Probabilistic Color-by-Numbers: Suggesting Pattern Colorizations Using Factor Graphs

Sharon Lin* Stanford University
Daniel Ritchie* Stanford University
Matthew Fisher* Stanford University
Pat Hanrahan* Stanford University

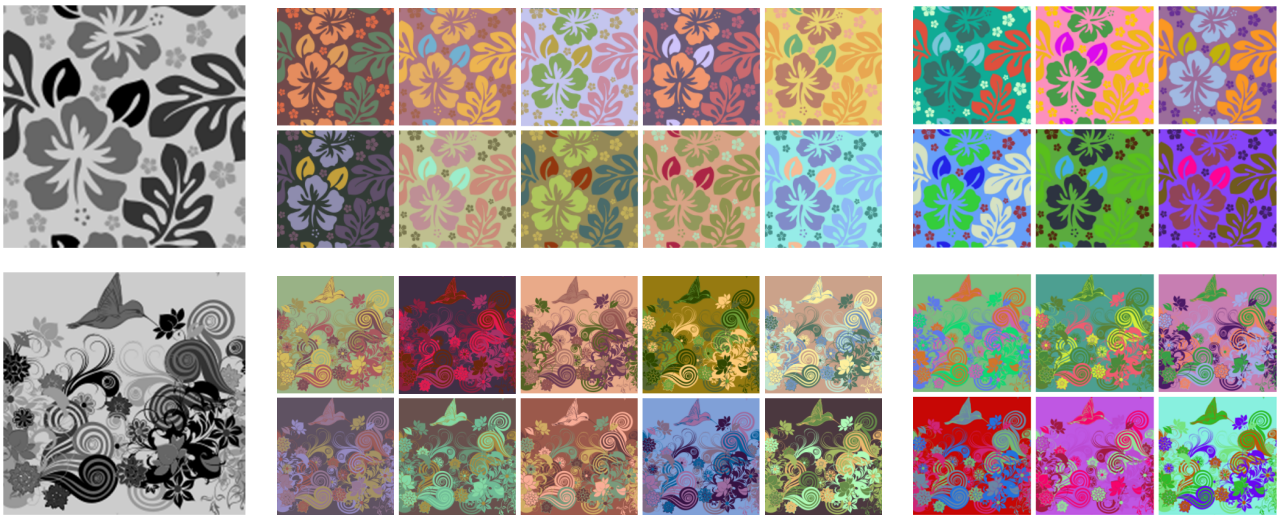


Figure 1: *Given the uncolored pattern templates shown on the left, we use a probabilistic factor graph model to generate the pattern colorings shown in the middle. The factor graph is trained on example patterns colored by human artists. For comparison, on the right we show randomized colorings of each template. The lightness information shown to visualize the input pattern is not used.*

Abstract

We present a probabilistic factor graph model for automatically coloring 2D patterns. The model is trained on example patterns to statistically capture their stylistic properties. It incorporates terms for enforcing both color compatibility and spatial arrangements of colors that are consistent with the training examples. Using Markov Chain Monte Carlo, the model can be sampled to generate a diverse set of new colorings for a target pattern. This general probabilistic framework allows users to guide the generated suggestions via conditional inference or additional soft constraints. We demonstrate results on a variety of coloring tasks, and we evaluate the model through a perceptual study in which participants judged sampled colorings to be significantly preferable to other automatic baselines.

Keywords: Probabilistic modeling, factor graphs, colorization, graphic design, data-driven methods, Markov Chain Monte Carlo

Links: DL PDF

*e-mail: {sharonl, dritchie, mdfisher, hanrahan}@stanford.edu

1 Introduction

From graphic and web design, to fashion and fabrics, to interior design, colored patterns are everywhere. Web designers use them as main images, backgrounds, or repeating page elements, fashion designers print them on clothing and accessories, and interior designers employ them on upholstery, wallpaper, drapes, and more.

A colored pattern has two parts: a *pattern template*, which is a creative decomposition of space into regions, and a set of *colors* assigned to those regions. Additionally, pattern templates often define constraints on which regions must be assigned the same color: childrens' color-by-numbers exercises and the patterns shared on the popular COLOURlovers[1] website are two such examples. It is this color-by-numbers pattern format that we explore in this paper.

While many people can easily distinguish patterns they find pleasing from those they do not, *creating* attractive pattern colorings takes much more time and effort. Because color appearance depends strongly on spatial arrangement, it can be difficult for both experienced artists and enthusiasts to anticipate how a specific coloring will appear. Thus, the coloring process involves much trial-and-error color tweaking. Experienced artists often create quick thumbnail colorings to explore the state space before diving into their final work [Meier et al. 2004].

Can computation make this process easier for artists of all levels by automatically suggesting colorings? To be an effective creative support tool, a coloring suggestion system should adapt to different usage scenarios. First, it should output diverse suggestions automatically for uncertain users who want to explore the space of good

[1] http://www.colourlovers.com/

colorings. Second, it should accommodate users with different aesthetic preferences by customizing suggestions to evoke a particular desired style. Finally, it should expose controls to let users refine their criteria and guide the suggestion process.

Building such a system requires a computational encoding of the properties that make a coloring desirable. There are many principles of aesthetics that might be relevant, such as the different color harmony rules [Sutton and Whelan 2004]. However, which of these principles apply to which patterns and coloring styles? For those that do apply, which are the most important? Even assuming an answer to these questions, there is still the problem of how to *generate* many diverse colorings that satisfy the desired properties.

In this paper, we present a probabilistic approach to automatic pattern colorization using factor graphs. A factor graph is a class of probabilistic graphical model well-suited to encoding complex distributions over multiple random variables. This formalism has been succesfully applied in the graphics literature to synthesize pattern tilings [Yeh et al. 2012a], generate object layouts [Yeh et al. 2012b], and assign materials to 3D objects [Jain et al. 2012].

Our main contribution is a probabilistic factor graph model that can be trained on example patterns and sampled to generate new colorings for a target pattern template. Our model incorporates functions to enforce plausible spatial arrangements of colors as well as overall color compatibility. The individual functions, as well as the relative importance of each one, are automatically trained using machine learning techniques, statistically capturing desirable properties of the example patterns. Using Markov Chain Monte Carlo sampling, our model can generate a wide variety of attractive colorings. In addition, via the use of conditional probabilistic inference or the addition of simple constraint factors, users can exercise control over the generated suggestions.

We demonstrate the effectiveness of our model for a variety of pattern coloring scenarios. We also show applications of our automatic coloring system to 3D scene design, web design, and fashion. Finally, we evaluate the quality of colorings generated by our model through a judgment study. Automatically-colored patterns were significantly preferred to both random colorings and colorings that respect only global color compatibility.

2 Background

Related work has tackled problems similar to the one we address in this paper. While our approach builds on some of the ideas and algorithms presented in past research, these methods alone are not sufficient to meet our goals.

Creative color support tools Addressing a problem similar to ours, Sauvaget and colleagues describe a system that takes an image divided into segments and a set of colors and computes a 'harmonious' assignment of colors to segments based on contrast of proportions guidelines [2010]. The system selects colors from a fixed palette, and users can specify the set of colors to be used and their relative amounts. However, these requirements can conflict with our goal of supporting exploratory coloring, in which the user may not know in advance the colors she wants and would like to see many plausible suggestions. In contrast, our model is trained from artist-created images. It does not require the user to specify colors in advance, but its probabilistic framework is general enough to support this input if the user desires it.

In the theme of user exploration, Meier and colleagues develop a suite of interactive interfaces to help users browse color themes and experiment with compositions [2004]. However, they do not focus on algorithms for generating coloring suggestions. In this paper, we explore generating suggestions while considering user input.

Color harmony & compatibility Computer graphics, aesthetics, and psychology research has introduced theories and models to predict the compatibility of colors [Cohen-Or et al. 2006; Munsell and Birren 1969; Palmer and Schloss 2010; Itten 1974]. Recently, O'Donovan and colleagues presented a data-driven model that predicts the numeric ratings people would give to five-color 'color themes' [2011]. Our model includes this compatibility function. However, as we demonstrate in Section 3, this term alone is insufficient to produce good colorings, as it does not take into account the spatial role each color plays in the pattern. Thus, our model also uses the distinguishing spatial features of each pattern region and the relationships between regions to constrain the space of colorings that it will suggest.

Natural image colorization Related to the pattern colorization problem, researchers have developed many approaches to colorizing grayscale photographic images [Levin et al. 2004; Welsh et al. 2002]. The most automatic of these methods builds distributions of color variability given local texture descriptors, combines them into one energy function, and minimizes it using a graph cut algorithm [Charpiat et al. 2008]. Our model employs similar local color distributions but bases them on features of pattern regions, rather than photographic texture patches. We also cast the problem of finding good colorizations as one of probabilistic inference, rather than optimization. This framing allows our algorithm to explore the space of colorings and suggest multiple good alternatives, whereas a graph cut yields only a single global optimum.

3D model colorization Other recent work has tackled automatic coloring of 3D objects. The DressUp! system suggests plausible outfits—including colors for each clothing item—for virtual characters [Yu et al. 2012]. It also employs the color compatibility function of O'Donovan et al. [2011] as part of a probabilistic model, but it does not consider the spatiality of color beyond a top-to-bottom ordering. The Material Memex system models the context-dependent correlation between geometric shape and material properties of 3D object parts [Jain et al. 2012]. In a related fashion, the system presented in this paper models the context-dependent correlation between pattern regions and color properties of those regions. The Material Memex does not include any explicit consideration of color compatibility. Additionally, its use of multinomial factors requires a quantization of material configuration space. Since it does not consider colors separately from materials, this quantization can restrict the set of colors the model can possibly suggest. In contrast, our model uses continuous probability distributions.

3 Approach

We seek to facilitate the creative coloring process by automatically suggesting pattern colorings. In doing so, we must keep in mind that users may or may not have a target coloring style in mind. In addition, aesthetic taste varies across users and can depend on the situation. Thus, an effective color support system should both output a variety of appealing colorings as well as provide controls for personalizing suggestions to a preferred coloring style.

Our system takes as input a pattern template and outputs suggested colorings for that template. A pattern template specifies which *segments*, or connected components, in an image can be colored in and which segments must map to the same color. For example, an image of a flower on a background may have a template that specifies all petals of the flower must be the same color, and all background

segments must be the same color. We refer to the set of segments that map to the same color as a *color group*. Figure 1 shows an example of a pattern template visualized in grayscale, where each lightness level identifies a different color group. This template representation is relatively easy to author from images composed of segments, such as web designs, 3D renderings, and line drawings.

To generate attractive pattern colorings, a reasonable first step is to enforce that colors are by some definition 'compatible' with one another. Figure 2 shows several patterns whose colors receive a high score under the color compatibility model of O'Donovan et al. [2011]. While these high-scoring colorings use attractive colors and exhibit a great degree of diversity, they also display several problems. Some background regions may be oversaturated, coming across as too 'loud.' Several foreground regions have insufficient contrast with the pattern background, causing them to blend uncomfortably into the background.

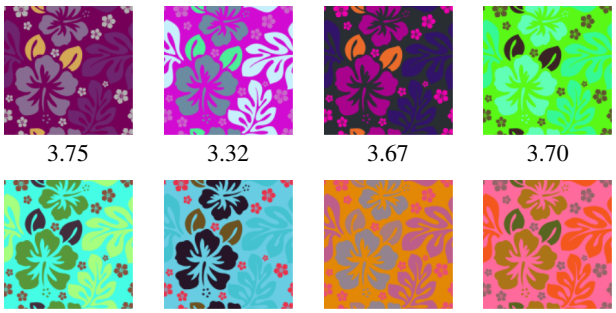


Figure 2: *Patterns whose colors receive high scores under the color compatibility model of O'Donovan et al. [2011]. The score is shown beneath each pattern; a typical pattern scores between 2 and 4. Many results exhibit problems such as adjacent equi-luminant regions and excessively saturated backgrounds.*

To overcome these problems, we turn to examples of well-colored patterns. If we inspect color groups that are large, highly connected, and spread across the entire pattern—indicative features of a background—we can see how saturated they are. This knowledge can prevent us from using excessively 'loud' background colors. If we examine the contrast between adjacent pattern regions, we might find that large foreground regions have high contrast with the background but less contrast with thin borders. Enforcing these same properties in our own colorings should lead to better results.

Consequently, the approach we take to pattern coloring is *data-driven*: given a dataset of example patterns, we learn distributions over color properties such as saturation, lightness, and contrast for individual regions and for adjacent regions. We predict these distributions using discriminative spatial features of the pattern, such as the size and shape of different regions. Finally, we use the predicted distributions to score the goodness of pattern colorings.

In the next sections, we introduce the dataset of patterns used for our experiments (Section 4). Next, we describe the *unary color functions* that we use to score the colors of indvidual pattern regions (Section 5), as well the *pairwise color functions* that score the colors of adjacent regions (Section 6). While color compatibility alone does not predict good pattern colorings, it helps enforce global consistency between colors, and our approach makes use of this ability (Section 7).

We then show how these three types of scoring functions—*unary*, *pairwise*, and *global*—can be combined into one unified model using the framework of probabilistic factor graphs (Section 8). The resulting model is very flexible: we can sample from it to generate a variety of new coloring suggestions, train it on different example sets to capture different coloring styles, and add additional constraints to it to support different usage scenarios (Section 9).

4 Dataset

To build our data-driven model, we use a dataset of colored patterns collected from COLOURlovers, an online community centered around creating and sharing color designs. Artists and enthusiasts can create colored patterns by creating a template from scratch or by coloring an existing template.

Our dataset contains 100 colored patterns for each of 82 artists—8200 colored patterns in total spread over 2908 unique pattern templates. We chose artists in order of their most popular pattern and if he or she had created at least 100 patterns total. The supplemental materials contain the complete list of patterns in our dataset.

These patterns originated as vector images, where each region is mapped to a color in a source color palette. However, our dataset only contains rasterized patterns, which is the format COLOURlovers makes available to the public. Thus, we must preprocess the rasterized patterns to create pattern templates for training. We first map each pixel to a color in the source palette. To reduce quantization noise, we examine the closest palette colors in the 8x8 neighborhood for each pixel, and map that pixel to the mode color. We group all connected components under a threshold size (0.05% image size) into one 'noise' segment. Each other pattern segment is defined as a set of connected components that are 2 pixels or closer to each other, to account for noise. Finally, we create color groups from segments with the same palette color.

5 Unary Color Functions

As discussed in Section 3, color compatibility alone does not predict good pattern colorings. We observed that properties of a pattern region's color can depend on some key spatial features of that region. In this section, we define a set of color properties that we hypothesize contribute to good colorings, a set of spatial features that we use to predict distributions over those properties, and a general-purpose method for performing this prediction.

5.1 Color Properties and Predictive Features

While we could directly predict distributions over colors for pattern regions, predicting distributions instead over *properties* of those colors (e.g. lightness or saturation) has benefits. First, it generalizes better to colors that do not occur in the training dataset but are consistent with that dataset's overall style: a training set that uses pastel colors might not contain a particular pastel blue, but that does not mean our system should not use it. Second, it allows the relative importance of different color properties to be tuned: it may be more critical to set the lightness of a pattern region correctly than to use the perfect hue. Similar approaches have been succesfully deployed to model the aesthetics of photographs [Datta et al. 2006].

Many different color properties can contribute to the appearance of a pattern coloring. For the colors of individual pattern regions, our method considers the following set:

Lightness is the L component of a color in the $L^*a^*b^*$ color space. This value affects how bright the color appears to an observer.

Saturation is the difference between a color and neutral gray, and affects how vivid the color appears. Rather than the typical HSV saturation, we use a more perceptually-based formula that operates in $L^*a^*b^*$ space: $\frac{\sqrt{a^2+b^2}}{\sqrt{a^2+b^2+L^2}}$ [Lübbe 2010].

Color Name Counts is a vector of counts that summarizes how frequently a color is referred to using different names. The common names of a color often convey higher-level stylistic information about it. These vectors were derived in previous work from data collected in a large online color naming survey [Heer and Stone 2012].

Color Name Saliency is a measure of how reliably a color is named [Heer and Stone 2012]. It is derived through an entropy-based formulation and conveys information about how 'instantly recognizable' a color is likely to be.

Distinctive spatial features of a color group, as well as the spatial features of the segments it contains, can affect the appearance of a color assignment. In our system, we use the following group features for color property prediction:

Relative Sizes The area occupied by the group divided by the total area of the pattern, and the area divided by the maximum group area in the template.

Segment Spread The 2D covariance matrix of the group's segment centroids. This feature captures whether the group is concentrated in one section of the pattern or spread across the whole pattern.

Segment Size Statistics The minimum, maximum, mean, and standard deviation of the sizes of segments within the group.

Number of Segments The number of segments in the group divided by the total number of segments in the image.

We also consider the following features for individual segments within a color group:

Relative Sizes The area occupied by the segment divided by the total area of the pattern, and the area divided by the maximum segment area in the template.

Normalized Discrete Compactness A relationship between the segment's boundary edges and its area [Bribiesca 1997].

Elongation The relative narrowness of a segment based on its minimum area bounding box: $1 - \frac{\text{boxWidth}}{\text{boxHeight}}$. A square is the least elongated.

Centrality Euclidean distance from the segment's centroid to the center of the pattern.

Role Labels A set of three binary values: *Noise*, *Background*, *Foreground*. *Noise* indicates if the segment was labeled as 'noise' during preprocessing (Section 4). *Background* indicates if a segment belongs to the group with the largest connected component. All other segments are labeled *Foreground*.

5.2 Color Property Distributions

With a set of color properties and predictive spatial features in hand, our goal is to predict distributions over those properties given the features. Then, for a color group g and a color property π, we define the scoring function:

$$\phi_\pi^{\text{Grp}}(\mathbf{c}_g) = \ln p(\pi(\mathbf{c}_g)|\mathbf{f}_g) \cdot A_g$$

where $\mathbf{c}_g$ is the color of group g and $\mathbf{f}_g$ are its spatial features. We weight by the area of the group A_g as larger regions tend to have more impact on the appearance of a coloring. Similarly, for a segment s, we define the function $\phi_\pi^{\text{Seg}}(\mathbf{c}_s) = \ln p(\pi(\mathbf{c}_s)|\mathbf{f}_s) \cdot A_s$. Evaluating these functions on a particular color results in a 'score' for how well that color fits the given group or segment, according to the property π.

Figure 3 shows predicted distributions over lightness for different pattern regions using training data from the top 10 artists in our dataset. The background color group, which is larger and more spread, exhibits a bimodal distribution. Intuitively, backgrounds are either dark or light but rarely of middling brightness. The distribution favors light backgrounds, reflecting the stylistic biases of the data used for training. The smaller, more concentrated, foreground flower color group, on the other hand, strictly prefers lighter colors.

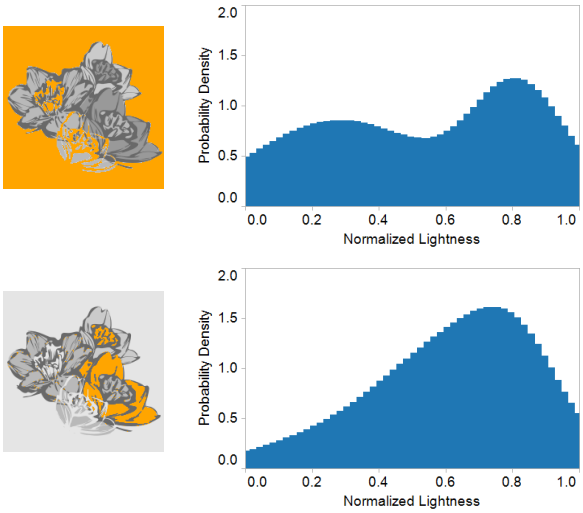


Figure 3: *Predicted distributions over lightness for two different color groups (higlighted in orange). The background has a bimodal distribution, whereas the foreground strictly favors lighter colors.*

How should we represent these distributions p? Closed-form continuous distributions, such as the normal distribution, are appealing for their simplicity. However, they are unlikely to capture the shape of distributions exhibited by real patterns, which are often *multimodal* in nature (Figure 3).

We adapt the method of Charpiat et al. [2008] to build multimodal distributions of color properties. We first discretize the space of possible color property values into a finite number of bins. Next, we train a multi-class classifier on $(\pi(\mathbf{c}), \mathbf{f})$ pairs extracted from the training dataset. This classifier predicts, given a feature vector $\mathbf{f}$, the probability that its corresponding property value $\pi(\mathbf{c})$ falls into each bin. Given a never-before-seen feature vector, the classifier can then output a *histogram* of these probabilities, one for each property value bin. The histogram is then smoothed using kernel density estimation, and the resulting density forms the final, continuous probability distribution.

In our implementation, we discretize the space of property values using k-means clustering with $k = 10$ on the values found in the training examples. We then use multinomial logistic regression to predict the histograms of color property values given features. Finally, we smooth the histograms by placing a Gaussian at the center of each histogram bin and setting the Gaussian bandwidth to the average distance to the nearest three other bins [Wang et al. 2010].

6 Pairwise Color Functions

While group and segment terms model the dependency of color assignments on spatial features of same-color regions, they do not capture relationships between different-color regions. Adjacent

color regions can have strong effects on their neighbor's perceived color, making colors appear more or less saturated or causing vibrating boundaries [Albers 1963]. Thus, we also predict distributions over color properties for adjacent segment pairs.

6.1 Color Properties and Predictive Features

As with individual pattern regions, there are many possible properties of the color relationship between two adjacent regions that could influence the appearance of a pattern coloring. Our method uses the following set:

Perceptual Difference is the Euclidean distance between two colors in $L^*a^*b^*$ space and is the primary descriptor of 'contrast' between two colors that we use in our model. This distance metric is simple and efficient to evaluate; more sophisticated formulae have also been proposed [Sharma et al. 2005].

Relative Lightness is the absolute difference between the L values of two colors in $L^*a^*b^*$ space. This 'difference of intensities' captures another important type of contrast.

Relative Saturation is the absolute difference between the saturation values of two colors, using the definition from Section 5. This property helps capture whether or not two colors should be mutually saturated/desaturated

Chromatic Difference is the squared fraction of perceptual distance due to the $L^*a^*b^*$ chroma channels: $\frac{\delta a^2+\delta b^2}{\delta a^2+\delta b^2+\delta L^2}$. This value measures the difference between two colors after factoring out lightness.

Color Name Similarity is the cosine similarity between the color name count vectors defined in Section 5 [Heer and Stone 2012]. This measure assesses whether two colors are typically referred to with the same set of names.

Good color assignments may depend on the sizes of participating regions and the nature of their adjacency. For example, a square enclosed by a thin border appears different from a square enclosed by a thick border, and different again from a square side-by-side with another square (Figure 4). Thus, to form a set of predictive spatial features for an adjacent segment pair, we use the features from both participating segments, concatenated such that the one with the smaller L_2 norm is first to enforce a consistent ordering. In addition, we add a pair of features we call **Enclosure Strengths**, which measure how much one segment in the adjacency encloses the other and vice versa. Enclosure Strength is defined as the number of pixels of the neighboring segment appearing within a 2-pixel neighborhood outside the segment's boundary, normalized by the area of that neighborhood. Out-of-image pixels are counted as part of the neighborhood area.

Figure 4: *Color appearance depends on relationships with surrounding regions.*

6.2 Color Property Distributions

For a particular pair of adjacent segments (s, s') and a color property π, we can define a scoring function:

$$\phi_\pi^{\text{Adj}}(\mathbf{c}_s, \mathbf{c}_{s'}) = \ln p(\pi(\mathbf{c}_s, \mathbf{c}_{s'})|\mathbf{f}_{s,s'}) \cdot \texttt{str}(s, s')$$

where $\texttt{str}(s, s')$ is the *strength* of the adjacency (s, s'). We define adjacency strength as the number of pixels from segments s or s' that are within a 2-pixel distance from their perimeters. All adjacency strengths in a given pattern are normalized to sum to 1. We learn the distributions p using the 'histogram regression' approach described in Section 5. This function scores how well a color assignment fits an adjacency according to color property π.

Figure 5 shows predicted distributions over relative lightness for different adjacent segment pairs. The two distributions are similar in shape and reflect the intuition that no two adjacent segments should be equi-luminant. However, the adjacency between the foreground flower and the background concentrates more mass toward higher lightness differences. Together, these two distributions suggest that foreground-background adjacencies should exhibit more lightness contrast than foreground-foreground adjacencies.

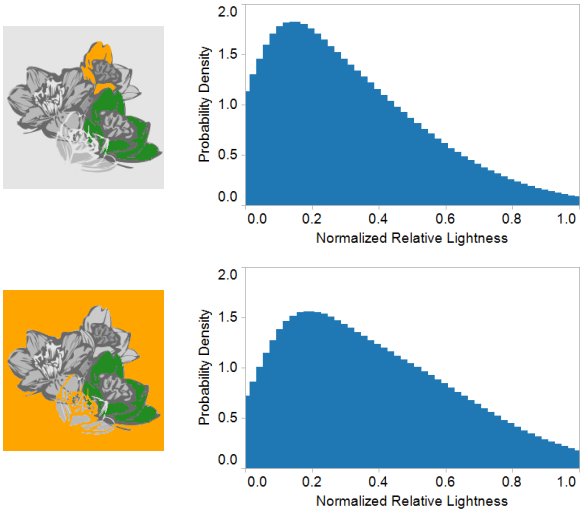


Figure 5: *Predicted distributions over relative lightness for two different segment adjacencies (participating segments higlighted in orange and green). A value of 0 indicates identical lightness. The foreground-foreground distribution permits more similar lightness values than the foreground-background distribution.*

7 Color Compatibility Function

The unary and pairwise color scoring functions defined in the previous sections indicate whether an individual pattern region is well-colored or whether two adjacent regions are colored well in concert, but they have no knowledge of the global harmony between all colors in the pattern. To enforce global consistency, we include a color compatibility function based on the model introduced by O'Donovan et al. [2011]. Their model predicts 0-5 numeric aesthetic ratings for five-color 'color themes,' which are ordered rows of five colors.

We extract such a color theme from a pattern by taking the colors of the five largest color groups and ordering them by size. If the pattern contains fewer than five color groups, we repeat colors in order of size to fill the rest of the theme. Inspection of these extracted themes revealed that size-ordering of colors tends to produce themes that are rated higher than random orderings but lower than the optimal ordering. Additionally, ordering has little effect on discriminative power: in general, low-scoring themes are rated lower than high-scoring themes, regardless of permutation.

To turn a theme's predicted rating into a score consistent with our unary and pairwise functions, we divide the rating by the maximum

possible rating (5) and treat the result as a probability:

$$\phi^{\text{Compat}}(\mathbf{c}_1 \dots \mathbf{c}_5) = \ln(\texttt{compat}(\mathbf{c}_1 \dots \mathbf{c}_5)/5)$$

where $\mathbf{c}_1 \dots \mathbf{c}_5$ are the colors of the five largest color groups in the pattern and `compat` is the O'Donovan color compatibility model. While our implementation uses the O'Donovan model of color compatibility, it is flexible enough to accomodate any other color compatibility that can assign a score to a set of colors.

8 Probabilistic Model

We have defined unary, pairwise, and global scoring functions for pattern color properties. Now, we combine them into one scoring function that evaluates the overall quality of a pattern coloring.

For this task, we turn to the language of *probabilistic factor graphs*. A factor graph is a probabilistic graphical model that decomposes a complex probability distribution over multiple variables into a set of smaller *factors* over subsets of the variables. In our case, the variables $\mathbf{C}$ are the colors of each color group in a pattern, and the factors $\mathcal{F}$ are derived from our scoring functions. Figure 6 shows an example of a factor graph for one simple pattern. The circles denote color variables, while squares denote different factors $\mathcal{F}$. Edges in the graph connect each factor to the variables within its scope.

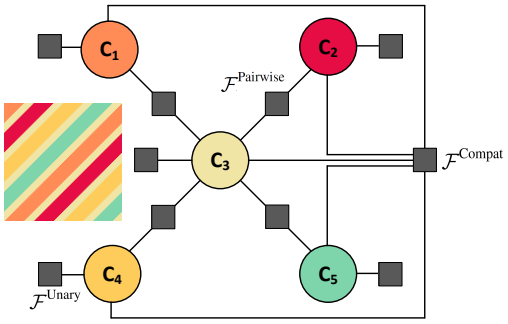


Figure 6: *A factor graph for an example pattern template. Variable nodes are colored according to their corresponding color group in the pattern.*

The factors connected to a single variable are derived from our unary scoring functions; they combine the score for a color group with the scores for all segments in that group:

$$\mathcal{F}_\pi^{\text{Unary}}(\mathbf{c}_g) = \exp(w_\pi^{\text{Grp}} \cdot \phi_\pi^{\text{Grp}}(\mathbf{c}_g) + w_\pi^{\text{Seg}} \cdot \sum_{s \in g} \phi_\pi^{\text{Seg}}(\mathbf{c}_g))$$

The w's are weights that control the relative importance of each function; we will see how to set them later in this section.

The factors connecting two variables come from our pairwise scoring functions and combine the scores for all adjacencies which involve segments from two different color groups:

$$\mathcal{F}_\pi^{\text{Pairwise}}(\mathbf{c}_g, \mathbf{c}_{g'}) = \exp(w_\pi^{\text{Adj}} \cdot \sum_{(s,s') \in \text{adj}(g,g')} \phi_\pi^{\text{Adj}}(\mathbf{c}_g, \mathbf{c}_{g'}))$$

Finally, the factor connected to all five color variables enforces color compatibility:

$$\mathcal{F}^{\text{Compat}}(\mathbf{c}_1 \dots \mathbf{c}_5) = \exp(w^{\text{Compat}} \cdot \phi^{\text{Compat}}(\mathbf{c}_1 \dots \mathbf{c}_5))$$

The probability distribution encoded by this factor graph is the normalized product of all of these factors:

$$p(\mathbf{c}|\mathcal{P} : \mathbf{w}) = \frac{1}{Z(\mathcal{P} : \mathbf{w})} \prod_{\mathcal{F}} \mathcal{F}(\mathit{Scope}_{\mathcal{F}}(\mathbf{c}))$$

Here, $\mathit{Scope}_{\mathcal{F}}$ selects the color variables connected to the factor $\mathcal{F}$ and $Z(\mathcal{P} : \mathbf{w})$ is the pattern-dependent partition function that normalizes the distribution. The distribution is parameterized by the vector of factor weights $\mathbf{w}$.

8.1 Sampling

Generating good coloring suggestions reduces to sampling high-probability colorings from our model. We use the Metropolis-Hastings algorithm (MH), a variant of Markov Chain Monte Carlo (MCMC) [Metropolis et al. 1953; Hastings 1970]. MH explores the coloring state space by *proposing* candidate new states, which are accepted with probability proportional to their model score. We would also like our sampler to output a variety of suggestions, which requires that it explore many modes of the distribution. To do this efficiently, we use parallel tempering, a technique that runs multiple MCMC chains in parallel at different 'temperatures' and swaps their states periodically [Geyer 1991]. 'Hot' chains are more likely to take large jumps across the state space, whereas 'cool' chains behave like local hill-climbing optimizers. The combined system of chains effectively explores and refines different coloring configurations.

Our sampler uses the following MH proposals:

- **Perturb** a randomly chosen color by $v \sim \mathcal{N}(0, \sigma)$ in RGB space
- **Swap** two randomly chosen colors

where σ varies linearly with the model temperature t, encouraging larger perturbations at high temperatures. The sampler chooses between these two proposals with a probability that also varies linearly with temperature. The sampler operates in RGB space rather than $L^*a^*b^*$ space, since all RGB colors fall in the display gamut, and the sampler should not waste time exploring colorings that cannot be visualized. Since the RGB color space is bounded, the perturbation proposal draws from a truncated normal distribution in order to maintain ergodicity of the MCMC chains [Robert 1995].

Finally, we use maximimum marginal relevance (MMR) to enforce diversity in the set of suggestions returned by the sampler [Carbonell and Goldstein 1998]. MMR is a technique from information retrieval that re-ranks every item in a list according to a linear combination of relevance (model score, in our case) and similarity to the items preceding it. The similarity metric we use for two colorings $\mathbf{c}$ and $\tilde{\mathbf{c}}$ of a pattern is $-\sum_{g \in \mathcal{G}} A_g \cdot ||\mathbf{c}_g - \tilde{\mathbf{c}}_g||$, which is the area-weighted sum of $L^*a^*b^*$ distances between the corresponding colors in each coloring.

8.2 Weight Learning

The factor graph model we have defined is parameterized by a vector of weights $\mathbf{w}$. Setting these weights manually proves challenging, as it is not obvious which color properties matter most to the quality of a pattern coloring. Instead, we would like to set them automatically, using our training dataset as a guide.

We formulate the weight-tuning problem as one of *maximum likelihood parameter estimation*: we would like to set the weights such that the training examples have high probability under the resulting model. We first rewrite the probability distribution encoded by our model from a weight-centric view

$$p(\mathbf{c}|\mathcal{P} : \mathbf{w}) = \frac{1}{Z(\mathcal{P} : \mathbf{w})} \prod_{w \in \mathbf{w}} \exp(w \cdot \Phi_w(\mathbf{c}, \mathcal{P}))$$

where $\Phi_w(\mathbf{c}, \mathcal{P})$ sums all the scoring functions ϕ that share the weight w. We can then express the log-likelihood of the weights

given a dataset $\mathcal{D}$ of pattern colorings:

$$\ell(\mathbf{w} : \mathcal{D}) = \sum_{(\mathcal{P},\mathbf{c})\in\mathcal{D}} \left(\sum_{w\in\mathbf{w}} w \cdot \Phi_w(\mathbf{c},\mathcal{P})\right) - \ln Z(\mathcal{P} : \mathbf{w})$$

Convex log-likelihoods such as this are typically maximized via gradient ascent. The partial derivatives of this function with respect to the weights are

$$\frac{\partial}{\partial w}\ell(\mathbf{w} : \mathcal{D}) = \sum_{(\mathcal{P},\mathbf{c})\in\mathcal{D}} \Phi_w(\mathbf{c},\mathcal{P}) - \mathbb{E}_{\mathbf{w}}[\Phi_w(\mathbf{C},\mathcal{P})]$$

where $\mathbb{E}_{\mathbf{w}}$ denotes an expectation under the model with weights $\mathbf{w}$. Unfortunately, these quantities are extremely expensive to compute: the expectation term requires probabilistic inference—an NP-complete problem—for every training pattern, for every iteration of gradient ascent.

This computational intractability has motivated the development of alternative, 'biased' parameter estimation schemes which do not directly maximize the likelihood function but nevertheless yield parameters that give high likelihoods. We use one such method called *Contrastive Divergence* (CD) [Hinton 2002]. CD uses the following approximation to the likelihood gradient:

$$CD_w^k(\mathbf{w} : \mathcal{D}) = \sum_{(\mathcal{P},\mathbf{c})\in\mathcal{D}} \Phi_w(\mathbf{c},\mathcal{P}) - \Phi_w(\hat{\mathbf{c}},\mathcal{P})$$

where $\hat{\mathbf{c}}$ is the coloring obtained by running an MCMC chain for k steps from the initial coloring $\mathbf{c}$. CD forms a local approximation to the likelihood gradient around the neighborhood of $\mathbf{c}$. Larger k yields more accurate approximations at additional cost; we use $k = 10$. We initialize the weights uniformly to 1 and constrain them to be non-negative, since all terms in the model are log-probabilities.

While the exact weights learned depend on the training dataset, we have noticed several persistent trends. The perceptual difference, color compatibility, and color name count terms receive the highest weights. These trends coincide well with our intuition that colors should be harmonious, adjacent regions should have sufficient contrast, and colors should be categorically similar to those in the training set. The lowest weight belongs to the color name similarity term, which suggests that the similarity in how two adjacent colors are named is not strongly predictive of their compatibility.

8.3 Implementation

Our prototype implementation of this model is written in the Scala programming language, using the Factorie toolkit for probabilistic modeling [Mccallum et al. 2009]. To evaluate the color compatibility term, it uses the reference MATLAB implementation provided by O'Donovan et al. [2011].

9 Results

In this section, we demonstrate the flexibility of our model when applied to a variety of pattern coloring scenarios. Results were generated using a model trained on patterns from the 10 most popular artists in our dataset, except for patterns with a template used in a result figure or experiment; the training patterns and final weights for each of our factors can be found in the supplemental materials. To sample from the model, we used parallel tempering with 5 chains at temperatures $(1, 0.5, 0.2, 0.05, 0.01)$, swapping colors between chains every 50 iterations. We then used MMR to retrieve a diverse set of samples from the resulting MCMC chains. Images were rendered using the COLOURlovers online pattern creator, from color assignments generated by our model.

9.1 Coloring Pattern Templates

People can color patterns with a variety of different criteria in mind, which may even change during the coloring process. These preferences can range from the more general preference for any appealing coloring to more specific preferences such as using a particular palette, looking for local improvements to an existing coloring, or matching a specific style. We explore how our model can facilitate these scenarios.

Automatic pattern coloring In the most direct application of our framework, we can sample from our model to produce colorings for a pattern template that are similar to the colorings used for training. Figure 1 shows two examples of this process. The sampled patterns exhibit a range of colors and styles employed by the COLOURLovers artists. For comparison, the same patterns colored with palettes randomly sampled from RGB-space are shown on the right. These patterns exhibit significant problems, such as low color harmony and adjacent regions with equi-luminant colors.

Coloring with fixed palettes Artists often draw inspiration from inspirational photographs and existing color themes and may have specific colors in mind when coloring in a pattern. Even with a fixed color palette, mapping different colors to different regions leads to a range of possible images, some more desirable than others. We can use our model's score to rank all possible permutations of the palette colors (excluding the color compatibility factor).

Figure 7 shows an example where colors from two photographs are applied to different pattern templates. A color palette is first automatically extracted from the input photograph [Lin and Hanrahan 2013]. Our algorithm then automatically computes a pleasing mapping from the colors in the extracted palette to regions in a given pattern template.

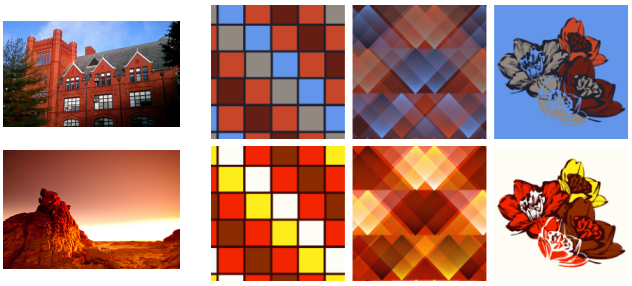


Figure 7: *Palettes are extracted from the two input photographs on the left, and on the right we show the top-scoring coloring suggested by our algorithm for three different pattern templates.*

Although using a well-designed color palette often improves the look of a pattern, the spatial arrangement of those colors can make a dramatic difference in appearance. Figure 8 shows two input color palettes for the bird pattern and the highest and lowest-scoring color assignments to the pattern using those palettes. These palettes were originally used by artists to color in this pattern on COLOURlovers. Because the pattern has 5 color groups, there are 120 possible color assignments for a given 5-color palette. Our model ranks the original artist-created color assignments as 39th (for the first palette) and 21st (for the second) out of the 120 permutations. Over a larger sampling of 188 patterns, artist-created colorings rank in the top 40% overall, suggesting that the model captures general artist trends in pattern coloring. However, there may be stylistic variations between different artists that are not well predicted by a single model.

(a) Input Palettes **(b)** Highest-scoring assignments **(c)** Lowest-scoring assignments

Figure 8: *Given a pattern template and corresponding palette as input, we use our color model to compute the score of each possible assignment of the palette to the image regions.* ***(b)*** *and* ***(c)*** *show the top-3 and bottom-3 assignments for two different palettes.*

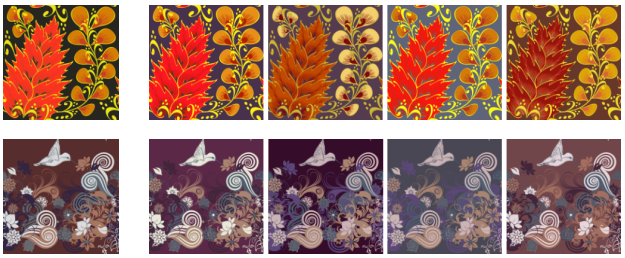
Initial automatic suggestions After fixing the stem to be green

Figure 9: *An artist coloring a pattern is presented with the results shown on the left, and decides that she only likes results where the stem of the plant is dark green. On the right, we use conditional inference to sample from our model subject to the constraint that the desired palette entry is fixed to a specific color.*

In this example, the model terms that most distinguish the highest-scoring assignments from the lowest-scoring ones are the color name counts factors. The model prefers colors such as Tan-Yellow, Grey, and Green-Black to Red-Orange and Orange for background regions. Colorings with orange background colors tend to score poorly, perhaps because there are few popular patterns with orange backgrounds in our training set. Our model also predicts that foreground regions should more perceptually distinct from the background than from other foreground regions. Thus, it tends to assign the most distinct color in the palette to the background region.

Hard color constraints In some cases, a user may only have partial knowledge about the colors she wants to see in particular pattern regions. Our model assists this situation by allowing users to fix the colors of certain pattern regions. The system then uses conditional inference to sample values for the remaining unconstrained color variables. Figure 9 shows an example. After viewing an initial set of coloring suggestions, the user decides she wants to see more results where the stem of the plant is a particular shade of green. Our system outputs new suggestions subject to this constraint.

Soft color constraints In another use case, a user might have strong preferences for the appearance of the pattern. She might manually color a pattern and then wish to see variations upon this theme, hoping to find better-looking alternatives nearby in the space of colorings. Our model can support this type of query by incorporating an additional soft constraint factor on color variable, constraining it to be close to a target color:



Original Suggestions

Figure 10: *An artist provides an initial color assignment and asks for patterns that are similar. We incorporate this request by adding an additional factor to our model, showing four samples drawn from the new model for each of the input images.*

$$\mathcal{F}^{\text{Target}}(\mathbf{C}_g|\mathcal{P}) = \mathcal{N}(||\mathbf{C}_g - \text{targetColor}(g)||, \sigma_{\text{user}}) \quad (1)$$

Here, $\text{targetColor}(g)$ is the desired color of group g and σ_{user} controls the extent to which the group is allowed to deviate from the desired color. We assign this factor a weight $w_{\text{user}} * w_{\text{model}}$, where w_{model} is the sum of the weights of all other factors in the model. w_{user} controls the tradeoff between satisfying the user-specified target colors for a region versus satisfying the color distribution encoded by the trained model.

Figure 10 shows two example scenarios where a proposed coloring is given and the model returns similar colorings. Here we choose

σ_{user} to be 10% of the diagonal length of the L*a*b* color space, and $w_{\text{user}} = 3$, indicating a desire to remain fairly close to the original coloring. Our model generates numerous variations on the input coloring while still preserving its overall character.

Style capture A powerful advantage of a data-driven approach is the ability to modify the underlying training source to achieve specialization of the resulting model. This ability allows our model to capture a specific style and color preferences such as "high-contrast patterns" by selecting a set of patterns with the desired property. Figure 11 demonstrates this behavior using four style categories: *Light*, *Dark*, *Bold*, and *Mellow*. Using 17 training examples, our model can capture general properties of the example patterns such as the distribution of colors over the background regions in *Light* and *Dark* and the level of contrast in *Bold* and *Mellow*. The model can also capture the style of a specific artist, as shown in Figure 12. Here, 100 images from each artist were used for training. The sampled images mimic properties of each artist's style, such as the light backgrounds preferred by Artist A and the bold colors and dark backgrounds preferred by Artist B. The complete list of patterns used as training data for these examples can be found in the supplemental materials.

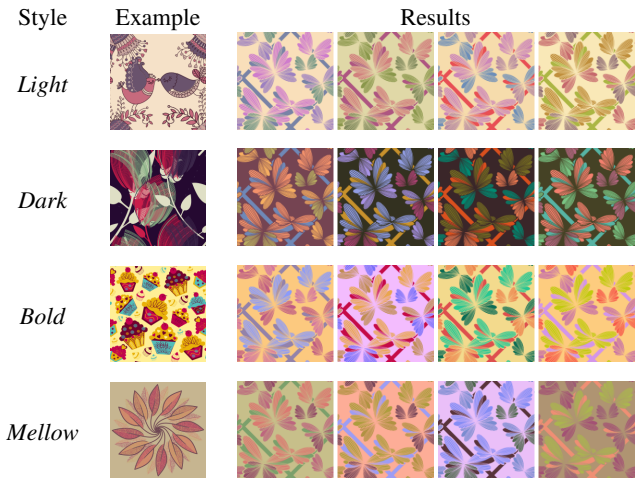


Figure 11: *Training different models on patterns with different styles. (Left) A representative pattern from each style. (Right) The top four samples drawn from each model.*

9.2 Applications

Next, we show how our model can be applied to different pattern coloring tasks, such as coordinating colors in interior scene design, coloring in webpage backgrounds, and suggesting clothing colorings to fit people with different physical attributes.

3D scene design Color-coordinating a virtual scene is another creative task that can benefit from computatational support. While repositories such as the 3D Warehouse provide a wealth of object models with which to build scenes, such models are rarely color coordinated with one another. Figure 13 shows how our model can make coordinating object colors easier. We can treat a scene as a recolorable pattern by rendering its material coefficients to an image. The user then provides a few additional annotations to specify which object components should be the same color; in this example, cushions, pillows, and wooden objects must take the same color. Finally, we add a factor to our model to enforce that the colors assigned to each component are similar to their original colors (see Equation 1) to keep the system from wandering into physically implausible colorings. The resulting model suggests plausible, improved recolorings of the scene. Recent related work transfers material properties from photographs to 3D scenes [Nguyen et al. 2012]; the objective functions used in this system could be adapted to soft constraints in our framework.

Web design A web designer may want to change a web page's color theme to fit the season, a special event, or even the time of day. Our model can support these tasks by suggesting recolorings for web page pattern elements. In Figure 14, the background pattern of a blog is automatically recolored in three distinct styles. Each style is defined by a manually-specified page body color and header text color. We generate these recolorings by adding new factors to the model. First, we add a unary factor to each color variable that encourages similarity to a color that occurs in the web page body:

$$\mathcal{F}^{\text{WebSim}}(\mathbf{c}_g|\mathcal{W}) = p(\mathbf{c}_g|\mathcal{W})$$

where $\mathcal{W}$ is a rendered image of the web page body. The distribution p is represented with a smoothed histogram of colors in the image, after quantizing to a small number of colors (20, in this experiment). We also add a factor to the pattern's background color group that penalizes similarity to the web page body background color:

$$\mathcal{F}^{\text{BgPen}}(\mathbf{c}_{\text{largest}(\mathcal{G})}|\mathcal{W}) = 0.05 \cdot ||\mathbf{c}_{\text{largest}(\mathcal{G})} - \text{bgColor}(\mathcal{W})||$$

This term ensures that the body of the page does not blend into the patterned background. A more sophisticated approach would treat the entire web page as a pattern and perform joint recoloring; we leave this extension to future work.

Fashion design Our model can also adapt the colorings of clothing items to fit different people. Figure 15 shows a shirt recolored to match different hair, eye, and skin tones. The hair, skin, eyes, and lips of the 3D person model were manually quantized to a single color; the resulting texture atlas functions as a recolorable pattern template. Fixing the colors of the hair, skin, eyes, and lips constrains the inference process to produce suitable colorings for a person with those physical attributes.

9.3 Performance

The time required to generate coloring suggestions varies depending on the visual complexity of the pattern. With the parallel tempering parameters described at the beginning of this section, it took 73.0s on average to retrieve 20 MMR-diversified results for a single input pattern on a 2.67GHz Intel Core i7. Running time is dominated by MCMC sampling, which on average accounts for 83% of the total time.

Real-time coloring suggestions would benefit many of the applications demonstrated in this paper, and there are several avenues for improving the performance of our unoptimized, JVM-based prototype to this end. We could leverage the massive parallelism of graphics hardware to speed up parallel tempering, as was done in related work on automatic furniture layout [Merrell et al. 2011]. We could also improve the convergence of the individual MCMC chains via gradient-based proposals, as in Hamiltonian MCMC [Neal 2010].

10 Evaluation

We conducted a judgement study to better understand how coloring suggestions from our model compare to colorings from simpler models as well as colorings made by artists. We recruited 16

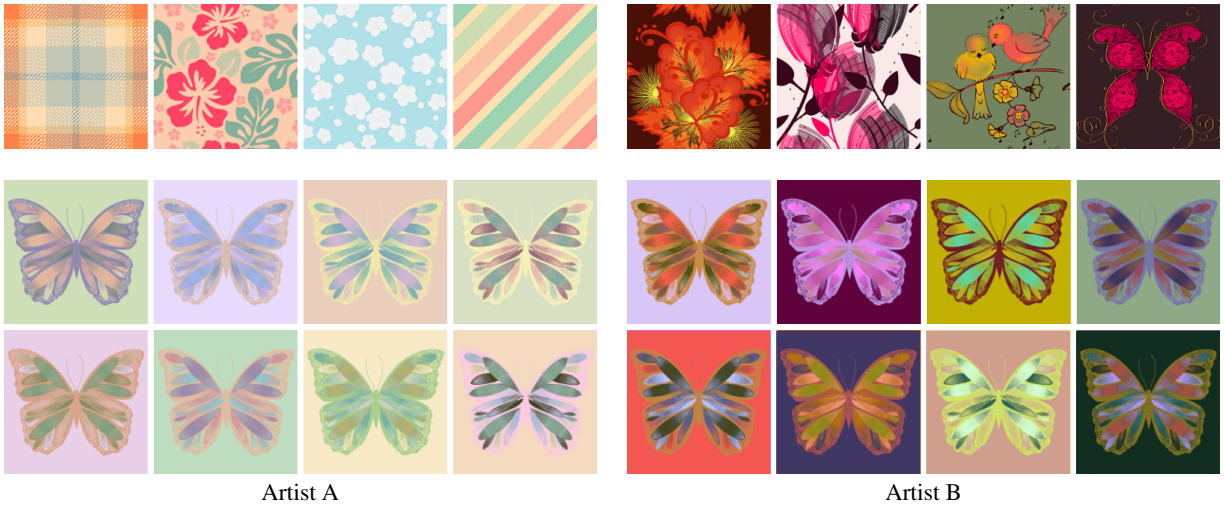


Figure 12: *Our data-driven approach makes it easy to capture the styles of different artists. Top: representative images from two different artists. Bottom: results sampled from a model trained on 100 images from the artist.*

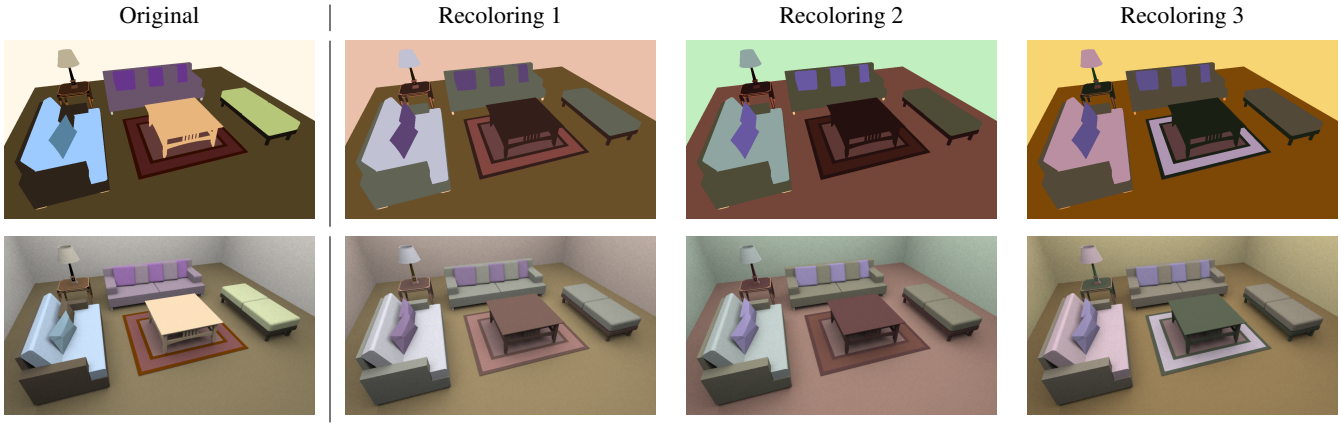


Figure 13: *A poorly color-coordinated 3D scene is recolored using our model. Using the original coloring as a soft constraint, our model suggests multiple novel recolorings. (Top) Diffuse material coefficients for each object. (Bottom) Final renderings.*

Computer Science graduate students (5 female) to participate in the study. All participants had normal color vision. To simulate the exploratory situations in which our model would be used, we generate and present multiple coloring suggestions. Although participants likely have different aesthetic preferences, we should see any general trends in the preferability of colorings generated by different methods.

We first generate coloring suggestions from four different sources—-*Artist* colorings, our *Model*, a *Color Compatibility*-only model using the measure by O'Donovan et. al. [2011], and *Random* colorings—to be compared in the study. Next, we select 15 different pattern templates that do not have strong semantic associations and are also outside of our training set. For each pattern template, we then generate 4 coloring suggestions per source. For the *Artist* source, we randomly choose 4 colorings from the top 45 artist colorings on COLOURlovers. For our *Model*, we sample colors using parallel tempering for 2000 iterations and choose the top 4 results using MMR with $\lambda = 0.5$. We similarly sample *Color Compatibility*-only colorings. Finally, for the *Random* source, we generate 4 colorings uniformly at random.

The study interface presents participants with a randomized grid of all 16 coloring suggestions for a pattern template. Participants are asked to choose the 4 colorings they like the most and the 4 they like the least from the grid before moving on to the next template. Each participant responds to 5 pattern templates randomly drawn from the pool of 15 and presented in random order. One template is presented twice to check for participant consistency.

Figure 16 shows the percentage of suggestions from each source that particpants chose as a 'Top 4' or 'Bottom 4' pattern. When counting, we do not include suggestions from the replicated template. Logistic regression shows that the source of a suggestion significantly affects the chances that it will be chosen as either 'Top 4' or 'Bottom 4' coloring. Tukey all-pair comparison tests show that all differences are significant ($p < 0.01$), except for the odds that an *Artist* suggestion will be chosen as a 'Bottom 4' suggestion versus a *Model* suggestion. *Artist* patterns are selected most often as a 'Top 4' pattern, with our *Model* patterns selected second most often. In addition, *Model* patterns are selected least as a 'Bottom 4' pattern, along with *Artist* patterns.

These results suggest that *Model*-generated patterns have significantly higher quality than patterns generated by automatic baselines. They do not yet achieve the same quality as *Artist*-created patterns, but since our model functions as part of an interactive coloring workflow, this result is acceptable. This research does not seek to replace human artists but rather to augment their abilities. The result that *Model* patterns are 'disliked' as infrequently

Figure 14: *With a few additional factors, our model can recolor the background pattern of a web page to match the colors in the page body.*

Figure 15: *Using our model to colorize a patterned shirt for different people. The hair, skin, eye, and lip colors are treated as fixed constraints, which encourage the model to produce compatible colorings via adjacencies and long-range dependencies.*

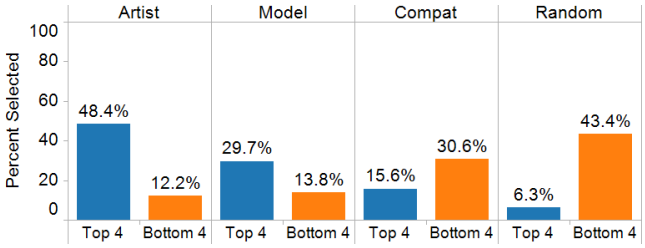


Figure 16: *The percentage of times that* Artist*-created,* Model*-generated,* Color Compatibility*-only, and* Random *patterns were chosen by participants as one of their 'Top 4' favorite or 'Bottom 4' least favorite patterns in our online experiment.*

as *Artist* patterns suggests that the *Model* patterns not chosen as favorites are likely good enough to serve as inspirational starting points for further creative work.

Finally, different groups of people likely have different biases in color preference, which is important to research in automatic coloring and color suggestion. Further experiments with different populations are needed to understand these biases.

11 Discussion and Future Work

In this paper, we present a probabilistic approach to automatically coloring 2D patterns. We develop a factor graph model that is trained on example pattern colorings to statistically capture their coloring style and sampled using MCMC to generate a variety of pattern coloring suggestions. We demonstrate the utility of the model on a range of coloring tasks, and a perceptual study showed that the colorings it generates compare favorably to those generated by other automatic methods.

There is still much work to be done to understand what makes a pattern coloring appealing. In this paper, we propose modeling one plausible set of color properties, and we gain some insight into which properties matter most by comparing their automatically-tuned weights. While colorings generated by our model are attractive and useful, our evaluation shows that they do not yet achieve the same quality as colorings created by human artists. More research is needed to close this gap, and our probabilistic framework provides a strong and flexible foundation for further investigation.

One limitation of the current model is that it does not encode any semantic constraints on pattern regions, such as skies being blue or plants being green. This is not always a problem, as even human artists do not always respect these semantics, and patterns often admit many attractive, non-semantic colorings. However, some patterns can appear odd or confusing when their colorings do not respect semantics. Labeling regions with semantic tags could address this problem, as the system could search for images on the web using these tags and build up a distribution of expected colors. It might also be possible to predict such labels automatically using sketch classification techniques [Eitz et al. 2012].

In addition, some COLOURlovers templates use extensive color blending in their original vector artwork. For these patterns, the rasterized image is a poor approximation to the original artwork, and our model may not capture good spatial features. In the future, we would like to apply our model to a dataset of vector patterns.

This research opens up several interesting avenues for future work. First, our experiments used patterns from COLOURlovers or renderings that can be easily coerced into the same format. What if any image found 'in the wild' could serve as a recolorable template? As a first approach, we could extract a color theme from the image, and then quantize the image using the theme's colors [Lin and Hanrahan 2013]. However, this simple approach is unlikely to perform well on complex images that use many distinct colors or color blending. Solving the problem in general requires further research.

Second, this work explores colorizing patterns of the 'color-by-numbers' format, in which some pattern regions are constrained to take the same color. What if this assumption were relaxed, and the color groups in the pattern template are unknown? A system that supported this more general type of template could operate on essentially any segmented image. This requires a more sophisticated model, as well as transdimensional inference techniques to generate colorings with a variable number of color groups [Yeh et al. 2012b].

Finally, embedding automatic coloring suggestions into interactive creative software presents an exciting opportunity. A vector art program could continually generate suggestions in the background while its user works on a pattern; the user could view the suggestions at any time, potentially picking one to use as an 'autocompletion' of her work-in-progress. These kinds of tools have the potential to drastically shift the way that artists and enthusiasts work with color on a daily basis.

Acknowledgments

Support for this research was provided by Intel (ISTC-VC) and an SAP Stanford Graduate Fellowship. We would also like to thank COLOURLovers *jilbert* (Fig 1,2), *symea* (Fig 1, 10), *ArrayOfLilly* (Fig 3,5,7), *COLOURLover* (Fig 6, 12), *timanttimaarit* and *dazzlement* (Fig 7), *Any Palacios* (Fig 8-12), *gregreis* [Gregorio Reis] and *praxicalidocious* (Fig 11), *bhsav* and *magg* (Fig 11, 12), *vaneea* and *casslovescolors* (Fig 12), *ivy21* (Fig 14), and *caseycastille* (Fig 15) for their pattern templates; *AlineDam* (Fig 11), and *sugar!* and *albenaj* (Fig 12) for the pattern coloring examples; and Flickr users *marctasman* and *zoomion* for the reference photographs (Fig 7).

References

ALBERS, J. 1963. The interaction of color. *Art news 62*, 1.

BRIBIESCA, E. 1997. Measuring 2-d shape compactness using the contact perimeter. *Computers & Mathematics with Applications 33*, 11.

CARBONELL, J., AND GOLDSTEIN, J. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. ACM SIGIR 1998*.

CHARPIAT, G., HOFMANN, M., AND SCHÖLKOPF, B. 2008. Automatic image colorization via multimodal predictions. In *Proc. ECCV 2008*.

COHEN-OR, D., SORKINE, O., GAL, R., LEYVAND, T., AND XU, Y.-Q. 2006. Color harmonization. In *Proc. SIGGRAPH 2006*.

DATTA, R., JOSHI, D., LI, J., AND WANG, J. Z. 2006. Studying aesthetics in photographic images using a computational approach. In *Proc. ECCV 2006*.

EITZ, M., HAYS, J., AND ALEXA, M. 2012. How do humans sketch objects? In *Proc. SIGGRAPH 2012*.

GEYER, C. 1991. Markov chain monte carlo maximum likelihood. In *Proc. of the 23rd Symposium on the Inteface: Computing Science and Statistics*, 156–163.

HASTINGS, W. K. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika 57*, 1.

HEER, J., AND STONE, M. 2012. Color naming models for color selection, image editing and palette design. In *Proc. ACM CHI 2012*.

HINTON, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*.

ITTEN, J. 1974. *The Art of Color: The Subjective Experience and Objective Rationale of Color*. Wiley.

JAIN, A., THORMÄHLEN, T., RITSCHEL, T., AND SEIDEL, H.-P. 2012. Material memex: automatic material suggestions for 3d objects. In *Proc. SIGGRAPH Asia 2012*.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. In *Proc. SIGGRAPH 2004*.

LIN, S., AND HANRAHAN, P. 2013. Modeling how people extract color themes from images. In *Proc. of CHI 2013*, ACM, New York, NY, USA, CHI '13.

LÜBBE, E. 2010. *Colours in the Mind - Colour Systems in Reality*. Books on Demand.

MCCALLUM, A., SCHULTZ, K., AND SINGH, S. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Proc. NIPS 2009*.

MEIER, B., SPALTER, A., AND KARELITZ, D. 2004. Interactive color palette tools. *Computer Graphics and Applications 24*, 3.

MERRELL, P., SCHKUFZA, E., LI, Z., AGRAWALA, M., AND KOLTUN, V. 2011. Interactive furniture layout using interior design guidelines. In *Proc. SIGGRAPH 2011*.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., AND TELLER, E. 1953. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics 21*, 6.

MUNSELL, A., AND BIRREN, F. 1969. *A Grammar of Color: A Basic Treatise on the Color System of Albert H. Munsell*. Van Nostrand Reinhold.

NEAL, R. M. 2010. Mcmc using hamiltonian dynamics. In *Handbook of Markov Chain Monte-Carlo (Steve Brooks, Andrew Gelman, Gailin Jones and Xiao-Li Meng, Eds.*

NGUYEN, C. H., RITSCHEL, T., MYSZKOWSKI, K., EISEMANN, E., AND SEIDEL, H.-P. 2012. 3d material style transfer. *Computer Graphics Forum*.

O'DONOVAN, P., AGARWALA, A., AND HERTZMANN, A. 2011. Color Compatibility From Large Datasets. *ACM Transactions on Graphics 30*, 4.

PALMER, S. E., AND SCHLOSS, K. B. 2010. An ecological valence theory of human color preference. *Proceedings of the National Academy of Sciences 107*, 19, 8877–8882.

ROBERT, C. P. 1995. Simulation of truncated normal variables. *Statistics and Computing 5*.

SAUVAGET, C., MANUEL, S., VITTAUT, J.-N., SUAREZ, J., AND BOYER, V. 2010. Segmented images colorization using harmony. In *Proc. of Signal-Image Technology and Internet-Based Systems (SITIS) 2010*, vol. 1.

SHARMA, G., WU, W., AND DALAL, E. N. 2005. The CIEDE2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations. *Color research and application 30*, 1.

SUTTON, T., AND WHELAN, B. 2004. *The Complete Color Harmony: Expert Color Information for Professional Color Results*. Color Harmony Series. Quayside Publishing Group.

WANG, B., YU, Y., WONG, T.-T., CHEN, C., AND XU, Y.-Q. 2010. Data-driven image color theme enhancement. In *Proc. SIGGRAPH Asia 2010*.

WELSH, T., ASHIKHMIN, M., AND MUELLER, K. 2002. Transferring color to greyscale images. In *Proc. SIGGRAPH 2002*.

YEH, Y.-T., BREEDEN, K., YANG, L., FISHER, M., AND HANRAHAN, P. 2012. Synthesis of tiled patterns using factor graphs. *ACM Transactions on Graphics*.

YEH, Y.-T., YANG, L., WATSON, M., GOODMAN, N. D., AND HANRAHAN, P. 2012. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. In *Proc. SIGGRAPH 2012*.

YU, L.-F., YEUNG, S.-K., TERZOPOULOS, D., AND CHAN, T. F. 2012. Dressup!: outfit synthesis through automatic optimization. In *Proc. SIGGRAPH Asia 2012*.