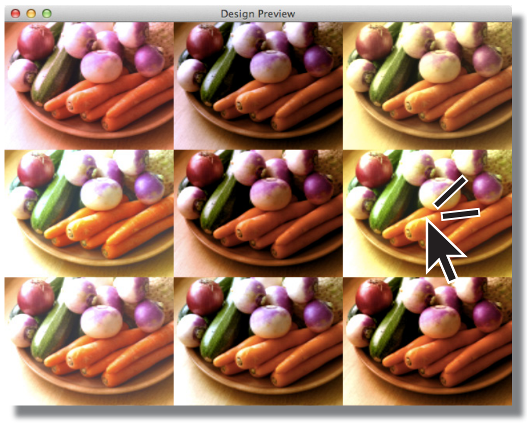


Figure 3.6: Appearance of Smart Suggestion interface. Nine candidates are presented in a 3×3 grid and the user can choose one by clicking it.

3.3 User Interface

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

3.3.1 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* [13] or the *dispersion* technique [97] 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.

3.3.2 VisOpt Slider

The VisOpt Slider (Figure 3.2) displays colored 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. That is, when a user starts to drag a slider, the other sliders' ticks also start to move simultaneously to a better direction according to the user's manipulation.

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 3.7). When the user modifies a certain parameter, the visualizations

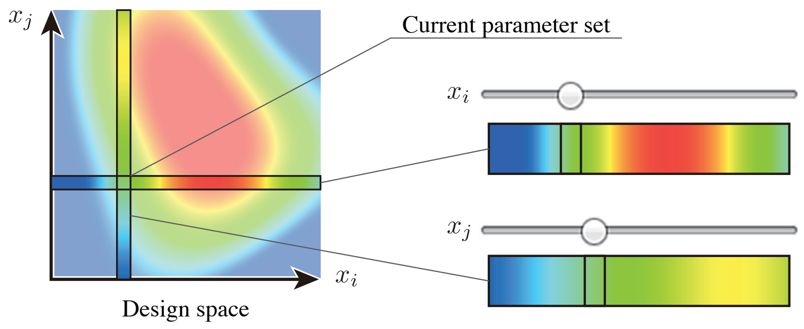


Figure 3.7: Visualization function in VisOpt Slider. Each colored bar shows the distribution of goodness values along the corresponding slider.

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 find better parameter sets quickly but also explore the design space effectively without visiting "bad" designs.

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 ascent method with a fixed number of iterations, following Prévost *et al.*'s and Umetani *et al.*'s work [115, 149]. 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

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla g(\mathbf{x}), \tag{3.9}$$

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 $\alpha = 0.005$.

3.4 Applications

We tested 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 [2] as the platform of crowdsourcing. Table 3.1 lists a summary of the crowdsourcing statistics. We analyzed each parameter space using all the "valid" comparisons we obtained as a result of crowdsourcing.

Table 3.1: Statistics of crowdsourcing in the 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 photo color enhancement, we paid $200 \times 0.02 = 4.00$ USD for workers in total. The number of valid comparisons indicates $|\mathcal{P}'|$, 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]
Photo Color Enhancement	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

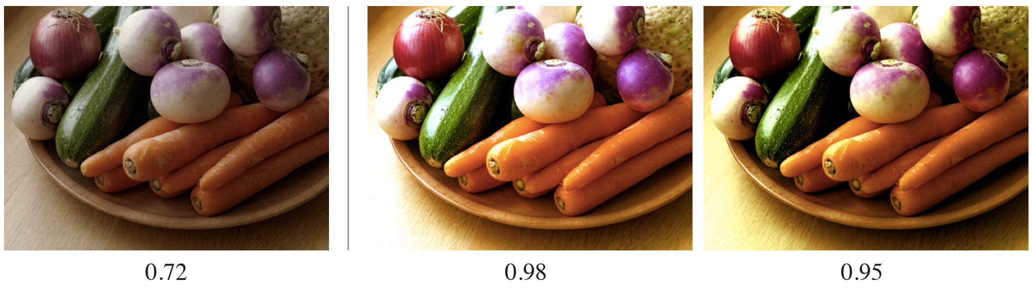


Figure 3.8: Original input image and images generated by Smart Suggestion in the photo color enhancement application. (Left) The original input image ($g(\cdot) = 0.72$). (Middle and Right) Images generated by Smart Suggestion ($g(\cdot) = 0.98, 0.95$).

3.4.1 Photo Color Enhancement

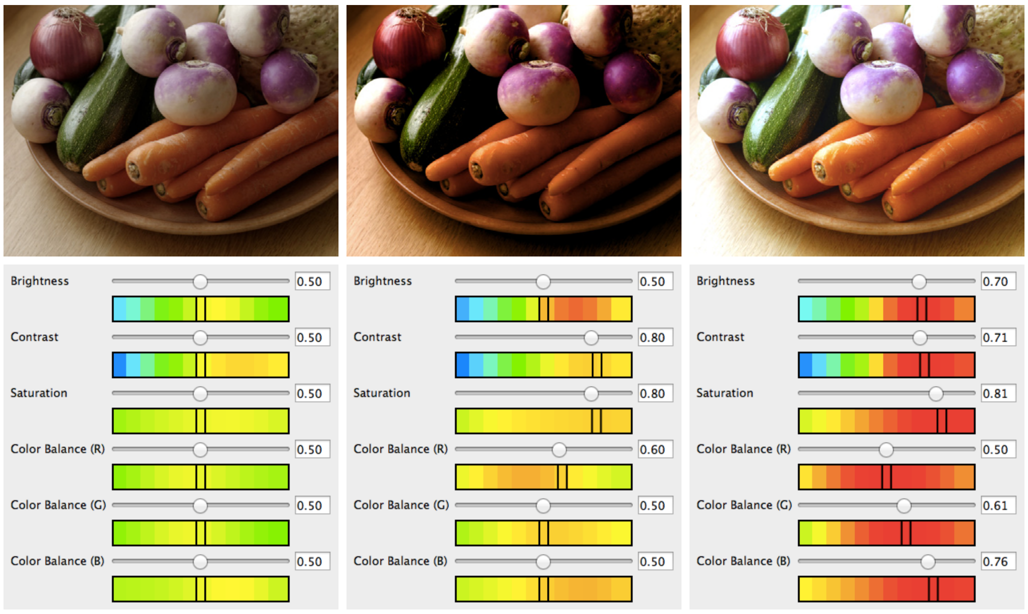
When enhancing colors of digital photos, users have to tweak many parameters, including unintuitive ones that are difficult for novices to understand. To facilitate this task, we selected six popular parameters for this application: brightness, contrast, saturation, and color balance (Red, Green, and Blue). For this experiment, we chose a photograph of vegetables (Figure 3.8 (Left)) from a photo sharing service (Morguefile [4]). In the crowdsourced microtasks, we asked crowd workers to choose the photograph that would be better to use in a magazine or product advertisement.

Figure 3.8 (Middle and Right) shows typical images suggested by Smart Suggestion. Examples of VisOpt Slider visualizations with typical parameter sets are shown in Figure 3.9. 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.

3.4.2 Camera and Light Control

Secord *et al.* [124] presented a computational perceptual model for predicting goodness of viewing directions for 3D models; however, their 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 consisting of a 3D model, a perspective camera, and a point light. There are eight parameters to tweak in total, 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 (which is almost symmetric), and the bunny model (which is asymmetric). The orientation of the camera is automatically set such 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 (*i.e.*, camera.$z < 0.0$, see Figure 3.10 (Middle)) from the right side (*i.e.*, camera.$z > 0.0$, see Figure 3.10 (Left)) of the dragon model, the visualization tells us that we should also move



Brightness 0.50	Brightness 0.50	Brightness 0.70
Contrast 0.50	Contrast 0.80	Contrast 0.71
Saturation 0.50	Saturation 0.80	Saturation 0.81
Color Balance (R) 0.50	Color Balance (R) 0.60	Color Balance (R) 0.50
Color Balance (G) 0.50	Color Balance (G) 0.50	Color Balance (G) 0.61
Color Balance (B) 0.50	Color Balance (B) 0.50	Color Balance (B) 0.76

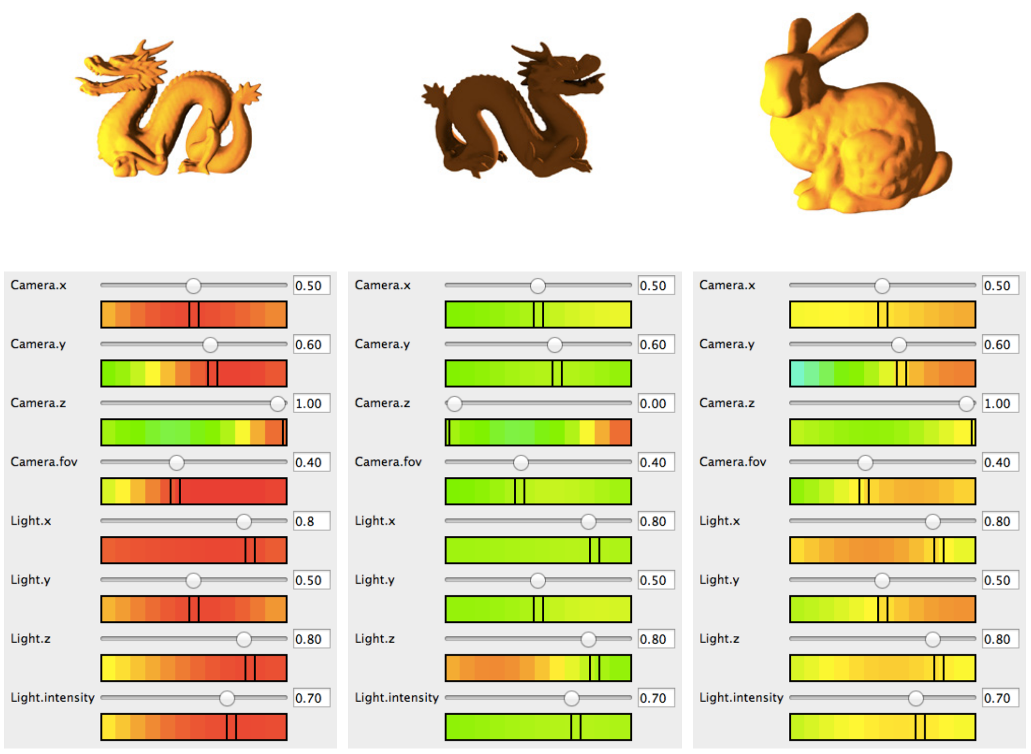
Figure 3.9: Designs and visualizations of goodness distributions in the photo color enhancement application.

the light to the left side (*i.e.*, light.$z < 0.0$) so that the model is adequately lit. Figure 3.10 (Right) shows the views using the bunny model, where we applied exactly the same parameter sets as to Figure 3.10 (Left). It is observed 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.

3.4.3 Shader (Material BRDF)

As Talton *et al.* [139] and many other researchers 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". 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 in Unity Asset Store, Hard Surface Shaders Free [3]. We applied this shader to a teapot model. We chose all the eight parameters provided by the shader, 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 3.11 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 3.11) 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



Camera.x	0.50	Camera.x	0.50	Camera.x	0.50
Camera.y	0.60	Camera.y	0.60	Camera.y	0.60
Camera.z	1.00	Camera.z	0.00	Camera.z	1.00
Camera.fov	0.40	Camera.fov	0.40	Camera.fov	0.40
Light.x	0.8	Light.x	0.80	Light.x	0.80
Light.y	0.50	Light.y	0.50	Light.y	0.50
Light.z	0.80	Light.z	0.80	Light.z	0.80
Light.intensity	0.70	Light.intensity	0.70	Light.intensity	0.70

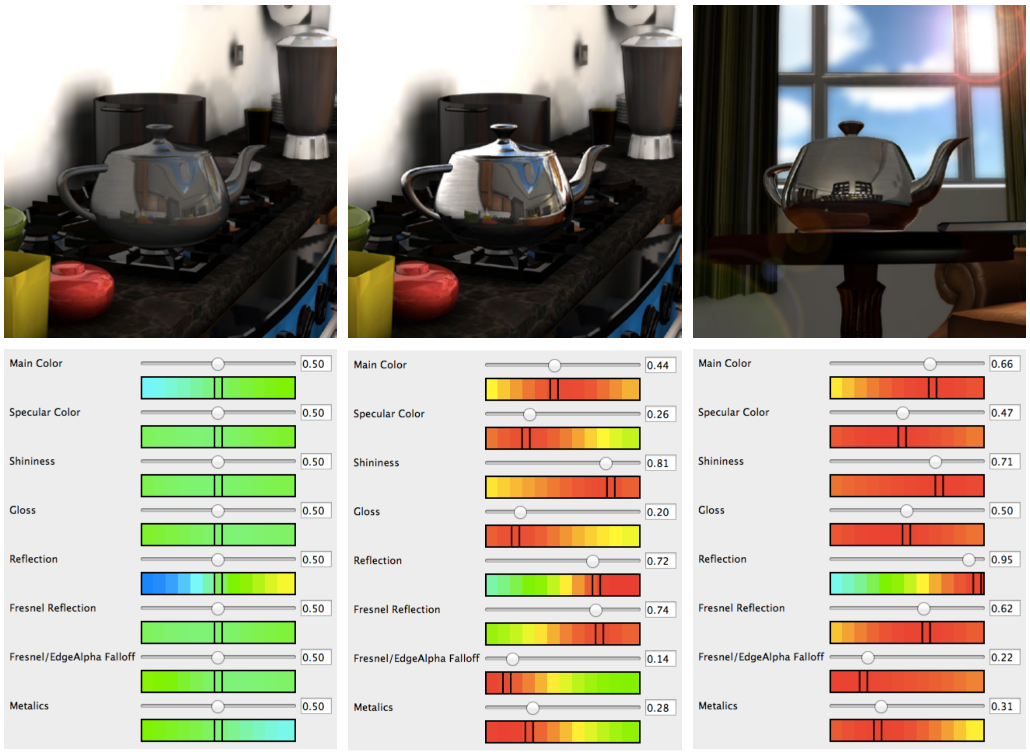
Figure 3.10: Designs and visualizations of goodness distributions in the camera and light application. (Left) A parameter set is applied to the dragon scene, whose camera.z (the third slider) is set to the maximum value. (Middle) Another parameter set whose values are the same as the left one except that camera.z is modified to the minimum value. (Right) The same parameter set as the left one is applied to the bunny scene.

was 0.67 in the kitchen scene had a goodness value of 0.93 in the window seat scene.

3.4.4 Blendshape Facial Expression

Blendshape is a standard approach to control the facial expressions of virtual characters [84], 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.* [84], 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 [120], and 5 for jaw control). We suppose that the goodness in this design scenario can be approximated as the validity of the facial expression, so that we asked crowd workers to choose the better "natural (unbroken)" expression.

Figure 3.12 shows typical designs and their estimated goodness values. We believe that the goodness function is successfully constructed and gives reasonable



Main Color 0.50
Specular Color 0.50
Shininess 0.50
Gloss 0.50
Reflection 0.50
Fresnel Reflection 0.50
Fresnel/EdgeAlpha Falloff 0.50
Metalics 0.50

Main Color 0.44
Specular Color 0.26
Shininess 0.81
Gloss 0.20
Reflection 0.72
Fresnel Reflection 0.74
Fresnel/EdgeAlpha Falloff 0.14
Metalics 0.28

Main Color 0.66
Specular Color 0.47
Shininess 0.71
Gloss 0.50
Reflection 0.95
Fresnel Reflection 0.62
Fresnel/EdgeAlpha Falloff 0.22
Metalics 0.31

Figure 3.11: Designs and visualizations of goodness distributions in the shader application. (Left and Middle) Kitchen scene. (Right) Window seat scene.

values for even this high-dimensional application. Unlike random suggestions (typical goodness values are between 0.3 and 0.6), where most of expressions are broken, Smart Suggestion can provide relatively better starting points (around 0.7) with a high enough quality to inspire users. The optimization function in VisOpt Slider also works well with this large number of parameters to avoid wasting time with "invalid" expressions. However, we found that the colored visualization was often useless because every visualization bar tends to show a similar color along its axis in this large parameter space.

3.5 Evaluation

We evaluated our method from two aspects: the quality of the estimation of the goodness function, and the usability of the user interfaces. As we do not have a "ground truth" of the goodness function, we indirectly evaluated the estimation quality by checking its convergence behavior with respect to the number of data, and its prediction ability by a hold-out test. For evaluating the usability, we conducted an informal user study.

3.5.1 Quality of Analysis

Convergence with respect to the 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

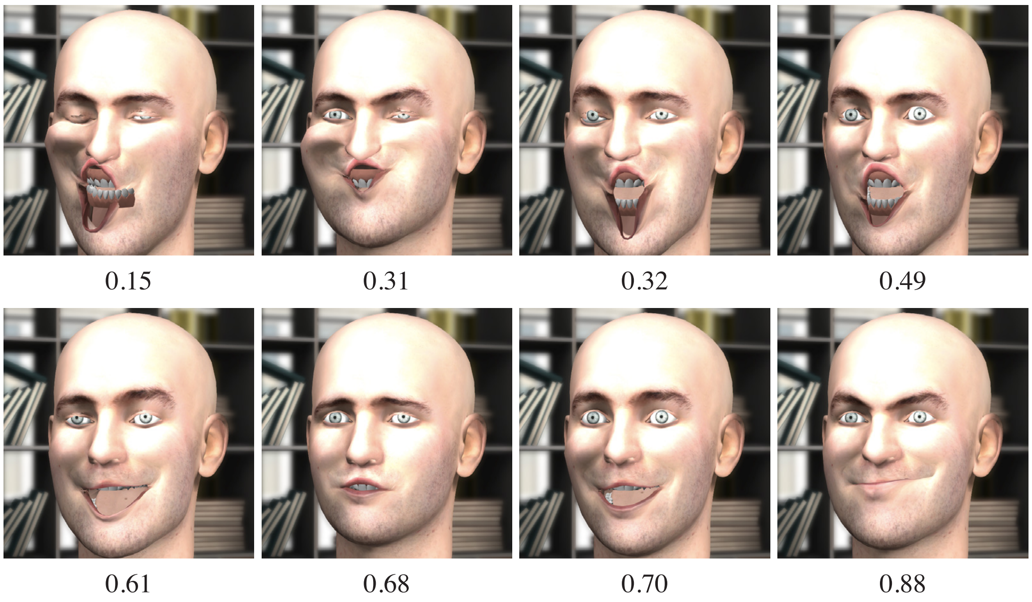


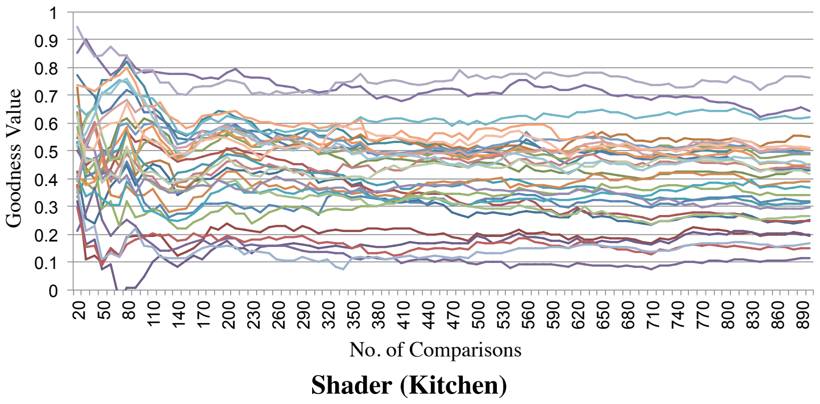
Figure 3.12: Typical facial expression designs controlled by 53 blendshape parameters. The values shown below pictures are the corresponding goodness values computed by our method.

important feature of the parameter space. Here, we discuss the convergence behavior graphs (Figure 3.13) 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 constructed 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 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 were required to be stable.

Adequacy of Estimated Goodness Functions

To check the adequacy of the estimated goodness functions, we applied a hold-out test to the acquired data set for the photo color enhancement, the shader, and the facial expression applications. We randomly selected 100 samples from the valid comparisons $\mathcal{P}'$ obtained via crowdsourcing as a testing set, which we denote $\mathcal{P}'_{\text{testing}}$. Then, we trained the goodness function $g(\cdot)$ by using the rest of the valid comparisons $\mathcal{P}'_{\text{training}} = \mathcal{P}' \setminus \mathcal{P}'_{\text{testing}}$ as a training set. The sizes of training data set are 995 for the photo color enhancement, 807 for the shader with the kitchen scene, and 1671 for the facial expression. If the estimated goodness function $g(\cdot)$ obtained from $\mathcal{P}'_{\text{training}}$ is adequate, it should be able to predict the



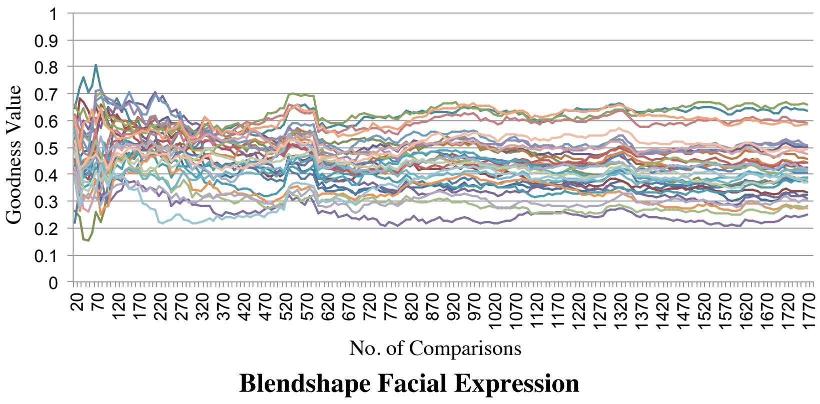


Figure 3.13: Convergence behavior of estimated goodness values with respect to the number of comparison data. Each curve shows the transition of the goodness value on a randomly selected parameter set.

distribution of the scores of the comparisons in $\mathcal{P}'_{\text{testing}}$. That is, as for a paired comparison in the testing set $(i, j) \in \mathcal{P}'_{\text{testing}}$, the difference between the estimated goodness values $g(\mathbf{x}_i) - g(\mathbf{x}_j)$ should be correlated to the score (1–5) given to the pair by a crowd worker. We show such plots in Figure 3.14. 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 is likely to rate the i-th design higher than the j-th design when the goodness value for i-th design (*i.e.*, $g(\mathbf{x}_i)$) is higher than that for j-th design (*i.e.*, $g(\mathbf{x}_i)$). This indicates that the trained goodness function $g(\cdot)$ successfully predicted the relative goodness values of paired comparisons that do not appear in the training set.

3.5.2 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) [19] was 77.5 on average (SD = 11.6),

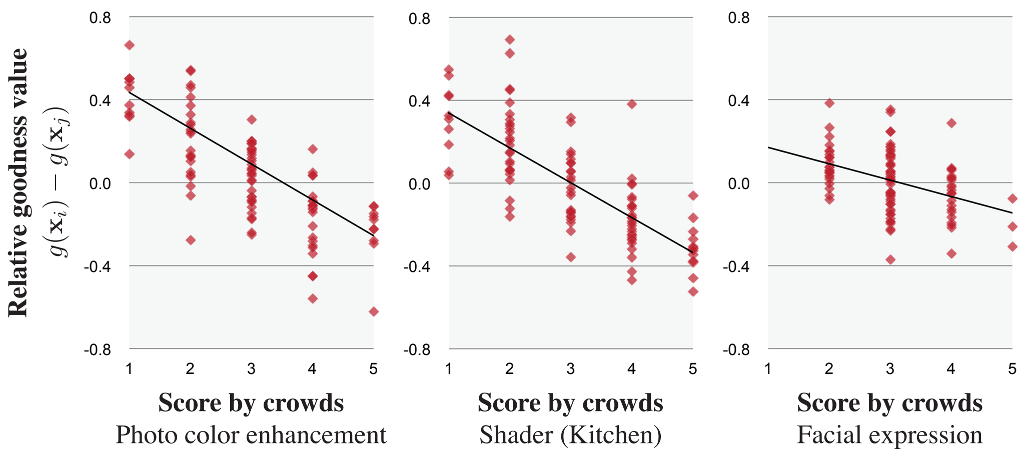


Figure 3.14: Correlation between the scores for pairwise comparisons by crowds and the estimated relative goodness values. Black lines are the fitted lines. Note that the plotted data are not included for estimating the goodness function $g(\cdot)$.

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).

3.6 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 during controlling one parameter for the camera. In the case of photo color enhancement, there are some automatic enhancement algorithms (*e.g.*, [35]); however, such algorithms typically cannot consider the semantic 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 typical automatic methods.

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 or 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.

3.6.1 Other Possible Representations of Goodness Function

In this work, we chose a non-parametric representation based on RBF network as a representation of goodness function. The role of this function is similar to the *ranking function*, for example, proposed by Chaudhuri *et al.* [39], where they represented it as a simple weighted sum of the input vector. Secord *et al.* [124] 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.* [124] 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 user interfaces.

The method proposed by Chu and Ghahramani [41] may be applied instead of ours for representing the goodness function. Their method assumes a Gaussian process prior on the goodness function, and estimates the function from pairwise-comparison data. While their method only handles data based on two-alternative forced choice (2FAC), our method can handle data based on n-pt Likert scales (in the experiment, we chose $n = 5$), which is potentially more informative than 2FAC. As an additional informal experiment, we compared estimated functions using their method and ours with $n = 2$ (this is equivalent to 2FAC), from the same synthetic data; we did not observe any significant difference between them.

3.6.2 Limitation

In this approach, every different image task requires re-crowdsourcing, and this limits the scalability; to improve this, it is possible to combine multiple crowd-sourced data sets or reusing previous ones. Another notable limitation is that the user has to make the instruction for crowdsourcing. This was not evaluated in the user study. Instruction templates can be utilized, but this part cannot be fully 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 (see Table 3.1). This might be a problem for users who cannot wait and want to obtain the results immediately. To reduce this latency, real-time crowdsourcing techniques [24] might be effective. 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.

3.6.3 Design Implication

Learning the *personal* preference of the users could be an interesting future direction, and we will seek such a method in Chapter 4. Because human preferences differ depending upon individuals and cultures, considering clusters of crowds [69] 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 [141]. Web design and presentation slide design contain many discrete parameters such as fonts so they are out of our scope, but considering such discrete parameters is an important direction to take in our future work. Similar to AttribIt [39], we feel that considering two or more types of goodness criteria for a design task could enable more useful interfaces. Generating a small number of new sliders by using dimensionality reduction techniques is also a potential focus of our future work.

Chapter 4

History-Based Parameter Preference Estimation

In this chapter, we describe the history-based estimation method, which learns "personal" aesthetic preference from the editing history of a target user and then supports the user to manually explore the design space by using the estimated preference. To investigate this concept and validate its effectiveness in practical scenarios, we focus on photo color enhancement application, specifically, the scenario that a photographer has tens of or hundreds of photographs to be enhanced. To support this repetitive task, we present a new workflow for color enhancement, called *self-reinforcing color enhancement*, for effectively gathering and utilizing editing history. In this workflow, the system *implicitly* and *progressively* learns the user's preference by training on their photo editing history, which means that the more photos the user enhances, the more effectively the system supports the user. Based on this workflow concept, we present a working prototype system called *SelPh*, and describe the algorithms to implement this system. We also report the results of a user study for investigating how photographers use this system in the targeted scenarios.

4.1 Introduction

In Chapter 3, we investigated how to estimate a "general" preference distribution by using crowdsourced human computation. Knowing such general trends is useful for designers in many scenarios, such as the case of designing a visual advertising for general audience. On the other hand, some designers or artists might have highly personalized preference, and it may be more effective to support them with estimating their personal preference in some situations; for example, a skilled designer might have a specific style, and want to design a set of visuals consistently based on the style. In this chapter, we are aimed at seeking a computational design method that estimates such "personal" preference for facilitating design exploration.

As the source of personal preference data, we use the editing history of a single user. However, unlike crowdsourced data generation, editing history cannot be generated on demand. Thus, we need a specialized workflow for effectively gathering and utilizing editing history. To investigate this workflow concept and validate its effectiveness in practical scenarios, in this chapter, we will focus on the photo color enhancement scenario. More specifically, we consider the scenario that a photographer has tens of or hundreds of photographs to be enhanced.

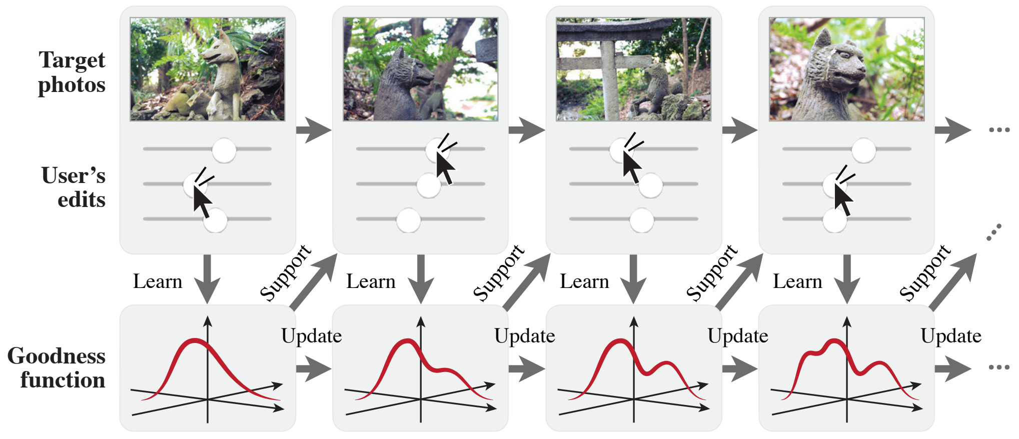


Figure 4.1: Illustration of our workflow, named *self-reinforcing color enhancement.* As more photos are enhanced by the user, the system *implicitly* and *progressively* learns the user's preferences and, as a result, the system is able to support the user in an increasingly effective manner.

When photographers enhance the color of a photo, they need to manually adjust many sliders such as brightness, contrast, and saturation. For skilled photographers, this approach is satisfactory when they have only a few photographs. However, if they have tens of or hundreds of photos to edit, manually adjusting every photograph independently would be an onerous task. This scenario often arises when, for example, a photographer returns from a long journey with a camera and wants to upload a photo album to a website.

One possible solution to avoid this tedious task is to use fully-automatic color enhancement (*auto-enhancement*) to batch the whole process. Commercial software often provides an option to perform this. However, there are several limitations associated with this solution. First, the auto-enhancement functions even in recent commercial software do not satisfy all users because every person has different preference and they do not reflect personal preference [71]. Second, even if personalized auto-enhancement (*e.g.,* [70, 71]) is available, there are still some non-negligible reasons why photographers find this fully-automatic solution unsatisfactory. For example, even state-of-the-art auto-enhancement algorithms cannot satisfactorily edit some types of photos, such as ones where highly semantic aesthetics are involved. Moreover, even if the auto-enhanced photograph appears satisfactory, photographers might still prefer exploring other possible enhancements by themselves to confirm that it is the best indeed, rather than blindly trusting the auto-enhancement. As a result, manually visiting every photograph is inevitable in this scenario.

We investigate a method of supporting this manual repetitive enhancement. The goal of this task is for the user to subjectively assess every enhanced photograph as being optimized, *i.e.,* aesthetically best. To determine whether this goal is met, the user has to view all of the photos and make independent decisions on each one. To facilitate this, we present a new photo enhancement workflow, named *self-reinforcing color enhancement*, where the system *implicitly* and *progressively* learns the user's personal preference from editing history. As the system learns the user's preference, it supports color enhancement more effectively (Figure 4.1). In contrast to most machine learning-based approaches

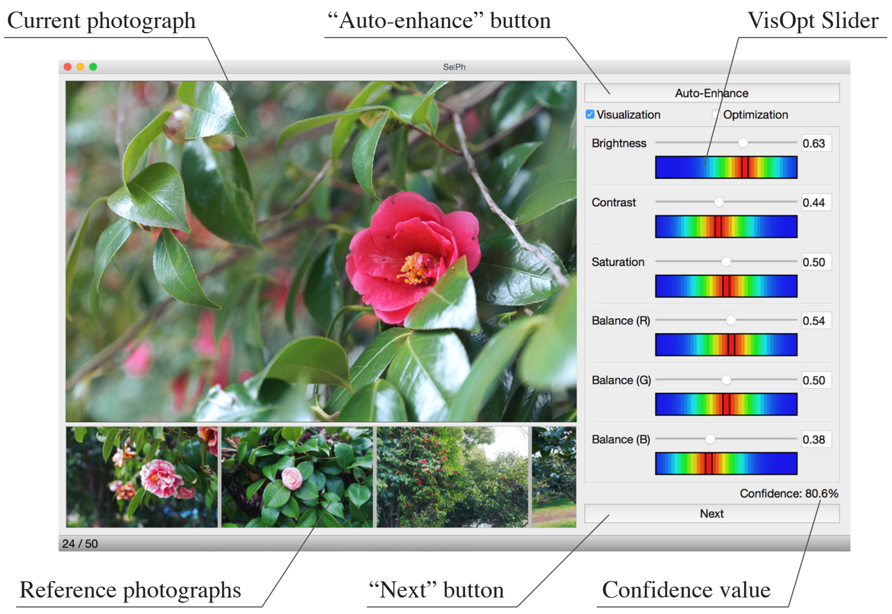


Figure 4.2: Screen capture of our working prototype system, named *SelPh*. The left top part is the preview widget that shows the currently-enhanced photo. The left bottom part is the reference photo widget that shows already-enhanced photos as reference. The right part is the control widget that provides functions for supporting efficient color enhancement.

[70, 26, 35, 71], in our workflow the user does not need to consider the training of the system as a separate preparation process. Instead, the user enhances photographs mostly as usual, but with help from the self-reinforcing system.

Based on the concept of self-reinforcing color enhancement, we present a working prototype system, named *SelPh* (Figure 4.2). The ability of the system to perform self-reinforcement enables it to provide several useful support functions to the user. By using our system, we conducted a user study to investigate how photographers enhance a collection of photos (*e.g.*, a photo album) with a self-reinforcing system, how effectively the self-reinforcement approach works, and the overall level of satisfaction with the system. This chapter reports insights from the user study; for example, all the participants agreed that the workflow with a self-reinforcing system is preferable to the traditional one, and all the participants found the support functions of SelPh to be satisfactory. Participants particularly liked the visualization of confidence of preference estimation, saying that it makes the system more trustworthy and enjoyable. We also discuss the design implications revealed by the study.

The main contribution provided in this chapter is the investigation of a novel computational design method that learns personal preference from editing history and facilitates manual design exploration. This method is tested on the workflow of self-reinforcing color enhancement for aiding repetitive manual enhancement. More specifically, we offer the following three contributions in this chapter:

System design. We designed a prototype system, SelPh, which offers five user support functions, including enhanced sliders and adaptation based on confidence of preference estimation.

Algorithms. To enable these functions, preference estimation and interaction techniques are combined into a system; a new joint-space formulation is introduced, as well as a non-trivial combination of machine learning techniques.

User study. We conducted a qualitative user study by using SelPh and obtained various implications for designing learning-based systems in general. To the best of our knowledge, this is the first study that investigates how photographers enhance photos with a self-reinforcing system.

4.2 Related Work

Since we specifically focus on the scenario of photo color enhancement of many photographs in this chapter, here we review related work specific to this chapter that were not detailed in Chapter 2, to clarify our contributions.

4.2.1 Manual Photo Color Enhancement

Photographs can be enhanced either by interactive methods or automatic methods. Shapira *et al.* [128] presented an interactive method for recoloring, which considers spatial conditions (2D distributions) of colored pixels in addition to the colors themselves, which enables complex color manipulations. Histomage [40] also provides interactive tools for color enhancement, which enable the user to easily and efficiently select spatially varying pixels. In contrast to these approaches, our current method does not take such complex spatial conditions into account; rather, our interest is the scenario where a large number of photographs require enhancement.

The function provided in Adobe Photoshop Elements called Auto Smart Tone [11] is conceptually related to our approach. Although it is named “auto,” the user is expected to manually fix the suggested auto-enhancement for each photo. The system learns from the user’s enhancement, and then uses this learning for suggesting improved auto-enhancement for a new image. This can thus be considered to be self-reinforcing photo color enhancement. Our work makes several contributions on top of this. Our novel formulation for learning users’ preferences (*e.g.,* the joint-space formulation) can be used to develop support functions beyond auto-enhancement, including enhanced sliders and confidence values of preference estimations. As well as developing the system, the other key contribution is the first user study of self-reinforcing systems.

4.2.2 Automatic Photo Color Enhancement

It is often considered (*e.g.,* [35]) that auto-enhancement algorithms used in most packages are based on simple heuristics such as histogram stretching, which cannot work for complex cases. To improve the quality of auto-enhancement, machine-learning techniques are often used [35, 71, 165], of which the most relevant is the personalized auto-enhancement proposed by Kapoor *et al.* [71]. In this formulation, the user trains the system by manually enhancing a set of carefully-selected training photos, and then the system provides an auto-enhancement function that reflects the user’s personal preference. Here, the training phase and the execution phases are completely separated, and it does not provide any support for manual editing. Although our work makes partial use of a similar underlying technique (*i.e.,* metric learning), the interaction workflow is completely different as there is a seamless transition from training to task execution. Our main contribution is in the investigation of this paradigm and the functions to support the seamless transition. In addition, we newly introduce algorithms to enable our support functions, including an algorithm to compute confidence values that are used to adjust the interface behavior, and a joint-space formulation that is necessary for enhanced sliders.

HaCohen *et al.* [59] presented a method to automatically achieve consistency within a collection of photos. This method also allows users to manually correct a small number of photos and then automatically propagates the correction to the other photos in the collection. Berthouzoz and her colleagues [56, 26] presented a method to create content-adaptive photo manipulation macros for batching the process. Similar to ours, this method learns the relationship between photo features and user-specified parameters. Their goal is to enable automatic batch enhancement of a large set of photos, and they do not aim at supporting one-by-one manual editing. Jaroensri *et al.* [65] presented a method to automatically predict the acceptability of a given photo enhancement. Their model is computed using dataset obtained via crowdsourcing in advance, while ours is computed progressively using the personal editing history. Additionally, no previous work evaluates how self-reinforcement can be used to improve the user experience with manual enhancement of individual photos.

4.2.3 Demonstration-Based Techniques

Our method learns the user's preference from the user's demonstration. This can be considered as a derivation of *programming by demonstration* [47, 81, 88]. The concept of programming by demonstration has been examined in previous research on the automation of photograph editing [56, 26]. It is also incorporated into commercial software such as the macro creation in Photoshop (called *Actions*) [12]. While they aim at automating tasks, our focus is to support manual tasks. Also, they usually require explicit training or authoring phases, while our approach seamlessly integrates training and task execution phases. Another domain of demonstration-based techniques is *adaptive user interface* [53, 51], which adapts to the user based on the user's behavior and context to improve usability and performance. While our system also adapts to the user, we aim at facilitating open-ended creative tasks, rather than improving usability of the interface.

4.3 Self-Reinforcing Photo Enhancement System

This section describes our prototype system, called SelPh. SelPh was designed based on the concept of self-reinforcing color enhancement. The underlying algorithms will be described in the next section. Figure 4.1 (b) shows the user interface of SelPh. The left top part is a preview widget that shows the currently-enhanced photo. The left bottom part is a reference widget that shows already-enhanced photos for reference. The right part is a control widget that provides functions for color enhancement. SelPh's basic functionality comprises six sliders that adjust enhancement parameters: brightness, contrast, saturation, and color balance with respect to red, green, and blue. SelPh also provides five user support functions enabled by the self-reinforcement.

User interaction proceeds as follows. First, the user imports all the target photos into SelPh. Then, SelPh displays the first photo, and the user starts enhancement. Once the user is satisfied with the enhancement result, the user pushes the "next" button, and then the next photo appears. This procedure is repeated until all the photos are enhanced. At the beginning of this repetition, the supports by SelPh are not provided or are not very effective; however, as this repetition proceeds, SelPh becomes more and more confident about its preference estimation and the supports become more and more effective. We will describe this behavior with an example later.

4.3.1 User Support Functions

Our self-reinforcement enables the following five functions to be made available to the user:

Visualization of goodness distribution on sliders. The system includes the VisOpt Slider interface presented in Chapter 3, where the system provides a colorful "bar" along each slider. A heat map is displayed to show the distribution of "good" parameters for each bar (Figure 4.3 (a)), where the red (blue) color indicates that it is a good (bad) parameter choice. This colorful visualization is expected to help the user to explore the parameter space more efficiently.

Interactive optimization of slider values. In addition to the visualization, the VisOpt Slider provides an interactive optimization function. When this function is enabled, as the user adjusts a slider value, all the other slider values are simultaneously optimized so that collectively the sliders values give better overall result. This function is useful for getting away from meaningless design spaces, thus reducing the user's effort during his or her exploration.

Variable confidence value. SelPh computes a value called *confidence value*. This indicates the confidence, or the certainty, of the estimation of the user's preference with respect to color enhancement. This confidence value is used to adjust the visualization and optimization functions. For visualization, the system modifies the color scheme as shown in Figure 4.3 (b). For example, when the confidence value is small, the color becomes monotonous (Figure 4.3 (c)). This prevents the system from displaying low quality estimation to the user. For optimization, the system adjusts the strength of the automatic guidance so that more optimization is performed when the confidence value is large, and less is performed when the confidence value is small.

Auto-enhancement. SelPh provides an "auto-enhance" button. Pressing this button automatically sets all the sliders values to the estimated optimal parameters. This optimization is computed using the preference model learned from the user's editing history. As noted, auto-enhancement cannot always work well; however, as our auto-enhancement adapts to the user's

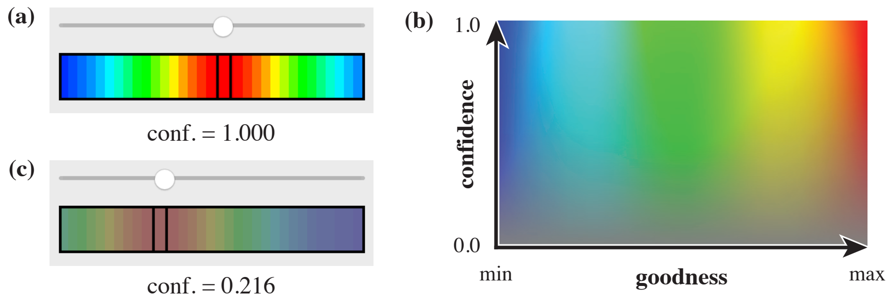


Figure 4.3: Color scheme of the visualization on sliders. (a) Heat-map visualization on sliders (conf. = 1.000). (b) Extended color scheme using the confidence value. (c) Visualization with low confidence (conf. = 0.216).

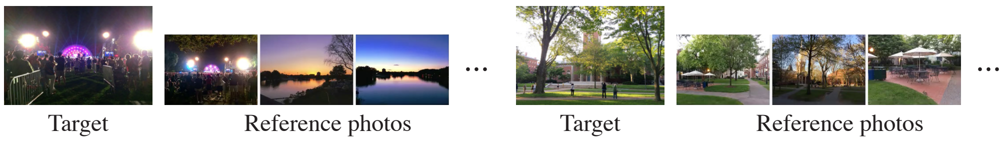


Figure 4.4: Examples of suggested reference photos. The system shows the already-enhanced photos sorted by the similarity to the current target photo.

preference, it is expected to provide a reasonable starting point for further exploration.

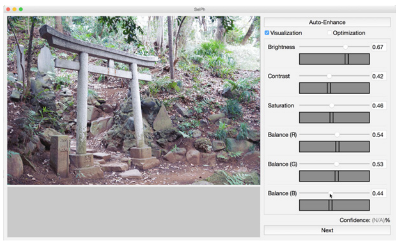
Reference photos. When a collection of photographs is enhanced, consistency among the enhanced photographs must be ensured [59]. To facilitate consistent photo enhancement, the system shows the user *reference photos*, which are the already-enhanced photos adaptively sorted by the estimated similarity to the photo that the user is currently editing. Figure 4.4 shows examples of reference photos. Note that the similarity between photos is personalized; *i.e.,* it is learned from the user's editing history.

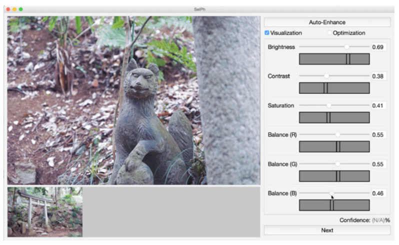
Figure 4.5 shows an example sequence of color enhancement performed by SelPh. For the first and second photos, no user support is provided because the system does not have enough data for reinforcement. From the third photo onwards, the guidance functions are provided. The confidence value increases from the third to the fifth photos (0.359, 0.760, and 0.785). As these photos are similar to each other, the system is able to estimate the user's preference effectively. However, for the sixth photo, which is quite different from any of the photographs that have already been enhanced, the confidence value decreases (0.109), because it is difficult for the system to estimate the enhancement preference based on the previously enhanced photos. For the seventh and eighth photos, which are similar to the sixth photo, the system successfully estimates the enhancement preference with higher confidence (0.432 and 0.865).

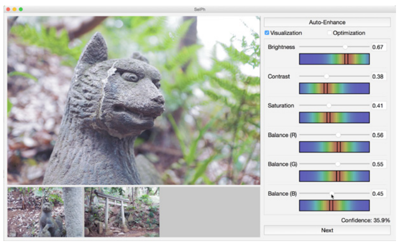
4.4 Algorithms

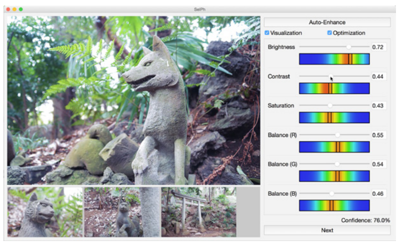
4.4.1 Overview of Self-Reinforcement Procedure

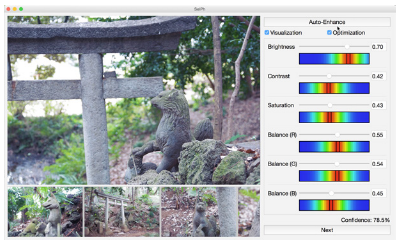
Every time the user finishes an enhancement of a photograph and then pushes the "next" button to go to the next photo, the system computes the self-reinforcement

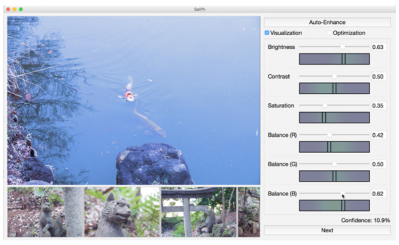

The 1st photo (no conf.)

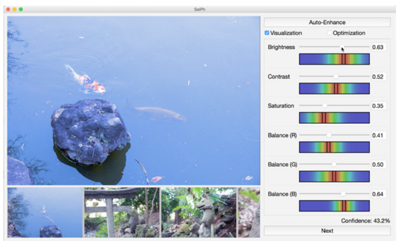

The 2nd photo (no conf.)


The 3rd photo (conf. = 0.359)


The 4th photo (conf. = 0.760)


The 5th photo (conf. = 0.785)


The 6th photo (conf. = 0.109)


The 7th photo (conf. = 0.432)

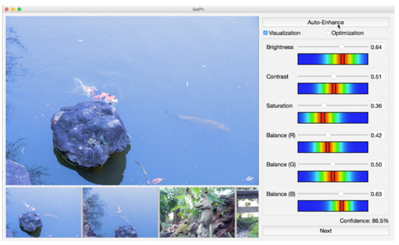

The 8th photo (conf. = 0.865)

Figure 4.5: An example sequence of color enhancement using SelPh.

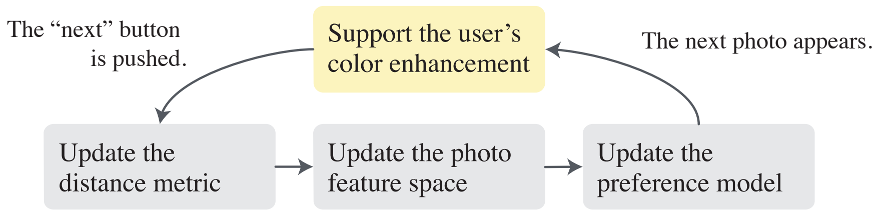

The "next" button is pushed.
Support the user's color enhancement
The next photo appears.
Update the distance metric
Update the photo feature space
Update the preference model

Figure 4.6: Overview of the self-reinforcing procedure. The yellow box is the design session, and the gray boxes are the self-reinforcing sessions.

procedure as shown in Figure 4.6. This procedure consists of the following three steps:

Step 1: Update the distance metric of photographs. The system learns the distances, or dissimilarities, between photographs based on the user's past enhancement history so that the user's personal preference is reflected.

Step 2: Update the photo feature space. The system learns personalized feature vectors as descriptors of photographs. These are computed based on the learned distance metric.

Step 3: Update the enhancement preference model. The preference model for estimating the quality of the enhancement is updated based on the learned photo feature space and the user's past enhancement history.

It is observed that this procedure can be computed in less than 40 milliseconds with maximum 50 photos (MacBook Pro with 3 GHz Intel Core i7). After the computation of this procedure, the next photo is loaded and shown to the user, with the updated user support functions. In the following subsections, we describe these three steps, and then show the way these techniques are used to enable the user support functions.

Assume that the user has just pushed the "next" button. Let m be the number of already-enhanced photos, I_i is then the i-th photo image, and $\mathbf{x}_i \in \mathcal{X} = [0, 1]^n$ is the enhancement parameter set that the user specified for I_i. The value n is the number of parameters (in our case $n = 6$), and the slider's minimum and maximum values are mapped to 0.0 and 1.0 respectively, and $\mathbf{x}^{\text{neutral}} = [\,0.5 \;\; \cdots \;\; 0.5]^T$ is a "neutral" parameter set that does not change the photo appearance. Given the data $\mathcal{D} = \{(\mathbf{x}_i, I_i)\}_{i=1}^{m}$, the goal is to estimate the user's preference of enhancement for the next photo I_{m+1}.

4.4.2 Distance Metric Learning of Photos

The goal in this step is to learn an appropriate distance metric for photos from the available data $\mathcal{D}$. This problem can be seen as a derivative of well-studied *metric learning* techniques [80]. In our case, the goal is to learn a distance function $d(I_i, I_j)$ between an arbitrary pair of photos.

For this purpose, we mostly follow the method presented by Kapoor *et al.* [71]; the distance metric is optimized so that the distances between photos are as proportional to the distances between associated parameters as possible. The distance function d is represented as a weighted sum of 38 types of non-linear distances between low-level photo features, including the symmetric KL-divergence between intensity histograms. The reader is encouraged to consult the original paper for details. Let $\mathbf{w} \in \mathbb{R}^{38}$ be the weights that parameterize the distance function. The weights $\mathbf{w}$ are computed by solving the minimization problem:

$$\min_{\mathbf{w}} \sum_{(i,j)\in\{(a,b) \mid 1 \leq a < b \leq m\}} \left\{ d(I_i, I_j; \mathbf{w}) - \alpha \|\mathbf{x}_i - \mathbf{x}_j\| \right\}^2, \tag{4.1}$$

where $\alpha > 0$ is a parameter that determines the relative scaling between the distances of parameters and those of photos. Here, we slightly modified the original formulation by introducing α; the original formulation is the special case

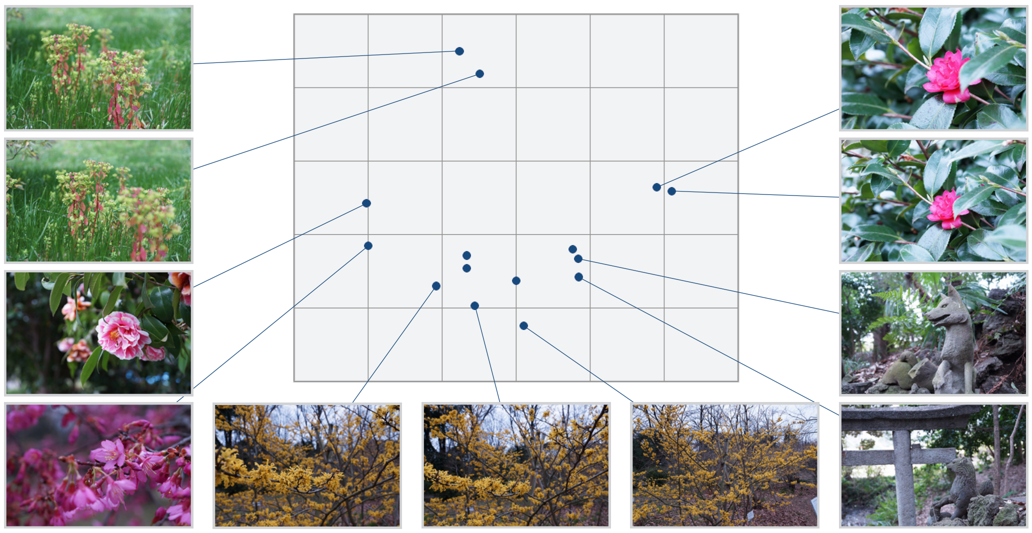


Figure 4.7: Visualization of a learned photo feature space computed by the metric learning and embedding. This space is updated based on the user-provided parameters each time the "next" button is pushed.

where $\alpha = 1.0$. We empirically found that a value of $\alpha \in [1.0, 5.0]$ works well, and we chose $\alpha = 3.0$ for all the examples. This minimization problem is solved using the L-BFGS [89], a local gradient-based optimization algorithm, provided in the nlopt library [66]. We observed that this minimization takes only negligible time, *e.g.,* typically 10–30 milliseconds for $m = 50$.

4.4.3 Photo Feature Space Computation

Having found an appropriate metric in the previous subsection, the distance between any pair of photographs can now be measured. Based on this distance metric, we define appropriate coordinates for any photos, *i.e.,* positions of photos in a Euclidean space. This operation of assigning coordinates in a particular space to elements is called *embedding*. To this end, we use *metric multidimensional scaling* (metric MDS) [46], which computes positions of target elements in a k-dimensional Euclidean space, given distances between them. In this work, we specify $k = 5$. We embed the already-enhanced photos $I_1, \ldots, I_m$ and the next photo I_{m+1} into the k-dimensional space. We refer to the resulting Euclidean space as *learned feature space*, and the position of I_i as the *learned feature vector* $\mathbf{f}_i \in \mathbb{R}^k$. Figure 4.7 shows an actual result of embedding 15 photos, where $k = 2$ is specified just for visualization purpose. It denotes that a closer pair of photos in this learned feature space is likely to be provided with more similar enhancement parameters, and *vice versa*.

Other algorithms such as Isomap [142] can also be used here. However, since this embedding is performed every time, it needs to be efficient. Thus, we chose the simple and fast metric MDS in our implementation rather than Isomap.

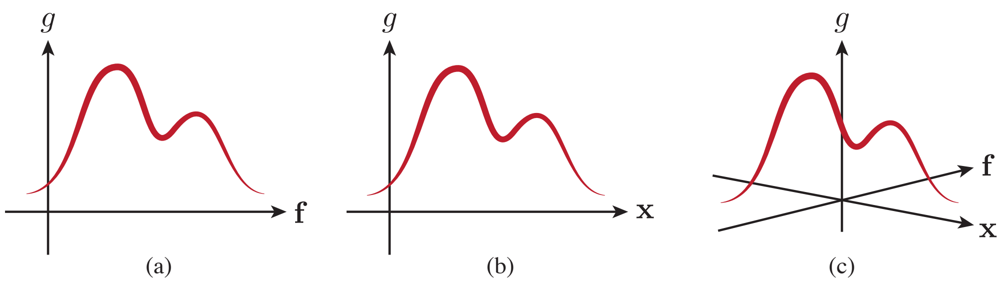


Figure 4.8: Comparison of goodness function definitions. (a) In typical preference learning approaches, the goodness is learned in feature spaces. (b) Some methods for facilitating parameter tweaking learn the goodness in parameter spaces. (c) In this work, the goodness is learned in the joint space of the features and the parameters.

4.4.4 Enhancement Preference Model

Joint-Space Formulation

As the computational model of photo enhancement preferences, we formulate the *goodness function* of color enhancement as

$$g(\mathbf{x}; \mathbf{f}) \in \mathbb{R}, \tag{4.2}$$

which returns a scalar-valued "score" (*i.e.,* the *goodness*) of the enhancement parameter $\mathbf{x}$ for the input photo whose learned feature vector is $\mathbf{f}$. For example, given a target photo whose learned feature vector is $\mathbf{f}$, the goodness function $g(\mathbf{x}; \mathbf{f})$ would return a large value if the enhancement parameter set $\mathbf{x}$ provides a good enhancement, and return a small value if it provides bad one.

To integrate these two different spaces, the parameter vector space $\mathbf{x}$ and the learned feature space $\mathbf{f}$, we introduce a higher-dimensional joint space of these two spaces:

$$\mathbf{x}' = \begin{bmatrix} \mathbf{x} \\ \mathbf{f} \end{bmatrix} \in \mathbb{R}^{n+k}, \tag{4.3}$$

and then formulate the goodness function in the joint space, *i.e.,*

$$g(\mathbf{x}; \mathbf{f}) = g(\mathbf{x}') \in \mathbb{R}. \tag{4.4}$$

Figure 4.8 shows a concept-level comparison of the goodness function definitions between previous ones and ours. Many computational aesthetics methods (*e.g.,* [124, 72, 96]) consider the relationship between the feature space and the goodness value (Figure 4.8 (a)). Most of these papers discuss the selection of features or the learning algorithms required for appropriate assessment. Some methods to facilitate design parameter adjustment (*e.g.,* [139] and the crowd-powered estimation method described in Chapter 3) analyze the parameter space and derive a distribution of "good" parameters (Figure 4.8 (b)). In contrast, this work considers *both* the feature space and the parameter space *jointly* (Figure 4.8 (c)). This is the key to facilitating parameter adjustment for new photos that have not been analyzed directly. As a concurrent work, Yumer *et al.* [166] proposed a joint-space machine learning technique; other than this, to the best of our knowledge, this specific formulation has not been previously investigated.

Goodness Function Representation

Given the jointed data $\mathcal{D}' = \{\mathbf{x}'_i\}_{i=1}^{m}$, the specific goal in this step is to derive a scalar-valued function $g(\mathbf{x}') \in \mathbb{R}$ that is smooth and has higher values at the points in $\mathcal{D}'$. To compute such a function, we use the *kernel density estimation* techniques, especially those from the work of Talton *et al.* [139]. The reader is referred to the original paper for details. Essentially, the goodness function is represented as a sum of m kernel functions:

$$g(\mathbf{x}') = \frac{1}{m} \sum_{i=1}^{m} K_i(\mathbf{x}'). \tag{4.5}$$

Here the Gaussian kernels $K_i(\mathbf{x}') = \mathcal{N}(\mathbf{x}'; \mathbf{x}'_i, \mathbf{\Sigma}_i)$ are used, where $\mathbf{x}'_i$ is the i-th data point, and $\mathbf{\Sigma}_i$ is the associated bandwidth matrix. $\mathbf{\Sigma}_i$ is computed *adaptively* by using the techniques described in [23, 122]. In contrast to Talton *et al.*'s work, we add a simple post-processing step on $\mathbf{\Sigma}_i$. In this step, the diagonal elements are forced to be more than $\epsilon > 0$. This prevents unnecessary discontinuity in the resulting function and enables smooth exploration. In SelPh, the value $\epsilon = 0.02$ was used.

4.4.5 Implementation of User Support Functions

Confidence value. The following assumption was made: if there is an already-enhanced photo that is similar to the next photo, the goodness function can provide a more accurate estimation of preference for the next photo. Based on this idea, we empirically define the confidence value c for a photo whose learned feature vector is $\mathbf{f}$ as

$$c(\mathbf{f}) = \left(\frac{1}{1 + d_{\text{closest}}} \right)^{\beta}, \tag{4.6}$$

where $d_{\text{closest}} = \min_i \|\mathbf{f} - \mathbf{f}_i\|$ is the distance between the next photo and its closest already-enhanced photo in the learned feature space, and β is a parameter to control the mapping curve. The fixed value $\beta = 3.0$ is used throughout this work. By this definition, the confidence value becomes nearly 1.0 if there is an already-enhanced photo similar to the next photo, and becomes nearly 0.0 in the opposite case. Note that better formulations for the definition of confidence might exist, but seeking it is not our goal and thus we would leave this as future work.

VisOpt Slider interface. For the visualization and optimization functions, the VisOpt Slider interface presented in Chapter 3 is used, where the input to the interface consists of a design parameter vector $\mathbf{x}$ and a scalar-valued function $f(\mathbf{x}) \in [0, 1]$. To fit our formulation to that required to use the VisOpt Slider interface, we define $f(\cdot)$ as

$$f(\mathbf{x}) = \frac{g(\mathbf{x}; \mathbf{f}) - g_{\text{min}}}{g_{\text{max}} - g_{\text{min}}}, \tag{4.7}$$

where $\mathbf{f}$ is the learned feature vector of the target photo, which is fixed during users editing. $g_{\text{min}}, g_{\text{max}}$ are the minimum and maximum values, respectively, when **f** is fixed. These values are computed by the L-BFGS [89]. In addition, we extend this VisOpt Slider to incorporate the confidence value. This is done by modifying the color scheme (Figure 4.3 (b)) and changing the strength of the interactive optimization.

Auto-enhancement. When the auto-enhancement function is called, the system computes the optimal parameter set for maximizing $g(\mathbf{x}; \mathbf{f})$ where **f** is considered to be fixed. The result is then applied to all of the sliders. The maximization problem is solved by the L-BFGS [89], typically taking less than 15 milliseconds.

4.5 User Study

We conducted a user study to investigate how photographers enhance photos repetitively with a self-reinforcing system, whether and how they are satisfied with the self-reinforcement approach and the user support functions of SelPh. The task was color enhancement of a collection of photos (*e.g.,* a photo album). In this study, the efficiency of the enhancement was not evaluated. This is because photo color enhancement is essentially a creative, open-ended exploration task. Hence, it is difficult to quantify by the task-completion time. We consider that the time taken to complete the task is not critical to the overall user experience. The quality of the enhancement result was not evaluated either. In our target use case, the resulting photos are considered to be always satisfactory for the photographer.

4.5.1 Participants

We recruited eight skilled photographers (male: 4, female: 4) via several community mailing lists and Facebook. To ensure they had the desired skills, participants were required to satisfy all of the following conditions:

- be majoring or had majored in Art or Design,
- own and be familiar with his or her own camera, not including casual cameras such as smartphones, and
- be familiar with software for photo color enhancement such as Adobe Photoshop CC [9].

As a result, three of the participants were students majoring in Art or Design ($P_1, \dots, P_3$) and the other five were professional designers ($P_4, \dots, P_8$).

4.5.2 Procedure

Preparation: Photo Taking

First of all, we asked participants to take a series of photographs for the user study. We asked participants to take 100 photographs in a single day by the same camera, with either manual or automatic exposure settings. We allowed participants to take photos in arbitrary times and places. In this study, we limited the saving file format to JPEG, rather than RAW. We instructed participants to assume that their assignment was to create a commercial photo book. The theme of the photo book was left to the discretion of the individual photographers but example themes such as "travel," "animal," "wedding party of your friends," and "walking around" were provided. Finally, we told the participants not to enhance any of the photographs prior to the commencement of the enhancement task.

Main Task: Photo Enhancement

We arranged a day to conduct the main task for each participant. The study was performed in our laboratory or classrooms. First we asked them to fill a preliminary questionnaire. This contained questions on biographic information and their opinions of the auto-enhancement in commercial software. Then, we introduced the systems for the study. We prepared two systems: SelPh and a baseline system (*Baseline*), which simulates a simple, typical software interface. Baseline was prepared by limiting the user support functions in SelPh. First, we omitted both the visualization and optimization functions from sliders. Second, as for the auto-enhancement, we removed the adaptation to individual photos from that of Baseline; it simply takes the average of all the user-specified parameters from the previous edits to perform auto-enhancement. Finally, instead of showing reference photos in the order of similarity, Baseline shows reference photos in the original order; that is, the most recently-edited photos are shown first as references.

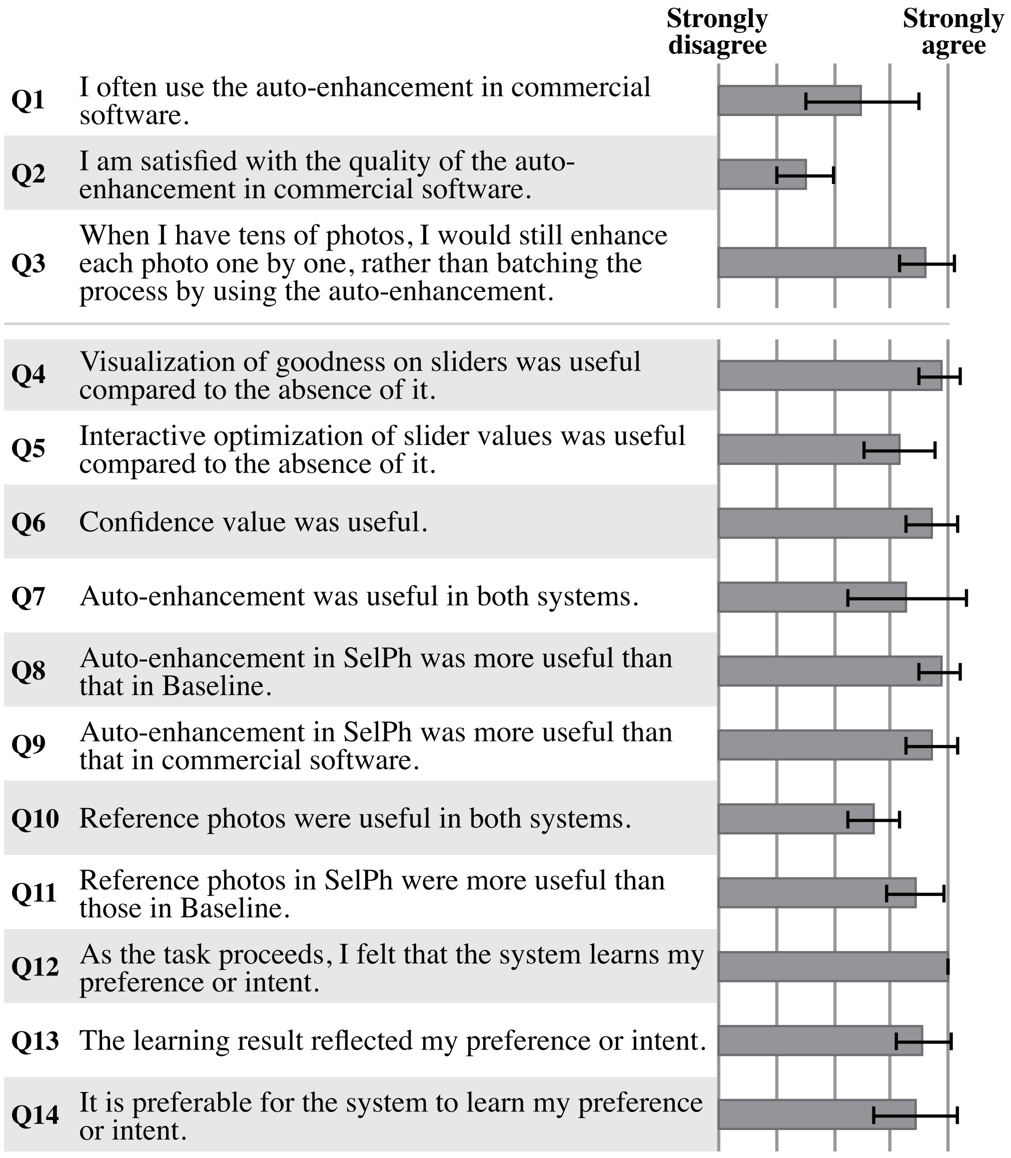
Each set of photographs was divided into two sets of 50. The order of the photos was retained so that the participant edited the photographs in the order in which they were taken. Participants were asked to enhance each set of photographs for each system. We instructed the participant how to use the system and its functions before he or she started the editing task. We then asked them to practice the system by editing 20 photos that we prepared. We alternated the order that the participants used SelPh and Baseline systems to counterbalance order effects. The participants were told to judge the speed to work at, and quality to aim for, themselves, bearing in mind that they were aiming to create a commercial photo book. The visualization and optimization functions, as well as the auto-enhancement function, were optional; the participants were shown how to toggle this functionality using check boxes and were told that they were free to do so as they wished.

After completing the tasks of enhancing 100 different photos in total, 50 using Baseline and 50 using SelPh, the participants filled in another questionnaire. This post-task questionnaire contains questions about the approach of self-reinforcement and the support functions available in SelPh. Then, we conducted informal interview. The questions in the interviews were based on individual's answers to the questionnaires, and the enhancements they made. We conducted this study in almost the same lighting conditions using the same display. The overall process took around 3 hours for each participant.

4.6 Results

4.6.1 Preliminary and Post-Task Questionnaires

Our preliminary and post questionnaires were arranged on a 5-pt Likert scale, where 5 corresponds to "strongly agree." Figure 4.9 (Q1–Q3) shows the results



Q1 I often use the auto-enhancement in commercial software.

Q2 I am satisfied with the quality of the auto-enhancement in commercial software.

Q3 When I have tens of photos, I would still enhance each photo one by one, rather than batching the process by using the auto-enhancement.

Q4 Visualization of goodness on sliders was useful compared to the absence of it.

Q5 Interactive optimization of slider values was useful compared to the absence of it.

Q6 Confidence value was useful.

Q7 Auto-enhancement was useful in both systems.

Q8 Auto-enhancement in SelPh was more useful than that in Baseline.

Q9 Auto-enhancement in SelPh was more useful than that in commercial software.

Q10 Reference photos were useful in both systems.

Q11 Reference photos in SelPh were more useful than those in Baseline.

Q12 As the task proceeds, I felt that the system learns my preference or intent.

Q13 The learning result reflected my preference or intent.

Q14 It is preferable for the system to learn my preference or intent.

Figure 4.9: Results of the preliminary (Q1–Q3) and post-task (Q4–Q14) questionnaires. We used a 5-pt Likert scale. The error bars represent the standard deviation.

of the preliminary questionnaire. These results validate the decision to work on improving repetitive manual enhancement rather than a fully-automated batch process. Figure 4.9 (Q4–Q14) shows the results of the post-task questionnaire with respect to the self-reinforcement approach and the user support functions. Overall, participants gave positive scores.

4.6.2 Feedback in Interview

Self-Reinforcement Approach

Compared to Baseline and traditional software where the enhancement of each photo is independent, P_8 said, *"(SelPh) gave me a sense of continuity"* between each enhancement. He said he often began by *"pushing the auto-enhancement* *button"* and then *"explored (the design space) from that point,"* from which he felt the sense of continuity in this repetitive task.

P_2 said, *"The functions based on the learning result, such as the visualization on sliders, evoke the feeling of collaborating with another me."* Similarly, P_8 said that, using Baseline, he felt *"lonely because I needed to do everything."* In contrast, he said that, in SelPh, *"there is interaction with the system"* through the support functions, thus *"executing the task (with SelPh) was fun."*

Overall P_3 was satisfied with our system, but she had one concern, namely that *"(my decision) was sometimes too affected by the visualization,"* while she was partially positive about this effect because it is helpful in ensuring consistency across edits. On the other hand, P_4 also mentioned a similar point but from a more positive stance, saying that *"This (visualization) is helpful (in making decisions) when I have little confidence in myself."*

P_5 mentioned that he often uses customizable filters in Aperture for preprocessing photos, but *"when the number of filters increases, it is tiresome to manage them"* because it is difficult to remember all the filters and find the most appropriate one. Compared to this, he said, *"the method of (automatically) showing the recommendation of parameters is effective for smoothly proceeding with a photo enhancement task."*

Regarding the preference learning, P_5 commented that *"I felt that my preferences and tendencies are gradually learned, which was good."* P_7 said, *"(SelPh) reflected my tendency of tweaking two parameters in pairs,"* and thus she realized that *"the system actually learned my preference."*

Visualization of Goodness on Sliders

As already mentioned above, P_4 said that the visualization helps with decision-making, especially in cases where he has little confidence in himself. This sentiment was echoed by P_3, P_5, P_7, and P_8. Also, P_7 mentioned that *"(the visualization was) useful for avoiding unnecessary exploration"*; P_5 agreed with this idea. While P_8 agreed that the visualization is helpful, he also said *"The information that I have to see during enhancement increased."*

Interactive Optimization of Slider Values

P_1, P_3, P_4, and P_6 agreed that the option to interactively optimize slider values was useful because it allows efficient exploration. P_6 liked it also because *"it is not fully-automatic"* but semi-automatic, where she *"can still tweak each slider"* as she desires. P_7 also liked it because *"Some common procedures that I have in my head were (automatically) done (by this function) so I did not have to do them by myself."* However, P_7 also said, *"When I was doing fine-tuning, I had to turn it off,"* because it conflicted with her edits. She suggested that the problem of having to turn the function on and off *"can be easily solved by introducing a shortcut key."*

Variable Confidence Values

All the participants agreed that the confidence value was useful. For example, P_7 said, *"By knowing the confidence, users can efficiently make decisions."* P_3 said that, according to the confidence value, she *"decided to use or not to use the*

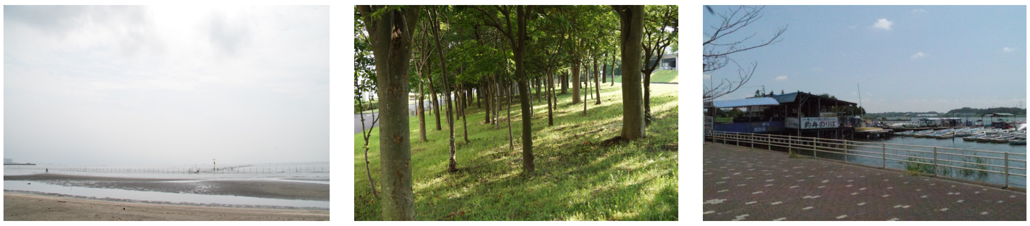


Figure 4.10: Photos that took a relatively long time to be enhanced during the study. From left to right, the photos were taken by P_1, P_3, and P_4, and the times were 56, 37, and 33 seconds, respectively.

optimization and the auto-enhancement," and that *"when the confidence value is relatively low, I did (explore the design space) by myself"* rather than using the support functions.

Some participants mentioned that the confidence value influenced other aspects of the user experience. P_5 commented, *"I could trust the system more"* by knowing the confidence of the estimation. P_7 felt *"a sense of closeness because it seems human especially when the confidence value decreased."* P_1 said, *"it was good to know the confidence because I could guess the thoughts of the system,"* and she *"could empathize with the system."* P_8 also felt *"humanity"* from the confidence value, and he said, *"It was an enjoyable experience to do the task while feeling that 'This guy (SelPh) failed to guess what I want to do!' "* when the confidence value decreased.

Auto-Enhancement

P_5 commented that the auto-enhancement in Baseline could make the design *"approach a little better, but it is still far (from the best design)."* He agreed that the auto-enhancement in SelPh was better than that in Baseline, and said that he *"used it (as a starting point) for further exploration."* P_4 commented that, compared to the auto-enhancement in SelPh, that in Photoshop *"cannot reflect my intent"* and *"only provides a safe enhancement."* P_3 said that, typically when using Photoshop, *"I rarely use the auto-enhancement,"* because *"usually it does not fit my preference."* On the other hand, she used the auto-enhancement in SelPh for about half of photographs during the study. She agreed that she *"would use the auto-enhancement if it learns from my past edits."*

Reference Photos

P_1 and other participants agreed that the reference photos helped them to edit the photographs in a consistent manner. She also commented that, when making an adjustment, references were helpful in avoiding being *"overly influenced by machine (the system recommendation)".* On the other hand, some participants (P_4, P_6, P_7, and P_8) did not agree that they made use of the reference photos during the study.

Time-Consuming Photos

Figure 4.10 shows some of photos that took relatively longer to edit during the task execution. We asked participants why these photos took longer to edit than the others. P_1 said that she wondered *“whether the sky should be completely ‘white-out’, or should retain its texture,”* so *“I had to explore a lot”* to enhance this photograph appropriately. P_3 said that he *“intended to take a special direction”* regarding the sunlight filtering through the trees. P_4 mentioned that he wanted to *“balance the blue in the sky with the red in the ground.”* To do so, he adjusted the green slider as well, to ensure balance between the blue and red sliders. These statements show that the exploration time is significantly dependent on the contents of photographs. Thus, it is not plausible to evaluate the effectiveness of SelPh by simply using the task-completion time.

Other Comments

P_3 and P_6 wanted the option to control which dataset would be used for learning and supports based on particular use cases. P_6 said that she changes the atmosphere of photos according to the clients, for example, *“(photos in) medicine-related and musical instrument-related web pages are different in saturation and color,”* and thus *“it is very helpful if I can switch (the datasets) (based on clients).”* P_4 commented that, when finishing the enhancement of 50 photos with SelPh, he *“wanted to re-visit some of the already-enhanced photos, with fully-trained user support functions.”* This idea was also mentioned by P_3, P_5, and P_6.

4.6.3 Quantitative Results

Although the goal of this study was to obtain qualitative feedback, we also recorded some quantitative variables such as the task-completion time and the amount by which the slider values were modified by mouse drags. Figure 4.11 (Top) show the mean task-completion time and the mean *slider-edit distance*. We refer the phrase ‘the slider-edit distance’ to the total amount of slider movements that were made by mouse drags, where the distance 1.0 corresponds to the width of a slider. The overall average of the mean task-completion time was 24.4 [sec] in Baseline and 17.0 [sec] in SelPh. The overall average of the mean slider-edit distance was 2.35 in Baseline and 1.05 in SelPh. The mean task-completion time of SelPh was marginally smaller than that of Baseline [$p < .10$, Wilcoxon signed-rank test], and the mean slider-edit distance of SelPh was significantly smaller than that of Baseline [$p < .05$, Wilcoxon signed-rank test]. Figure 4.11 (Bottom) shows the task-completion time and slider-edit distance sequences through 50 photos with SelPh, by P_2 and P_7. As for preference learning performance, we recorded the prediction errors, *i.e.,* the L^2-norms between the predicted optimal parameter sets and the user-provided ones, and the confidence values. The mean prediction error in each editing sequence was $\{\mathrm{min} = 0.05\ (P_3), \mathrm{max} = 0.26\ (P_8), \mathrm{mean} = 0.11\}$. The mean confidence value was $\{\mathrm{min} = 0.18\ (P_8), \mathrm{max} = 0.65\ (P_3), \mathrm{mean} = 0.46\}$. In addition to the prediction accuracy, these data also indicate strong differences between individuals; P_8 was the most difficult to predict among the participants (thus the system tended to have the lowest confidence) and, on the other hand, P_3 was the easiest (thus the highest confidence).

When participants were using SelPh, they chose to use the auto-enhancement feature for 83.0% of the photographs. We did not record how often the interactive

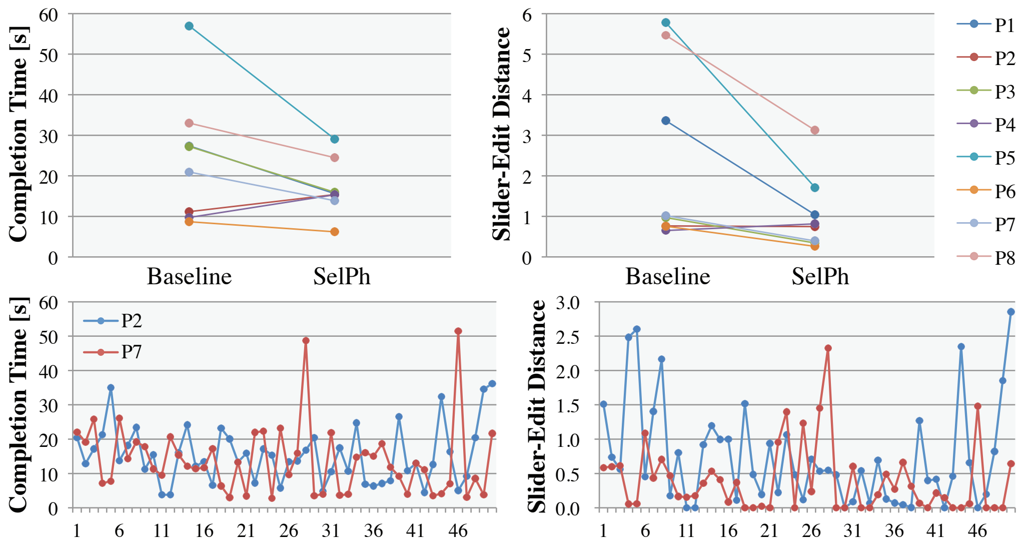


Figure 4.11: Quantitative results. (Top) The mean completion time and slider-edit distance per photo. (Bottom) Series of the completion time and slider-edit distance.

slider optimization was actually used during exploration, but the check box was in the "checked" state for 62.8% photographs when the "next" button was pushed. Similarly, the check box for the visualization function was in the "checked" state for 100% photographs when the "next" button was pushed.

4.7 Discussion

4.7.1 Summary of User Study

- **All the participants were satisfied with self-reinforcing color enhancement.** This was validated by responses to the post questionnaire items (Q12–Q14) and by the feedback comments given in the interview. We obtained positive comments compared to both fully-automatic and fully-manual approaches.

- **All the participants were satisfied with the support functions.** This was validated by questions (Q4–Q10) of the post-task questionnaire. While some participants were indifferent to the reference photo function, other functions were overall rated highly positively.

- **SelPh is useful for avoiding unnecessary exploration.** Participants explicitly mentioned this aspect. Also, compared to the baseline condition, the mean slider-edit distances significantly decreased when the participants were using SelPh.

- **SelPh is useful for making decisions efficiently.** According to the participants' comments, this is true especially when users are not confident in themselves.

- **SelPh affects decisions.** A few of the participants were aware of this effect. It is possible that the decisions of other participants were also affected, but unconsciously.

- **The inclusion of the confidence value renders SelPh more helpful.** Many participants were excited about the confidence value, which they found very useful. For example, depending on the confidence, a user can decide whether he or she first applies the auto-enhancement, or adjusts sliders from the beginning.

- **The inclusion of the confidence value makes SelPh more trustworthy and enjoyable to use.** This was the effect that we found most interesting. The confidence value gave the system a sense of humanity, and also gave participants a sense of interaction and collaboration with the system. This facilitated trust between the users and the system, and helped users to take more enjoyment.

4.7.2 Design Implications

Several lessons for further developing a self-reinforcing color enhancement system can be learned from the study. We believe that these lessons are also applicable to the design of systems that use machine learning in general.

- It is preferable to show users what the system is thinking of, rather than to make the system a "black box." In the study, the confidence value played this role; it helped users with the editing task, and also made the system more trustworthy and enjoyable to use. Showing "why the system thinks so" is a possible future direction in this aspect.

- The workflow should be semi-automatic rather than fully-manual nor fully-automatic. For example, participants liked the interactive optimization of slider values because, in addition to guiding slider values automatically, this optimization allows the user to retain the freedom to adjust each slider individually.

- Rather than guiding the user to a possible best design, enhancement systems should support the user's own exploration of the design space. From our observations and the feedback by P_4 and P_8, we found that photographers often manipulate sliders back and forth. This helps them to be sure that they have found a good design.

- In practice, photographers enhance photographs in different ways according to their intended usage. Thus, unlike the setting in our study, the system has to support a function to switch between several modes for self-reinforcement.

4.7.3 Limitations

User study method. We conducted the user study to collect initial feedback from participants, thus the method is not comprehensive and has several limitations. For example, we assumed that the same person takes photos and enhances them, but in practical usage, it is possible that different persons work on them separately. In addition, we required participants to enhance photos only in a linear order and disallowed revisiting previous photos for balancing the conditions across participants. We prepared our baseline by simply limiting the user support functions of SelPh so that participants could easily notice the qualitative properties of SelPh. However, other baseline conditions are also possible and could be useful for further investigation. Finally, the participants might have had a bias towards SelPh because of novelty factor; this needs to be taken care of for more rigorous evaluation.

Quantitative evaluation. As the focus of our study was on the qualitative properties, quantitative aspects were not fully investigated. It is difficult to evaluate the effectiveness of the self-reinforcement in terms of either the task-completion time or the amount of mouse interactions; for example, Figure 4.11 (Bottom-left) reveals highly random variation in task-completion times during the study. During the interviews, it was made clear that the difficulty of photo enhancement and hence the time it takes to complete this task, is highly dependent on the target photo's properties, including context, elements, and subjects. While we used an open-ended task, where participants could freely explore and decide on the final appearance of photos, more concrete goals for enhancement may need to be specified to achieve a more accurate quantitative evaluation. It is also important to design the study in such a way as to reduce the learning effects of participants and to ensure that the participants remain sufficiently motivated throughout the study.

Simple global photo color enhancement. The current system only has six parameters for color enhancement. This choice was based on previous work [75], as a first step for investigation of self-reinforcing systems. Future system should include more types of supported parameters (*e.g.,* Highlights/Shadows, parametric tonal curves [58, 59], and filters in Instagram) to see how this affects performance. The current system is limited to global enhancement; therefore, when a parameter is varied, it is varied for the entire image. Local enhancement is important for achieving more detailed enhancement [128, 40].

Empirically-set parameters. In our algorithms, there are several empirically-set parameters such as k, α, β, and ϵ. Although these parameters are not essential, they might affect user experience, which was not fully investigated in the study. A possibly better way is to set them using cross-validation, which is a future work.

Accuracy of machine learning. There is a lot of scope for improving the accuracy of the system's predictions of user preferences. Our current implementation of distance metric learning uses only simple low-level photo features such as color histograms, following a previous work [71]. Investigating the use of semantic features such as object recognition [35, 85], local correspondences between photos [58, 59], or generic features [96, 79] represents a promising future direction.

4.7.4 Future Work

Extension to multiple users. Our current implementation of preference learning is limited to a single user; thus a user enhances photos based on only his or her own editing history. As some of the participants mentioned, it would also be interesting to develop algorithms and interaction techniques to share the learning results among many users and utilize them in a collaborative manner [36, 71]. Relying on learned data from others may cause loss of a sense of "ownership and achievement" [22] in the resulting enhancement; we consider that the high satisfaction in our study was partially achieved by the fact that the learned data was from the user herself. We need to carefully design such a collaborative system considering this idea.

Other support functions. By using our self-reinforcement techniques, it is possible to provide more user support functions that were not available in SelPh. One interesting direction is to sort photos; it could be helpful to manipulate the order of photos, rather than providing them in the original order. For example, the system can always select the current-best-confident photo from the ones which remain to be enhanced as the next target, for avoiding to work on low-confident photos. On the other hand, the inverse order is also possible; the current-worst-confident photo could always be selected so that the system can effectively learn a user's preference (*i.e.,* active learning [126]). Another direction would be to develop a suggestive interface [97] based on the learned preference model. When a user has many photos to edit, they may want to simultaneously edit multiple photos [58], where the learned distance metric between photos could be used to propagate edits to similar photos.

Towards casual users. Although adjusting sliders is a popular interface for skilled users, casual users might prefer alternative interfaces. Investigating a combination of the self-reinforcement approach and a palette-based [38] or gallery-based interface [97, 128] would be an important future work for casual usage. Also, it is important to evaluate how casual users enhance photos with a self-reinforcing system.

Other design domains. In this chapter, we focused on the specific scenario of professional photo color enhancement to test the approach of learning personal preference from editing history. While our approach was evidenced to be effective in this scenario, it is an important future work to test our approach on other design domains. Our method relies on only a minimum amount of domain-specific formulation (only the naïve definition of feature vector of photographs), so applying our method for other domains should be technically easy. The challenge is in the user experience design, *i.e.,* how to effectively gather and utilize editing histories without sacrificing good user experience.

Chapter 5

Crowd-Powered Parameter Preference Maximization

In this chapter, we describe the crowd-powered maximization method, a novel method for maximizing the perceptual goodness of a visual design by using crowdsourced human computation. Specifically, we present a novel concept of *crowd-powered visual design optimizer*, which queries human processors employed via crowdsourcing to perform perceptual microtasks, and using the responses it proceeds the optimization to find the best parameter set. We also present the first working implementation of this concept using a novel extension of *Bayesian optimization* techniques, where the system decomposes the high-dimensional parameter tweaking task into a sequence of one-dimensional *line search* queries that are easy for human to perform by manipulating a single slider. Our *single-slider manipulation* microtask design accelerates the convergence of the optimization relative to existing comparison-based microtask designs. We applied our framework to two different design domains: photo color enhancement, and material appearance design, and thereby showed its applicability to various design domains.

5.1 Introduction

The goal in this chapter is to seek a computational automatic solution for parameter tweaking tasks in visual design; we envision that human perceptual "auto" buttons are equipped in various design interfaces for automatically setting "best" parameter sets (see Figure 5.1). The parameter set provided by this button can be used, for example, as final products (enabling "batch" tweaking) or a good starting point for further tweaking (reducing unnecessary exploration). This is inspired by the auto buttons provided in photo color correction software such as Adobe Photoshop CC [9] and GIMP [145]. They are, as far as we guess, heavily based on domain-specific techniques (*e.g.,* heuristics such as histogram stretching or machine learning-based techniques [35]). In this chapter, we investigate how to enable similar functions in general applications without relying on domain-specific knowledges.

As described in Chapter 1, we consider a parameter tweaking process as a mathematical optimization problem where the perceptual (*e.g.,* aesthetic) preference is used as the objective function to be maximized. That is, given n sliders

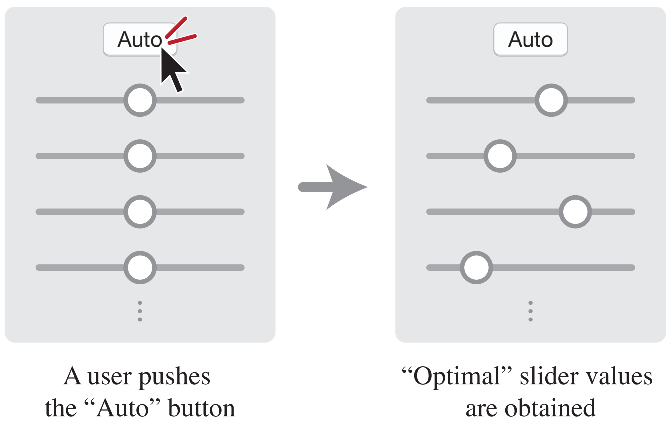

Auto
Auto
A user pushes the "Auto" button
"Optimal" slider values are obtained

Figure 5.1: Our vision of perceptual "auto" buttons for design interfaces. We investigate how to realize a human perceptual "auto" button that automatically sets slider values so that the target design is aesthetically "best".

to be tweaked, we are going to solve

$$\mathbf{x}^* = \mathop{\arg\max}_{\mathbf{x} \in \mathcal{X}} g(\mathbf{x}), \tag{5.1}$$

where $g : \mathcal{X} \to \mathbb{R}$ is the *goodness function*, $\mathcal{X} = [0, 1]^n$ is the target *design space*, and $\mathbf{x}^*$ is the optimal parameter set which maximizes the aesthetic preference of the target visual design. There are many challenges to automatically solve this problem as we discussed so far in this thesis.

5.1.1 Existing Methods

Regression methods, such as the crowd-powered estimation described in Chapter 3, estimate the shape of the function *everywhere* in $\mathcal{X}$. However, our interest lies only in the maximum of the function. Thus, formulating a method as *optimization*, instead of regression, should be more straightforward. A method for this direction has been proposed by Brochu *et al.* [33], where they constructed a Bayesian optimization framework using *pairwise comparison* oracles; the user is iteratively asked to select the better design from two candidates, and the candidates are selected strategically by Bayesian optimization techniques. Their method is called a *human-in-the-loop* optimization in that the process of optimization is converted into a series of interactive comparison tasks performed by the user.

A problem with their comparison-based method is efficiency. It requires many iterations (*i.e.,* interactive comparison tasks) to reach an optimum. This is because very limited information (*i.e.,* the relative order of two discrete samples) is obtained from a single iteration. There is an extension of the original method for reducing the number of necessary iterations by incorporating a domain-specific data as prior knowledge [31]; however such data is not always available and not suitable for on-demand use. Another limitation is that their implementation is currently limited to a single user. The method converts a complicated compound task into a sequence of simple tasks, so it has a potential to be used in a crowdsourcing setting where many workers complete simple microtasks in parallel. However, this possibility has not been investigated so far.

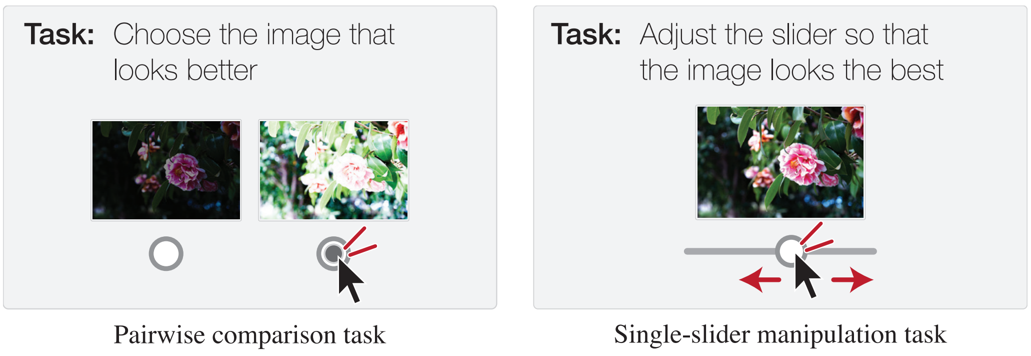

Task: Choose the image that looks better
Pairwise comparison task
Task: Adjust the slider so that the image looks the best
Single-slider manipulation task

Figure 5.2: Microtask design. (Left) Existing methods use pairwise comparison microtasks. (Right) We propose to use single-slider manipulation microtasks, which provide much richer information while the task difficulty is still comparable.

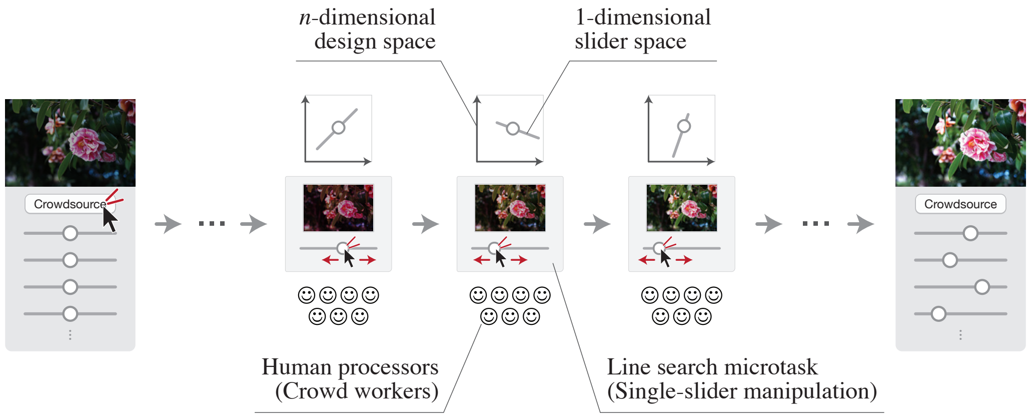
5.1.2 Contributions

To address these problems, we propose two extensions over Brochu *et al.*'s method [33]. First, we propose a Bayesian optimization framework based on *line search* oracles instead of pairwise comparison oracles; our framework decomposes the entire problem into a sequence of one-dimensional slider manipulation tasks (Figure 5.2). This makes it possible to obtain much richer information in a single iteration compared with a pairwise comparison of discrete samples and to reach the optimum much more efficiently. The difficulty of this task is slightly greater, but it is still comparable to the comparison-based task.

The second extension over the previous methods is to implement an optimization framework using *crowdsourced human computation* instead of having the user in the loop, which enables "semi-automatic" execution of the optimization (though crowds interact with the system, the user experience is automatic). We present a novel concept of a *crowd-powered visual design optimizer*, and we offer the first implementation of this concept within the Bayesian optimization framework based on line search oracles. Once the user submits a high-dimensional design task to the system, it generates a sequence of *single-slider manipulation* microtasks and deploys them to a crowdsourcing platform. Crowd workers complete the tasks independently, and the system gradually reaches the optimal solution using the crowds' responses. Figure 5.3 illustrates this concept.

We demonstrate the effectiveness of our crowd-powered optimizer using two different design domains: photo color enhancement with a 6-dimensional design space, and material appearance design based on parametric bidirectional reflectance distribution function (BRDF) models with 3- and 7-dimensional design spaces. We also show that our slider-based method makes the optimization converge faster and yields better solutions than comparison-based methods do, both in a synthetic simulation and in an actual crowdsourcing setting.

Organization. The rest of this chapter is organized as follows. First, we introduce the overview of existing Bayesian optimization techniques for completeness in Section 5.2, based on which our framework is constructed. Then, in Section 5.3, we describe a novel human-in-the-loop Bayesian optimization based on line search oracles. We adapt this framework in the crowdsourcing setting in Sec-



n-dimensional design space
1-dimensional slider space
Crowdsource
Crowdsource
Human processors (Crowd workers)
Line search microtask (Single-slider manipulation)

Figure 5.3: Concept of crowd-powered visual design optimization based on sequential line search. To enable a button for running the crowd-powered search for the slider values that provide perceptually "best" design, we present a novel extension of *Bayesian optimization*, where the system decomposes the n-dimensional optimization problem into a sequence of one-dimensional line search queries that can be solved by crowdsourced human processors.

tion 5.4, and describe our crowd-powered visual design optimizer. In Section 5.5, we demonstrate the results in two applications. In Section 5.6, our framework is evaluated by comparing with existing comparison-based approaches. Finally, we conclude this chapter with the limitations and future works in Section 5.7.

5.2 Background: Bayesian Optimization

In this section, we briefly introduce an overview of existing Bayesian optimization techniques for completeness, based on which our framework is built. Readers are encouraged to find more general and comprehensive introductions in [68, 32, 132, 127]. Here we only focus on the details that are necessary to discuss our framework, which will be described later.

5.2.1 Overview

Suppose that $\mathcal{A}$ is a d-dimensional bounded space, $f : \mathcal{A} \to \mathbb{R}$ is an unknown black-box function, and we want to find its maximum:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{A}} f(\mathbf{x}). \tag{5.2}$$

Suppose as well that the function value $f(\mathbf{x})$ can be computed for an arbitrary point $\mathbf{x}$, but $f(\cdot)$ is an *expensive-to-evaluate* function, *i.e.*, it entails a significant computational cost to evaluate the function value. Thus, while there are many optimization algorithms that can be used for solving this maximization problem (*e.g.*, the DIRECT algorithm [67]), here we are especially interested in making the number of necessary function evaluations as small as possible.

Suppose that we currently have a set of t function-value observations:

$$\mathcal{D}_t = \{(\mathbf{x}_i, f_i)\}_{i=1}^{t}, \tag{5.3}$$

where $f_i = f(\mathbf{x}_i)$. Intuitively, for each iteration in Bayesian optimization, the next evaluation point $\mathbf{x}_{t+1}$ is determined such that it is “the one most worth observing” based on the previous data $\mathcal{D}_t$. Suppose that $a : \mathcal{A} \rightarrow \mathbb{R}$ is a function that quantifies the “worthiness” of the next sampling candidate. We call this function an *acquisition function*. For each iteration, the system computes the maximization of the acquisition function to determine the most effective next sampling point:

$$\mathbf{x}_{t+1} = \mathop{\arg\max}_{\mathbf{x} \in \mathcal{A}} a(\mathbf{x}; \mathcal{D}_t). \tag{5.4}$$

The following subsections explain how to model and calculate such an acquisition function. Before introducing the detailed equations of the acquisition function, we begin with a prior assumption put on the objective function, based on which the acquisition function is calculated.

5.2.2 Gaussian Process Prior

In Bayesian optimization, the *Gaussian process* (GP) prior is often assumed on $f(\cdot)$. According to [50], a GP is described as follows:

> *“Formally, a Gaussian process generates data located throughout some domain such that any finite subset of the range follows a multivariate Gaussian distribution.”*

This is expressed as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{5.5}$$

where $m : \mathcal{A} \rightarrow \mathbb{R}$ is the *mean function* and $k : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}^+$ is the *covariance function* of the GP. When prior knowledges about $f(\cdot)$ are available, $m(\cdot)$ can be set to reflect those knowledges (*e.g.,* [31]). In this thesis, as we do not assume any domain-specific prior knowledge, we simply set

$$m(\mathbf{x}) = 0. \tag{5.6}$$

For the covariance function representation, we use the *automatic relevance determination* (ARD) *squared exponential kernel* [117]:

$$k(\mathbf{x}, \mathbf{x}') = \theta_{d+1} \exp \left\{ -\frac{1}{2} \sum_{i=1}^{d} \frac{(x_i - x_i')^2}{\theta_i^2} \right\} + \theta_{d+2} \delta(\mathbf{x}, \mathbf{x}'), \tag{5.7}$$

where $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{d+2}$ are the model hyperparameters that should be determined somehow, which will be discussed later, and $\delta(\cdot, \cdot)$ is the Kronecker-Delta function.

Since any data should follow a multivariate Gaussian distribution under the GP prior, an unobserved function value $f(\mathbf{x}_*)$ on an arbitrary parameter set $\mathbf{x}_*$ is considered to follow the distribution:

$$\begin{bmatrix} \mathbf{f} \\ f(\mathbf{x}_*) \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right), \tag{5.8}$$

where

$$\mathbf{f} = \begin{bmatrix} f_1 & \cdots & f_N \end{bmatrix}^T, \tag{5.9}$$

$$\mathbf{k} = \begin{bmatrix} k(\mathbf{x}_*, \mathbf{x}_1) & \cdots & k(\mathbf{x}_*, \mathbf{x}_N) \end{bmatrix}^T, \tag{5.10}$$

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}. \tag{5.11}$$

Using some matrix algebra, we can derive

$$f(\mathbf{x}_*) \sim \mathcal{N}\left(\mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}, k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}\right). \tag{5.12}$$

This equation provides a predictive distribution about the unobserved function value, which follows a simple Gaussian distribution. We represent $\mu(\cdot)$ and $\sigma^2(\cdot)$ are the predicted mean and the variance, respectively, *i.e.,*

$$\mu(\mathbf{x}_*) = \mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}, \tag{5.13}$$

$$\sigma^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}. \tag{5.14}$$

Note that we can use this predictive distribution as a means of scattered data interpolation, although our goal is not interpolation. This usage is referred to as *Gaussian process regression* (GPR). See the tutorial by Ebden [50] for this direction.

5.2.3 Covariance Hyperparameters

To predict $\mu(\cdot)$ and $\sigma^2(\cdot)$, the model hyperparameters $\boldsymbol{\theta}$ have to be determined. Here, we consider to determine them using maximum *a posteriori* (MAP) estimation, while other options (*e.g.,* maximum likelihood estimation) are also possible. Given the data $\mathcal{D}$, the model hyperparameters are determined by maximizing the posteriori distribution of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}). \tag{5.15}$$

By applying Bayes' theorem, we have

$$\begin{aligned} \boldsymbol{\theta}^{\mathrm{MAP}} &= \arg\max_{\boldsymbol{\theta}} \left\{ \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})} \right\} \\ &= \arg\max_{\boldsymbol{\theta}} p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}). \end{aligned} \tag{5.16}$$

From the definition of the GP prior, the conditional probability $p(\mathcal{D} \mid \boldsymbol{\theta})$ follows

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}; \mathbf{0}, \mathbf{K}). \tag{5.17}$$

The probability $p(\boldsymbol{\theta})$ is an arbitrary prior distribution of $\boldsymbol{\theta}$. In this study, we assume log-normal distributions for each hyperparameter:

$$p(\theta_i) = \begin{cases} \mathcal{LN}(\ln 0.500, 0.10) & (i = 1, \ldots, d+1) \\ \mathcal{LN}(\ln 0.005, 0.10) & (i = d+2) \end{cases}. \tag{5.18}$$

Thus, we have

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{d+2} p(\theta_i). \tag{5.19}$$

As the gradient of the objective function in Equation 5.16 can be expressed in closed form (see [117]), this maximization can be efficiently performed by using standard gradient-based optimization techniques, *e.g.,* L-BFGS [89].

5.2.4 Acquisition Function

So far, we have discussed computational tools for predicting unobserved function values. By using them, the next sampling point is chosen. Intuitively, we want to choose the next sampling point so that it is likely to have a larger value (since we want to find the maximum) and at the same time its evaluation is more informative (*e.g.,* visiting a point that is very close to already visited points should be less useful). To realize such properties, researchers have proposed several types of acquisition function for choosing the next sampling point, including

- *probability of improvement* (PI),
- *expected improvement* (EI), and
- *Gaussian process upper confidence bound* (GP-UCB).

See [127] for detailed discussions. Among them, we adopt the EI criterion [102, 68], following the previous works [33, 31].

Let f^+ be the maximum value among the currently observed data. The acquisition function based on EI is defined as

$$a^{\mathrm{EI}}(\mathbf{x}; \mathcal{D}) = \mathbb{E}_f[\max\{f(\mathbf{x}) - f^+, 0\}], \tag{5.20}$$

where $f(\cdot)$ is considered as a probabilistic variable that depends on the data $\mathcal{D}$. After some integral calculations, this can be analytically expressed in closed form as

$$a^{\mathrm{EI}}(\mathbf{x}; \mathcal{D}) = (f^+ - \mu(\mathbf{x}))\Phi(\gamma(\mathbf{x})) + \sigma(\mathbf{x})\mathcal{N}(\gamma(\mathbf{x}); 0, 1), \tag{5.21}$$

where $\gamma(\mathbf{x}) = (f^+ - \mu(\mathbf{x}))/\sigma(\mathbf{x})$, $\mu(\cdot)$ and $\sigma(\cdot)$ are the ones calculated in Equation 5.13 and Equation 5.14 using the MAP-estimated model hyperparameters $\boldsymbol{\theta}^{\mathrm{MAP}}$, and $\Phi(\cdot)$ is the cumulative distribution function of the standard normal. Since $a^{\mathrm{EI}}(\cdot)$ can have multiple local maximums, we use the DIRECT algorithm [67], which is a global optimization algorithm, to solve the maximization of this acquisition function.

5.2.5 Example Optimization Sequences

Figure 5.4 shows example sequences of applying Bayesian optimization to one-dimensional test functions. Intuitively, the next sampling point $\mathbf{x}^{\mathrm{next}}$ is selected such that both $\mu(\mathbf{x}^{\mathrm{next}})$ and $\sigma(\mathbf{x}^{\mathrm{next}})$ are large. Note that we do not intend that $\mu(\cdot)$ eventually converges to $f(\cdot)$ because this is not a regression but an optimization. For example, some regions remain uncertain (*i.e.,* having large $\sigma(\cdot)$ values) but are not sampled even after several iterations; this is because they are unlikely to contain the maximum. On the other hand, $\mathbf{x}^+$ is expected to converge to the maximum.

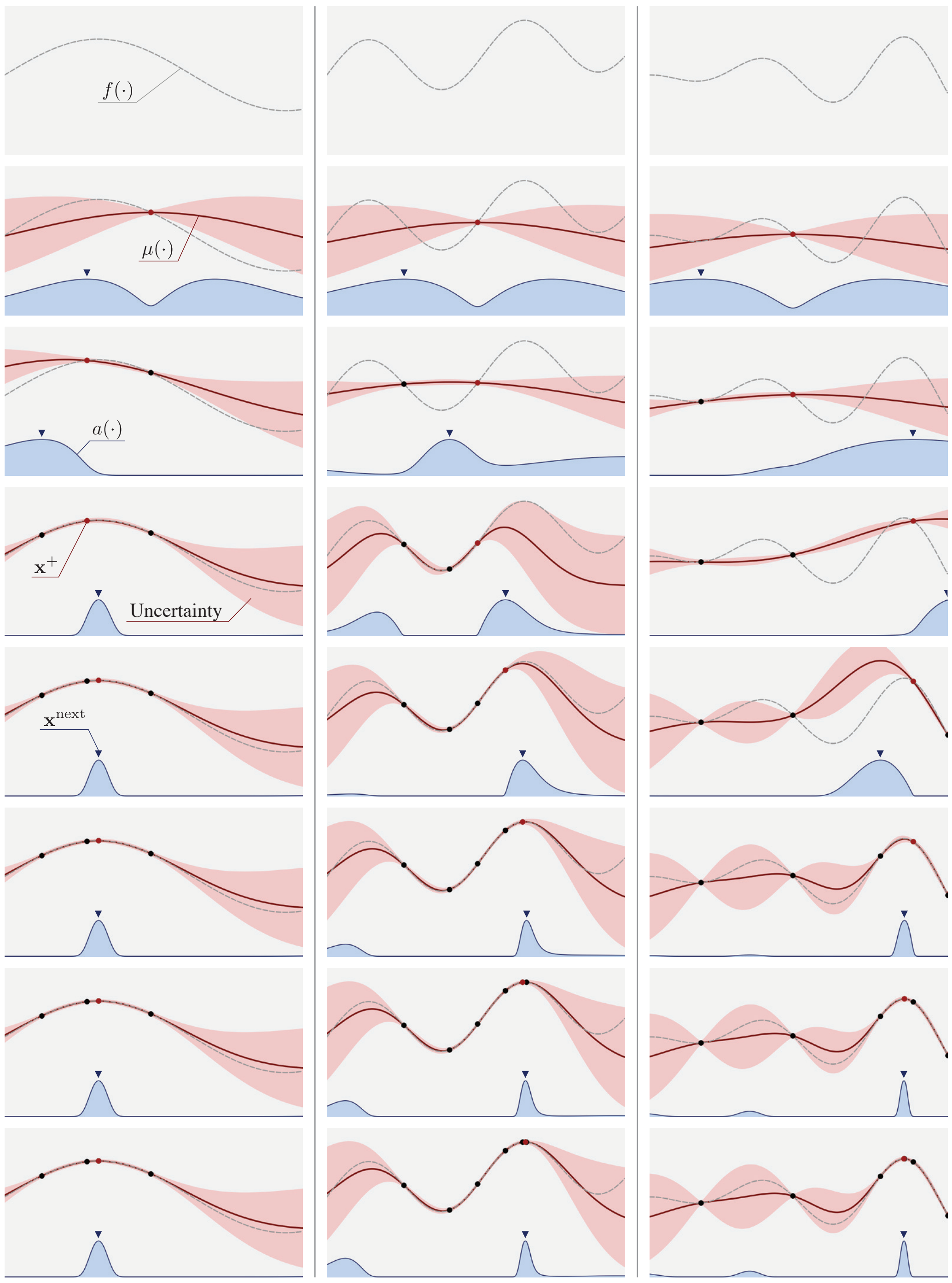


Figure 5.4: Example sequences of Bayesian optimization, applied to one-dimensional test functions. Each sequence proceeds from top to bottom. The gray dot line indicates the unknown black-box function $f(\cdot)$, the red line indicates the predicted mean function $\mu(\cdot)$, the blue line indicates the acquisition function $a(\cdot)$, the pink region indicates the 95% confidence interval (*i.e.*, $[\mu(\cdot) - 1.96\sigma(\cdot), \mu(\cdot) + 1.96\sigma(\cdot)]$), and the dots indicate the observed data (the red one is the maximum at each moment). Note that $a(\cdot)$ is scaled for visualization purpose.

5.3 Bayesian Optimization Based on Line Search Oracle

The standard Bayesian optimization in the previous section requires that function values can be observed for any argument. In other words, it is based on a *function-value* oracle. However, in our problem setting, it is not realistic to use a function-value oracle for the perceptual goodness function $g(\cdot)$. For example, suppose that you are asked to provide a real-valued goodness score for a certain visual design; this task would be rather difficult without knowing all possible design alternatives. This approach may result in unstable and inconsistent scoring even within a single person. Furthermore, if we ask the same question of many crowd workers with various backgrounds, this may be even worse.

For this reason, Brochu *et al.* [33] extended Bayesian optimization so that is could use a *function-value-comparison* oracle instead of a function-value oracle; their form of optimization iteratively queries a (human) processor about which design is better in pairwise comparison of two designs. As we noted, a problem with this solution is efficiency; it requires many iterations to reach an optimal solution. The reason is that only limited information (*i.e.,* a relative order of two sampling points) is obtained by a single query.

In this section, we describe a novel extension of Bayesian optimization based on a *line search* oracle instead of function-value or function-value-comparison oracles. The line search oracle is provided by a *single-slider manipulation* query; human processors are asked to adjust a single slider for exploring the design alternatives mapped to the slider and to return the slider value that provides the best design configuration. This oracle can be rephrased as *function-maximization-in-one-dimensional-space* oracle; mathematically speaking, given a one-dimensional subspace of the entire search space, this oracle provides a solution of a maximization problem within this subspace.

Our framework is expected to be used in human-in-the-loop settings, *e.g.,* crowdsourcing in our case. We expect that exploring a multi-dimensional space (*i.e.,* adjusting multiple sliders that correlate with each other) is difficult and intractable for (unmotivated) non-experts, while finding a best option from a one-dimensional space (*i.e.,* adjusting a single slider) is tractable and valid as a microtask design in crowdsourced human computation.

5.3.1 Slider Space

We let human processors adjust a slider, *i.e.,* find a maximum in a one-dimensional continuous space. We call this space the *slider space.* Technically, this space is not necessarily linear with respect to the target design space $\mathcal{X}$ (*i.e.,* forming a line segment in $\mathcal{X}$); however, in this study, we will consider only the linear case for simplicity and for the sake of not confusing the human processors by introducing non-linearities.

At the beginning of the optimization process, the algorithm does not have any data about the target design space $\mathcal{X}$ or the goodness function $g(\cdot)$. Thus, for the initial slider space, we simply choose two random points in $\mathcal{X}$ and connect them by a line segment.

For each iteration, we want to arrange the next slider space so that it is as "meaningful" as possible for finding $\mathbf{x}^*$. We propose to construct the slider space $\mathcal{S}$ such that one end is at the *current-best* position $\mathbf{x}^+$ and the other one is at the *best-expected-improving* position $\mathbf{x}^{\mathrm{EI}}$. Suppose that we have observed t responses so far, and we are going to query the next oracle. The slider space for the next iteration, *i.e.,* $\mathcal{S}_{t+1}$, is constructed by connecting

$$\mathbf{x}_t^{+} = \mathop{\arg\max}_{\mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^{N_t}} \mu_t(\mathbf{x}), \tag{5.22}$$

$$\mathbf{x}_t^{\mathrm{EI}} = \mathop{\arg\max}_{\mathbf{x} \in \mathcal{X}} a_t^{\mathrm{EI}}(\mathbf{x}), \tag{5.23}$$

where $\{\mathbf{x}_i\}_{i=1}^{N_t}$ is the set of observed data points, and $\mu_t(\cdot)$ and $a_t^{\mathrm{EI}}(\cdot)$ are the predicted mean function and the acquisition function calculated from the current data. The calculation of $\mu(\cdot)$ and $a^{\mathrm{EI}}(\cdot)$ is less trivial than in the case of standard Bayesian optimization; we will detail it in the following subsections.

Optionally, we can enlarge the line segment with a fixed scale, *e.g.,* 1.25. This is for avoiding cognitive biases; human processors might feel uncomfortable choosing the ends of the slider. Another reason is that the neighborhoods of $\mathbf{x}^{+}$ and $\mathbf{x}^{\mathrm{EI}}$ are each likely to be good and worth exploring. Moreover, to avoid meaningless slider tweaking, we ensure that the length is not less than 0.25.

5.3.2 Likelihood of Single-Slider Manipulation Responses

Bradley-Terry-Luce model. In crowdsourced perceptual user studies, pairwise comparison tasks, also known as two-alternative forced choice (2AFC) tasks, are frequently used. To model pairwise comparison responses from a probabilistic viewpoint, many recent studies (*e.g.,* [111, 168]) have used the Bradley-Terry (BT) model [30]. For handling cases in which more than two options are involved, the Bradley-Terry-Luce (BTL) model, which is an extension of the BT model, can be used (see [146]). Suppose that there are m design options corresponding to parameter sets $\mathcal{P} = \{\mathbf{x}_i\}_{i=1}^{m}$, and the design corresponding to $\mathbf{x}_j$ is chosen out of the m options. We describe this situation as

$$\mathbf{x}_j \succ \mathcal{P} \setminus \{\mathbf{x}_j\}. \tag{5.24}$$

The BTL model describes the likelihood of this situation as

$$p\left(\mathbf{x}_j \succ \mathcal{P} \setminus \{\mathbf{x}_j\} \mid \{g_i\}_{i=1}^{m}\right) = \frac{\exp(g_j/s)}{\sum_{i=1}^{m} \exp(g_i/s)}, \tag{5.25}$$

where g_i denotes the goodness value on $\mathbf{x}_i$, and s is a scaling factor that affects the likelihood; when s is smaller, the likelihood is more sensitive to the goodness function values, and when s is larger, the likelihood becomes closer to $1/m$ and less sensitive to the goodness function values. Here, we use a fixed value of $s = 0.01$.

Modeling slider responses. To model the likelihoods of single-slider manipulation responses, we propose to use the BTL model as follows. Let $\mathbf{x}^{\mathrm{chosen}}$ be the parameter set that a human processor chooses from the slider space $\mathcal{S}$ constructed from $\mathbf{x}^{+}$ and $\mathbf{x}^{\mathrm{EI}}$. Also let $\mathcal{S}'$ be a discretized form of $\mathcal{S}$ consisting of a finite number of points including $\mathbf{x}^{\mathrm{chosen}}$. We describe this situation as

$$\mathbf{x}^{\mathrm{chosen}} \succ \mathcal{S}' \setminus \{\mathbf{x}^{\mathrm{chosen}}\}, \tag{5.26}$$

and then apply the BTL model by considering that $\mathbf{x}^{\text{chosen}}$ is chosen out of the finite number of options. In our current implementation, we define $\mathcal{S}' = \{\mathbf{x}^{\text{chosen}}, \mathbf{x}^{+}, \mathbf{x}^{\text{EI}}\}$. We tested several definitions of $\mathcal{S}'$ that included more sampling points, but we did not observe any significant improvement in the optimization behavior; thus, we chose the minimal representation.

5.3.3 Data Representation

Suppose that we have observed t single-slider manipulation responses so far. We represent this data as

$$\mathcal{D}_t = \{\mathbf{x}_i^{\text{chosen}} \succ \{\mathbf{x}_{i-1}^{+}, \mathbf{x}_{i-1}^{\text{EI}}\}\}_{i=1}^{t}, \tag{5.27}$$

where each $\mathbf{x}_i^{\text{chosen}}$, $\mathbf{x}_{i-1}^{+}$, and $\mathbf{x}_{i-1}^{\text{EI}}$ corresponds to a certain element in a set of N_t observed data points $\{\mathbf{x}_i\}_{i=1}^{N_t}$. Note that, when a new single-slider manipulation response is added, we merge the same or very close points so that the set $\{\mathbf{x}_i\}_{i=1}^{N_t}$ does not contain any duplicate points.

5.3.4 Inference from Single-Slider Manipulation Data

Let g_i be the goodness function value at the data point $\mathbf{x}_i$ (*i.e.*, $g_i = g(\mathbf{x}_i)$) and $\mathbf{g}$ be an N-dimensional vector that concatenates the goodness values at all the observed data points:

$$\mathbf{g} = \begin{bmatrix} g_1 & \cdots & g_N \end{bmatrix}^T. \tag{5.28}$$

Unlike standard Bayesian optimization, in our case, the function values $\mathbf{g}$ are *latent*; they are not explicitly observed, but rather implicitly inferred from the single-slider manipulation responses $\mathcal{D}$. As the goodness values $\mathbf{g}$ and the model hyperparameters $\boldsymbol{\theta}$ are correlated, we infer $\mathbf{g}$ and $\boldsymbol{\theta}$ *jointly* by using MAP estimation:

$$\begin{aligned}(\mathbf{g}^{\text{MAP}}, \boldsymbol{\theta}^{\text{MAP}}) &= \mathop{\arg\max}_{(\mathbf{g},\boldsymbol{\theta})} p(\mathbf{g}, \boldsymbol{\theta} \mid \mathcal{D}) \\ &= \mathop{\arg\max}_{(\mathbf{g},\boldsymbol{\theta})} p(\mathcal{D} \mid \mathbf{g}, \boldsymbol{\theta}) p(\mathbf{g}, \boldsymbol{\theta}) \\ &= \mathop{\arg\max}_{(\mathbf{g},\boldsymbol{\theta})} p(\mathcal{D} \mid \mathbf{g}, \boldsymbol{\theta}) p(\mathbf{g} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}). \end{aligned} \tag{5.29}$$

Since $\mathcal{D}$ and $\boldsymbol{\theta}$ are conditionally independent given $\mathbf{g}$, we have

$$p(\mathcal{D} \mid \mathbf{g}, \boldsymbol{\theta}) = p(\mathcal{D} \mid \mathbf{g}). \tag{5.30}$$

The conditional probability $p(\mathcal{D} \mid \mathbf{g})$ is calculated using the BTL model:

$$p(\mathcal{D} \mid \mathbf{g}) = \prod_i p(\mathbf{x}_i^{\text{chosen}} \succ \{\mathbf{x}_{i-1}^{+}, \mathbf{x}_{i-1}^{\text{EI}}\} \mid \mathbf{g}). \tag{5.31}$$

The conditional probability $p(\mathbf{g} \mid \boldsymbol{\theta})$ is calculated from the definition of the GP prior:

$$p(\mathbf{g} \mid \boldsymbol{\theta}) = \mathcal{N}(\mathbf{g}; \mathbf{0}, \mathbf{K}), \tag{5.32}$$

Algorithm 1 Bayesian optimization based on line search oracle.

```
1: for $t = 1, 2, \ldots$ do
2:     $(\mathbf{g}_t^{\mathrm{MAP}}, \boldsymbol{\theta}_t^{\mathrm{MAP}}) =$ compute_MAP_estimation($\mathcal{D}_t$)
3:     $\mathbf{x}_t^{+} = \arg\max_{\mathbf{x} \in \{\mathbf{x}_i\}} \mu_t(\mathbf{x})$
4:     $\mathbf{x}_t^{\mathrm{EI}} = \arg\max_{\mathbf{x} \in \mathcal{X}} a_t^{\mathrm{EI}}(\mathbf{x})$
5:     $\mathcal{S}_{t+1} =$ construct_slider_space($\mathbf{x}_t^{+}, \mathbf{x}_t^{\mathrm{EI}}$)
6:     $\mathbf{x}_{t+1}^{\mathrm{chosen}} =$ query_line_search($\mathcal{S}_{t+1}$)
7:     $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{\mathbf{x}_{t+1}^{\mathrm{chosen}} \succ \{\mathbf{x}_t^{+}, \mathbf{x}_t^{\mathrm{EI}}\}\}$
8: end for
```

where $\mathbf{K}$ is the covariance matrix of this GP, which depends on $\boldsymbol{\theta}$. For $p(\boldsymbol{\theta})$, we assume a log-normal distribution for each hyperparameter as in the previous section. As the derivatives can be analytically derived, this MAP estimation can be performed by gradient-based optimization algorithms such as L-BFGS [89].

Once $\mathbf{g}^{\mathrm{MAP}}$ and $\boldsymbol{\theta}^{\mathrm{MAP}}$ have been obtained, we can compute the predictive distribution of the goodness function values for an arbitrary argument, *i.e.*, $\mu(\cdot)$ and $\sigma(\cdot)$, in the same way as in the previous section. Consequently, we can compute the acquisition function $a^{\mathrm{EI}}(\cdot)$.

5.3.5 Example Optimization Sequence

Algorithm 1 summarizes the procedure of our Bayesian optimization framework based on line search oracle. In line 6, the system queries a human. Figure 5.5 illustrates an example optimization sequence in which the framework is applied to a two-dimensional test function and the oracles are synthesized by a machine processor. The process begins with a random slider space. After several iterations, it reaches a good solution. Again, as this is not regression, the predicted mean function $\mu(\cdot)$ does not converge to the goodness function $g(\cdot)$, which is the key that enables it to find maximums efficiently.

5.4 Crowd-Powered Visual Design Optimizer

We define a crowd-powered visual design optimizer as a system that finds an optimal design which maximizes some perceptual function from a given design space and, to enable this, bases its optimization algorithm upon the use of crowd-sourced human computation. In this section, we describe the first implementation of a crowd-powered visual design optimizer based on the framework described in the previous section.

User experience. We consider a scenario in which a user pushes a “Crowd-source” button in design software for running the crowd-powered optimization process, and then he or she obtains results without any further interaction, as shown in Figure 5.3. For the user, this seems to be a fully automatic process; indeed, he or she does not need to know that many crowd workers are involved in the computation. Currently, the entire computation takes a few hours in our proof-of-concept implementation, and minimization of this latency is out of our

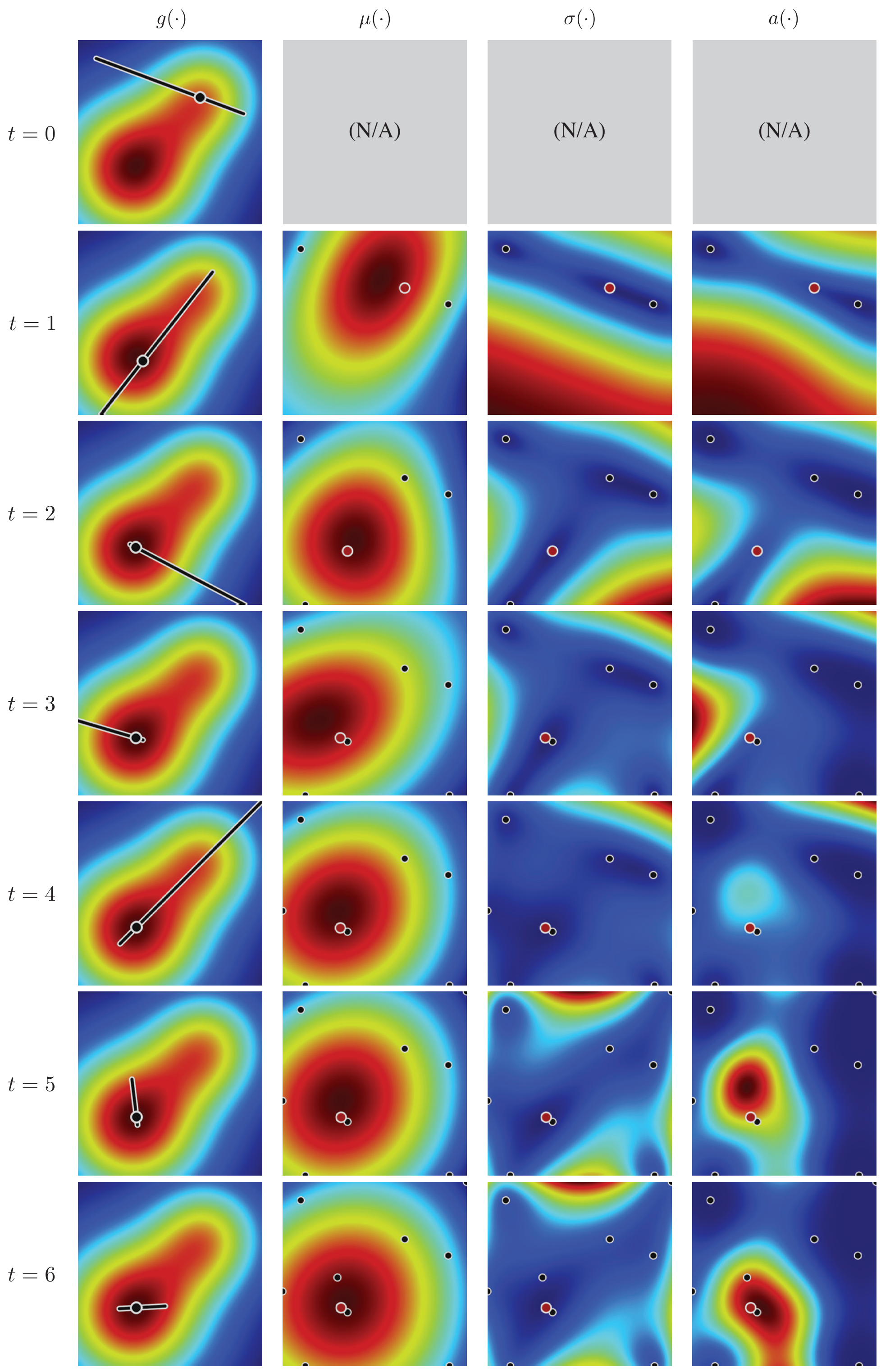


Figure 5.5: An example sequence of Bayesian optimization based on line search oracle, applied to a two-dimensional test function. The iteration proceeds from top to bottom. From left to right, each column visualizes the black-box function $g(\cdot)$ along with the slider space $\mathcal{S}$ and the chosen parameter set $\mathbf{x}^{\text{chosen}}$, the predicted mean function $\mu(\cdot)$, the predicted standard deviation $\sigma(\cdot)$, and the acquisition function $a(\cdot)$, respectively. The red dots denote the best parameter sets $\mathbf{x}^{+}$ among the observed data points at each step.

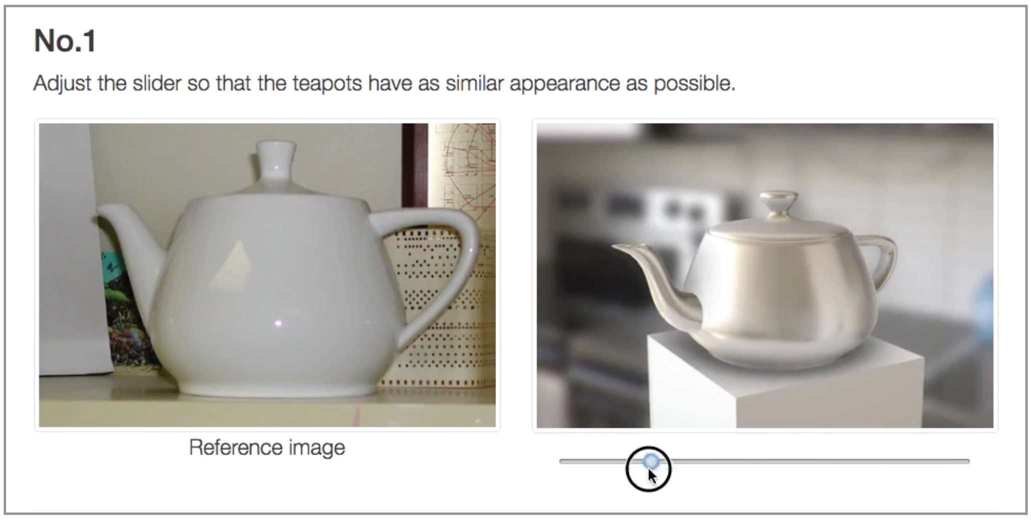

No.1
Adjust the slider so that the teapots have as similar appearance as possible.
Reference image

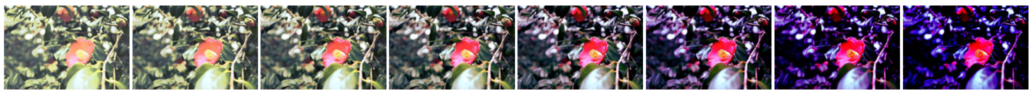
Figure 5.6: A screen capture of the web interface for crowdsourced microtasks. In this example, a reference image is shown in the left. The main image shown in the right is dynamically updated according to the manipulation of the slider.

scope. Incorporating real-time crowdsourcing techniques [24] could be used to reduce the latency.

Implementation details. We implemented a microtask platform that crowd workers access through standard web browsers. Instead of generating visual images in real time on web browsers, our platform pre-renders a finite number of images on the server by using uniformly sampled parameter sets along with the slider space. These images are loaded by the crowd workers' web browsers only once when the page is loaded; then the shown image is dynamically updated through slider manipulation in real time. Note that this strategy makes our framework applicable for domains that entail high computational costs for rendering images. To reduce cognitive bias, we set the initial slider tick positions randomly.

Task deployment. We used CrowdFlower [2] as the microtask-based crowdsourcing platform. Other platforms, including Amazon Mechanical Turk [1], can also be used. We paid 0.05 USD for each task. Figure 5.6 shows a screen capture of the web page for microtasks.

Gathering multiple responses. As we noted, we assume that there exists a common preference shared by crowd workers, *i.e.*, the goodness function $g(\cdot)$. Each crowd worker may respond with some "noise", so averaging the responses from a sufficient number of crowd workers should provide a good approximation of the underlying common preference. To take this into account, we modify the line search query in line 6 in Algorithm 1 as follows. In each iteration, the system gathers responses from m crowd workers by using the same slider space (*e.g.*, $m = 5$). After gathering the necessary number of responses, the system calculates the median of the provided slider tick positions and uses it for calculating $\mathbf{x}^{\text{chosen}}$. In the actual implementation, we deploy several additional tasks so that there



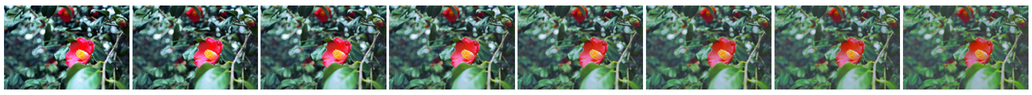


Figure 5.7: Visualization of slider spaces (for photo color enhancement) that received lower- (Top) and higher-variance (Bottom) responses from crowds.

would be no long waits for unrealistically slow workers (*e.g.,* over 30 minutes).

Quality control. Crowd workers might cheat or misunderstand their tasks and thus make poor-quality responses. It is important to detect such low quality responses and omit from data. This is called *quality control*, and various quality control methods have been investigated [64, 83]. We adopt a simple quality control approach based on redundancy; we duplicate each task and let each worker do the same task twice, but the slider ends are inverted in the second time. If a crowd worker submits contradicted values, *i.e.,* the distance between the slider tick positions is over 25% of the slider length, we consider that he or she is "lazy", so that we ignore the data.

Taking variance in responses into consideration. The variance of chosen slider positions among crowd workers depends on the configuration of slider spaces. Figure 5.7 shows examples of low- and high-variance queries. If there is a perceptually clear maximum in the slider space, crowd workers tend to make similar responses; on the other hand, if there is no clear maximum, crowd workers tend to provide high-variance responses. In the latter case, the data likelihoods should be less influential in the MAP estimation. To make our MAP estimation variance-adaptive, we modify the scale factor s in Equation 5.25 as

$$s = a \exp(b\sigma^2), \tag{5.33}$$

where a and b are fixed parameters for controlling the behavior (we currently use $a = 0.01$ and $b = 50.0$), and σ^2 is the variance of the chosen slider positions, where the width of the slider is taken to be 1.0. When the variance is zero (*i.e.,* all crowd workers provide the same responses), this becomes identical to the unmodified formulation. When the variance is higher, s becomes larger, which means the likelihood calculated by Equation 5.25 becomes less influential in the MAP estimation.

Another implementation strategy. Another possible implementation of our Bayesian optimization in crowdsourcing setting is to parallelize overall iterations for each crowd worker; *i.e.,* each worker iterates the single-slider manipulation tasks several times and finds his or her optimal solution, and then to incorporate solutions from multiple workers somehow. This approach is more parallel than our implementation and thus potentially shortens necessary timing costs. However, in this approach, it may be difficult to detect "lazy" workers because

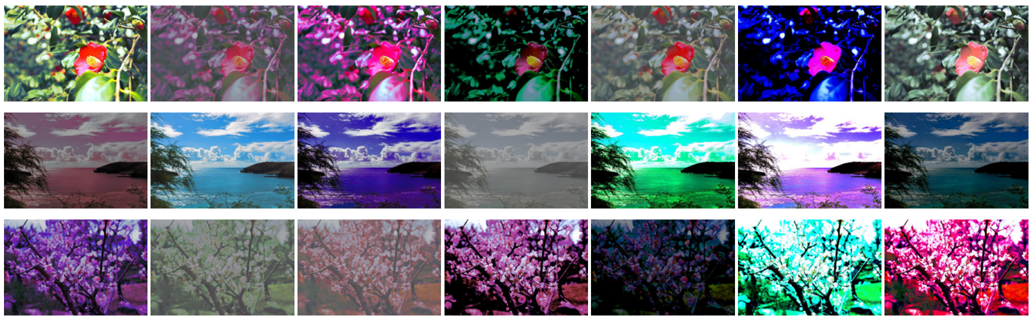


Figure 5.8: Randomly enhanced photographs, showing the design space of the photo color enhancement application.

each worker engages in completely independent tasks. Also, as the overall task becomes less "micro", which is generally undesirable because, *e.g.,* it could cause more uncertainty. Furthermore, the way of incorporating individual solutions is non-trivial. We would like to leave this approach as a future work.

5.5 Example Scenarios and Results

We tested our framework in two typical parameter tweaking scenarios: photo color enhancement and material BRDF design. In both cases, domain-specific approaches are possibly more effective; however, we emphasize that our framework does not rely on any domain knowledge (the application domains are not limited to these two) and thus it can be applied to a wide range of scenarios. In addition, ours can be combined with domain-specific approaches to build more practical domain-specific systems (we leave this for future work).

Costs. All results shown in this section were generated with 15 iterations. For each iteration, our system deployed 7 microtasks, and it proceeded to the next iteration once it had obtained at least 5 responses. We paid 0.05 USD for each microtask execution, so that the total payment to the crowds was 5.25 USD for each result. Typically, we obtained a result in a few hours (*e.g.,* the examples in Figure 5.9 took about 68 minutes on average).

5.5.1 Photo Color Enhancement

We chose the following six parameters as the target design space: brightness, contrast, saturation, and color balance with respect to red, green, and blue, as used in Chapter 3 and Chapter 4. Note that our system can also handle tonal curves by parameterizing them (*e.g.,* [58]), though we did not use them in this example. Figure 5.8 visualizes this design space. In the microtasks, we instructed the crowd workers simply to adjust a slider until the image looked the best.

We compared our crowd-powered optimization with auto enhancement functions in commercial software packages. Although such enhancement functions heavily utilize domain knowledge, they still may not be robust enough to handle certain classes of photographs, *e.g.,* ones that require semantic interpretation. We compared the results of enhancement among ours (with 15 iterations), Adobe Photoshop CC [9] (applying "Auto Tone" and then "Auto Color") and

Adobe Photoshop Lightroom CC [10] (setting both “WB” (white balance) and “Tone” to “Auto”). Figure 5.9 shows the results. To quantify how successful each enhancement is, we conducted a crowdsourced user study, where we asked crowd workers to identify which image looks best among the three enhancement results and the original one. For quality control, we duplicated each questionnaire and discarded answers from participants who provided inconsistent answers. The numbers in Figure 5.9 show the results. The photos enhanced by our crowd-powered optimization were preferred over the others in these cases.

Next, to see the robustness of our framework with respect to varying the initial randomized seeds, we repeated the same optimization procedure three times (Trial A, B, and C). Figure 5.10 (Top) shows the sequences of enhanced photographs over the iterations. We measured the differences between the trials by using two metrics: a *parameter space metric* and a *perceptual color metric*. The former is based on the l^2-norm in the space $\mathcal{X}$ between the corresponding parameter sets. The latter is based on the perceptual color distance metric called CIEDE2000 [129]; we measured the perceptual distance for each pixel in the enhanced photographs and calculated the mean over all the pixels. The reason why we need the perceptual metric is that the parameter space is non-linear with respect to visual perception; very distant parameter sets can produce very similar visuals (*e.g.,* setting equally higher color balance values for all the RGB channels does not change the visual). Figure 5.10 (Bottom) shows the results. It shows that the distances become small rapidly in the first 4 or 5 iterations, and they converge to mostly the same enhancement even though the initial conditions are quite different.

5.5.2 Material BRDF Editing

Material BRDF editing is a complex and unintuitive task even for experts in film production, so that animation studios have developed parameter tweaking systems specifically for this purpose [101]. Researchers have also presented systems for editing BRDF parameters [43, 105, 125], using many domain-specific solutions.

By recent spread of game engines, opportunities for casual game developers or designers to design a number of materials in game scenes have increased. To support especially this scenario, we consider to tweak parameters in “Standard Shader” provided in Unity 5 [150] as the target design space. This shader provides physically based shading and can be used to express various BRDFs such as plastic, metal, and fabric. In this shader, BRDF is parametrized by albedo lightness, specular lightness, and smoothness. The number of free parameters is three in monotone and seven in full color (we parametrize the color space using HSV). Figure 5.11 shows randomly sampled BRDFs, illustrating the expressiveness of this shader. When rendering images, we set an HDR skybox and reflective probes to this scene so that the material appearance would be effectively expressed.

Novice users might have goal visions (*i.e.,* make this teapot look shiny gray metal), but might not know how each of parameters affects the material appearance. Our framework enables automatic adjustment of BRDF parameters if a user has a reference photograph; our crowd-powered optimizer can be used to minimize the perceptual distance between the appearance in the photograph and the produced appearance by the shader. In the microtasks, we showed both the

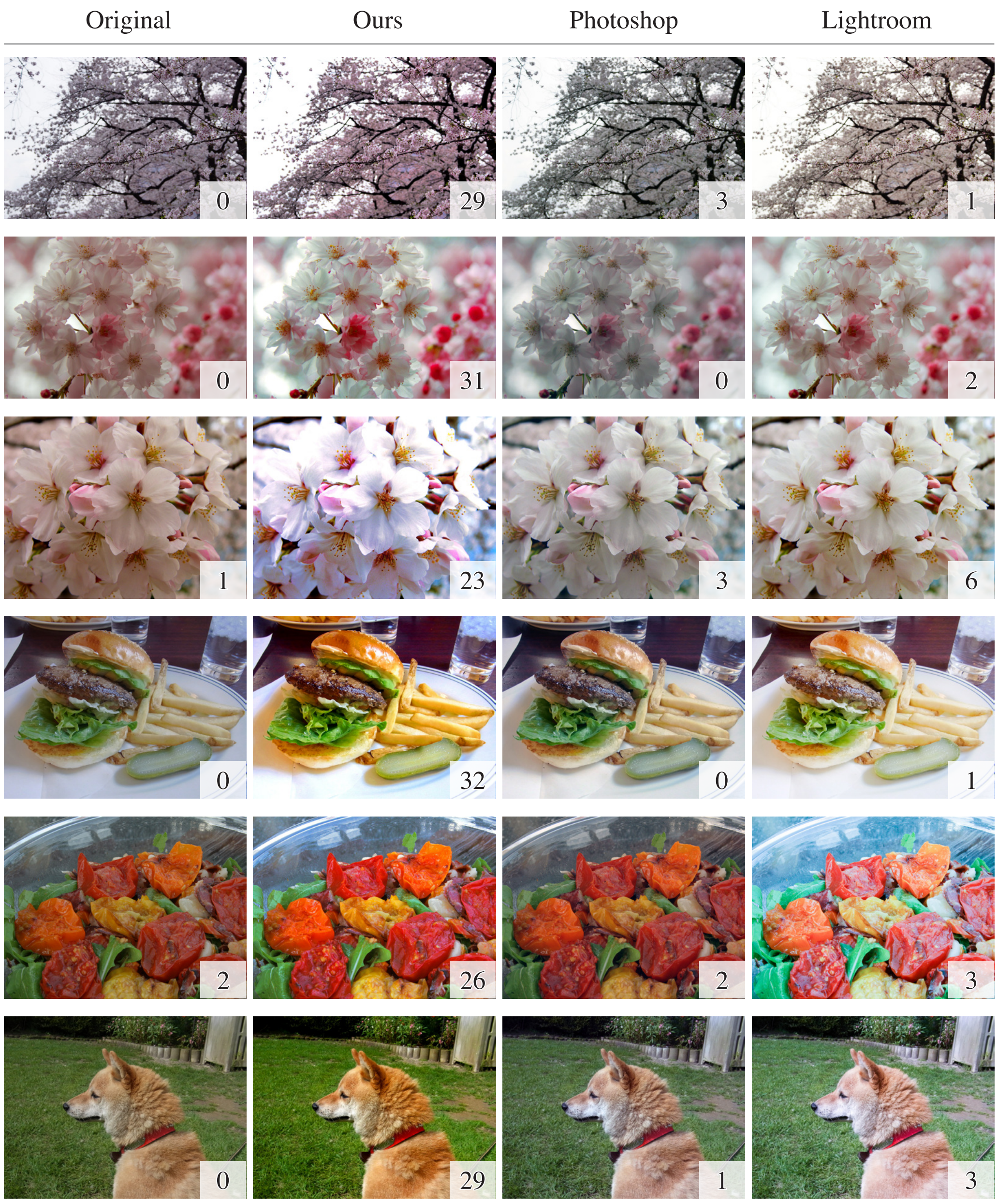


Figure 5.9: Comparison of photo color enhancement between our crowd-powered optimization and auto enhancement in commercial software packages, Adobe Photoshop CC [9] and Adobe Photoshop Lightroom CC [10]. The number on each photograph indicates the number of participants who preferred the photograph to the other three in the user study. The second and third photographs are provided by Flickr users, Kathleen Conklin and houroumono, respectively.

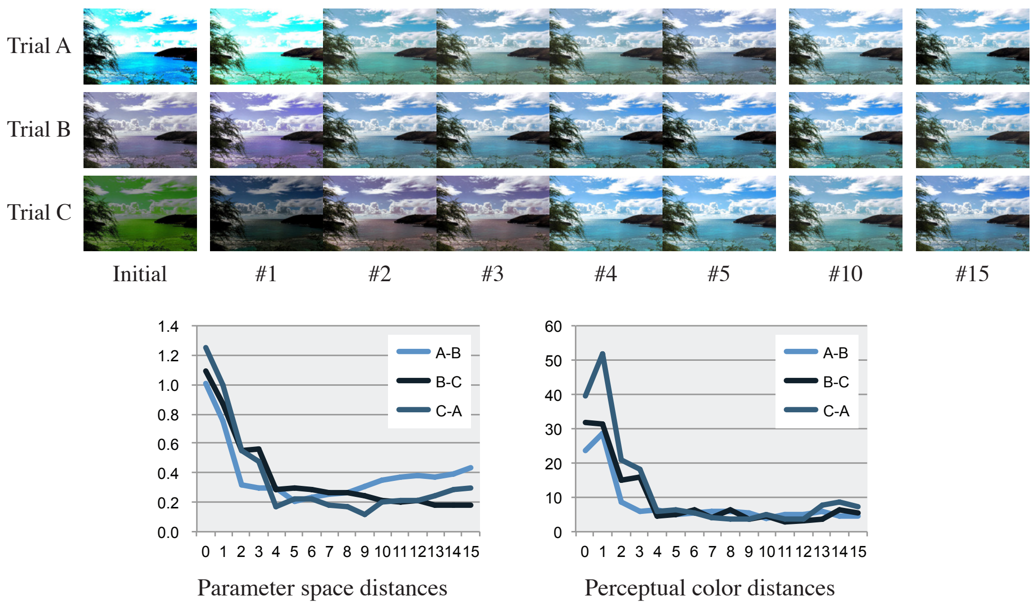


Figure 5.10: Transitions of three optimization trials over iterations in photo color enhancement. (Top) Transitions of the enhanced images. (Bottom) Transitions of the differences between each trial, measured by the parameter space metric and the perceptual color metric.

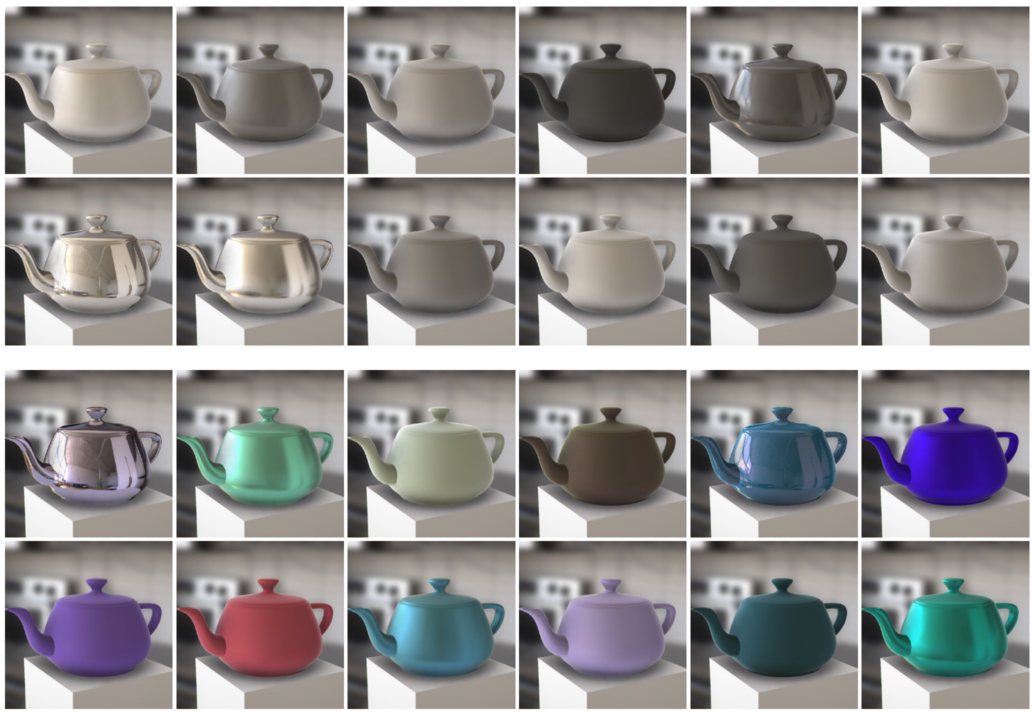


Figure 5.11: Randomly generated BRDFs, showing the design space of the material BRDF editing application with 3-dimensional (Top) and 7-dimensional (Bottom) settings.

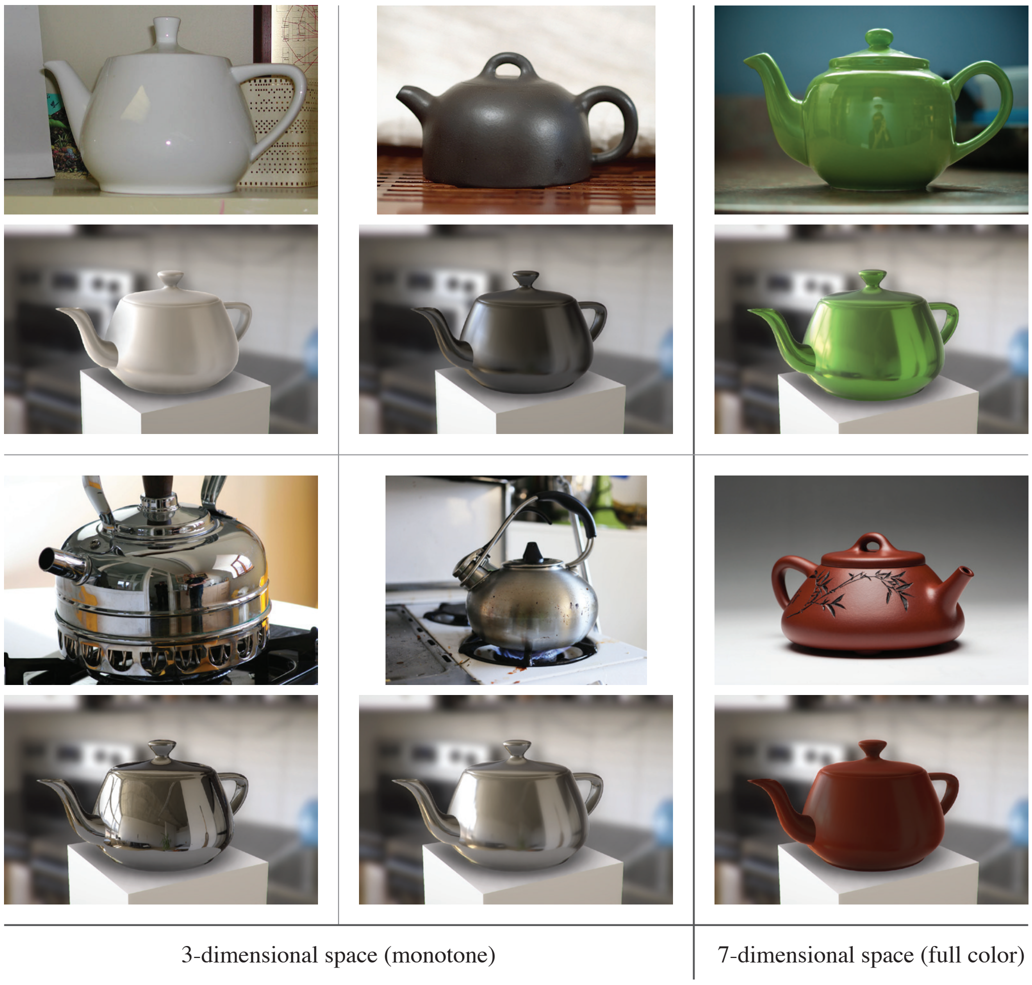

3-dimensional space (monotone)
7-dimensional space (full color)

Figure 5.12: Results of the crowdsourced BRDF editing with reference photographs. In each pair, the top image corresponds to the reference photograph and the bottom image corresponds to the computed BRDF after 15 iterations. The left four and the right two examples are computed in the 3-dimensional parameter space and the 7-dimensional parameter space, respectively. Some of photographs are provided by Flickr users, Russell Trow, Alexandr Solo, Angie Stalker, Gwen, and lastcun.

reference photograph and a rendered image with a slider side by side as shown in Figure 5.6 and asked the crowd workers to adjust the slider until their appearances were as similar as possible. Figure 5.12 shows the results for both monotone and full color spaces.

In a sense, this can be considered to be BRDF acquisition from a casual photograph. This is analogous to the concept of *crowdshaping* [135], where a human body shape is constructed from a casually taken photograph through crowdsourcing. Our framework does not rely on domain-specific knowledges and can be used on demand, while crowdshaping is based on a specific model between human shape and specific attributes, and requires comprehensive perceptual user study in advance to build the model.

Another usage of our framework is that the user can specify textual instructions instead of reference photographs. Figure 5.13 illustrates the results of this usage, where we instructed crowd workers to adjust the slider so that it looks like

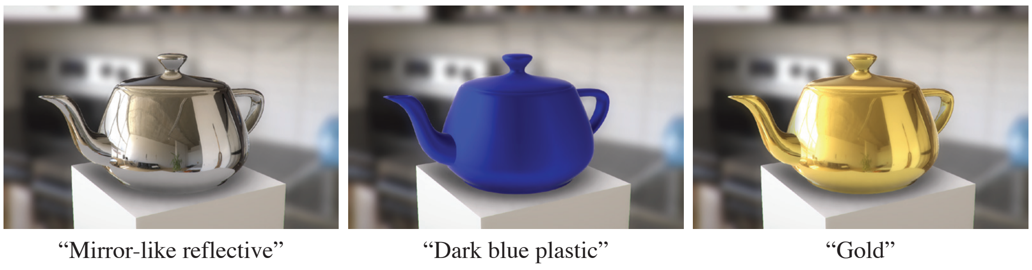


Figure 5.13: Results of the crowdsourced BRDF editing with textual instructions. For the left two and the right two images, the 3-dimensional and the 7-dimensional parameter spaces are used, respectively.

"brushed stainless", "dark blue plastic", *etc.* This is not easy when a human-in-the-loop approach is not taken.

5.6 Evaluation

In Section 5.5, we showed that our method can produce practical-quality results in two different design domains. The remaining questions to be answered here are:

Q1: *Does the use of single-slider manipulation (SSM) oracle improve optimization performance?*

Q2: *How much does the task burden increase as a result of using SSM?*

To answer them, we consider the following two baseline conditions. The first is the *2-gallery comparison* (2GC) oracle as used by Brochu *et al.* [33, 34], where a human processor is asked to choose one option from two given options. The two options are sampled at $\mathbf{x}^{+}$ and $\mathbf{x}^{\mathrm{EI}}$. The second condition is called a *4-gallery comparison* (4GC) oracle, where four (instead of two) options are presented and one option is selected from them. The four options are sampled using Schonlau *et al.*'s method [123], following [31]. We modeled the data likelihood in 2GC and 4GC by using the BT and BTL models, respectively.

First, we compared our SSM approach with 2GC and 4GC using *synthetic* settings (**Exp1**), where we simulated responses from crowds by using a known test function. Then we compared the three approaches in a *crowdsourcing* setting (**Exp2**). In both **Exp1** and **Exp2**, we evaluated the number of iterations required to get good solutions, to answer **Q1**. In **Exp2**, we also evaluated the microtask burden on the crowds, to answer **Q2**.

5.6.1 Experiment 1: Synthetic Setting

We optimized an n-dimensional test function:

$$g(\mathbf{x}) = \exp\left\{-\frac{(\mathbf{x}-\boldsymbol{\mu})^2}{2\sigma^2}\right\}, \tag{5.34}$$

where we set $\boldsymbol{\mu} = [\,0.5 \;\; \cdots \;\; 0.5\,]^T$ and $\sigma = 0.5$. This function has its maximum at $\boldsymbol{\mu}$. We synthesized oracles from this function and tested $n \in \{2, 3, 4, 6, 12, 20\}$.

For each iteration, we recorded the residuals:

$$r = \|\mathbf{x}^{+} - \boldsymbol{\mu}\|. \tag{5.35}$$

Figure 5.14 shows the results. In general, the residuals drop faster at the beginning and converge to smaller values at the end in SSM than in 2GC and 4GC. The gallery comparison approaches often provide "flat" graphs where the residual does not decrease for several iterations. The reason for this may be that, especially at the beginning of the iteration, the algorithm is likely to sample the "boundary" of $\mathcal{X}$ because the uncertainty is very large around the boundary. On the other hand, this "boundary-exploration" stage is not a critical problem in the SSM approach, because the slider space lies across the design space even when the one end is on the boundary.

5.6.2 Experiment 2: Crowdsourcing Setting

To quantify the optimization performance in crowdsourcing settings, we used photo color enhancement with a *reference image*. We manually chose a reference parameter set $\mathbf{x}^{\text{ref}}$ and generated a corresponding image as the ground truth (*i.e.,* a reference image) in advance. In the SSM setting, crowd workers were shown a reference image and an editable image with a slider, and asked to adjust the slider so that the edited image would be as similar to the reference image as possible (Figure 5.15 (Left)). In the 2GC and 4GC settings, the crowd workers were shown a reference image and options and asked to find the most similar option (Figure 5.15 (Middle) and (Right)). As the quality control in the 2GC and 4GC settings, we duplicated each comparison task while showing the options in opposite order and omitted workers whose responses were contradictory. Figure 5.16 shows the change in the error as measured by the perceptual color metric over the iterations. We can see that the SSM approach performs better than the 2GC and 4GC approaches do. The trends are mostly consistent with the results of the synthetic settings (Figure 5.14). Figure 5.17 visualizes sequences of the images enhanced by the predicted best parameter set $\mathbf{x}^{+}$ at each step.

Microtask burden. A possible drawback of the single-slider manipulation task is that it can be more tedious than a comparison task. To respond to this concern, we compared SSM, 2GC, and 4GC in terms of the task-completion time. For each microtask executed by a crowd worker, we measured the elapsed time from the moment that the HTML documents in the task page were loaded to the moment that the submission button was pushed. The task-completion time included the time for reading the task instructions and the time for conducting duplicate tasks for quality control. Figure 5.18 shows the results using box plots (the maximum values are not shown for space reasons). It indicates that the SSM microtask requires more time than the other microtasks, but its time is still comparable (less than twice). Considering that the convergence of the SSM approach is much (at least more than two times) faster than those of the others, we argue that our SSM approach is preferable even though the task burden is moderately heavier. Note that this increase of task-completion time does not badly affect the entire latency in practice because other overhead (*e.g.,* between requests of task deployment and findings of the tasks by crowds) is dominant.

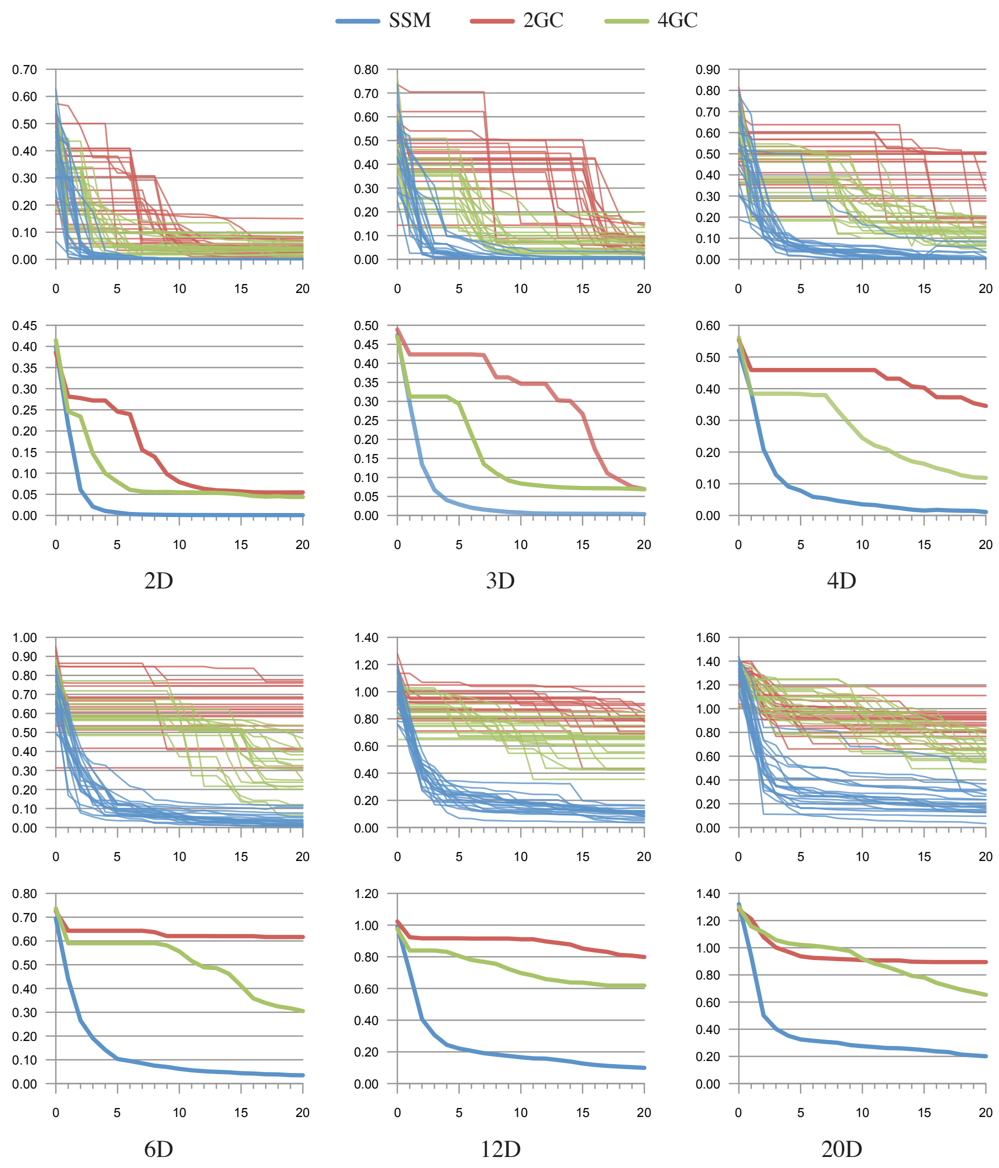
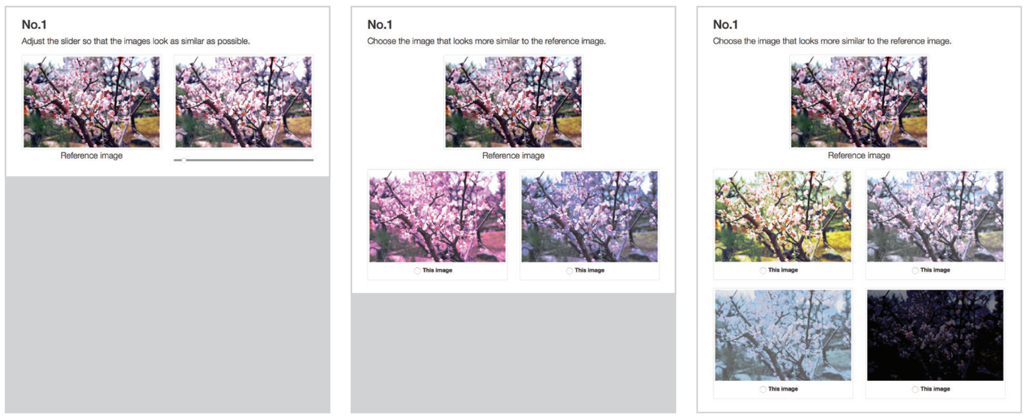


Figure 5.14: Results of the synthetic experiment. We compare the residuals (vertical axis; lower is better) over iterations (horizontal axis) among the single-slider manipulation (SSM), the 2-gallery comparison (2GC), and the 4-gallery comparison (4GC) settings. We repeated the same procedure 20 times for each condition. In each pair of graphs, the top graph shows the raw data, and the bottom graph shows the means for each condition.

5.7 Limitations and Future Work

As we described in Chapter 1, our framework is built upon many assumptions, some of which are difficult to validate quantitatively. For example, we assumed that there exists a common goodness function shared among crowd workers. In Figure 5.10, we observed that even if the initial parameter sets were different, they eventually converged to similar designs, indicating that this assumption seems valid in this specific situation. However, it is difficult to determine whether this assumption is valid in other situations. For example, there are various color


No.1
Adjust the slider so that the images look as similar as possible.
Reference image

No.1
Choose the image that looks more similar to the reference image.
Reference image
This image
This image

No.1
Choose the image that looks more similar to the reference image.
Reference image
This image
This image
This image
This image

Single-slider manipulation (SSM) 2-gallery comparison (2GC) 4-gallery comparison (4GC)

Figure 5.15: Screen captures of microtasks used for performance comparison in the crowdsourcing setting. Single-slider manipulation (SSM), 2-gallery comparison (2GC), and 4-gallery comparison (4GC) approaches are shown from left to right. The application of photo color enhancement with reference images is used.

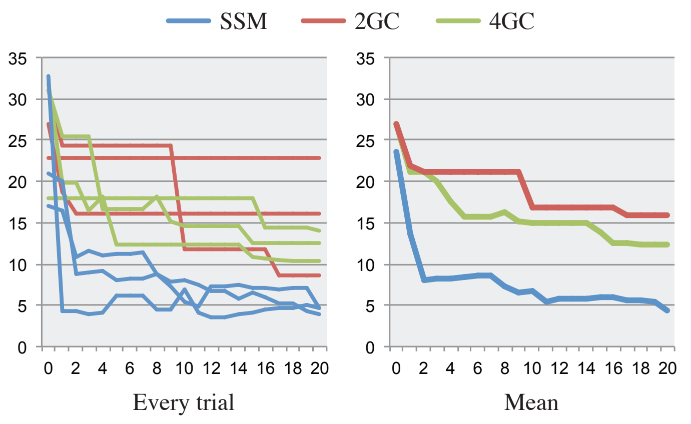


Figure 5.16: Results of the crowdsourcing experiment. We compare the residuals (vertical axis; lower is better) over iterations (horizontal axis) among the single-slider manipulation (SSM), 2-gallery comparison (2GC), and 4-gallery comparison (4GC) settings. The application of photo color enhancement with reference images is used. We repeated the same procedure 3 times for each condition.

palette styles; some people might prefer a certain style while others might prefer another style. In this case, the assumption of a common goodness function may be invalid.

For better understanding of the behavior of our framework, it may be interesting to evaluate the consistency between the goodness function $g(\cdot)$ constructed by regression methods (*e.g.,* the crowd-powered estimation method described in Chapter 3) and the optimal parameter set $\mathbf{x}^*$ obtained by our framework. It is expected that, for the same design space, they eventually provide peaks at the same location. However, especially when the design space is high-dimensional, our framework is expected to find the maximum more rapidly thanks to both the single-slider manipulation microtask design and the Bayesian optimization-based sampling strategy, while the crowd-powered estimation method described in Chapter 3 tries to sample points randomly to cover the entire design space.

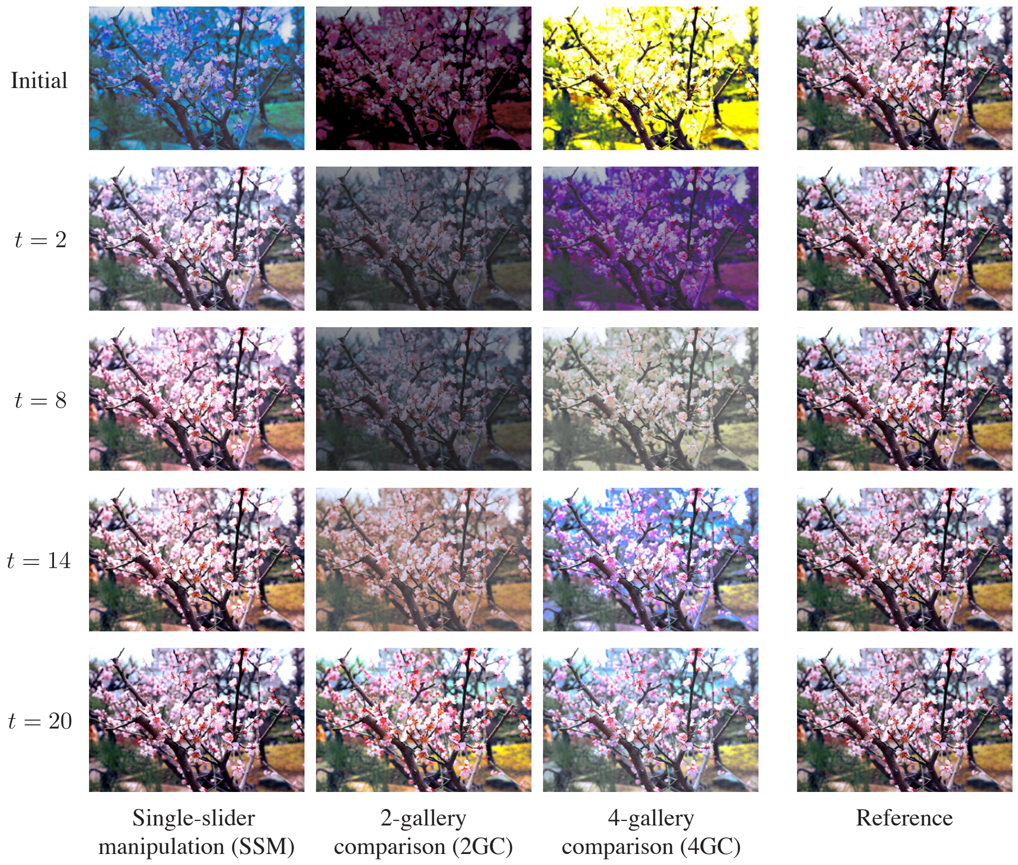

Initial
$t = 2$
$t = 8$
$t = 14$
$t = 20$
Single-slider manipulation (SSM)
2-gallery comparison (2GC)
4-gallery comparison (4GC)
Reference

Figure 5.17: Sequences of the current-best images over iterations in the performance comparison in the crowdsourcing setting.

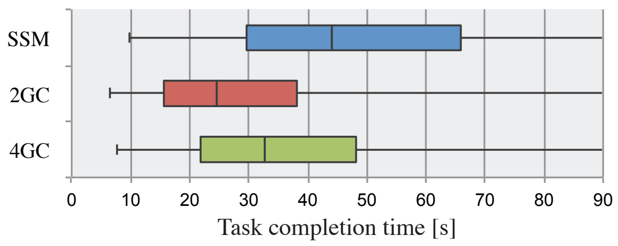


Figure 5.18: Comparison of task-completion times among single-slider manipulation (SSM), 2-gallery comparison (2GC), and 4-gallery comparison (4GC) microtasks. The times are measured using the photo color enhancement application.

We have discussed how to reduce the number of queries, but we have not touched on how the time and monetary costs of crowdsourcing can be minimized. To reduce the time cost (*i.e.*, the latency to obtain the optimization results), our framework could incorporate real-time crowdsourcing techniques [27, 24]. Also, parallelizing (or batching) Bayesian optimization [17, 44] may be useful for reducing the entire latency. Reducing the monetary cost may be more challenging. Currently, we always employ a fixed number of crowd workers in each iteration; this number could be adaptively adjusted to each application and each step, but we have not investigated strategies for this. Also, we need to develop a criterion for detecting convergence and thereby stopping the iteration automatically; this will prevent unnecessary tasks from being deployed.

We assumed that design spaces are adequately parametrized in advance. We do not intend to handle very high-dimensional spaces (*e.g.,* over 100 parameters), which need to be reduced beforehand by other methods. Also, design spaces need to be visually smooth, because our formulation assumes that the goodness function is smooth.

Our framework does not rely on domain-specific knowledges; this enables it to be used in various design domains. However, to build optimizers for specific design scenarios, it would be more effective to use domain-specific rules or data. One possibility for this direction is to incorporate such prior knowledges as the mean function $m(\cdot)$ in Gaussian process (Equation 5.6). For example, Brochu *et al.* [31] took this approach to incorporate other animators' editing results as prior knowledges to support an animator to design fluid animations.

Investigating the use of the slider-based optimization in a single-user setting is also an interesting idea. In this case, it is advantageous that the parameters can be optimized based on the user's personal preference. Unlike tweaking raw sliders simultaneously, users do not have to learn and remember the effects of raw parameters; what users have to care about is the maximum of the slider in each step.

We believe that the single-slider manipulation microtask design could be effective for regression purpose (*e.g.,* the method described in Chapter 3) as well, while in this work we focused on its use for optimization. We also expect that our microtask design can be useful for generating a new type of data annotation for machine learning (*e.g.,* [124, 39, 111, 54, 135, 125]), where comparison-based or Likert-scale-based methods are currently used.

Chapter 6

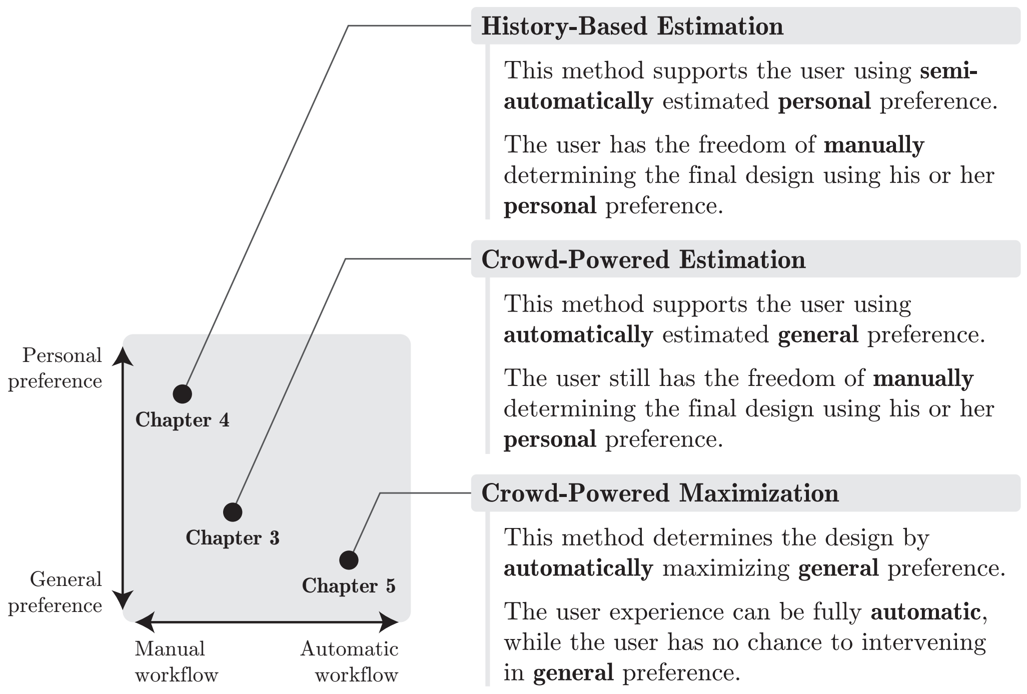
Conclusion

6.1 Summary

Our goal was to explore the potential of computational methods for facilitating parameter adjustment in design activities in which aesthetic preference is used for assessing the quality of design and is to be maximized. One of the challenges to achieve this facilitation is that, as the objective of parameter adjustment is based on human perception, it is non-trivial how we can handle it in computational ways. Our key idea is that a parameter adjustment task can be supposed as a mathematical optimization problem, and from this viewpoint, we can think of usages of computational techniques in a structured manner.

In this thesis, we specifically explored three computational design methods: the crowd-powered estimation method (Chapter 3), the history-based estimation method (Chapter 4), and the crowd-powered maximization (Chapter 5) method. Two of the three methods are designed to estimate preference by computational techniques and then facilitate users' manual design exploration using the estimated preference. The other one is designed to automatically search the design space for the optimal solution by computational decision making. Investigation of these two usages of computational techniques is one of our primary contributions. For computationally handling aesthetic preference, we sought to gather necessary preference data from crowdsourced human computation and from users' editing history. We showed both of the two data gathering approaches were able to be effective, which is another primary contribution in this thesis.

Figure 6.1 summarizes the relationship among these three methods from the users' viewpoint. We illustrate their characteristics from two aspects: *whose preference is involved*, and *how the user interacts*. In the aspect of the preference owner, the history-based estimation method is the most *personal*; both the estimation computation and the determination of the final design are based on the user's personal preference. On the other hand, the crowd-powered maximization method is the most *general*; the design process is completed using only crowds' general preference. The crowd-powered estimation method is in-between; it relies on crowds' general preference in its computation but the final design is determined based on the user's personal preference. These three methods also has different characteristics in the aspect of the user interaction; the history-based estimation method is the most *manual*, the crowd-powered maximization method is the most *automatic*, and the crowd-powered estimation method is in-between. We consider that these three methods provide solutions for the same problem from



History-Based Estimation

This method supports the user using **semi-automatically** estimated **personal** preference.

The user has the freedom of **manually** determining the final design using his or her **personal** preference.

Crowd-Powered Estimation

This method supports the user using **automatically** estimated **general** preference.

The user still has the freedom of **manually** determining the final design using his or her **personal** preference.

Crowd-Powered Maximization

This method determines the design by **automatically** maximizing **general** preference.

The user experience can be fully **automatic**, while the user has no chance to intervening in **general** preference.

Figure 6.1: Summary of the three methods from the users' viewpoint. We illustrate their relationship using two dimensions: their preference (personal vs. general) and workflow (manual vs. automatic) properties.

qualitatively different approaches, and thus we do not intend to argue that, for example, some method is simply superior (or inferior) to the others.

6.2 Discussions

6.2.1 Estimation of $g(\cdot)$ vs. Maximization of $g(\cdot)$

We have considered two usages about computational techniques. The first approach, *i.e.,* the estimation of the goodness function $g(\cdot)$, which we investigated in Chapter 3 and Chapter 4, has several advantages compared to the other approach:

- The user can maintain the control about how he or she explores the design space $\mathcal{X}$. He or she is not forced to follow the computational guidance by the system. This was appreciated by experts in the user study in Chapter 4.

- The chosen solution $\mathbf{x}^*$ is always ensured to be optimal for the target user, as the "true" goodness function used for deciding the final solution is owned by the user, which can be different from the "estimated" goodness function used for guidance.

- Even if the estimation is not very accurate, it can still guide the user to explore the design space $\mathcal{X}$ effectively. We observed that, in most cases, the estimated goodness function is useful for providing a good starting point for exploration and for eliminating meaningless visits of bad designs.

- When the algorithm is less confident about the estimation, the system can behave still effective by making the computational guidance less influential, rather than guiding the user in an inaccurate way. This was evidenced to be effective and preferable in Chapter 4.
- This approach can be seamlessly integrated in existing practical scenarios as evidenced in Chapter 4, because it does not intend to replace existing workflows but does augment (or enhance) existing worklows.

On the other hand, the second approach, *i.e.,* the maximization of the goodness function $g(\cdot)$ by computational techniques, investigated in Chapter 5, has different advantages:

- The user does not need to care about the strategy of how design exploration should proceed. This enables a new paradigm for designs driven by aesthetic preference, and it solves many constraints with respect to user experience. For example, users are released from the need to understand and learn the effects of each design parameter in this approach.
- By implementing this approach using crowdsourced human computation, the user no longer needs to interact with the system, enabling fully automatic workflows. This further broadens possible usage scenarios.
- The found solutions by this approach can be used as either final products or good starting points for further manual refinement. In Chapter 5, we observed that most results were not necessarily perfect but quite acceptable as final products. Note that future investigation may further improve the quality.
- This approach aims to find the optimal solution as efficiently as possible, based on computational optimization techniques. In Chapter 5, we sought to use Bayesian optimization techniques so as to reduce the number of iterations. Compared to the other approach of estimating the goodness function $g(\cdot)$ everywhere in the design space $\mathcal{X}$, whose computational cost is essentially exponential with respect to the dimensionality, this approach may ease this problem in high-dimensional design spaces.

A hybrid approach between these two approaches is also possible. For example, partially constructing the goodness function $g(\cdot)$ around the expected optimal solution may be useful for supporting users to explore the design space. Investigating this possibility is an important future work.

6.2.2 Crowdsourced Human Computation vs. Editing History

We have investigated two data sources: crowdsourced human computation in Chapter 3 and Chapter 5, and editing history in Chapter 4. Here we summarize the advantages and the disadvantages of these two data sources learned through our investigation:

Application domain. In the human computation approach, the use of micro-task-based crowdsourcing is inevitable, and it limits its application to the domains where even unskilled, non-expert crowd workers can adequately assess the quality of designs. An example of such domains is photo color enhancement; it may be a valid assumption that most crowd workers have their preference on photo color enhancement, since enhanced photographs are ubiquitously seen in daily lives (*e.g.,* in product advertisements). On the other hand, by using editing history of expert users, we can possibly apply computational techniques to the domains where only experts can adequately assess the quality of designs.

Whose preference? When using crowdsourcing, we assume the existence of "general" preference shared among crowd workers, and query microtasks to observe the general preference. On the other hand, an advantage of using editing history as the data source is that we can learn "personal" preference of the user. In the user study with eight expert users in Chapter 4, although we have not quantitatively measured, we observed clear differences in preference between each participant. Some participants appreciated that the system is useful and trustful because it learns personal preference from personal data. This suggests the importance of learning personal preference.

Difficulty in data gathering. Microtask-based crowdsourcing enables on-demand generation of new data as we demonstrated in Chapter 3 and Chapter 5, which is a large advantage of the human computation approach. On the other hand, editing history cannot be generated on demand, and has to be gathered implicitly. To overcome this difficulty of data gathering, in Chapter 4, we focused on a very specific scenario where a user has many photographs that are going to be enhanced manually. While this is a realistic scenario in photo color enhancement, it may not always be easy to find similar situations in other design domains.

6.3 Remaining Challenges

We have investigated computational design methods under many assumptions. To ease assumptions and make our methods more practical, there are a number of possible remaining challenges as next steps.

6.3.1 Design Space Parametrization

We have assumed that the target design space is appropriately parametrized in advance. It is an important future work to seek appropriate parametrization techniques, to broaden (currently limited) applications of computational design methods driven by aesthetic preference.

Currently, we do not try to directly handle very high-dimensional design spaces, *e.g.,* with thousands of degrees of freedom. For example, "naïve" 3D shape deformation is out of our scope; when a designer tweaks a 3D shape represented by a triangular mesh by deforming it, the degrees of freedom of this deformation are considered $3 \times |\mathcal{V}|$, where $\mathcal{V}$ is a set of all vertices. Directly manipulating each vertex position is the most naïve approach, but unsurprisingly, this easily breaks the aesthetic quality of the shape and results in unacceptable designs, as shown in Figure 6.2 (Left). This means that, although the design space is wide, acceptable designs are lying on only quite limited subspaces, or a lower-dimensional manifold. This fact makes it challenging to directly solve this design

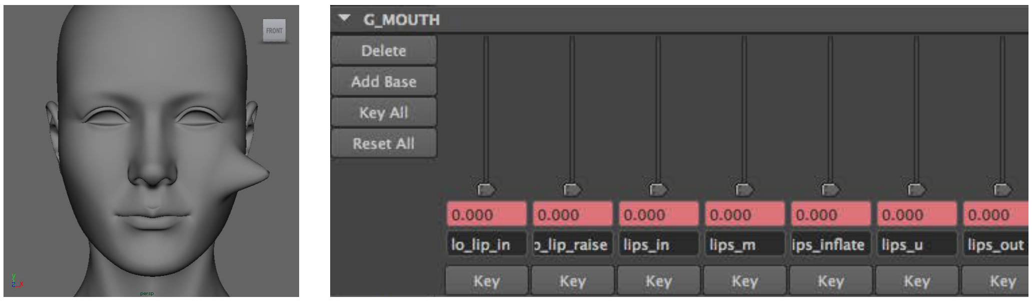


Figure 6.2: Tweaking for shape deformation. (Left) To deform a 3D model, naïvely tweaking each vertex position easily breaks the aesthetic quality of a shape. (Right) To prevent this, designers manually define deformation bases (rigs) that can be tweaked by sliders to reduce the design space. The images are taken from [84].

task as an optimization problem, and similar situations are likely to happen in many high-dimensional design spaces.

Note that, in practical scenarios on shape deformation, designers often manually define *rigs*—a set of bases of mesh deformation that are often tweak-able via sliders (see Figure 6.2 (Right))—in advance, to reduce the design space appropriately. In Chapter 3, we applied our crowd-powered estimation method to a shape deformation application (*i.e.,* facial expression modeling) where the design space is parametrized using a reasonable number of rig parameters in advance, rather than directly manipulating vertex positions.

The rigging process in shape deformation is considered as *dimensionality reduction* that is manually processed by designers. We consider that dimensionality reduction techniques [151] could be helpful for many problems with high-dimensional design spaces. However, this is still challenging because, unlike typical problems in data science, the resulting space in our case has to be either designer-friendly or optimization-friendly (or, both) for maximizing aesthetic preference. Recently, Yumer *et al.* [166] presented that autoencoder networks can be used for converting a high-dimensional, visually discontinuous design space to a lower-dimensional, visually continuous design space that is more desirable for design exploration. Incorporating human preference in dimensionality reduction of design spaces is an interesting future work.

6.3.2 Discrete Parameters

As we focused on only continuous parameters, our methods cannot handle discrete design parameters, such as fonts [111] and web design templates [39]. Remaining challenges to handle such discrete parameters include how to represent goodness functions for design spaces including discrete parameters, and how to facilitate users' interactive exploration. It is also a future work to extend our methods so that they can jointly handle discrete and continuous parameters.

6.3.3 Locally Optimal Design Alternatives

In some scenarios, totally different design alternatives can be equally "best" and it can be hard to determine which is better. For example, in Adobe Color CC [8], which is a user community platform to make, explore, and share *color palettes* (a set of colors usually consisting of five colors), there are a number of (almost)

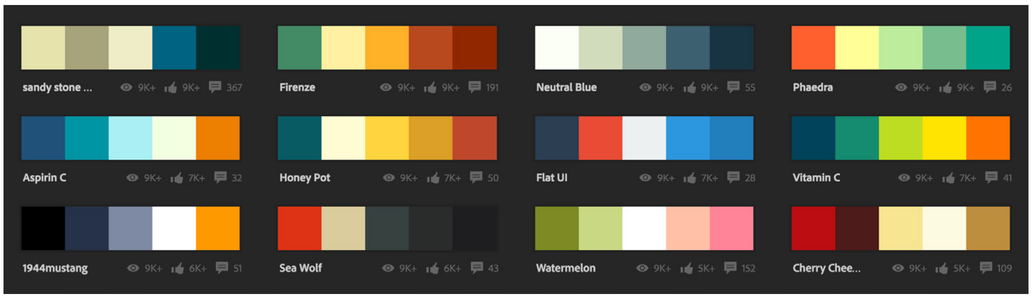


Figure 6.3: "Most Popular" color palettes in the user community of Adobe Color CC [8]. Though visually different from each other, they are (mostly) equally popular and preferred by many users.

equally popular color palettes that have been preferred by many users, as shown in Figure 6.3. In this case, if we assume the existence of a goodness function for color palettes, the popular palettes can be considered as local maximums of the goodness function. Considering that the goal is to support design activities, it may not be effective to assume that there is a sole global maximum in this design space and guide the user towards the global maximum; rather, it may be more desirable to provide a variety of good design alternatives so that the user can efficiently learn and choose from them. There is a room for investigation about how computation can support such design scenarios.

6.3.4 Evaluation Methodology

One of the issues in computational design driven by aesthetic preference is the lack of established, general methodology of evaluating each new method. Unlike other long-standing domains such as object recognition in images, where the validity of a method could be quantitatively measured via error rates, validation in our target domain is highly challenging by several reasons. The first reason is the difficulty of defining "correct" aesthetic preference, which can be highly dependent on scenarios. Also, as the ultimate goal is the support of design activities, the effectiveness needs to be assessed by human designers. In this thesis, we evaluated each method heuristically; for example, the effectiveness of the history-based estimation method is qualitatively evidenced through a user study from the viewpoint of human-computer interaction, and that of the crowd-powered maximization method is evaluated by comparing previous methods. Methods in this domain including ours are built on many assumptions, each of which is difficult to validate and in itself could be independent research topics. We consider that it is an important future work to establish general evaluation schemes.

6.3.5 More Complex Models of Crowd Behaviors

We built our crowd-powered methods on an assumption on crowd workers: crowd workers share a common goodness function, and each crowd worker responses based on the common goodness function with some noises. Thus, we assumed that we can observe the common goodness function by asking many crowd workers and then averaging their responses. This assumption may be valid in some scenarios but may not in many other scenarios; for example, aesthetic preference may differ between each individual, or crowds may form several clusters with respect to their aesthetic preference. Modeling such more complex properties of crowds is an important future challenge for enabling more broadly applicable crowd-powered computational design.

6.3.6 Incorporating Domain-Specific Heuristics

We have tried to use minimal domain-specific knowledges so that our methods are as general as possible. This allows our methods to be applied in various design domains, including photo color enhancement, light and camera setting, facial expression modeling, and material BRDF as demonstrated in this thesis. However, we suggest that our methods should be combined with domain-specific heuristics when deployed in practical specific scenarios. For example, if one builds a software to tweak viewpoints of 3D objects, the heuristic features and the pre-trained model in [124] could be jointly used with our methods. Seeking effective ways of such combinations is one of our future work.

6.3.7 Combining Crowd and Personal Preference

As we discussed, both the approach of learning crowds' general preference and that of learning users' personal preference have its advantages and disadvantages. To complement disadvantages of each approach, we envision that the combination of these two approaches is useful and worth investigating. For example, to support users' interactive design exploration, it could be effective to learn crowds' preference as pre-processing (*e.g.,* Chapter 3) and then learn personal preference in on-line session (*e.g.,* Chapter 4) but as the difference from crowds preference. This combinational approach has been partially investigated so far in specific domains [31, 71], but further investigation is desired to support more various design scenarios.

6.4 Future Directions

Finally, we conclude this thesis with discussions of several future research directions on computational design methods beyond parameter tweaking driven by aesthetic preference.

6.4.1 Free-Form Design from Scratch

Our target problem is parameter tweaking, which means that, in most cases, there exists a certain content before editing and it will be modified somehow (*e.g.,* an existing photograph will be modified by applying tonal curves). Thus, more *free-form* design activities, *e.g.,* drawing a picture on a canvas from scratch, are not considered. Yet, existence of goodness functions even for such design activities can be supposed. It is an open question whether we can handle such free-form designs as an extension of parameter tweaking, or we need to consider completely different approaches.

Some of from-scratch design activities can be interpreted as procedures (or sequences of commands) that can be simulated by machine processors. For example, even drawing acrylic paintings can be described as a sequence of executable painting commands [86]. In this case, the design space can be formulated as a tree structure whose leaves and edges represent visual designs and executable commands, respectively, and the design goal can be considered to find the best leaf node from this true. For this, tree search optimization techniques (*e.g.,* the branch and bound method) might be useful. Actually, this idea has been partially investigated in the research domain of computational design of graphical user interface [52]. While we focused on continuous optimization problems, various optimization formulations including tree search optimization are worth investigating as the next step.

6.4.2 More Complex Design Criteria

In practical design scenarios, designers may have to solve complex problems with more than one design criteria. For example, when a graphic designer designs an advertisement targeted at mainly women, he or she has to bias the goodness function towards women's aesthetic preference. In this case, it is possible to formulate the design problem as

$$\mathbf{x}^* = \mathop{\arg\max}_{\mathbf{x} \in \mathcal{X}} \{w_{\mathrm{male}} g_{\mathrm{male}}(\mathbf{x}) + w_{\mathrm{female}} g_{\mathrm{female}}(\mathbf{x})\}, \tag{6.1}$$

where $g_{\mathrm{male}}(\cdot)$ and $g_{\mathrm{female}}(\cdot)$ are the goodness functions owned by men and women, respectively, and w_{male} and w_{female} are the weights for adjusting the bias which can be $w_{\mathrm{male}} < w_{\mathrm{female}}$ in this case. Using crowdsourced human computation, this could be solved by utilizing demographic information of crowd workers, in the similar way by Reinecke and Gajos [118].

Another scenario is the case where functional criteria as well as aesthetic preference criteria are involved. For example, a designer may design a chair that has at least a certain durability and at the same time he or she may try to maximize its aesthetic quality. In this case, the problem can be formulated as a constrained optimization:

$$\mathbf{x}^* = \mathop{\arg\max}_{\mathbf{x} \in \mathcal{X}} g(\mathbf{x}) \text{ subject to } C(\mathbf{x}) \geq 0, \tag{6.2}$$

where $C(\cdot)$ is a function for measuring the functional criterion, *i.e.,* durability in this case, which returns positive values if the criterion is satisfied, *i.e.,* the chair has at least required durability. Note that, while several previous works on interactive design of functional objects (*e.g.,* [147, 130]) have tackled similar scenarios, their computational supports are typically only for ensuring the functional constraint $C(\mathbf{x}) \geq 0$, and the maximization of aesthetic preference totally relies on users' trials and errors. Investigating methods for such design scenarios with complex design criteria is an important extension of our research.

6.4.3 Computational Creative Design

In this thesis, we have considered aesthetic preference as the target criterion in design activities. Another important aspect of design activities may be *creativity.* While we did not included creativity in our scope, we believe that our computational design formulations are extensible for incorporating creativity along with aesthetic preference as the criteria. For this purpose, discussions and techniques in the emerging research field called *computational creativity* [158] may be useful. For example, Cohen-Or and Zhang [42] described about creativity in the geometric modeling context that

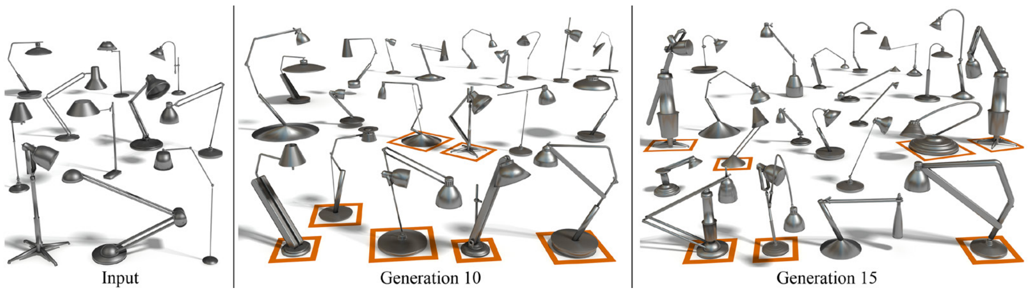


Figure 6.4: An example of design optimization frameworks that explicitly consider creativity. Xu *et al.* [163] proposed an evolutionary computation method that takes *diversity* into account for enabling creative 3D shape modeling.

> "*. . . creative inspirations to modelers are often in the form of new models that were not envisioned and contain certain elements of surprise or unexpectedness.*"

According to them, the key to provide creative inspirations to designers is considered to be *unexpectedness*. Some researchers have interpreted such unexpectedness as *diversity* in design alternatives and explicitly formulated optimization problems for finding the most diverse (and plausible) set of design alternatives [163, 13, 161]. For example, Xu *et al.* [163] proposed an evolutionary computation method for enabling creative 3D shape modeling, in which a set of designs evolves so that each of them is preferable and, at the same time, they are diverse (see Figure 6.4). We believe that investigating new ways for incorporating creativity into our design formulations (*e.g.,* adding a new term for enhancing creativity to the objective function) or our entire frameworks (*e.g.,* providing additional creative cues in the user interfaces) is a very interesting future direction from the viewpoints of both developing new computational technology and designing new user experience.

References

[1] Amazon Mechanical Turk. https://www.mturk.com/. Last checked: October 24, 2016.

[2] CrowdFlower. https://www.crowdflower.com/. Last checked: October 31, 2016.

[3] Hard Surface Shaders Free. https://www.assetstore.unity3d.com/#/content/729. Last checked: November 12, 2016.

[4] Morguefile. https://morguefile.com/. Last checked: November 12, 2016.

[5] Thingiverse. http://www.thingiverse.com/. Last checked: December 8, 2016.

[6] Upwork. https://www.upwork.com/. Last checked: October 24, 2016.

[7] Adobe Systems Inc. Adobe After Effects CC. http://www.adobe.com/products/aftereffects.html.

[8] Adobe Systems Inc. Adobe Color CC. https://color.adobe.com/.

[9] Adobe Systems Inc. Adobe Photoshop CC. http://www.adobe.com/products/photoshop.html.

[10] Adobe Systems Inc. Adobe Photoshop Lightroom CC. http://www.adobe.com/products/photoshop-lightroom.html.

[11] Adobe Systems Inc. Photoshop elements help | auto smart tone. https://helpx.adobe.com/photoshop-elements/using/auto-smart-tone.html. Last checked: September 23, 2015.

[12] Adobe Systems Inc. Photoshop help | creating actions. https://helpx.adobe.com/photoshop/using/creating-actions.html. Last checked: September 23, 2015.

[13] Shailen Agrawal, Shuo Shen, and Michiel van de Panne. Diverse motion variations for physics-based character animation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, pages 37–44, 2013.

[14] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph.*, 22(3):587–594, July 2003.

[15] Ken Anjyo, J. P. Lewis, and Frédéric Pighin. Scattered data interpolation for computer graphics. In *ACM SIGGRAPH 2014 Courses*, SIGGRAPH '14, pages 27:1–27:69, 2014.

[16] Autodesk Inc. Maya. http://www.autodesk.com/products/maya/overview.

[17] Javad Azimi, Alan Fern, and Xiaoli Z. Fern. Batch bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems 23*, NIPS '10, pages 109–117, 2010.

[18] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph.*, 33(4):96:1–96:10, July 2014.

[19] Aaron Bangor, Philip T. Kortum, and James T. Miller. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[20] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. Graph.*, 33(4):159:1–159:12, July 2014.

[21] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Opensurfaces: A richly annotated catalog of surface appearance. *ACM Trans. Graph.*, 32(4):111:1–111:17, July 2013.

[22] Luca Benedetti, Holger Winnemöller, Massimiliano Corsini, and Roberto Scopigno. Painting with bob: Assisted creativity for novices. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 419–428, 2014.

[23] Yoshua Bengio and Pascal Vincent. Locally weighted full covariance gaussian density estimation. Cirano working papers, CIRANO, 2004.

[24] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 33–42, 2011.

[25] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: A word processor with a crowd inside. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 313–322, 2010.

[26] Floraine Berthouzoz, Wilmot Li, Mira Dontcheva, and Maneesh Agrawala. A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. *ACM Trans. Graph.*, 30(5):120:1–120:14, October 2011.

[27] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom Yeh. Vizwiz: Nearly real-time answers to visual

questions. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 333–342, 2010.

[28] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., 1995.

[29] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 187–194, 1999.

[30] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[31] Eric Brochu, Tyson Brochu, and Nando de Freitas. A bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 103–112, 2010.

[32] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010. arXiv:1012.2599.

[33] Eric Brochu, Nando de Freitas, and Abhijeet Ghosh. Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems 20*, NIPS '07, pages 409–416, 2007.

[34] Eric Brochu, Abhijeet Ghosh, and Nando de Freitas. Preference galleries for material design. In *ACM SIGGRAPH 2007 Posters*, SIGGRAPH '07, 2007.

[35] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 97–104, 2011.

[36] Juan C. Caicedo, Ashish Kapoor, and Sing Bing Kang. Collaborative personalization of image enhancement. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 249–256, June 2011.

[37] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 67–76, 2001.

[38] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Trans. Graph.*, 34(4):139:1–139:11, July 2015.

[39] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. Attribit: Content creation with semantic attributes.

In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 193–202, 2013.

[40] Fanny Chevalier, Pierre Dragicevic, and Christophe Hurter. Histomages: Fully synchronized views for image editing. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 281–286, 2012.

[41] Wei Chu and Zoubin Ghahramani. Preference learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 137–144, 2005.

[42] Daniel Cohen-Or and Hao Zhang. From inspired modeling to creative modeling. *The Visual Computer*, 32(1):7–14, 2016.

[43] Mark Colbert, Sumanta Pattanaik, and Jaroslav Krivanek. Brdf-shop: Creating physically correct bidirectional reflectance distribution functions. *IEEE Comput. Graph. Appl.*, 26(1):30–36, January 2006.

[44] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, ECML-PKDD '13, pages 225–240, 2013.

[45] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popovic, and Foldit players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 08 2010.

[46] Michael A. A. Cox and Trevor F. Cox. Multidimensional scaling. In *Handbook of Data Visualization*, Springer Handbooks Comp.Statistics, pages 315–347. Springer Berlin Heidelberg, 2008.

[47] Allen Cypher. Eager: Programming repetitive tasks by example. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 33–39, 1991.

[48] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Studying aesthetics in photographic images using a computational approach. In *Proceedings of the 9th European Conference on Computer Vision*, ECCV'06, pages 288–301, 2006.

[49] Dynamo. WeAreDynamo Wiki. http://wiki.wearedynamo.org/. Last checked: October 24, 2016.

[50] Mark Ebden. Gaussian processes: A quick introduction, 2015. arXiv:1505.02965.

[51] Leah Findlater and Jacob Wobbrock. Personalized input: Improving ten-finger touchscreen typing through automatic adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 815–824, 2012.

[52] Krzysztof Gajos and Daniel S. Weld. Supple: Automatically generating user interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, IUI '04, pages 93–100, 2004.

[53] Krzysztof Z. Gajos, Mary Czerwinski, Desney S. Tan, and Daniel S. Weld. Exploring the design space for adaptive graphical user interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '06, pages 201–208, 2006.

[54] Elena Garces, Aseem Agarwala, Diego Gutierrez, and Aaron Hertzmann. A similarity measure for illustration style. *ACM Trans. Graph.*, 33(4):93:1–93:9, July 2014.

[55] Yotam Gingold, Ariel Shamir, and Daniel Cohen-Or. Micro perceptual human computation for visual tasks. *ACM Trans. Graph.*, 31(5):119:1–119:12, September 2012.

[56] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph.*, 28(3):66:1–66:9, July 2009.

[57] Anhong Guo, Xiang 'Anthony' Chen, Haoran Qi, Samuel White, Suman Ghosh, Chieko Asakawa, and Jeffrey P. Bigham. Vizlens: A robust and interactive screen reader for interfaces in the real world. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 651–664, 2016.

[58] Yoav HaCohen, Eli Shechtman, Dan B. Goldman, and Dani Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM Trans. Graph.*, 30(4):70:1–70:10, July 2011.

[59] Yoav HaCohen, Eli Shechtman, Dan B. Goldman, and Dani Lischinski. Optimizing color consistency in photo collections. *ACM Trans. Graph.*, 32(4):38:1–38:10, July 2013.

[60] Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R. Klemmer. Design as exploration: Creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pages 91–100, 2008.

[61] Jeff Howe. Crowdsourcing: A definition. http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html, 2006. Last checked: October 23, 2016.

[62] Jeff Howe. The rise of crowdsourcing. https://www.wired.com/2006/06/crowds/, 2006. Last checked: October 23, 2016.

[63] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 409–416, 1999.

[64] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, 2010.

[65] Ronnachai Jaroensri, Sylvain Paris, Aaron Hertzmann, Vladimir Bychkovsky, and Frédo Durand. Predicting range of acceptable photographic tonal adjustments. In *Proceedings of the 2015 IEEE International Conference on Computational Photography*, ICCP '15, pages 1–9, April 2015.

[66] Steven G. Johnson. The nlopt nonlinear-optimization package, 2015.

[67] Donald R. Jones, Cary Drake Perttunen, and Bruce E. Stuckman. Lipschitzian optimization without the lipschitz constant. *J. Optim. Theory Appl.*, 79(1):157–181, October 1993.

[68] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, December 1998.

[69] Hiroshi Kajino, Yuta Tsuboi, and Hisashi Kashima. Clustering crowds. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI '13, pages 1120–1127, 2013.

[70] Sing Bing Kang, Ashish Kapoor, and Dani Lischinski. Personalization of image enhancement. In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 1799–1806, June 2010.

[71] Ashish Kapoor, Juan C. Caicedo, Dani Lischinski, and Sing Bing Kang. Collaborative personalization of image enhancement. *International Journal of Computer Vision*, 108(1):148–164, 2014.

[72] Yan Ke, Xiaoou Tang, and Feng Jing. The design of high-level features for photo quality assessment. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '06, pages 419–426, 2006.

[73] Naoki Kita and Kazunori Miyata. Aesthetic rating and color suggestion for color palettes. *Computer Graphics Forum*, 35(7):127–136, 2016.

[74] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, 2008.

[75] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. Crowd-powered parameter analysis for visual design exploration. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 65–74, 2014.

[76] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. Selph: Progressive learning and support of manual photo color enhancement. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2520–2532, 2016.

[77] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. Sequential line search for visual design optimization by crowds. Under review.

[78] Yuki Koyama, Shinjiro Sueda, Emma Steinhardt, Takeo Igarashi, Ariel Shamir, and Wojciech Matusik. Autoconnect: Computational design of 3d-printable connectors. *ACM Trans. Graph.*, 34(6):231:1–231:11, October 2015.

[79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, NIPS '12, pages 1097–1105, 2012.

[80] Brian Kulis. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

[81] Tessa Lau, Lawrence Bergman, Vittorio Castelli, and Daniel Oblinger. Sheepdog: Learning procedures for technical support. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, IUI '04, pages 109–116, 2004.

[82] Lasse Farnung Laursen, Yuki Koyama, Hsiang-Ting Chen, Elena Garces, Richard Harper Diego Gutierrez, and Takeo Igarashi. Icon set selection via human computation. In *Proc. Pacific Graphics 2016 – Short Papers*, pages 1–6, 2016.

[83] Matthew Lease. On quality control and machine learning in crowdsourcing. In *Proceedings of AAAI Workshops*, pages 97–102, 2011.

[84] J. P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. Practice and theory of blendshape facial models. In *Eurographics 2014 - State of the Art Reports*, Eurographics '14, pages 199–218, 2014.

[85] Congcong Li, Alexander C. Loui, and Tsuhan Chen. Towards aesthetics: A photo quality assessment and photo selection system. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 827–830, 2010.

[86] Thomas Lindemeier, Jens Metzner, Lena Pollak, and Oliver Deussen. Hardware-based non-photorealistic rendering using a painting robot. *Computer Graphics Forum*, 34(2):311–323, 2015.

[87] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. Turkit: Human computation algorithms on mechanical turk. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 57–66, 2010.

[88] Greg Little, Tessa A. Lau, Allen Cypher, James Lin, Eben M. Haber, and Eser Kandogan. Koala: Capture, share, automate, personalize business processes on the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 943–946, 2007.

[89] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(3):503–528, December 1989.

[90] Ligang Liu, Renjie Chen, Lior Wolf, and Daniel Cohen-Or. Optimizing photo composition. *Computer Graphics Forum*, 29(2):469–478, 2010.

[91] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. Style compatibility for 3d furniture models. *ACM Trans. Graph.*, 34(4):85:1–85:9, July 2015.

[92] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6):248:1–248:16, October 2015.

[93] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph.*, 33(4):97:1–97:10, July 2014.

[94] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. Elements of style: Learning perceptual shape style similarity. *ACM Trans. Graph.*, 34(4):84:1–84:14, July 2015.

[95] Yiwen Luo and Xiaoou Tang. Photo and video quality evaluation: Focusing on the subject. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 386–399, 2008.

[96] Luca Marchesotti, Florent Perronnin, Diane Larlus, and Gabriela Csurka. Assessing the aesthetic quality of photographs using generic image descriptors. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 1784–1791, 2011.

[97] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 389–400, 1997.

[98] Tobias Martin, Nobuyuki Umetani, and Bernd Bickel. Omniad: Data-driven omni-directional aerodynamics. *ACM Trans. Graph.*, 34(4):113:1–113:12, July 2015.

[99] Jonàs Martínez, Jérémie Dumas, Sylvain Lefebvre, and Li-Yi Wei. Structure and appearance optimization for controllable shape design. *ACM Trans. Graph.*, 34(6):229:1–229:11, October 2015.

[100] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3):759–769, July 2003.

[101] Stephen McAuley, Stephen Hill, Naty Hoffman, Yoshiharu Gotanda, Brian Smits, Brent Burley, and Adam Martinez. Practical physically-based shading in film and game production. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, pages 10:1–10:7, 2012.

[102] Jonas Mockus. On bayesian methods for seeking the extremum. In *Proceedings of IFIP Technical Conference on Optimization Techniques '74*, pages 400–404, 1974.

[103] Morihiro Nakamura, Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. An interactive design system of free-formed bamboo-copters. *Computer Graphics Forum*, 35(7):323–332, 2016.

[104] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, GRAPHITE '06, pages 381–389, 2006.

[105] Addy Ngan, Frédo Durand, and Wojciech Matusik. Image-driven navigation of analytical BRDF models. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, EGSR '06, pages 399–407, 2006.

[106] Jannik Boll Nielsen, Henrik Wann Jensen, and Ravi Ramamoorthi. On optimal, minimal brdf sampling for reflectance acquisition. *ACM Trans. Graph.*, 34(6):186:1–186:11, October 2015.

[107] Masashi Nishiyama, Takahiro Okabe, Imari Sato, and Yoichi Sato. Aesthetic quality classification of photographs based on color harmony. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 33–40, June 2011.

[108] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Color compatibility from large datasets. *ACM Trans. Graph.*, 30(4):63:1–63:12, July 2011.

[109] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning layouts for single-page graphic designs. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1200–1213, August 2014.

[110] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. DesignScape: Design with interactive layout suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1221–1224, 2015.

[111] Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory font selection using crowdsourced attributes. *ACM Trans. Graph.*, 33(4):92:1–92:9, July 2014.

[112] Makoto Okabe, Yasuyuki Matsushita, Li Shen, and Takeo Igarashi. Illumination brush: Interactive design of all-frequency lighting. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, PG '07, pages 171–180, 2007.

[113] Ken Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, July 1985.

[114] Ken Perlin. Improving noise. *ACM Trans. Graph.*, 21(3):681–682, July 2002.

[115] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph.*, 32(4):81:1–81:10, July 2013.

[116] Alexander J. Quinn and Benjamin B. Bederson. Human computation: A survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1403–1412, 2011.

[117] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

[118] Katharina Reinecke and Krzysztof Z. Gajos. Quantifying visual preferences around the world. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 11–20, 2014.

[119] Daniela Retelny, Sébastien Robaszkiewicz, Alexandra To, Walter S. Lasecki, Jay Patel, Negar Rahmati, Tulsee Doshi, Melissa Valentine, and Michael S. Bernstein. Expert crowdsourcing with flash teams. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 75–85, 2014.

[120] Mark Sagar. Facial performance capture and expressive translation for king kong. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, 2006.

[121] Niloufar Salehi, Lilly C. Irani, Michael S. Bernstein, Ali Alkhatib, Eva Ogbe, Kristy Milland, and Clickhappier. We are dynamo: Overcoming stalling and friction in collective action for crowd workers. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1621–1630, 2015.

[122] Juliane Schäfer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1), November 2005.

[123] Matthias Schonlau, William J. Welch, and Donald R. Jones. *Global versus local search in constrained optimization of computer models*, volume 34 of *Lecture Notes–Monograph Series*, pages 11–25. Institute of Mathematical Statistics, Hayward, CA, 1998.

[124] Adrian Secord, Jingwan Lu, Adam Finkelstein, Manish Singh, and Andrew Nealen. Perceptual models of viewpoint preference. *ACM Trans. Graph.*, 30(5):109:1–109:12, October 2011.

[125] Ana Serrano, Diego Gutierrez, Karol Myszkowski, Hans-Peter Seidel, and Belen Masia. An intuitive control space for material appearance. *ACM Trans. Graph.*, 35(6):186:1–186:12, November 2016.

[126] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, Madison, WI, USA, 2009.

[127] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, January 2016.

[128] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Image appearance exploration by model-based navigation. *Computer Graphics Forum*, 28(2):629–638, 2009.

[129] Gaurav Sharma, Wencheng Wu, and Edul N. Dalal. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1):21–30, 2005.

[130] Maria Shugrina, Ariel Shamir, and Wojciech Matusik. Fab forms: Customizable objects for fabrication with validity and geometry caching. *ACM Trans. Graph.*, 34(4):100:1–100:12, July 2015.

[131] Leonid Sigal, Moshe Mahler, Spencer Diaz, Kyna McIntosh, Elizabeth Carter, Timothy Richards, and Jessica Hodgins. A perceptual control space for garment simulation. *ACM Trans. Graph.*, 34(4):117:1–117:10, July 2015.

[132] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, NIPS '12, pages 2951–2959, 2012.

[133] Olga Sorkine. Differential representations for mesh processing. *Computer Graphics Forum*, 25(4):789–807, 2006.

[134] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. Stress relief: Improving structural strength of 3D printable objects. *ACM Trans. Graph.*, 31(4):48:1–48:11, July 2012.

[135] Stephan Streuber, M. Alejandra Quiros-Ramirez, Matthew Q. Hill, Carina A. Hahn, Silvia Zuffi, Alice O'Toole, and Michael J. Black. Body talk: Crowdshaping realistic 3d avatars with words. *ACM Trans. Graph.*, 35(4):54:1–54:14, July 2016.

[136] Ryo Suzuki, Niloufar Salehi, Michelle S. Lam, Juan C. Marroquin, and Michael S. Bernstein. Atelier: Repurposing expert crowdsourcing tasks as micro-internships. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2645–2656, 2016.

[137] Daniel Sýkora, David Sedlacek, Sun Jinchao, John Dingliana, and Steven Collins. Adding depth to cartoons using sparse depth (in)equalities. *Computer Graphics Forum*, 29(2):615–623, 2010.

[138] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[139] Jerry O. Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. Exploratory modeling with collaborative design spaces. *ACM Trans. Graph.*, 28(5):167:1–167:10, December 2009.

[140] Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2):11:1–11:14, April 2011.

[141] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Kalai. Adaptively learning the crowd kernel. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pages 673–680, June 2011.

[142] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[143] Michael Terry and Elizabeth D. Mynatt. Side views: Persistent, on-demand previews for open-ended tasks. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, UIST '02, pages 71–80, 2002.

[144] Michael Terry, Elizabeth D. Mynatt, Kumiyo Nakakoji, and Yasuhiro Yamamoto. Variation in element and action: Supporting simultaneous development of alternative solutions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 711–718, 2004.

[145] The GIMP Team. GIMP - GNU Image Manipulation Program. https://www.gimp.org/.

[146] Kristi Tsukida and Maya R. Gupta. How to analyze paired comparison data. Technical report, University of Washington, 2011.

[147] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4):86:1–86:11, July 2012.

[148] Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph.*, 30(4):90:1–90:12, July 2011.

[149] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.*, 33(4):65:1–65:10, July 2014.

[150] Unity Technologies. Unity. https://unity3d.com/.

[151] Laurens van der Maaten, Eric Postma, and Jaap van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC TR 2009–005, TiCC, Tilburg University, 2009.

[152] Luis von Ahn. *Human Computation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2005. AAI3205378.

[153] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, 2004.

[154] Luis von Ahn and Laura Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8):58–67, August 2008.

[155] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[156] Lingfeng Wang and Emily Whiting. Buoyancy optimization for computational fabrication. *Computer Graphics Forum*, 35(2):49–58, 2016.

[157] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 119–128, 1995.

[158] Wikipedia. Computational creativity - Wikipedia. https://en.wikipedia.org/wiki/Computational_creativity. Last checked: November 11, 2016.

[159] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, November 2007.

[160] Andrew Witkin and Michael Kass. Spacetime constraints. *SIGGRAPH Comput. Graph.*, 22(4):159–168, June 1988.

[161] Jungdam Won, Kyungho Lee, Carol O'Sullivan, Jessica K. Hodgins, and Jehee Lee. Generating and ranking diverse multi-character interactions. *ACM Trans. Graph.*, 33(6):219:1–219:12, November 2014.

[162] Anbang Xu, Shih-Wen Huang, and Brian Bailey. Voyant: Generating structured feedback on visual designs using a crowd of non-experts. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 1433–1444, 2014.

[163] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Fit and diverse: Set evolution for inspiring 3d shape galleries. *ACM Trans. Graph.*, 31(4):57:1–57:10, July 2012.

[164] Daiki Yamanaka, Hiromasa Suzuki, and Yutaka Ohtake. Density aware shape modeling to control mass properties of 3d printed objects. In *SIGGRAPH Asia 2014 Technical Briefs*, SA '14, pages 7:1–7:4, 2014.

[165] Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. Automatic photo adjustment using deep neural networks. *ACM Trans. Graph.*, 35(2):11:1–11:15, February 2016.

[166] Mehmet Ersin Yumer, Paul Asente, Radomir Mech, and Levent Burak Kara. Procedural modeling using autoencoder networks. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*, UIST '15, pages 109–118, 2015.

[167] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K. Hodgins, and Levent Burak Kara. Semantic shape editing using deformation handles. *ACM Trans. Graph.*, 34(4):86:1–86:12, July 2015.

[168] Jun-Yan Zhu, Aseem Agarwala, Alexei A. Efros, Eli Shechtman, and Jue Wang. Mirror mirror: Crowdsourcing better portraits. *ACM Trans. Graph.*, 33(6):234:1–234:12, November 2014.