Doctral Dissertation

Computational Design of Discrete Element Layouts

Naoki Kita

Supervisor: Kazunori Miyata

Graduate School of Advanced Science and Technology (Knowledge Science)

Japan Advanced Institute of Science and Technology

March 2019

Abstract

Arranging elements in a design domain is an essential task in visual design because human visual perception is sensitive to the layout of the elements. Moreover, aesthetic preferences for a design differ from person to person in essence. Such multimodality complicates the formulation of an optimization problem of whether the human visual perception or aesthetic preference is the objective to be optimized. Due to these difficulties, computational design tools for discrete element layouts are presently limited.

In this thesis, we propose three computational design tools for discrete element layouts. We tackle these problems with reasonable assumptions by making them tractable. In the first method, we focus on a spatial distribution of discrete elements with different appearances. Because it is difficult and tedious for a user with manual operations to distribute different element spatially in a uniform manner, we propose a procedural method to distribute multi-class (different appearances) elements in which the spatial uniformity of the element's distribution is considered as the objective.
In the second method, we focus on a discrete color arrangement, i.e., color palettes, and propose a computational design tool for rating a given palette and suggesting an additional compatible color for the palette. As human color preference differs from person to person or one culture to another, we employ a machine learning approach to address this problem. By customizing a training dataset, we can tailor a model for any color preference and suggest compatible colors to users. Finally, based on a visual cryptography scheme, particularly the secret sharing scheme, we propose a method for generating special patterns that can reveal secret patterns when superimposed on other patterns. The interesting part of our method is that the secret to be decrypted is changed based on the other patterns to be superimposed. We optimized the generated patterns with visual quality as the objective.

We analyzed the proposed methods both quantitatively and qualitatively including user studies. Additionally, we demonstrated various applications of the proposed methods, which show their applicability in broadening areas of discrete element layouts. However, we only cover portions of discrete element layouts. Notably, owing to the abstract nature of the problem, the discrete element layout fields are vast and include many concrete application scenarios. We believe that this thesis is a significant contribution to the advancement of the study of computational design for discrete element layouts.

Keywords*: Computer Graphics, Computational Design, Discrete Element Layout, Color Palette Design, Visual Cryptography*

Acknowledgments

Without discussions and support from many people, I could never have completed this work. While I explore the possibility of computational design tools to support users' creativity in this thesis, this creative and labored work itself is mostly made by crystallization of creative discussions with people.

I would like to express my sincere gratitude to my supervisor Prof. Miyata. He has provided me with several opportunities to advance this research. I have learned a lot from him since I started my research in the master course and realized that I could grow as a person. I have learned how to collaborate with colleagues through group work on *Spider Hero*. Additionally, I have learned how to proceed with my own research topics from him. Owing to his suggestion, I could have an opportunity to research at a design school in France.

I am sincerely grateful to Grégoire Cliquet and members of READi Design Lab at L'École de design Nantes Atlantique. I have greatly enjoyed my research life as well as daily life in Nantes, France. The staffs and students in READi Design Lab are very kind. The stay with them is invaluable. I also appreciate Toni Da Luz and Remy Eynard. I enjoyed working with them. Remy invited me on a short trip when I stayed in France, which is also a treasured memory for me.

I am grateful to all the thesis committee members: Prof. Jun Mitani from the University of Tsukuba, Prof. Kazushi Nishimoto, Prof. Takashi Hashimoto, and Prof. Hideaki Kanai, for their insightful comments regarding improvement of this thesis.

I am also grateful to the members of Miyata laboratory. In particular, I thank Matthieu Tessier. He is very creative and his ideas always inspired me. I would also like to thank Akiyoshi Itoda for organizing several recreations, including dinner parties on alternating weeks. I also appreciate Keisuke Terada and Susumu Shimonaka for continual discussions about various interesting topics.

Finally, I am grateful to my family for their continuous support.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

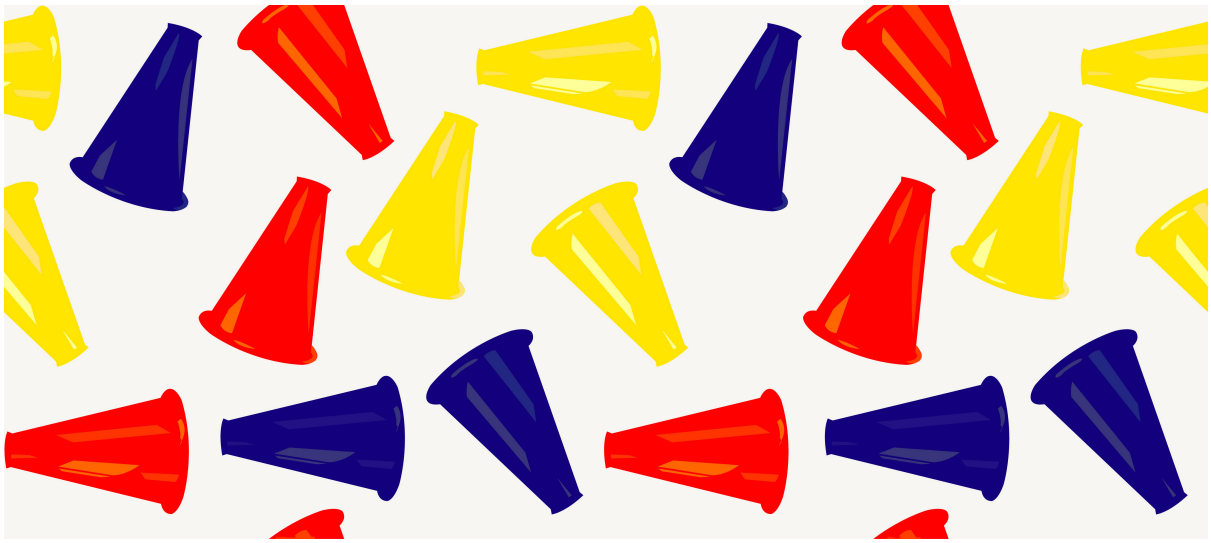


FIGURE 1.1 Example of a design composed of discrete elements.

Layout plays a crucial role in visual design. Moreover, it is well-known that human visual perception is sensitive to how the elements—*components composing the layout*—are arranged. Furniture layouts, patterns on wallpaper, products on display in a shop, to mention a few are examples of a layout. Figure 1.1 shows an example of a layout design composed of discrete elements (colored traffic cones). Is it arranged in a regular pattern? Periodic? Are there salient regions in the layout? Do you like the layout?

In this thesis, we studied a layout design which is composed of different discrete elements, by mainly focusing on its geometric/non-geometric attributes. The task of arranging discrete elements is mechanical rather than intellectual, but considering the aesthetics or functionalities of layout design is a highly intellectual and creative activity. It is worth noting that designing a layout may take dozens of trial-and-error attempts. For each trial-and-error loop, one has to arrange the elements to obtain a new design, which is tedious but can be supported by some computational tools.

For example, consider designing a new wallpaper with floral patterns, and flowers as the

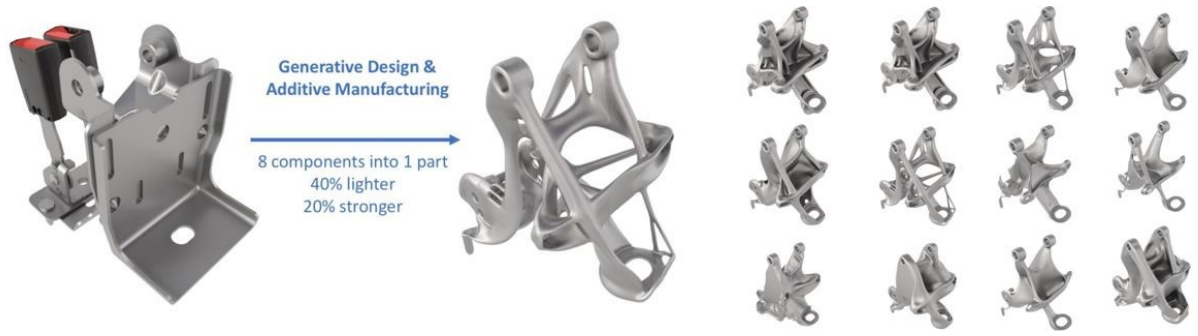


FIGURE 1.2 Example of a generative design. Image courtesy of Autodesk [3].

discrete elements. In this case, one can use a graphics editor to arrange the elements in a given domain. If the pattern is simple and\or periodic, it may not be difficult but tedious. On the other hand, if the pattern is aperiodic and\or intricate, that is, one has to consider design specifications such as spatial uniformity or inter-element distance (e.g., overlap-free), it will be a difficult task and also more tedious. Although several graphics editors provide basic editing tools, they do not fully support intricate editing; hence, the need for some manual work effort to accomplish such tasks. The motivation of this thesis is to provide methods that efficiently support such an intellectual and creative task. Aside from supporting the tedious arrangement tasks, the proposed methods have the capability to explore new designs which are difficult to obtain by human imaginations only. Precisely, in this thesis, we propose *computational design* tools for discrete element layouts, which allow a user to arrange a large number of elements efficiently. In addition, the proposed methods are controllable and support the exploration of the user's design in a vast solution space.

Computational design is a concept that involves the creation of new designs by a user with the aid of a computer. While the computer generates design candidates from user-specified parameters or design specifications, the user modifies the candidates, tweak the parameters, or provide other design specifications to create optimal designs. Although there are several definitions of *computational design*, it is often referred to as *parametric design*. In this thesis, we use the term computational design rather than parametric design; because in our tools, parameters are not explicitly provided to a user, though, of course, a user can tweak them. We provided user interactive tools for design exploration. By *generative design* tools, we refer to tools that can generate multiple variants of a design using the user-provided design specifications.

In recent times, the computational (parametric) design is considerably becoming popular because of its capability to generate diverse designs from its defining parameters. Similarly, the generative design is also gaining more publicity because of its capability to generate diverse variants of a design which are highly unimaginable. Moreover, a generative design tool can generate various designs that satisfy the user-provided design specifications (see Figure 1.2) [3]. For example, a computer-aided design application such as the Autodesk Fusion 360, provides a generative design tool for optimizing material distribution in a volumetric model using the user-provided design specifications.

While computational design tools are being employed in architectural designs, digital fabri-

cation, and similar design fields that are related to geometric modeling, there are not applicable in the visual design field. Because to implement computational design tools for visual designs one has to consider human preferences and\or human visual perception; and these human-related objectives are difficult to formulate due to their multimodalities. Therefore, there is a need to carefully consider the objectives, its assumptions, and algorithms that will solve it.

1.2 Scope of this Thesis

A layout is composed of arranged elements. Conversely, each element is arranged in a target domain, or we can say that every element has spatial attributes such as position, shape, size, and orientation. Besides, an element can have attributes such as colors and textures. Both spatial arrangement and element's appearance affect our visual perception because the visual attention is sensitive to those factors. Therefore, it is of interest to focus the scope of this thesis on the geometric spatial arrangements and non-geometric attributes of elements (see Figure 1.3).

It is well-known that the discrete element layouts cover various specific scenarios including page layouts (such as web page) and poster layouts, or furniture layouts. These specific sub-domains have been extensively studied. Elements of these layouts play specific roles. For example, a web page (poster) layout is composed of discrete elements such as texts and images. The texts play the role of explaining the other elements or the creators' intention, while images or graphs are used to represent the creator's intention. Therefore, there exist strong relationships among the elements, which introduce complex constraints for their arrangements. Similarly, each furniture arrangement is constrained by the other furniture. Because the design specifications constraints can clearly be defined, it is possible to formulate the layout problem as an optimization problem that optimizes a user-provided objective, subject to these constraints. This framework has been adopted by many researchers, which resulted in attracting considerable attention to this type of domain. In fact, discrete element layouts of specific scenarios with concrete elements, have been widely studied.

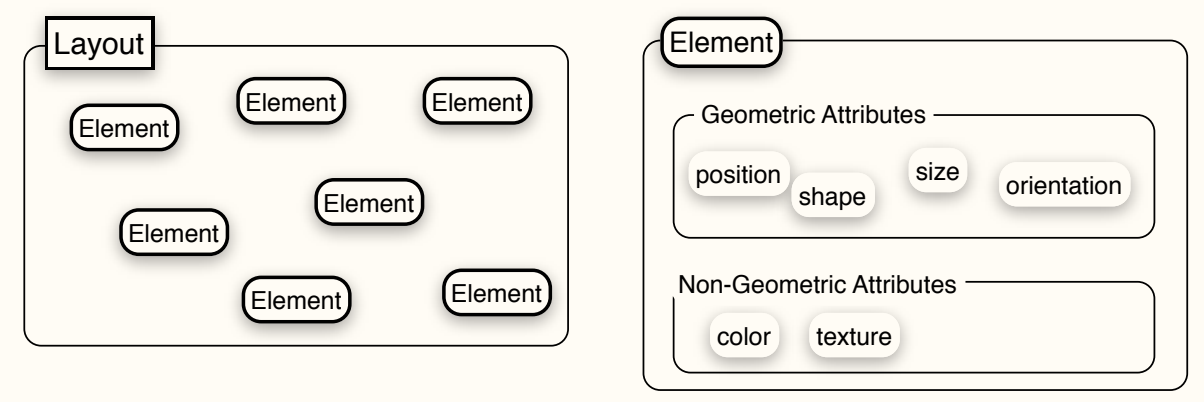


FIGURE 1.3 A layout is composed of elements. An element has attributes including geometric attributes such as spatial position, shape, size, and orientation; and non-geometric attributes such as colors and textures.

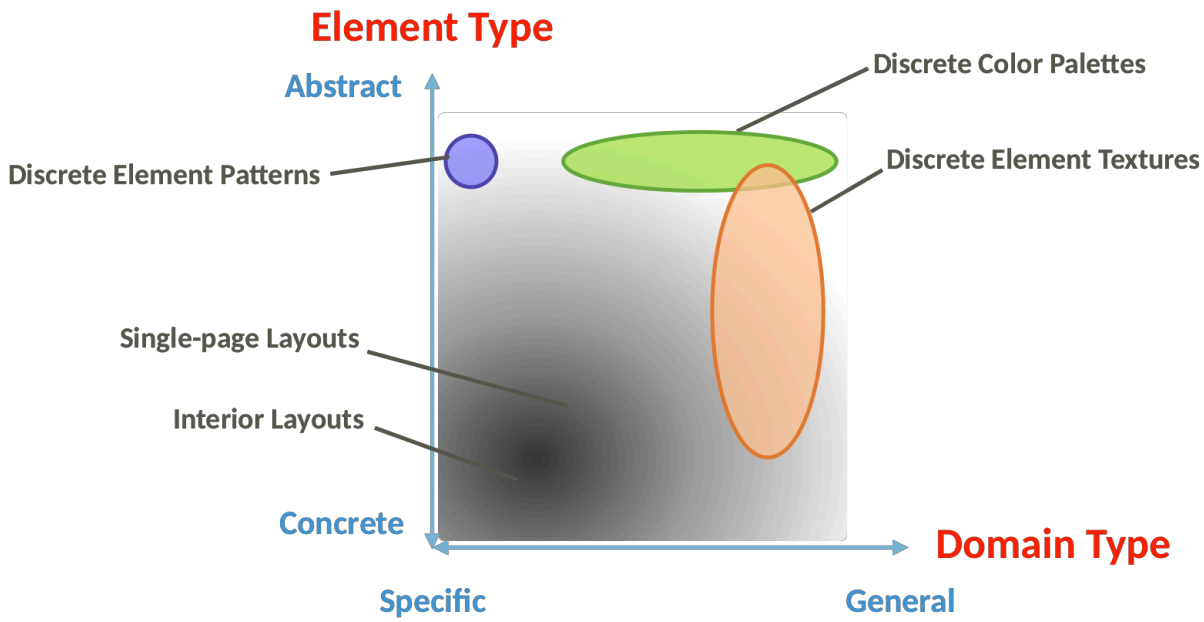


FIGURE 1.4 An overview of the focal areas of the proposed methods (areas in orange, green, and purple colors). In this thesis, we focus mainly on general purpose discrete element layouts with more abstract elements, rather than specific target domain with concrete elements which have been extensively studied (areas colored by darker gray, including single-page and interior layouts).

We remark that discrete element layouts that arrange abstract elements such as primitives or colors have not gained considerable attention to the present. Notwithstanding, layouts of abstract elements are worth studying because the methods of arranging these elements are widely applicable in various scenarios such as spatial and color designs. Obviously, spatial and color designs are covered by a wide area of discrete element layouts. In Chapter 2, we reviewed in detail, related works on spatial layouts and color designs.

Figure 1.4 describes the areas that can be covered by the proposed methods in comparison with existing results colored by gray (areas with darker gray suggest these areas gain extensive attention and vice versa). As described above, we mainly focus on widely applicable, i.e., general purpose discrete element layouts with more abstract elements rather than specific target domain (scenarios) with concrete elements; which have been extensively studied to the present. Among various non-geometric attributes of an element, we focused on colors rather than textures, especially in the combination of colors. This is because texture tends to be more application specific, while color is a more abstract attribute and employed in wider applications.

1.3 Goal of this Thesis

The goal of this thesis is to provide computational design tools for discrete element layouts that have not been extensively explored, in particular, the general layout domain with abstract elements shown in Figure 1.4; so as to allow users to easily and efficiently explore those design space. To support the exploration of users' design, the following requirements must to be met:

R1 The proposed tools should be able to generate design suggestions at an interactive rate to

provide efficient trial-and-error design loop.

R2 The proposed tools should cover a wider range of design domains, specifically, the general layout domain with abstract elements.

R3 The proposed tools should have the capacity to support both upcoming and professional designers in carrying out design exploration.

To ascertain the proposed tools satisfy the above requirements, we provide a validation list below.

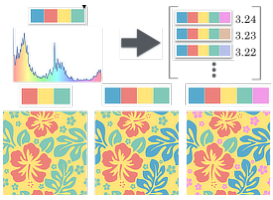
V1 For each tool, the performance evaluations of its functionalities should be provided.

V2 The applications of the tools should be provided so as to determine its versatility.

V3 The user studies or feedbacks from new and professional users of the tools should be provided to confirm its usefulness.

1.4 Our Approach

TABLE 1.1 An overview of our approach.

Research Topic	Discrete Element Textures (Chapter 3)	Discrete Color Palettes (Chapter 4)	Discrete Elemenet Patterns (Chapter 5)
			
Element	object image	color	pixel
Layout Domain	spatial distribution	color combination	pixel arrangement
Constraint Type	geometric (spatial)	non-geometric (color)	both (spatial/color)
Objective	spatial uniformity	color compatibility	image quality
Method	procedural	machine learning	combinatorial optimization

We investigated discrete element layouts with human visual perceptions. Table 1.1 shows the summary of our approaches. We considered the following two aspects of discrete element layouts with human visual perceptions: geometric attributes such as spatial distributions and non-geometric attributes such as colors. We also present in this thesis, a spatial element distribution method, and a color combination evaluation method. Then, considering both geometric and non-geometric attributes, we propose a method.

1) **Discrete Element Textures.** As mentioned earlier, in the case of floral patterns, each flower is regarded as a discrete element since it has a distinctive boundary; hence, we can easily distinguish any two of them. Therefore, we call textures which comprise discrete

elements as *discrete element textures*. In contrast, tree bark textures, for example, are regarded as *continuous textures*, and commonly referred to as *textures*. While creating periodic patterns composed of discrete elements is relatively simple, it is not straightforward to create aperiodic patterns; which of course requires lots of manual effort. We propose a method for generating discrete element patterns. The human-related objective here is "spatial uniformity" in a given domain. By spatial uniformity, we mean that each discrete element is located uniformly at random in the domain. When we looked at the whole domain as well as the sub-domain, the elements were observed to be uniformly distributed. The method proposed in this thesis is shown to have the capacity to generate discrete element textures at an interactive rate by employing a grid subdivision algorithm. In addition, a user can control and edit the distribution of the elements. Therefore, the proposed method makes the user's trial-and-error loop more efficient. Furthermore, we approximated user-provided object images by proxy shapes that are used when performing elements' overlap checks during object distribution generation process. As the proposed method does not depend on the element type, it can be applied to various element types ranging from abstract to concrete (see Figure 1.4). By integrating previous object distribution approaches as well as fast distribution generation algorithm, the proposed method was shown to have the capability to generate various visually appealing textures.

2) **Discrete Color Palettes.** In one of our research, we propose a method for rating and suggesting color palettes using a machine learning approach. A discrete color palette refers to a color palette composed of a set of distinctive colors (color combination). Discrete color palettes are used in data visualization for representing quantitative data or as "color theme" in art and graphic design. Due to cultural influence or habit, preference of color combinations differs from person to person or region to region. Therefore, it is challenging to create a general color rating model. To tackle this difficulty, we employed a machine learning approach. Precisely, we train a color rating model using a large dataset of human aesthetic ratings of color palettes. The model can be tweaked by training with a dedicated dataset (e.g., *pastel colors*, or *Japanese traditional colors*). This approach is very efficient when dealing with human-related multimodal objectives. Therefore, a wide variety of domains ranging from general to specific purpose can be covered using custom model (cf. Figure 1.4). In addition, using the proposed trained model, new color candidates that will be compatible with a given color palette can be suggested; this, of course, broadens users' exploration space of color design.

3) **Discrete Element Patterns.** In another of our research, we proposed a method for creating striking patterns that excite people's *sense of wonder* when superimposing a couple of patterns. In visual cryptography scheme (VCS) a secret image is encrypted into two unstructured, noisy patterns (*shares*). The secret image can be decrypted if and only if the two shares are superimposed; otherwise no information can be obtained from a single pattern. Inspired by VCS, we extended this method to a more entertainment-oriented one, in which two or more secret images are encrypted into a couple of structured shares.

One of the shares acts as a *key* (*common share*). The secret can be decrypted by superimposing the common share and an appropriate share corresponding to a specific secret. In this research, share patterns composed of image pixels are regarded as discrete elements. We optimize the pixel arrangements of shares to improve the visual quality while the superimposing results remain unchanged. Compared to the two methods discussed earlier, the proposed method here focuses on a more domain-specific application as shown in Figure 1.4. However, in this thesis, we considered both geometric and non-geometric constraints as design specifications, and optimize the output patterns under these constraints, hence the method provides a thought-provoking framework for tackling other domains including general purpose discrete element layouts with concrete elements (the lower right region in Figure 1.4).

1.5 Thesis Overview

This thesis is organized as follows:

- In **Chapter 2**, a detailed review of related work for discrete element layouts followed by related work for each topic is provided, in preparation for subsequent chapters (Chapter 3 and 4).

- In **Chapter 3**, a description of the computational design tools for **discrete element textures** is given. Presented also, are details on how to integrate previous methods to generalize them, and also some considerations for performance improvements. The demonstration of the controllability of elements distribution, as well as several results, is presented in this chapter. The proposed algorithm was evaluated using an analysis tool for a point set distribution. In comparison to state-of-the-art methods, a user study was conducted to evaluate users' preference for the generated textures. It i worthy of note that these methods are not directly comparable to ours; since they are based on texture synthesis and the resulting patterns highly depend on the distribution properties of its exemplar, whereas the method proposed in this thesis generates textures from the scratch. This research reveals that "spatial uniformity" plays a crucial role in human preference of element distribution.

- In **Chapter 4**, the computational design tools for **discrete color palettes** are described. Provided also, are details on the kind of dataset used, how to extract feature vectors, train a model, and customize the model to a specific color tone. Using the trained model, a description of a compatible color suggestion and color space exploration methods is also provided. Furthermore, a detailed analysis of the trained model, the effects of adjustable parameters, and the model customization results, is presented. The results of the user study evaluation conducted are also presented in this chapter. Since the color design has a wide range of applications in graphic design, several applications of the proposed method including pattern coloring and photo recoloring, are provided.

- In **Chapter 5**, a description of the computational design tools for **discrete element patterns** is presented. Details on how to decompose secret images into structured patterns (with a *key* pattern) and how to optimize these patterns are provided. The resulting patterns with\without the share pattern optimization were analyzed. To demonstrate the proposed method, the proposed algorithm is applied to binary, grayscale, and color images. In addition, the patterns were printed on transparencies and superimposed to decrypt the secret images.

- Finally, a summary of the research carried out here a conclusion of this thesis, as well as future perspectives, are presented in **Chapter 7**.

CHAPTER 2

Related Work

This chapter introduces related work for this thesis. First, we review the related work for discrete element layouts, followed by the review of related work for *discrete element textures* (Chapter 3), *discrete color palettes* (Chapter 4), and *discrete element patterns* (Chapter 5).

2.1 Related Work for Discrete Element Layouts

In this section, we review the related work for discrete element layouts. As the target domain of discrete element layouts is large, there are various research topics in this category. Here we review typical topics in computer graphics community; page and interior layouts. We also provide a review of approaches to generating versatile layouts. We will review more specific research topics in Section 2.2, 2.3, and 2.4.

2.1.1 Single-page Layouts

O'Donovan et al. proposed an approach for automatically creating single-page graphic design layouts using an energy-based model derived from design principles such as alignment and balance [74]. Similarly, Bylinskii et al. proposed a model for predicting the relative importance of different elements in data visualizations and graphics designs [12]. Cao et al. proposed a method for automatically generating manga layouts from a set of artworks with user-specified semantics using a generative probabilistic model [13]. Several approaches focus on generating layouts for directing users' attention to a given path [14, 81]. These approaches are suitable for creating single-page layouts under the constraints of inter-element relationships.

2.1.2 Interior Layouts

Yu et al. proposed an approach for automatically synthesizing furniture layouts using simulated annealing [108]. Merrell proposed an interactive furniture layout system that assists users by suggesting furniture arrangements that are based on interior design guidelines [67]. By employing parallel tempering, the proposed system can generate a variety of optimized suggestions in less than seconds. Similarly, several approaches have been proposed for generating building layouts (see [4, 33, 66, 106]). To create generative models, most of these approaches utilize

design principals specific to the target layouts or learn relationships between elements. Therefore, they cover more domain-specific areas with concrete element types shown in Figure 1.4. Numerous researches have been focused on similar areas, while we focus on the more general purpose, abstracted element layouts.

While single-page layouts focus on arranging discrete elements such as texts and images, interior layouts focus on arranging furniture and rooms in a layout domain. However, both layouts are similar in that layouts are to be created taking into account the domain-specific constraints of the inter-element relationships. Both elements and layouts tend to be domain-specific; they cover the lower left regions in Figure 1.4. Depending on the abstraction levels, the target domain and element types can be expanded to some extent, resulting in covering areas from lower left to the center in Figure 1.4.

2.1.3 Versatile Layout Generation Techniques

Gomez-Nieto et al. proposed a technique for generating structured layouts by formulating the problem as a mixed integer optimization problem, where multiple requirements are considered simultaneously [36]. Fried et al. proposed a framework for creating informative layouts with user-provided pairwise distances of elements [34]. The proposed method first projects elements using a multidimensional projection. Then, the projected elements are arranged in a pre-defined grid by computing a bipartite matching with user-provided distances as the weights between the locations of the elements and those of grid cells.

Ritchie et al. proposed a probabilistic programming technique for generating design suggestions under tight constraints [88]. They employed the Hamiltonian Monte Carlo algorithm, which can efficiently generate design suggestions even though the solution space is highly-constrained.

Since these approaches can arrange abstract elements in versatile layouts, the techniques can be applied to a wide range of discrete element layouts. Both approaches cover similar areas to our methods or cover more general and abstract areas than ours in Figure 1.4. However, there are limitations to those approaches when applied to our purpose. Although both approaches arrange abstract elements in a space based on user-provided design requirements, it is difficult for a user to provide the requirements since they can be mathematically formulated constraints and are not intuitive. For example, a user has to explicitly provide pairwise distances of any pairs of elements, which is a difficult and time-consuming task. In addition, both approaches [34, 36] employ optimization techniques and cannot produce results at a moderate speed if the number of elements gets larger, hence restricting users' efficient design exploration.

2.2 Related Work for Discrete Element Textures

As a basis for the proposed algorithms in Chapter 3, we review blue noise sampling methods and its applications in discrete element distribution.

2.2.1 Blue Noise Sampling

Since the first dart-throwing method was proposed [29, 68], various methods have been proposed to generate samples with blue noise properties. In a frequency domain, a blue noise spectrum has a signature lack of low-frequency components and concentrated spikes. Samples with blue noise properties are distributed randomly but remain spatially uniform. Hence, sampling with blue noise properties is essential for rendering in computer graphics. Numerous applications have already been proposed. Therefore, here we refer to excellent survey papers [51, 107] and focus on methods more closely related to our application.

In a simple dart-throwing method, all samples are at least distance r apart from each other, where r is specified by a user, and we call the size r exclusive region a disk. While such a disk is isotropic, Li et al. proposed an anisotropic sampling method [54]. By employing a Jacobian matrix applied locally to the domain-specific function used to determine a desired sample distance and anisotropy, its distance metric approximates to an inter-sample distance. On the other hand, Wei proposed multi-class blue noise sampling [103]. In multi-class sampling, while each single class sample has blue noise properties, their union also has blue noise properties (Figure 2.2). After Wei's proposal, various multi-class blue noise sampling approaches have been proposed [19, 44, 93].

We extend the anisotropic sampling method proposed by Li et al. [54] and the multi-class blue noise sampling method proposed by Wei [103] to accommodate element distributions. Therefore we call our sampling method *multi-class anisotropic blue noise sampling*. In our configuration, we approximate an element shape by two types of proxy shape: circle and ellipse. When checking for conflicts between two elements, we apply a local Jacobian matrix if the proxy shape is an ellipse; otherwise, we apply Euclidean distance for our distance metric. In addition, our samples corresponding to each element have class IDs. The class IDs are used for conflict checks as well as local Jacobian matrix computation. Since the original multi-class sampling method cannot explicitly specify the inter-class distance, we modify the $\mathbf{r}$-matrix

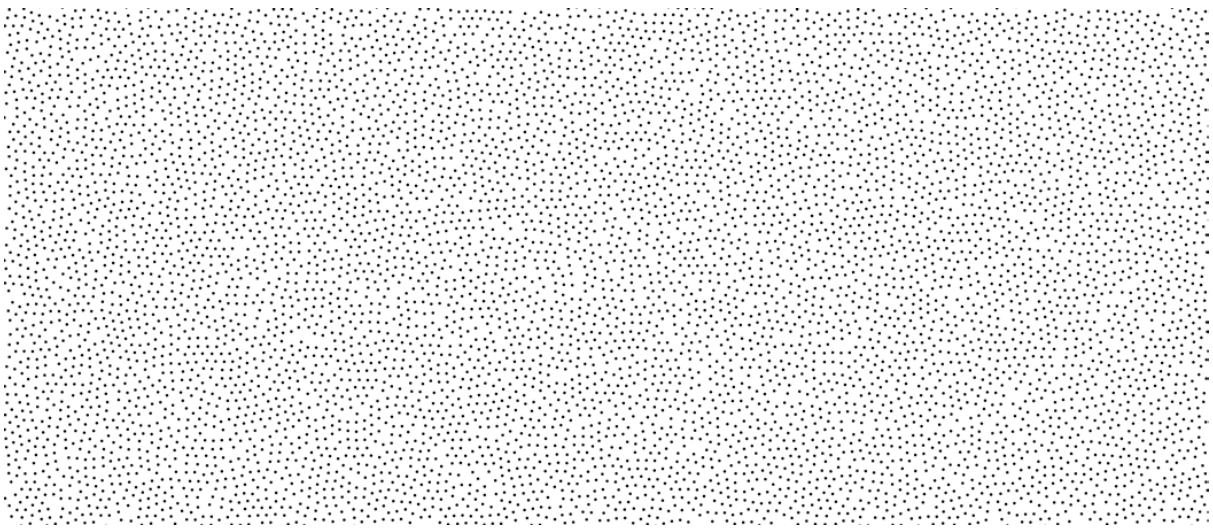


FIGURE 2.1 Poisson-disk distributions have excellent blue noise characteristics [31].

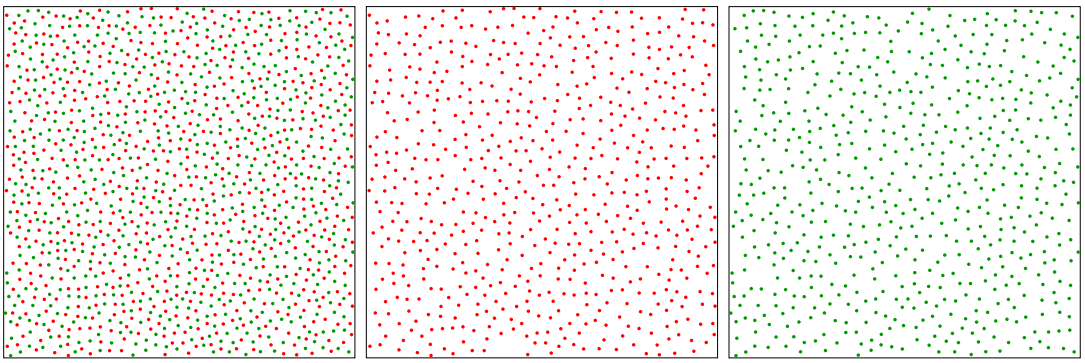


FIGURE 2.2 Multi-class blue noise sampling [103]. In the two-class case, the both class 0 and class 1 exhibit blue noise properties as well as the total set.

proposed in [103]. In our method, the $\mathbf{r}$-matrix acts as a scaling factor to a proxy shape.

2.2.2 Spectral Analysis of Point Sets

For analyzing the quality of a sampling pattern generated using the method proposed in this thesis, we employ the tool provided by Schlömer and Deussen [92]. The description of the analysis is provided in the following paragraphs.

The quality of the distribution of a point set is measured by applying spectral analysis. The *power density spectrum* $P(f)$ of a stationary stochastic process is the Fourier transform of its autocorrelation function. Although we cannot know the autocorrelation function in reality, we can estimate it. The estimate of the autocorrelation function $P(f)$, i.e., *periodogram*; $\hat{P}(f)$ for a given point set $\{x_0, \ldots, x_{n-1}\} \in [0,1)^2$ can be obtained as follows:

$$\hat{P}(f) = \frac{1}{n}\left|\mathcal{F}\sum_{i=0}^{n-1}\delta(x - x_i)\right|^2, \tag{2.1}$$

where $\mathcal{F}$ is the Fourier transform and δ is Dirac's delta function. Averaging K periodograms yields an unbiased and consistent estimate for the power spectrum [85].

Ulichney provided two tools with which we can measure the directional artifacts for one-dimensional statistics using a power spectrum estimate $\hat{P}(f)$ [100]. The first one is the *radially averaged power spectrum* and is defined as follows:

$$P_r(f_r) = \frac{1}{N_r(f_r)}\sum_{i=1}^{N_r(f_r)}\hat{P}(f), \tag{2.2}$$

where $\hat{P}(f)$ is partitioned by concentric annuli of size Δ and averaging $N_r(f_r)$ is frequency samples within each annulus with radius f_r.

The second one is the *anisotropy* and is defined as follows:

$$A_r(f_r) = \frac{s^2(f_r)}{P_r^2(f_r)}, \tag{2.3}$$

where $s^2(f_r)$ is the variance of the frequency samples and given by

$$s^2(f_r) = \frac{1}{N_r(f_r) - 1} \sum_{i=1}^{N_r(f_r)} \left(\hat{P}(f) - P_r(f_r)\right)^2. \tag{2.4}$$

The anisotropy measures the radial symmetry of the spectrum in decibels.

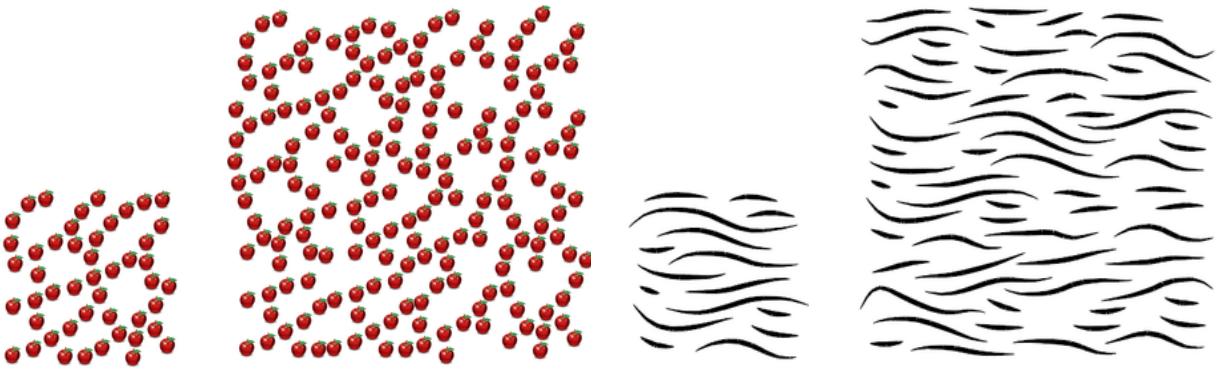
2.2.3 Discrete Element Distribution

Several discrete element distribution methods have been proposed. The packing approach includes an extension of Lloyd's Method [38], spectral packing [30], and relaxation [86,87]. The example-based texture synthesis approach employs statistical analysis of the exemplar [41,42], synthesis with element properties and its positions [62], patch-based synthesis [2], and sampling from a stochastic model [52]. Few approaches employ dart throwing methods [50,90].

While the packing approaches can densely arrange objects, they cannot generate spatially uniform multi-class (e.g. color attributes) element distribution. It can be the same for the dart throwing approaches (Figure 2.3).

In contrast, example-based approaches successfully capture the statistics of the input exemplar and synthesize larger textures similar to the exemplar (Figure 2.4). However, creating such an appropriate exemplar (in most cases, visually appealing ones might be expected) is not easy for most people, especially for novices. In addition, the above-mentioned approaches except for [42] lack user controllability for distributed elements while our approach has a user interface for controlling element distributions.

FIGURE 2.3 Procedural object distribution using Poisson-disk distribution [50].



(a) input exemplar (b) synthesized (c) input exemplar (d) synthesized

FIGURE 2.4 Discrete element texture synthesis proposed by [62]. The input exemplars (a) and (c) are used to synthesize larger textures (b) and (d), respectively.

2.3 Related Work for Discrete Color Palettes

In this section, we briefly review color harmony theories and its applications related to the proposed method in Chapter 4.

2.3.1 Color Harmony Theory

Color harmony theory as a research topic did not begin until after Newton reported his experiments with the light spectrum in 1672. Various color harmony theories, such as Goethe's thought-provoking color theory [35] (Figure 2.5 left), were proposed near the end of the 18th century and throughout the 19th century. In the early 20th century, the first practical *color-order-system* were derived from the color theories proposed by Munsell [8] and Ostwald [76]. The color theory proposed by Moon and Spencer [71], which was based on the mathematical analysis of a user study, represents the beginning of modern color harmony theory. Following Moon and Spencer, Itten proposed a color theory which is defined on a hue wheel [43] (Figure 2.5

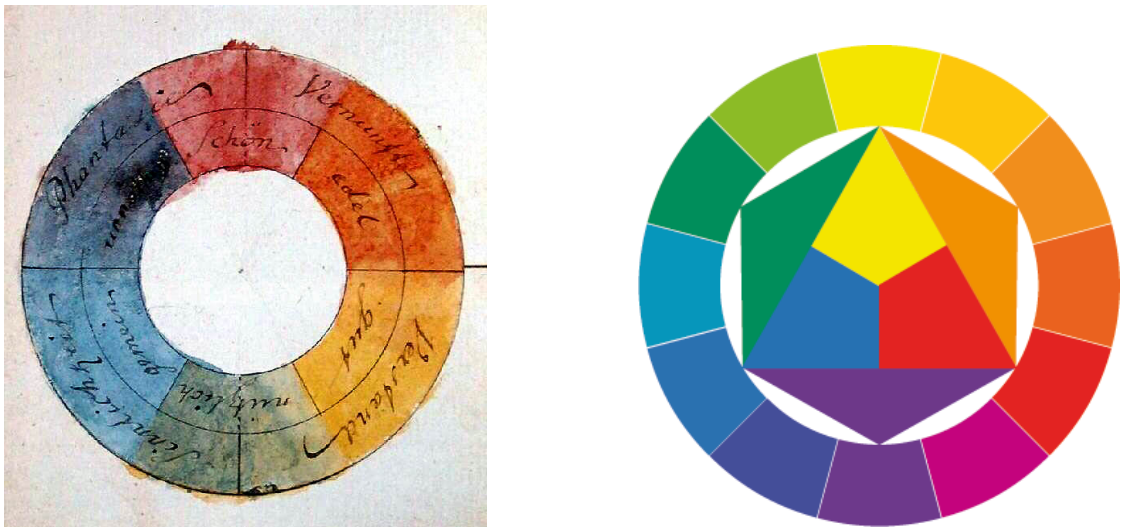


FIGURE 2.5 Left: Goethe's color wheel, where colors at opposite side of the color wheel are compatible according to his theory. Right: Itten's color wheel. According to his theory, colors are compatible if they have geometric relationships on the color wheel.

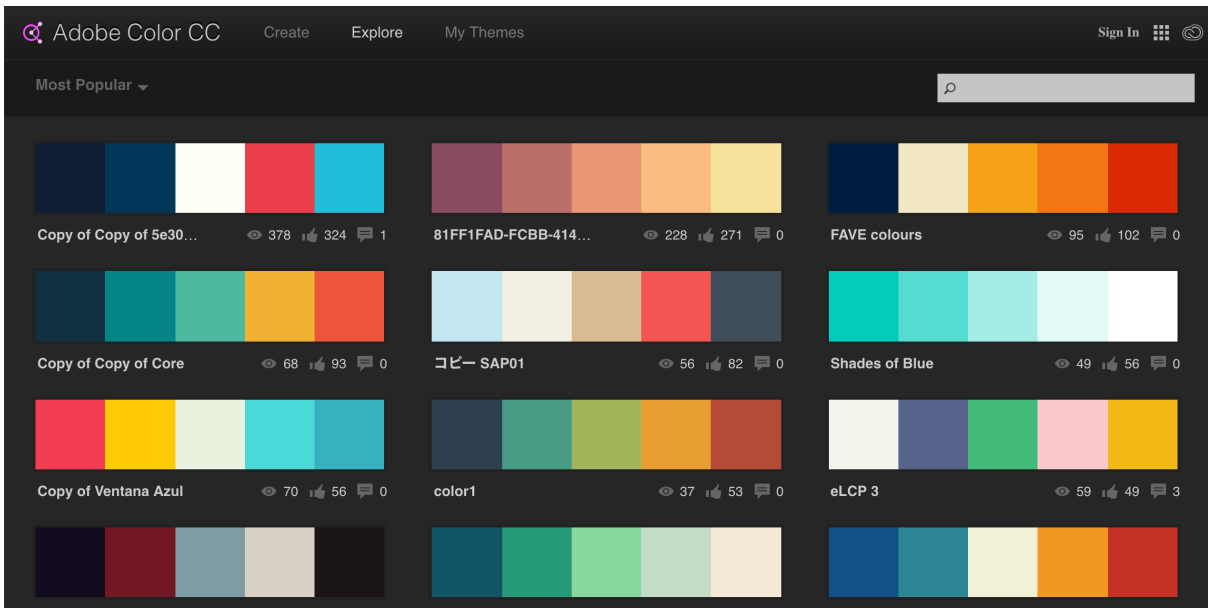


FIGURE 2.6 Adobe Color CC [23]. We can create custom color palettes and share with the community.

right). While colors at opposite side of the wheel are compatible in Goethe's theory, colors are compatible if they have geometric relationships on the color wheel according to Itten.

2.3.2 Color Palette Design

Wijffelaars et al. proposed a tool for generating univariate lightness ordered palettes [105]. Their tool is suitable for generating sequential and diverging color palettes. Phan et al. proposed Color Orchestra, which can predicts and interpolates palettes by learning color palette manifold from palettes extracted from fine art collections [83]. Shugrina et al. proposed Playful Palette, a color picker interface that combines the advantages of both digital and physical painting [96]. Mellado et al. proposed a constrained-based interactive palette exploration system [65]. With their graph-based palette representation, the system optimizes the palettes efficiently and allows real-time feedback to the users. These tools, however, do not aim for creating color palettes composed of compatible (harmonious) colors.

There are online tools available for creating color palettes. We can look for color themes, create, or share own themes on Adobe Color CC [23] (Figure 2.6) or COLOURlovers [24]. ColorBrewer provides well-designed color palettes for cartography [37].

2.3.3 Color Harmony Model

Many color harmony models based on the mathematical analysis of user study results have been proposed [77–79, 98]. However, such models only evaluate two or three color combinations with a small number of participants (fewer than one hundred), which restricts the generalization of the model. On the other hand, O'Donovan et al. collected $327,381$ human ratings of $22,376$ color themes (palettes) using Amazon Mechanical Turk (MTurk) and proposed a model trained using a machine learning algorithm [73]. They applied the trained model for various ap-

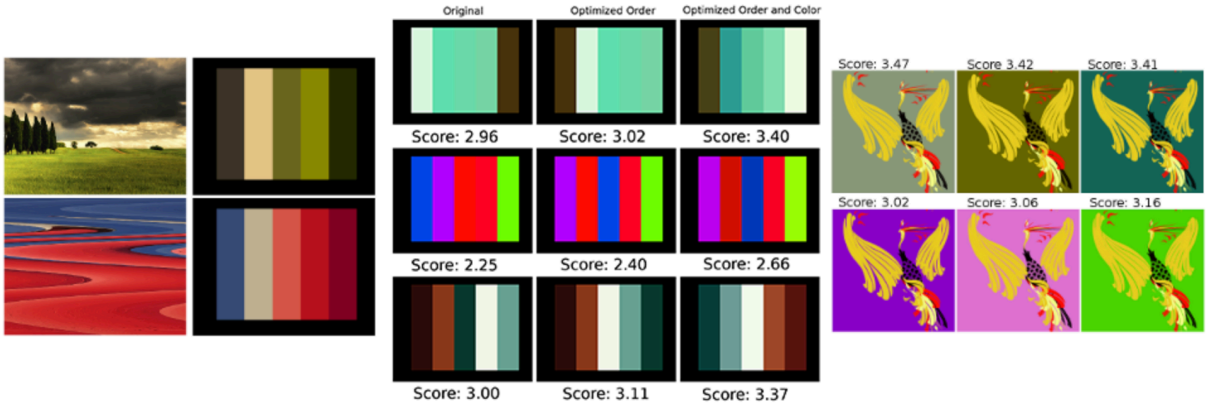


FIGURE 2.7 O'Donovan et al. applied their color theme rating model for theme extraction (left), theme optimization (middle), and color suggestions (right) [73].

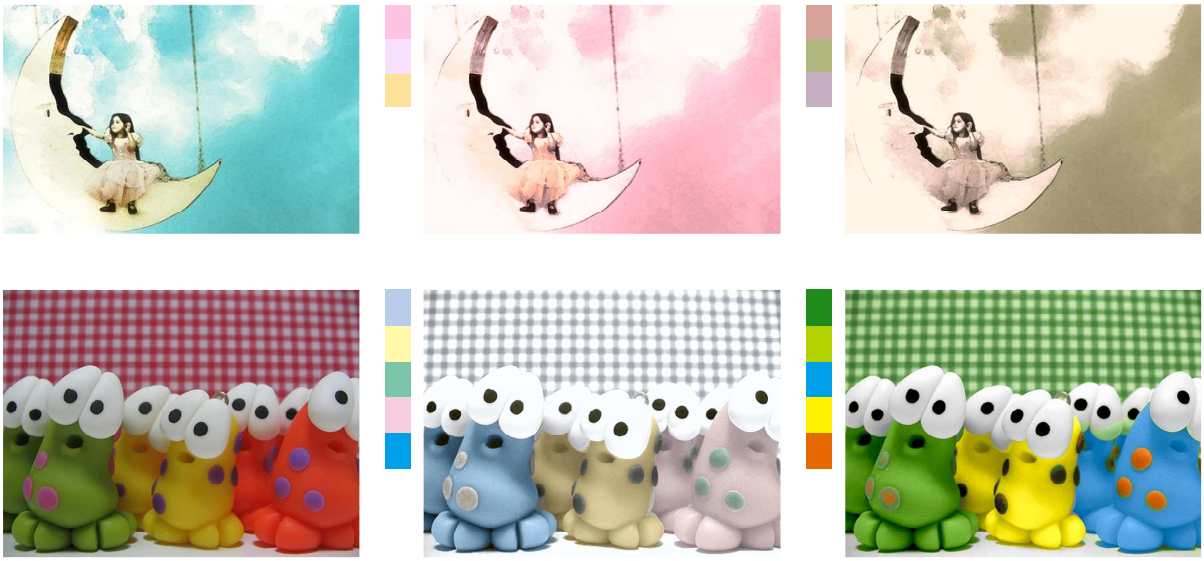


FIGURE 2.8 Color theme enhancement results from [101]. The original images (first column) are recolored with different color theme enhancements using different color palettes (second and third columns).

plication (Figure 2.7). Lin et al. also collected $1,600$ data items relative to how people extract color themes from images from 160 MTurk participants [56].

These models can only rate two-, three-, or five-color palettes, which is too limited because color palettes are typically composed as many as seven colors. In contrast to previous approaches, the proposed method is not limited to a specific number of colors in a palette, i.e., our feature extraction method and machine learning method can handle a color palette composed of any number of colors, which allows us to rate any color palette. Therefore, the proposed method can be applied to a wide range of applications, such as compatible color suggestion, 2D pattern coloring, and other fields of color design.

2.3.4 Applications

Cohen-Or et al. proposed a method that employed a *harmonic template* on a hue wheel to harmonize the target image colors [22], and Li et al. proposed a fast image recolorization algo-

FIGURE 2.9 Palette-based photo recoloring results from [15].

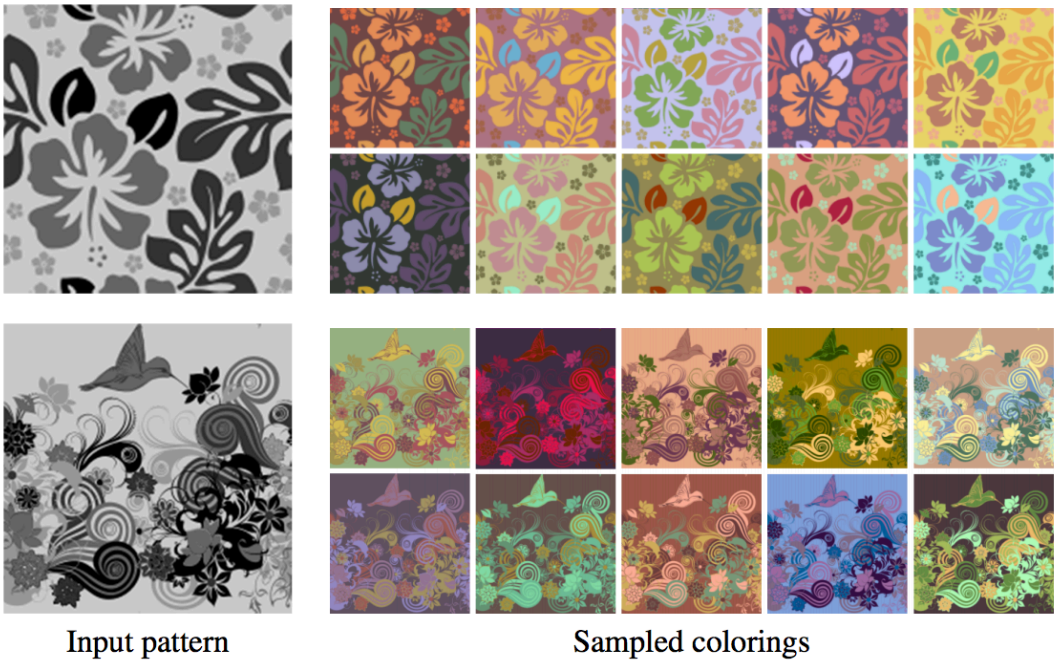


FIGURE 2.10 Pattern coloring using a probabilistic factor graph model [57].

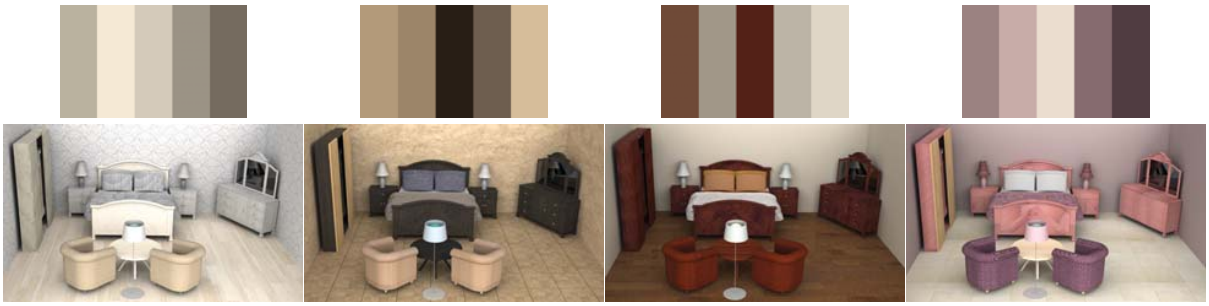


FIGURE 2.11 Color themes are used for stylized room decorations in *magic decorator* proposed by Chen et al. [18].

rithm by employing geodesic distance-based color harmonization [55]. Sawant et al. applied color harmonization for videos [91], and Zhang et al. employed a color scheme replacement method in their video stream abstraction method for stylization [111]. Wang et al. proposed a data-driven approach to convert a target image based on a given color theme [101] (Figure 2.8), while Chang et al. proposed a method for recoloring a photograph based on a color palette [15] (Figure 2.9). Lin et al. proposed a probabilistic factor graph model to color 2D patterns au-

tomatically [57] (Figure 2.10), and Kim et al. proposed a method based on color perception theories to assign color to 2D patterns automatically [46]. Employing the color compatibility rating model proposed in [73] as the one of the terms to be optimized in the optimization problems they formulated, Yu et al. proposed outfit synthesis (fashion design) [109], and, as mentioned previously, Lin et al. employed the model ratings as a global aesthetic term for 2D pattern coloring [57]. Chen et al. proposed automatic material suggestion for indoor digital scenes [18] (Figure 2.11).

Since [18, 57, 109] employed the rating model proposed in [73], their method is limited to only five-color palettes in their methods, which limits the scope of application. In contrast, we propose a feature extraction method that does not depend on the number of colors in a given palette. Therefore, we can suggest a compatible color for a given palette with any number of colors. For example, given a three-color palette as input, the palette can be expanded to four-, five-, or even seven-color palettes using our color suggestion method while maintaining color harmony.

2.4 Related work for Discrete Element Patterns

In preparation for Chapter 5, we provide a brief history of *visual cryptography*. We also review the related researches in hiding multiple images.

2.4.1 Visual cryptography

VC was first proposed by Naor and Shamir [72]. Visual cryptography scheme (VCS) is a secret sharing scheme [94] in which a secret is encrypted to patterns (referred to as *shares*), shared, and correctly decrypted by participants. Encryption and decryption are performed by a computer. In contrast, in some VCSs, the human visual system is used to decrypt the encrypted secret. Typically, such schemes require complex computations. In other words, with such schemes, the human eye can decrypt secrets easily where decryption would be difficult for a computer. In typical secret sharing schemes, the secret is either numbers or text, and in VCSs, the secret is an image.

In the traditional VCS, the inputs are binary images. However, methods that use grayscale and color images as input have also been developed [40]. In addition, more sophisticated approaches [45, 58, 59] that improve the visual quality of shares have been proposed. Figure 2.12 shows an example of the extended VCS (EVCS), where a secret image is encrypted to meaningful, structured shares rather than noisy, unstructured shares in the traditional VCS. In addition, the traditional VCS, which encrypts only a single image, has been extended to handle multiple secrets using two circle shares and different rotation angles [97]. *Universal Share* can decrypt multiple images using a unique share [32, 64]. Among the various approaches, methods that employ *Universal Share* [32, 64] are closely related to the proposed approach, i.e., we introduce a *common share* in our work, which is a type of *Universal Share*. A comprehensive review of VCSs can be found in the literature [60].

FIGURE 2.12 Extended VCS proposed by [59]. Two images shown on the left are the shares and the right image is the secret image. While shares are noisy, unstructured patterns in the traditional VCS, the extended VCS generates structured, meaningful patterns, i.e., images for shares.

Such previous methods have limitations, i.e., they generate meaningless shares and are based on Boolean operations, such as XOR and bit shift operations, making it difficult to decrypt secrets physically. In contrast, the proposed method can encrypt multiple secrets in meaningful physically realizable shares, because it is based on the physical superimposition of shares printed on transparencies, which corresponds to an *AND* operation.

2.4.2 Hiding multiple images

Recently, various approaches have been proposed to hide visual information in 2D images or 3D objects. Mitra and Pauly proposed *Shadow Art*, which casts multiple images of a sculpture onto walls [70]. Baran et al. proposed layered attenuators that cast different colored shadow images under specific lighting conditions [5] (Figure 2.13). Alexa and Matusik proposed a method to create relief surfaces whose diffuse reflection approximates the given images under known directional illumination [1]. ShadowPix is a surface that displays multiple images using self-shadowing [7]. In *emerging images* [69] and *camouflage images* [21], one or more figures are embedded into a busy apparent background and remain imperceptible. Papas et al. proposed *Magic Lens*, a passive display device that exposes hidden messages and images from seemingly random and structured source images [82] (Figure 2.14). Other interesting approaches to hide images have also been proposed, e.g., hiding patterns on a metallic substrate [84] and on a level line moiré [20].

In most approaches, fabrication costs are significant because a high-resolution 3D printer or milling machine is required. In contrast, images produced by the proposed *Magic Sheet* method can be printed on inexpensive transparencies using consumer-grade printers.

FIGURE 2.13 Layered attenuators cast different colored shadow images under specific lighting conditions [5].

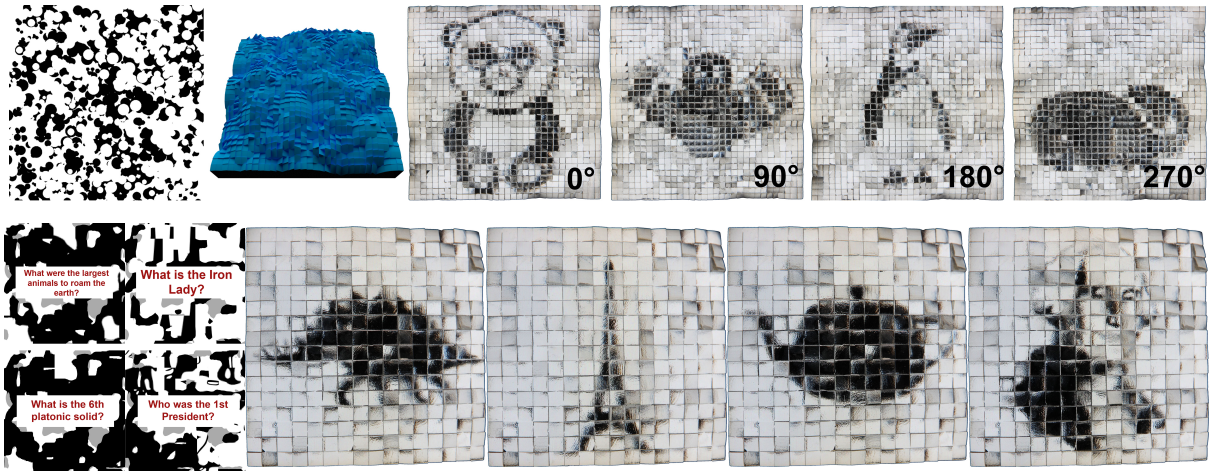


FIGURE 2.14 The magic lens proposed by Papas et al. [82] is a passive display device that exposes hidden images from seemingly random and structured source images. Top: the 1st column shows the source image. The 2nd column shows the magic lens. The 3rd-6th columns show the hidden images viewed from specific orientations. Bottom: four source images with the question are warped to reveal pictographic answers with the magic lens.

CHAPTER 3

Discrete Element Textures

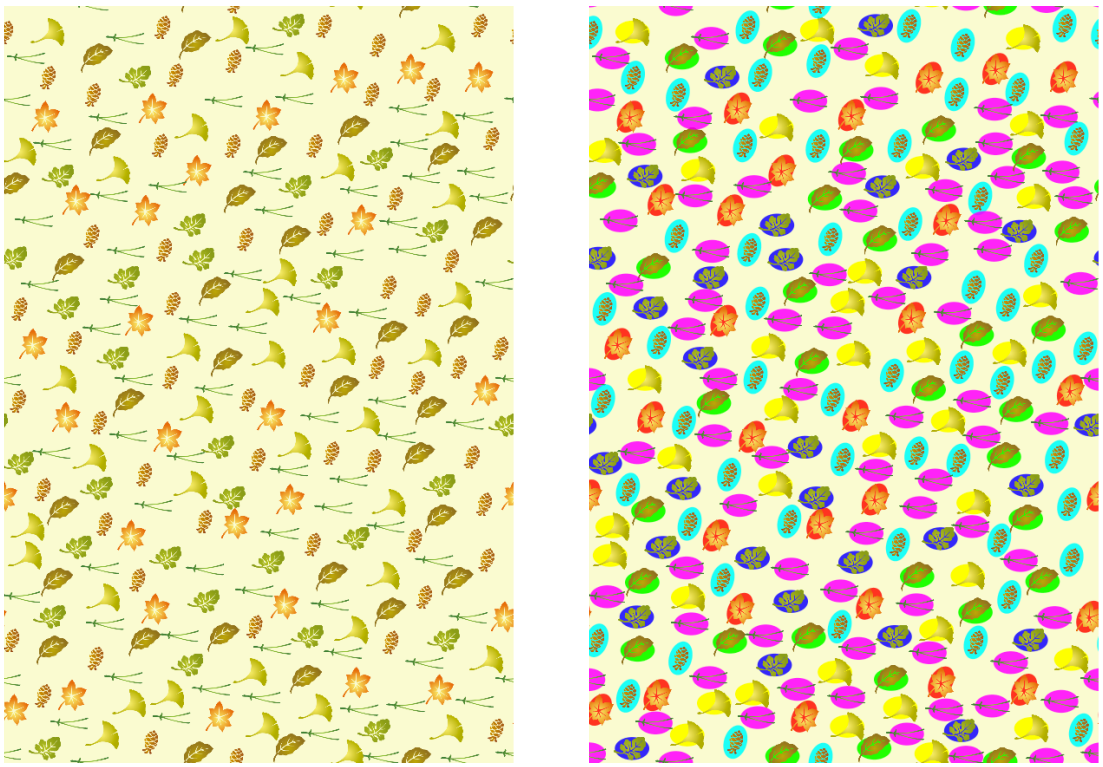
In this chapter, we present an element placement method for generating patterns containing "discrete elements". By extending various blue noise sampling methods, we propose a visually uniform distribution of multi-class elements. Our method also supports tileable aperiodic distribution. Instead of actual elements, for fast calculation, we use a circular or elliptic disk as a proxy of an element when checking conflicts with nearby elements during the distribution process. The nature of our results is comparable to swatches in books, which shows that our method is capable of generating visually appealing swatches for a set of elements. The user study showed that our method outperformed state-of-the-art discrete element texture synthesis approaches in terms of pattern visual quality.

3.1 Introduction

We can see a vast variety of patterns composed of design elements on books, websites, artwork, building facades, wallpaper in a room, and so on. When a pattern is tiled to fill a target domain, it should be possible to seamlessly place multiple occurrences of the pattern side-by-side. When selecting the pattern for such applications, we can look at pattern swatches listed in books or on websites, or actual products in real shops. We can also create our original patterns from design elements. Therefore, it is important for pattern designers or users to create a visually appealing swatch or a desirable original pattern.

Recently, several discrete element distribution methods have been proposed in computer graphics. Discrete elements mean visually distinguishable objects. The term *discrete element textures* (or *geometric textures*) is used when a texture is composed of discrete elements. Unlike pixel-based texture synthesis [104], discrete texture synthesis synthesizes a larger texture using a vectorized approach involving exemplars (which correspond to the aforementioned swatches). However, because this method needs exemplar patterns, we still have an issue on how to create visually appealing or desirable exemplar patterns.

To tackle this issue, we propose a procedural distribution approach. While previous approaches lack the capability for dealing with anisotropy and/or multi-class properties (multi-attributes), our method can handle them appropriately and generate a variety of patterns composed of discrete elements. To handle these properties, we propose an algorithm called *multi-class anisotropic blue noise sampling*. In addition, a user can control the element arrangement



(a) Generated pattern (b) Visualized proxy shapes

FIGURE 3.1 Result of our element distribution with six classes (six different elements).

or orientations by specifying the element flow (Figure 3.5). We provide three types of control: random orientation (Figure 3.5a), constrained random orientation (Figure 3.5b), or a user-controlled flow field using a stroke interface (Figures 3.5c-3.5d). Using our method, a user can easily create a pattern composed of anisotropic and multi-class elements, as shown in Figure 3.1, Figure 3.6, Figure 3.7, and so on. Our method efficiently computes element arrangements by approximating an element with a proxy shape. A user can adjust a proxy size, which results in some overlaps of each element (if a smaller size is specified than the element size), as well as a sparse or dense distribution. We performed a user study to compare our approach with state-of-the-art discrete element texture synthesis approaches and found that our patterns significantly outperformed them with respect to visual qualities and reduced computation time.

In summary, our contributions are as follows:

- We integrate the multi-class and the anisotropic blue noise sampling method to a more generalized method and employ it for discrete element distributions.
- We generate tileable textures and control an element arrangement in which they face each other along an underlying flow field with our method and state-of-the-art discrete element texture synthesis approaches.

3.2 Algorithm Overview

Before we describe our algorithm in detail, a brief description follows. Our algorithm is shown in Algorithm 1. Our goal is to implement a function Element-Distribution(·), which takes a user-specified proxy shape size (i.e., a user specify major and minor radius of an proxy

ellipse), per-class values for $\mathbf{r}$-matrix described later, a flow field, and a domain. It then returns a distributed sample set $\mathcal{S}$. The following subsections describe the backbone of our algorithm (3.2.1) and how to accommodate the tiling case (3.2.2). The details are described in section 3.3.

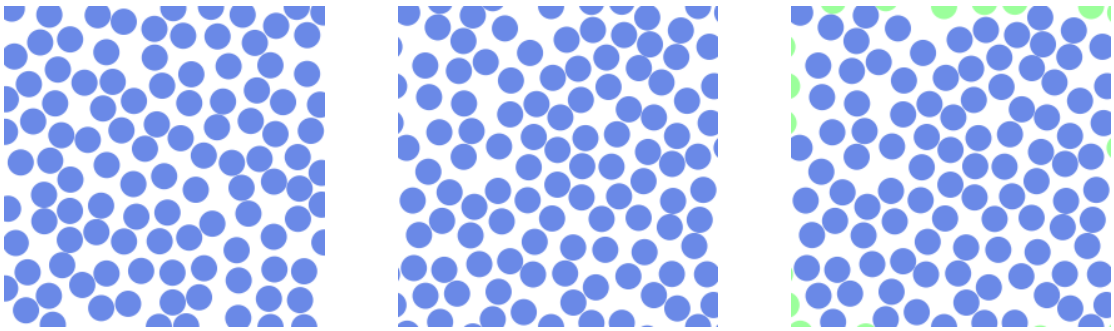
3.2.1 Fast Poisson Disk Sampling

We use a background grid $\mathcal{G}$ for storing samples and accelerating spatial searches [10]. In our configuration, we calculate the cell size to be bounded by $r_{min}/\sqrt{n}$, where r_{min} is the minimum radius of a proxy ellipse and n is the dimension of the sample domain $\mathbb{R}^n$ ($n = 2$). We first select the initial sample s_0 randomly chosen uniformly from the domain. Insert it into $\mathcal{G}$ and initialize ActiveList (an array of sample indices). When the active list is not empty, choose a random index i from it and generate up to k points chosen uniformly from the sample annulus. Here we use $k = 30$.

Unlike the original method, our method takes the ellipse shape into account (we describe the algorithm later). For each point, we calculate IsConflict$(\cdot)$ with existing samples. For fast calculation, testing only the nearby samples stored in $\mathcal{G}$ works well. If there is no conflict, accept it as a sample and add its index to the active list. If after k attempts, there are no such

Algorithm 1 ElementDistribution($(a, b), \{r_i\}$, vf, Ω)

```
// major-/minor- radius of an ellipse; (a, b),
// user specified per-class values; {r_i},
// flow field; vf,
// sampling domain; Ω ⊂ ℝ²
BuildRMatrix({r_i})  // see Supplemental material
generate background grid 𝒢 in Ω
s_0 ← choose a point uniformly at random ∈ Ω
𝒮 ← s_0  // 𝒮 is a set of sampling points
ActiveList.append(0)  // index of s_0
while ¬ActiveList.empty() do
    i ← index uniformly at random ∈ [0,ActiveList.size)
    p_i ← i-th sample ∈ 𝒮
    for attempt ← 0 to k do
        t ← GenerateTrial(p_i)  // see Algorithm 2
        j ← t's index
        if ¬IsConflict(t) then
            𝒮 ← t  // accept the trial
            𝒢_i ← j  // store the index to the grid cell
            ActiveList.append(j)
            break
        end if
        if k < attempt then
            ActiveList.remove(i)
        end if
        attempt++
    end for
end while
return 𝒮
```



(a) Without tiling (b) With tiling (c) Visualized

FIGURE 3.2 Tileable distribution.

points to be found, remove i from ActiveList.

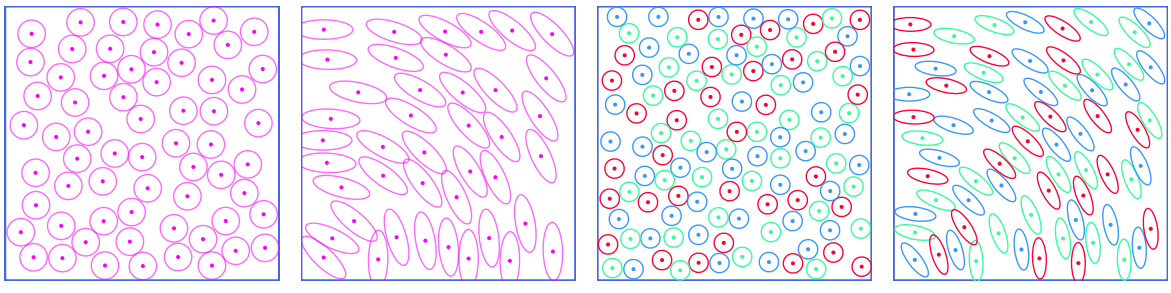
3.2.2 Tileable Distribution

For generating a tileable texture, we should consider the boundary cases for conflict check. For example, when the bounding box of an element exceeds the boundaries of the sampling domain, it can provide a potential conflict with an element at the other side of the boundary. To address this problem, we check such potential conflicts in our conflict check procedure by double checking conflicts in the current domain and the virtually warped domain. Figure 3.2 shows the result. While Figure 3.2a does not consider tiling, Figure 3.2b does consider it, and Figure 3.2c visualizes such a tiling effect. The green samples are identical to the elements at the other side, showing the texture to be seamlessly tileable.

3.3 Multi-class Anisotropic Blue Noise Sampling

We propose a sampling method called Multi-class Anisotropic Blue Noise Sampling (Figure 3.3d), which extends existing methods, anisotropic blue noise sampling [54] (Figure 3.3b) and multi-class blue noise sampling [103] (Figure 3.3c). In addition to explaining how we build such a sampling algorithm, we consider the following matters for incorporating the method with an element distribution:

- How to deal multi-class attributes (3.3.1)



(a) Isotropic (b) Anisotropic (c) Multi-class (d) Ours (b + c)

FIGURE 3.3 Proxy disk types. The same color represents the same class. (a) and (b) have a single class, while (c) and (d) have three classes.

- How to measure our anisotropy (3.3.2)
- How we can generate a sample candidate (3.3.3)
- How to calculate IsConflict(·) (3.3.4)

We describe these matters in the following subsections. We also describe in 3.3.5 how to control the elements' directions shown in Figure 3.5.

3.3.1 r-matrix Construction

In dart throwing methods, two samples are at least distance r away from each other, where r is a user-specified value that corresponds to an exclusive disk radius. In a multi-class case, the **r**-matrix is used as an inter-class distance instead.

The **r**-matrix is built from user-specified values for each class, $\{r_i\}_{i=0:c-1}$, where c is the number of the class. In the **r**-matrix, $\{r_i\}$ is used for diagonal entries. Off-diagonal entries are computed by the BuildRMatrix ($\{r_i\}_{i=0:c-1}$) function [103]. Here off-diagonal entries are used for inter-class distance, i.e., if two samples are in different classes, a user cannot explicitly specify inter-class distance.

For example, let us consider $c = 3$ with $r_0 = 40.0$, $r_1 = 20.0$, and $r_2 = 10.0$. In this configuration off-diagonal entries are computed and the resulting **r**-matrix is given by

$$\mathbf{r} = \begin{bmatrix} 40.0 & 17.9 & 8.7 \\ 17.9 & 20.0 & 8.7 \\ 8.7 & 8.7 & 10.0 \end{bmatrix}. \tag{3.1}$$

We can see that off-diagonal entries, i.e., inter-class distances are no longer explicitly specified. For a multi-class element distribution, a user should be able to explicitly specify inter-class distance. Hence, we modify the **r**-matrix because it acts as a scaling factor $\mathbf{r}^{\text{scale}}$ when we check inter-class conflict. $\mathbf{r}^{\text{scale}}$ is computed by element-wise division of the $\min(\mathbf{r})$ entry, where $\min(\mathbf{r})$ is the minimum value of **r**. In the above case, $\min(\mathbf{r}) = \mathbf{r}(0,2) = \mathbf{r}(2,0) = \mathbf{r}(1,2) = \mathbf{r}(2,1)$ and thus $\mathbf{r}^{\text{scale}}$ is given by

$$\mathbf{r}^{\text{scale}} = \begin{bmatrix} 4.6 & 2.0 & 1.0 \\ 2.0 & 2.3 & 1.0 \\ 1.0 & 1.0 & 1.2 \end{bmatrix}. \tag{3.2}$$

We use $\mathbf{r}^{\text{scale}}$ for our IsConflict(·).

3.3.2 Anisotropic Sampling

In anisotropic blue noise sampling in [54], the authors employed the following distance metric to approximate the geodesic distance from sample $\mathbf{p}$ to $\mathbf{q}$ [49].

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(J(\mathbf{p})(\mathbf{q}-\mathbf{p}))^T(J(\mathbf{p})(\mathbf{q}-\mathbf{p}))}, \tag{3.3}$$

Algorithm 2 GenerateTrial($\mathbf{p}$)

// sample; $\mathbf{p}$
$\mathbf{t}_c \leftarrow \arg\min_i N_i$ // see Equation 3.8
r $\leftarrow$ computed by Equation 3.9
loop
    $\mathbf{t} \leftarrow$ generate random point $\in [-2r, 2r]$ from $\mathbf{p}$
    warp $\mathbf{t}$'s coordinate if it's out of the domain Ω
    $d_1 \leftarrow d(\mathbf{p}, \mathbf{t}),\ d_2 \leftarrow d(\mathbf{t}, \mathbf{p})$
    if $\neg(d_1 < 1$ **or** $d_2 < 1)$ **and** $(d_1 < 1.5$ **or** $d_2 < 1.5)$ **then**
        break
    end if
end loop
return $\mathbf{t}$

where $J(\cdot)$ is a local Jacobian matrix applied to the domain. Since the distance metric is not symmetric, i.e., $d(\mathbf{p}, \mathbf{q}) \neq d(\mathbf{q}, \mathbf{p})$, a conflict check is calculated using the following formula:

$$d(\mathbf{p}, \mathbf{q}) < 1 \textbf{ or } d(\mathbf{q}, \mathbf{p}) < 1. \tag{3.4}$$

To cope with our multi-class elements and their anisotropy, we compute the distance from sample $\mathbf{p}$ to $\mathbf{q}$ as follows:

$$d(\mathbf{p}, \mathbf{q}) = \left\| \begin{bmatrix} 1/a' & 0 \\ 0 & 1/b' \end{bmatrix} \mathbf{R}(\mathbf{p}_\theta) \begin{bmatrix} \mathbf{p}_x - \mathbf{q}_x \\ \mathbf{p}_y - \mathbf{q}_y \end{bmatrix} \right\|, \tag{3.5}$$

where $\mathbf{R}(\theta)$ is a rotation matrix of angle θ, $\mathbf{p}_\theta$ is the sample direction of $\mathbf{p}$ picked from an underlying vector field value, and

$$a' = \frac{\mathbf{r}^{\text{scale}}(\mathbf{p}_\text{c}, \mathbf{q}_\text{c}) \cdot \mathbf{p}_a + \mathbf{r}^{\text{scale}}(\mathbf{q}_\text{c}, \mathbf{p}_\text{c}) \cdot \mathbf{q}_a}{2}, \tag{3.6}$$

$$b' = \frac{\mathbf{r}^{\text{scale}}(\mathbf{p}_\text{c}, \mathbf{q}_\text{c}) \cdot \mathbf{p}_b + \mathbf{r}^{\text{scale}}(\mathbf{q}_\text{c}, \mathbf{p}_\text{c}) \cdot \mathbf{q}_b}{2}, \tag{3.7}$$

where $\mathbf{p}_\text{c}$ is $\mathbf{p}$'s class and $\mathbf{q}_\text{c}$ is $\mathbf{q}$'s class. Due to its non-symmetric distance measure, it is just an approximation. In cases where two samples have the same size and the underlying flow field is smooth enough, the distance metric works well; otherwise, two samples can conflict. We can see some conflict in Figures 3.5a and 3.5b due to the non-smooth flow field. In contrast, in Figure 3.5d, there are no conflicts owing to its smooth vector field.

3.3.3 Trial Generation

Our sample candidate (a *trial*) generation algorithm is shown in Algorithm 2. During element distributions, we consider how to generate a trial. Since we have multi-class elements, we also have to consider which class element should be chosen. To this end, we compute a *fill rate*

of existing classes [103]. The fill rate of i-th class N_i is computed by

$$N_i = N \frac{1/r_i^n}{\sum_{j=0}^{c-1} 1/r_j^n}, \tag{3.8}$$

where N is the total number of target samples, n is the sample space dimension, and $\{r_i\}$ is the specified per-class minimum distance. We choose a class with a minimum fill rate. Until a trial is chosen, we loop trial generation steps.

First a random point $\mathbf{t}$ is generated from $-2r$ to $2r$ distance from a given sample $\mathbf{p}$, where r is calculated by the inter-class distance between $\mathbf{p}$'s class $\mathbf{p}_c$ and the trial's class $\mathbf{t}_c$ multiplied by $\mathbf{p}$'s major-radius $\mathbf{p}_a$:

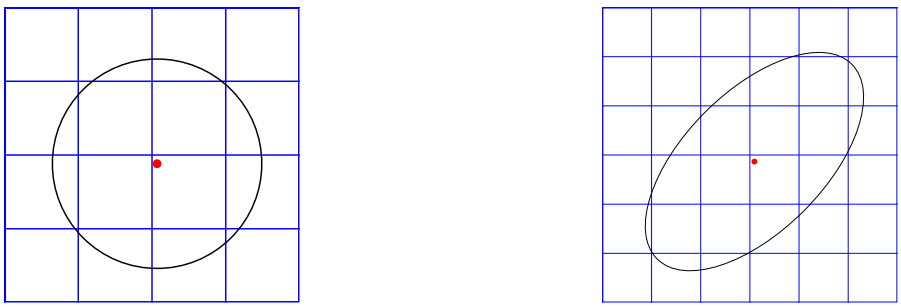
$$\mathbf{r}^{\text{scale}}(\mathbf{p}_c, \mathbf{t}_c) \cdot \mathbf{p}_a. \tag{3.9}$$

Then we check whether a trial's coordinate exceeds the domain. If it exceeds the domain, we can warp the trial in a toroidal manner or simply reject it and go to the next loop. If a trial is within the domain, we check for conflict with $\mathbf{p}$. To get a dense sampling result, we restrict the trial's position such that it is not far from a given sample $\mathbf{p}$ and is determined by the predicate listed at line 8 of Algorithm 2. If the predicate holds true we get out of the loop and return the trial $\mathbf{t}$; otherwise, we go to the next loop.

3.3.4 Conflict Check

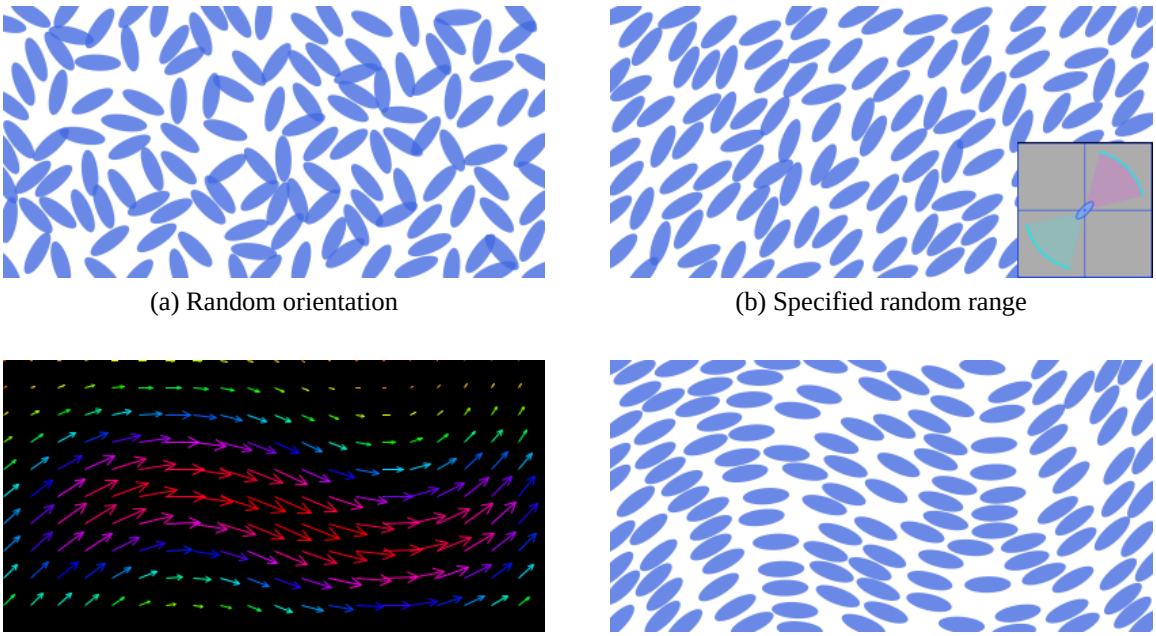
To check conflict between a sample candidate (trial) and already accepted samples, we only check nearby existing samples based on a grid acceleration approach [10].

In the case of a circular disk, we restrict the search region to the grid cells that overlap with the bounding box of the disk (Figure 3.4a). In a similar fashion, in the case of an elliptic disk, it is ideal to conflict-check the grid cells that overlap with the bounding box of the disk. However, taking into account the computational cost for identifying these cells, it is reasonable and enough to conflict-check the grid cells within the axis-aligned bounding box of the elliptic disk (Figure 3.4b). The grid cells shown in Figure 3.4 are to be checked in the IsConflict(·) function. As we can see, all the cells in Figure 3.4a are necessary and sufficient for conflict-check, while in Figure 3.4b, there are cells that obviously conflict-free with the sample of focus



(a) Circular disk (b) Elliptic disk

FIGURE 3.4 Conflict checks of the shown sample (indicated by a red point) and the nearby samples. The shown grid cells are checked in the IsConflict(·) function.



(a) Random orientation (b) Specified random range

(c) Visualized vector field (d) With vector field

FIGURE 3.5 Controlling element orientations along with an underlying flow field. In (b), a user can restrict orientations of elements by specifying a range of random orientations as shown in the inset (the arc and the fan indicate that random orientations are generated within the angle).

as we check all the cells within the axis-aligned bounding box of the disk. Given a trial $\mathbf{t}$, the IsConflict($\mathbf{t}$) returns true or false based on equation 3.4.

3.3.5 Control Element Orietation

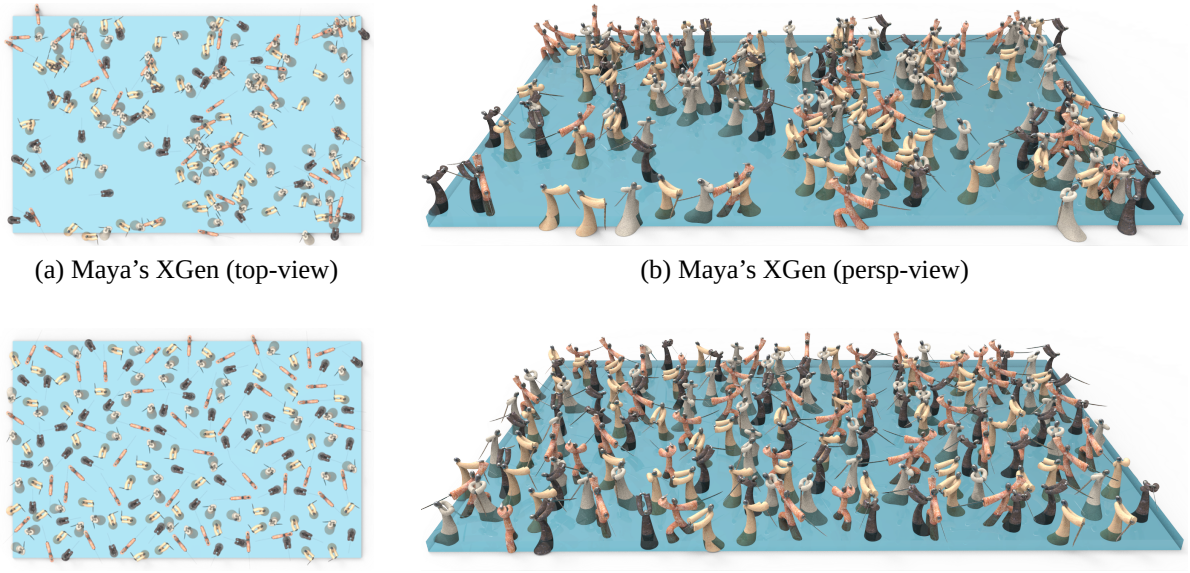
Given a user-stroke, we compute the background flow field along which distributed elements are directed. Our flow field is designed as a vector [110]. Similar to the brush interface of [17, 47], we divide a user stroke into strips composed of vertices $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_i, \ldots\} \in \mathcal{V}$ and call them design elements of the vector field. A vector field at a specific point $V(\mathbf{p})$ is computed using the following equation:

$$V(\mathbf{p}) = \sum_i e^{-\kappa|\mathbf{p}-\mathbf{v_i}|^2} V_i(\mathbf{p}), \tag{3.10}$$

where κ is a decay constant, $\mathbf{p}$ is a point in the domain Ω, $V_i(\cdot)$ is the basis vector field corresponding to a design element, and $\mathbf{v}_i$ is the position of the design element. We compute the vector field on the entire domain Ω (Figures 3.5c and 3.5d), and when a trial point is generated, its vector direction is picked and assigned to it. Because of its simple summation calculation, a user can edit the flow field multiple times to get a desirable flow field.

3.4 Results

We can generate a variety of patterns with our application. In Figure 3.1 we use six different elements (= 6-class) in the configuration. We set proxy shapes for each element and adjust the size of the shape. In this configuration, we do not incorporate an underlying flow field, which



(a) Maya's XGen (top-view)

(b) Maya's XGen (persp-view)

(c) Ours (top-view)

(d) Ours (persp-view)

FIGURE 3.6 Rendering results of Figurines (4-class). Same number of objects (=total 176) are distributed inside the blue stage.

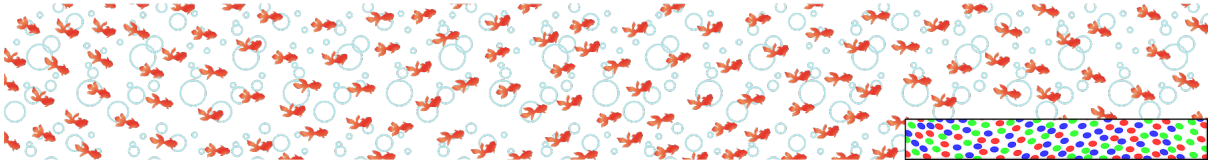
results in elements in the same class facing the same orientation. In Figure 3.6, we compared the 3D object distributed scene generated by Maya's XGen and our distribution pattern. We used four figurines (= 4-class) for the distribution. Compared to the scene generated by XGen's random distribution, ours have fewer object collisions and a visual uniformity, provides a more visually appealing impression. Other various results are shown in Figure 3.7. The sampling quality of our method is analyzed and a further comparison to Refs. [54] and [103] is available in Section A.

3.4.1 Reproducing Reference Swatch

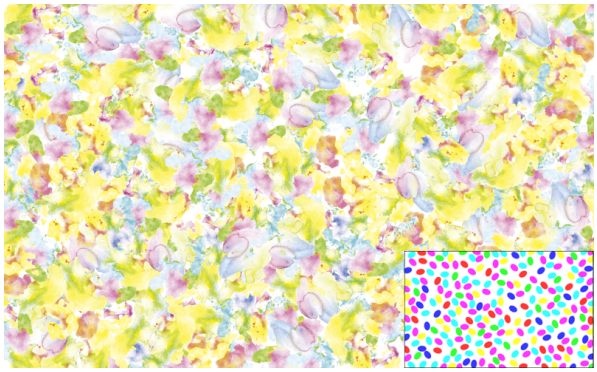
In Figure 3.8, we compare how our method can reproduce a swatch in a reference book [9]. In this case, we use nine elements (i.e. 9-class) listed in Figure 3.8a and set each proxy shape to be smaller than an element image. This allows some overlap when reproducing the swatch (Figure 3.8d). Compared to the pseudo random placement (Figure 3.8b), our approach captures the reference distribution properties well, and thus a visually plausible pattern is reproduced (Figure 3.8c).

3.4.2 Tileable Texture Generation

In Figure 3.9, we generate a tileable texture and compare it with one that avoids boundaries. In Figure 3.9a, we can see visual artifacts such as vertical or horizontal lines with no elements across them. In contrast, the tileable image generated by our method has enhanced seamless boundaries (Figure 3.9b). In Figure 3.9 we simply tile a texture in a 2×2 matrix for demonstrating the effect of a tileable distribution. We can still perceive repetitive elements placed side-by-side on it. To avoid this, we can employ Wang tiles with pre-distributing elements similar to that described in [50].



(a) Fish (3-class)



(b) Splash (6-class)

(c) Marché (7-class)

(d) Halloween (8-class)

(e) Felt-like (9-class)

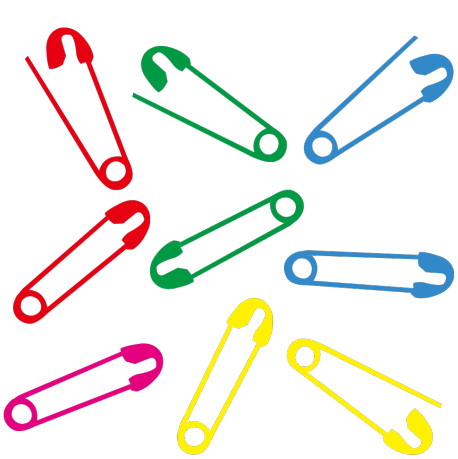
FIGURE 3.7 Additional resutls.

3.4.3 Control Inter-class Ratios

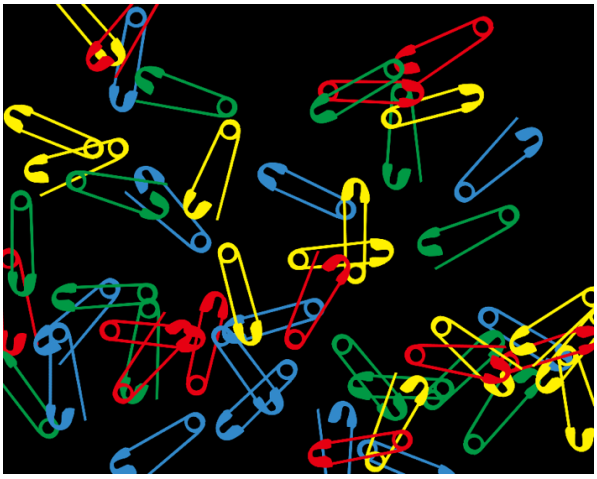
We can reproduce a swatch using our method (Figure 3.10). Since our method can specify ratios of multi-class elements, we can generate various images similar to the swatch in a reference book but which differ in the extent of inter-class ratios. In Figure 3.10a, two classes of elements are evenly distributed in the image. While each single class element is placed spatially in a uniform manner, the union of two classes retains spatial uniformity owing to its multi-class blue noise properties. In Figure 3.10b, the two classes of elements are distributed by a ratio of $2:1$, which nearly equals that of the reference swatch shown in Figure 3.10c.

3.4.4 Control Element Orientation

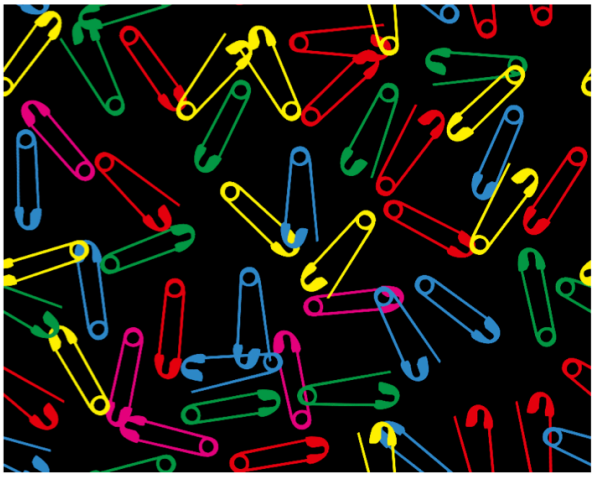
Figure 3.11 demonstrates the effect of flow-guided pattern generation. 3-class fish elements are distributed along with an underlying flow field. Figure 3.12 demonstrates an oriented distribution result. In the swatch shown in Figure 3.12c, the elements are randomly oriented, while ours shown in Figure 3.12b have a similar orientation by restricting the orientation to a specific range (Figure 3.5b). In cases when the input elements have semantics such as an expectation



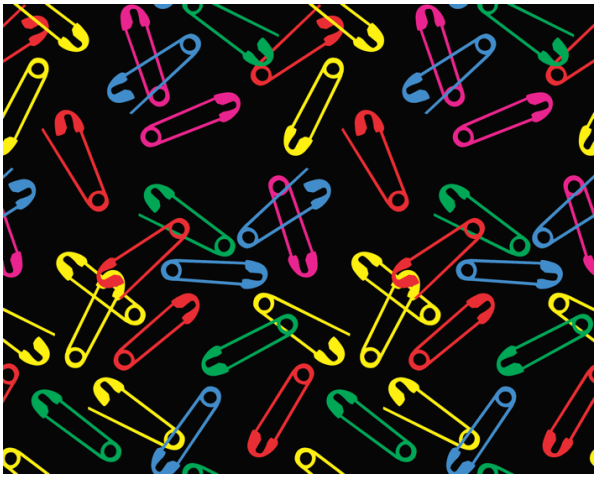
(a) Elements (9-clip)



(b) Pseudo-random distribution

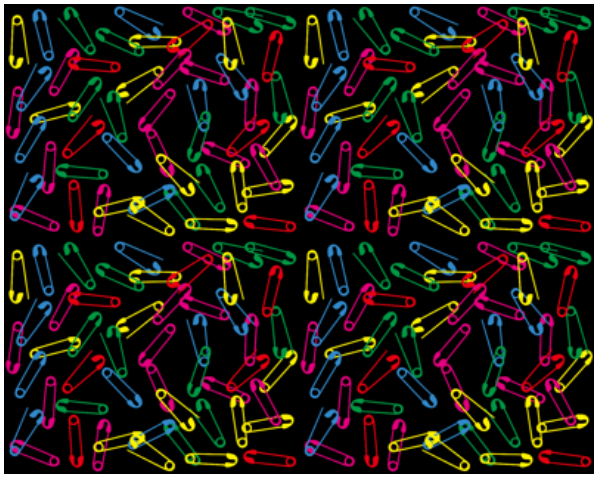


(c) Our distribution

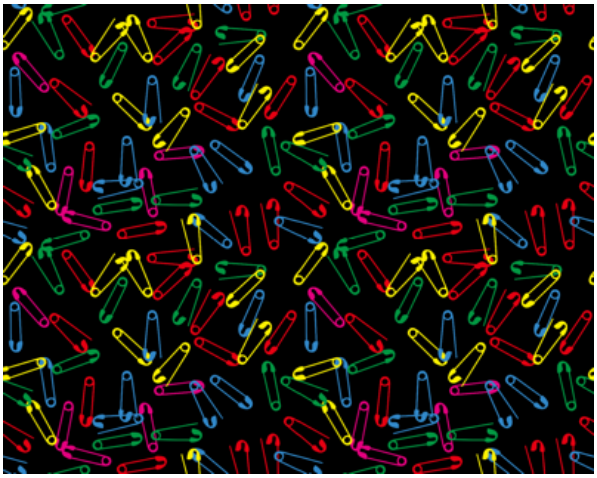


(d) Swatch in a book

FIGURE 3.8 Comparison of our pattern with a pseudo-random pattern and a swatch in a book.



(a) Avoid boundaries



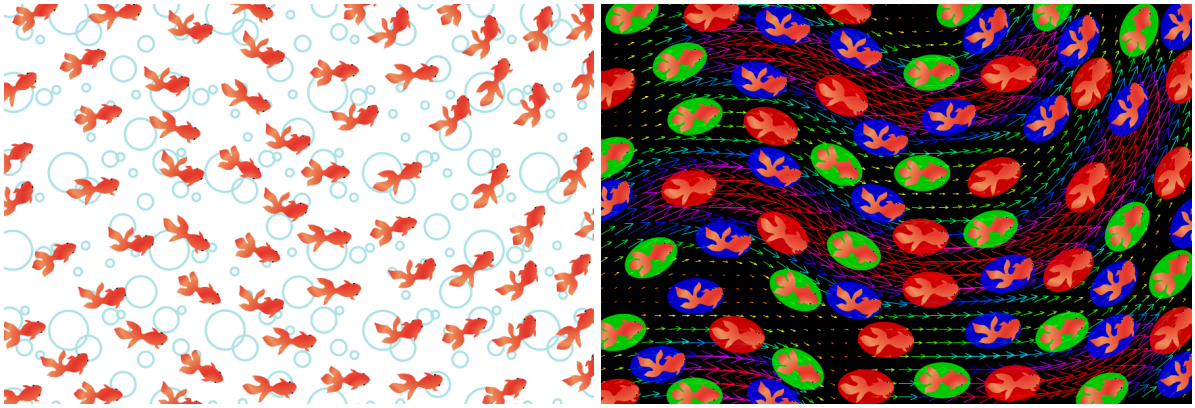
(b) Tileable

FIGURE 3.9 Effect of tileable generation. A texture is tiled 2×2 side-by-side.

that fishes swim or that horses run in the same direction, our method is capable of generating an expected arrangement.

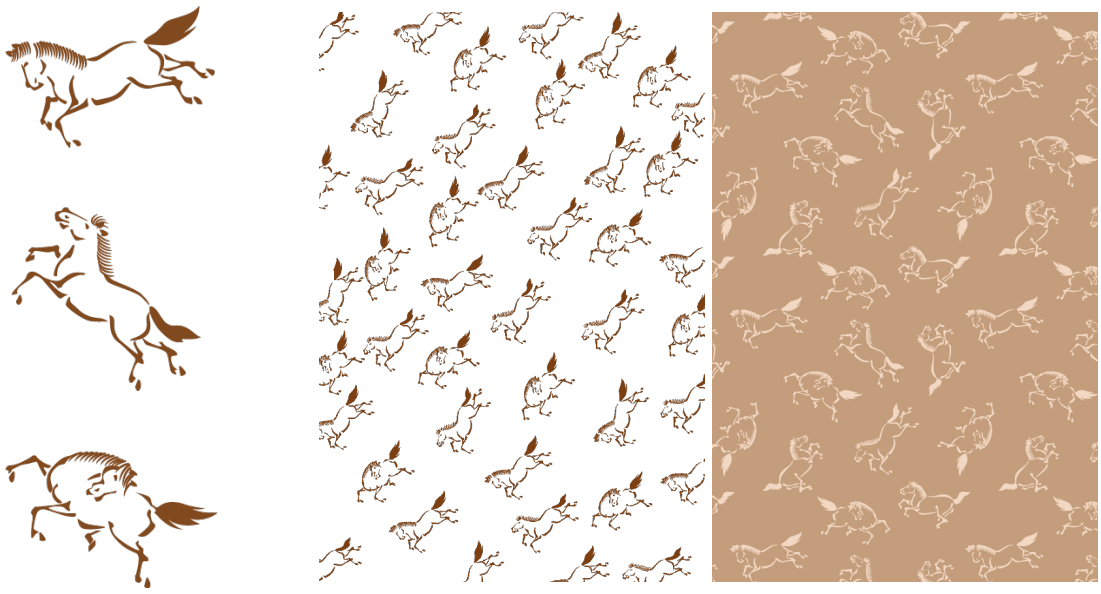
(a) 50%:50% (b) 68%:32% (c) Swatch in [9]

FIGURE 3.10 Effect of changing *fill rate* of two elements.



(a) Generated pattern (b) Underlying flow field

FIGURE 3.11 Effect of flow field editing. 3-class elements approximated by ellipses are distributed along with a flow field.



(a) Elements (b) Oriented (c) Swatch in [9]

FIGURE 3.12 Effect of specifying element orientations.

3.5 User Study

We evaluated our approach by comparing with the state-of-the-art methods with respect to the generation of visually appealing patterns. We invited 29 users to participate in the following user study. All participants were graduate students, two of whom were majoring in design. Since previous discrete element placement approaches lack the ability to distribute multi-class and/or anisotropic elements, we compare our approach with the following discrete element texture synthesis approaches: BBT06 [6], HLT09 [41], IMIM08 [42], LGH13 [52], and MWT11 [62]. All patterns shown to the participants in our study are available in Section A.4.

All synthesis results of these methods as well as the elements used for our user study were taken from those listed in [52]. Figure 3.13 shows our patterns for each element used in our user study. For each element, we adjusted the proxy size and the element size so that our result had almost the same size and number of elements in the domain. Since our approach only checks whether a sampling point is inside the domain, the result does not ensure that a whole element is inside the domain this leads to a *clipped* pattern. Therefore we also generated a pattern where whole elements had to be inside the domain and compared our *clipped* and *without clipped* patterns with previous methods. We asked the users to rate the patterns between 0 (worse) and 5 (best), indicating visual quality. The seven patterns (the previous five methods and our *clipped* and *without clipping* patterns) for each element (=total 42 patterns) were randomly displayed to the users. The first set of patterns was displayed to the participants in a random order, and the following sets were displayed in the same manner. We invited the participants to our controlled room so that all participants experienced the same conditions. It took approximately $3-5$ min per person to rate all patterns.

Figure 3.14 plot the average score along with the associated 95% confidence interval for each case. We performed an analysis of variance and found that there are significant differences between the methods. Following the result, we then performed a Scheffe's test for multiple comparisons ($\alpha = 0.05$), and the results are listed in Table 3.1. It showed that our *clipped* patterns outperformed the other methods, followed by our *without clipping* for any kind of element.

In addition, our method can generate a pattern faster than LGH13 [52], which takes several seconds or minutes to generate a pattern. Our computation time for generating Figure 3.13 on a 3.4GHz Core i7, 16GB RAM machine is given in Table 3.2.

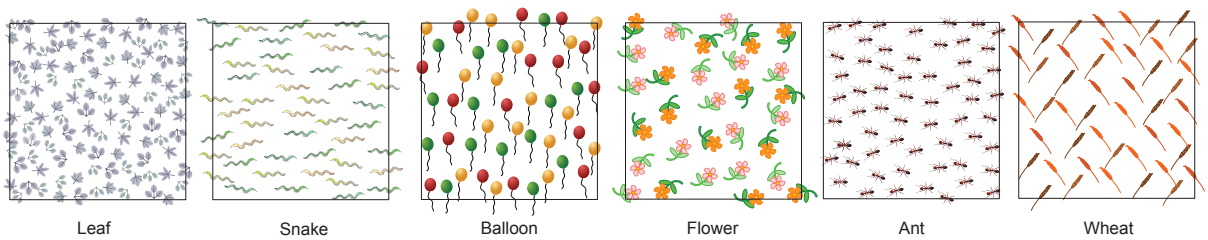


FIGURE 3.13 Our discrete element patterns used for our user study.

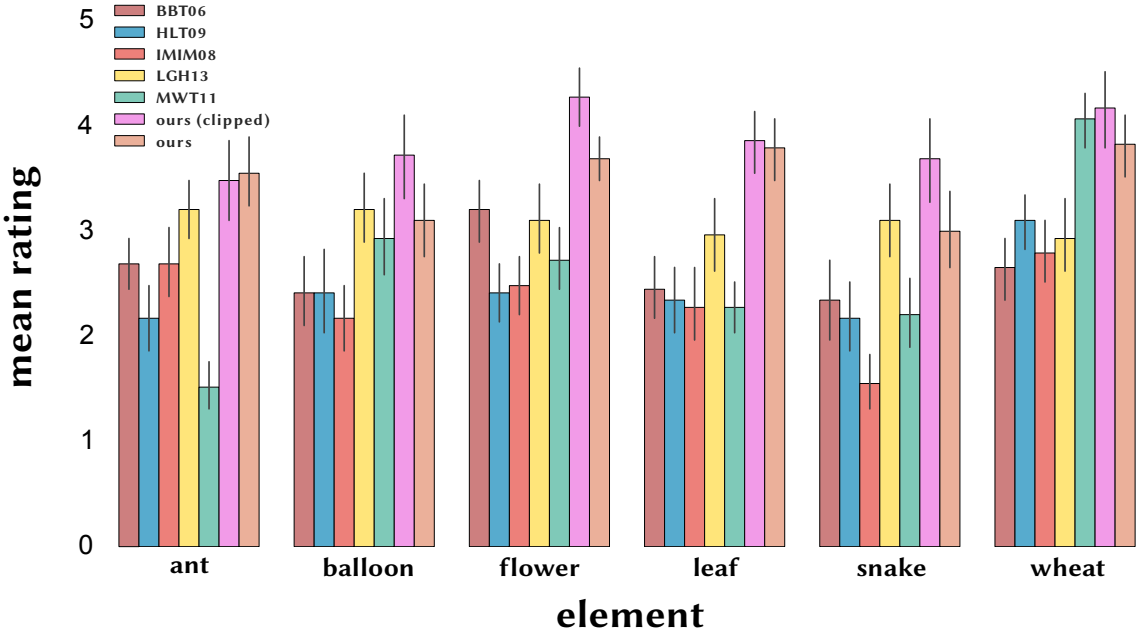


FIGURE 3.14 Comparison results for each element.

TABLE 3.1 Scheffe's test (corresponding to Figure 3.14). Means designated with the same letter (group) are not significantly different.

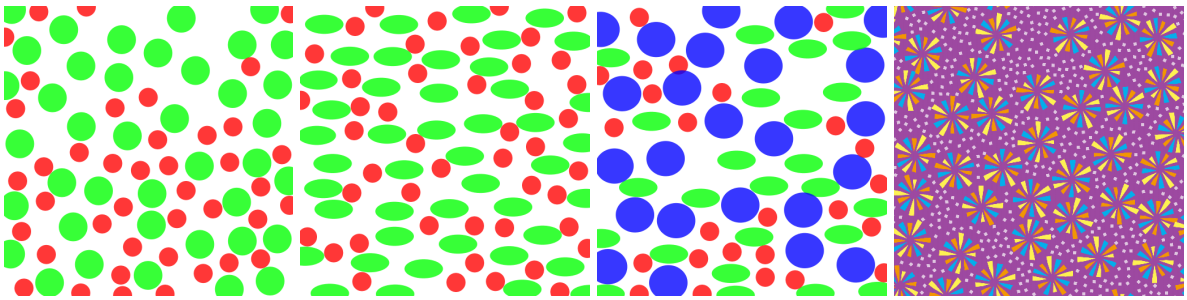
group	method	mean
a	ours (clipped)	3.868
b	ours	3.494
c	LGH13	3.086
d	BBT06	2.626
d	MWT11	2.621
d	HLT09	2.437
d	IMIM08	2.328

TABLE 3.2 Computation time for the distributions shown in Figure 3.13.

element	#elements	time
leaf	103	41ms
snake	47	46ms
balloon	48	9ms
flower	30	16ms
ant	59	60ms
wheat	46	12ms

3.6 Conclusions and Discussion

We have proposed an algorithm called Multi-class Anisotropic Blue Noise Sampling and employed it for the distribution of elements. A user can generate discrete element patterns using



(a) (b) (c) (d)

FIGURE 3.15 Limitations. Two different sized elements (size ratio = 1:2) (a), or elements with different shaped proxies (b) result in a non-uniform distribution (in these cases, the red circle class does not distribute spatially in a uniform manner). Distributions of three or more classes suffer as well (c). Note that a multi-sized element distribution can be achieved by a simple hierarchical dart throwing approach (d); therefore, we do not focus on such a pattern in this work.

our application interface. The user can produce tileable patterns, and modify their arrangements via a stroke interface or manual editing. In addition, the user can specify the ratios of each element to be placed. In our application, the method is applied to each proxy shape of the element. An element is approximated by a circle or an ellipse to efficiently compute a spatially uniform distribution. A user study showed that our generated patterns significantly outperform patterns generated by previous discrete element texture synthesis approaches in terms of visual quality.

There are several limitations and future directions. Although our method can quickly produce visually plausible results, and if required, a user can modify the distributed elements arrangement, our proxy shape is restricted to a circle or an ellipse, which does not accurately approximate a non-convex element. Therefore, we cannot produce a high-density distribution well with those elements. In cases that involve dense packing of non-convex elements, appropriate shapes must be used for their proxies. In the case of a sparse distribution or where elements are far from each other to some extent, our shape proxy works well.

Moreover, in our method, we cannot handle conflict check of proxies that have different sizes and/or shapes from each other (Figure 3.15). In such cases, the distances tend to be more asymmetric, and our distance metric cannot capture their characteristics well. Note that a multi-sized element distribution can easily be achieved by a simple hierarchical dart throwing approach (Figure 3.15d), is a different concept from that proposed in this work. Therefore, for a multi-sized element distribution, a simple hierarchical dart throwing approach can be used instead of our multi-class anisotropic approach. In Figure 3.15d two largely different sized elements are distributed by descending order of its size. In this study, we assume similar proxy shapes and sizes, and therefore our method is useful when distributing elements that have similar shapes and sizes but different colors or textures. Related to this point, our method only considers spatial uniformity, and we do not take into account an element's attributes such as colors. One promising future direction is to take such visually meaningful attributes into consideration when generating patterns.

CHAPTER 4

Discrete Color Palettes

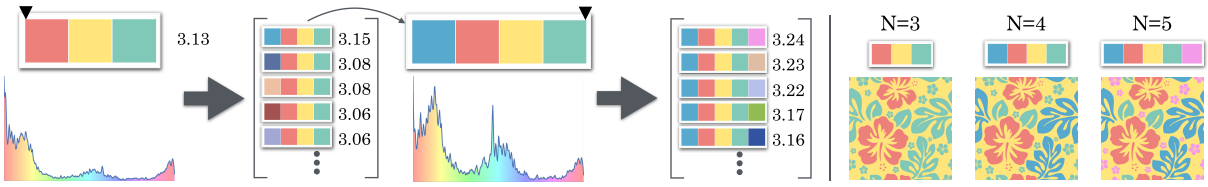


FIGURE 4.1 The proposed method can rate a given color palette with any number of colors relative to human aesthetic preferences. The proposed method suggests a compatible color for the given palette, which allows us to expand the palette while retaining color harmony to support user exploration of color design. Left: given a color palette composed of three colors, our model rate the palette (score: $3.13/5.00$). With a user-provided index (▼), we can suggest compatible colors to the given palette. We explore the HSV color space and sampling the candidates (e.g., the hue value is sampled by rejection sampling using the hue probability distribution as illustrated below the palette). Then, these candidates are rated and the top colors are return to the user. This process can be repeated until the palette has the desired number of colors. Right: a template pattern is colored by the 3-, 4-, and 5-color palette, respectively, where the user provided 3-color palette is expanded to the 4- or 5-color palette using the proposed method.

In this chapter, a model to rate color combinations that consider human aesthetic preferences is proposed. The proposed method does not assume that a color palette has a specific number of colors, i.e., an input is not restricted to a two-, three-, or five-color palettes. We extract features from a color palette whose size does not depend on the number of colors in the palette. The proposed rating prediction model is trained using a human color preference dataset. The model allows a user to extend a color palette, e.g., from three colors to five or seven colors while retaining color harmony. In addition, we present a color search scheme for a given palette and a customized version of the proposed model for a specific color tone. We demonstrate that the proposed model can also be applied to various palette-based applications.

4.1 Introduction

Color palettes and patterns are used in various fields, such as graphics, web, packaging, fashion, and interior design. Such design elements are also used in business, e.g., illustrations in documents and presentations, and play an important role in scientific visualizations. In addition, individuals make personal decisions about aesthetics and harmonious color combinations

(e.g., the color of curtains and wallpaper, room theme colors, or even cushions and other decorative objects in a room). Therefore, evaluating the aesthetics and color harmony of color combinations is not only important for design professionals, but it is also important for many people. However, creating and evaluating a set of colors (i.e., a color palette) is difficult for most people.

Many books, such as [48], that help a user select a color palette based on a *color image*[1] (e.g., *casual*, *elegant*, or *romantic*) have been published. Such books illustrate the relationship between a color image and a three-color palette. On the websites, such as Adobe Color (formerly known as Adobe Kuler) [23] and COLOURlovers [24], users can share the color palettes and patterns with other community members. These services are useful for novices and designers because they can use others' palette designs as references or customize palettes to suit personal requirements and taste. However, customizing a palette composed of N colors that can be expanded to $N+\alpha$ while retaining the original color harmony remains difficult because choosing a color that is compatible with the original palette from many possible colors is challenging and tedious.

We propose a color palette rating model that considers human aesthetic preferences. In addition, we propose a compatible color suggestion method, which is based on the color palette rating model, to expand a given palette while retaining color harmony (Figure 4.1). First, we construct a model that predicts the ratings of a given palette composed of any number of colors by learning from a large dataset of human-rated color palettes. Here, we propose a feature extraction method that does not depend on the size of a color palette, which enables us to rate a color palette composed of any number of colors. We also propose a color suggestion method, wherein we employ a sampling method to search color candidates efficiently from a large color space.

In summary, our contributions are as follows:

- a model that can evaluate a given color palette aesthetics regardless of size, i.e., the number of colors in the palette;
- a feature extraction method to learn the weights of the model;
- a method to suggest a compatible color for palette expansion, i.e., a color that retains color harmony;
- a sampling method to search the compatible color from a color space.

4.2 Model Training for Color Palette Rating

We employ a machine learning approach to learn the weights of a model for rating a color palette. First, we extract features from palettes in a dataset. Then, multivariable regression analysis is applied to learn the weights for each feature vector.

[1] As is used in the other chapters in this thesis, the term *color image* is normally referred to as a two-dimensional pixel array with multiple channels (e.g., RGB). In addition, in this chapter, we use *color image* to refer to the impression we have when we see a set of color combination.

4.2.1 Dataset

We use a MTurk dataset from O'Donovan et al. [73]. In the dataset, $10,743$ color palette ratings by MTurk users are available. The palettes were collected randomly from the Adobe Kuler website. Each palette was rated on a scale of 1-5 by 40 participants.

4.2.2 Feature Extraction

O'Donovan et al. extracted 334 features from a palette [73]. We extract 118 equivalent features from a palette. These include palette colors, mean, standard deviation, median, max, min, and max minus min across a single channel in each color space, i.e., RGB, CIELAB, HSV, and CHSV[2]. We also extract plane-fitting features, i.e., a 2D plane is fit to 3D color coordinates using PCA in RGB, CIELAB, and CHSV color spaces, while in the HSV color space, we extract hue entropy and hue joint/adjacent probabilities. Please refer to the original paper for the details [73]. On the other hand, we exclude features that depend on the number of colors in a palette, i.e., we do not extract colors sorted by lightness, by differences between adjacent colors, and by color differences. Using such features, the total length of the feature vector varies based on the number of colors in a given palette. This is why O'Donovan's model can only be applied to a five-color palette. However, our proposed model does not suffer such restrictions.

In addition, we include a color harmony term [78]. In addition, a "gradation of lightness" and a "gradation of hue" of colors in a palette are included in our feature extraction method. These are based on the fact that as the order of colors in a palette becomes increasingly linear, human ratings tend to increase. The details of feature extraction are described in Chapter B. In total, 121 features were extracted and used for learning and rating.

4.2.3 Learning Model Weights

We use LASSO regression [99] in the same way as [56, 73] for learning the weights of each feature. The equation for rating color palette $\mathbf{t}$ is given by $r(\mathbf{t}) = \mathbf{w}^\top \mathbf{f}(\mathbf{t}) + b$, where $\mathbf{w}$ is a weight vector, $\mathbf{f}$ is a feature vector, and b is a bias term. Then, the weight vector and the bias are leaned with L_1 regularization:

$$\min_{\mathbf{w},b} \sum_i (\mathbf{w}^\top \mathbf{f}_i + b - r_i)^2 + \lambda \|\mathbf{w}\|_1, \tag{4.1}$$

where r_i is an actual user rating (1-5) from the dataset described in Section 4.2.1 and λ is a regularization parameter ($\lambda = 2.2 \times 10^{-6}$ by 10-fold cross validation). The dataset is split into learning and test datasets at a ratio of $6:4$. Due to the L_1 regularization, the extracted features are expected to have adequate weight even if there are similar features in the extracted feature vector.

[2] A space where hue θ and saturation s are remapped to Cartesian coordinates: $d_1 = s\cos(\theta)$ and $d_2 = \sin(\theta)$.

4.3 Suggesting Compatible Colors

Using the proposed model, we can suggest a compatible color for a given color palette, while in [73], they can only suggest a fifth color for a given four-color palette.

Given a color palette with any number of colors, an index position the user would like to insert a color in the palette (e.g., the i-th index from the left of the palette), and optionally the number of colors M to be suggested, the proposed method can suggest compatible colors. First, we search a compatible color and add the color to a given palette. Then, the palette is rated by the rating prediction model. The top M high-rated colors are suggested to the user. To avoid suggesting overly similar colors for the given palette to the human eye, we calculate CIEDE2000 (ΔE_{00}) [95] for each color in the palette and the candidate color, and we collect only candidates with ΔE_{00} values greater than a threshold τ. Similar to the τ thresholding, we calculate the ΔE_{00} for each two candidate color combinations out of M. Note that we collect only candidates with ΔE_{00} values greater than a threshold κ. We explore the color space with those constraints and select candidate colors by rejection sampling, which allows us to suggest various colors to the user.

4.3.1 Compatible Color Suggestion

Given a color palette $\mathbf{t} = [\mathrm{c}_1, \mathrm{c}_2, \ldots, \mathrm{c}_n]$, we solve the following optimization problem for a compatible color candidate c^{cand} with the given palette:

$$\begin{aligned} \max_{\mathrm{c}^{\text{cand}}} \quad & r(\mathbf{t}) \\ \text{subject to} \quad & \Delta E_{00}(\mathrm{c}^{\text{cand}}, \mathrm{c}_i) \geq \tau, \ \forall i \in \{1, 2, \ldots, n\}, \end{aligned} \tag{4.2}$$

where, $r(\mathbf{t})$ is a color palette rating by the learned model, $\Delta E_{00}(\mathrm{c}_1, \mathrm{c}_2)$ is CIE Delta E of c_1 and c_2, and τ is the user-defined threshold to suggest colors that are substantially different from the colors in the given palette.

4.3.2 Color Space Exploration

Since the color space is vast, it is necessary to search color candidates efficiently in order to suggest compatible colors to the user. It took 12 hours to brute-force search a candidate in the RGB color space (i.e., the total number of the solution candidates is 255^3) with a Core i7 2.3 GHz, 16 GB RAM machine. Therefore, we perform the following procedure to sample candidates in the HSV color space.

First, we calculate an adjacent hue probability distribution of a color c. Here, $\mathbf{p}_{\mathrm{c}}^{\text{joint}} = [P_{h_{\mathrm{c}},0}^{\text{joint}}, P_{h_{\mathrm{c}},1}^{\text{joint}}, P_{h_{\mathrm{c}},2}^{\text{joint}}, \ldots, P_{h_{\mathrm{c}},359}^{\text{joint}}]$, where $h_{\mathrm{c}} \in [0, 360)$ is the hue of c and $P_{h_1,h_2}^{\text{joint}}$ is the probability of co-occurrence of colors with its hues h_1 and h_2 in the same palette. The adjacent hue probability distribution of color c denoted by $\mathbf{p}_{\mathrm{c}}^{\text{adj}}$, which is the probability of co-occurrence of the adjacent hues h_1 and h_2 in the same palette, is calculated in the same manner. With this configuration, we perform rejection sampling using a hue probability distribution function to

select a candidate hue $\mathrm{c}^{\text{cand}} = (h, s, v)$ depending on the index position at which the candidate is to be inserted.

A case whereby the specified index in a palette is the head or tail of the palette, we compute the following as the hue probability distribution function:

$$\alpha \cdot \mathbf{p}_{\mathrm{c}_i}^{\text{adj}} + (1-\alpha) \cdot \frac{1}{n-1} \sum_{\mathrm{c}_k \in \mathcal{C} \setminus \{\mathrm{c}_i\}} \mathbf{p}_{\mathrm{c}_k}^{\text{joint}}, \tag{4.3}$$

where c_i is a color adjacent to c^{cand}, n is the number of colors in the given palette, c_k represents non-adjacent colors to c^{cand}, and α is a balancing parameter of the two terms ($\alpha = 0.5$ in this study). We compute P^{adj} with the adjacent color and P^{joint} with the remaining non-adjacent colors. When a candidate is to be inserted between two colors in a palette, we compute the following hue probability distribution function:

$$\alpha \cdot \frac{\mathbf{p}_{c_i}^{\text{adj}} + \mathbf{p}_{c_j}^{\text{adj}}}{2} + (1-\alpha) \cdot \frac{1}{n-2} \sum_{c_k \in \mathcal{C} \setminus \{c_i, c_j\}} \mathbf{p}_{c_k}^{\text{joint}}, \tag{4.4}$$

where c_i and c_j are adjacent colors of c^{cand}, and c_k represents non-adjacent colors of c^{cand} in the palette. Figure 4.2 shows the hue probability distribution function calculated for various color palettes where a candidate is to be inserted at ▼.

We also sample the candidate's saturation s with $s \sim \mathcal{N}(\mu_s, \sigma_s)$ and the value v with $v \sim \mathcal{N}(\mu_v, \sigma_v)$ from the given color palette, where $\mathcal{N}(\mu, \sigma)$ is a normal probability distribution with mean μ and standard deviation σ.

In addition, we constrain the color differences of any two color combinations from M candidates by computing the following constraint, which ensures the diversity of the suggested colors:

$$\Delta E_{00}(\mathrm{c}_{\mathrm{j}}^{\text{cand}}, \mathrm{c}_{\mathrm{k}}^{\text{cand}}) \geq \kappa, \ j \neq k, \ \forall j, k \in \{1, 2, \ldots, M\}, \tag{4.5}$$

where κ is a user-defined threshold. The proposed sampling method results in few seconds to several tens of seconds depending on the number of samples (a few thousand to $100,000$), and we also find that the rejection sampling algorithm is approximately 10 times faster than naive random hue sampling. The pseudo code of the rejection sampling algorithm is provided in Chapter B.

4.4 Customizing the Model to a Specific Color Tone

We use the general-purpose dataset described in Section 4.2.1 rather than a dataset with a specific concept. Therefore, it sometimes does not give the user a satisfactory rating or color suggestion with the model if the given palette has a specific context, such as *pastel colors*. The suggested colors tend to be relatively dark with low saturation.

To address this issue, we customize the model based on the idea proposed by Chen et al. [18],

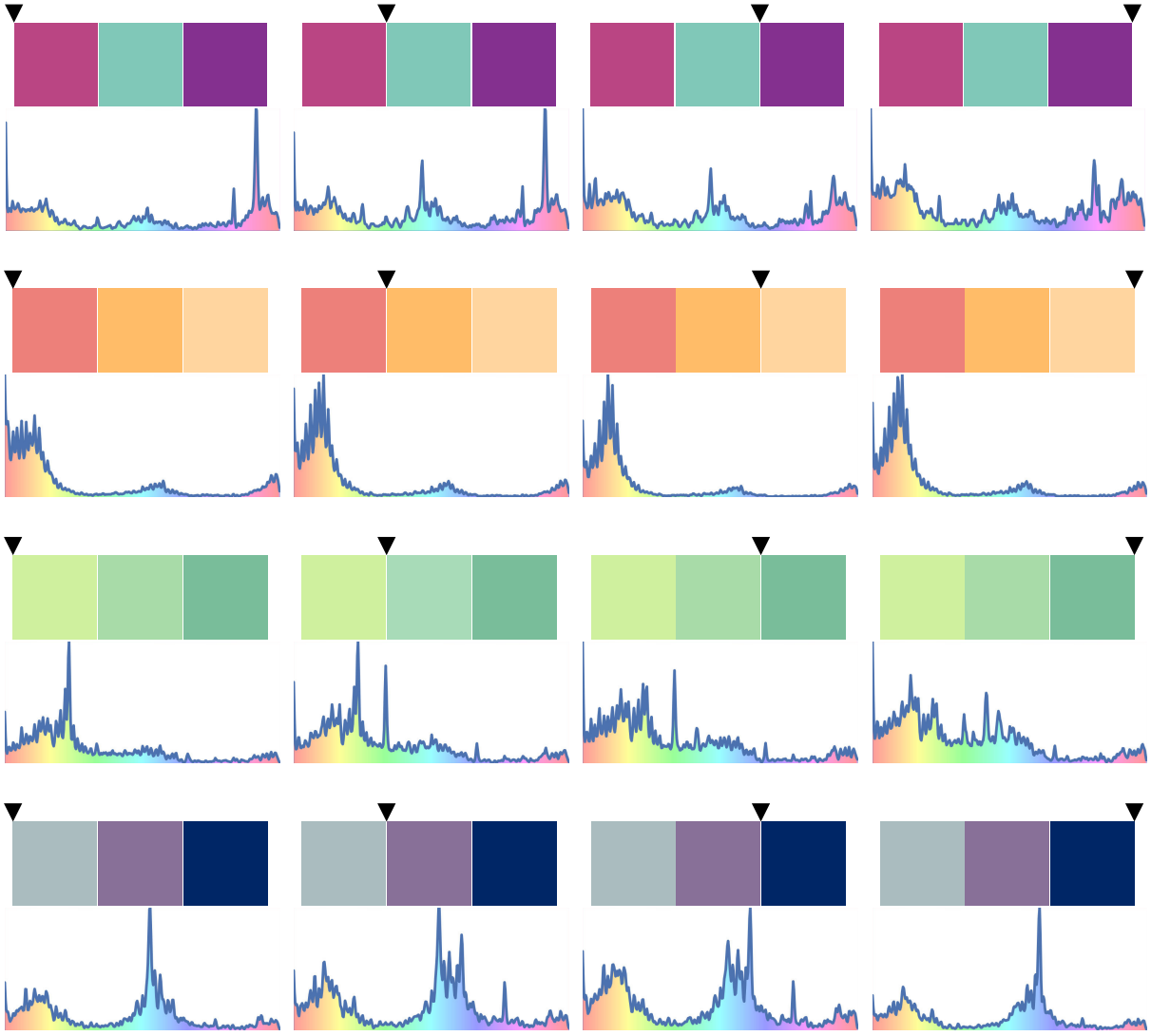


FIGURE 4.2 Hue probability distribution functions for various color palettes when a color is to be inserted between the index specified by ▼. The distribution functions are calculated by equations 4.3 and 4.4, and they are different from each other as it depends on the colors comprising the palette, the location of the index ▼, and its adjacent colors.

i.e., we rerate the all color palettes in the dataset using an additional dataset:

$$r'(\mathbf{t}) = \delta \cdot r(\mathbf{t}) + (1-\delta) \cdot e^{-\min_{\mathbf{t}' \in T} d(\mathbf{t}, \mathbf{t}')}, \tag{4.6}$$

where $\mathbf{t}$ is a color palette in the dataset described in Section 4.2.1 and $\mathbf{t}' \in T$ is a color palette in the new dataset composed of a specific color to be used for customization. δ is a balancing parameter for the two terms ($\delta = 0.6$ in this study).

In [18], they employed a weighted sum of the squared Euclidean distance of all five colors in a HSV space palette to calculate $d(\mathbf{t}, \mathbf{t}')$. Note that their method can only deal with five-color palettes; however, since our method is not restricted to five-color palettes, we must be able to compute the distance of two palettes with different numbers of colors (e.g., the distance between a three-color palette and a five-color palette). To do so, we use the following distance function:

$$d(\mathbf{t}, \mathbf{t}') = \mathrm{EMD}_{\Delta E_{00}}(\mathbf{t}, \mathbf{t}'), \tag{4.7}$$

where $\mathrm{EMD}_{\Delta E_{00}}(\mathbf{t}, \mathbf{t}')$ is the Earth Mover's Distance (EMD) [89], and we employ ΔE_{00} as the distance metric. The EMD with ΔE_{00} is defined as follows:

$$\mathrm{EMD}_{\Delta E_{00}}(\mathbf{t}, \mathbf{t}') = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}\Delta E_{00}(\mathbf{t}_i, \mathbf{t}'_j)}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}}, \tag{4.8}$$

where $f_{i,j}$ is a flow between two histograms or probability distributions that move one distribution to the other. In our context, we consider a flow from t_i to t'_j, where t_i and t'_j are the i-th and j-th color in each color palette with m and n colors, respectively. The flow $F = [f_{i,j}]$ is computed by solving the following optimization problem:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}\Delta E_{00}(\mathbf{t}_i, \mathbf{t}'_j) \\
\text{subject to} \quad & f_{i,j} \geq 0, 1 \leq i \leq m, 1 \leq j \leq n, \\
& \sum_{j=1}^{n} f_{i,j} \leq w_{\mathbf{t}_i}, 1 \leq i \leq m, \\
& \sum_{i=1}^{m} f_{i,j} \leq w_{\mathbf{t}'_j}, 1 \leq j \leq m, \\
& \sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j} = \min\left(\sum_{i=1}^{m} w_{\mathbf{t}_i}, \sum_{j=1}^{n} w_{\mathbf{t}'_j}\right),
\end{aligned}$$

where $\mathbf{w_t} = \mathbf{w_{t'}} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ is the weight vector. The above problem is an optimal transport problem and solved using mathematical programming [39].

After rerating all palettes in the original dataset, we then combine the additional and original datasets by giving full-rate to all new color palettes to build a customized dataset for *pastel* colors and so on.

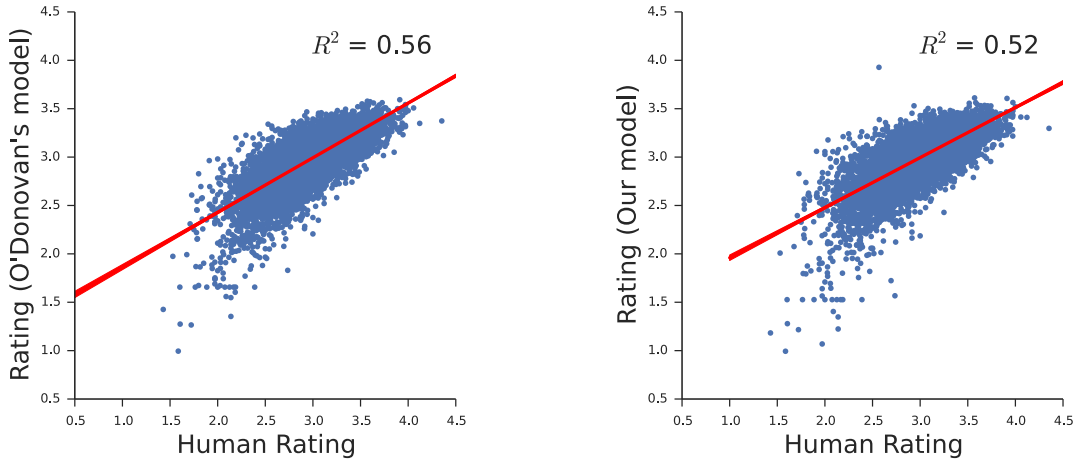
4.5 Results

4.5.1 Model Analysis

Here, we compare our learned model with a previously proposed model [73]. The mean absolute errors (MAE) and mean squared errors (MSE) are shown in Table 4.1. The baseline was computed by the difference between the test and training data calculated by the proposed method [73]. For [73], the MAE decreased by 33% and MSE decreased by 55% compared to the baseline. For the proposed method, MAE and MSE decreased by 30% and 51%, respectively. We also performed a correlation analysis of ratings for both models (Figure 4.3). As can be seen, $R^2 = 0.56$ in [73] and $R^2 = 0.52$ for the proposed method. Although the proposed method is less accurate than [73] in both cases, the MAE and MSE values are reduced compared to the baseline and R^2 is greater than 0.5, which indicates that the proposed model can still rate color palettes with respect to human aesthetic preferences with sufficient accuracy. In addition, the proposed model can rate color palettes with any number of colors, while the model proposed in [73] can only rate a five-color palette.

TABLE 4.1 Comparison of MAE and MSE for [73] and the proposed model.

	Fixed (baseline)	[73]	Proposed model
MAE	0.267	0.180	0.187
MSE	0.116	0.052	0.057



(a) [73] (b) Ours.

FIGURE 4.3 Comparison of correlation coefficient R of human ratings for [73] (a) and the proposed model (b).

As the learning result with L_1 regularization shows, 19 features out of 121 received zero weights, and the other 102 features received nonzero weights. The Color Harmony (CH) and Gradation (Grads) terms added to the feature extraction method receive nonzero weights, which contributes to increased correlation. We further analyzed these features and found that, although the CH term received nonzero weights, it does not sufficiently contribute to correlation. On the other hand, the correlation coefficient R increases by 0.01 due to the Grads term (Table 4.2). A more detailed analysis is provided in Chapter B.

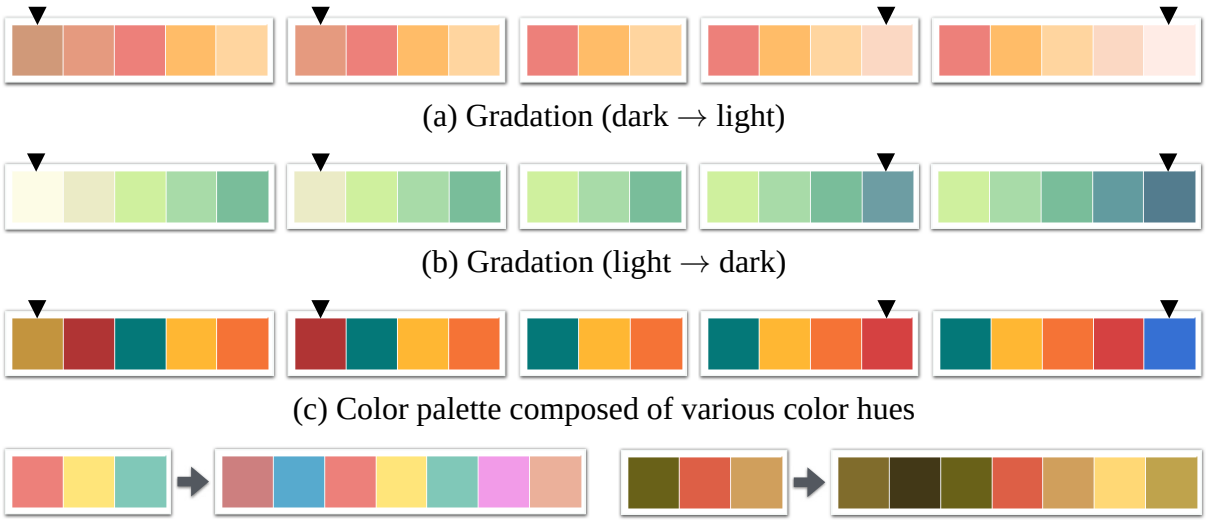
TABLE 4.2 Comparison of the correlation coefficient of predicted rating by the proposed model and human ratings. The R value was calculated with and without the CH and Grads terms.

	[73]	w/o	w/ CH	w/ Grads	w/ both
R value	0.76	0.71	0.71	0.72	0.72

4.5.2 Analysis of Compatible Color Suggestion

Figure 4.4 shows the results of the color suggestions for the given palettes. The proposed method can suggest appropriate colors for the gradation color palettes (Figures 4.4a and 4.4b) and can suggest well-chosen colors for a palette that includes various hues (Figures 4.4c and 4.4d). Using the proposed color suggestion method, the input three-color palettes are expanded to four-, five-, or seven-color palettes while retaining color harmony.

We conducted an experiment to evaluate the performance of the palette expansion algorithm. For 10 three-color palettes and 10 five-color palettes, we applied our compatible color



(a) Gradation (dark → light)

(b) Gradation (light → dark)

(c) Color palette composed of various color hues

(d) Palette expansion from 3-color to 7-color

FIGURE 4.4 Palette expansion results of the proposed method. Rows 1-3: a suggested color is added at the index specified by ▼. Row 4: two types of palettes are expanded from three-color to seven-color palettes by adding suggested colors side-by-side. $\tau = 5$ for all results.

suggestion method (*Ours(Best)*) to expand these palettes to four-color and six-color palettes, respectively. For comparison, we generated palettes using the *incompatible* color suggestion method (*Ours(Worst)*), in which we sampled the lowest rated hue by our model in HSV color space, and the saturation and value were sampled in the manner described in Section 4.3.2. We also generated palettes with additional colors sampled randomly in RGB color space (*Random*). For 20 cases, we asked 17 participants (four females) with a normal color vision to rank the three palettes generated by the different methods relative to *Naturalness ("expansion with additional color is natural")* and *Compatibility ("the expanded palette has compatibility to the original palette")*. In the experiment, each participant was shown an original three-color palette at the top of the display, and three expanded palettes with four colors were displayed at the bottom, side-by-side in a random order.

We applied Friedman's test and if a significant difference ($p < 0.05$) was observed, we proceeded to perform Wilcoxon tests with Holm corrections for multiple comparisons. The results showed that, for 12 of the 20 cases, the palettes expanded by our method were statistically significant for both the *Ours(Worst)* and *Random* methods. In addition, even in cases where our method had no statistical significance, Figure 4.5 shows that the palettes generated by our method were most frequently chosen by participants as the best palette, which demonstrates the effectiveness of the proposed method.

4.5.3 Palette Index for Variations of Color Suggestions

As shown in Figure 4.6, the suggested colors can vary depending on where the color is to be inserted because the rating model considers the order of colors in a palette. One may consider a color palette as simply a set of colors that is independent of the order of the colors in the palette, however, we must consider color order because such sets of colors should be arranged to form a single group. Therefore, we consider that the order is important for us to evaluate the aesthetics

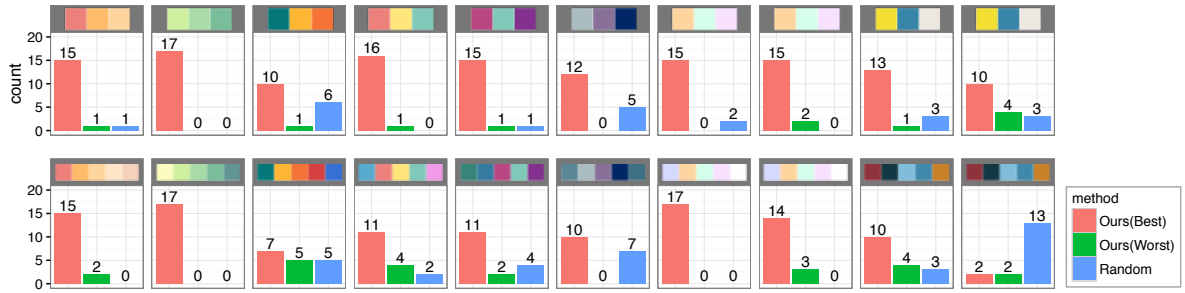


FIGURE 4.5 Number of times that palettes generated by three different methods were chosen by participants as the best compatible palette. Top: color suggestions for three-color palettes to be expanded to four-color palettes. Bottom: color suggestions for five-color palettes to be expanded to six-color palettes. We expanded the palettes listed at the 8th and 10th columns with *pastel* and *retro* customized models, respectively.

of a color palette.

FIGURE 4.6 The suggested colors (▼) are varied depending on the index of the palette ($\tau = 5$).

We also apply a palette re-ordering to a palette $\mathbf{t}$, where we select the highest rated palette from all possible permutations; $\mathbf{t} = \arg\max_{r \in \mathcal{P}(\mathbf{t})} r(\mathbf{t})$, where $\mathcal{P}(\mathbf{t})$ denotes the $7! = 5,040$ possible permutations of a palette in the results shown in Table 4.3.

TABLE 4.3 Comparison of ratings for original and re-ordered palettes.

	original palette		re-ordered palette	
#1		3.40		3.41
#2		3.43		3.49
#3		3.41		3.45
#4		3.37		3.37
#5		3.09		3.21
#6		3.52		3.52

In addition, we conduct a user study to analyze user preferences of a color order of a palette. In the study, we show a participant a pair of color palettes (an original and the re-ordered) listed in Table 4.3 except for #6 since it is not re-ordered. We shuffle the left or right side of the two palettes randomly and ask "*Which color palette is more harmonious?*" to 48 participants (36 males and 12 females), 27 of them major in design. The age of the participants lies in 20 – 25. The result is shown in Figure 4.7. According to the histogram, the re-ordering approach is effective for improving color palette evaluation of human aesthetics preferences.

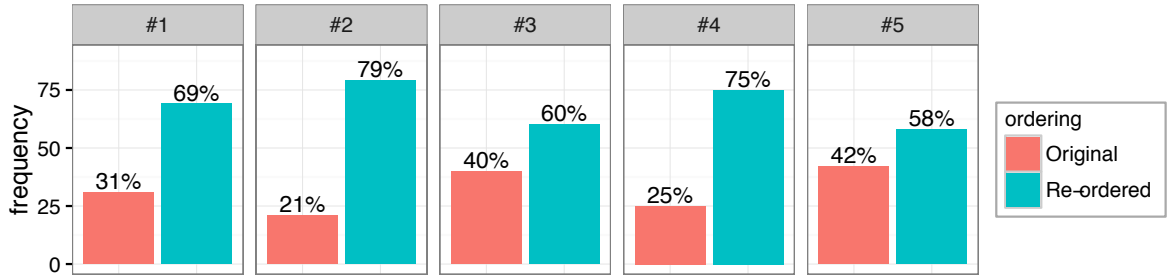
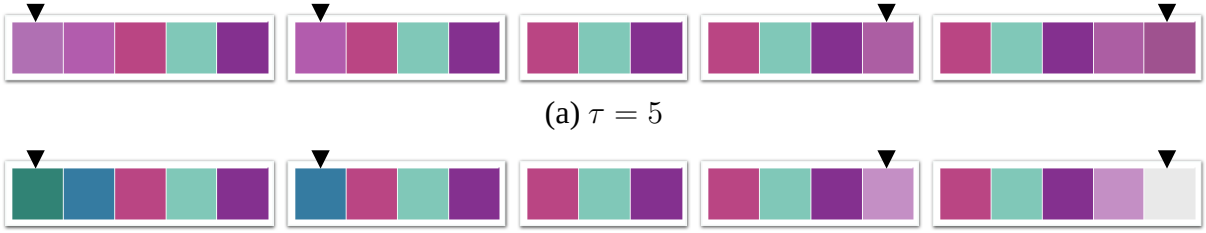


FIGURE 4.7 Preference analysis from a user study. For each palette, the participants selected either an original or the re-ordered palette. The palettes used in the study were taken from #1 – #5 shown in Table 4.3.

4.5.4 Choice of τ

Figure 4.8 shows the results of choosing different τ values in Equation 4.2. Although Figure 4.6 does give color suggestion results with $\tau = 5$ and sufficient hue variations while maintaining overall color compatibility, it does not always suggest sufficient hue variation, as shown in Figure 4.8a. When the user desires a variety of colors that differ sufficiently from the colors in the original palette, the proposed color suggestion method can suggest various colors that are sufficient to perceive the differences for the human visual system by setting τ to a larger value. As can be seen in Figure 4.8a, $\tau = 5$ does not give sufficient differences because the value is close to a just-noticeable difference (the value around 2.3 [63]). With a larger value, e.g., $\tau = 20$, in Figure 4.8b, the proposed method can suggest a wide variety of colors.



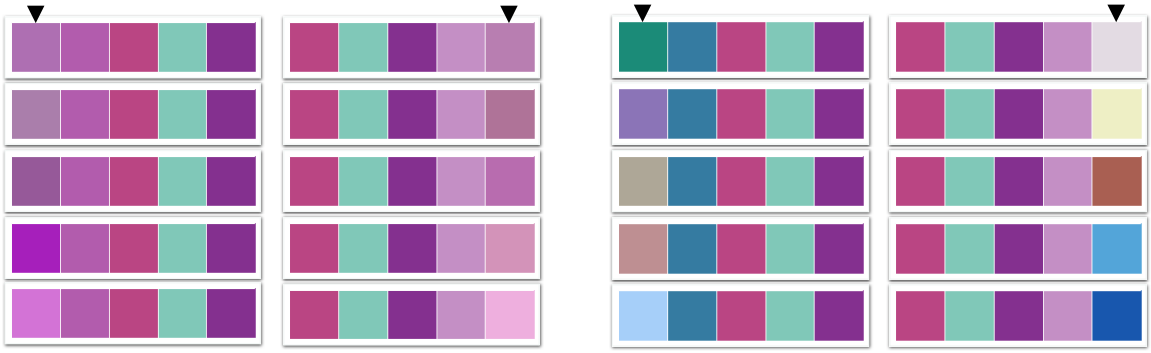
(a) $\tau = 5$

(b) $\tau = 20$

FIGURE 4.8 τ affects the color suggestion results. With a small τ value, the results can be similar to the colors already in the given palette. With a greater τ value, perceptually different but compatible colors can be suggested. For a three-color palette with different τ values ((a) $\tau = 5$ and (b) $\tau = 20$), the suggested colors are marked as ▼.

4.5.5 Choice of κ

As shown in Figure 4.9a, the color candidates tend to be similar to the colors in the given palette if κ in Equation 4.5 is small. To avoid this, greater κ will suggest colors with more variety (Figure 4.9b) while the palette maintains color harmony.



(a) $\tau = 5,\ \kappa = 5$ (b) $\tau = 20,\ \kappa = 20$

FIGURE 4.9 κ affects the color suggestion results. Given a four-color palette with different τ and κ, the suggested colors are inserted at both ends (the ▼ columns). Top to bottom is higher to lower ratings. $N = 4,\ M = 5$.

4.5.6 Analysis of Model Customization

For the model customization, we collected $1,403$ *pastel* color palettes from Pastel_-L♥vers [26] and $3,176$ *retro* color palettes from the Retro group [27], and we customized the model with these additional datasets by applying the method described in Section 4.4.

Table 4.4 shows a comparison of the color suggestion results with and without the *pastel*-customized models. As can be seen, the customized model can suggest more compatible colors than the model without customization.

TABLE 4.4 Comparison of color suggestions with and without the model customized with *pastel* colors. The suggested colors are listed in the columns marked ▼, and the rating is shown next to it. $N = 3,\ M = 5,\ \tau = 5$, and $\kappa = 20$.

	w/o custom model		w/ *pastel* model	
#1		3.16		4.47
#2		3.14		4.35
#3		3.10		4.35
#4		3.07		4.23
#5		3.07		4.18

Table 4.5 shows a comparison of the ratings results with and without the *pastel* and *retro* customizations. We rated various three-color palettes using these models. As can be seen for the *pastel*-customized model, the ratings are higher than those obtained without the customized model for the palette shown by (a) in Table 4.5.

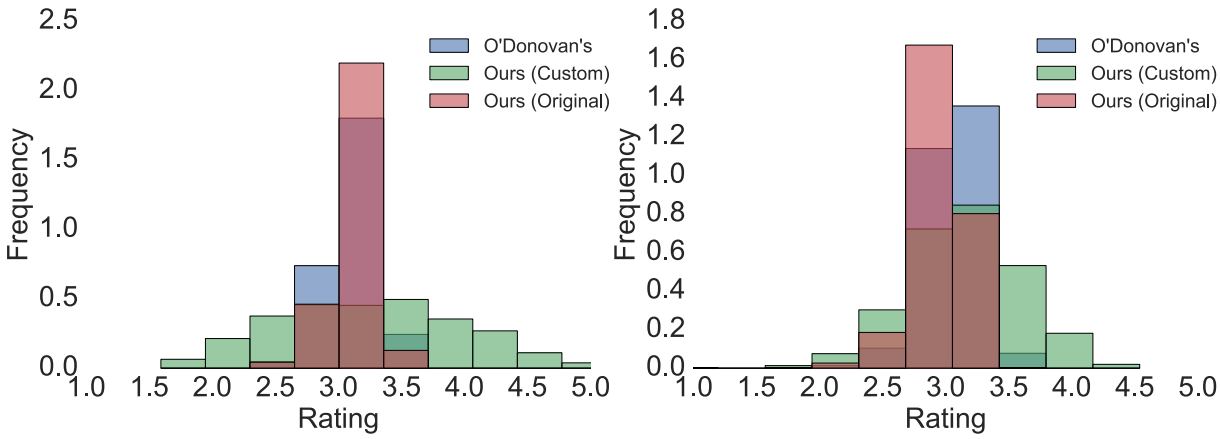
In addition, the *pastel* model gives higher ratings to gradation palettes, as shown in Table 4.5b and c, while the palette shown by (e) – (i) in Table 4.5 obtained lower ratings. On the

other hand, the *Retro* model rated the palette shown in Table 4.5h higher, and the *pastel* color palette shown in Table 4.5a was also rated higher, which suggests that the *retro* model gives higher ratings to palettes comprising colors where saturation values are very similar. We also found that the *pastel* model rates were lower for Table 4.5i, while the *retro* model rates it with nearly the same rating as the non-customized model.

TABLE 4.5 Comparison of ratings for various color palettes with and without model customization.

	color palette	model customization		
		w/o	*pastel*	*retro*
(a)		2.90	4.24	3.50
(b)		3.18	3.73	2.75
(c)		2.95	3.20	3.12
(d)		3.01	3.16	2.85
(e)		3.21	3.02	3.23
(f)		2.76	2.45	2.94
(g)		2.95	2.24	3.07
(h)		2.84	2.63	4.02
(i)		2.65	1.57	2.66

Furthermore, we compared our *pastel* and *retro* customized model to our original and O'Donovan's model [73]. For the comparison, we collected 664 *pastel* color palettes and $1,222$ *retro* color palettes from P a s t e l C u t i e s [25] and RETROpolis [28], respectively. Then, we rated the *pastel* dataset by our *pastel*-customized and original model, and O'Donovan's model (Figure4.10a), and the rated *retro* dataset by our *retro*-customized and original model, and O'Donovan's model (Figure4.10b). These means and standard deviations (StdDev) are listed in Table 4.6. They showed that, the customized model obtained higher mean ratings in both cases, and larger StdDevs are considered to these abilities to rate the specific colors in detail, while O'Donovan's and our original model can not capture the differences of the specific colors well since these models have smaller StdDevs. Although the original model has a less correlation to human ratings in Table 4.1 and Figure 4.3, the customized version of the proposed models compare favorably to O'Donovan's model [73].



(a) *Pastel* dataset (b) *Retro* dataset

FIGURE 4.10 Histograms of ratings of *pastel* dataset (a) and *retro* dataset (b) of our *pastel*-customized and *retro*-customized and original model, and O'Donovan's model [73].

TABLE 4.6 Mean and StdDev for Figure 4.10.

Dataset		[73]	Ours (Original)	Ours (Custom)
Pastel	Mean	3.11	3.13	3.24
	StdDev	0.197	0.166	0.719
Retro	Mean	3.06	2.95	3.15
	StdDev	0.207	0.198	0.452

4.6 Applications

4.6.1 Coloring 2D Patterns

We applied the proposed color suggestion method to a 2D pattern template from COLOURlovers (Figure 4.11). First, we selected a three-color palette with color images (e.g., *mysterious*, *noble*, *dreamy*, and *progressive*) [48]. Then, the three-color palette was expanded to a five-color palette using the proposed color suggestion method. We performed pattern coloring with three-color (Figure 4.11, left) and five-color palette (Figure 4.11, right). As can be seen in the five-color palette and its pattern coloring results, the color image does not change from that of the original three-color palette. Since the pattern coloring with the five-color palette has $5! = 120$ possible pattern coloring combinations, we employed the color assignment method proposed in [46] for coloring and selected the top-ranked results in Figure 4.11.

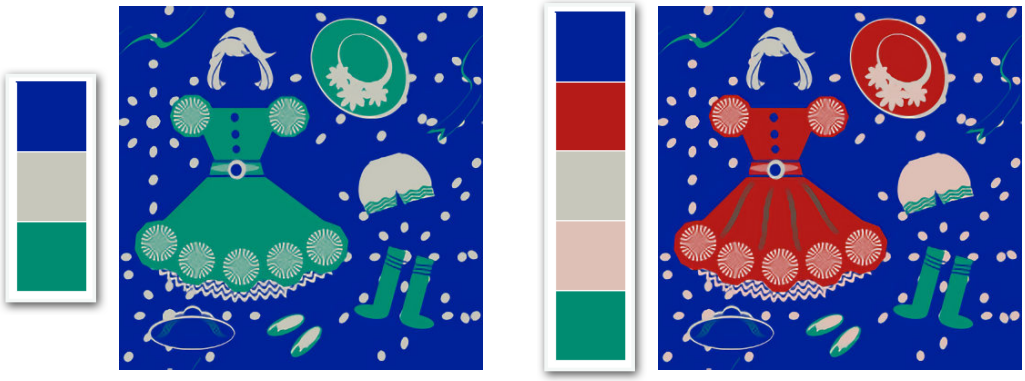
4.6.2 Coloring Segmented 3D Model

We also applied the method to 3D model coloring. Given a segmented model, we group these segments into three to seven group and color them with three-color, five-color, or seven-color palettes (Figure 4.12). It can be noted that such color variations are useful, for example, a user already has a basic color concept (e.g., three basic colors) and would like to refine the color design while retaining the color image.

(a) Color image: *Mysterious*

(b) Color image: *Noble*

(c) Color image: *Dreamy (pastel model)*



(d) Color image: *Progressive (retro model)*

FIGURE 4.11 Coloring pattern templates. The original three-color palette color image (left) was expanded to a five-color palette using the colors suggested by the proposed method while retaining its color image.

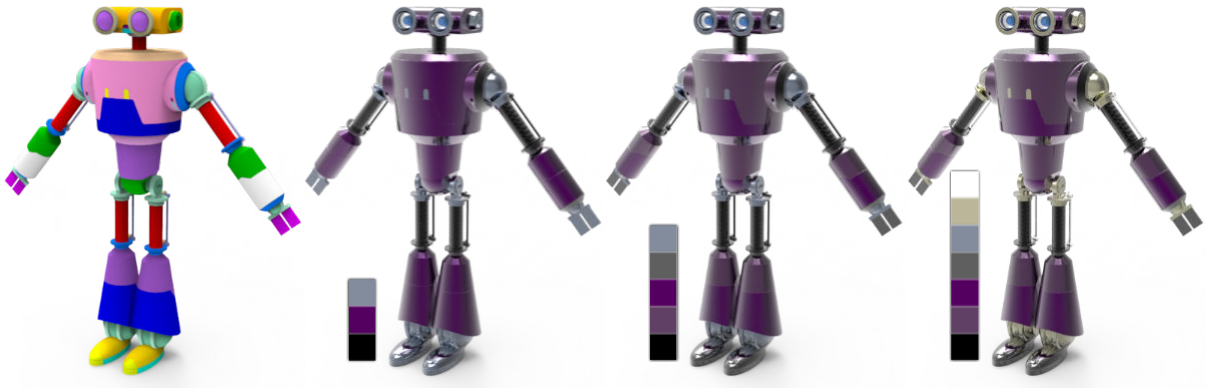


FIGURE 4.12 Robot color design. Left to right: segmented model, model colored by a three-color palette, model colored by a five-color palette, and model colored by a seven-color palette while retaining the color image.

4.6.3 Photo Recoloring

A user can produce enhanced photo-editing results by incorporating a palette-based photo recoloring method [15] with our palette expansion method. First, we prepare a three-color palette with a target color image. Such a palette with a certain color image can be obtained from the Internet (e.g., by searching for *Pastel* or *Retro* palettes) or from a book [48]. Second, we apply the palette expansion method to the palette to obtain additional compatible colors. We also prepare a target photo to be recolored and apply the palette-based photo recoloring method to extract a source palette from the photo. Then, each color in the source palette is replaced with our palette colors using the color transfer function to recolor the photo. Figures 4.13 and 4.17 show the results. Using our palette expansion method, a user can obtain more colors in the palette while retaining its color image when the colors of the collected palette are limited, which provide more flexibility.

FIGURE 4.13 Recoloring results with *Mysterious* palettes. The original photo (left) is recolored using the palette-based photo recoloring method with three- (middle) and seven-color palettes (right). *Photo courtesy of Tommie Hansen (Flickr).*

4.6.4 Adding Items

A user can add items with different colors to a set according to the color suggestions. For example, for a set of pens with different colors (Figure 4.14 left), the additional pens with

FIGURE 4.14 Pen set

different yet compatible colors are added to the set (Figure 4.14 right). This is useful when providing color variations. Another example is provided in Figure 4.15, in which additional objects (e.g., a blind and a sofa) with colors that are compatible with the room's color theme are added to the room. Here, we assign the color to a diffuse color of the object.

FIGURE 4.15 For a bedroom with an associated color theme (left), a window blind and a sofa with colors assigned according to the extended palette are added to the room (right). *Model courtesy of 3D Bar.*

4.7 Conclusions and Limitations

4.7.1 Conclusions

In this work, we have proposed a rating prediction model for a given color palette that can comprise any number of colors. We have also proposed a compatible color suggestion method for palette expansion while retaining color harmony, as well as a method to explore color space

to select color candidates. With the proposed model, we can evaluate simple palettes, such as gradations, and we can rate palettes comprising various color tones and hues.

4.7.2 Limitations

Since the proposed method employs a machine learning approach, the rating prediction of the model depends strongly on the quality of the dataset. We specialized our model by collecting *pastel* and *retro* palettes from COLOURlovers with relatively little effort. However, if only a few datasets are available or it is difficult to collect datasets from the Internet, it is difficult to customize the model. It is also difficult to determine how many palettes need to be customized for sufficient rating accuracy. In addition, we have demonstrated coloring pattern templates as an application; however, the color harmony of a palette is not directly related to the target pattern mood, which can result in mismatched coloring. Figure 4.16 shows an example, in which we prefer Figure 4.16c over Figure 4.16b relative to the mood of the target content in the pattern template (Figure 4.16a). That is, although we proposed a palette rating model, we cannot rate the goodness of colored pattern using our rating model as it has semantics and our model cannot capture the semantics.

(a) Template (b) *Merry* (c) *Noble*

FIGURE 4.16 Coloring pattern templates using two color palettes with different color images.

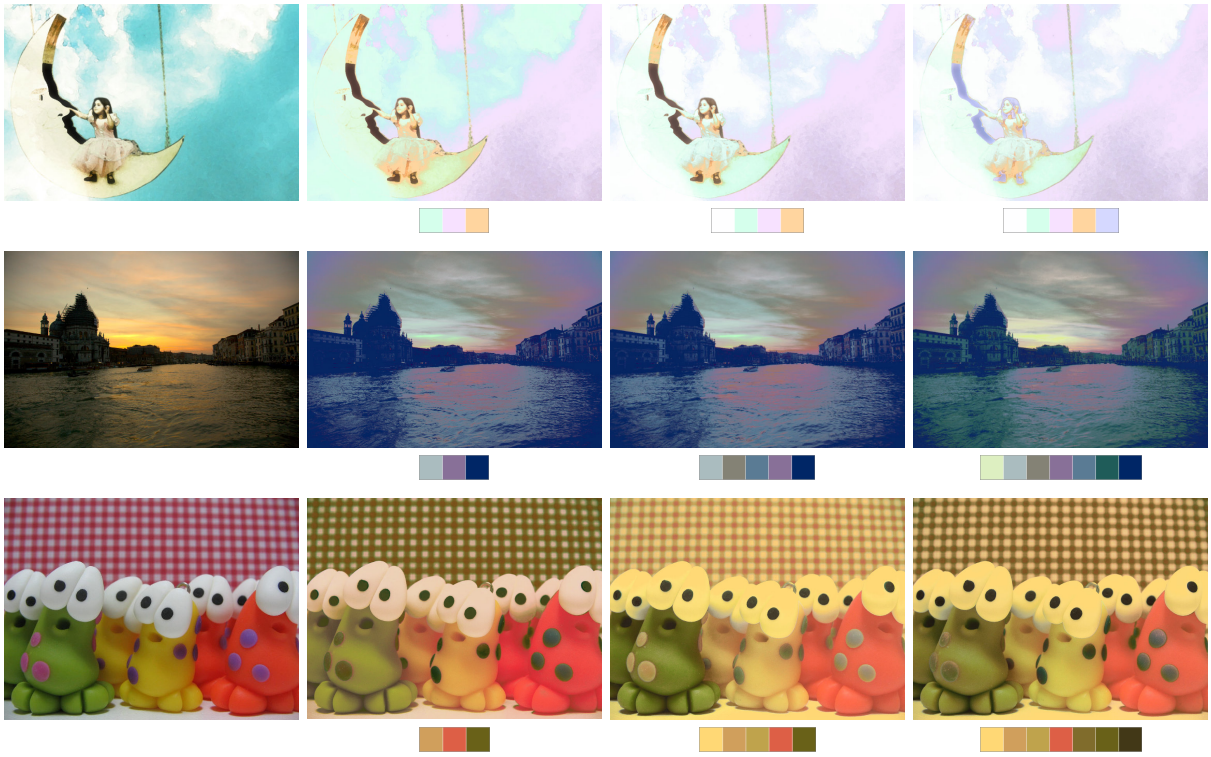


FIGURE 4.17 Top: recoloring results with *Dreamy* palettes. Middle: recoloring results with *Noble* palettes. Bottom: recoloring results with *Retro* palettes. The first column shows the original images. *Photos courtesy of Celine Nadeau (top) and Ana Kuhnen (bottom) (Flickr), and the MIT-Adobe FiveK Dataset [11] (middle).*

CHAPTER 5

Discrete Element Patterns

Visual cryptography (VC) is an encryption technique for hiding a secret image in distributed and shared images (referred to as *shares*). VC schemes are employed to encrypt multiple images as meaningless, noisy patterns or meaningful images. However, decrypting multiple secret images using a unique share is difficult with traditional VC.

In this chapter, we propose an approach to hide multiple images in meaningful shares. We can decrypt multiple images simultaneously using a common share, which we refer to as a Magic Sheet. The Magic Sheet decrypts multiple secret images depending on a given share. The shares are printed on transparencies, and decryption is performed by physically superimposing the transparencies. The proposed method was evaluated using binary, grayscale, and color images.

5.1 Introduction

Visual cryptography (VC) is a secret sharing scheme where secrets are hidden in distributed and shared images. The secrets can be decrypted successfully if shared images (hereafter *shares*) printed on transparencies are stacked (superimposed). Decryption can be performed by the human visual system; therefore, computational resources are not required for decryption. VC applications include secret message sharing, authentication and identification, and watermarking. To encrypt secrets securely, the secrets also need to be split and embedded into shares with high-quality images. VC has been studied extensively by the cryptography community and other communities related to visual media, including computer vision and computer graphics.

In the computer graphics community, various approaches for hiding images in a surface/display have been investigated. In such approaches, a different image is displayed on the surface depending on the viewer's perspective or lighting conditions. Here the surface displays the hidden images itself or projects it onto a wall from an unstructured meaningless pattern or a different image shown on the surface. Such surprising behavior evokes a sense of wonder; therefore, the technique can be applied to various entertainment applications. This is also true for a VC scheme (VCS). In other words, although VC is essentially a cryptography/steganography method, it can also be used in entertainment applications.

We propose *Magic Sheet*, an approach to hide multiple images in sheets. The proposed method is based on the (k, n)-VCS, where a secret can be decrypted by stacking k out of n images using a VCS, whereas any $k-1$ of n images cannot decrypt the secret successfully. Note

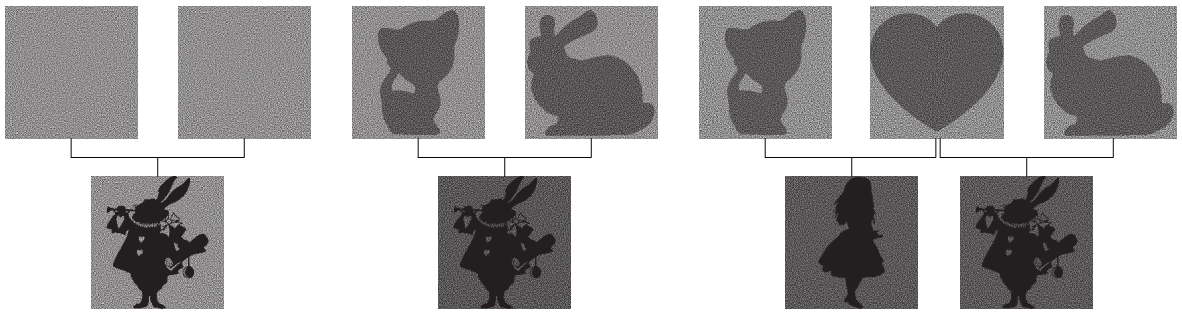


FIGURE 5.1 Left: example of basic $(2,2)$-VCS. Middle: example of $(2,2)$-EVCS. Right: proposed EVCS with common shares. The share images (top row) are stacked (physically superimposed) to decrypt the secret images (bottom row). The proposed method can decrypt multiple images using a common share. Input images are 200×200 pixels and output images are 400×400 pixels.

that shares in the traditional (k,n)-VCS are meaningless, noise-like images. In the extended (k,n)-VCS ((k,n)-EVCS), shares are composed of meaningful images. *Magic Sheet* takes three share images $\{I_1,\ I_c,\ I_2\}$ and two secret images $\{I_{S_1},\ I_{S_2}\}$ as input and computes three output share images $\{S_1,\ S_c,\ S_2\}$ such that I_{S_1} and I_{S_2} can be decrypted by stacking S_1 and S_c and S_c and S_2, respectively. This is similar to $(2,2)$-EVCS; however, in the proposed approach, share S_c is a *common share* used to decrypt two secrets simultaneously, which differentiates our approach from the conventional EVCS (Figure 5.1).

By printing output shares on transparencies, we can physically decrypt secret images. This can be applied to various recreational purposes, such as cooperative games, in which a player with a share looks for a player with the other share to reveal secrets.

Contributions. The primary contributions of this study are as follows.

- We propose an EVCS that uses a unique common share to decrypt multiple secret images by employing a bitwise AND-like operation.
- We demonstrate the effectiveness of the proposed method using binary, grayscale, and color images.
- We demonstrate decryption by superimposing shares printed on transparencies.

Focus. In this work, we aim to apply VCS for entertainment purposes, i.e., hiding and revealing images to evoke a sense of wonder. We do not focus on theoretical aspects, such as security analysis of the proposed method.

5.2 Background

In this section, we briefly describe the traditional (k,n)-VCS and the (k,n)-EVCS for binary, grayscale, and color images.

TABLE 5.1 Construction of $(2,2)$-VCS. An input *white*/*black* pixel p in a share image I_1/I_2 is encoded to a block p' composed of 2×2 subpixels. The stacked result $S_S^{p'}$ representing a pixel color in a secret image I_S^p is obtained by superimposing $S_1^{p'}$ and $S_2^{p'}$.

$I_S^p = White$	Input	I_1^p	White	Black	White	Black
	Output	$S_1^{p'}$				
	Input	I_2^p	White	White	Black	Black
	Output	$S_2^{p'}$				
Stacked: $S_S^{p'} = S_1^{p'} \star S_2^{p'}$						

$I_S^p = Black$	Input	I_1^p	White	White	Black	Black
	Output	$S_1^{p'}$				
	Input	I_2^p	White	Black	White	Black
	Output	$S_2^{p'}$				
Stacked: $S_S^{p'} = S_1^{p'} \star S_2^{p'}$						

5.2.1 (k,n)-VCS

With the (k,n)-VCS, a secret image is decomposed into n shares. The secret is decrypted by the human visual system if k out of n images are physically superimposed; however, any $k-1$ out of n images cannot decrypt the secret and there should be no information leakage [72]. Meaningless random dot patterns are used as shares in the traditional (k,n)-VCS.

Meaningless shares in the traditional (k,n)-VCS are extended to meaningful images. Note that we can construct a (k,n)-EVCS using meaningful shares. Traditional schemes are only used for binary images, and grayscale images can be converted to binary images by halftoning. In addition, many color VCSs have been proposed. By employing a subtractive color model, such as the CMY model, a color image can be decomposed into CMY images and the traditional EVCS can be applied to each C-/M-/Y-channel image. The C-/M-/Y-channel images are then merged into a single color image. Here, the resulting shares do not demonstrate excellent quality, and more sophisticated approaches are available [45]. In addition, developing color EVCSs remains a challenging problem [60].

5.2.2 Pixel Representation

Here, we describe pixel (block) representations in a VCS with a focus on the $(2,2)$-VCS. Each of the following patterns is used to represent a white/black pixel in the output share images.

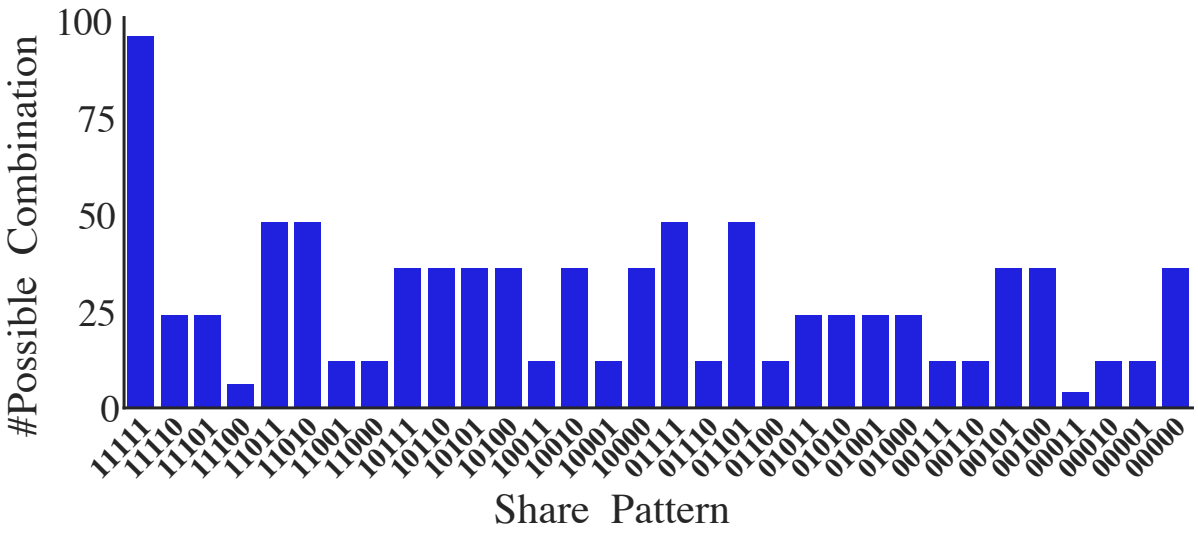


FIGURE 5.2 Number of possible share combinations for each share pattern. A share pattern is represented by five 0-1 digits (0 and 1 indicate black and white pixels, respectively). In order, the five digits represent the block color of share $S_1^{p'}$, common share $S_c^{p'}$, share $S_2^{p'}$, the stacking result of $S_1^{p'} \star S_c^{p'} = S_{S_1}^{p'}$, and $S_c^{p'} \star S_2^{p'} = S_{S_2}^{p'}$. The results are stored in look-up table Lut.

White:

Black:

One of { , , , } is used to represent a *white* pixel in a stacked result of two shares, and we use to represent a *black* pixel. This means that the output shares and decrypted secret have lower contrast compared to the original images.

5.2.3 Example

Construction of a $(2, 2)$-VCS is described in Table 5.1. Given share images I_1, I_2, and a secret image I_S, we can compute subpixels $S_1^{p'}$ and $S_2^{p'}$ in the output shares for each corresponding pixel $I_S^p = p \in I_S$ such that the stacked subpixels of $S_1^{p'}$ and $S_2^{p'}$ $(= S_1^{p'} \star S_2^{p'})$ represents the color of the corresponding pixel I_S^p, where $\star$ is a stacking operator. Note that a bitwise AND-like operator is employed if *white* and *black* are represented by 1 and 0, respectively. We can select $S_1^{p'}$ and $S_2^{p'}$ randomly if the shares represent meaningless noise-like images. For meaningful shares, we must select subpixels such that $S_1^{p'}$ and $S_2^{p'}$ represent I_1 and I_2 as closely as possible.

5.3 EVCS with Common Share

In the proposed method, we use a *common share* to decrypt multiple $(2, 2)$-EVCS images. One secret is decrypted by superimposing one share and the common share, and the other secret is decrypted by superimposing the other share and the common share. In this section, we describe the proposed EVCS with the common share approach.

5.3.1 Share Combinations

First, we compute all possible share combinations. We represent a share combination in the following tabular form:

$\left\{S_1^{p'},\ S_c^{p'},\ S_2^{p'},\ S_{S_1}^{p'},\ S_{S_2}^{p'}\right\}$		
$S_1^{p'}$	$S_c^{p'}$	$S_2^{p'}$
$S_{S_1}^{p'}$	$S_{S_2}^{p'}$	

.

Here, the notations are same as in Section 5.2. The top row represents a share combination comprising subpixels p' of share S_1, common share S_c, share S_2, secret S_{S_1}, and secret S_{S_2} (in this order). For example, in the left table shown below, we have $\left\{S_1^{p'}, S_c^{p'}, S_2^{p'}\right\}$ =$\{$  ,  ,  $\}$. Then, the stacking results would be $S_{S_1}^{p'} = S_1^{p'} \star S_c^{p'} =$, and $S_{S_2}^{p'} = S_c^{p'} \star S_2^{p'} =$, and the resulting combination would be represented as $\left\{S_1^{p'}, S_c^{p'}, S_2^{p'}, S_{S_1}^{p'}, S_{S_2}^{p'}\right\} = \{1, 1, 0, 1, 1\}$, where a *white*/*black* block is represented by $1/0$ and $\star$ is the bitwise AND operator. Similarly, the right table shown below represents $\left\{S_1^{p'}, S_c^{p'}, S_2^{p'}, S_{S_1}^{p'}, S_{S_2}^{p'}\right\}$ = { , , , , } = $\{1, 0, 0, 0, 1\}$.

$\{1, 1, 0, 1, 1\}$		
		
		

$\{1, 0, 0, 0, 1\}$		
		
		

Here, we attempt to compute all possible share combinations, which is performed as follows. First, for each possible combination of $S_1^{p'}$, $S_c^{p'}$, and $S_2^{p'}$, we compute the stacking result of $S_1^{p'} \star S_c^{p'}$ and $S_c^{p'} \star S_2^{p'}$ to obtain $S_{S_1}^{p'}$ and $S_{S_2}^{p'}$, respectively. Then, we store the pattern $\left\{S_1^{p'}, S_c^{p'}, S_2^{p'}, S_{S_1}^{p'}, S_{S_2}^{p'}\right\}$ in look-up table Lut. In Figure 5.2, we plot all possible share patterns and the number of possible combinations. Note that, although we can find symmetric patterns and consider them when reducing the computation time of enumerating all possible combinations to some extent, we compute all possible patterns due to the simplicity of our implementation. The Lut is referenced from the EVCSwithCommonShare procedure to obtain a valid *white*/*black* block arrangement.

5.3.2 Proposed Algorithm

The algorithm is given in Algorithm 3. Given input images (share I_1, common share I_c, share I_2, secret I_{S_1}, and secret I_{S_2}), we assign *white*/*black* to the corresponding output share images S_1, S_c, and S_2, where I_1, I_c, and I_2 have the same width and height, and S_1, S_c, and S_2 have twice the width and height of the input. Note that we apply the algorithm directly if the inputs are binary images. If the inputs are grayscale images, we first apply halftoning using Ostromoukhov's error diffusion algorithm [75] to convert the images to binary images. We describe extension to color input images in Section 5.3.4.

To assign *white*/*black* to the output images, we obtain *white*/*black* colors from the input images. Since the output images are twice the size of the input images, an (x, y) pixel in an

Algorithm 3 EVCSwithCommonShare

1: **for** $y \leftarrow 0 : height$ **do**
2: **for** $x \leftarrow 0 : width$ **do**
3: $\mathcal{P} \leftarrow \{I_1(x,y), I_c(x,y), I_2(x,y), I_{S_1}(x,y), I_{S_2}(x,y)\}$
4: $\mathcal{C} \leftarrow$ RandomSelect(Lut$[\mathcal{P}]$)
5: **for** $k \in \{1, c, 2\}$ **do**
6: AssignColor($S_k(x', y'), \mathcal{C}$)
7: **end for**
8: **end for**
9: **end for**
10: **return** $S_1,\ S_c,\ S_2$

input image corresponds to a (x', y'), 2×2 pixel block in the output image. Then, a block color pattern is constructed from the input colors (Algorithm 3, line 3). We select a valid combination randomly from the Lut by querying the pattern (Algorithm 3, line 4). Finally, the *white*/*black* blocks are assigned to each output image (Algorithm 3, lines 5-7).

5.3.3 Share Optimization

After computing EVCSwithCommonShare, we improve the visual quality of the output shares by applying a share optimization algorithm [58]. For each pixel block, we rearrange the *white/black* such that the resulting shares have better visual quality while the remaining stacking results remain unchanged. The results are shown in Figure 5.3. Note that the algorithm is described in detail in the Appendix C.

5.3.4 Extension to Color Images

The proposed method can be extended to a color EVCS. As described in Section 5.2, we decompose the given images into C-/M-/Y-channel images. Then, we apply Ostromoukhov's error diffusion method [75] to convert the images to halftone images. Note that we initially applied the structure-aware error diffusion method [16]; however, this method did not show significant improvement. Therefore, we used Ostromoukhov's method, which is simple and faster. Contrast-aware halftoning [53] or other state-of-the-art error diffusion approaches [61] can be employed for faster and better image preparation. Regarding the color EVCS, although it is not obvious that the proposed method can be incorporated, we can improve the visual quality of the output shares using a more sophisticated approach [45]. However, the color EVCS remains a challenging and open problem [60].

5.4 Results

We applied the proposed method to binary, grayscale, and color images. The results for binary images are shown in Figure 5.1. Figures 5.3 and 5.9 show the results for grayscale images, and Figures 5.4, 5.6, and 5.10 show the color EVCS results.

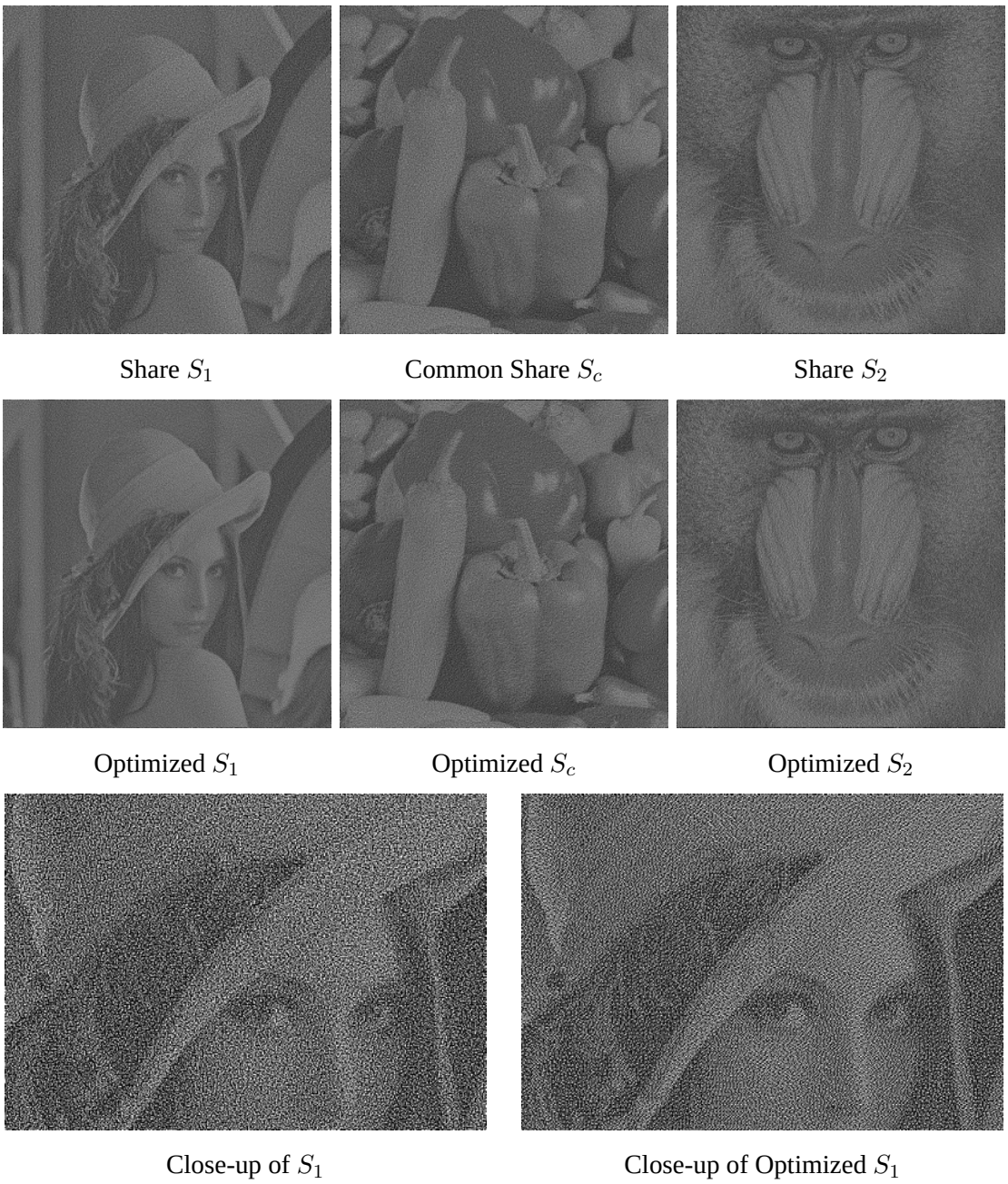


FIGURE 5.3 Results of proposed EVCS. Input images are 512×512 pixels and outputs are 1024×1024 pixels. Top: unoptimized results. Middle: optimized results. Bottom: close-ups of unoptimized and optimized S_1.

We generated output results printed on transparencies using a Canon ImageRUNNER ADVANCE C3330i (Figures 5.5 and 5.6 (bottom)). To improve contrast, we printed each share on two transparencies and superimposed them. As shown in Figures 5.5 and 5.6 (right), the *crosstalk* effect is observable in the decrypted images due to the influence of the transmitted background light; however, the secret images are revealed successfully, and the secret text is readable.

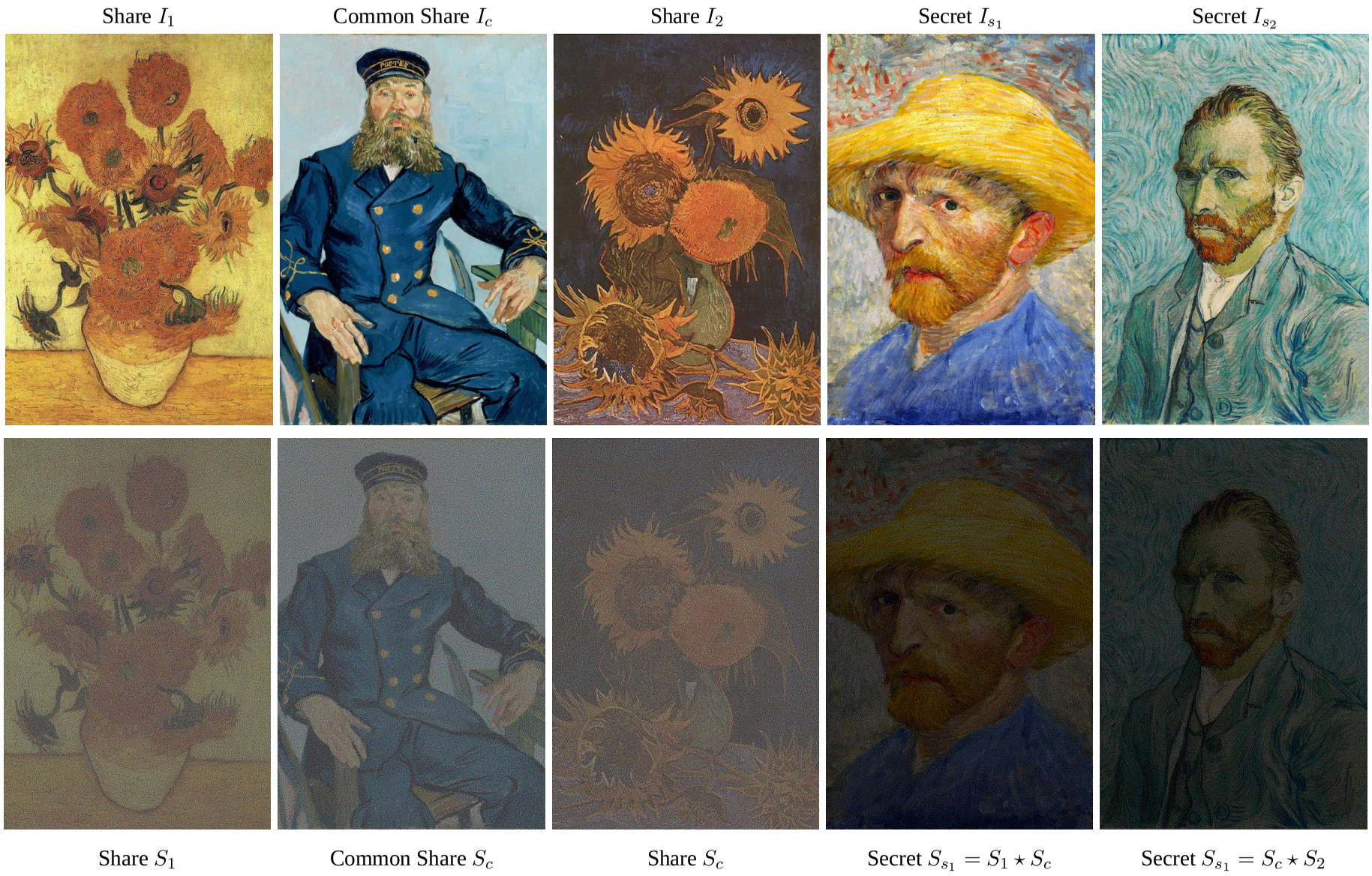


FIGURE 5.4 Top: input images; original images are 463×680 pixels. Bottom: proposed color EVCS results; the two portraits on the right are the computed superimposition results of a common share and the corresponding share.

5.4.1 Visual Comparison

Figure 5.3 compares the source input images and output images with/without optimization (Section C.1). For example, as can be seen in the optimized result shown in Figure 5.3 (Bottom), the structures are better preserved and textureless areas are smoother. Although the quality improvements are not always obvious, the results demonstrate higher quality relative to both visual quality and quantitative evaluations.

5.4.2 Numerical Comparison

We followed previous approaches to evaluate the results quantitatively [16,54,58,80]. Here, we computed the mean structural similarity index measure (MSSIM) [102] and the peak signal-to-noise ratio (PSNR) to compare the input and output images with/without optimization. We measured the average MSSIM and PSNR over 10 runs. Note that the color results were converted to grayscale images for comparison.

The MSSIM and PSNR results are shown in Tables 5.2 and 5.3, respectively. Since the algorithm outputs binary images, we first applied the Gaussian blur with a kernel size of 5×5 and compared them to the input source images. Note that the input source images were scaled to the same size as the output images. Generally, the results with optimization achieved higher MSSIM and PSNR values.

Although both MSSIM and PSNR achieved higher values after optimization, in the case shown in Figure 5.1 (binary image) in Tables 5.2 and 5.3, the visual quality was worse due to information leakage, as shown in Figure 5.7. In other words, there is a tradeoff between visual quality and the MSSIM and PSNR values. Note that such visual artifacts resulting from the optimization procedure were only observed for binary images, i.e., the optimized grayscale and color images did not demonstrate such artifacts.

This is because, in the binary cases, the figures (contents) in the images have no textures, i.e., these images consist of black and white regions with boundaries. Therefore, in a share image, boundaries (from the other image) in the textureless regions are noticeable in the optimized share images. In contrast, we did not notice such information leakages in the optimized share images in the grayscale and color image results (Figures 5.9 and 5.10) because these images have textures and a busy appearance compared to the binary images.

However, since optimization is an optional process, we can simply omit this procedure when we apply the proposed method to binary images, whereas optimization can be applied to grayscale and color images to improve visual quality. To alleviate and improve optimization, a different objective function could be introduced rather than the current mean-squared-error and MSSIM-based function (Equation (C.1) in the Appendix).

TABLE 5.2 Comparison of MSSIM between original input shares and output shares with/without optimization. We applied Gaussian blur with a kernel size of 5×5 to the output images and compared them to the source images.

	Figure 5.1 right (binary)			Figure 5.3 (grayscale)			Figure 5.4 (color)			Figure 5.9 (grayscale)			Figure 5.10 (color)		
	S_1	S_c	S_2	S_1	S_c	S_2	S_1	S_c	S_2	S_1	S_c	S_2	S_1	S_c	S_2
w/o Opt.	0.113	0.228	0.091	0.194	0.186	**0.225**	0.297	0.296	0.333	0.188	0.206	0.239	0.321	0.319	0.314
w/ Opt.	**0.450**	**0.808**	**0.302**	**0.246**	**0.202**	0.223	**0.312**	**0.328**	**0.336**	**0.228**	**0.231**	**0.245**	**0.369**	**0.346**	**0.321**

TABLE 5.3 Comparison of PSNR between original input shares and output shares with/without optimization. We applied Gaussian blur with a kernel size of 5×5 to the output images and compared them to the source images.

	Figure 5.1 right (binary)			Figure 5.3 (grayscale)			Figure 5.4 (color)			Figure 5.9 (grayscale)			Figure 5.10 (color)		
	S_1	S_c	S_2	S_1	S_c	S_2	S_1	S_c	S_2	S_1	S_c	S_2	S_1	S_c	S_2
w/o Opt.	7.45	10.24	8.28	14.18	13.87	**14.16**	12.93	13.23	18.70	15.20	14.05	13.38	15.69	14.45	13.64
w/ Opt.	**7.55**	**10.40**	**8.38**	**14.38**	**13.94**	14.14	**12.97**	**13.27**	18.70	**15.40**	**14.15**	**13.39**	**15.78**	**14.50**	13.64

S_1 S_c S_2 $S_1 \star S_c$ $S_c \star S_2$

FIGURE 5.5 Output shares printed on transparencies.

S_1 S_c S_2 $S_1 \star S_c$ $S_c \star S_2$

FIGURE 5.6 Top: computed color EVCS results. The original images are 480×360 pixels. Bottom: output shares printed on transparencies. The two sheets on the right are the superimposition results of a common share and the corresponding share on the left. The input images are listed in Figure C.1 (top) in the Appendix.

Share S_1 Common Share S_c Share S_2

FIGURE 5.7 Information leakage on images after optimization applied to output shares with binary images as input (Figure 5.1 right).

5.4.3 Computation Time

We measured computation time using an Intel Core i5 @ 2.9 GHz personal computer with 16 GB RAM. We implemented our algorithms in C++ using OpenCV. The computation of all possible share combinations to prepare the look-up table Lut (Section 5.3) took approximately 0.5 ms. Table 5.4 summarizes the computation times for the EVCSwithCommonShare procedure with and without optimization. As can be seen, the optimization process is computationally expensive because calculating the MSSIM for each loop is computationally expensive, and this may not be worth the cost.

Rather than using the MSSIM, we can use another structure similarity measure; however, this will be the focus of future work. For color images, we simply compute EVCSwithCommonShare for each CMY channel for color images, which can be parallelized, or we can employ a more sophisticated approach for a color EVCS.

TABLE 5.4 Computation time of EVCSwithCommonShare with/without optimization.

	Input size	Without Opt.	With Opt.
Figure 5.1 right (binary)	200×200	0.034 sec	75 sec
Figure 5.3 (grayscale)	512×512	0.232 sec	482 sec
Figure 5.4 (color)	463×680	0.785 sec	1738 sec
Figure 5.9 (grayscale)	600×375	0.198 sec	431 sec
Figure 5.10 (color)	600×375	0.570 sec	1238 sec

5.4.4 $(2, n)$-EVCS

In Section 5.3, we described the proposed method for the (k, n)-EVCS where $k = 2$ and $n = 2$. Due to its simplicity and scalability, it is straightforward to extend the proposed method

to greater n values. The $(2, 4)$-EVCS and $(2, 5)$-EVCS computed by the proposed method are shown in Figure 5.8.

5.5 Conclusion and Future Work

We have presented *Magic Sheet*, a VCS that uses common shares (a type of universal share). With the common shares, we can decrypt multiple secrets simultaneously depending on a given share. *Magic Sheet* hides secret images in meaningful shares and can be applied to binary, grayscale, and color images. Since the proposed method is based on a bitwise AND-like operation, we can physically realize shares on transparencies and retrieve secrets by superimposing these shares.

Future Work. In this study, we applied the proposed method to a pair of $(2, 2)$-EVCSs for both grayscale and color images, and we demonstrated a $(2, n)$-EVCS, where $n > 2$. Note that the proposed method can also be applied to pairs of general (k, n)-EVCSs. In addition, although we employed 2×2 pixel expansion in the proposed method, it would be interesting to investigate an algorithm with no pixel expansion to achieve higher contrast while maintaining visual quality. In addition, the proposed method employs a simple optimization algorithm to improve the visual quality of the output shares, and this algorithm requires significant time to achieve optimization. Thus, in the future, we will investigate a faster approach.

With the color EVCS, we can observe some information leakage in the stacked result of S_1 and S_2, i.e., $S_1 \star S_2$ in Figure 5.10. Although we have focused more on the entertainment purposes of the proposed method and did not focus on security aspects, it is important to analyze such aspects quantitatively.

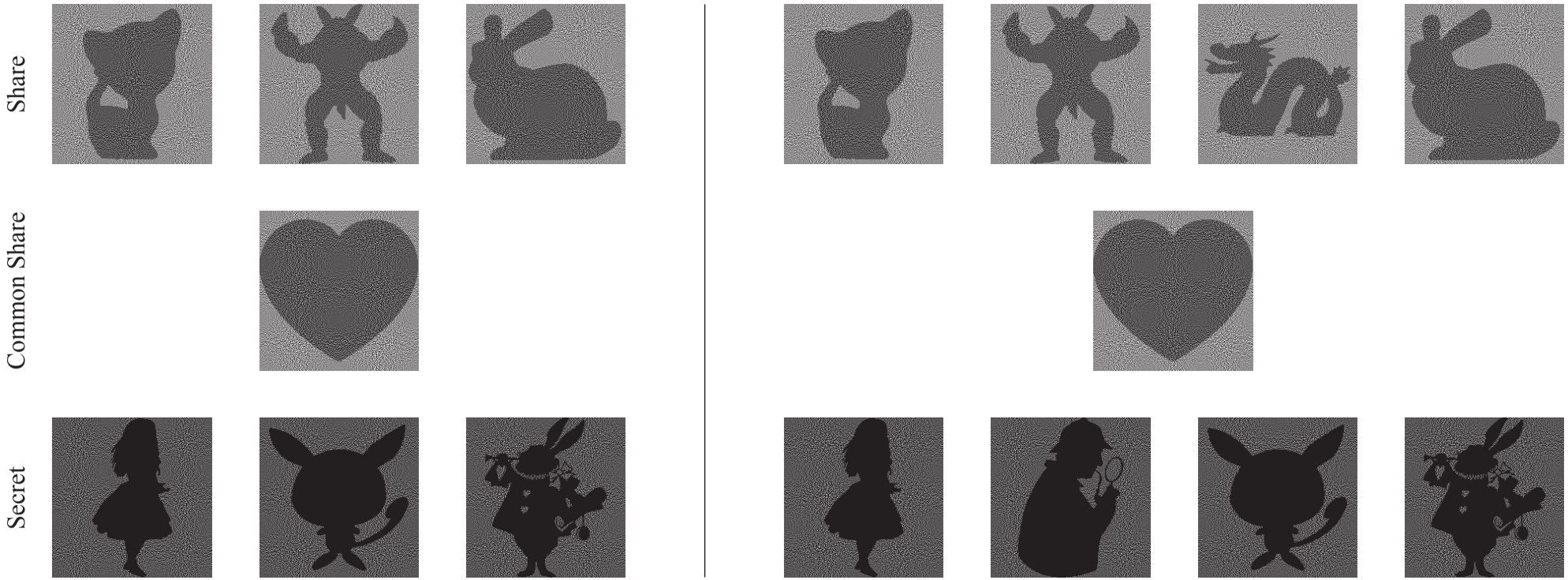


FIGURE 5.8 Proposed method applied to $(2, 4)$-EVCS (left) and $(2, 5)$-EVCS (right). Input images are 200×200 pixels and output images are 400×400 pixels.

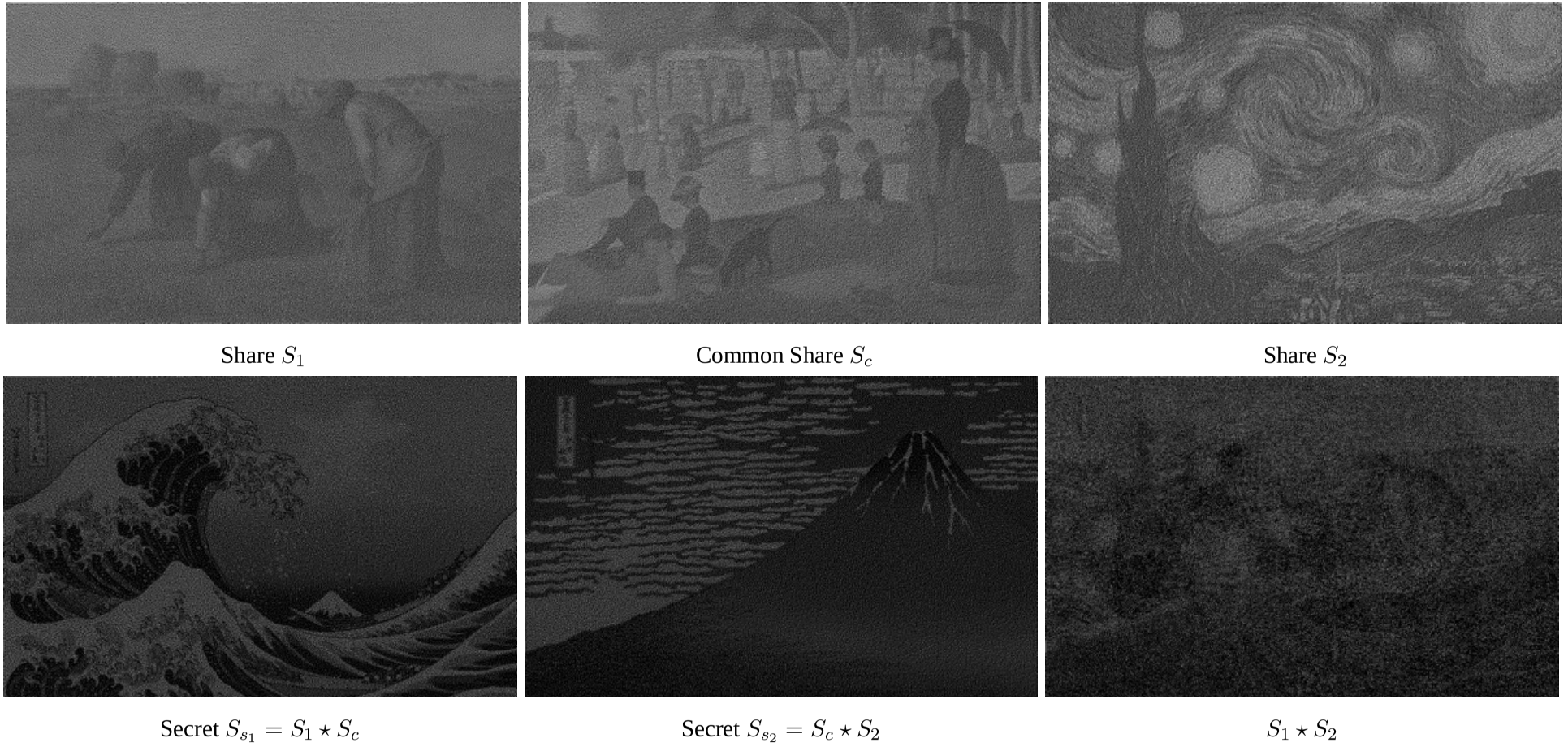
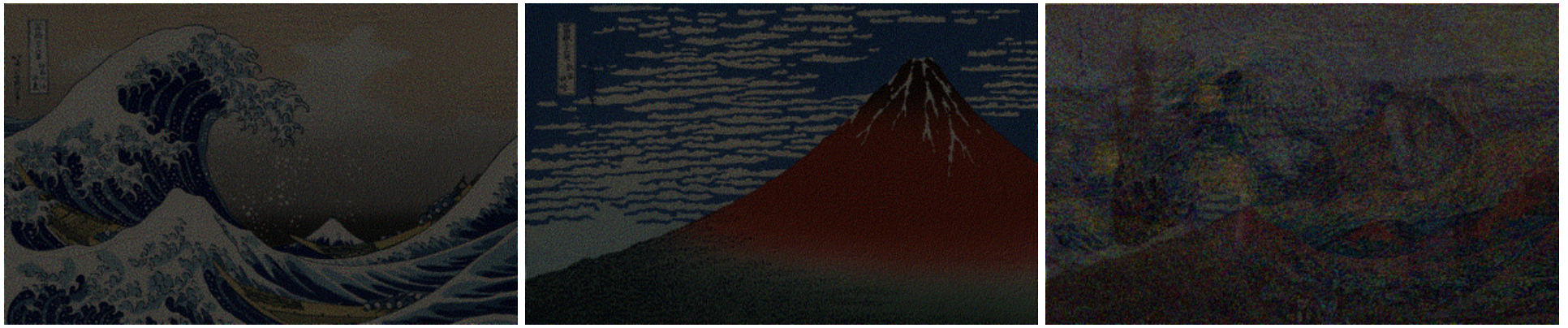


FIGURE 5.9 EVCS results for grayscale images. The original images are 1200×750 pixels. The input images are listed in Figure C.1 (bottom) in the Appendix.

Share S_1 | Common Share S_c | Share S_2



Secret $S_{s_1} = S_1 \star S_c$ | Secret $S_{s_2} = S_c \star S_2$ | $S_1 \star S_2$

FIGURE 5.10 EVCS results for color images. The original images are 1200×750 pixels. The input images are listed in Figure C.1 (bottom) in the Appendix.

CHAPTER 6

Feedback from an Expert

In each chapter, we validated the proposed method by conducting user study and performance analysis. In addition, we asked a professional designer whether the proposed tools are sufficient for practical purposes so as to support him in his work.

We conducted an informal interview with him, which lasted for about two hours. First, we briefly introduced the proposed tool, i.e., the functionalities of the tool, and then asked several questions regarding the functionalities, usefulness in actual design scenarios, as well as, free comments.

6.1 Comments on Discrete Element Textures

Functionality. He commented, *"For functionality, it's a good starting point for non-professional as it gives a good solid base for wallpaper or background composition. For professional it can be a good tool to quickly generate different proposal rapidly to test and verify different potential composition."* This is actually our aim in this thesis; to support users by the proposed tools. In particular, he preferred the flow editing functionality of the tool as indicated by this statement: *"Most of the time, it's cumbersome to edit that–arrangement of numerous elements–by hand."*

Usefulness. For the usefulness, he commented, "Sometimes, graphic designer doesn't have much time to work on projects, so reducing the production time during the experimentation phases can be of good help." This is one of the requirements of the proposed tool (R1 in Section 1.3). His comment shows that the requirement captured the practical setting.

Free Comments. *"One improvement could be on modifying the size of some of the elements,"* he commented. Although we did not mention the limitations of the proposed tool during the interview, he mentioned the very limitation, which confirms our submission that the future research direction is promising.

6.2 Comments on Discrete Color Palettes

Functionality. He commented, *"Color is really important in any design, color balance can be difficult to achieve, especially if attached to a specific style like pastel or warm colors."* This is the comment referring to the customization of the functionality of the proposed model. He

also commented that *"Not everybody has an education in art and color sensitivity, so allowing the user to choose a pre-selected style and allowing for customization is a good tool for anyone. A professional designer doesn't have much time, so being able to rapidly test out different color combination is a great time saver,"* he continued.

Usefulness. He also commented about its usefulness in practical scenarios: *"For a professional creating a design identity for a client, maybe a pack for a website, business card and graphic identity for a logo,"* and *"For an amateur, if you are trying to design a holiday card it can be a good starter to find a good color balance based on a defined style, like Christmas for example."* He emphasized on the importance of *"easier access to modification, and also easier access to iterative design."*

Free Comments. He mentioned that the proposed tool focuses on solid color palettes. He suggested we should also consider non-solid palettes, i.e., *"use of gradient could be a good addition,"* which would improve its practical usage in actual scenarios.

6.3 Comments on Discrete Element Patterns

Functionality. Since the proposed method is very specific, he had no comments on the specific functionalities of the proposed tool.

Usefulness. He commented that the results generated by the proposed method are of interest: *"very impressive result, surprising use of image combination,"* and suggested several practical applications listed below:

- Online security and robot verification
- Commercial campaign, e.g., hiding a winning message on a bottle of soda for a campaign
- Kids party or any kind of secret message, on cereals package for example
- Encrypted arts

"This is more like an interesting experience," he commented.

Free Comments. He suggested several insightful ways for the fabrication, e.g., using functional inks.

6.4 General Comments

We also asked him for comments on the proposed tools from a comprehensive point of view. He commented that the proposed tool for discrete element textures and discrete color palettes *"do help visualize faster nice and balanced images,"* and *"allows less able amateur to generate personal content, for their own website or cards or something"* as *"you can quickly generate proposal"* using the proposed tools.

As regard to the practical aspect, he commented: *"If you design patterns, and if it's the product to sell, then it's good for a first stage exploration"* and *"It could be part of a final product if used for procedural generation of patterns on background elements for a poster."*

In summary, he concluded with a comment: *"Anything that would need speed will be applicable for this tool, may it be quick creation of proposal design or the design of a poster with intricate elements where some require to be wrapped with patterns."*

Overall, he repeated that the speed of *computational performance* is very important because professional designers have to create designs for clients and the time to create designs for each client is limited in most cases. In addition, the speed is also important in the cases where time is not limited since users can repeat trials and errors to create desired designs, which is a tedious and time-consuming stage of the design exploration and to be streamlined by the computational design tools.

CHAPTER 7

Conclusion and Future Work

We conclude this thesis by outlining the goals we have achieved and also discuss future work.

7.1 Summary

The motivation of this thesis is to support users' intellectual and creative works, and the goal is to provide computational tools for discrete element layouts by focusing especially on its geometric\non-geometric aspects: since the arrangement of discrete elements affects human visual perception, which is difficult to formulate because of the multimodality of human preference for visual designs. These challenges consequently prevent researchers and developers from providing tools for supporting users' layout design exploration. To tackle these problems, we employed a machine learning approach or assumed several constraints and formulated the design goal as an objective function. Furthermore, we have proposed computational design tools for abstract elements in a general layout design domain by parameterizing the design space. While we mathematically formulated the spatial uniformity in the discrete element texture tool and the image quality in the discrete element pattern tool, the human color preferences we dealt with in the discrete color palettes hardly could be formulated mathematically. However, the L1 regularization of the LASSO–*one of the sparse modeling approach* we employed in the proposed method allows us to parameterize the multimodality of the human color preferences.

In summary, we have tackled a design problem of abstract element layouts in a general domain by parameterizing the design space while putting the human visual perception or human visual preferences into consideration. The proposed computational (parametric) design tools support users' design exploration of spatial element arrangements and color combination easily and efficiently. Even if the objective function is multimodal such as human color aesthetic preferences, we have shown that it can be formulated by employing a sparse modeling approach. Specifically, we have proposed three computational design tools summarized below.

First, we have provided a computational design tool for creating a spatial uniform distribution of discrete elements by applying multi-class anisotropic blue noise sampling. The multi-class blue noise distribution allows a user to create a spatially uniform discrete element distribution. By approximating an element as an ellipse or a disk, the collision detection can be checked efficiently, which makes the distribution generation at an interactive rate. Besides, the

user can edit elements layout by directly manipulating an element or editing the underlying flow field to control elements orientation in a macroscopic way. We have demonstrated the proposed method by applying it to create various textures composed of discrete elements.

Second, using a machine learning approach we have provided a computational design tool for creating discrete color palettes. We trained the proposed palette rating model using a large dataset including general human color palette preferences. Using the palette rating model, we have proposed an additional compatible color suggestion method by efficiently exploring a huge color space. We have also provided a way to customize (specialize) the palette rating model to a specific color tone (e.g., *pastel*). We have demonstrated the proposed method by applying it to 2D pattern coloring, photo recoloring, product color design exploration, and room decoration.

Third, we have provided a computational design tool for creating discrete element patterns using an EVCS with common shares. Our tool is the first to possess all of the following characteristics:

- A method that can generate meaningful shares, i.e., composed of a structured pattern, rather than a noisy unstructured pattern.
- A method that can generate a common share used as a *key* to decrypt multiple secrets.
- The secrets can be decrypted by superimposition of shares printed on transparencies.
- A method that can be applied to binary, grayscale, and color images
- A method that optimizes the visual quality of shares.

We expect that this method should be employed in entertainment applications since patterns created by it can evoke *a sense of wonder* in people as we have demonstrated in Section 5.4.

In the first method, we employed a procedural approach to generate the spatially uniform distribution. This is because spatial uniformity can be formulated as an objective function. Therefore, we have employed a procedure that generate such a distribution and analyzed the quality of the distribution using point set analysis tools. In contrast to the first method, we employed a machine learning approach in the second method. This is because people's aesthetic preference is difficult to formulate due to the nature of its multimodality. A machine learning approach can be effective in this situation, since we can train a model using a dataset composed of ratings based on human aesthetic preferences. By alternating\customizing the dataset, a model can be trained with a bias. Also, the color palette preference model can be trained for not only specific color such as *pastel* or *retro*, but also for *Japanese*, *French*, or even just for *myself*. In the third method, we employed meta-heuristic approach to solve a combinatorial optimization problem. It is worth noting that the visual quality of a pattern can be formulated as an objective function. However, in this case, solving the combinatorial optimization problem may be time-consuming due to the nature of the combinatorial complexity. Therefore, a meta-heuristic approach can work well in this situation.

Both first and second tools can be used for general spatial layouts and color designs, respectively, since these tools handle abstract data structures such as proxy polygons or color itself. In contrast to the first and second tools, the third tool is more application specific, i.e., it provides a tool for creating EVCS with common shares, and it is difficult to apply for general discrete

element pattern generation purpose. However, we are positive that the methodology we have proposed for the constrained discrete element pattern optimization can be applied widely.

7.2 General Discussions

In Section 1.3, we enumerated the requirements for the proposed computational design tools and the corresponding validations to be assessed. In each chapter that describes the proposed tool, we have performed quantitative and qualitative evaluations including computational performances analysis (V1) to check whether the proposed methods can generate design suggestions at an interactive rate so as to efficiently support users' trial-and-error design process (R1), and user studies (V3) to check if the proposed methods provide better functionalities in comparison to the state-of-the-art equivalents. We have also demonstrated various applications of the proposed methods (V2) to demonstrate their versatility (R2). Finally, we asked a professional designer to assess the usefulness of the proposed tools (V3). As described in Section 6.4, he emphasized on the importance of the computational performance of the design tools in practical design scenarios. Hence, we have set up several requirements including the computational performance (R1), the usefulness (which is confirmed by the professional designer's emphasis), as well as, V1. The requirements R2 and R3 refer to the applicability of the proposed tools, the usefulness of the proposed tools are also validated by the comments of the professional designer on each tool, as well as, V2 and V3. However, he also mentioned several improvements of the proposed tools, which are important for practical usage and suggested some fruitful future direction to be pursued.

In Figure 1.4, we showed the focal areas of the proposed methods, i.e., abstract element arrangements in a general layout domain. By providing three different computational design tools and focusing on the element's geometric\non-geometric aspects, we tackled some challenging layout problems including the human visual perception and the human aesthetic preferences as its important visual factors. In the discrete element texture design tool, we approximated an element as a proxy shape. Such abstraction allows us to deal with an arbitrary shape of elements. In the discrete color palette design tool, we customized the palette aesthetic rating model to any type of color images, by which we can capture the multimodality of the human color aesthetic preferences in a parametric manner. In addition, we trained an aesthetic rating model by a machine learning approach. Hence, we have provided a generalized color combination design tool. Consequently, users can explore the layout design space easily and efficiently. Besides, owing to the abstract and general purpose nature of the proposed tools, users can use the tools at the early stage of the design exploration.

7.3 Contribution to Knowledge Science

As we stated in Chapter 1.1, the main motivation of this thesis is to support users' intellectual and creative work by providing computational design tools for visual design, especially in discrete element layouts. The computational design tools proposed in this thesis can not only

generate various design candidates efficiently, but also produce more design solutions which cannot be obtained from one's idea and intuition. Moreover, these tools have the potential to open up a new frontier of various design fields. In this thesis, we focused on human-related objectives including human visual perception and aesthetic preferences. At the present moment, supporting tools for visual designs that can optimize human-related design specifications are limited. Aside from we covering several portions of the visual design field, the proposed tools in this thesis solve several challenging problems with human-related objectives, and we have demonstrated their various applications; hence the proposed methods can be applied to several other problems and design fields. From the perspective of knowledge science, our computational design tools provide a foundation for users' design exploration of discrete element layouts.

7.4 Limitations and Future Work

Although the tools proposed in this thesis satisfied all the requirements validations discussed here, we have not fully covered the discrete element layout domains shown in Figure 1.4. Areas such as the gradient regions between the darker gray regions (covered by previous researches) and regions we have covered (orange, green, and purple areas), are still remaining.

In addition, although we have covered several regions mostly by the computational design tools for discrete element textures and discrete color palettes, we have not provided an integrated tool that can simultaneously deal with both geometric and non-geometric aspects.

For example, in regard to the tool for the discrete element textures, we generated the element distributions in a spatially uniformly manner. However, the spatial uniformity differs from *visual uniformity* as we have to take into account both the spatial arrangement and the appearance of the elements simultaneously so as to generate the visually uniform distribution. Accomplishing this task entails solving a joint optimization problem.

In regard to the tool for discrete element patterns, utilizing an energy function composed of structure and coherence terms, which captures the spatial (neighboring) relationships, we jointly optimized the sub-pixel (element) arrangement to improve the visual qualities (appearance) of shares. Although this topic focuses on a very specific application, the proposed framework can be employed for the joint optimization of spatial and appearance of elements in various design areas.

Evaluation of Computational Design Tools. Computational design tools support users' design exploration. The tool not only supports in the provision of useful methods for efficient design creation but also provides a user with new unimaginable ideas. Although an interview can be conducted for the qualitative evaluation of the tools, it is difficult to quantitatively assess whether they could successfully provide the user with new ideas; which raises an important question of this research domain, and it is one of the fruitful future directions to be explored. It is also important to observe how users employ computational design tools: When do they use the tools? At what stage of the design exploration do they use the tools? How do they collaborate with the tools?

Finally, we also have to think of what is the ideal design support in proposing computational design tools. For example, if a user is satisfied with a design suggestion generated by the design tool, he\she may stop further design exploration. Therefore, appropriate support is needed to bring out the user's potential. The tools proposed in this thesis aim at supporting general layout design and first stage exploration, as well as the creation of new opportunities for further design improvements. However, we have not proved that our design tools appropriately provide these opportunities. The quantitative assessment of whether the proposed tools provide sufficient rooms for a user's further design exploration is an important research question and will open up a new frontier in the computational design field.

Bibliography

[1] Alexa, M., and Matusik, W. Reliefs as images. *ACM Trans. Graph. 29*, 4 (July 2010), 60:1–60:7.

[2] AlMeraj, Z., Kaplan, C. S., and Asente, P. Patch-based geometric texture synthesis. In *Proceedings of the Symposium on Computational Aesthetics* (New York, NY, USA, 2013), CAE '13, ACM, pp. 15–19.

[3] Autodesk. http://blogs.autodesk.com/netfabb/2018/06/21/design-for-additive-manufacturing-connecting-additive-w-subtractive/, Accessed in Nov 2018.

[4] Bao, F., Yan, D.-M., Mitra, N. J., and Wonka, P. Generating and exploring good building layouts. *ACM Trans. Graph. 32*, 4 (July 2013), 122:1–122:10.

[5] Baran, I., Keller, P., Bradley, D., Coros, S., Jarosz, W., Nowrouzezahrai, D., and Gross, M. Manufacturing layered attenuators for multiple prescribed shadow images. *Comput. Graph. Forum 31*, 2pt3 (May 2012), 603–610.

[6] Barla, P., Breslav, S., Thollot, J., Sillion, F., and Markosian, L. Stroke pattern analysis and synthesis. In *Computer Graphics Forum (Proc. of Eurographics 2006)* (2006), vol. 25.

[7] Bermano, A., Baran, I., Alexa, M., and Matusk, W. Shadowpix: Multiple images from self shadowing. *Comput. Graph. Forum 31*, 2pt3 (May 2012), 593–602.

[8] Birren, F., and Cleland, T. *A grammar of color: a basic treatise on the color system of Albert H. Munsell*. Van Nostrand Reinhold Co., 1969.

[9] Bnn. *Petit Pattern Book - Pop & Modern (Bnn Pattern Book Series)*. Ram Distribution, 2007.

[10] Bridson, R. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches* (New York, NY, USA, 2007), SIGGRAPH '07, ACM.

[11] Bychkovsky, V., Paris, S., Chan, E., and Durand, F. Learning photographic global tonal adjustment with a database of input/output image pairs. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2011), CVPR '11, IEEE Computer Society, pp. 97–104.

[12] Bylinskii, Z., Kim, N. W., O'Donovan, P., Alsheikh, S., Madan, S., Pfister, H., Durand, F., Russell, B., and Hertzmann, A. Learning visual importance for graphic designs and data visualizations. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2017), UIST '17, ACM, pp. 57–69.

[13] Cao, Y., Chan, A. B., and Lau, R. W. H. Automatic stylistic manga layout. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 141:1–141:10.

[14] Cao, Y., Lau, R. W. H., and Chan, A. B. Look over here: Attention-directing composition of manga elements. *ACM Trans. Graph. 33*, 4 (July 2014), 94:1–94:11.

[15] Chang, H., Fried, O., Liu, Y., DiVerdi, S., and Finkelstein, A. Palette-based photo recoloring. *ACM Trans. Graph. 34*, 4 (July 2015), 139:1–139:11.

[16] Chang, J., Alain, B., and Ostromoukhov, V. Structure-aware error diffusion. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 162:1–162:8.

[17] Chen, G., Esch, G., Wonka, P., Müller, P., and Zhang, E. Interactive procedural street modeling. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 103:1–103:10.

[18] Chen, K., Xu, K., Yu, Y., Wang, T.-Y., and Hu, S.-M. Magic decorator: Automatic material suggestion for indoor digital scenes. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 232:1–232:11.

[19] Chen, Z., Yuan, Z., Choi, Y.-K., Liu, L., and Wang, W. Variational blue noise sampling. *IEEE Transactions on Visualization and Computer Graphics 18*, 10 (Oct. 2012), 1784–1796.

[20] Chosson, S. M., and Hersch, R. D. Beating shapes relying on moirÉ level lines. *ACM Trans. Graph. 34*, 1 (Dec. 2014), 9:1–9:11.

[21] Chu, H.-K., Hsu, W.-H., Mitra, N. J., Cohen-Or, D., Wong, T.-T., and Lee, T.-Y. Camouflage images. *ACM Trans. Graph. 29*, 4 (July 2010), 51:1–51:8.

[22] Cohen-Or, D., Sorkine, O., Gal, R., Leyvand, T., and Xu, Y.-Q. Color harmonization. *ACM Trans. Graph. 25*, 3 (July 2006), 624–630.

[23] Color, A. https://color.adobe.com/, Accessed in Oct 2018.

[24] COLOURlovers. https://www.colourlovers.com/, Accessed in Oct 2018.

[25] COLOURlovers. P a s t e l c u t i e s. https://www.colourlovers.com/group/P_a_s_t_e_l_C_u_t_i_e_s, Accessed in Oct 2018.

[26] COLOURlovers. Pastel_l♥vers. https://www.colourlovers.com/group/Pastel_L%E2%99%A5vers, Accessed in Oct 2018.

[27] COLOURlovers. Retro. https://www.colourlovers.com/group/Retro, Accessed in Oct 2018.

[28] COLOURlovers. Retropolis. https://www.colourlovers.com/group/RETROpolis, Accessed in Oct 2018.

[29] Cook, R. L. Stochastic sampling in computer graphics. *ACM Trans. Graph. 5*, 1 (Jan. 1986), 51–72.

[30] Dalal, K., Klein, A. W., Liu, Y., and Smith, K. A spectral approach to npr packing. In *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering* (New York, NY, USA, 2006), NPAR '06, ACM, pp. 71–78.

[31] Dunbar, D., and Humphreys, G. A spatial data structure for fast poisson-disk sample generation. *ACM Trans. Graph. 25*, 3 (July 2006), 503–508.

[32] Fang, W., and Lin, J. Universal share for the sharing of multiple images. *Journal of the Chinese Institute of Engineers 30*, 4 (2007), 753–757.

[33] Feng, T., Yu, L.-F., Yeung, S.-K., Yin, K., and Zhou, K. Crowd-driven mid-scale layout design. *ACM Trans. Graph. 35*, 4 (July 2016), 132:1–132:14.

[34] Fried, O., DiVerdi, S., Halber, M., Sizikova, E., and Finkelstein, A. Isomatch: Creating informative grid layouts. *Comput. Graph. Forum 34*, 2 (May 2015), 155–166.

[35] Goethe, J. v. W. *Goethe's Color Theory*. Van Nostrand Reinhold Co., 1971.

[36] Gomez-Nieto, E., Casaca, W., Motta, D., Hartmann, I., Taubin, G., and Nonato, L. G. Dealing with multiple requirements in geometric arrangements. *IEEE Transactions on Visualization and Computer Graphics 22*, 3 (March 2016), 1223–1235.

[37] Harrower, M., and Brewer, C. A. ColorBrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal 40*, 1 (June 2003), 27–37.

[38] Hiller, S., Hellwig, H., and Deussen, O. Beyond stippling — methods for distributing objects on the plane. *Computer Graphics Forum 22*, 3 (2003), 515–522.

[39] Hillier, F. S., and Lieberman, G. J. *Introduction to Mathematical Programming*. McGraw-Hill, 1990.

[40] Hou, Y. Visual cryptography for color images. *Pattern Recognition 36*, 7 (2003), 1619–1629.

[41] Hurtut, T., Landes, P.-E., Thollot, J., Gousseau, Y., Drouillhet, R., and Coeurjolly, J.-F. Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2009), NPAR '09, ACM, pp. 51–60.

[42] Ijiri, T., Mech, R., Igarashi, T., and Miller, G. S. P. An example-based procedural system for element arrangement. *Comput. Graph. Forum 27*, 2 (2008), 429–436.

[43] Itten, J. *The Art of Color: The Subjective Experience and Objective Rationale of Color*. A VNR book. Wiley, 1974.

[44] Jiang, M., Zhou, Y., Wang, R., Southern, R., and Zhang, J. J. Blue noise sampling using an sph-based method. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 211:1–211:11.

[45] Kang, I., Arce, G. R., and Lee, H. K. Color extended visual cryptography using error diffusion. *IEEE Transactions on Image Processing 20*, 1 (Jan 2011), 132–145.

[46] Kim, H.-R., Yoo, M.-J., Kang, H., and Lee, I.-K. Perceptually-based color assignment. *Computer Graphics Forum 33*, 7 (2014), 309–318.

[47] Kita, N., and Miyata, K. Interactive procedural modeling of pebble mosaics. In *SIGGRAPH Asia 2011 Sketches* (New York, NY, USA, 2011), SA '11, ACM, pp. 35:1–35:2.

[48] Kobayashi, S. *Color Image Scale*. Kodansha International, 1991.

[49] Labelle, F., and Shewchuk, J. R. Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry* (New York, NY, USA, 2003), SCG '03, ACM, pp. 191–200.

[50] Lagae, A., and Dutré, P. A procedural object distribution function. *ACM Trans. Graph. 24*, 4 (2005), 1442–1461.

[51] Lagae, A., and Dutré, P. A comparison of methods for generating poisson disk distributions. *Computer Graphics Forum 27*, 1 (2008), 114–129.

[52] Landes, P.-E., Galerne, B., and Hurtut, T. A shape-aware model for discrete texture synthesis. *Computer Graphics Forum (Proc. EGSR) 32* (2013).

[53] Li, H., and Mould, D. Contrast-aware halftoning. *Computer Graphics Forum 29*, 2 (2010), 273–280.

[54] Li, H., Wei, L.-Y., Sander, P. V., and Fu, C.-W. Anisotropic blue noise sampling. In *ACM SIGGRAPH Asia 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH ASIA '10, ACM, pp. 167:1–167:12.

[55] Li, X., Zhao, H., Nie, G., and Huang, H. Image recoloring using geodesic distance based color harmonization. *Computational Visual Media 1*, 2 (2015), 143–155.

[56] Lin, S., and Hanrahan, P. Modeling how people extract color themes from images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2013), CHI '13, ACM, pp. 3101–3110.

[57] Lin, S., Ritchie, D., Fisher, M., and Hanrahan, P. Probabilistic color-by-numbers: Suggesting pattern colorizations using factor graphs. *ACM Trans. Graph. 32*, 4 (July 2013), 37:1–37:12.

[58] Liu, B., Martin, R. R., Huang, J.-W., and Hu, S.-M. Structure aware visual cryptography. *Comput. Graph. Forum 33*, 7 (Oct. 2014), 141–150.

[59] Liu, F., and Wu, C. Embedded extended visual cryptography schemes. *IEEE Transactions on Information Forensics and Security 6*, 2 (June 2011), 307–322.

[60] Liu, F., and Yan, W. Q. *Visual Cryptography for Image Processing and Security*, 2 ed. Springer International Publishing, 2015.

[61] Liu, L., Chen, W., Zheng, W., and Geng, W. Structure-aware error-diffusion approach using entropy-constrained threshold modulation. *Vis. Comput. 30*, 10 (Oct. 2014), 1145–1156.

[62] Ma, C., Wei, L.-Y., and Tong, X. Discrete element textures. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 62:1–62:10.

[63] Mahy, M., Van Eycken, L., and Oosterlinck, A. Evaluation of uniform color spaces developed after the adoption of cielab and cieluv. *Color Research & Application 19*, 2 (1994), 105–121.

[64] Meghrajani, Y. K., and Mazumdar, H. S. Universal share for multisecret image sharing scheme based on boolean operation. *IEEE Signal Processing Letters 23*, 10 (Oct 2016), 1429–1433.

[65] Mellado, N., Vanderhaeghe, D., Hoarau, C., Christophe, S., Brédif, M., and Barthe, L. Constrained palette-space exploration. *ACM Trans. Graph. 36*, 4 (July 2017), 60:1–60:14.

[66] Merrell, P., Schkufza, E., and Koltun, V. Computer-generated residential building layouts. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 181:1–181:12.

[67] Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. 30*, 4 (July 2011), 87:1–87:10.

[68] Mitchell, D. P. Generating antialiased images at low sampling densities. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 65–72.

[69] Mitra, N. J., Chu, H.-K., Lee, T.-Y., Wolf, L., Yeshurun, H., and Cohen-Or, D. Emerging images. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 163:1–163:8.

[70] Mitra, N. J., and Pauly, M. Shadow art. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 156:1–156:7.

[71] Moon, P., and Spencer, D. E. Geometric formulation of classical color harmony∗. *J. Opt. Soc. Am. 34*, 1 (Jan 1944), 46–59.

[72] Naor, M., and Shamir, A. *Visual cryptography*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995, pp. 1–12.

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

[74] ODonovan, P., Agarwala, A., and Hertzmann, A. Learning layouts for single-pagegraphic designs. *IEEE Transactions on Visualization and Computer Graphics 20*, 8 (Aug. 2014), 1200–1213.

[75] Ostromoukhov, V. A simple and efficient error-diffusion algorithm. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 567–572.

[76] Ostwald, W., and Birren, F. *The color primer: a basic treatise on the color system of Wilhelm Ostwald*. Van Nostrand Reinhold Co., 1969.

[77] Ou, L.-C., Chong, P., Luo, M. R., and Minchew, C. Additivity of colour harmony. *Color Research & Application 36*, 5 (2011), 355–372.

[78] Ou, L.-C., and Luo, M. R. A colour harmony model for two-colour combinations. *Color Research & Application 31*, 3 (2006), 191–204.

[79] Ou, L.-C., Ronnier Luo, M., Sun, P.-L., Hu, N.-C., Chen, H.-S., Guan, S.-S., Woodcock, A., Caivano, J. L., Huertas, R., Treméau, A., Billger, M., Izadan, H., and Richter, K. A cross-cultural comparison of colour emotion for two-colour combinations. *Color Research & Application 37*, 1 (2012), 23–43.

[80] Pang, W.-M., Qu, Y., Wong, T.-T., Cohen-Or, D., and Heng, P.-A. Structure-aware halftoning. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 89:1–89:8.

[81] Pang, X., Cao, Y., Lau, R. W. H., and Chan, A. B. Directing user attention via visual flow on web designs. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 240:1–240:11.

[82] Papas, M., Houit, T., Nowrouzezahrai, D., Gross, M., and Jarosz, W. The magic lens: Refractive steganography. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 186:1–186:10.

[83] Phan, H., Fu, H., and Chan, A. Color orchestra: Ordering color palettes for interpolation and prediction. *IEEE Transactions on Visualization and Computer Graphics PP*, 99 (2017), 1–1.

[84] Pjanic, P., and Hersch, R. D. Color imaging and pattern hiding on a metallic substrate. *ACM Trans. Graph. 34*, 4 (July 2015), 130:1–130:10.

[85] Proakis, J. G., and Manolakis, D. G. *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[86] Reinert, B., Ritschel, T., and Seidel, H.-P. Interactive by-example design of artistic packing layouts. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 218:1–218:7.

[87] Reinert, B., Ritschel, T., Seidel, H.-P., and Georgiev, I. Projective blue-noise sampling. *Computer Graphics Forum* (2015), n/a–n/a.

[88] Ritchie, D., Lin, S., Goodman, N. D., and Hanrahan, P. Generating design suggestions under tight constraints with gradient-based probabilistic programming. *Comput. Graph. Forum 34*, 2 (May 2015), 515–526.

[89] Rubner, Y., Tomasi, C., and Guibas, L. J. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on* (Jan 1998), pp. 59–66.

[90] Sakurai, K., and Miyata, K. Generating layout of nonperiodic aggregates. In *NICOGRAPH International* (2012), pp. 68–75.

[91] Sawant, N., and Mitra, N. J. Color harmonization for videos. In *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing* (Washington, DC, USA, 2008), ICVGIP '08, IEEE Computer Society, pp. 576–582.

[92] Schlömer, T., and Deussen, O. Towards a standardized spectral analysis of point sets with applications in graphics, 2010.

[93] Schmaltz, C., Gwosdek, P., and Weickert, J. Multi-class anisotropic electrostatic halftoning. *Computer Graphics Forum 31*, 6 (2012), 1924–1935.

[94] Shamir, A. How to share a secret. *Commun. ACM 22*, 11 (Nov. 1979), 612–613.

[95] Sharma, G., Wu, W., and Dalal, E. N. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application 30*, 1 (2005), 21–30.

[96] Shugrina, M., Lu, J., and Diverdi, S. Playful palette: An interactive parametric color mixer for artists. *ACM Trans. Graph. 36*, 4 (July 2017), 61:1–61:10.

[97] Shyu, S. J., Huang, S.-Y., Lee, Y.-K., Wang, R.-Z., and Chen, K. Sharing multiple secrets in visual cryptography. *Pattern Recognition 40*, 12 (2007), 3633 – 3651.

[98] Szabó, F., Bodrogi, P., and Schanda, J. Experimental modeling of colour harmony. *Color Research & Application 35*, 1 (2010), 34–49.

[99] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B) 58* (1996), 267–288.

[100] Ulichney, R. *Digital Halftoning*. MIT Press, Cambridge, MA, USA, 1987.

[101] Wang, B., Yu, Y., Wong, T.-T., Chen, C., and Xu, Y.-Q. Data-driven image color theme enhancement. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 146:1–146:10.

[102] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (April 2004), 600–612.

[103] Wei, L.-Y. Multi-class blue noise sampling. *ACM Trans. Graph. 29*, 4 (July 2010), 79:1–79:8.

[104] Wei, L.-Y., Lefebvre, S., Kwatra, V., and Turk, G. State of the art in example-based texture synthesis. In *Eurographics '09 State of the Art Reports (STARs)* (March 2009), Eurographics.

[105] Wijffelaars, M., Vliegen, R., Van Wijk, J. J., and Van Der Linden, E.-J. Generating color palettes using intuitive parameters. *Computer Graphics Forum 27*, 3 (2008), 743–750.

[106] Wu, W., Fan, L., Liu, L., and Wonka, P. Miqp-based layout design for building interiors. *Computer Graphics Forum 37*, 2 (2018), 511–521.

[107] Yan, D.-M., Guo, J.-W., Wang, B., Zhang, X.-P., and Wonka, P. A survey of blue-noise sampling and its applications. *Journal of Computer Science and Technology 30*, 3 (2015), 439–452.

[108] Yu, L.-F., Yeung, S.-K., Tang, C.-K., Terzopoulos, D., Chan, T. F., and Osher, S. J. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph. 30*, 4 (July 2011), 86:1–86:12.

[109] Yu, L.-F., Yeung, S.-K., Terzopoulos, D., and Chan, T. F. Dressup!: Outfit synthesis through automatic optimization. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 134:1–134:14.

[110] Zhang, E., Mischaikow, K., and Turk, G. Vector field design on surfaces. *ACM Trans. Graph. 25*, 4 (Oct. 2006), 1294–1326.

[111] Zhang, S. H., Li, X. Y., Hu, S. M., and Martin, R. R. Online video stream abstraction and stylization. *IEEE Transactions on Multimedia 13*, 6 (Dec 2011), 1286–1294.

CHAPTER A

Appendix for Discrete Element Textures (Chapter 3)

A.1 BuildRMatrix

In the proposed algorithm, RMatrix is constructed according to the literature [103]. In our case, the normalized RMatrix is returned by the BuildRMatrix(·) function.

Algorithm 4 BuildRMatrix($\{r_i\}_{i=0:c-1}$)

// user specified per-class values; $\{r_i\}$,
// number of classes; c
for $i = 0$ to $c - 1$ **do**
    $\mathbf{r} \leftarrow r_i$ // initialize diagonal entries
end for
sort the c classes into priority group $\{\mathbf{P}_k\}_{k=0:p-1}$ with descreasing r_i
$C \leftarrow \emptyset$ // the set of classes already processed
$D \leftarrow 0$ // the density of classes already processed
for $k = 0$ to $c - 1$ **do**
    $C \leftarrow C \bigcup \mathbf{P}_k$
    for each class $i \in \mathbf{P}_k$ **do**
        $D \leftarrow D + \frac{1}{r_i^2}$
    end for
    for each class $i \in \mathbf{P}_k$ **do**
        for each class $j \in C$ **do**
            if $i \neq j$ **then**
                $\mathbf{r}(i, j) \leftarrow \mathbf{r}(j, i) \leftarrow \frac{1}{\sqrt{D}}$
            end if
        end for
    end for
end for
$\mathbf{r}^{\text{scale}} \leftarrow \mathbf{r} / \min(\mathbf{r})$ // element-wise division
return $\mathbf{r}^{\text{scale}}$

A.2 Statistical Analysis for Multi-class Anisotropic Blue Noise Sampling

We analyzed our algorithm by calculating its power spectrum and anisotropy according to the literature [54]. Figures A.1 to A.3 show the analysis results for 3-class, 5-class, and 7-class distribution respectively. We plotted the total set and each class. To verify the anisotropic sampling, a distribution is warped back from a scaling projection to a unit square. In the original distribution, we employed a simple scaling transform from $\mathbf{p} = (x, y)$ to $\mathbf{q} = (u, v)$, i.e., $\mathbf{q} = \varphi(\mathbf{p})$, where

$$\varphi : (u, v) = (x/2, y). \tag{A.1}$$

Therefore, the points can be warped back by applying a Jacobian matrix expressed as

$$J(\varphi^{-1}(\mathbf{q})) = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial x}{\partial y} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}. \tag{A.2}$$

The spectrum analysis results are shown in Figures A.1 to A.3. Here, simple scaling is used for the warp to test its anisotropic and multi-class sampling properties (left to right: original (anisotropic) samples, warped (isotropic) samples, power spectrum averaged over 10 runs, and the corresponding radial mean and anisotropy plots). The total set contains approximately 3800 samples, and each class contains an almost equal number of samples.

As can be seen, the total set of all cases exhibits blue noise properties. In Figure A.1, each class also exhibits blue noise properties. However, in Figures A.2 and A.3, each class is slightly biased due to a smaller number of points and may also suffer from a scaling effect and constrained distribution resulting from multi-class sampling. However, we aim for multi-class element distribution rather than a pure sampling application, such as anti-aliasing; thus, this is acceptable because we can generate visually appealing results, which are shown in Section 3.4.

The spectrum analysis for 16-/32-/64-classes are shown in Figure A.4. As can be seen, our algorithm can also generate distributions with blue noise properties even with a large number of classes.

A.3 Comparison of Sampling Approaches

We compare the distribution methods in Section 3.5. Here, we compare our method (Figure A.1) to anisotropic sampling with random class assignment (Figure A.6). We find that there are significantly sparser and denser areas of samples in each class in Figure A.6 than those shown in Figure A.1, which results in fewer blue noise qualities.

We also generate a discrete element pattern using three methods: (a) multi-class isotropic sampling [103], (b) anisotropic sampling [54] with random class assignment, and (c) our multi-class anisotropic sampling (Figure A.5). As can be seen, the anisotropy of an element is ignored and the elements are not well populated in Figure A.5a. Although Lagae and Dutré employed

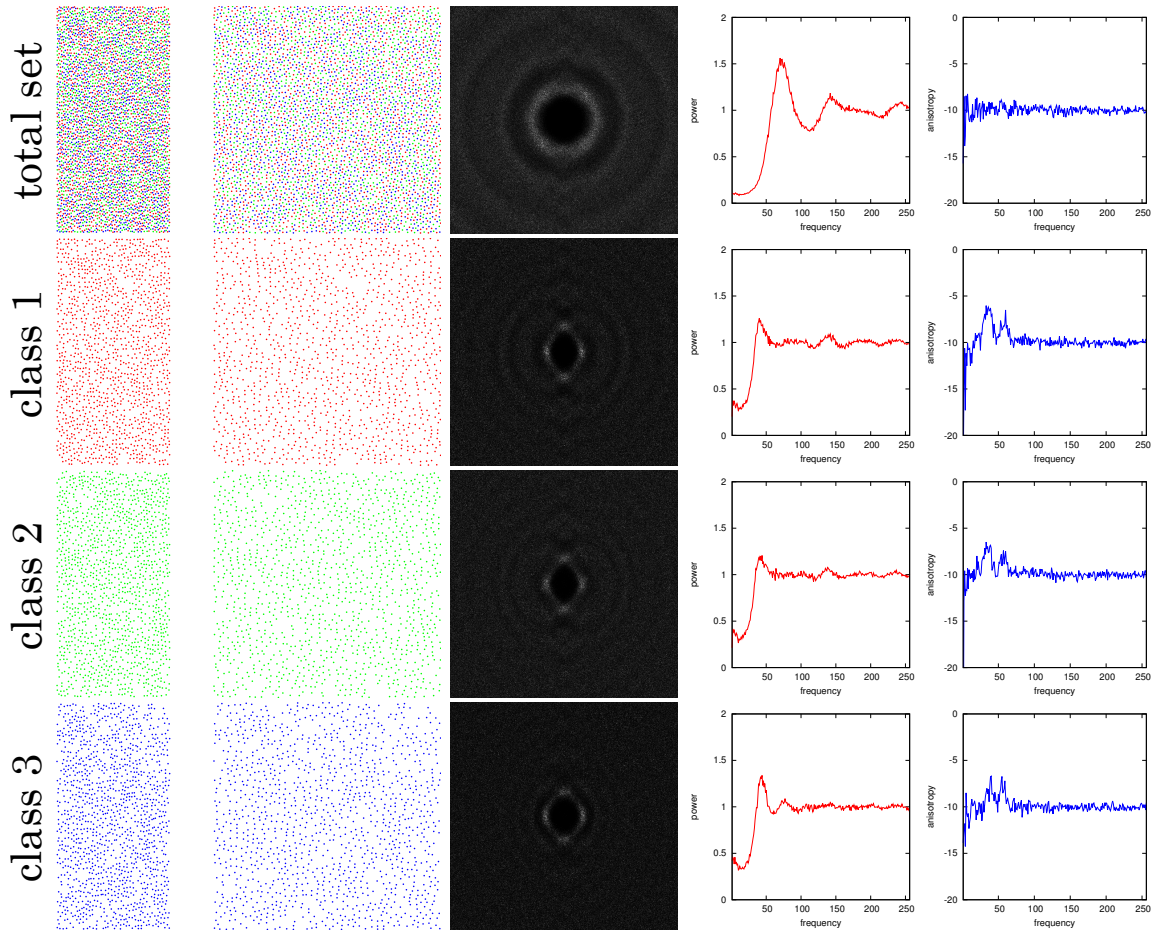


FIGURE A.1 Spectrum results for 3-class distribution.

an isotropic Poisson disk sampling approach for object distribution [50], Figure A.5 shows that isotropic Poisson disk sampling is unsuitable for anisotropic element distribution. In Figure A.5b, the elements are well populated; however, it does not consider multi-class distribution as in Figure A.5a. As a result, the same class elements are not well uniformly distributed in the whole domain. As shown in Figure A.5c, ours is well populated and the elements of the same class are uniformly distributed in the domain.

A.4 All Patterns Shown to Participants

All of the patterns shown to the participants in our experiment are shown in Figure A.7. Since the previous discrete element placement approaches lack the ability to distribute multi-class or anisotropic elements, we compare our approach to discrete element texture synthesis approaches.

Although the statistical test shows that our method outperforms other methods, the results from these synthesis approaches are highly dependent on the input exemplar. As mentioned in Section 3.1, creating a visually appealing swatch (pattern), i.e., an exemplar, is difficult. This indicates that the proposed method might be useful while creating a visually appealing pattern, and the resulting pattern can be used as input exemplar for discrete texture synthesis approaches.

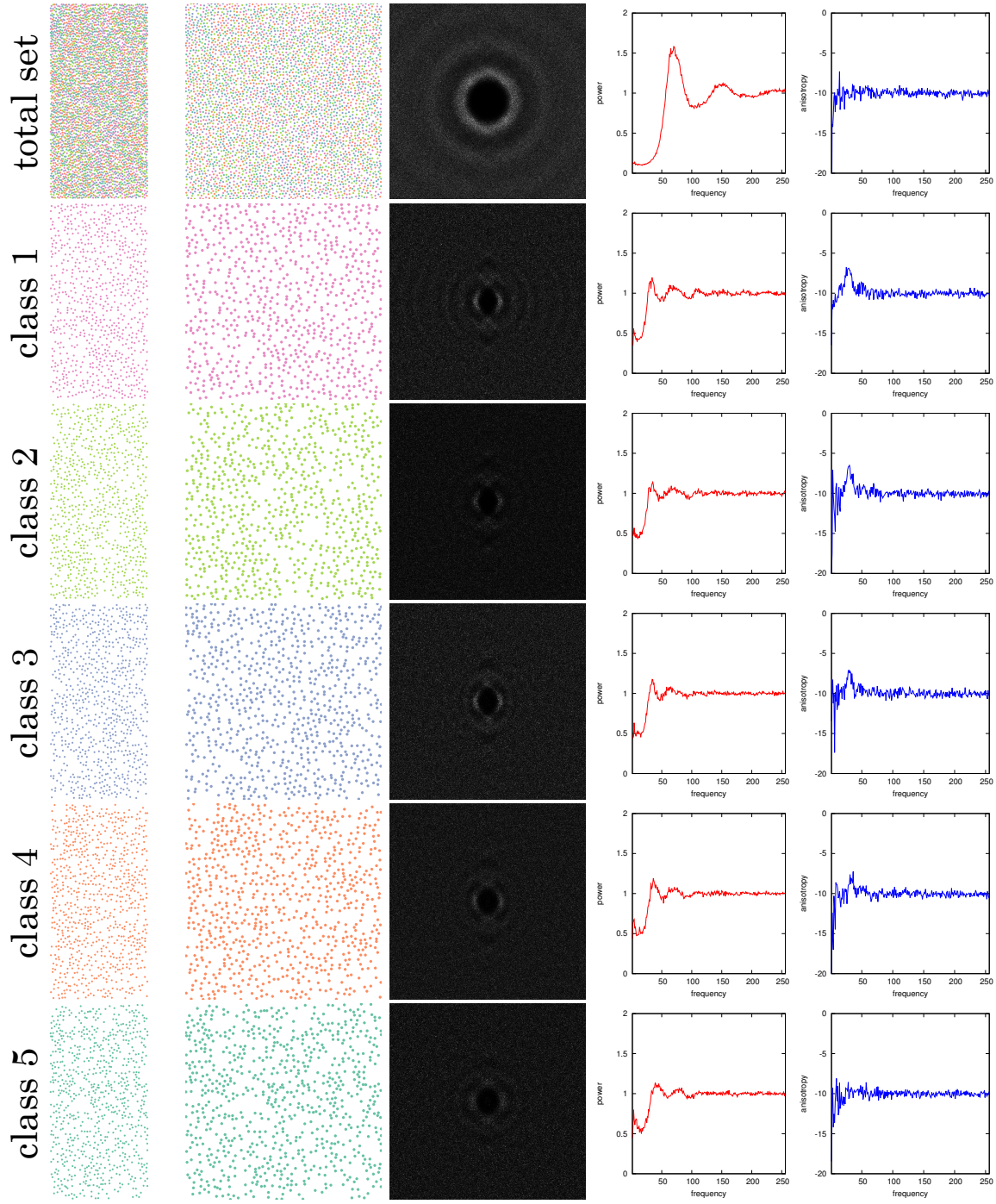


FIGURE A.2 Spectrum results for 5-class distribution.

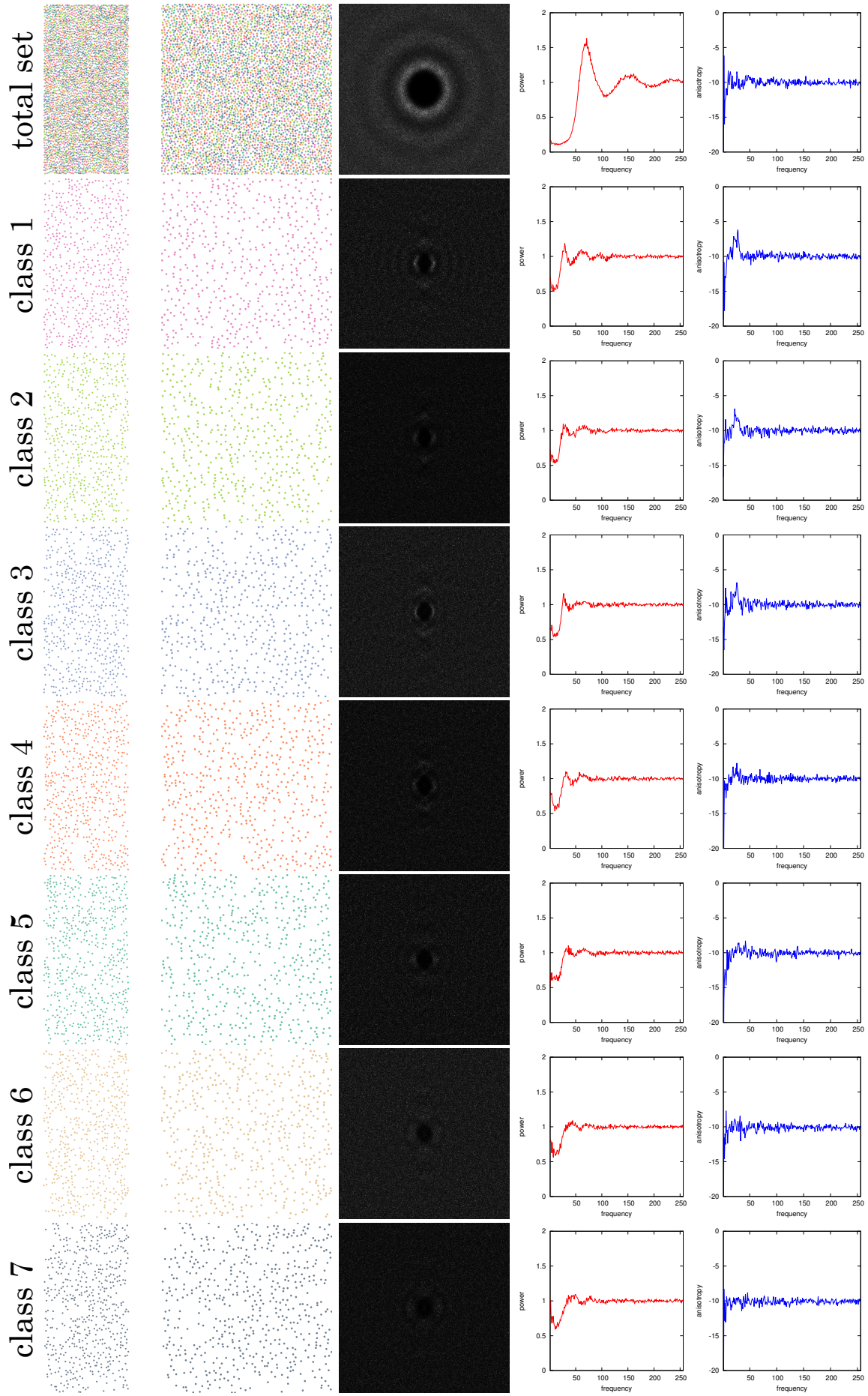


FIGURE A.3 Spectrum results for 7-class distribution.

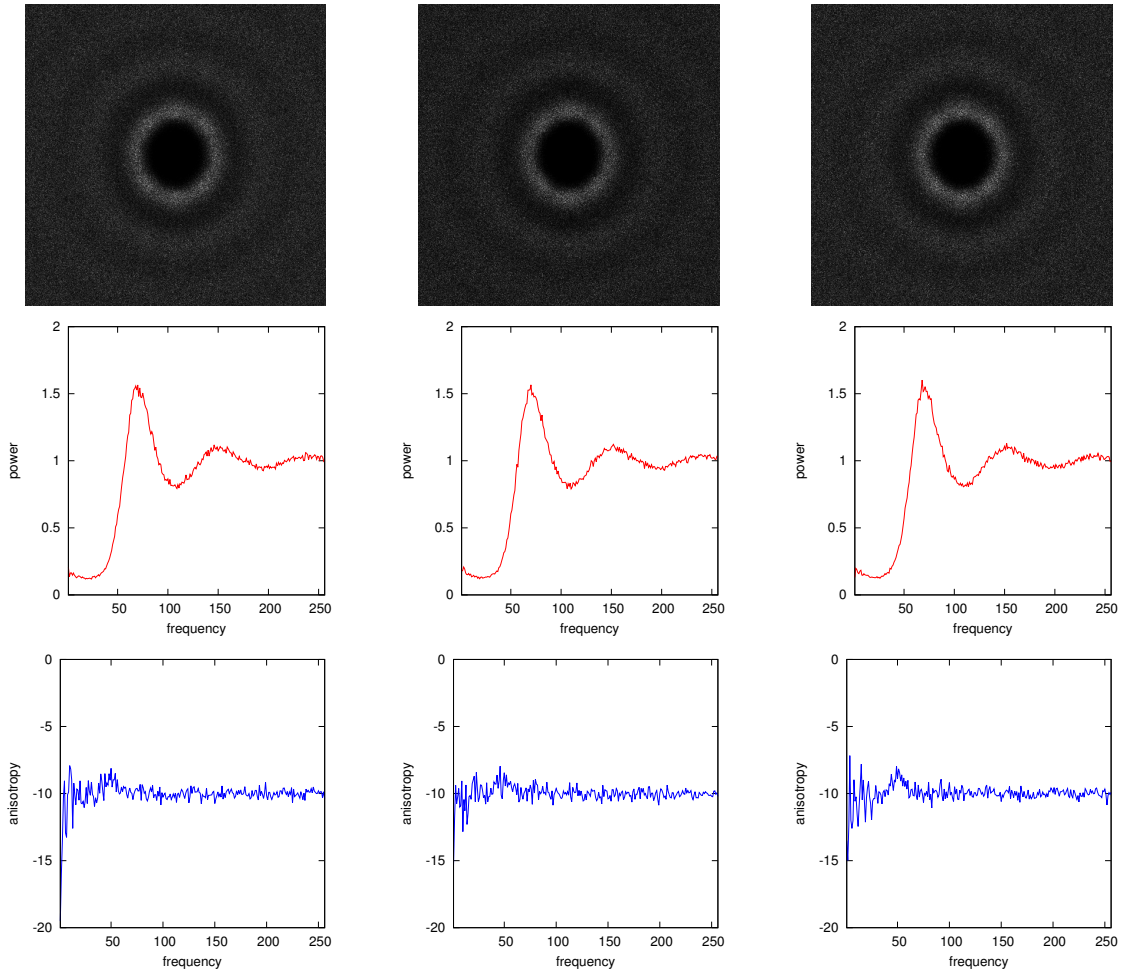
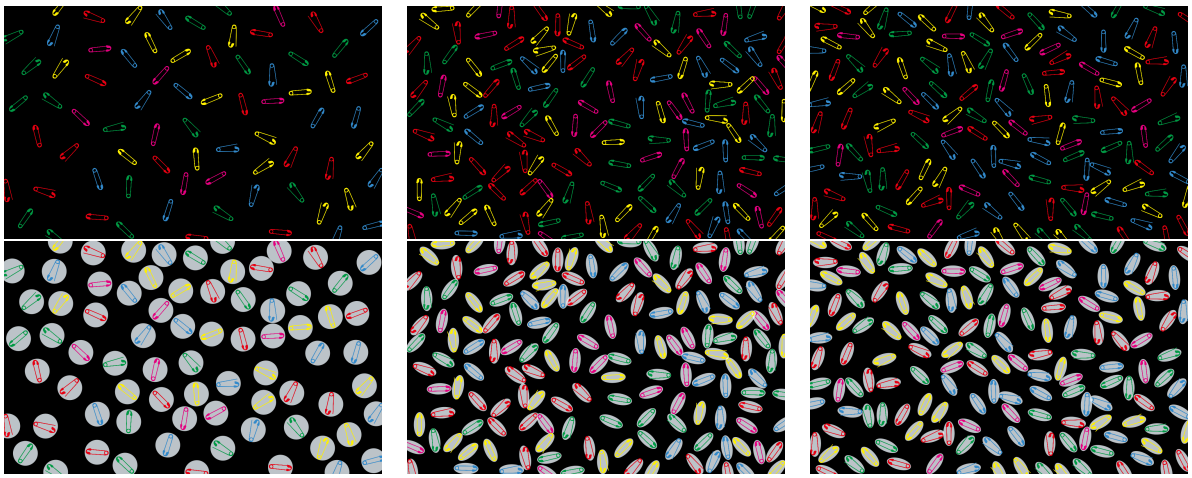


FIGURE A.4 Spectrum results of distribution with more classes. Left to right: 16-/32-/64-classes. In these cases, simple scaling is applied, as well as warp back from a scaling projection to a unit square for spectrum and anisotropy analysis.



(a) Multi-class (isotropic) (b) Anisotropic (random class) (c) Ours

FIGURE A.5 Comparison of three methods: (a) multi-class isotropic sampling [103], (b) anisotropic sampling [54] with random class assignment, and (c) our multi-class anisotropic sampling. The generated patterns are shown in the top row, and the elements with proxy shapes are shown in the bottom row.

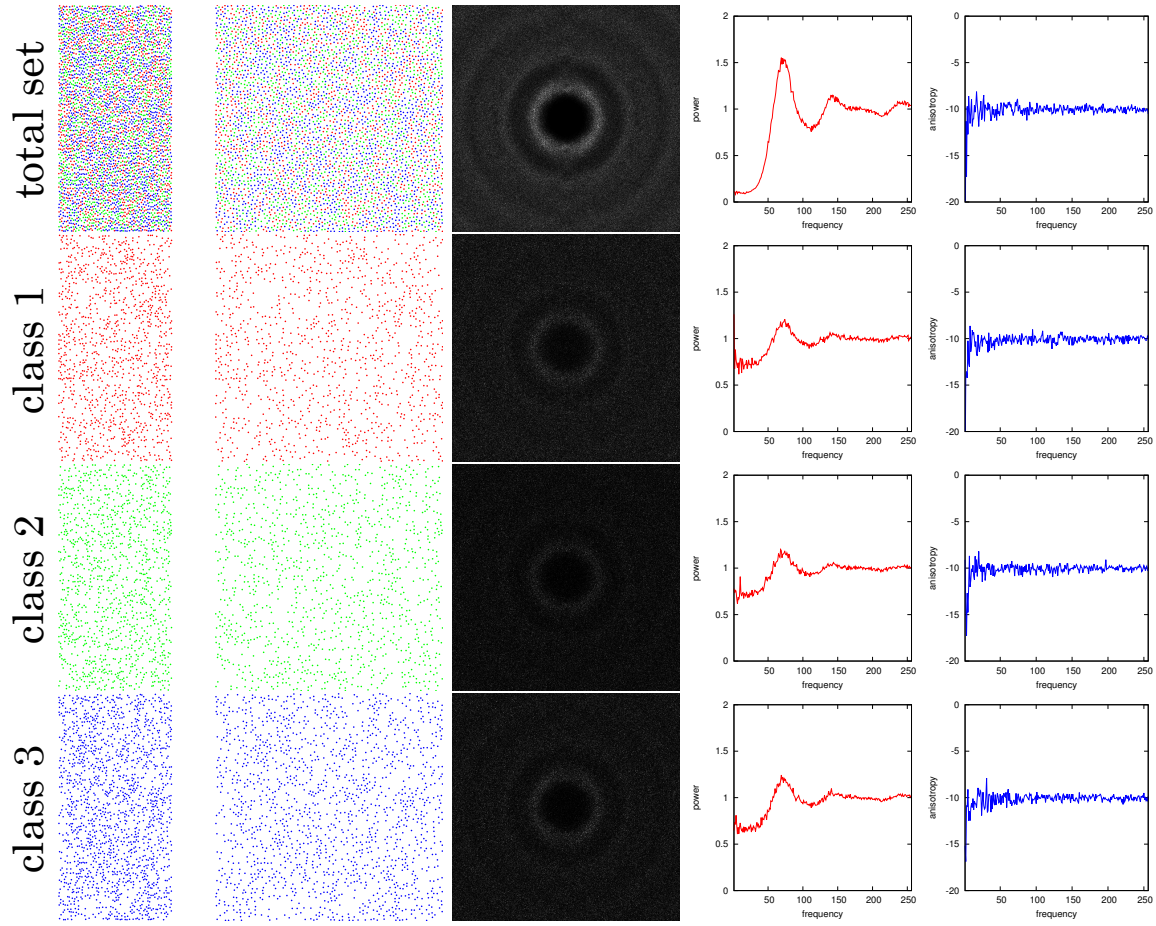


FIGURE A.6 Spectrum results for 3-class distribution. In this case, we first generate a distribution by anisotrpic sampling. Then, each sample is assigned a class ID randomly while maintaining a nearly equal ratio relative to the number of classes.

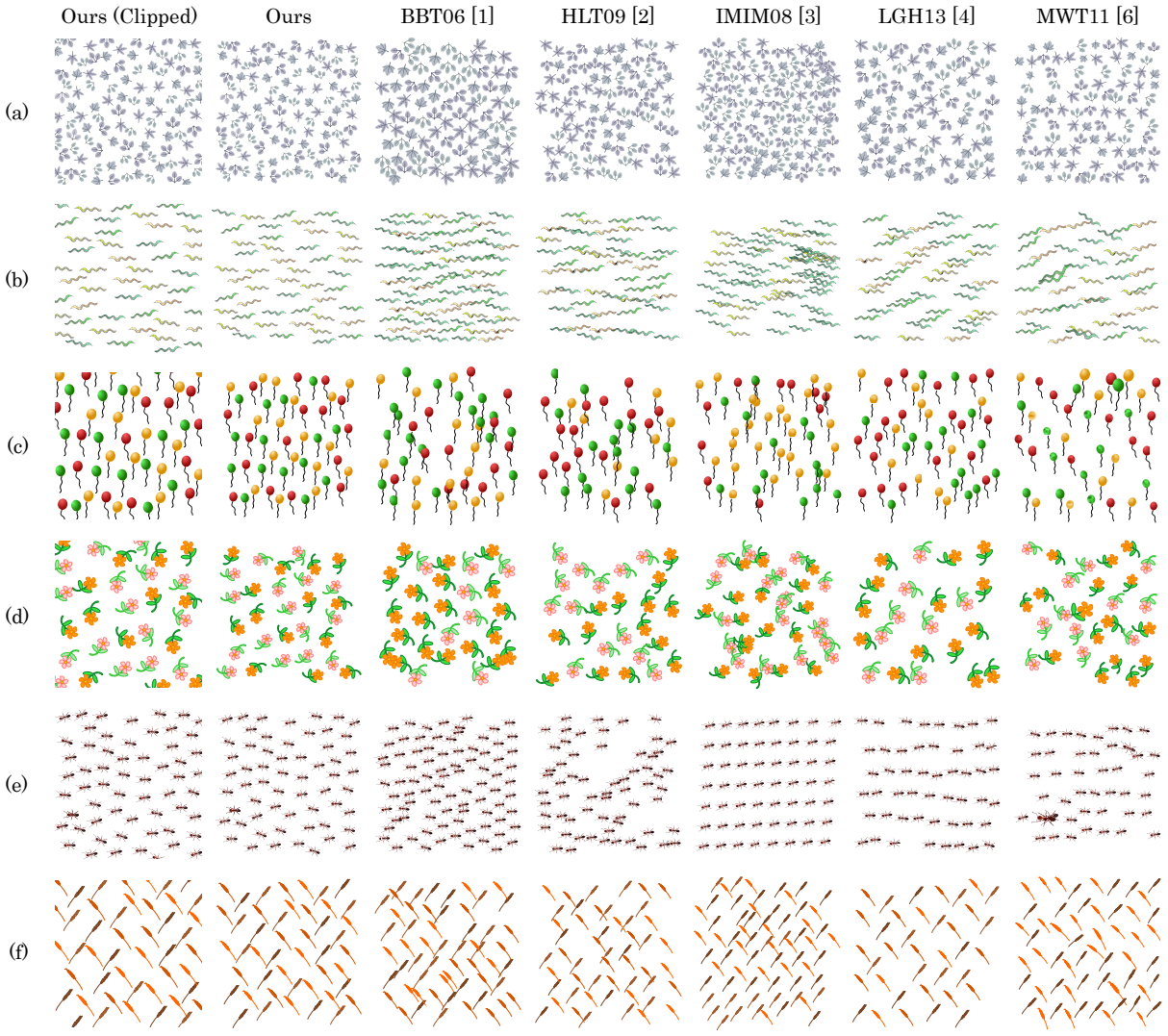


FIGURE A.7 All patterns shown to participants in our experiment. Left to right: ours (*clipped*), ours, BBT06 [6], HLT09 [41], IMIM08 [42], LGH13 [52], and MWT11 [62]. Top to bottom: (a) leaf, (b) snake, (c) balloon, (d) flower, (e) ant, and (f) wheat. The results other than ours are courtesy Landes et al. [52].

CHAPTER B

Appendix for Discrete Color Palettes (Chapter 4)

B.1 Proposed Algorithm

The algorithms proposed in Section 4.3 are given below.

Algorithm 5 SuggestCompatibleColors($t, k, N_{\text{cand}}, N_{\text{sample}}, \tau, \kappa$)

// palette; t,
// index; k,
// #cands; N_{cand},
// #samples; N_{sample},
// thresholds; τ, κ
▷ *Sampling candidate's HSVs*
$f \leftarrow$ ComputeHueProbability(t, k) ▷ *Equation 4.3 or Equation 4.4*
$h \leftarrow$ SamplingFromHueProb(f, N_{sample})
$s \sim \mathcal{N}(\mu_s, \sigma_s)$ ▷ §*4.3.2*
$v \sim \mathcal{N}(\mu_v, \sigma_v)$ ▷ §*4.3.2*
▷ *Compute rating*
for $i = 1 \rightarrow m$ **do**
    $c_i \leftarrow (h_i, s_i, v_i)$
    $C_i^{\text{cand}} \leftarrow$ CompatibleCandidates(t, c_i, τ) ▷ *Equation 4.2*
end for
$C \leftarrow$ PerceptualThresholding(C^{cand}, N_{cand}, κ) ▷ *Equation 4.5*
return C

B.2 Feature Extraction

We extract the following features from a given palette. The basic, plane, and HSV features are derived from those used in [73].

Color Space. We extract the features of a given palette from RGB, CIELAB, HSV, and CHSV color spaces.

Algorithm 6 SamplingFromHueProb(f, n)

// function; f,
// #samples; n
$H \leftarrow \emptyset$ ▷ initialize a set that holds accepted hue samples
for $i = 1 \rightarrow n$ **do**
    ▷ *Rejection sampling from* f
    $h_i \leftarrow$ RandomInteger[0,360] ▷ Select a hue value
    $u_i \sim \mathcal{U}_{[0,\max f]}$
    if $u_i \leq f(h_i)$ **then**
        $H \leftarrow H \cup \{h_i\}$
    end if
end for
return H ▷ Out of n samples, m accepted hue samples are returned

Basic Feature (BF). Basic statistics, i.e., `Mean`, `StdDev`, `Median`, `Max`, `Min`, and `MaxMinDiff`, for each of the three dimensions of a color space are extracted. For the BF vector, 72 features are extracted.

Plane Feature (PF). A 2D plane is fitted to 3D color coordinates using PCA in RGB, CIELAB, and CHSV color spaces, and `Normal`, `Variance`, and `SSE` are computed as the plane features. Twenty-one features are extracted for the PF vector.

HSV Feature (HsvF). In HSV color space, we extract the `Mean`, `StdDev`, `Min`, `Max`, `LogMean`, `LogStdDev`, `LogMin`, and `LogMax` features from the hue probability of color c; p_c. We also extract these features from the hue adjacent probability of colors b and c; p_{bc}^{adj} and the hue joint probability of b and c; p_{bc}^{joint}. Here p_c is the percentage of colors in training palettes with hue c, p_{bc}^{adj} is the percentage of adjacent colors b and c, and p_{bc}^{joint} is the percentage of colors b and c in the same palette. Twenty-five features are extracted for HsvF.

In addition, we extract the hue entropy, which is defined as a probability distribution $p(\theta)$ computed using the hues of a given palette comprising n colors as a mixture of von Mises distribution, i.e., $p(\theta) \propto \sum_{i=1}^{n} \exp(2\pi \cos(\theta - \theta_i))$.

Additional feature (CH and Grads). In addition to the above 118 features, we also extract three additional features. The first feature is Color Harmony (CH) [78]. The process to calculate CH is given in Section B.4. The other features are lightness and hue gradation (Grads), according to the analysis described in Section B.5. We compute linear regressions in ascending/descending order of colors in a palette, and the resulting R^2 values are used as the gradation features. In total, we extract 121 features from a given palette.

B.3 Model Analysis

A detailed analysis of the contributions of each feature extracted from a given palette is shown in Table B.1. The results show that BF has the largest contribution. According to the

results, we include all these features for feature extraction (row 13).

TABLE B.1 Contributions of each feature (BF, Basic Feature; PF, Plane Feature; HsvF, HSV Feature; CH, Color Harmony; Grads, Gradation).

	BF	PF	HsvF	CH	Grads	R value
1	✓					0.70
2		✓				0.38
3			✓			0.27
4				✓		0.33
5					✓	0.14
6		✓	✓			0.42
7				✓	✓	0.37
8	✓			✓	✓	0.71
9		✓	✓	✓	✓	0.51
10	✓	✓	✓			0.71
11	✓	✓	✓	✓		0.71
12	✓	✓	✓		✓	0.72
13	✓	✓	✓	✓	✓	**0.72**

B.4 Two-color Harmony Model

Ou et al. proposed the following two-color harmony model based on the results of a user study [78]:

$$CH = H_C + H_L + H_H, \tag{B.1}$$

where H_C, H_L, and H_H are the terms related to chroma, lightness, and hue, respectively, calculated by the following equations.

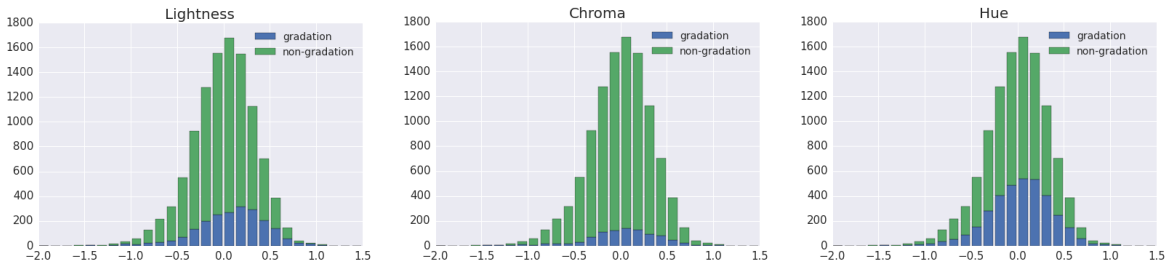
$$\begin{aligned}
H_C &= 0.04 + 0.53\tanh(0.8 - 0.045\Delta C) \\
\Delta C &= [(\Delta H^*_{ab})^2 + (\Delta C^*_{ab}/1.46)^2]^{\frac{1}{2}} \\
H_L &= H_{L_{\text{sum}}} + H_{\Delta L} \\
H_{L_{\text{sum}}} &= 0.28 + 0.54\tanh(-3.88 + 0.029 L_{\text{sum}}) \\
&\quad \text{in which } L_{\text{sum}} = L^*_1 + L^*_2 \\
H_{\Delta L} &= 0.14 + 0.15\tanh(-2 + 0.2\Delta L) \\
&\quad \text{in which } \Delta L = |L^*_1 - L^*_2| \\
H_H &= H_{\text{SY1}} + H_{\text{SY2}} \\
H_{\text{SY}} &= E_C(H_S + E_Y) \\
E_C &= 0.5 + 0.5\tanh(-2 + 0.5 C^*_{ab})
\end{aligned}$$

$$H_S = -0.08 - 0.14\sin(h_{ab} + 50°) - 0.07\sin(2h_{ab} + 90°)$$

Here L^*, C^*_{ab}, and h_{ab} are the lightness, chroma, and hue values in the CIELAB color space, respectively. We calculated CH as a feature for all combinations of colors in a given palette.

B.5 Gradation Analysis

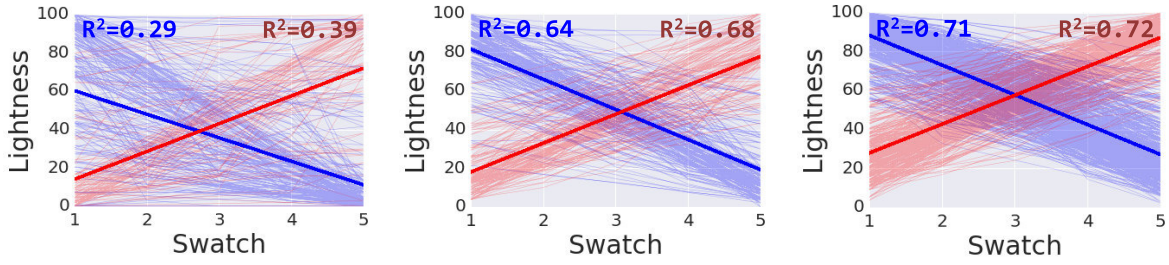
We analyzed whether a color palette in the [73] dataset displays linearity in the order of colors. We counted the lightness/chroma/hue gradation of a palette for all palettes in the dataset to determine if the order of colors decreases (increases) monotonically in the palette's line plot (Figure B.1). A palette with lightness gradation tends to obtain a higher rate than those without lightness gradation (Figure B.1a). As can be seen, hue gradation has the same tendency as lightness gradation (Figure B.1c); however, no significant differences were observed for chroma (Figure B.1b).



(a) Lightness (b) Chroma (c) Hue

FIGURE B.1 Histogram of #palette of (non-)gradations in the dataset.

We also analyzed the quality of linearity of the lightness gradation. We split the dataset to lower-/middle-/higher-rated groups. Each group had $1,000$ palettes. Then, all the palettes in the group were plotted and linear regression was computed (Figure B.2). The bold blue and red lines are the fitting results. From the R^2 value, it can be observed that linearity tends to increase as the rating increase. Therefore, we include the gradation term in the feature extraction method.



(a) Low-rated (b) Mid-rated (c) High-rated

FIGURE B.2 Linearity of lightness gradation.

CHAPTER C

Appendix for Discrete Element Patterns (Chapter 5)

C.1 Share Optimization

We minimize the following objective function:

$$E_{\text{total}}(I^p, S^{p'}) = E_s(I^p, S^{p'}) + \lambda E_c(I^p, S^{p'}), \tag{C.1}$$

where E_s and E_c are structure and coherence terms, respectively, defined as follows.

$$\begin{aligned} E_s(I^p, S^{p'}) &= \sum_{k\in\{1,c,2\}} w_k \Big(w_g G(I_k^p, S_k^{p'}) + w_t \big(1 - \text{MSSIM}(I_k^p, S_k^{p'})\big)\Big) \\ E_c(I^p, S^{p'}) &= \sum_{k\in\{1,c,2\}} \sum_{q\in N(p)} \left\| S_k^{q'} - S_k^{p'} \right\| e^{-\beta\cdot\left\| I_k^q - I_k^p \right\|} \end{aligned} \tag{C.2}$$

Here, p is the pixel location in input I and p' is the corresponding block of subpixels in the output share S.

Structure Term. The structure term E_s comprises a tone similarity term and a structure similarity term [58, 80]. Tone similarity is measured by $G(I_k^p, S_k^{p'})$, where I^p and $S^{p'}$ are the local region, which is one-half the size of the pixel expansion plus one-half the size of the Gaussian kernel, and $G(\cdot,\cdot)$ measures the mean-squared-error of the Gaussian-blurred input images. Here we use an 11×11 Gaussian kernel. We employ the MSSIM [102]; thus, the structure term $(1 - \text{MSSIM}(I_k^p, S_k^{p'}))$ measures the dissimilarity of the structures of I_k^p and $S_k^{p'}$. Note that parameters w_k, w_g, and w_t are weight factors. Here, $w_k = 1$ for all k, and $w_g = w_t = 0.5$.

Coherence Term. The coherence term E_c smooths noise that appears in textureless areas [58]. Here, p and q are the pixel locations in input I, and p' and q' are the corresponding blocks of subpixels in the output share S. $N(p)$ is a set of eight connected neighboring pixels of p. The exponential term is used to attenuate the energy as the difference between I_k^p and I_k^q increases, and β controls the attenuation rate ($\beta = 5$). Note that we use $\lambda = 0.01$ to balance the structure and coherence terms in E_{total}.

Algorithm 7 Optimization procedure

$T \leftarrow T_0$
repeat
  for $p \in I$ **do**
    $E_{old} \leftarrow E(I^p, S^{p'})$
    $\mathcal{C} \leftarrow$ RandomSelect(Lut[ShareComb(p)])
    for $k \in \{1, c, 2\}$ **do**
      Rearrange($S_k^{p'}, \mathcal{C}$)
    end for
    $E_{new} \leftarrow E(I^p, S^{p'})$
    $\Delta E = E_{new} - E_{old}$
    Sample $r \in [0, 1]$ at random
    if $r < \exp(\min(0, -\Delta E/T))$ **then**
      $E_{old} \leftarrow E_{new}$
    else
      Undo rearrange $S_k^{p'},\ \forall k \in \{1, c, 2\}$
    end if
  end for
  $T \leftarrow c \times T$
until $T < T_{end}$
return $S_1,\ S_c,\ S_2$

Procedure. The optimization procedure is given in Algorithm 7. We attempt to optimize the output image quality by rearranging *white*/*black* pixels into blocks. Here, we employ simulated annealing optimization. The temperature T is initialized as T_0 and is gradually reduced by cooling factor c until $T < T_{end}$. Here, $T_0 = 0.2$ and $c = 0.95$. At the beginning of the loop, the new arrangement of *white*/*black* pixels in a block has a higher probability to be accepted as a new arrangement even if the arrangement results in worse quality. This is helpful for us because we attempt to minimize error in the local blocks because measuring global error for each loop is computationally expensive.

For each loop, we select a rearrangement candidate from the look-up table Lut. Since we have already computed the share combination of p in the EVCSwithCommonShare procedure, we can retrieve the combination using ShareComb(p), and this combination is used as a Lut query to obtain a new arrangement. We calculate an objective function (Equation (C.1)) for each local region I^p and the corresponding block $S^{p'}$ before and after rearranging $S^{p'}$. Note that acceptance/rejection follows typical simulated annealing techniques.

C.2 Input Images

The input images for Figures 5.9 and 5.10 are listed below.

FIGURE C.1 Top: input images for Figure 5.6. The original images are 240×180 pixels. Bottom: input images for Figures 5.9 and 5.10. The original images are 600×375 pixels. From left to right, Share I_1, Common Share I_c, Share I_2, Secret I_{s_1}, and Secret I_{s_2}.