Context-Aware Asset Search for Graphic Design

Balazs Kovacs, Peter O'Donovan, Kavita Bala *Member*, Aaron Hertzmann, *Senior Member, IEEE*

Abstract—Graphic design tools provide powerful controls for expert-level design creation, but the options can often be overwhelming for novices. This paper proposes *Context-Aware Asset Search* tools that take the current state of the user's design into account, thereby providing search and selections that are compatible with the current design and better fit the user's needs. In particular, we focus on image search and color selection, two tasks that are central to design. We learn a model for compatibility of images and colors within a design, using crowdsourced data. We then use the learned model to rank image search results or color suggestions during design. We found counterintuitive behavior using conventional training with pairwise comparisons for image search, where models with and without compatibility performed similarly. We describe a data collection procedure that alleviates this problem. We show that our method outperforms baseline approaches in quantitative evaluation, and we also evaluate a prototype interactive design tool.

Index Terms—Graphic design, machine learning, AB testing, image search, user interfaces, color

1 Introduction

Graphic design tasks are time-consuming and, often, overwhelming for novices. Recent research has improved the process by using learned aesthetic prediction models that classify and rank assets by aesthetic style or quality [1], [2], [3], [4], [5]. These predictors can then provide better search results, e.g., allowing a user to select fonts according to style or images according to quality. However, these methods overlook a crucial factor, namely, the context of the user's current project. As a result, suggestion and search interfaces will show users many possible images or colors that fit their query, but do not fit with their current design. To date, current interfaces do not help users mix different types of elements.

This paper considers the problem of pairing elements of different types within a design. We propose Context-Aware Asset Search, in which the user interface takes the designer's current project into account when providing search results. In particular, we focus on arguably the most important pairing in many designs: images and colors. Image search is often an important step in design, such as selecting "hero images" for webpages and brochures, or stock photos that illustrate concepts in slide presentations. Currently, image search interfaces are not influenced by the design, and, typically, search happens within a separate application. This means that the user must manually filter out images that do not match their current design-in-progress, in addition to whatever other criteria they have. Instead, we include image search within the design interface itself, and adjust the search results so that images with compatible colors will be ranked higher. Conversely, we propose to revise the color selection tool for the choice of a text-background color pair to be compatible with the images already selected. Figure 1 compares conventional design approaches with our context-aware approach that suggests images and colors that are compatible with the current user design in progress.

Context-aware asset search introduces several unexpected challenges. To train our algorithms, we collect crowdsourced data that evaluates which images pair well with which colors. Past research on aesthetic and style prediction has been trained for accuracy on labeled data. Previous labeling has taken the form of absolute quality ratings or pairwise comparisons. In our initial experiments, training this way led to poor results and, often, trivial models. We analyze the reasons for this failure. Based on these insights, we develop a new training procedure based on asking crowdworkers to compare sets of retrievals from different scoring functions. In addition, we develop a new evaluation procedure for comparing search algorithms. These approaches should be useful for learning in other pairing tasks in the future.

We perform quantitative evaluations, showing that retrievals using our context-aware method are superior to methods that ignore the design context. Furthermore, we implement our context-aware search in a prototype graphic design tool and collect hundreds of designs from crowdworkers for 10 different themes, both with context-aware retrievals enabled and with a conventional baseline. The results from our context-aware method are preferred 69% of the time, in addition to being preferred for each individual theme.

As we are first to demonstrate this approach, we make several assumptions to limit the scope of the problem. In particular, we focus only on the interaction of an image and a color theme, without considering multiple design elements or different types of assets such as fonts, clip art, etc. We also do not consider the role of layout and positioning in design quality. Preliminary experience suggests that image-color pairing is the most significant contextual element, which entailed significant challenges in itself, and we leave these other factors for future work.

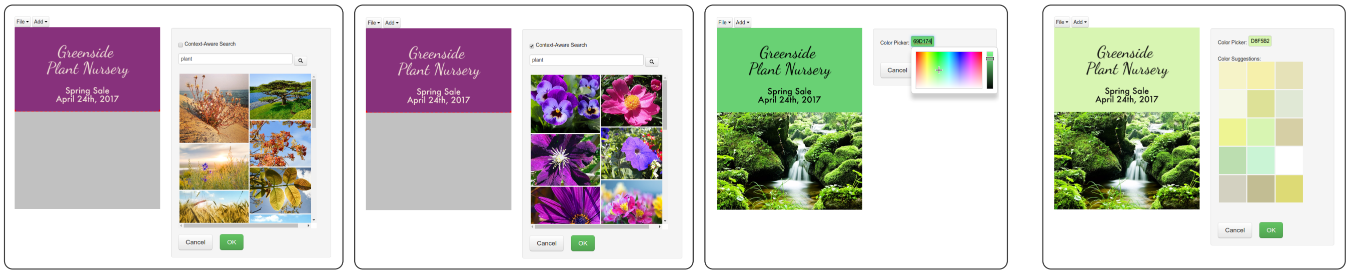
In summary, our contributions are: we introduce context-aware asset search for graphic design, a new interaction technique for design suggestions; a model for learning aesthetic compatibility between images and colors; analysis of problems that arise in training the model with conventional approaches; a new crowd-sourced training methodology for image and color retrieval; a new crowdsourced methodology for comparing retrieval algorithms; and, a prototype design tool that implements these interaction techniques.

2 Related Work

Search-in-context is a classic topic in HCI and information retrieval. Web search can be augmented based on the context

- *B. Kovacs and K. Bala were with Cornell University.*
- *P. O'Donovan and A. Hertzmann were with Adobe Systems, Inc.*

Manuscript received XXX; revised XXX.



(a) Conventional image search (b) Our context-aware search (c) Conventional color picker (d) Our context-aware suggestions

Fig. 1. *Image and Color Search Interfaces, in our prototype design tool.* **(a)** Conventional image search, incorporated into the interface. The results do not depend on the design; here, the green image search results do not match the purple design. **(b)** Our proposed method ranks images higher when they are compatible with the existing colors. **(c)** A conventional color picker provides all possible colors to the user, including many that are not compatible with the image. **(d)** Our color suggestions are designed to complement the current image choices.

of the user's current document [6], [7]. The Blueprint system augments programming user interfaces with contextual search tools for programming tasks [8]. CommunityCommands [9] recommends commands to a user, based on their current task. Chateau [10] suggests modifications when sketching 3D models, and Attribit [11] suggests 3D parts and webpage designs based on a learned model. We apply contextual search for graphic design, and, unlike in previous methods, our model learns how the context can improve search results.

For 2D design, Garces et al. [12] demonstrate a simple form of in-app search, based only on style similarity for a homogeneous class of assets, i.e., clip-art. Conventional color picking tools normally show every possible color, without making suggestions based on context. Current design applications, such as Canva and Adobe Photoshop, provide stock image search within the application, but the search is not affected by the current design choices. One can include color keywords in image search engines, but this does not explicitly make styles compatible, and requires conscious effort on the part of the user.

Our model builds on numerous previous projects in aesthetic quality and style prediction for graphic design and images. The earliest work in this area has focused on learning quality and style attributes for individual elements, including scene type [13], photographic style and quality [1], [2], [14], font attributes [3], and geometric objects [11], [15]. These methods can help users find individual elements that fit their goals, but are not designed to consider pairings of multiple elements. A related problem is to identify stylistically similar elements, including images [16], fonts [3], clip-art [12], and product images [17]. Style similarity is a useful criterion for search, but is not the same as compatibility, since contrasting elements often go well together.

A few previous methods specifically focus on compatibility. Methods for learning color compatibility [4], [18], [19] have used large databases of rated color schemes, an approach which requires a suitable dataset. Recent work has learned furniture compatibility [5], [20] from pairwise comparisons. We show that the class of approaches in these methods does not generalize to image search, likely because images are a much larger and more heterogeneous space than furniture. Several previous methods have been proposed for fashion pairing recommendation. Yu et al. [21] suggest pairing clothing for 3D models according to a small set of categories and color pairings. Several methods [22], [23], [24], [25] have been proposed for clothing recommendation from fashion image databases. These methods are tested by conventional retrieval metrics on hold-out data, which are not available for our problem. In contrast, our work proposes pairing algorithms for heterogeneous graphic design elements, namely colors and images. Obrador [26] suggests searching for images given colors or vice versa based on hand-coded heuristics, but does not learn the model from data or describe a user interface, or perform user evaluation. We describe new training and testing issues for image search, which is an extremely heterogeneous search space.

Many previous methods also address the relationships of colors to images. Most relevant are methods for extracting color themes from images [4], [27], [28]. While extracted color themes may pair well with images, there are cases where they may not, such as a lack of contrasting colors in a homogeneous theme. Cohen-Or et al. [29] harmonizes image colors by matching the color histogram of an image to hue templates. The hue templates were invented by Matsuda [30] based on observations from late 1970s fashion surveys. The templates were first evaluated quantitatively by O'Donovan et al. [4], who found that, while they were effective to reduce color spread in images, they did not accurately predict compatibility. Our method indirectly learns color compatibility from data and thus is not constrained to predetermined color templates. Moreover, our context-aware asset search takes the design context into account and provides suggestions based on the asset type (image, text, background) and its compatibility to other assets in the design.

Our retrieval training and evaluation is inspired by conventional techniques for learning to rank and retrieve search results using clickthrough data [31] and relevance metrics [32], [33], [34], [35]. However, obtaining clickthrough or relevance scores is not presently possible for our problem, and we present a new method for crowdsourcing search evaluation.

3 User Experience

This paper proposes Context-Aware Asset Search, which closely integrates image search and color selection with the user's current design. In our interface, image search is included within the design app, as shown in Figure 1 and demonstrated in the accompanying video. To perform an image search, a user may type a text query in the app, and search results are shown in the app. The user may easily preview each image in-place in the document. The image search results are ranked, in part, according to their compatibility with the current design, so that clashing results are omitted. As illustrated in the figure, this yields search results better-suited to the current design. The user may enable or disable this function

(a)

(b)

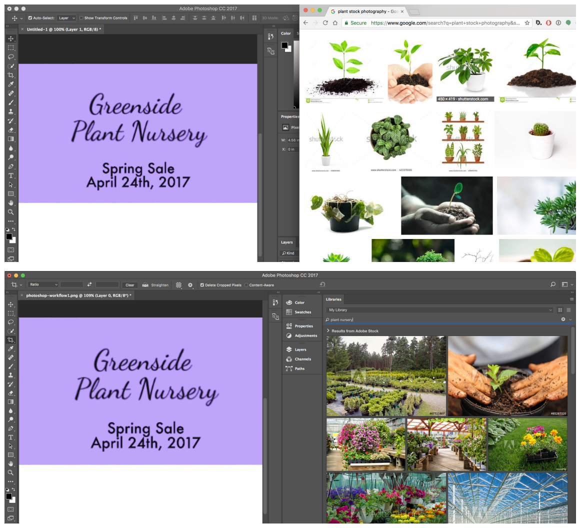


Fig. 2. Conventional workflows for asset search with graphic design. **(a)** Most commonly, search and design happen in separate windows, which requires the user to perform considerable mental context-switching, and makes it hard to preview how images will look in the design. Search results are not adapted to the target design. **(b)** Recently, several design apps have introduced stock search within the app, but the problem remains of being able to view very few assets at a time, and search results are not adapted to the task at hand.

with the "Context-Aware Search" checkbox. Likewise, for color selection, our system recommends color choices that go well with the currently-selected images.

In the conventional image search workflow, design and search happen in separate windows (Figure 2(left)). This requires the user to juggle separate windows and contexts. Previewing how an image would look in the design entails considerable overhead. Moreover, many image search results are clearly inappropriate because they do not fit aesthetically with the rest of the design, typically because their colors contrast poorly with the current design's colors. Since only a fraction of the screen is available for search, it is essential to make the first few search results as useful as possible. Some recent apps do include stock search within their interfaces (e.g., Figure 2(right)), but without otherwise integrating search with the current design task.

Likewise, conventional color pickers require the user to select from among all possible colors, whereas our approach suggests colors that fit well with the currently-selected images in the document.

In summary, context-aware search improves the design process by (a) providing search results in the same interface as the user's task, rather than requiring them to manage separate contexts, (b) adjusting search results to better match the aesthetics of the current document, and (c) providing a quick preview of how the asset would look in the current document. Our tool can be especially useful for novice and casual designers, who may be overwhelmed by too many options, many of which are inappropriate. The most basic way that our tool can be useful is to weed out incompatible combinations to facilitate the design workflow of these users.

The main technical contribution of our work is to learn to rank images and colors based on the current design. The next section describes a scoring model for ranking. The following section then discusses how this model is trained.

4 Energy Functions for Compatibility

In this section, we describe functions to rank images given a color scheme, and to rank color schemes given an image.

We define an energy function $E(\mathbf{I},\mathbf{C})$ that takes as input an image $\mathbf{I}$, and a color scheme $\mathbf{C}$ which includes a text color and a background color: $\mathbf{C} = (\mathbf{C}_T, \mathbf{C}_B)$. Then, in a user interface (Figure 1), which includes a color scheme $\mathbf{C}_0$, candidate images can be ranked according to the values of $E(\mathbf{I},\mathbf{C}_0)$. Conversely, if an image $\mathbf{I}_0$ has already been selected, candidate color schemes can be ranked by $E(\mathbf{I}_0,\mathbf{C})$.

We model the energy of a design by decomposing it into unary terms ("How good is this image/color scheme by itself?") and compatibility terms ("How compatible is this image with this text-background color combination?"). Specifically,

$$E(\mathbf{I},\mathbf{C}) = E_{img}(\mathbf{I}) + E_{color}(\mathbf{C}) + E_{compat}(\mathbf{I},\mathbf{C}) \tag{1}$$

where smaller energies indicate better designs. This decomposition allows us to separate distinct concerns in the score function without retraining. For example, in a layout with multiple images, the same energy could be applied by summing over the terms for each image.

Unary terms. Given an image $\mathbf{I}$ and a color scheme $\mathbf{C}$, the score function converts these to feature vectors $\mathbf{x}_I$ and $\mathbf{x}_C$. The image feature vector $\mathbf{x}_I$ includes three components: the dominant colors of the image, in multiple representations (inspired by [4]); a deep feature vector from a ResNet-152 model [36], meant to identify the objects and scene present in the image; the aesthetic score of the image, predicted by the model of Mai et al. [2]. The image feature vector is 198-dimensional in total. The color feature vector $\mathbf{x}_C$ is generated in the same manner as the features for the dominant image colors, yielding an 81-dimensional feature vector. Details of the feature vectors are given in Appendix A.

Given these feature vectors, the unary terms are linear:

$$E_{img}(\mathbf{I}) = \mathbf{w}_I^T \mathbf{x}_I \tag{2}$$
$$E_{color}(\mathbf{c}) = \mathbf{w}_C^T \mathbf{x}_C \tag{3}$$

where $\mathbf{w}_I$ and $\mathbf{w}_C$ are the weight vectors, to be learned as described in the next section.

Previous work on aesthetic compatibility has not used explicit unary terms, perhaps because it has focused on search spaces where filtering by quality is not necessary. We believe that quality is a significant factor in image search; conventional image search is primarily a combination of filtering by the query (e.g., does the image title contain the search keywords), along with unary terms preferring more-popular or better-looking images. Indeed, in our experiments, we find that the aesthetic score feature gets a significant weight. We show the learned weights in the Supplemental Material.

Compatibility terms. For the compatibility terms, we use reduced feature vectors, since the additional features were not useful in preliminary experiments. The reduced image feature vector $\mathbf{x}_I^r$ is a 9-dimensional vector containing the three dominant image colors, represented in *Lab* space. The reduced color feature vector $\mathbf{x}_C^r$ is a 6-dimensional vector of the text and background colors, also in *Lab* space.

Previous work has represented compatibility with a squared-distance in an embedding space, either using the same neural embedding for all categories [25], or using separate linear embeddings for each category [20]:

$$E^{SqEuc}(\mathbf{I},\mathbf{C}) = ||\mathbf{W}_I\mathbf{x}_I^r - \mathbf{W}_C\mathbf{x}_C^r||^2 \tag{4}$$

where $\mathbf{W}_I \in R^{D\times 9}, \mathbf{W}_C \in R^{D\times 6}$ are D-dimensional learned embedding matrices, where $D = 18$ was chosen empirically. However, this embedding prefers similar colors, even though distinct colors often pair well. For this reason, we also consider a scaled Euclidean distance:

$$E^{ScEuc}(\mathbf{I},\mathbf{C}) = \mathbf{w}_B^T(\mathbf{W}_I\mathbf{x}_I^r - \mathbf{W}_C\mathbf{x}_C^r)^2 \tag{5}$$

where the term in the parentheses $(\cdot)^2$ is squared elementwise, and $\mathbf{w}_B$ is an additional weighting vector. Because this vector can include negative entries, this compatibility term is capable of penalizing color schemes that are too similar.

Comparison probabilities. Given an energy function, we can describe the probability that a rater will rank one design over another, which will be used for training from comparisons. Specifically, suppose we are given design d_1 and design d_2, each of which comprises the same layouts, changing only the specific image and/or color choices. We model the probability that a viewer prefers the first design over the second with a logistic function [37]:

$$P(d_1 > d_2) = \frac{1}{1+e^{-E(d_2)+E(d_1)}} \tag{6}$$

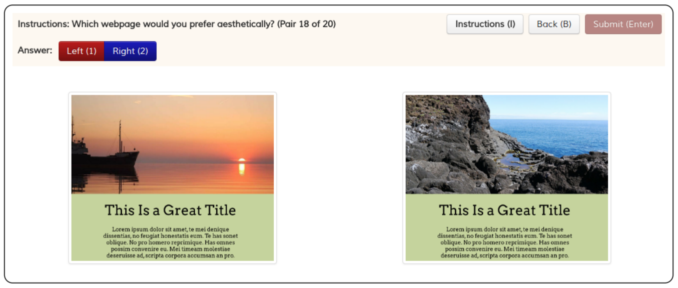
5 Data Collection and Model Evaluation

In this section, we describe procedures by which we train and evaluate different energy models. We begin by describing a standard pairwise data collection approach to collect training data. We then show how the standard data collection fails, and then describe new data collection and evaluation procedures to address these issues.

Energy models. We consider two forms of energy models. In a *unary model*, the energy includes only the unary terms that score the elements individually, similar to how current image search works. In a *compatibility model*, both unary and compatibility terms are included.

Design template. We consider only simple designs comprising a single "hero" image, a foreground color, and a background color (Figure 3). We experimented with several variants of this simple design, e.g., our initial experiment followed typical web layouts that place the text over the image, but this created many confounding factors, such as whether the text occludes salient image features. Therefore we did not allow the images to be occluded by text in our design templates.

Images. Creating a beautiful design requires high-quality images. We downloaded images from Pixabay (www.pixabay.com), a website hosting photos of amateur photographers. These images are both freely-available, and of high enough quality that several design apps use this dataset as their source, including Adobe Spark, BeFunky, and Word Swag. We downloaded 358,731 photos in 20 categories (e.g., "Nature/Landscapes", "Animals", etc.). Full details are provided in the Supplemental Material. We assigned each image into the train/validation/test set randomly with probability 0.64/0.16/0.2, respectively. The most consistently high-quality imagery is on professional stock photography websites, but these images are not freely-available for research. We also considered training from Flickr, but decided against this since


Instructions: Which webpage would you prefer aesthetically? (Pair 18 of 20)

Answer:

This Is a Great Title

Lorem ipsum dolor sit amet, te mei denique dissentias, no feugiat honestatis eum. Te has sonet oblique. No pro homero reprimique. Has omnes possim convenire eu. Mei timeam molestiae deseruisse ad, scripta corpora accumsan an pro.

This Is a Great Title

Lorem ipsum dolor sit amet, te mei denique dissentias, no feugiat honestatis eum. Te has sonet oblique. No pro homero reprimique. Has omnes possim convenire eu. Mei timeam molestiae deseruisse ad, scripta corpora accumsan an pro.

Fig. 3. *Pairwise data crowdworker task*. We ask workers to pick the more aesthetically compatible design.

Flickr photography is captured with different intent from stock photography, and thus is not normally used for design applications.

5.1 Pairwise comparisons

We first describe our approach here, which follows the pairwise comparison methodology used in previous work, e.g., [4]. We then describe problems with this methodology, and analyze why they occur. In the next section, we describe additional data collection procedures that remedy these issues.

For the initial data generation, we generate a collection of random test pairs, as in Figure 3. To make the comparison easier, each comparison keeps either the color or images fixed. Since uniformly sampling colors in HSV produces poor color choices, we randomly sample colors from the designer-created hue distribution provided by O'Donovan et al. [4]. Additionally, we always filter out candidate designs with low text-background contrast (e.g., white on light-gray) to save on training time. Finally, to ensure complete data separation between the train/validation/test split, we make sure that each design contains images from only one split and we only compare designs which come from the same split.

We use Amazon MTurk (AMT) for all our data collection and evaluation. Our assumption is that AMT workers are typically design novices, which is appropriate for our task, since assistive design interfaces are most useful for helping novices. We perform quality control [38] by blocking workers that fail over 25% of the sentinel questions. We choose these sentinels so that the comparisons are easy to make. These sentinels tested whether participants understood that they need to pick aesthetically compatible combinations over just "better looking" photos.

Finally, we ask 5 workers for each comparison and filter out questions with agreement below 75%. We keep approximately 40% of all comparisons we collected, which is a reasonable percentage for these subjective and often-ambiguous tasks. See the supplemental material for more details about our instructions to workers.

Learning. Given a collection of comparisons, we follow previous work to train a model by minimizing the negative log-likelihood of the data given the model, plus regularization:

$$L(\theta) = -\sum_i \ln P(d_1^i > d_2^i) + \lambda||\theta||_2^2 \tag{7}$$

where θ is a vector of all model weights ($\mathbf{w}$ and $\mathbf{W}$ terms). Although the comparisons may be subjective, this will learn a model maximally-consistent with typical behavior of the workers. We perform optimization by gradient descent with momentum, and implement our model using the Caffe framework [39]. The regularization weight λ is determined by picking the best performing model on the validation set.

Active learning. Since the space of possible designs is very big, we reduce the cost of data collection by uncertainty sampling [40], a widely used active learning technique. Specifically, in each active learning batch, we generate 100,000 designs, evaluate them with the current model and pick the top 5,000 most uncertain comparisons among them: 2,500 where the colors are fixed, and the other half where the images are fixed. A design comparison prediction is uncertain if the predicted probability for choosing the left vs. the right design is close to 50%. We ultimately collected 55,000 comparisons.

Accuracy metric. To evaluate a model on a test set, we can measure Accuracy. The Accuracy is defined as the percentage of holdout comparisons correctly predicted by the model, i.e., the number of hold-out comparisons that the model assigns higher probability to as compared to the option chosen by the crowdworkers.

5.2 Problems with Pairwise data

Following the procedure in the previous section, we trained two separate models: the full model in Equation 1, including both unary and compatibility terms, and a simpler version with unary terms only. In each case, active learning was used separately for each model, e.g., the unary term's dataset was augmented with uncertainty sampling according to the unary term's model. A shared test set was constructed by combining hold-out data from both models.

When initially conducting this research, we expected that the compatibility term would play a substantial role, because it would filter out image and color pairs that clash. Unexpectedly, the compatibility term had a negligible effect on the results: the compatibility model was accurate $62 \pm 1.5\%$ of the time, versus $60 \pm 1.5\%$ for the unary model, a difference which is both small and also not statistically significant (the 95% confidence intervals were computed by bootstrap sampling). As we experimented with other algorithm variants, or even minor variations in the data collection, the model sometimes learned zero weights for the compatibility terms, thus omitting them entirely.

This presents a conundrum: why were the compatibility terms largely inconsequential? This outcome suggests that color compatibility is irrelevant to how a viewer perceives a design. But we did not believe this, and there are many other possible explanations for this outcome. For example, it could be that our representation was too simple; we attempted replacing linear elements of the model with simple neural networks, but did not find a representation that helped.

One clue comes from looking at how we qualitatively evaluate results. The Accuracy metric in the previous section averages over, roughly speaking, a uniformly random sample of all possible comparisons. In contrast, our goal is to learn to retrieve good images and colors in a user-interface, as in Figure 1. This implies that Accuracy over all possible comparisons is not the correct metric, because we only care about the quality of the top retrievals.

It was surprising to us that this difference is important. Many previous methods for learning style and aesthetics have optimized pairwise Accuracy and demonstrated good retrievals, e.g., [3], [5], [11], [12], [20]. Pairwise preferences have also been used for optimizing search [41]. What is different about our problem?

We believe the explanation is that, on a *random* design comparison, the compatibility term *is* largely irrelevant. There is enormous variability in image appeal, much more so than for the compatibility problems considered in previous work [5], [12], [20]. In most cases, a crowdworker comparing two designs is driven by the appeal of the images and the colors independently. People would usually rather pick a beautiful image than a compatible one, if forced to choose, and, unlike the datasets in previous aesthetics work, images vary dramatically in how appealing they are. Furthermore, some colors are compatible with most images (e.g., a white background goes with most images). This would explain why adding a pairwise term only gives a small boost to accuracy.

But we are not interested in average performance over randomly-sampled comparisons. We want to aid designers who are trying to pick the very best image for their design. This means we need to develop new ways to gather data for training and evaluation.

Retrieval metrics. These observations are similar to the well-known difference between Accuracy and retrieval metrics, such as Precision and Recall. It is well-known that an accurate model can give bad retrievals [35]. For example, a classifier trained to recognize whether an image contains an umbrella can get 99% accuracy by always returning "No", if 99% of images do not include umbrellas. However, this classifier would not make for a good image search engine. Retrieval metrics are defined in terms of absolute relevance, i.e., which search results or image classifications are *correct*. Some work has optimized directly for precision [32], [35], assuming ground-truth relevance values are provided. The crucial difference to our case is that there is no meaningful notion of "relevance" for style-based search. We can only say whether one retrieval is better than another, averaged over some pool of users or crowdworkers.

5.3 Rank data

We wish to define a data collection procedure that addresses this problem. Here we propose the use of *rank data*, which simulates a user performing a search and selecting the best results. Specifically, given the top search results of a retrieval algorithm, which of these results are the best? All other possible comparisons are ignored by this metric; this metric ignores results on, for example, comparisons involving bad images that would never be retrieved. This approach is similar to training methods sometimes used for search engines [31], but focused on relative aesthetic quality rather than relevance. Furthermore, because it is not possible to deploy a real design tool to a large number of users as part of an academic research project, a crowdworker setup is required.

Generating this data requires beginning with an initial model θ; in our experiments, we begin with a model trained by the active learning procedure in the previous sections. Prior to generating data, we also cluster all potential test images into 200 clusters using k-means applied to their features (Appendix A). Likewise, the color schemes are clustered into 200 clusters. The clustering will be used to ensure diversity in queries and results.

To generate a color scheme for an image query, the algorithm selects a color cluster uniformly at random, and then randomly selects a color scheme from that cluster. The top 8 image search results are then generated by the current model, that is, by listing the images with the lowest energy values for this color scheme. A

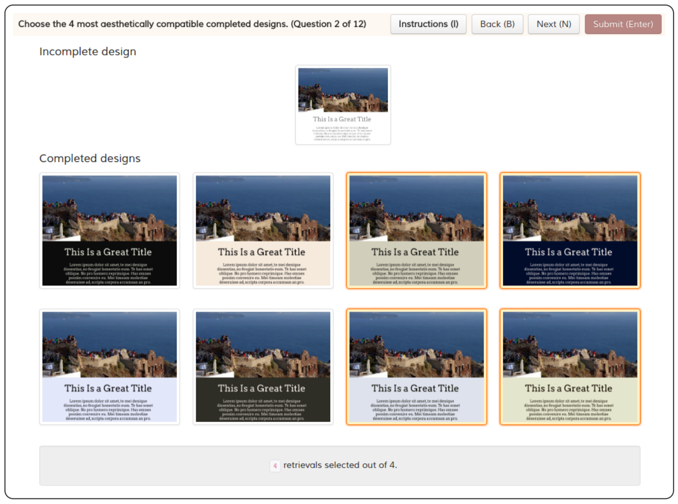

Choose the 4 most aesthetically compatible completed designs. (Question 2 of 12)

Fig. 4. *Rank data crowdworker task,* for color search. We ask workers to pick the 4 most aesthetically compatible completed designs out of 8 retrievals generated based on a query incomplete design. In this example, we queried for a text-background color combination. The retrievals chosen by the worker are highlighted with orange. A similar task is used for image search.

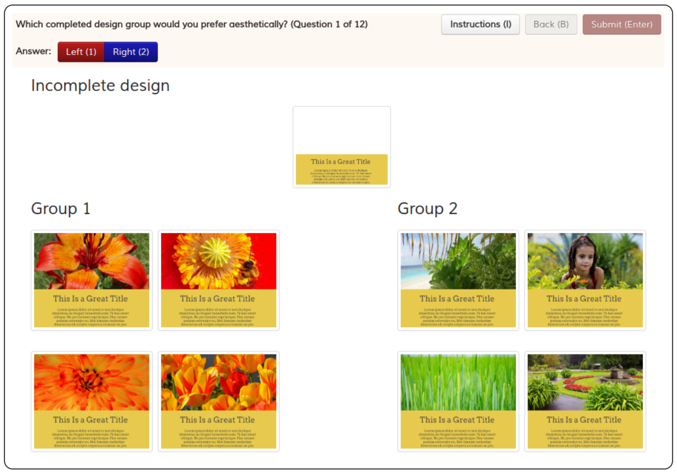

Which completed design group would you prefer aesthetically? (Question 1 of 12)

Fig. 5. *AB evaluation crowdworker task.* Workers are shown the top 4 retrievals of two algorithms for a query. They have to choose the group of designs which they prefer aesthetically.

crowdworker is then asked to select the best 4 search results. An example of the search interface is shown in Figure 4. User votes are aggregated; every image selected by at least 3 crowdworkers is designated as *selected*, and the others are non-selected. These selections are converted into a set of pairwise comparisons, where each pair comprises one selected and one non-selected image. For example, if there are 4 selected, and 4 non-selected images, this is converted into 16 pairs of selected/non-selected images. Each pair is used as a pairwise comparison data point, and included in the optimization objective (Equation 7).

Conversely, the algorithm also generates queries in which a query image is selected uniformly at random from the image clusters. To yield diverse color schemes in the results, the top 8 color schemes are constrained to come from the medoids of the 200 clusters. The process for generating pairwise comparisons is otherwise the same. Finally, we collected rank data for 250 color and 250 image queries.

We also experimented with running multiple rounds of this process, i.e., generating Rank Data, retraining, and repeating, but this did not seem to improve the results.

5.4 AB evaluation

We also need a better quantitative method to compare two different retrieval algorithms. Previous work has used testing on hold-out data from crowdworkers. However, as discussed above, quantitative measurement on the active learning data does not simulate the retrieval task. Moreover, the rank data is based on the results of a specific model θ. It gives a way to improve that model, but would not be useful for evaluating models that retrieve very different results. Furthermore, we would like to evaluate entire lists of results, following the recommendations of McNee et al. [42].

In search engines, these comparisons are typically done by deploying A/B tests, i.e., showing different search results to different users [43], and computing metrics (such as clickthrough or purchases) for each conditions. Since deploying a real system is impractical, we again simulate this with crowdworker evaluation.

First, we generate a set of test queries. Then, given two methods to compare, we generate the top 4 search results from the two methods, and then ask crowdworkers to determine which list of results is better (Figure 5). The AB comparison between the two methods is then the percentage of queries for which workers preferred one method over the other. We also considered interleaving search results from the two methods; we did not try this, in the belief that the AB comparison would yield clean data in a smaller number of measurements, as well as a more holistic evaluation of results.

As compared to Accuracy metrics, these metrics have the disadvantage that new data must be collected for every comparative evaluation; we cannot simply test on the same hold-out data. However, just as in A/B testing with real deployments [43], this seems to be necessary for evaluating subjective retrieval quality.

6 Results

In this section, we present detailed evaluation of our method using the AB evaluation introduced in the previous section, and comparisons of designs generated with our interactive design tool prototype.

6.1 AB evaluation results

AB evaluation allows us to directly compare the retrievals of two models using judgments made by crowdworkers. We generate 473 test queries which are held fixed for all algorithm comparisons and all of them are used in each AB evaluation. We ensure that each query and retrieved image comes from the test set. We performed 24 AB evaluations in total; results are shown in Figures 6 and 7. The confidence intervals shown in the figures are generated with the z-test at 95% confidence level (each algorithm pair has at least 1,000 AB comparison votes). We chose these tests to evaluate each element of our approach, in both modeling and training. We now discuss evaluation of these elements in detail.

In these experiments, we perform these comparisons on models trained with pairwise data only (PW), or pairwise plus rank data ($PW + R$). We found in preliminary experiments that training only on rank data (R) gave poor results, and did not include it in our evaluations. We believe this is the case because R models do not learn to filter out bad images and colors, since none of these pairings are included in the data.

Image search. We consider the value of including context in image search, which we test by comparing models trained with compatibility terms, to models trained only with unary terms. In AB comparisons, the compatibility model is preferred over the unary model 54.1±2.9% of the time for image search, when trained with $PW + R$ data. The difference is even larger for PW data: the compatibility model is preferred 65.6±2.7% of the time. This indicates that PW training does, in fact, give reasonable results, but this is only visible when evaluated on an appropriate metric. Still, we did sometimes get degenerate models when training on PW in early experiments.

For each type of model, including $PW + R$ data improves the quality of the model: the compatibility model trained with $PW + R$ data is preferred 58.7±2.9% of the time over PW training. A similar outcome is observed for the unary model. (One subtlety here is that the dataset is slightly larger with rank data than without; we also gathered additional pairwise data to compensate, retrained the PW model, and the result does not change.)

These results clearly validate the benefit of both compatibility terms and rank data for image search.

Color search. For color search, the compatibility model performs better than the unary model 55±2.9% of the time, when trained with PW data. However, we found that, in comparisons between models trained with and without rank data, the difference was either not statistically significant, or the rank data made the results worse. Overall, the compatibility model trained without rank data was our best-performing model. We find that the model without rank data presents more conservative color suggestions.

We interpret this result by noting that training only on pairwise comparisons was the method used in many previously-successful works for design. The principal benefits of rank data are for image search, which is a much harder problem.

Non-negative compatibility. We also compare models trained with our compatibility term (Equation 5) to those trained with conventional non-negative compatibility (Equation 4). The non-negative model is a special case of our model, so we would expect ours to perform better, given sufficient quantities of data. We found this to be the case, but the difference was not statistically significant when we compared the best model for each case. For color search, our PW model is preferred 50.7±2.7% over the non-negative ($PW + R$) model, and, for image search, the best compatibility ($PW + R$) model ties with the best non-negative (PW) model. Given the slight improvement of our model, we use it for all other experiments in this paper.

Discussion. Our compatibility model was preferred over the unary model by a statistically-significant difference, but one might wonder why the gap is not larger. The subjectivity of the tasks, particularly for randomly-generated tests, makes it hard to achieve larger gaps. Indeed, similar scores can be found in user evaluations from previous work on learning for graphic design [3], [4], [44]. In fact, small percentage improvements are usually considered very significant in online search, our main application area, where a few percentage points can increase revenue by thousands or millions of dollars (e.g., Kohavi et al. [43]).

Having performed these quantitative tests, we can then proceed to more involved evaluations that better test the method by

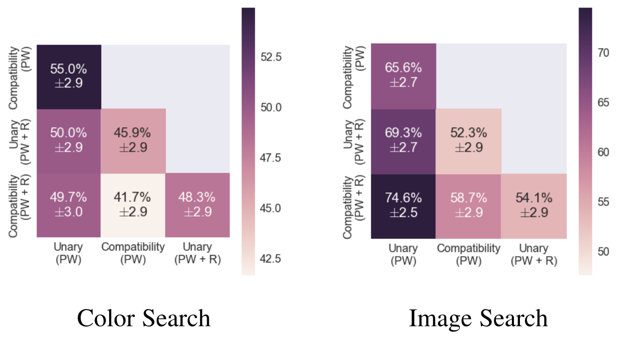


Fig. 6. *AB evaluation results: unary vs. compatibility, PW vs. PW + R.* The figure summarizes all 12 AB evaluation results testing every possible pair of unary and compatibility models, and PW and $PW + R$ training, for both image and color search. Each entry is the percentage of tests won by the method named on the vertical axis over the method named on the horizontal axis. For example, for Color Search, "Compatibility (PW)" won 55.0% percent of tests over "Unary (PW)", with confidence interval of 2.9%. See text for explanations of the different models and interpretations of the results. The confidence intervals are generated with the *z-test* at 95% confidence level (each algorithm pair has at least 1,000 votes).

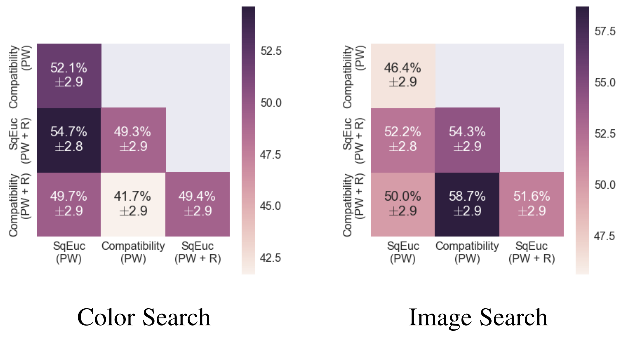


Fig. 7. *Non-negative (SqEuc) vs. our compatibility (ScEuc) model AB evaluation results.* The compatibility model outperforms the SqEuc model which uses squared Euclidean distance to represent compatibility in color search and ties in image search, but the difference is not statistically significant. See Figure 6 for explanation on how to read the figure.

embedding it into an interactive design study.

6.2 Interactive design study

We created a prototype design tool, as shown in Figure 1 and the accompanying video. We asked crowdworkers to use this tool to create new designs. Since our focus is on testing color and image suggestions, we did not allow workers to change the design layout or the text attributes. All image search results were provided from the test set.

Interfaces. We test two versions of the interface: *conventional* and *context-aware*. The conventional interface includes a standard color picker, and textual image search. The image search filters results by keyword, and sorts them according to a unary model only ("Unary ($PW + R$)"), mimicking standard image search. The context-aware interface uses the compatibility ($PW + R$) model to sort image search results, and also includes color suggestions based on the compatibility (PW) model. Users are not allowed to select whether the search is contextual. In addition, we show the color picker in this variant to allow users to fine-tune the suggested colors. Color suggestions are made based on the current image and the other color already in the design. To ensure that our color suggestions are diverse, we cluster colors into 200 clusters (as in rank data collection, but for individual colors) and evaluate the energy function on the medoids of these clusters to generate recommendations. As in training, low-contrast color suggestions are filtered from the search results.

Task. We created 10 design templates, each with a different theme (e.g., "Travel agency", "Bakery", "Bike shop"). The text in each design template is specific to its theme. All design templates contain the same generic default image (not related to any of the themes), and have neutral colors (black, gray and white). See Figure 9(a) for examples of the design templates.

We asked workers to change the image and colors of these templates to repurpose designs for a specific theme, and to create designs which are aesthetically pleasing. They were not given any information on what was being tested or how the recommendation engines worked.

Data collection. For each design template, we collected 100 design submissions: 50 for the context-aware and 50 for the conventional interface. We manually removed designs in which the worker did not actively explore background or foreground colors different from the ones originally provided in the design template, or in which the image was not related to the theme. After filtering, we had 881 curated designs. See Figure 9 for examples of submitted designs for both interfaces.

Design evaluation. We generated 200 randomly-sampled comparisons between designs for each of the 10 themes. Each comparison includes two designs for the same theme that were generated with the two different interfaces. To reduce the noise in submissions, we collected 5 votes for each comparison, and aggregated the votes with majority voting.

Over all themes, the context-aware method was chosen in 69% of the comparisons. Furthermore, we found that the context-aware method was preferred for all themes individually and the difference is statistically significant for all themes except the "Bakery" theme. We suspect that the difference is smaller for the "Bakery" and "Bike shop" themes, because there is a smaller set of related images in our dataset compared to other themes, thus there is a smaller variety of images to reorder with our context-aware image search. We saw the biggest difference in votes for "Coffee shop" and "Dog show", where our method was picked more than 75% of the time. See Figure 8 for detailed results.

We recorded activity logs of workers' actions during the design sessions. We found that, in an average session, a worker interacted with the color interfaces 30% fewer times with our interface than with the conventional interface. For more details, see the Supplemental Material.

7 Conclusion

Image search and selection of other assets play a significant role in modern design practice. To unify search and design interfaces, this paper introduces Context-Aware Asset Search, and shows, in simple but important cases, that it can improve the design process. We believe that this will be most useful for novice designers, to weed out the set of searches that they have to sort through, though it may streamline the process for professional designers as well.

We have focused on the most important pairing in design—images and colors—to demonstrate the value of this approach. There are many elements of design not currently handled, to be

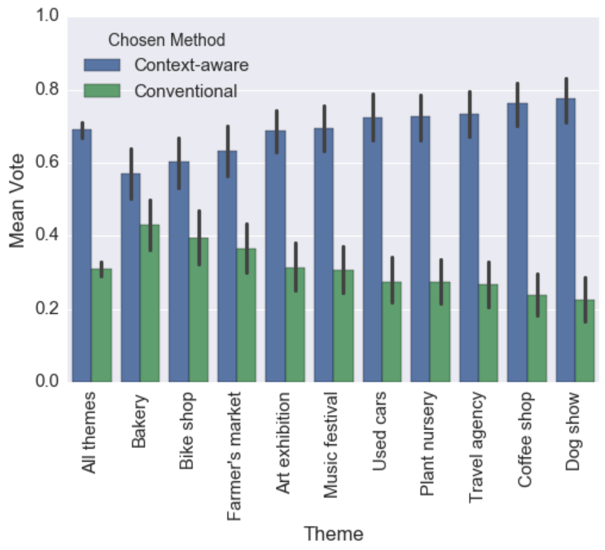


Fig. 8. *Interactive design study results.* We show the mean votes of each method over 200 randomly sampled design comparisons for each theme (2,000 design comparisons for all themes). A mean vote value of 1 would mean that a method won all comparisons for a certain theme. The context-aware interface outperforms the conventional in all design tasks. The difference is statistically significant with 95% confidence in all but one case.

addressed in future work. We did some preliminary experiments on learning for designs with multiple images, also considering different relative placements of images, but the effects of the inter-image terms were dwarfed by the image-color terms, which seem to be far more important. We did not consider compatibility between other types of elements, such as between fonts and images. We suspect that these are much more subtle combinations, for which it is harder to elicit meaningful effects. For more general designs, it may be necessary to model the higher-order interactions between multiple elements in a design. This could be done by decomposing the model into multiple pairwise interactions, and/or using image-based models as in [45]. Ultimately, we believe that learning general models for graphic design quality and style could lead to very powerful user interfaces.

Appendix

We built overcomplete feature vectors, on the assumption that learning with large enough datasets will allow the method to handle any redundant or unnecessary features.

Image features. We compute the three dominant colors of the image, extracted with k-means clustering of the image pixels in RGB color space. These three colors are converted into features in a manner inspired by [4]. Specifically, we represent a color in the Lab color space. We also square each entry of this color space. Additionally, we include the mean, standard deviation, minimum, maximum and maximum minus minimum values over the dominant color for each color dimension in Lab, HSV and RGB color spaces. Finally, we include the same statistics for the joint probabilities of the hues of each color pair (three pairs for three dominant colors), and the hue entropy, based on the joint hue distribution estimated from aggregated color scheme judgments in [4].

We also use deep features of the image obtained by running ResNet-152 [36] trained on ImageNet [46] on the image and taking the features of the layer before the last layer projected down to

128 dimensions with PCA. This provides a compact semantic representation of the image. Finally, we use the photo aesthetic score predicted by the model of [2] as an additional feature. This yields a 198 dimensional feature vector $\mathbf{x}_I$.

Color scheme features. For the text-background color scheme, we generate the same color features as for the dominant colors of the images, but additionally we include the colors in HSV, RGB color spaces and get a 81 dimensional feature vector $\mathbf{x}_C$.

Whitening. Since the feature dimensions have a wide range of scales, we apply a simple whitening transformation on the data by subtracting the mean and dividing by the standard deviation separately for each feature dimension. We estimate the means and standard deviations based on designs in the training set.

Acknowledgments

Thanks to Andy Edmonds for useful feedback. This work was supported by the National Science Foundation (grant IIS-1617861), and Adobe.

References

[1] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller, "Recognizing image style," in *Proc. British Machine Vision Conference*, 2014.

[2] L. Mai, H. Jin, and F. Liu, "Composition-preserving deep photo aesthetics assessment," in *Proc. Computer Vision and Pattern Recognition*, 2016, pp. 497–506.

[3] P. O'Donovan, J. Lībeks, A. Agarwala, and A. Hertzmann, "Exploratory font selection using crowdsourced attributes," *ACM Trans. on Graphics*, 2014.

[4] P. O'Donovan, A. Agarwala, and A. Hertzmann, "Color Compatibility From Large Datasets," *ACM Trans. on Graphics*, vol. 30, no. 4, 2011.

[5] Z. Lun, E. Kalogerakis, and A. Sheffer, "Elements of style: learning perceptual shape style similarity," *ACM Trans. on Graphics*, 2015.

[6] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: the concept revisited," *ACM Trans. Info. Sys.*, vol. 20, no. 1, 2002.

[7] J. Pitkow, H. Schutze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel, "Personalized search," *CACM*, 2002.

[8] J. Brandt, M. Dontcheva, M. Weskamp, and S. R. Klemmer, "Example-centric programming: Integrating web search into the development environment," in *Proc. CHI*, 2010.

[9] W. Li, J. Matejka, T. Grossman, J. Konstan, and G. Fitzmaurice, "Design and evaluation of a command recommendation system for software applications," *ACM TOCHI*, vol. 18, Jun. 2011.

[10] T. Igarashi and J. F. Hughes, "A suggestive interface for 3d drawing," in *Proc. UIST*, 2001, pp. 173–181.

[11] S. Chaudhuri, E. Kalogerakis, S. Giguere, and T. Funkhouser, "Attribit: Content creation with semantic attributes," in *Proc. UIST*, 2013.

[12] E. Garces, A. Agarwala, D. Gutierrez, and A. Hertzmann, "A similarity measure for illustration style," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 33, no. 4, 2014.

[13] D. Parikh and K. Grauman, "Relative attributes," in *Proc. International Conference on Computer Vision*, 2011.

[14] N. Murray, L. Marchesotti, and F. Perronnin, "Ava: A large-scale database for aesthetic visual analysis," in *Proc. Computer Vision and Pattern Recognition*, 2012.

[15] M. E. Yumer, S. Chaudhuri, J. Hodgins, and L. B. Kara, "Semantic shape editing using deformation handles," *ACM Trans. Graph*, 2015.

[16] C. Fang, H. Jin, J. Yang, and Z. Lin, "Collaborative feature learning from social media," in *Proc. Computer Vision and Pattern Recognition*, 2015.

[17] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM Trans. Graph.*, 2015.

[18] S. Lin, D. Ritchie, M. Fisher, and P. Hanrahan, "Probabilistic color-by-numbers: Suggesting pattern colorizations using factor graphs," in *ACM Trans. on Graphics (SIGGRAPH)*, ser. SIGGRAPH '13, 2013.

[19] A. Jahanian, S. Keshvari, S. Vishwanathan, and J. P. Allebach, "Colors—messengers of concepts: Visual design mining for learning color semantics," *ACM TOCHI*, vol. 24, no. 1, Jan. 2017.

[20] T. Liu, A. Hertzmann, W. Li, and T. Funkhouser, "Style compatibility for 3D furniture models," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 34, no. 4, Aug. 2015.

[21] L.-F. Yu, S.-K. Yeung, D. Terzopoulos, and T. F. Chan, "Dressup! outfit synthesis through automatic optimization," *ACM Trans. on Graphics*, 2012.

[22] V. Jagadeesh, R. Piramuthu, A. Bhardwaj, W. Di, and N. Sundaresan, "Large scale visual recommendations from street fashion images," in *Proc. KDD*, 2014.

[23] S. Liu, J. Feng, Z. Song, T. Zhang, H. Lu, C. Xu, and S. Yan, "Hi, magic closet, tell me what to wear!" in *Proc. MM*, 2012.

[24] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. KDD*, 2015.

[25] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala, and S. Belongie, "Learning visual clothing style with heterogeneous dyadic co-occurrences," in *Proc. International Conference on Computer Vision*, Santiago, Chile, 2015.

[26] P. Obrador, "Automatic color scheme picker for document templates based on image analysis and dual problem," in *Proc. SPIE 6076, Digital Publishing*, 2006.

[27] H. Chang, O. Fried, Y. Liu, S. DiVerdi, and A. Finkelstein, "Palette-based photo recoloring," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 34, no. 4, Jul. 2015.

[28] S. Lin and P. Hanrahan, "Modeling how people extract color themes from images," in *Proc. CHI*, 2013.

[29] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu, "Color harmonization," *ACM Trans. on Graphics*, vol. 25, no. 3, pp. 624–630, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141933

[30] Y. Matsuda, *Color Design*. Asakura Shoten, 1995.

[31] T. Joachims, "Optimizing search engines using clickthrough data," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2002, pp. 133–142.

[32] P. Henderson and V. Ferrari, "End-to-end training of object class detectors for mean average precision," in *Proc. ACCV*, 2016.

[33] K. Järvelin and J. Kekäläinen, "Ir evaluation methods for retrieving highly relevant documents," in *Proc. SIGIR*, 2000.

[34] A. Turpin, F. Scholer, S. Mizzaro, and E. Maddalena, "The benefits of magnitude estimation relevance assessments for information retrieval evaluation," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. New York, NY, USA: ACM, 2015, pp. 565–574. [Online]. Available: http://doi.acm.org/10.1145/2766462.2767760

[35] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proc. SIGIR*. ACM, 2007, pp. 271–278.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[37] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952. [Online]. Available: http://www.jstor.org/stable/2334029

[38] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality control in crowdsourcing systems: Issues and directions," *IEEE Internet Computing*, vol. 17, no. 2, pp. 76–81, Mar. 2013. [Online]. Available: http://dx.doi.org/10.1109/MIC.2013.20

[39] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[40] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '94. New York, NY, USA: Springer-Verlag New York, Inc., 1994, pp. 3–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=188490.188495

[41] B. Cartere, P. N. Bennett, D. M. Chickering, and S. T. Dumais, "Here or there: Preference judgments for relevance," in *Proc. ECIR*, 2008.

[42] S. M. McNee, J. Riedel, and J. A. Konstan, "Being accurate is not enough: how accuracy metrics have hurt recommender systems," in *Proc. CHI Extended Abstracts*, 2006.

[43] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: Survey and practical guide," *Data Min Knowl Disc*, 2009.

[44] X. Pang, Y. Cao, R. W. H. Lau, and A. B. Chan, "Directing user attention via visual flow on web designs," *ACM TOCHI*, vol. 35, no. 6, 2016.

[45] Z. Bylinskii, N. W. Kim, P. O'Donovan, S. Alsheikh, S. Madan, H. Pfister, F. Durand, B. Russell, and A. Hertzmann, "Learning visual importance for graphic designs and data visualizations," in *Proc. User Interface Software and Technology*. ACM, 2017, pp. 57–69.

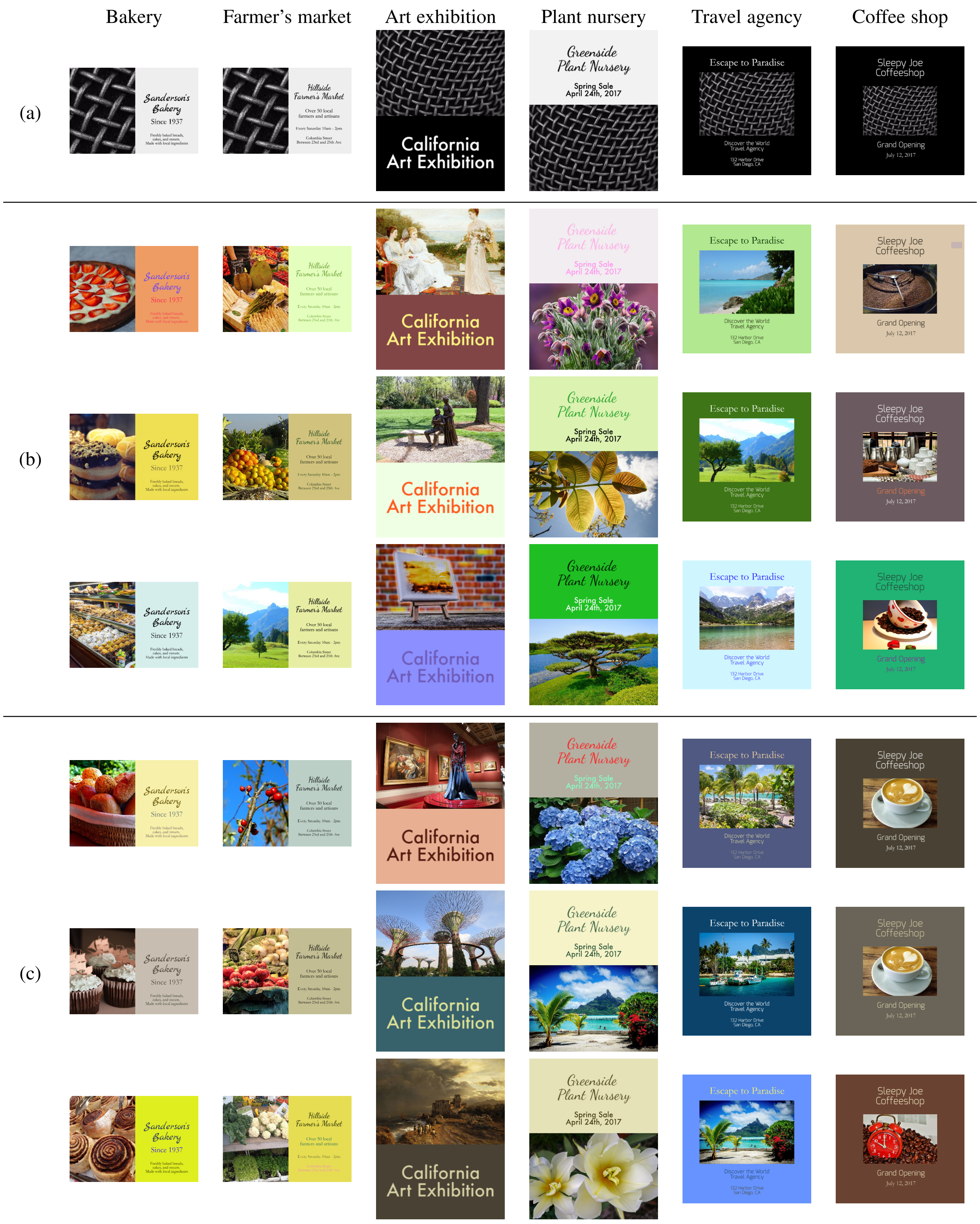


Fig. 9. *Examples of submitted designs.* For fairness, these results are *randomly-sampled* from our results, and thus some examples are much better than others. **(a)** Design templates that were given to workers as a starting point. They were asked to modify these for a specific theme by choosing a relevant image and compatible background and text colors. **(b)** Randomly sampled designs submitted with the conventional interface. **(c)** Randomly sampled designs submitted with the context-aware interface. On average the results with our method are better, even though some examples have poor color choices, since workers have the freedom to use the color picker and ignore our suggestions.

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

Balazs Kovacs received his BSc from Budapest University of Technology and Economics in 2012, and his MSc in Computer Science from Cornell University. He is currently a research engineer at Zoox.

Peter O'Donovan received his BSc in Computer Science from the University of Saskatchewan in 2005, and PhD from the University of Toronto in 2015. He currently works on graphic design software for novices at Adobe Systems.

Kavita Bala is a Professor in the Computer Science Department and Program of Computer Graphics at Cornell University. She received her S.M. and Ph.D. from the Massachusetts Institute of Technology (MIT), and her B.Tech. from the Indian Institute of Technology (IIT, Bombay). She co-founded GrokStyle, and serves as Chief Scientist (2015–), and is a faculty Fellow with the Atkinson Center for a Sustainable Future.

Bala is the Editor-in-Chief of *ACM Transactions on Graphics* (TOG). She has also served on the Papers Advisory Board for SIGGRAPH and SIGGRAPH Asia, and as Associate Editor for TOG, *IEEE Transactions on Visualization and Computer Graphics*, and *Computer Graphics Forum*. Bala has co-authored the graduate-level textbook "Advanced Global Illumination". She has chaired SIGGRAPH Asia 2011, and co-chaired Pacific Graphics (2010) and the Eurographics Symposium on Rendering (2005).

Aaron Hertzmann is a Principal Scientist at Adobe and an ACM Distinguished Scientist. He received a BA in Computer Science and Art/Art History from Rice University in 1996, and a PhD in Computer Science from New York University in 2001. He was a Professor at University of Toronto from 2003 to 2013, and has also worked at Pixar Animation Studios, University of Washington, Microsoft Research, Mitsubishi Electric Research Lab, Interval Research Corporation and NEC Research. He is an Associate Editor for *ACM Transactions on Graphics*, and has served as an Associate Editor for *IEEE Transactions on Visualization and Computer Graphics*. His awards include the MIT TR100 (2004), a Sloan Foundation Fellowship (2006), a Microsoft New Faculty Fellowship (2006), the CACS/AIC Outstanding Young CS Researcher Award (2010), and the Steacie Prize for Natural Sciences (2010), as well as several conference best paper awards.