BUTTONTIPS: DESIGNING WEB BUTTONS WITH SUGGESTIONS

Dawei Liu *Ying Cao* *Rynson W.H. Lau* *Antoni B. Chan*

Department of Computer Science, City University of Hong Kong
daweiliu2-c@my.cityu.edu.hk caoying59@gmail.com {rynson.lau, abchan}@cityu.edu.hk

ABSTRACT

Buttons are fundamental in web design. An effective button is important for higher click-through and conversion rates. However, designing effective buttons can be challenging for novices. This paper presents a novel interactive method to aid the button design process by making design suggestions. Our method proceeds in three steps: 1) *button presence prediction*, 2) *button layout suggestion* and 3) *button color selection*. We investigate two distinct but complementary interfaces for button design suggestion: 1) *region selection interface*, where the button will appear in a user-specific region; 2) *element selection interface*, where the button will be associated with a user-selected element. We compare our method with an existing website building tool, and show that for novice designers, both interfaces require significantly less manual efforts, and produce significantly better button design, as evaluated by professional web designers.

Index Terms— web design, data-driven method, button layout

1. INTRODUCTION

Buttons are fundamental in web design as they often represent some call-to-action (CTA), which prompts visitors to perform certain actions (e.g., sign up) to achieve the ultimate goal of the webpage. An effective button is important for higher click-through and conversion rates, e.g., it can direct users to click it to subscribe services, order products or sign up for an account. However, creating effective designs can be challenging since designers need to balance numerous properties of the buttons including position, size and color [1]. Any small changes in these factors may have significant influence upon its functionality [2]. Existing tools range from simple template-based interfaces like WIX [3], to complex systems like Illustrator. However, these tools do not provide suggestions and users (novice designers in particular) have difficulties to decide where to put a button, how large it should be and what color should be used. Therefore, designing effective buttons can be frustrating for novice designers.

In this paper, we propose a novel interactive method to aid the button design process by making button design suggestions. In particular, we take a data-driven approach, wherein we learn several regression models to capture how position, size and color of buttons are determined by professional designers. With these models, we investigate two interfaces for button design suggestion. First, we develop the *region selection interface* , where the button suggestions appear inside the user-specific region. Second, we develop the *element selection interface*, where button suggestions are associated with the user-selected element.

Given a web design being edited, a user can use either interface to indicate where to place a button. Our method will then provide several button suggestions with size and alignment determined. The user can select a preferred button suggestion. Finally, our method will automatically select the color to fill the button. The data-driven strategy allows us to learn implicit principles of button design from examples and thus generate buttons that are visually and functionally analogous to what professional web designers create.

To evaluate our method, we have conducted several user studies to compare our method against a commercial web design tool. Results show that the proposed method significantly reduces the time required for the button design process, while remarkably improving the quality of the produced buttons.

2. RELATED WORK

To the best of our knowledge, there is little work automating button design on webpages. We first discuss prior works on layout generation, and then review some recent efforts on aiding web design.

2D Layout Problems. 2D layout problems have been receiving growing interests. Cao et al. [4, 5] introduced statistical models for automating comic layout and content synthesis. Reinert et al. [6] proposed a method for graphical primitive packing. Jiang et al. [7] proposed a method for layout regularization. O'Donovan et al. [8] presented a system for suggesting single-page graphic design layout. However, all of these models are using existing elements to optimize a current design, and unable to create new elements to add to an existing design, and thus cannot be used to address our problem. We follow this line of research, but focus on an unexplored problem – aiding users to create professional-looking button designs.

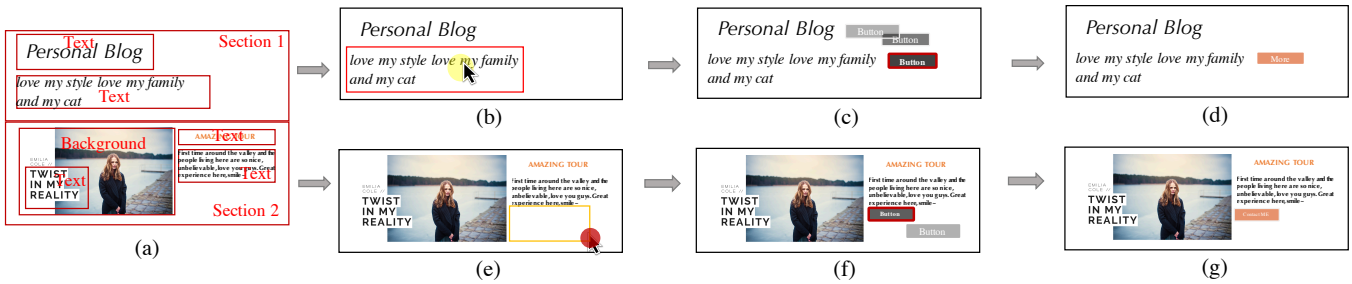


Fig. 1. Button Design Interface. Given a web design with some already-placed page elements (a), a user may add a button to it by either selecting an element that the button is associated with (the text element outlined in red in (b)) or drawing a region to indicate the spatial scope that the button should lie within (the orange rectangle in (e)). Our method will then provide multiple layout suggestions, i.e., position and size of buttons (gray buttons in (c) and (f)). The user may select a desired button (outlined in red in (c) and (f)). Finally, our method will automatically select the color to fill the button as shown in (d) and (g).

Web Design Analyses. There are some works for analyzing and assisting web designs. For example, Kumar et al. [9] introduced a scalable platform for crawling a large-scale repository of webpages. Lee et al. [10] presented an interactive interface for navigating a large corpus of web design examples. Doosti et al. [11] and Jahanian et al. [12] studied styles of web design in history and Bylinskii et al. [13] predicted visual importance of graphic designs. Pang et al. [14] optimized webpage elements to direct user attention. Zhao et al. [15] proposed a model to predict web font properties.

3. INTERFACE OVERVIEW

We next provide a high-level view of the button design interface (Fig. 1). See the Supplemental Video for demonstrations. The key goal of our interfaces is to aid novices in the button design process.

Region Selection Interface. In this mode, our method generates a set of button design suggestions in a user-specified region. In particular, a user can draw a rectangle on a web design being edited to indicate where to place a button. Our method will analyze the region properties (e.g., width, height and center position) to rank the button suggestions.

Element Selection Interface. Our method can also help suggest button design for an existing element in an existing web design. Particularly, a user can mouse-click to choose which element a button should be associated with. This interface is useful to generate element-associated buttons.

4. METHOD OVERVIEW

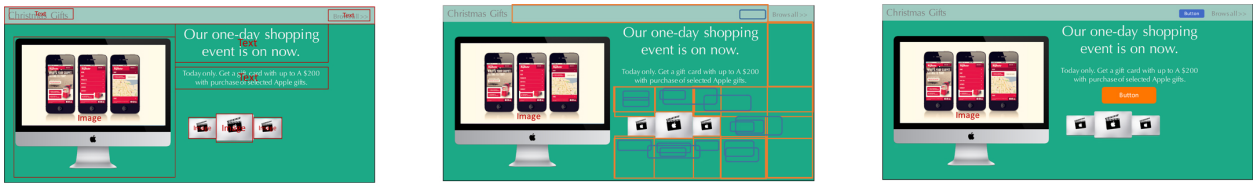
To provide button design suggestions, our method proceeds in three steps: 1) *button presence prediction*, in which we predict button presence for a rough region; 2) *button layout suggestion*, in which a button is automatically placed and scaled; 3) *button color selection*, in which the button color is automatically selected. We leverage professional web design examples to train several machine learning models to perform these steps.

Web design dataset. We collect 100 professionally designed webpages as our training dataset. In particular, we access these webpages from WIX [3], where professional web designers provide their webpages for novice designers as reference. In addition, we choose webpages that contain buttons as Call-To-Action buttons like "Subscribe", which represent the main task of the webpages.

5. BUTTON PRESENCE PREDICTION

Candidate Region Generation. Before designing a button on a webpage, professional web designers often have a rough idea about where to place the button. To mimic this process, we define a "candidate region" by enforcing two properties: 1) *Rectangular area*, which is widely used in grid-based system; 2) *Snapping to other elements*, which is easy to extract local context for button presence analysis (see Fig. 2). Our proposed "candidate region" is also inspired by the concept of "object proposal" in computer vision area, where it is very useful to detect a bounding box containing an object before classifying the object types. Here, our strategy is to "detect" a set of candidate regions that possibly contain a button before predicting the button attributes. Based on the two properties, we propose a seed expansion method to detect candidate regions for a web design (see Algorithm 1 and Fig. 3). Please check our supplemental materials for the detailed algorithm.

Button Presence Score. There can be some candidate regions that are less likely to contain a button. Hence, we need to estimate a button presence score. So, we train a one-class SVM model [16] by using our hand-designed features: (1) the candidate region's location, size, area and aspect ratio; (2) the k-th neighbor's width, height, area and aspect ratio; (3) position differences of the upper-left, center, and bottom-right points between the k-th neightbor and candidate region. k is defined in the following order: top, left, bottom, right. This results in a 48-dimensional feature vector.



(a) (b) (c)

Fig. 2. Overview of our method. Given a web design being edited without buttons (a), the *button presence prediction* step proposes a set of candidate regions (orange boxes) that roughly contains buttons, and the *button layout prediction* step will then provide a button layout (blue box) for each candidate region. After the user selects a button layout, the *color selection* step will automatically select suitable color for it (c). Best viewed in color.

Algorithm 1 Seed expansion algorithm. Please check our supplementals for detailed description.

procedure EXPANDSEED($Seed_i$)
  Queue $\mathcal{Q} \leftarrow Seed_i$
  List $\mathcal{L} \leftarrow null$
  while $\mathcal{Q} \neq null$ **do**
    $S^{new} \leftarrow \mathcal{Q}$.pop()
    $S^H \leftarrow$ HorizontalExpand(S^{new})
    $S^V \leftarrow$ VerticalExpand(S^{new})
    $R \leftarrow (\{S^V, S^H\} \bigcup \mathcal{L}) \ominus \mathcal{L}$ ▷ new regions
    $\mathcal{Q}$.append(R)
    $\mathcal{L} \leftarrow \mathcal{L} \bigcup R$
  end while
  return rs ▷ candidate regions for $Seed_i$
end procedure

6. BUTTON LAYOUT PREDICTION

Width and Height. The feature vector aforementioned is also useful to predict width and height of a button by using Gaussian process regression [17]. We find that this method is not able to capture the following two design principles: 1) set button height equal to its left input box, if it exists; and 2) set button height equal to its left/right button, if it exists. Thus, we simply predefine them in our model.

Position. In web design, visual alignment is more important than precise position, which can also be extremely complicated to predict. Inspired by [18, 19], we create a set of customized alignment types to place a button (see Fig. 4), We then use random forests [20] to predict one of the 9 alignment types with respect to its neighboring elements in both horizontal and vertical directions.

7. INCORPORATING USER CONSTRAINTS

To incorporate the user-specified constraints, i.e., *region selection interface* and *element selection interface*, we build an intention map on the input web design to capture the user behavior. For the *region selection* interface, our method uses the user-specific region structures (i.e., width, height and

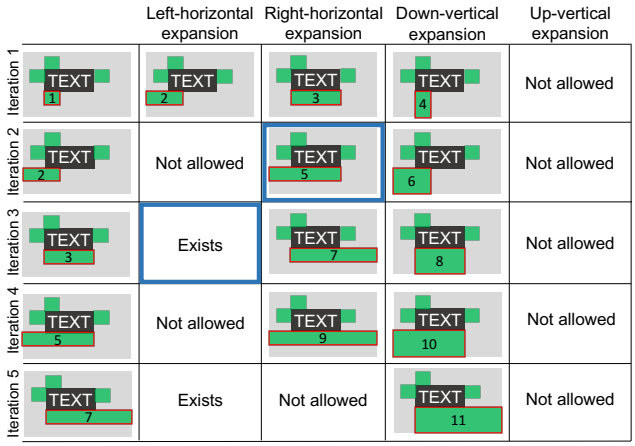


Fig. 3. Seed expansion method. We define a set of small squares (green) as seeds and attach them to the four edges of an element. Each row shows the generated regions resulting from an iteration of expanding one specified seed (red). A newly generated region will be ignored if it already exists ("Exists") or it cannot be expanded anymore ("Not allowed"). For example, the blue cell in iteration 3 is the same as the blue cell iteration 2. Thus, we mark it "Exists".

center position) to parameterize a Gaussian-weighted map $M(x, y) = \mathcal{N}([x, y]; \mu, \Sigma)$ as the intention map. For the *element selection* interface, our method uses the selected element structures instead. The final score of the button centered at (x', y') is defined as a weighted sum of its *button presence score* and its corresponding value at the intention map:

$$S_f = \alpha S_p + \beta \mathcal{N}([\hat{x}, \hat{y}]; \mu, \Sigma), \quad (1)$$

where S_p is computed by multiplying the confidence scores output by the models for button presence and layout prediction. α and β are both set to 0.5 in our implementation. We present the top 5 button suggestions to the user, and translate the normalized score of each button to the degree of transparency of the button.

8. BUTTON COLOR SELECTION

After predicting the layout of the button, we next aim to select the button color. Our method suggests a color based on

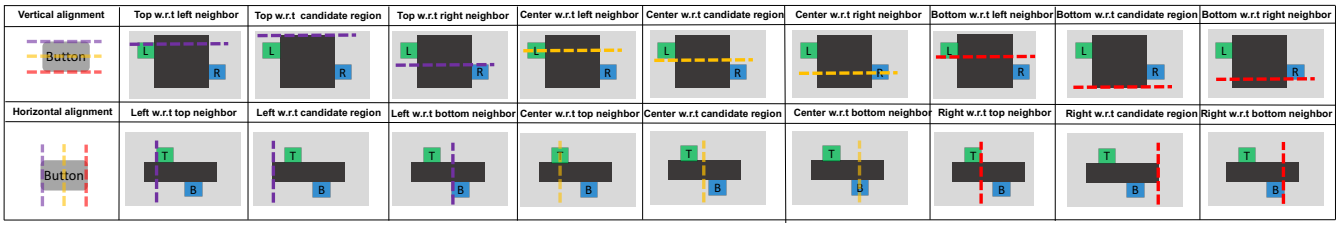


Fig. 4. Horizontal and vertical alignment types. For vertical alignment types (the first two rows), green/black/blue boxes represent left neighbor/candidate region/right neighbor to a button. For horizontal alignment types (the last two rows), green/black/blue boxes represent top neighbor/candidate region/bottom neighbor to a button.

balancing the *compatibility*, where button color is harmonious to its surrounding elements, and *contrast*, where the button is standing out from its local context. For compatibility, we apply the approach from [21]. In particular, we compose a color theme (5 colors) as input for [21], by extracting 4 colors from a 400×300 region centered around the button using K-means clustering, the predicted button color c_b as the 5-th color, then we can obtain a color compatibility score. For contrast, we compute the Euclidean distance as the evaluation metric. Then the button color generation amounts to finding a color that maximizes a score composed of compatibility and contrast scores:

$$c_b^* = \underset{c_b}{\operatorname{argmax}}\, \gamma C(c_b) + \lambda T(c_b), \tag{2}$$

where $C(c_b)$ is the compatibility score between c_b and the 4 extracted colors, $T(c_b)$ is the euclidean distance between c_b and the average of the 4 extracted colors, $\gamma = 0.7$ and $\lambda = 4.5$ are the weight factors. Rather than searching over the whole color space, consider a list of 7 web-friendly colors from the colormap theme "lines" in MATLAB[1].

9. EVALUATION

To evaluate the effectiveness of our method, we have conducted several user studies where participants are asked to design buttons using our interfaces and an existing web design tool without suggestions. We then evaluate the quality of the generated button designs using both quantitative metrics and subjective ratings from professional designers.

9.1. User Study Procedure

For our study, we recruited 12 novice users. We divided them into 3 groups to design buttons for a given web design. Each group used one of the three design interfaces: the *element selection* interface, the *region selection* interface, and the traditional drag-and-drop interface (either *Photoshop* or *Powerpoint* as their preference). Note that for the traditional interface, the participants needed to manually position and resize a rectangle button and used a color wheel for color selection. A total of 32 test webpages were used in our study. We removed some of the existing buttons from the webpages, and asked the participants to provide new button designs for the webpages. Those removed buttons were used to serve as the ground-truth for further comparison.

During the study, each participant was assigned to design buttons for 8 webpages, which were randomly selected from the 32 webpages. For each webpage, a participant was given the elements or regions that the buttons should be associated with, and was asked to design the buttons using one of the three interfaces.

9.2. Evaluation Metrics

Quantitative Metrics. We use the original buttons as ground-truth to evaluate the quality of the buttons generated by the participants. We use the following quantitative metrics: 1) *Position* - the root-mean-square-error (RMSE) between upper-left corner coordinates of the ground-truth and the generated button. 2) *Size* - the RMSE of the button size between the ground-truth and generated button. 3) *Color* - the RMSE of the button color in the Lab color space, between the ground-truth and the generated button. 4) *Jaccard index* - the ratio of the intersection area to the union area between the ground-truth and generated button.

Ratings from Professional Web Designers. We recruited 21 professional web designers to rate the button designs generated in the user study, by using a scale of 1 (worst) to 5 (best). As we found that our *region selection interface* and *element selection interface* perform statistically equal (Section 9.3), we only ask web designers to rate our *region selection interface*. Then, for each webpage, we asked the designers to rate 4 versions of it: button by our tool (with the *region selection interface*), button by the traditional tool, the ground-truth button, and the "best fitting" button. The "best fitting" button is the one suggested by our tool that is most similar to the ground-truth button in terms of button Jaccard index value. This is to provide an upper bound performance of our method, without being influenced by users' preferences.

9.3. Results and Analysis

Fig. 6 shows some button designs created by participants using our tool and the traditional tool. Fig. 5 shows the average time for our participants to complete an 8-button-design

[1]https://www.mathworks.com/help/matlab/ref/colormap.html

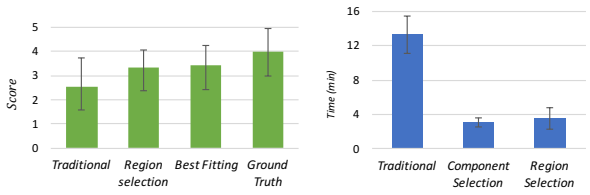


Fig. 5. Left: the rating from professional web designers on different types of button designs. Right: the average time used by participants to complete one design session using each of the three interfaces.

session using each of the three interfaces. In particular, we note that participants using our interfaces completed the design task approximately three times faster than those using the traditional interface.

Quantitative results. Fig. 7 shows the comparison of the button designs from the ground-truth, the traditional tool, our tool, and "best fitting" buttons, using different quantitative metrics aforementioned. Our tool has smaller position and size errors, and a larger Jaccard index value than the traditional tool. This means that our tool enables users to design buttons similar to the professional ones than the traditional tool. On the other hand, the traditional tool has a slightly lower color error than ours. This is probably because we only consider 7 discrete candidate colors in predicting button colors, for computational efficiency. We believe that this can be improved by increasing the number of candidate colors, at the cost of a higher computational time.

Among the buttons of our tool (*region selection interface* or *element selection interface*), the traditional and the "best-fitting", there is a significant difference in the position error, color error and Jaccard index (Kruskal-Wallis test, $p < 0.05$). In addition, there are no significant differences in the size error (Kruskal-Wallis test, $p > 0.05$). For the position error and Jaccard index, our upper-bound performance ("Best fitting") is much better than the traditional tool (Wilcoxon signed-rank test, $p < 0.001$ for position; $p < 0.001$ for Jaccard index). Also, there are no significant differences between our *region selection interface* and *element selection interface* by position, size and color errors and Jaccard index (Kruskal-Wallis test, $p > 0.05$, respectively).

Designer ratings. We show the ratings from the web designers on button designs produced using the four approaches as shown in Fig. 5. Overall, there are significant differences in ratings among the four types of button designs (Kruskal-Wallis test: $p < 0.001$). In particular, while the ground-truth was rated significantly higher than the best-fitting and our tool (region selection), the best-fitting and our tool (region selection) were rated significantly higher than the traditional tool (Wilcoxon signed-rank test: $p < 0.001$; $p < 0.001$). This result suggests that our method can significantly reduce the quality gap between buttons generated by novices and those generated by professional web designers.

Blind test for button scores. Finally, we evaluate the scores used for the button transparency by comparing with the "Best Fitting" button, which is most similar to the ground-truth. In particular, we record the frequency that the top-1 and top-2 scoring buttons contain the "Best Fitting" button. We also record the frequency of the top-scoring buttons that are selected by users during the user study (with button transparency disabled).

The top-scoring button matched the "Best Fitting" button (closest to the ground-truth) 74% of the time, while one of the top-2 scoring buttons matched the best-fitting button 87% of the time. This suggests that the score used for the transparency is accurate, and accepting the suggestion can lead to the best button design. With button transparency disabled, users selected the highest-scoring button 66% of the time, or one of the top-2 buttons 79% of the time.

10. CONCLUSION AND DISCUSSION

In this paper, we propose a novel method to aid novices in the button design process, by providing design suggestions. Our method can automatically predict button layout and select button color, allowing the users to design high-quality buttons with ease. We envision that our tool can be easily integrated into existing commercial web design tools (e.g., [3]) to make web design more accessible to the general population.

As a future work, we would like to incorporate global information into our button layout prediction method and use more complicated CSS properties to generate more nuanced button styles. In this paper, we only address button design. However, our framework is generic and can be adapted to help design other webpage elements (e.g., input field), which is left as a future work.

Acknowledgements. This work was supported by the Research Grants Council of the Hong Kong SAR, China (CityU 11200314).

11. REFERENCES

[1] Rakesh Soni, "Why the cta button is the most important part of your site," `http://blog.loginradius.com/2015/05/cta-button-site/`, 5 2015.

[2] VWO, `https://vwo.com/`, 2010.

[3] WIX, `http://www.wix.com/`, 2006.

[4] Cao, Y., Chan, A.B. and Lau, R.W., "Automatic stylistic manga layout," *ACM TOG*, 2012.

[5] Cao, Y., Lau, R.W. and Chan, A.B., "Look over here: Attention-directing composition of manga elements," *ACM TOG*, 2014.

[6] Reinert, B., Ritschel, T. and Seidel, H.P., "Interactive by-example design of artistic packing layouts," *ACM TOG*, 2013.

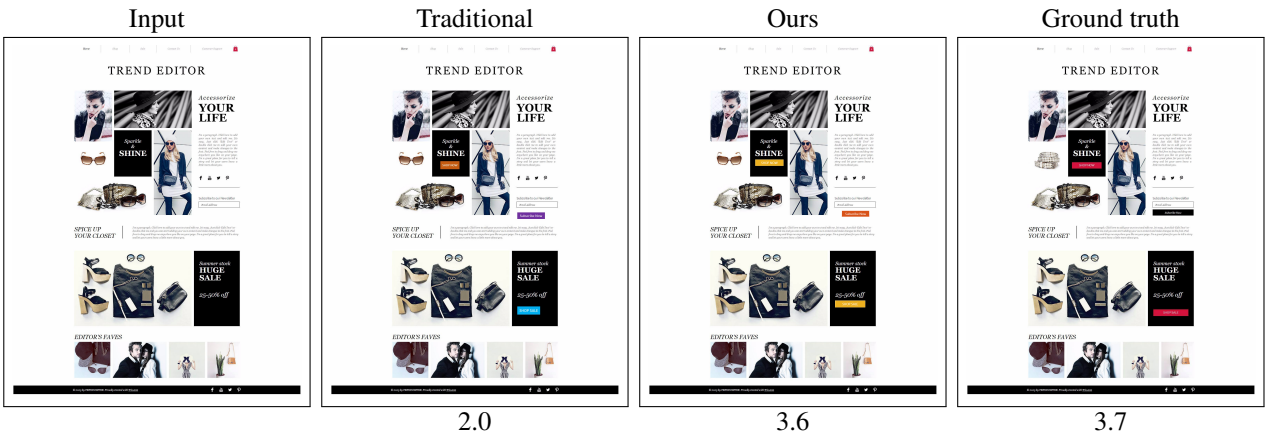
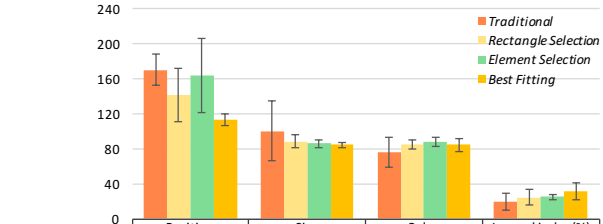


Fig. 6. Button design comparison based on using the traditional tool (Traditional), our tool (Ours) and from the ground truth. We show the average rating score by professional designers below each webpage. Best viewed in color. Please also check our supplemental materials for more results with higher resolution.



	Position	Size	Color	Jaccard index (%)
Traditional	170.6	100.9	76.6	20.00
Rectangle Selection	141.3	89.0	85.9	24.70
Element Selection	164.2	86.1	88.8	25.2
Best Fitting	113.2	85.1	84.7	32.0

Fig. 7. Comparison of the traditional tool and our tool using different metrics. For position, size, and color, a smaller value indicates a smaller error. For the Jaccard Index, a higher value indicates a larger overlap with the ground-truth.

[7] Jiang, H., Nan, L., Yan, D.M., Dong, W., Zhang, X. and Wonka, P., "Automatic constraint detection for 2d layout regularization," *IEEE TVCG*, 2016.

[8] O'Donovan, P., Agarwala, A. and Hertzmann, A., "Designscape: Design with interactive layout suggestions," in *Proc. ACM SIGCHI*, 2015.

[9] Kumar, R., Satyanarayan, A., Torres, C., Lim, M., Ahmad, S., Klemmer, S.R. and Talton, J.O., "Webzeitgeist: Design mining the web," in *Proc. ACM SIGCHI*, 2013.

[10] Lee, B., Srivastava, S., Kumar, R., Brafman, R. and Klemmer, S.R., "Designing with interactive example galleries," in *Proc. ACM SIGCHI*, 2010.

[11] Doosti, B., Crandall D.J. and N.M. Su, "A deep study into the history of web design," in *Proc. ACM on Web Science Conference*, 2017.

[12] Jahanian, A., Isola, P. and Wei, D., "Mining visual evolution in 21 years of web design," in *Proc. ACM SIGCHI*, 2017, pp. 2676–2682.

[13] Bylinskii, Z., Kim, N.W., O'Donovan, P., Alsheikh, S., Madan, S., Pfister, H., Durand, F., Russell, B. and Hertzmann, A., "Learning visual importance for graphic designs and data visualizations," *ACM UIST*, 2017.

[14] Pang, X., Cao, Y., Lau, R.W. and Chan, A.B., "Directing user attention via visual flow on web designs," *ACM TOG*, 2016.

[15] Zhao, N., Cao, Y. and Lau, R.W., "Modeling fonts in context: Font prediction on web designs," in *Proc. Pacific Graphics*, 2018.

[16] SchÃűlkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J. and Williamson, R.C., "Estimating the support of a high-dimensional distribution," *Neural Computation*, 2001.

[17] Rasmussen, C.E., "Gaussian processes in machine learning," *Summer School on Machine Learning. Springer, Berlin, Heidelberg*, 2003.

[18] Xu, P., Fu, H., Igarashi, T. and Tai, C.L., "Global beautification of layouts with interactive ambiguity resolution," in *Proc. ACM UIST*, 2014.

[19] Xu, P., Fu, H., Tai, C.L. and Igarashi, T., "Gaca: Group-aware command-based arrangement of graphic elements," in *Proc. ACM SIGCHI*, 2015.

[20] Liaw, A. and Wiener, M., "Classification and regression by randomforest," *R News*, 2002.

[21] O'Donovan, P., Agarwala, A. and Hertzmann, A., "Color compatibility from large datasets," in *ACM TOG*, 2011.