Automatic Web Page Coloring

Polina Volkova(✉), Soheila Abrishami, and Piyush Kumar

Florida State University, Tallahassee, FL 32306, USA
{volkova,abrisham,piyush}@cs.fsu.edu

Abstract. We present a new tool for automatic recoloring of web pages. Automatic application of different color palettes to web pages is essential for both professional and amateur web designers. However no existing recoloring tools provide full recoloring for web pages. To recolor web page entirely, we replace colors in .css, .html, and .svg files, and recolor images such as background and navigation elements. We create new color theme based on a color guide image provided by user. Evaluation shows a high level of satisfaction with the quality of palettes and results of recoloring. Our tool is available at http://chameleon.cs.fsu.edu/.

1 Introduction

Color is one of the most important components in web page design. The ability to automatically recolor a web page with a given color palette would be very valuable for web designers. Unfortunately the problem of automatic coloring of web pages has not been fully addressed. There exists a plethora of tools that can assist with web page coloring tasks such as palette selection, image recoloring, and image color adjustment. However, these tasks have to be performed separately. An automated web page recoloring tool should combine palette selection, web page and image recoloring. To the best of our knowledge, currently no tool provides this functionality.

The problem of web page recoloring has been partially addressed in research. Works on website recoloring for people with vision deficiencies [1,4] focus specifically on accessibility and cannot be used generically, because they do not ensure color harmony, do not allow users pick the colors, or provide full coloring. We found only one work that addresses a problem similar to ours, by Gu et al. [5]. Gu et al. presented a tool for redefining web page color scheme based on a mood board, using a genetic algorithm to generate assignment of palette colors to .css colors. Their work has several weaknesses, such as simplistic palette extraction method based on K-means, no image recoloring, and not adjusting size of new palette to match the variety of colors on the web page. We found patent applications for a website colorization system[1], and for recoloring images on a web page[2], which confirms that our problem is of practical interest but under researched.

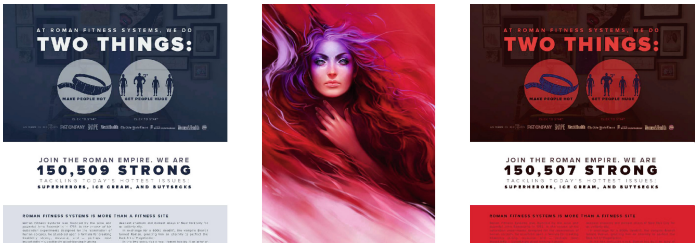
[1] https://www.google.com/patents/US20090031213.
[2] https://www.google.com/patents/US8731289.

G. Bebis et al. (Eds.): ISVC 2016, Part I, LNCS 10072, pp. 91–100, 2016.
DOI: 10.1007/978-3-319-50835-1_9

The goal of this paper was to create a web page recoloring method that fulfills the following objectives:

- *Esthetics*: give users a simple way to specify a harmonious color scheme.
- *Full coloring*: recolor web pages including images, keeping in mind that some images such as photos should not be recolored.
- *Consistency*: preserve color proportions and color variety of the original page.
- *Readability*: recolored web page should have proper contrast.
- *Availability*: the tool should be intuitive to use and publicly available.

To the best of our knowledge, our work is the first website coloring system that achieves the above goals simultaneously. Our contributions include a novel approach to palette extraction that combines human input and automation, and a method for palette expansion that preserves consistency of the color theme and ensures proper contrast. Our system works as follows: users submit their web page and a color guide image via a web interface. Based on the submitted image, a new color palette is created, and assignment for substituting colors in .css, .html, and .svg files is computed. Images such as logos, banners, and background tiles are recolored to reflect the new palette. Figure 1 gives an example of recoloring produced by our system. A survey conducted for evaluation shows that our palette extraction method outperforms other methods, and that recoloring results are rated well by users.



(a) Original web page (b) Color guide image (c) Recolored web page

Fig. 1. Web page recoloring example

Notation. Vectors are denoted by lower-case Roman letters. For a vector p, p_i denotes its ith component. i, j, k, l, m, n are positive integers. We reserve w to denote weight and $d(\cdot)$ to denote Euclidean distance. Scalars are represented by lower-case Greek letters. Upper-case script letters are used for all other objects such as sets, images, and matchings. LAB denotes LAB space and RGB denotes RGB space. $\hat{x}$ denotes a weighted version of x, for instance if x is a color, $x = \{\iota, \alpha, \beta\}$, where ι, α, β are coordinates in LAB space, then $\hat{x} = \{\iota, \alpha, \beta,\ w\}$, where $0 \leq w < 1$ is weight. $\widehat{LAB}$ denotes LAB space with additional weight coordinate, as in $\hat{x} \in \widehat{LAB}$. We will use LAB space for all color manipulation, and K-means algorithm for all clustering tasks.

2 Automatic Web Page Recoloring

This section describes our approach to automatic recoloring of web pages. In Sect. 2.1 we introduce common color operations. Section 2.2 describes related work, and our method for automatic palette selection based on a guide image. Section 2.3 explains the steps of web page recoloring, which are color extraction from the web page, assignment of colors, and additional color generation. In Sect. 2.4 we describe images classification for recolorability, and explain our method for image recoloring. Figure 2 shows the organization of our system. Users submit a query consisting of an image and a web page URL. Query is queued and processed as described in Sects. 2.2, 2.3 and 2.4. Result is rendered and displayed to user.

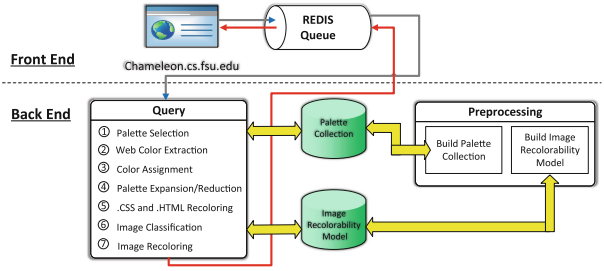


Fig. 2. System modules and data flow

2.1 Color Operations

Summarizing Images. To capture color characteristics of an image for comparison with other images, we summarize each image in a set of its cluster colors. Due to perceptual uniformity of LAB space, clustering works very well for grouping similar colors. Although it was pointed out that clustering is not a proper way to extract a palette from an image [6,9], it suits our purpose since we use it not to obtain the final palette, but rather as a fast and simple way to extract color features of an image.

We downscale an image for faster processing, represent it as an array of pixels in LAB space, and cluster it into k clusters, $k = 5$. Clustering gives us centroid colors $\mathcal{C} = \{c_1, ..., c_k\} \in LAB$, and cluster weights $\mathcal{W}_c = \{w_1, ..., w_k\} \in \mathbb{R}$. As a result, an image is represented by k weighted colors, $\hat{\mathcal{C}} = \{\hat{c_1}, ..., \hat{c_k}\} \in \widehat{LAB}$.

Matching Two Sets of Colors. The purpose of matching is to find best color-to-color assignment for all colors in two sets. It is useful for mapping to a new palette, and for evaluating image similarity.

Let a, b be two colors $\in LAB$. *Perceptual Difference* $d(a, b)$ is a good measure of color similarity due to perceptual uniformity of LAB space. Adding weight, we get $d^w_\lambda(\hat{a}, \hat{b}) = \sqrt{d(a, b)^2 + \lambda(w_a - w_b)^2}$. In our context weight w is color

proportion. We experimented with coefficient λ and found that $\lambda \approx 1$ works best for evaluating image similarity. Intuitively, color proportion is important but it is secondary to color information.

We use *Kuhn-Munkres* algorithm [8] with cost function $d(\cdot)$ to find minimum cost bipartite matching $\mathcal{M}^{\mathcal{A},\mathcal{B}}$ between two sets of colors $\mathcal{A} = \{a_1, ..., a_n\} \in LAB, \mathcal{B} = \{b_1, ..., b_n\} \in LAB$, $\mathcal{M}^{\mathcal{A},\mathcal{B}} = \{\langle a_i, b_j \rangle | a_i \in \mathcal{A}, b_j \in \mathcal{B}\}$. We use $d^w_\lambda(\cdot)$ to find weighted matching $\mathcal{M}^{\hat{\mathcal{A}},\hat{\mathcal{B}}}$ if weights are known.

2.2 Automatic Palette Selection from Image

Our goal is to give users an easy way to select a high-quality palette. Color theory states that color distribution templates can be used to create harmonious color themes [7]. However it has been shown that people do not prefer palettes based strictly on these templates [6]. Palette extraction from images is another popular approach. Some works use histograms [3] and clustering [2,5], but more advanced methods involve human input. For example, a regression model trained on color themes created by people can extract themes from images that closely match human-extracted themes [6]. A color compatibility model learned by linear regression on palette datasets collected online can be used for improving existing palettes and extracting color themes from images [9].

Using an image as a color guide provides an intuitive way to specify a palette. To ensure palette quality, we decided to combine automatic extraction with human expertise, because experiments show that artists create better palettes than extraction algorithms [9]. We will automatically select a palette generated by a professional color designer using a color guide image uploaded by user.

Preprocessing. Our approach requires a palette collection. We assembled a palette source[3] where each record consists of a palette created by a color expert, and an image on which the palette was based (Fig. 3d). We will use the palette for recoloring, and associated image for comparison with the user image. For each record, we retrieve palette colors $\mathcal{P} = \{p_1, ..., p_n\} \in LAB$, $|\mathcal{P}|$ can be different for different records. To compute weights for $p_i \in \mathcal{P}$, we cluster the image using

(a) User image $\mathcal{U}$ (b) Summary $\mathcal{G}$ (c) Palette collection $\mathcal{E}$ (d) Match from palette collection $\mathcal{I}$

Fig. 3. Automatic palette selection example

[3] 4561 palettes obtained from color blog Design Seeds, http://design-seeds.com/.

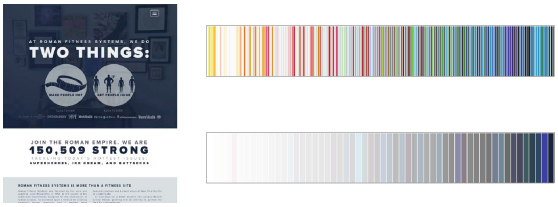
$k = |\mathcal{P}|$ into $\mathcal{C} = \{c_1, ..., c_k\} \in LAB$ and $\mathcal{W}_\mathcal{C} = \{w_1, ..., w_k\} \in \mathbb{R}$. Next we compute a matching $M^{\mathcal{P},\mathcal{C}} = \{\langle p_i, c_j\rangle | p_i \in \mathcal{P}, c_j \in \mathcal{C}\}$, and assign weights w_j to palette colors p_i, $\mathcal{W}_\mathcal{P} = \{w_j | \langle p_i, c_j\rangle \in M^{\mathcal{P},\mathcal{C}}\}$. Now we have a weighted palette $\hat{\mathcal{P}} = \{\hat{p}_1, ..., \hat{p}_n\} \in \widehat{LAB}$. Finally, we re-cluster the image with $k = 5$ to get summary $\hat{\mathcal{R}} = \{\hat{r}_1, ..., \hat{r}_5\} \in \widehat{LAB}$. $\hat{\mathcal{R}}$ and $\hat{\mathcal{P}}$ for all records comprise our palette collection $\mathcal{E} = \{\hat{\mathcal{R}}_i, \hat{\mathcal{P}}_i\}$. This data are stored in binary files and used in palette selection: $\hat{\mathcal{R}}_i$ for matching collection image to the user image, and $\hat{\mathcal{P}}_i$ as palette for recoloring.

Automatic Palette Selection. To automatically select a palette $\mathcal{T}$ that closely matches colors of a *guide image* $\mathcal{U}$, we find an image $\mathcal{I}$ in our collection that is most similar to $\mathcal{U}$, and retrieve its palette (Fig. 3). First we cluster $\mathcal{U}$ with $k = 5$ into $\hat{\mathcal{G}} = \{\hat{g}_1, ..., \hat{g}_5\} \in \widehat{LAB}$. Then we iterate through palette collection $\mathcal{E}$ to find a record $\{\hat{\mathcal{R}}_{i^*}, \hat{\mathcal{P}}_{i^*}\}$ such that bipartite matching cost between $\hat{\mathcal{G}}$ and $\hat{\mathcal{R}}^*_i$ is minimum. Finally, we retrieve the palette $\hat{\mathcal{T}} = \hat{\mathcal{P}}_{i^*}$ to be used for recoloring the web page. We will refer to $\hat{\mathcal{T}}$ as *target palette*.

2.3 Automatic Web Page Coloring

To recolor webpage with palette colors $\mathcal{T}$, we need to extract all colors from web page, expand or shrink $\mathcal{T}$ to match the number of web page colors, map target palette colors to web page colors, and replace colors in .css, .html, and .svg files.

Extracting Colors from Web Page. We implemented our own color extraction for the following reasons. First, we encountered an issue of unused colors in the .css files. Web designers often reuse same .css files for multiple projects and do not remove unused styles. If we simply take all colors from .css files, we get many colors that do not actually appear on a web page (Fig. 4). This negatively affects speed and quality of recoloring. Cleaning up .css files turned out unreliable[4] or hard to automate[5]. Color extraction from website did not remove



(a) Web page screenshot (b) All colors from .css and .html (top), colors used on web page (bottom)

Fig. 4. Color extraction from a web page.

[4] https://github.com/peterbe/mincss.
[5] https://chrome.google.com/webstore/detail/css-remove-and-combine/cdfmaaeapjmacolkojefhfollmphonoh?hl=en-GB.

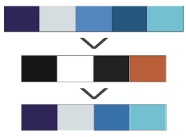
unused colors[6]. In addition, we needed to calculate color proportions, which is not provided by existing tools.

We solved the problem by discarding colors that do not appear in the screenshot of the web page. First, we find all hexadecimal, RGB, RGBA colors in .html, .css, and .svg files, and convert them to RGB format. Let's call this set $\mathcal{H} = \{h_1, ..., h_n\} \in RGB$. Then we take screenshot of the webpage without images. We extract all distinct colors from the screenshot and calculate their weights, getting $\mathcal{R} = \{r_1, ..., r_m\} \in RGB$, $\mathcal{W}_\mathcal{R} = \{w_1, ..., w_m\} \in \mathbb{R}$.

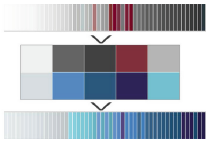
Next, we find mapping between colors in $\mathcal{H}$ and $\mathcal{R}$ to detect unused colors. Colors in $\mathcal{R}$ may slightly differ from corresponding colors in $\mathcal{H}$ due to image compression or use of gradients. We say that $\langle r_i, h_j \rangle$ is a match if $d(r_i, h_j) < \zeta$, where $\zeta = 20$ is a threshold derived experimentally. There can be multiple matches r_i to the same h_{j^*}. Let $\mathcal{X}_j$ be the set of indexes i for r_i matched to the same h_{j^*}, then $w'_{j^*} = \sum_{l \in \mathcal{X}_{j^*}} w_l$. Now we have used colors $\mathcal{H}' = \{h_1, ..., h_l | h_j \in \langle r_i, h_j \rangle\} \subset \mathcal{H}$ and $\mathcal{W}_{\mathcal{H}'} = \{w'_1, ..., w'_l\} \in \mathbb{R}$. We convert all $h_j \in \mathcal{H}'$ to LAB, getting the set of weighted *web page colors* $\hat{\mathcal{S}} = \{\hat{s}_1, ..., \hat{s}_l\} \in \widehat{LAB}$.

Assigning Palette Colors to Web Elements. To replace colors in files, we need to compute a mapping between old and new colors. As input to this step, we have two sets of colors: target palette $\hat{\mathcal{T}}$ and web page colors $\hat{\mathcal{S}}$. Most likely, $|\hat{\mathcal{T}}| \neq |\hat{\mathcal{S}}|$: $\hat{\mathcal{T}}$ can be larger or smaller than $\hat{\mathcal{S}}$. We need to expand or reduce palette $\hat{\mathcal{T}}$ into a new palette $\hat{\mathcal{T}}'$ of size $|\hat{\mathcal{S}}|$.

Let $\mathcal{Q}$ denote the smaller palette, and $\mathcal{G}$ denote the larger palette. $\hat{\mathcal{Q}} = \hat{\mathcal{T}}, \hat{\mathcal{G}} = \hat{\mathcal{S}}$ if $|\hat{\mathcal{T}}| < |\hat{\mathcal{S}}|$, otherwise $\hat{\mathcal{Q}} = \hat{\mathcal{S}}, \hat{\mathcal{G}} = \hat{\mathcal{T}}$. Let $|\mathcal{Q}| = n$. First, we cluster $\mathcal{G}$ with $k = n$ to get $\mathcal{G}' = \{g'_1, ..., g'_k\} \in LAB$, $\mathcal{W}_{\mathcal{G}'} = \{w'_1, ..., w'_k\} \in \mathbb{R}$. We replace centroids g'_i with actual palette colors $g_j \in \hat{\mathcal{G}}$ such that $j = \arg\min_{i,j} d(g'_i, g_j)$ and keep w'_i. That gives us $\hat{\mathcal{G}}' = \{\hat{g}_1, ..., \hat{g}_k\} \in \widehat{LAB}$. Now we can find a bipartite matching between two sets of colors of same size n: $\mathcal{M}^{\hat{\mathcal{G}}', \hat{\mathcal{Q}}} = \{\langle \hat{g}_j, \hat{q}_i \rangle \ | \hat{g}_j \in \hat{\mathcal{G}}', \hat{q}_i \in \hat{\mathcal{Q}}\}$. If $\hat{\mathcal{Q}} = \hat{\mathcal{S}}$, we can use $\mathcal{T}' = \mathcal{G}'$ for recoloring (Fig. 5(a)). Otherwise, we need to expand palette $\mathcal{Q}$ (Fig. 5(b)).



(a) Target palette $\mathcal{T}$ ($\mathcal{G}$) (top), web palette $\mathcal{S}$ ($\mathcal{Q}$) (center), reduced palette $\mathcal{T}'$ ($\mathcal{G}'$) (bottom)

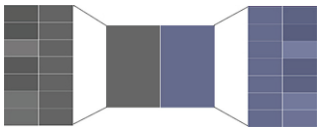


(b) Web palette $\mathcal{S}$ ($\mathcal{G}$)(left), centroids $\mathcal{G}'$ matched to target palette $\mathcal{T}$ ($\mathcal{Q}$) (center), extended palette $\mathcal{T}'$ (right)

Fig. 5. Palette reduction (a) and extension (b)

[6] http://www.colorcombos.com/.

Palette expansion. We need to create suitable replacement colors for all colors on the web page, staying true to palette $\mathcal{T}$. It is critical to set the luminance of new colors correctly for the recolored web page to be legible. We could achieve same contrast as on the original web page by copying luminance of old colors to new colors, $\iota_t = \iota_s, \forall t \in \mathcal{T}', \forall s \in \mathcal{S}$. However this changes the appearance of colors, which may result in unpleasant palette, or a palette that does not represent user image well. A better solution is to preserve the original palette colors, and to shift ι for additional colors. This preserves contrast because new colors will be distributed similarly to the original colors, with respect to luminance.



(a) Cluster of website colors $\{g_1, ..., g_k\}$ and centroid color g_{i^*} (b) g_{i^*}, q_{j^*} (c) q_{j^*} and additional colors generated from q_{j^*}

Fig. 6. Creating additional colors

Figure 6 demonstrates the process of creating additional colors on the example of one color. One of web colors $g_{i^*} \in \mathcal{G}'$ represents a cluster of web colors $\{g_1, ..., g_k\} \in \mathcal{G}$ (Fig. 6a). g_{i^*} is matched to a target palette color $q_{j^*} \in \hat{\mathcal{Q}}$ (Fig. 6b), $\langle g_{i^*}, q_{j^*} \rangle$ is added to final mapping $\mathcal{F}^{\mathcal{S},\mathcal{T}}$. We create new shades from q_{j^*} for the remaining web colors $g_i \in \{g_1, ..., g_k\}$ (Fig. 6c). We need to create $|\{g_1, ..., g_k\}| = m$ additional colors. Replacement color q_i for g_i starts with $q_i = q_{j^*}$, but we set $\iota_{q_i} = \iota_{q_{j^*}} + (\iota_{g_{i^*}} - \iota_{g_i})$ for all $g_i \in \{g_1, ..., g_k\}$, where ι_x is the luminance of color $x \in LAB$. We check that q_i is within the boundaries of LAB space and add $\langle g_i, q_i \rangle$ to $\mathcal{F}^{\mathcal{S},\mathcal{T}}$. Once we compute a replacement for each web page color, we convert colors in $\mathcal{F}^{\mathcal{S},\mathcal{T}}$ back to hexadecimal/RGB/RGBA format and substitute corresponding colors in .html, .css, and .svg files.

2.4 Image Classification and Recoloring

One of our objectives was to recolor images on a web page. However, not all images should be recolored. It makes sense to recolor images that contain few colors, e.g. text, logos, background tiles. Art and photographs are examples of images that should not be recolored (Fig. 7). To distinguish recolorable images, we use a decision tree trained on a sample set of images classified by hand. We pass only recolorable images to the image recoloring module.

Image Recoloring is a well-covered topic. Chang et al. [2] presented a tool for image color adjustment by editing palette extracted from an image. Reinhard et al. [10] gave a method for color correction that uses statistical analysis to transfer color characteristics from one image to another. Xia [11] extended the work of [10] by including saliency map into the color transfer formula.

(a) Original webpage (b) Recolorable images (top) and non-recolorable images (bottom) (c) Recolored result

Fig. 7. Image classification and recoloring

We borrowed ideas from Chang et al. [2] because their method is very suitable for our task of recoloring images with a set palette. We used a simplified version of their algorithm. The input is an image, and the target palette $\mathcal{T}$. We extract the initial palette $\mathcal{Y}$ of the image using K-means with $k = |\mathcal{T}|$. Our setup is different from the original algorithm by Chang et al., where it is known which color is modified in each step of the recoloring. We start with two palettes $\mathcal{T}$ and $\mathcal{Y}$, unknown order of recoloring, and unknown relation between colors of the palettes. We first perform matching to find the pairing of colors in the initial and target palettes $M^{\mathcal{T},\mathcal{Y}}$. Then, a sequential color transformation algorithm is used to recolor the image.

The transformation function $f(x)$ transforms a color $x \in LAB$ in the original image to a color $x' \in LAB$ in the recolored image. If $|\mathcal{Y}| = |\mathcal{T}| = 1$ we need one transformation function $x' = f(x)$ as $f(x) = (y - t) + x$. For $|\mathcal{T}| > 1$ we need k transfer functions $f_i(x), i = 1 \ldots k$. In order to smooth the effect of the individual transfer functions $f_i(x)$ at y_i, the functions are blended with different weights. For weight calculation details refer to Section 3.5 of [2].

We treat background separately. The old background color $a \in \mathcal{S}$ is set to the color that appears most on the web page. The new background color $b \in \mathcal{T}'$ was calculated for a per Sect. 2.3. Recoloring procedure takes a, b. If a certain percentage of pixels in the edges of the image passed for recoloring is of color a, we say that the image has background, and we replace all pixels of color $x \simeq a$ with b (we allow for some color variance due to image compression). All other colors are replaced as described above.

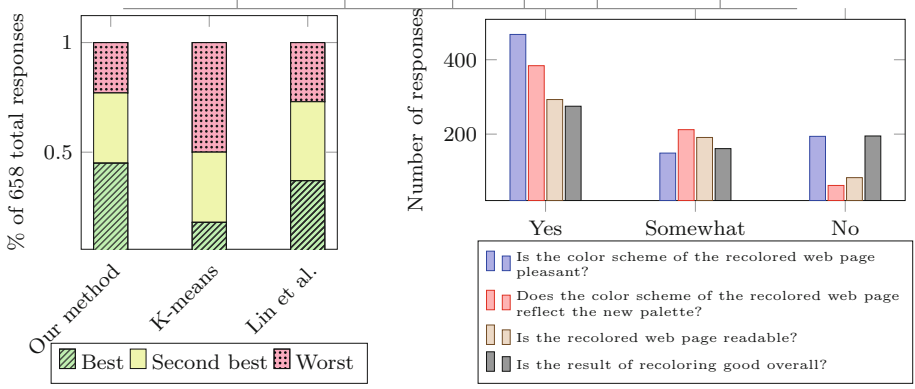
3 Experiments

Image Classification. We assembled a sample set of images from Fortune 500 websites, labeled them by hand, and extracted the following features: number of colors, aspect ratio, size, grayscale or color, and histogram. Half of the sample set was used for training and half for testing. We experimented with multiple classifiers from Python package *sklearn*[7]. Classification works very well, all classifiers get above 90% of cases right (Table 1).

[7] http://scikit-learn.org/stable/.

Table 1. Classification precision on a test set of 1063 images (510 recolorable and 553 non-recolorable).

Classifier	Decision tree	Random forest	SVM	KNN	Logistic regression	Naive Bayes
Precision	1	0.99	0.96	0.93	0.93	0.91



(a) Palette rating (1-best, 3-worst) (b) Web page recoloring evaluation

Fig. 8. Survey results

Palette and Recoloring Evaluation. We conducted two surveys to confirm that our palette extraction method produces good palettes, and to evaluate the overall result of web page recoloring. For both surveys we collected responses from same 86 participants in the United States, who were not compensated. In order to evaluate the quality of palettes, we presented ten sets of three palettes, generated as follows. Ten images were picked at random from an art collection[8]. For each image, we extracted a palette using our method (Sect. 2.2), method by Lin et al. [6], and K-means. Resulting palettes were same size and rendered exactly same way. We asked to arrange palettes in order from best to worst. Initial ordering of palettes in the survey was randomized. Despite the fact that color scheme ratings are very subjective, the results are clear (Fig. 8(a)). Our palettes received more best ratings and fewer worst ratings than other palettes. We performed a Wilcoxon Rank-Sum test with Bonferroni correction on the pairs of ratings, showing that the results were significant ($p < .0001$ for our vs. K-means, $p < .005$ for our vs. Lin et al.). This is an encouraging validation of our palette extraction method.

To evaluate the quality of web page recoloring, we presented the participants with ten renderings of the original web page next to the web page recolored using a randomly chosen picture from an art collection (see footnote 8) and the new palette. We carefully selected representative websites for this experiment, aiming to cover typical web page types. We picked an information page, a forum-style page, and a blog; each with a light, medium, and dark color scheme, similar to [4]. The majority of respondents rated our recoloring positively (Figure 8(b)).

[8] Private digital art collection of 382 images.

For each question we found statistical significance at $p < .001$ using t-test on proportion of positive answers.

4 Conclusion and Future Work

We presented a new method for automatic recoloring of web pages. We achieved the objectives of esthetics, full coloring, consistency, and readability, and built a publicly available web based tool. Evaluation shows that palette extraction and overall web page recoloring results are good. We believe that our approach produces better results than closest comparable work by Gu et al. [5]. Future work will focus on making our tool interactive. We want users to be able to modify palette, manually map colors, and specify which images to recolor.

References

1. Aupetit, S., Mereuta, A., Monmarché, N., Slimane, M.: Comparison of interruptible meta heuristics for automatic recoloring of web pages with an accessibility goal. In: AMSE - Advance in modelling, Handicap, 2012 revised selected papers. C Automatic Control (Theory and Applications), vol. 73, pp. 11–21 (2013)
2. Chang, H., Fried, O., Liu, Y., DiVerdi, S., Finkelstein, A.: Palette-based photo recoloring. ACM Trans. Graph. **34**(4), 139:1–139:11 (2015)
3. Delon, J., Desolneux, A., Lisani, J.L., Petro, A.B.: Automatic color palette. In: IEEE International Conference on Image Processing 2005, vol. 2, p. II-706-9, September 2005
4. Flatla, D.R., Reinecke, K., Gutwin, C., Gajos, K.Z.: SPRWeb: preserving subjective responses to website colour schemes through automatic recolouring. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2013, NY, USA, pp. 2069–2078. ACM, New York (2013)
5. Gu, Z., Wu, Z., Yu, J., Lou, J.: A color schemer for webpage design using interactive mood board. In: Kurosu, M. (ed.) HCI 2013. LNCS, vol. 8004, pp. 555–564. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39232-0_60
6. Lin, S., Hanrahan, P.: Modeling how people extract color themes from images. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2013, NY, USA, pp. 3101–3110. ACM, New York (2013)
7. Matsuda, Y.: Color Design. Asakura Shoten, Tokyo (1995)
8. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**(1), 32–38 (1957)
9. O'Donovan, P., Agarwala, A., Hertzmann, A.: Color compatibility from large datasets. ACM Trans. Graph. **30**(4), 63:1–63:12 (2011)
10. Reinhard, E., Adhikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Comput. Graph. Appl. **21**(5), 34–41 (2001)
11. Xia, J.: Saliency-guided color transfer between images. In: Bebis, G., et al. (eds.) ISVC 2013. LNCS, vol. 8033, pp. 468–475. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41914-0_46