Learning Layouts for Single-Page Graphic Designs

Peter O'Donovan, Aseem Agarwala, *Member, IEEE*, and Aaron Hertzmann, *Senior Member, IEEE*

Abstract—This paper presents an approach for automatically creating graphic design layouts using a new energy-based model derived from design principles. The model includes several new algorithms for analyzing graphic designs, including the prediction of perceived importance, alignment detection, and hierarchical segmentation. Given the model, we use optimization to synthesize new layouts for a variety of single-page graphic designs. Model parameters are learned with Nonlinear Inverse Optimization (NIO) from a small number of example layouts. To demonstrate our approach, we show results for applications including generating design layouts in various styles, retargeting designs to new sizes, and improving existing designs. We also compare our automatic results with designs created using crowdsourcing and show that our approach performs slightly better than novice designers.

Index Terms—Graphic design, layout, modeling, learning, crowdsourcing, nonlinear inverse optimization

1 Introduction

Graphic design is ubiquitous in modern life, including in packaging, advertisements, books, and websites, among many others. Designs are often difficult to create, as they must clearly convey information while also satisfying aesthetic goals. Designers must now also create designs for a wide variety of display sizes, from mobile phones to posters, and must often retarget designs from different sizes. Furthermore, many designs are created by inexperienced users with little training in graphic design. Automatic tools for creating, adapting, and improving graphic designs could greatly aid designers, and in particular, novice users.

This paper considers an important class of designs: single-page graphic designs such as advertisements, fliers, or posters (Fig. 1). These designs often consist of a small number of text and graphical elements. While there is previous work on automatically creating webpages and article-type designs, there is little research on generating single-page graphic designs. These designs are challenging as they are less structured and have a wider range of sizes and aspect ratios. Aesthetics are also important, as designs must often be eye-catching and visually pleasing.

Automating single-page graphic design is a complex and unsolved problem. In this paper we focus on the *layout problem*: specifying the locations and sizes of design elements. We assume a set of text and graphical elements are provided as inputs along with associated meta-data, such as the number of lines for text elements. Our goal is to output a visually pleasing arrangement of elements in a particular style. Modeling layout (i.e., element position and scale) is an important first step towards formalizing the difficult problem of design. More generally, design requires making many choices, including colours, fonts, and line breaks; we leave selecting these variables to future work.

We present a new energy-based model for evaluating layouts based on graphic design principles and stylistic goals. Due to the complexity of graphic designs, our approach has two stages. Given a design, we first perform a novel analysis stage that infers *hidden variables* corresponding to perceptual properties such as perceived importance, alignment, and grouping. These variables are then used as part of the model to evaluate a design. For example, the system analyzes perceived element alignment in a layout, and then an energy term penalizes misalignments. Given this model, we use optimization to synthesize a new layout. Our model has a large number of parameters, and we show how to learn their values from one or more examples using Nonlinear Inverse Optimization (NIO) [1].

We apply our model to three applications. First, the system can synthesize new design layouts in different styles learned from examples. Second, given an existing design, the system can retarget the design to a new size and/or aspect ratio. Lastly, we show a "design checker" that can improve an existing design to better match basic principles of graphic design. Model parameters are learned independently for each application using NIO and a few examples. Our system is designed for quality rather than efficiency, and currently is too slow for interactive applications, which we leave for future work (Section 10).

To evaluate our method, we automatically generate, retarget, and improve a number of designs. We compare our retargeting results with designs created by a professional designer, and also with novice users performing the same tasks on Mechanical Turk (MTurk). We show that our results are generally as good as those produced by the

- *P. O'Donovan is with the Department of Computer Science, University of Toronto, Bahen Centre, 40 St. George Street, Toronto, Ontario M5S 2E4, Canada. E-mail: odonovan@dgp.toronto.edu.*
- *A. Agarwala is with Adobe Research, Adobe Systems Inc., Seattle, WA. E-mail: asagarwa@adobe.com.*
- *A. Hertzmann is with the Department of Computer Science, University of Toronto, 10 King's College Rd, Rm 3302, Toronto, ON M5S3G4, Canada, and Adobe Research, Adobe Systems Inc., San Francisco, CA. E-mail: hertzman@dgp.toronto.edu.*

Manuscript received 6 Apr. 2013; revised 4 Sept. 2013; accepted 5 Nov. 2013. Date of publication 20 Mar. 2014; date of current version 27 June 2014. Recommended for acceptance by G. Drettakis.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2014.48

Fig. 1. We present a model of single-page graphic designs based on design principles such as alignment and balance. The model can synthesize layouts in various styles learned from examples, retarget layouts to different sizes, and improve designs based on design principles.

average human in our crowdsourced study, though not at the level of a professional designer. Graphic design is a particularly hard task to automate, as even humans who are untrained in design are not particularly good at it; in contrast, classic problems like speech recognition and computer vision are performed well by most humans. Nonetheless, automatically producing results on par with untrained humans represents significant progress on this unsolved problem.

2 Related Work

Remarkably, there is little work on automating single-page graphic design. A closely-related problem is the adaptive layout of primarily text-based documents such as articles [2]. In this case, templates and dynamic programming can be used to efficiently generate layouts [3], [4], [5]. However, single-page graphic designs are more free-form and do not easily conform to templates or a linear read-order. Harrington et al. [6] present an energy function to measure the aesthetics of a layout that also includes terms like alignment and balance; however, they do not show any designs and offer no evaluation. Balinsky et al. describe measures of alignment in documents [7] as well as an aesthetics-driven layout engine for non-flow documents [8]; however, only a single result is shown, and no evaluation is presented. Gonzalez-Morcillo [9] present a system for creating single-page graphic designs. However, this approach cannot learn different styles and uses a simple layout technique that normally aligns elements to margins. Jahanian et al. [10] analyze the visual saliency of photographs to guide placement of text in magazine covers.

Our general approach of defining and optimizing an energy function is common for other layout problems, such as generic text and figure blocks [11], photo albums [12], route maps [13], and furniture layout [14], [15]. These approaches generally involve simple hand-tuned energy functions with a few terms such as alignment or balance.

A few approaches in design synthesis learn model parameters. In interface design, Gajos and Weld [16] define a model to specify the position and types of widgets. Users select between different interfaces and a margin-based learning approach sets the linear weights of the objective function. Vollick et al. [17] model layouts of labels of parts in technical diagrams. An energy-based model evaluates label layouts, and Nonlinear Inverse Optimization learns parameters to create layouts in different styles.

Analyzing document structure is a well-studied problem, particularly for understanding scanned documents such as articles, book pages, or reports [18]. One basic problem is inferring the parts of the design and the overall structure. Another important task is analyzing the logical structure of design, for example, determining the title, abstract, and paragraphs in a document. A common approach for such analysis is grammar-based parsing, where parameters are either hand-tuned or learned using labelled training documents [19]. Talton et al. [20] presents an approach that learns grammar production rules to parse webpages.

A related analysis problem is design segmentation. Rosenholtz et al. [21] segment user interfaces and infographics using orientation or lightness. Designers often use grids, either explicitly or implicitly, to organize elements [22]. Baluja [23] uses a grid-based segmentation of webpages for mobile browsing, while Krishnamoorthy et al. [24] hierarchically segment journal pages into rectangular regions.

A related problem to our graphic design retargeting application is the retargeting of webpages to different display sizes. Kumar et al. [25] present a learning-based system for example-based webpage retargeting. Mappings between Document Object Model (DOM) elements of two webpages are learned from user data, allowing style transfer and retargeting. Baluja [23] retargets by segmenting a webpage into a 3×3 grid and magnifies these regions. However, these approaches are specific to webpage design; single-page graphic designs are more free-form and graphical, and often

do not have innate structure that maps easily to a DOM model. There has also been significant recent progress in image retargeting [26]. Liu et al. [27] use retargeting to optimize image composition using a simple objective function based on photographic principles. However, retargeting algorithms are inappropriate for design retargeting, which can modify element positions and scales.

3 Overview

This work defines a graphic design as a set of visual *elements*, including text and graphics, and represented as images with associated meta-data. We focus on the *layout problem*: determining the position and scale of these elements, denoted $\mathbf{X}$. Our overall goal is to create layouts which respect the principles of graphic design, such as alignment and symmetry in a variety of styles. Our main contribution is an energy-based model $E(\mathbf{X};\theta)$ which evaluates a layout $\mathbf{X}$ using parameters θ which define the desired style and intentions of the user (Section 4).

Due to the complexity of the problem, we use a multi-stage approach. To evaluate the energy of a layout, the system first estimates *hidden variables* ($\mathbf{h}$) that correspond to important visual properties. The hidden variables are the perceived importance of each element, a grid-based segmentation of the layout, and labels specifying alignment groups (Section 5). We then define our energy terms in terms of these hidden variables as $E_h(\mathbf{X},\mathbf{h};\theta)$. The energy function enforces many other design principles, including alignment, symmetry, and white space (Section 6).

Given this energy function and a suitable optimizer for $\mathbf{X}$ (Section 7), the system can generate design layouts in a variety of styles. Due to its high dimensionality, an important goal is learning θ without requiring time-consuming manual parameter tuning. We use Nonlinear Inverse Optimization (Section 8) to learn parameters θ, which are used to generate layouts for new designs.

We present applications of the model, including results and evaluation, in Section 9. First, we demonstrate layout synthesis, where layouts are generated for designs in a variety of styles with learned parameters. We then show design retargeting results, where a previous layout $\mathbf{X}_p$ is modified for a different output size. We also show results of design improvement, which takes an existing layout and optimizes it to enforce design principles.

4 Graphic Design Model

We measure the overall quality of a layout as a weighted sum of energy terms:

$$E_h(\mathbf{X},\mathbf{h};\theta) = \sum_i w_i E_i(\mathbf{X},\mathbf{h};\,\alpha_i,\mathbf{X}_p). \quad (1)$$

A design layout $\mathbf{X}$ is defined as the x and y positions, height, and alternate ID of each element. Alternate IDs select between different alternate elements, usually text blocks with different internal alignments. The energy terms E_i are defined in Section 6 and the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2014.48. $\mathbf{h}$ are the hidden variables described in Section 5, and θ are the model parameters. When the model is used for design retargeting or improvement, the user provides a previous layout $\mathbf{X}_p$.

The inputs to the model include the design elements, meta-data for each element, an output width and height, and optionally a previous layout $\mathbf{X}_p$. Given the inputs and the model, our system optimizes $\mathbf{X}$ to synthesize a design layout. By changing the parameters θ, we can generate layouts in various styles.

Elements are provided as images, along with user-defined meta-data. The system only requires three meta-data values for each element: the class (text or graphics), the number of lines (for text elements), and an importance value (low, medium, high, or very high). Optionally, if the element is part of a group, a group ID may be provided. The user may also provide binary masks to specify if a person or face is present in the graphic, or an important region which cannot be obscured by text. However, these masks are not required nor must they be precisely drawn.

Model parameters are divided into two groups, $\theta = [\mathbf{w},\boldsymbol{\alpha}]$. Each energy term E_i has a positive weight w_i, and most terms have a non-linearity parameter α_i. The weights are constrained to be positive. For example, a positive weight for the misalignment energy term means the model can only encourage alignment. We also include some reversed energy terms. For example, one term encourages symmetry, and a reversed term encourages asymmetry. The sum of these two non-linear terms, both with positive weights, allows the model control over the preferred amount of symmetry.

We use a sigmoid function in many of our energy terms: $\mathcal{S}(x;\,\alpha) = \arctan(x\alpha)/\arctan(\alpha)$. We often use it to reshape energy terms as $\mathcal{S}(E_i(\mathbf{X});\alpha)$, as large values of α make the energy more sensitive to small changes, such as in styles with little white space between elements. In the appendix, available in the online supplemental material, we include a full listing of the parameters as well as their initial values.

5 Design Analysis/Hidden Variables

Before evaluating a layout, the system performs an analysis stage to infer hidden variables corresponding to how a human viewer perceives the layout. We infer three key variables: the perceived importance of each element, labels specifying element alignment, and a grid-based segmentation. These perceptual properties cannot be provided initially as meta-data or a document structure; they are a function of the arrangement of elements on the page. This section explains our approaches for computing these hidden variables; Section 6 explains how the variables are used in the energy function.

5.1 Importance Map

When creating a design, controlling the perceived importance of various elements is crucial, and designers often arrange elements to convey their importance [28]. Colour, size, and location all contribute to an element's perceived importance, but formulating a mathematical formula is difficult. There are obviously relative differences in importance: a large graphic in the center of the design is far more important than a small URL in a corner. But how does

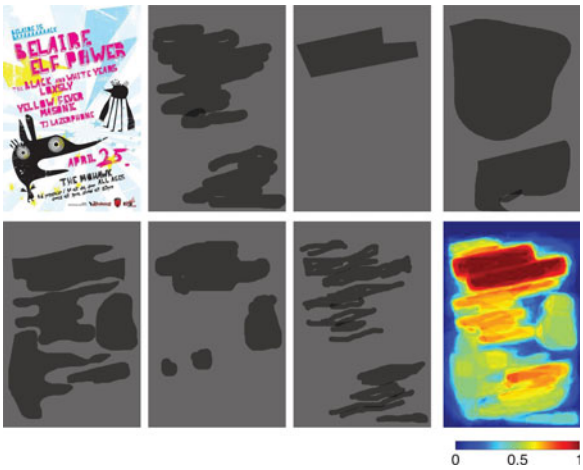

Fig. 2. *Importance maps.* Given a design (top left), MTurk users mark what they consider important. While the individual maps are noisy, when averaged over 20-30 users (bottom right), the mean maps are reasonable. Design courtesy of Flickr user Dániel Perlaky.

location affect importance generally? How does the importance depend on other elements?

Importance is related to image saliency [29], [30], [31], [32], which is usually equated to predicting eye-fixations. Eye fixations vary significantly over text blocks for example, even though the text's importance is uniform. Eye-fixations may also occur in unimportant regions as the user scans the design.

Crowdsourced design importance. Inspired by Judd [31], we model importance using a data-driven approach. First, we collected 1,075 graphic designs from Flickr. We then performed an MTurk study asking 35 users to label important regions in a design, and averaged the responses over all users. The system then computes per-pixel features, described below, and trains a linear regression model on the mean importance map. This model is then applied to new designs to predict the importance of elements.

The individual MTurk importance maps are often quite noisy, with significant variation between users. However, averaged over many users (20-30), the mean importance maps often give a plausible ranking of importance. See Fig. 2 for examples of individual user's importance maps, Fig. 3 for several mean importance maps, and the supplementary material, available online, for more examples. This result is surprising, because the mean importance should not necessarily create a relative ranking of elements; if everyone performed the task the same, there would be no relative difference.

We hypothesize that each element has an unknown importance rank, and that people choose different numbers of elements to label. Some users label only the most important element, others only a few, others most of the elements. If each person marks the top k elements, for an individual value of k, we will produce the correct ranking by averaging the "votes" of all users.

Features. To learn these importance maps, we calculate per-pixel features to train a linear regression model. Features include RGB colour channels, efficient low-level saliency models [29], [30], and multi-scale contrast features [33]. Many designers suggest placing elements on the

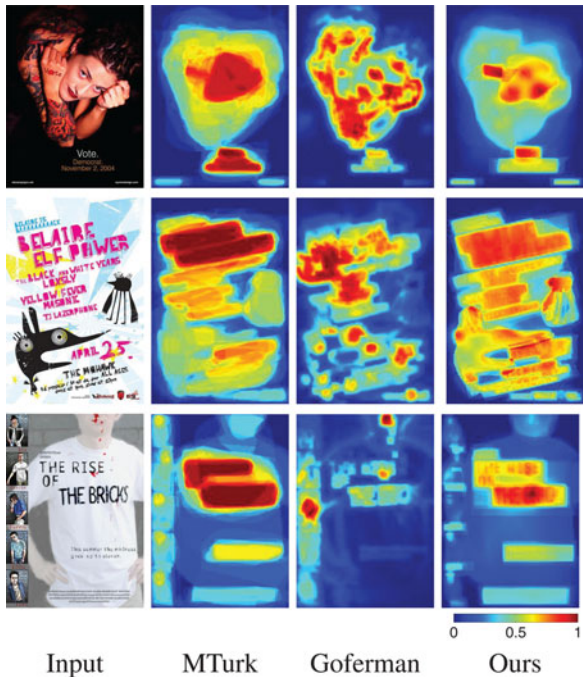

Fig. 3. *Design importance.* Given a design, we show the mean MTurk importance map, the saliency model of Goferman [32], and our importance model. Designs courtesy of William Berry, Dániel Perlaky, and Ben Keenan.

"Third lines" of the image; this is known as the Rule of Thirds. We include global position features including the distance to the Third lines, power points (intersections of the Third lines), image center, boundaries, and diagonals.

Our features include labeling of people, faces, and text. Because existing methods for face and person detection often fail for graphic designs; we use crowdsourcing to find these labels. For our training data, MTurk users drew binary masks for both people and faces and the average for each was taken over approximately 30 users. In a separate task, workers marked text blocks along with the number of lines in each block. See Appendix A, available in the online supplemental material, for examples. The system computes per-pixel features including the fraction of labels over all users, at least one user labeling, the user count, and the mean labeling over the entire image. The connected components of the label maps are computed and the segment size (both absolute and relative to the largest segment) and the number of segments are used as features. Text features include the size (both absolute and relative to the height), and number of lines.

Note that these crowdsourced features are used only for training the importance model. When we apply the model on synthesized layouts (Fig. 4), text location and size are known during synthesis, and person and face label images can be created by a single user. We do not require MTurk labels when synthesizing designs.

Prediction. The model computes features $y(\mathbf{p})$ for each pixel $\mathbf{p}$ and predicts a per-pixel importance value $r(\mathbf{p})$ using a linear regression of the features, learned with LASSO [34]. Fig. 3 shows a few example designs. 90 percent of the designs were used for training, and 10 percent for testing. These results show that our model predicts human

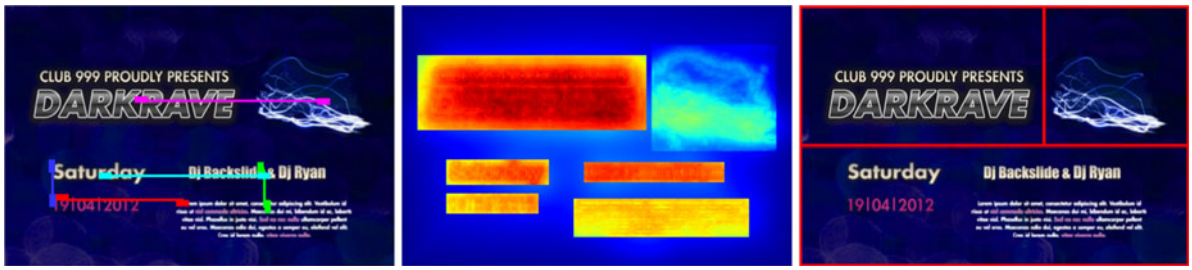

Fig. 4. *Analysis algorithms. Left.* Alignment detection. Rectangle colours indicate the detected alignment groups, and the deviation of the rectangles from the connector lines indicates misalignment. *Middle.* Importance prediction. Warmer colours indicate higher importance. *Right.* Segmentation of the design into text and graphic regions.

importance maps reasonably well, while saliency models often fail. This result is unsurprising since these methods predict eye-fixations, not importance. Saliency methods are also designed for natural images, and fail to capture the importance of text, for example. See Appendix A, available in the online supplemental material, for a quantitive comparison with various saliency models.

5.2 Alignment

The alignment of elements is crucial to how viewers perceive a design. Our system performs an analysis stage which labels all aligned elements as well as alignment groups. Elements with slight misalignments are also labeled as aligned, as the model will later penalize them. We define six possible *alignments types*: Left, X-Center, Right Top, Y-Center, and Bottom. Alignment indicator variables between elements i and j are denoted I^a_{ij}.

We use simple heuristics to compute alignments using element bounding boxes. First, the difference in the bounding box edge or center position must be below a threshold:

$$A^a_{ij} = \left(d^a_{ij} < \tau_{align}\right), \tag{2}$$

where a is the alignment types, d^a_{ij} indicates the distance between two elements i and j depending on the alignment type using element's bounding boxes. For example, if $a = L$ then d^L_{ij} measures the difference in the left edge of the bounding box for both elements. The threshold was set to $\tau_{align} = 0.065$. Note that slightly misaligned elements are still labelled as aligned, allowing slightly misalignments to be penalized by the energy model.

Second, elements may not align if another element lies between them:

$$B_{ij} = (b_{ij} < 1), \tag{3}$$

where b_{ij} is the number of elements between i and j.

Lastly, if a text block is internally aligned, it may only align with other elements with that alignment type. For example, a left-aligned text block may only left-align with another element. We denote the internal alignment indicator variable as N^a_i. Single-line text and graphical elements can align to any type.

The alignment labeling is the conjunction of these terms:

$$I^a_{ij} = A^a_{ij} \wedge B_{ij} \wedge N^a_i \wedge N^a_j. \tag{4}$$

Within each axis, two elements may normally align by only a single type. The indicator variable with the minimum alignment distance d^a_{ij} is set to 1, and the other two are set to 0. However, if all types align perfectly ($d^a_{ij} = 0$ for all types) then all three indicators are set to 1.

We next define an *alignment group* as a connected set of aligned elements. If elements i and k are aligned, and elements k and j are aligned by the same type, then i and j are set to aligned ($I^a_{ij} = I^a_{ik} \wedge I^a_{kj}$). We denote group membership using binary indicator variables I^i_g. Fig. 4 illustrates the alignment labeling graphically. See Appendix C.1, available in the online supplemental material, for more examples of alignment and grouping.

5.3 Hierarchical Segmentation

Designers often use grids or rectangular regions to organize elements. A viewer perceives this structure and relates alignment, grouping, and symmetry to these regions. Our system estimates this layout structure and calculates energy terms based on it. The algorithm takes as input the layout, binary masks for each element, and element classes (graphic or text); the output is a hierarchical segmentation of the design into non-overlapping rectangular regions.

The proposed algorithm segments a design by vertically or horizontally splitting regions which contain both text and graphics. Each split is evaluated using a cost function which measures the intersection of the split with elements, the separation between graphic and text elements, and distance to the region center. The algorithm recursively segments regions until a region contains only elements of the same class, or a user-specified maximum depth is reached. Lastly, empty or adjacent regions with the same element classes are merged. See Appendix B, available in the online supplemental material, for more examples and visualizations of the intersection costs.

Objective function. Given a rectangular region r, a cut c is defined as an x or y position in r which splits the region into two rectangular sub-regions r_1 and r_2. The system evaluates cuts based on three simple criteria. First, cuts should not be placed over elements, especially near an element's center. One energy term penalizes cuts based on the distance to each element's bounding box. Cuts nearer the region boundaries pay a low cost, while cuts near the center pay a high cost:

$$F_{int}(c) = \frac{1}{n}\sum_{p \in c} \max_i \left(I^p_i \delta^c_i(p)\right)^2, \tag{5}$$

where $p \in c$ are the pixels p along the cut c, I^p_i is an indicator variable indicating if element i overlaps with pixel p, and

$\delta_i^c(p)$ is the distance of pixel p to the bounding box of element i. This distance depends on the cut type c; vertical boundary distances are used for horizontal cuts, and vice-versa for vertical cuts.

Second, the algorithm prefers that regions contain only text or graphical elements. An energy term evaluates the sub-regions r_1 and r_2 and counts the number of elements of the same class (text or graphics) in both regions.

$$F_{elm}(c) = -(N(r_1) + N(r_2)), \tag{6}$$

where $N(r)$ defines the number of elements in region r if all elements have the same class, and 0 otherwise.

Third, the algorithm prefers evenly splitting regions, so the normalized distance of the cut to the region center r_c is

$$F_{cen}(c) = -\frac{|c - r_c|}{r_l}, \tag{7}$$

where r_c is the region center's location, and r_l is the region length. The system evaluates a cut using the following function:

$$F(c) = w_{int}F_{int}(c) + w_{elm}F_{elm}(c) + w_{cen}F_{cen}(c). \tag{8}$$

In all experiments, $w_{int} = 100$, $w_{elm} = 100$, $w_{cen} = 1$. Using the cumulative sum of the intersection distances allows the parse to be computed quite efficiently (5-10 ms for 100×150 pixels). Fig. 4 (right) shows an example.

6 Graphic Design Energy Terms

We next describe our energy-based model of graphic design. Because of the complexity of the model, we describe these terms at a high level, and provide mathematical details in Appendix C, available in the online supplemental material. Our model is designed to balance many goals, such as clearly conveying information, aesthetics, and stylistic variation. Our modeling choices are inspired by principles gleaned from the graphic design literature [28], [35], [36], [37], [38], and our own observations. We outline these principles, and the energy terms we created to capture them, as well as others we found necessary.

Building a model for graphic design is challenging for several reasons. We want an energy function that captures a range of styles. However, the model must also produce a consistent style on designs with different content. Furthermore, many principles of design, and their corresponding energy terms, are related and interact with one another. For example, white space is an important part of graphic design. But many aspects of design affect white space, such as element scale or symmetry. Finding a working set of energy terms is non-trivial; a variety of terms are required to capture a range of styles, but, with too many, the model may overfit while learning.

Lastly, modeling graphic design well is difficult because of our exposure to it and sensitivity to slight mistakes. Most people see many graphic designs daily, so we are attuned to successful designs. It is worth noting that, while previous work have proposed related terms, none have created a full model which can synthesize designs such as posters or advertisements in a variety of styles.

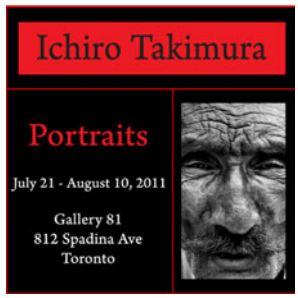

Fig. 5. *Symmetry types. Global symmetry* is defined over the entire design; *region symmetry* is defined with respect to the segmentation regions (shown with red lines). The top region has both symmetry types; the bottom two regions have only region symmetry.

Our methodology involved studying design literature and examples for stylistic variation or functionality. We then created relevant energy terms, optimized designs with these terms, then refined the terms and re-optimized. Parameters were initially hand-tuned, but, as the model increased in complexity, Nonlinear Inverse Optimization (Section 8) was used to learn model parameters. NIO was also used to remove redundant energy terms; if after learning on several styles, an energy term was not used, it was removed. For example, one term measured the fraction of the design covered by graphical elements, and was removed after it was found to be redundant. In Appendix C, available in the online supplemental material, we show the effect of removing various energy terms from the model.

Alignment. Correct alignment is an important aspect of design that has been modelled in other layout applications [3], [14], [17]. Elements are aligned on the page to indicate organizational structure, and for aesthetics.

We define energy terms measuring the fraction of element pairs that align with a particular alignment type (Left, X-Center, Right, Top, Y-Center, Bottom). Slight misalignment of elements is also visually displeasing; the model uses a robust cost function that heavily penalizes slight misalignments. The model encourages larger alignment groups with an energy term that measures the mean alignment group size. Larger alignment groups are preferred as they produce simpler designs, with more unity between elements.

Balance. One of the most common design principles is visual balance [28], [35], [36]. Symmetric balance is a common way of organizing elements in a stable and conventional layout. Many modern design styles use asymmetry, as these designs are often more dynamic, but require greater design skill. To measure design symmetry, the model calculates the fraction of element pixels which have a symmetric counterpart along the x and y axis. Separate energy terms are defined for x and y axis symmetry, as well for text and graphical elements. We define a similar asymmetry measure for asymmetric styles.

Designers often symmetrize elements not with the overall page, but with regions of the page (Fig. 5). Given the hierarchical segmentation of Section 5.3, we define a symmetry term using the symmetric counterpart of each pixel along the x-axis within the region. This region-based

symmetry is the motivation for our segmentation algorithm, and allows simple hierarchical design styles like Fig. 5.

Emphasis. Successful designs highlight important elements and lead the viewer's eye to key text or graphics. Many factors influence emphasis, including isolation, placement, scale, or contrast [36]. However, it is unclear how these factors relate, or should even be defined, to determine visual importance. Designers also often establish a desired hierarchy of importance for elements [28], [35], [36], with more important elements being visually emphasized.

The model matches the perceived importance of elements to their desired importance. The system estimates the perceived visual importance of each element (Section 5.1). Estimated values are compared to fixed scalar importance values for each element using Pearson correlation, with separate energy terms for graphical and text elements. The desired importance values are provided as meta-data during the design creation process and are not estimated by the system. In practice, these values are usually simple to specify. White [28] recommends designers establish an importance hierarchy of at most 3 levels, as more can become confusing. In our examples we usually specify three levels of importance: low, medium, and high, though occasionally we used four levels.

White space. White space in graphic designs is fundamental for readability and aesthetics. Element distance is also closely related to the principle of *proximity* [28], [37], [38], as elements placed near each other may appear to be related. White space also influences the overall design style; many modern designs use significant white space. White space 'trapped' between elements can also be distracting [6], [28]. Our solution is to model white space with two types of terms: high-level terms which model the overall white space in the design, and element terms which model pairwise distances or margins.

For our high-level terms, the model encourages white space based on the fraction of the design not covered by an element. The model can also penalize large regions of empty white space based on a distance map to the nearest element, with the mean taken over the entire image.

We next model different kinds of white space with separate energy terms. First, the system measures the mean of the distances for each element to the nearest element, allowing energy terms to increase or decrease the white space between elements. Second, energy terms encourage uniform vertical spacing of text elements by measuring the variance in vertical distances between adjacent elements. Lastly, energy terms also capture margin white space. Border margins for each element are defined as the distance of the bounding box edge to the respective boundary. Energy terms measure the mean of the nearest margin over all text and graphical elements, and control the desired margin size. The final white space in the design results from a combination of these and other terms.

Scale. Elements must be large enough to view, but not so large that the design becomes cluttered and aesthetically displeasing. Other terms like emphasis and white space indirectly affect the scale of elements, but we found it useful to include separate energy terms to model element size. White space in designs with a few elements may be quite different than designs with many elements, so directly modeling scale helps with transferring styles to other designs.

Fig. 6. *Design flow.* The model uses simple heuristics for specifying read-order. *Left:* the unimportant elements are above and to the right of more important elements. *Right:* improved read-order.

Text element size is defined as the element height divided by the number of lines, weighted by a scaling parameter, and normalized by the design height. The size for graphical elements is the bounding box area, normalized by the design size. Energy terms for text and graphical elements encourage larger sizes, and also penalize the variance of text and graphic sizes, which are useful for styles with less contrast between elements. Readability is a basic requirement; the model penalizes elements below a minimum size.

Flow. A good design layout presents information in a clear read-order. However, modeling the visual flow of graphic designs is a difficult open problem. English is read top-down, left-right; designs which deviate from this norm must do so knowingly [28], [35]. Our model uses a simple positioning heuristic which places important elements higher and to the left of less important elements. See Fig. 6 for an example. These flow heuristics also interact with emphasis energy terms, as elements nearer the design center are considered more important.

The overall location of graphical and text elements also affects the design style. The model also uses positioning terms using the mean distances from the margins of the page, for both text and graphical elements. The model can thus specify the rough position of text and graphics for particular styles. We also measure the variance of the element positions for both text and graphical elements.

Overlap and boundaries. Overlapping elements are common in many designs and absent from others. We define several types of overlap, including overlap of elements on text, overlap of text on graphics, and overlap of graphics on other graphics. Users may also provide fixed binary masks, to prevent overlapping important regions such as faces or logos. The model includes an energy term which measures the overlap in these regions. The model also penalizes hard-to-read text using a text contrast measure based on the colour difference between an element and the design before the element was drawn.

The model controls how much elements may extend past the boundaries by computing the fraction of the element beyond the boundary. Similar energy terms for text and important regions are defined.

Unity. Contrast is a common design principle that is often used to differentiate and emphasize certain elements [35],

[37]. Conversely, decreasing contrast between elements increases their unity, and can visually group elements.

We model element unity by allowing users to group elements with scalar group IDs provided as meta-data. The model encourages group members to have a similar size and perceived importance. Energy terms measure the average variance of sizes or importance values for all groups. We also enforce a position constraint that group members should be close.

Previous layout. When improving or retargeting designs, the model uses a previous layout of the same elements. We often wish to preserve properties of the original design such as the sizes of elements. Given a previous design, the model penalizes the deviation of several element properties including the absolute and relative heights, locations, and importance values. The model also compares two global properties of the designs: the overlap of text on graphics and how much elements extend beyond the boundaries.

7 Optimization

The energy function is extremely multi-modal, and the variables are highly coupled because of alignment constraints. To optimize this difficult problem, we follow previous work in layout optimization and use simulated annealing [13], [15]. The optimization takes an initial layout where elements are placed along the left boundary and proposes changes to the elements' positions and scales. Proposed changes resulting in lower energy are always accepted. Early in the optimization, there is a greater probability of accepting layouts with higher energy, allowing escape from local minima. As the optimization progresses, the algorithm is less likely to accept higher-energy layouts. See Appendix D, available in the online supplemental material, for details of the proposal distribution.

Optimization takes approximately 40 minutes on a MacBook Pro (Intel Xeon 2.6 Ghz). The optimizer runs for 30,000 iterations, using a linearly-decreasing annealing schedule. The highly multi-modal nature of the problem often results in the optimizer finding different local minima. This can be useful for sampling layout suggestions for a designer however (e.g., [14]), as the local minima are often visually different. For each result in this paper, we ran eight optimizations in parallel and selected the minimum.

8 Learning Model Parameters

The parameter vector θ that defines a layout energy includes 122 parameters. Manually setting this large number of parameters is prohibitively time-consuming. We use Non-linear Inverse Optimization [1], [17] which learns non-linear model parameters θ based on one or more examples. NIO is an instance of structured learning [39], though unlike most structured learning methods, NIO works for non-linear parameters and continuous outputs. Given an example layout $\mathbf{X}_T$, we assume it is optimal according to an unknown parameter vector θ. In order to estimate θ, NIO iteratively reduces the difference in energy between the example layout $\mathbf{X}_T$ and the optimal layout for the current θ, which is found using optimization. These learned parameters are then applied to other designs to produce a similar style. We also make a few small changes to the algorithm to improve performance, including using the previous optimal layout as the next iteration's starting point. We include a detailed explanation and pseudocode in Appendix E, available in the online supplemental material.

9 Applications

We demonstrate our model with three applications. Separate parameters are learned for each application using NIO and a small number of examples, as the energies of each application differ significantly. For learning styles, there is no previous layout and the system attempts to match a particular style. Both retargeting and improvement measure the difference from a previous layout, but the energy is different. For example, when retargeting it is more important to match the previous layout's perceived element importance. See the Appendix, available in the online supplemental material, for the training data. For all applications, we include more examples in the supplementary materials, available online.

Design synthesis. The most basic goal of the system is to generate design layouts in a variety of styles. Fig. 8 shows examples of synthesized layouts. Based on two example layouts, the system learns the style parameters with NIO and then generates other layouts in the same style. Learning using a single layout is possible, but, because of the model complexity, we found using two similar layouts reduced over-fitting and produced better results.

In Fig. 7 we demonstrate our approach with three landscape-ratio styles: a simple symmetric style, a two column center-aligned style, and a style with a large center graphic with surrounding smaller text. Our results show that the model is able to learn style parameters and reproduce the style on new design problems.

Design retargeting. The system can retarget designs to new sizes. Retargeting is an important task for designers as designs are now often viewed in a variety of sizes and aspect ratios. In our tests, we primarily focus on retargeting between landscape and portrait sizes, but our approach works for arbitrary sizes (Fig. 9). Model parameters were learned from 12 pairs of designs in two different aspect ratios (landscape and portrait). For each design in the pair, the alternate design was used in the previous layout energy terms of Section 6, giving 24 examples total. Retargets on new designs were generated using these learned parameters.

To evaluate our algorithm, we hired a professional graphic designer who was experienced in resizing posters and other print materials to different dimensions. The designer was provided 98 original designs and created layouts in a new size (from portrait to landscape or vice-versa) which matched the original design's style while also being aesthetically pleasing. The designer faced the same constraints as our system, and could only translate and scale elements. In Fig. 10 we show several retargeting examples. We then compared our automatic retargets using AB comparisons on Mechanical Turk. 45 users were paid 5¢ to compare 10 designs based on aesthetics and similarity to the original design. Duplicates were added in each task and inconsistent workers removed.

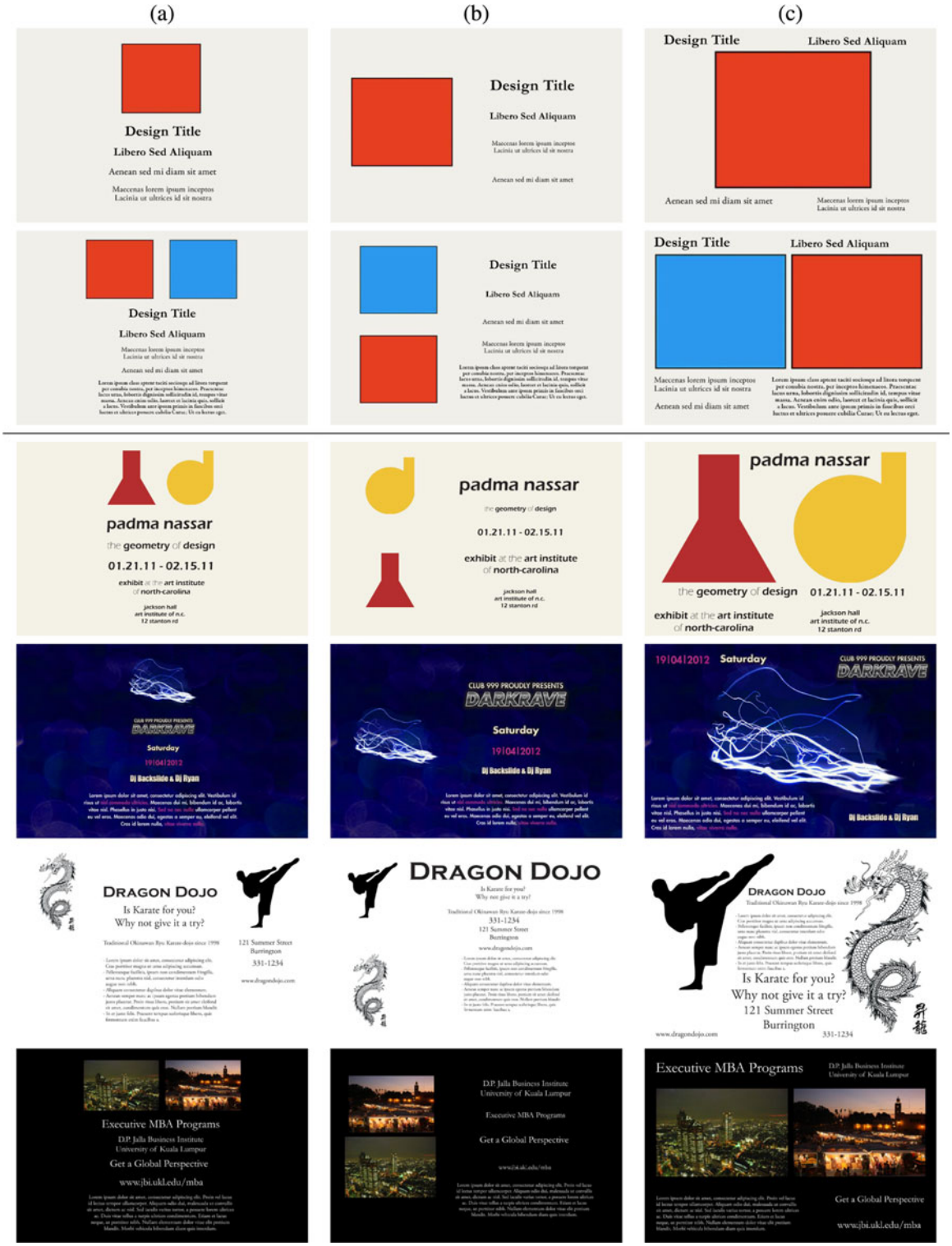

Fig. 7. *Learning design styles.* Parameters are learned from the top two examples of each column and used to generate layouts for other designs. (a) a simple symmetric style, (b) two columns with center-aligned elements, and (c) a large center graphic with smaller text surrounding.

To evaluate how well our algorithm compares to novice users, we performed a crowdsourced version of the same study using MTurk. Users were paid 5¢ per retarget, and took a median of 2.5 minutes per design, similar to the 2.4 minutes for the professional designer. Because of the subjectivity of the task, to encourage higher quality results, users were also informed that designs would be evaluated by other workers and bonuses would be given. In a second study, users were shown nine MTurk retargets and our automatic retarget in random order, and ranked the retargets from best to worst. Workers were paid 5¢ per evaluation. 25 workers were used, with the most inconsistent of those users removed. The top 20 percent of designs received 15¢; the top 10 percent of rankings (measured by distance to

Fig. 8. Layouts generated in a variety of styles. Parameters are learned from example layouts (see supplementary materials, available online).

the mean ranking) received 25¢. Fig. 11 shows an example of the top five retargets with their average rankings.

Fig. 10 shows a comparison of automatic and human retargeting. Figs. 10a and 10b show more complicated designs where our automatic retargets are similar to human retargeting. In Fig. 10c the automatic retarget has produced a reasonable layout, though fairly different than human retargets.

The mean preference of our automatic retargets compared to designer retargets, after removing any layouts used as training data, was 0.39 (see Fig. 12). However, analyzing the individual AB tests using the binomial test show that many failed to show a statistically significant preference for the designer retargets, suggesting the algorithm often retargets reasonably. To compare with novice MTurk retargets, we performed the ordering test described earlier. Our retargets achieved an average rank of 4.61 (std. err of 0.29) . Fig. 12 shows a histogram of the rankings for our automatic retargets after sorting by the mean rank. While

Fig. 9. *Design retargeting.* Given a design (top left), the system can retarget to a variety of sizes and aspect ratios.

our approach cannot consistently beat the best human retargets, we do perform better than the average MTurk user. We use a one-sample Student-t test to evaluate if the mean of all automatic retarget rankings is less than 5.5, the midpoint between rank 1 and 10 ($p < 0.05$). As further support for our model, in the Appendix, available in the online supplemental material, we show statistically significant correlations between many energy terms and the ranking scores.

Considering the problem's difficulty and lack of previous automatic algorithms, merely replicating novice human ability is significant. Furthermore, while users usually prefer designer retargets to ours, they often make no distinction between the two, indicating that our automatic retargeting is often doing as well as a professional.

Design improvement. The system can take an existing layout and improve it to better match principles of graphic design. Parameters are learned using 12 examples of an original and improved design. Given a new design, the learned parameters are used to optimize a new improved version. Fig. 14 shows the results of our approach improving a variety of designs, from very poor initial layouts which are changed significantly, to good initial layouts which are only changed slightly. For example, Fig. 14a is improved by grouping and aligning the text elements. Fig. 14d is improved by increasing the graphic size while aligning the elements. Figs. 14b and 14e are changed more significantly, by changing the placement of elements to improve the read order, as well as improving alignment, and symmetry. The right columns gives some failure cases of our algorithm. In Fig. 14c the symmetry of the text in the original is lost, producing a worse design. In Fig. 14f, the new design is significantly worse due to a poor grouping of elements in the top, resulting in smaller text and a more confusing read order.

To evaluate our improvement approach, we performed another MTurk study. The manually created and ordered designs from the retargeting task were divided into three groups: the best rated designs, the worst rated designs, and all designs. Out of the nine human retargets, the top 2, bottom 2, and two random designs were each chosen, giving

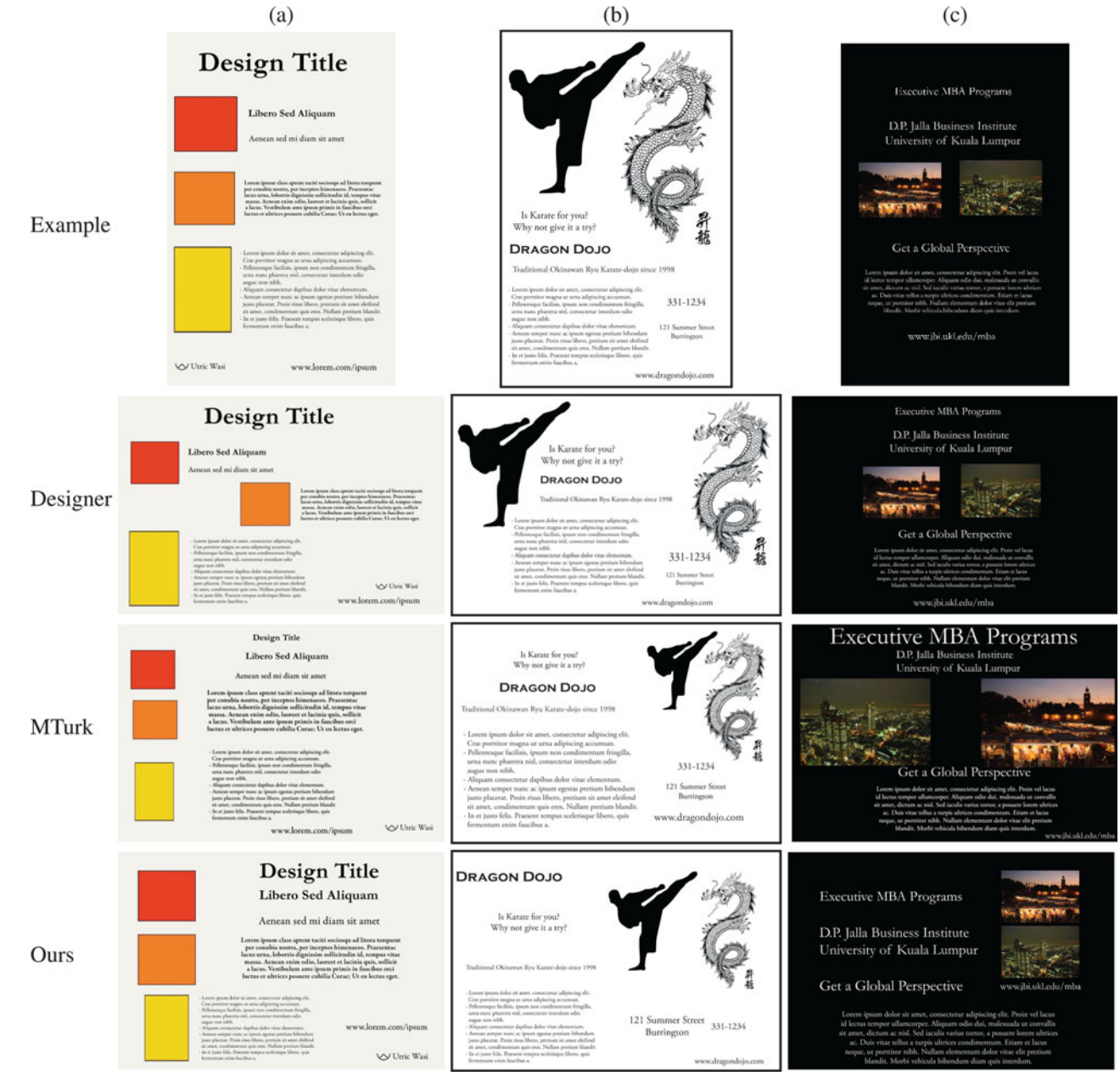

Fig. 10. *Portrait-to-landscape retargeting*. We show retargeting results from an example layout by a professional designer, the best crowdsourced retarget (out of nine), and our automatic retarget.

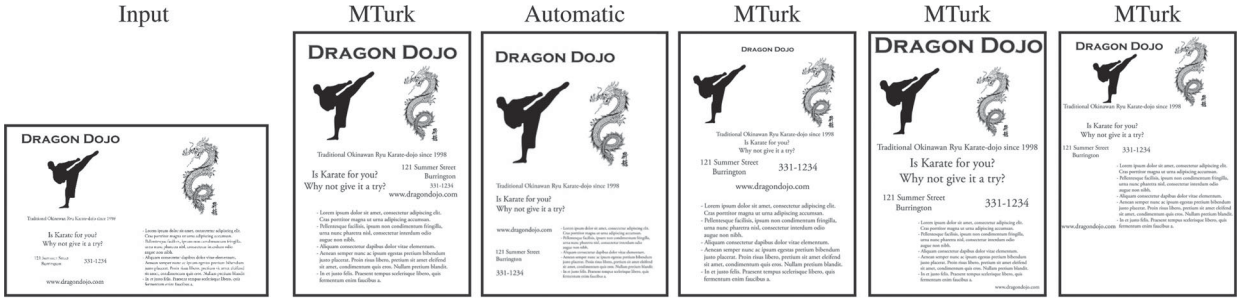

Fig. 11. *Retarget ordering*. The portrait retargets were manually created by MTurk users in one task, and ranked in a second task. The mean/std dev of all rankings are reported, with the designs sorted by mean rank.

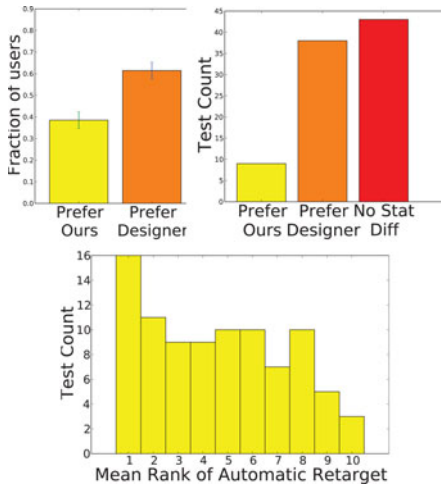

Fig. 12. *Automatic retarget evaluation. Top left*: mean preference of AB comparisons between automatic and designer retargets. *Top right*: histogram of tests which had a statistically significant preference for either retarget. *Bottom*: histogram of rankings for automatic retargets compared to novice MTurk users (1 is the best rank). Our automatic retargets cannot beat designer retargets, though often perform comparably, and are often highly rated compared to novice human retargets.

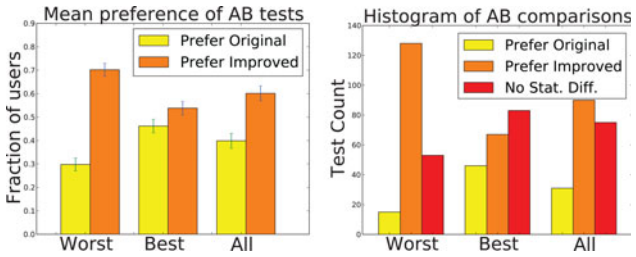

Fig. 13. *Design improvement evaluation*. We show improvement results for the worst, best, and all user designs. *Left*: the overall mean preference for the original or the improved version (two std err). *Right*: histogram of tests which had a statistically significant preference for either original or improved version.

196 designs per set. These designs were then optimized with the learned improvement parameters. Finally, 45 MTurk users selected their preferred design in an AB comparison, with a randomized left/right position for the improved design. Users were paid 5¢ to compare 10 designs based on aesthetics and clarity. Duplicates were added in each task and inconsistent workers removed.

Fig. 14. *Design improvement*. Based on layouts created by MTurk users (top), we generate improved layouts (below). $\mathbf{p}_i$ are the fraction of MTurk users who prefer the improved designs. The right column shows failure cases.

Fig. 13 shows the overall preference for the original designs compared to the improved versions. We also show a histogram of the tests which had a statistically significant preference for the original or improved (using the binomial test with $p < 0.05$), or where there was no statistical difference. Our approach often improves the worst designs while matching the best designs. For the overall set of designs, our approach often improves the design or produces no difference; only rarely does the algorithm produce a worse design. These results show that the system successfully models basic principles of graphic design, and can generate appealing layouts.

10 Conclusions

Automatic tools for understanding and creating graphic designs are important for both professional and novice designers. Design is an extremely difficult task, and the vast number of devices and viewing conditions for designs have increased the burden on designers significantly. However, many books on graphic design principles are vague and difficult to build tools from directly. By contrast, we model design principles explicitly, synthesize layouts using optimization, and directly evaluate modeling choices with user studies. This general approach allows a deeper understanding of graphic design principles, and will hopefully lead to tools for aiding novice and expert designers.

There are currently several limitations with our approach. Though complex, our model barely scratches the surface of possible graphic design layout styles. The model only optimizes element position and scale, and ignores rotations, font types, text line breaks, and optional elements. Other possible extensions include modeling of element read-order and more complex constraints as in article layouts.

Our optimization and learning procedure are currently too slow for real-time interaction. Predicting element importance is currently an expensive operation; investigating simpler models of importance is potential future work. Our model also performs expensive image-based operations like compositing; a vector-based representation could be significantly faster. While we run multiple optimization in parallel, our approach is not intrinsically parallelizable due to the simulated annealing algorithm. However, parallel tempering has been used to parallelize layout synthesis on the GPU [14]. To address these limitations, we have begun development of a GPU-based model which greatly improves efficiency and allows our approach to be extended to interactive design applications.

Acknowledgments

This work was supported by Adobe, NSERC, and CIFAR.

References

[1] C. K. Liu, A. Hertzmann, and Z. Popovic, "Learning physics-based motion style with nonlinear inverse optimization," *ACM Trans. Graph.*, vol. 24, pp. 1071–1081, 2005.

[2] D. E. Knuth, *The TeXbook*. Reading, MA, USA: Addison-Wesley, 1986.

[3] C. Jacobs, W. Li, E. Schrier, D. Bargeron, and D. Salesin, "Adaptive grid-based document layout," in *Proc. ACM SIGGRAPH*, 2003, pp. 838–847.

[4] N. Damera-Venkata, J. Bento, and E. O'Brien-Strain, "Probabilistic document model for automated document composition," in *Proc. ACM Symp. Document Eng.*, 2011, pp. 3–12.

[5] N. Hurst, W. Li, and K. Marriott, "Review of automatic document formatting," in *Proc. 9th ACM Symp. Document Eng.*, 2009, pp. 99–108.

[6] S. J. Harrington, J. F. Naveda, R. P. Jones, P. Roetling, and N. Thakkar, "Aesthetic measures for automated document layout," in *Proc. ACM Symp. Document Eng.*, 2004, pp. 109–111.

[7] H. Y. Balinsky, A. J. Wiley, and M. C. Roberts, "Aesthetic measure of alignment and regularity," in *Proc. 9th ACM Symp. Document Eng.*, 2009.

[8] H. Balinsky, J. Howes, and A. Wiley, "Aesthetically-driven layout engine," in *Proc. 9th ACM Symp. Document Eng.*, 2009, pp. 119–122.

[9] C. Gonzalez-Morcillo, V. J. Martin, D. Vallejo-Fernandez, J. J. C Sanchez, and J. Albusac, "Gaudii: An automated graphic design expert system," in *Proc. 22nd Conf. Innovative Appl. Artif. Intell.*, 2010, pp. 1775–1780.

[10] A. Jahanian, L. Jerry, D. Tretter, L. Qian, N. Damera-Venkata, E. O'Brien-Strain, L. Seungyon, F. Jian, and J. Allebach, "Automatic design of magazine covers," in *Proc. SPIE.*, vol. 8302, pp. 83020N–83020N-8, 2012.

[11] L. Purvis, S. Harrington, B. O'Sullivan, and E. C. Freuder, "Creating personalized documents: An optimization approach," in *Proc. ACM Symp. Document Eng.*, 2003, pp. 68–77.

[12] J. Geigel and A. Loui, "Using genetic algorithms for album page layouts," *IEEE Multimedia*, vol. 10, no. 4, pp. 16–27, Oct. 2003.

[13] M. Agrawala and C. Stolte, "Rendering Effective Route Maps," in *Proc. ACM SIGGRAPH*, 2001, pp. 241–249.

[14] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun, "Interactive furniture layout using Interior design guidelines," in *Proc. ACM SIGGRAPH*, 2011, pp. 87:1–87:10.

[15] L.-F. Yu, S. K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. Osher, "Make it home: automatic optimization of furniture arrangement," in *Proc. ACM SIGGRAPH*, 2011, pp. 86:1–86:12.

[16] K. Gajos and D. S. Weld, "Preference elicitation for interface optimization," in *Proc. 18th Annu. ACM Symp. User Interface Softw. Technol.*, 2005, pp. 173–182.

[17] I. Vollick, D. Vogel, M. Agrawala, and A. Hertzmann, "Specifying label layout styles by example," in *Proc. 20th Annu. ACM Symp. User Interface Softw. Technol.*, 2007, pp. 221–230.

[18] S. Mao, A. Rosenfeld, and T. Kanungo, "Document structure analysis algorithms: A literature survey," in *Proc. SPIE*, vol. 5010, pp. 197–207, 2003.

[19] M. Shilman, P. Liang, and P. Viola, "Learning non-generative grammatical models for document analysis," in *Proc. IEEE 10th Int. Conf. Comput. Vis.*, 2005, pp. 962–969.

[20] J. Talton, L. Yang, R. Kumar, M. Lim, N. D. Goodman, and R. Měch, "Learning design patterns with bayesian grammar induction," in *Proc. 25th Annu. ACM Symp/ User Interface Softw. Technol.*, 2012, pp. 63–74.

[21] R. Rosenholtz, N. R. Twarog, N. Schinkel-Bielefeld, and M. Wattenberg, "An intuitive model of perceptual grouping for HCI design," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2009, pp. 1331–1340.

[22] J. Müller-Brockmann, *Grid Syst. in Graphic Design*. Sulgen, Switzerland: Niggli Verlag, 1996.

[23] S. Baluja, "Browsing on Small Screens," in *Proc. 15th Int. Conf. World Wide Web*, 2006, pp. 33–42.

[24] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan, "Syntactic segmentation and labeling of digitized pages from technical journals," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 7, pp. 737–747, Jul. 1993.

[25] R. Kumar, J. O. Talton, S. Ahmad, and S. R. Klemmer, "Bricolage: Example-based retargeting for web design," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 2197–2206.

[26] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir, "A comparative study of image retargeting," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 160:1–160:10, 2010.

[27] L. Liu, R. Chen, L. Wolf, and D. Cohen-Or, "Optimizing photo composition," *Comput. Graph. Forum*, vol. 29, pp. 469–478, 2010.

[28] A. White, *The Elements of Graphic Design*. New York, NY, USA: Allworth Press, 2002.

[29] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, pp. 145–175, 2001.

[30] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[31] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 2106–2113.
[32] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-aware saliency detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2376–2383.
[33] T. Liu, J. Sun, N. Zheng, X. Tang, and H. Shum, "Learning to detect a salient object," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2007.
[34] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.
[35] L. Graham, *Basics of Design: Layout and Typography for Beginners*. Albany, NY, USA: Delmar, 2002.
[36] R. Landa, *Graphic Design Solututions*. Stamford, CT, USA: Cengage Learning, 2010.
[37] R. Williams, *The Non-Designer's Design Book*. San Francisco, CA, USA: Peachpit, 2008.
[38] J. H. Bear. (2012) Principles of Design. [Online]. Available: http://desktoppub.about.com/od/designprinciples/.
[39] S. Nowozin and L. Lampert, "Structured learning and prediction in computer vision," in *Found. Trends® Comput. Graph. Vis.*, vol. 6, 2011, pp. 185–365,.

Peter O'Donovan received the BSc degree in computer science from the University of Saskatchewan in 2005, and the MSc degree from the University of Toronto in 2009. He has worked in the past for Adobe Systems and is currently working toward the PhD degree with Aaron Hertzmann in modeling of aesthetic preferences.

Aseem Agarwala received the PhD degree in the Department of Computer Science and Engineering from the University of Washington in June 2006, where he is currently an affiliate assistant professor. He is a principal scientist at Adobe Systems, Inc. He won an Honorable Mention (runner-up) for the 2006 ACM Doctoral Dissertation Award, and is an associate editor for the *ACM Transactions on Graphics*. His inventions can be found in multiple products, including Adobe Photoshop, Premiere, and After Effects. He is a member of the IEEE.

Aaron Hertzmann received the BA degree in computer science and art/art history from Rice University in 1996 and the PhD degree in computer science from New York University in 2001. He is a senior research scientist at Adobe Systems. He was a professor at the University of Toronto for 10 years, 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 a fellow of the Canadian Institute for Advanced Research, and an associate editor for the *ACM Transactions on Graphics*. His awards include the MIT TR100 (2004), an Ontario Early Researcher Award (2005), 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). He is a senior member of the IEEE.