APPROVAL SHEET

Title of Thesis: MY-THESIS-TITLE

Name of Candidate: MY-FULL-NAME
DEGREE-NAME,
GRADUATION-YEAR

Thesis and Abstract Approved: _____

MY-ADVISORS-NAME
MY-ADVISORS-TITLE
Department of Computer Science and
Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: MY-FULL-NAME.

Permanent Address: MY-FULL-ADDRESS.

Degree and date to be conferred: DEGREE-NAME, GRADUATION-MONTH GRADUATION-YEAR.

Date of Birth: MY-BIRTHDATE.

Place of Birth: MY-PLACE-OF-BIRTH.

Secondary Education: MY-HIGH-SCHOOL, MY-HIGH-SCHOOLS-CITY, MY-HIGH-SCHOOLS-STATE.

Collegiate institutions attended:

University of Maryland Baltimore County, DEGREE-NAME MY-MAJOR, GRADUATION-YEAR.
MY-OTHER-DEGREES.

Major: MY-MAJOR.

Minor: MY-MINOR.

Professional publications:

FULL-CITATION-INFORMATION.
FULL-CITATION-INFORMATION.

Professional positions held:

EMPLOYMENT-INFO. (START-DATE – END-DATE).
EMPLOYMENT-INFO. (START-DATE – END-DATE).

ABSTRACT

Title of Thesis: Recoloring Web Pages For Color Vision Deficiency Users.

Vikas Bansal, Masters in Science, 2014

Thesis directed by: Dr. Lina Zhou, Associate Professor
Department of Information Systems

Dr. Tim Finin, Professor
Department of Computer Science and
Electrical Engineering

Colors are an important part of our life. They are commonly used to represent important information, specially, categories. Ability to differentiate between colors is important in performing routine tasks such as reading content, following traffic lights etc. This ability varies from person to person. Many people experience difficulty in reading content on web pages due to this variation. These difficulties result from the inability of individuals to sufficiently differentiate between colors. This condition of an individual is called Color Vision Deficiency (CVD). More than four percent of current population suffer from some kind of CVD, significantly affecting their web experience.

To improve the web experience of CVD users, we have presented an algorithm which can be used to recolor web pages such that the recolored web pages do not pose any difficulty to a CVD user. Replaced colors are chosen from a fixed set of color called Dichromacy Trichromacy Equivalency Plane (DTEP) set. While recoloring we also preserve the naturalness and contrast among foreground and background colors in different sections of the web page. A quantitative comparison with the existing tool SPRWeb[] shows that our algorithm performs better in preserving contrast among different sections of the web pages and doesnt differ much in preserving naturalness.

An additional step in to algorithm was added to induce the contrast in pairs according to the W3C guidelines. Quantitative experimentation of modified algorithm shows that

contrast ratio in each replacement pair is more than 4.5 as required for readability.

TITLE-OF-THESIS

by

MY-FULL-NAME

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
DEGREE-NAME
CURRENT-YEAR

INSERT-DEDICATION-HERE

ACKNOWLEDGMENTS

Write your acknowledgment here.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Website colors specially which are used in foreground text and background are responsible to provide legibility in the content. In addition to providing legibility, they also influence the subjective responses from the users. Depending on the color, a website may seem heavy or light, high or low in temperature and may tell about busyness.

While one solution to the problem of illegibility could be to educate web designers so that they only use a set of colors which are comfortable to both CVD and normal users. [we have developed one such technique - webpages colors downloaded]. Which is a kind of a prevention step. One such color combination could be to develop all the webpages using only black and white. But that itself defeats the purpose of existence of colors. Most of the web developers today, even being aware of the existence of CVD users, develop web pages considering only normal web users leading to confusion and frustration among CVD users.

Another solution is more of a rectifying step. In this, we recolor existing bad web pages to suit the need of both normal and CVD users.

1.1 SECTION-TITLE

1.2 SECTION-TITLE

1.2.1 SECTION-TITLE

1.2.2 SECTION-TITLE

BACKGROUND AND RELATED WORK

In this section we will learn about CVD and the existing related work to deal with it. Which also formed the basis of this thesis.

2.1   Color Vision Deficiency (C.V.D.)

Color vision deficiency, is the inability or decreased ability to see color, or perceive color differences, under normal lighting conditions. Color blindness affects a significant percentage of the population. There is no actual blindness but there is a deficiency of color vision. The most usual cause is a fault in the development of one or more sets of retinal cones that perceive color in light and transmit that information to the optic nerve. This type of color blindness is usually a sex-linked condition. The genes that produce photo-pigments are carried on the X chromosome; if some of these genes are missing or damaged, color blindness will be expressed in males with a higher probability than in females because males only have one X chromosome (in females, a functional gene on only one of the two X chromosomes is sufficient to yield the needed photo-pigments).

Color blindness can also be produced by physical or chemical damage to the eye, the optic nerve, or parts of the brain. For example, people with achromatopsia suffer from a completely different disorder, but are nevertheless unable to see colors.

By cause CVD can be classified in to three types:

- Acquired

- Inherited: Inherited can further be classified in to three types:

    - Monochromacy: Also known as "total color blindness", is the lack of ability to distinguish colors (and thus the person views everything as if it were on a black and white television); caused by cone defect or absence. Monochromacy occurs when two or all three of the cone pigments are missing and color and lightness vision is reduced to one dimension. We are not dealing with this type in our current version of the algorithm.

    - Dichromacy: Dichromacy is a moderately severe color vision defect in which one of the three basic color mechanisms is absent or not functioning. Our algorithm can recolor web pages to suit all the Dichromats.

        * Protanopia (1% of the males): Protanopia is a severe type of color vision deficiency caused by the complete absence of red retinal photoreceptors. It is a form of dichromatism in which the subject can only perceive light wavelengths from 400 to 650nm, instead of the usual 700nm. Pure reds cannot be seen, instead appearing black; purple colors cannot be distinguished from blues; more orange-tinted reds may appear as very dim yellows, and all orange-yellow-green shades of too long a wavelength to stimulate the blue receptors appear as a similar yellow hue. It is hereditary and sex-linked.

        * Deuteranopia (1% of the males): Deuteranopia is a color vision deficiency in which the green retinal photoreceptors are absent, moderately affecting redgreen hue discrimination. It is a form of dichromatism in which

there are only two cone pigments present. It is likewise hereditary and sex-linked.

* Tritanopia (Less than 1% of males and females): Tritanopia is a very rare color vision disturbance in which there are only two cone pigments present and a total absence of blue retinal receptors. Blues appear greenish, yellows and oranges appear pinkish, and purple colors appear deep red. It is related to Chromosome "7".

– Anomalous trichromacy

2.2 Diagnosis

The Ishihara color test, which consists of a series of pictures of colored spots, is the test most often used to diagnose redgreen color deficiencies. A figure (usually one or more Arabic digits) is embedded in the picture as a number of spots in a slightly different color, and can be seen with normal color vision, but not with a particular color defect. The full set of tests has a variety of figure/background color combinations, and enable diagnosis of which particular visual defect is present.

To test the type of color blindness users have, we did the Ishihara color test on each of them during our user study of the algorithm.

2.3 Related Work

Quite a work has been done to both simulate and correct CVD. Ichikawa et al. [16sprweb] were the first publishers of a website recolouring tool that improves accessibility for people with CVD. Iaccarino et al. [15sprweb] proposed the use of edge services to recolour website images in transit to the user. ColorBlindnessSimulateCorrect by Seewald Solutions is one of the examples of an application developed to both sim-
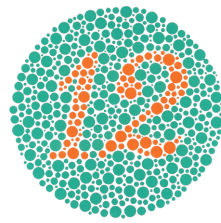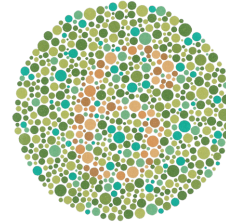
(a) Plate No. 1 (12) (b) Plate No. 13 (6)

FIG. 2.1: Ishihara Test [Wikipedia]

ulate and correct CVD effect. It uses a linear transformation on rgb input. Daltonize developed by www.daltonize.org can be used to recolor web pages. They first convert RGB values of a color to LMS values and then compensate for color blindness by shifting wavelengths away from the portion of the spectrum invisible to the dichromat, towards the visible portion. Eyepilot [http://www.colorhelper.com/] enables computer users with red-green color blindness to decipher color-coded maps and graphs by letting the user place a floating window on top of any picture to distinguish among the various color fields. The GNOME[http://en.wikipedia.org/wiki/GNOME] desktop environment provides users with the option to switch a color filter on and off choosing from a set of possible color transformations that displace the colors in order to disambiguate them. While all these tools have proved to be effective in inducing differentiability in the content and thus providing legibility, but none of them have been able to retain the naturalness1, differentiability2 of the original web page.

One such tool which preserves naturalness and improves color differentiability is kuhns tool. One improvement on naturalness preservation was done by David et al. in SPRWeb[]. Which is also the basis of this thesis. SPRWeb performs better in preserving naturalness and differentiability than Kuhns tool. But it fails to make sure than all the fore-

ground background color pairs have the minimum required contrast ratio threshold of 4.5 as per W3C guidelines.

2.4 Definitions

- **O**: Original set of colors as parsed from CSS files associated with the web page.

- **R**: Replaced set of colors as computed by the recoloring algorithm. And obviously, size of **R** is always equal to size of **O**.

- **P**: Number of color *pairs* in a web page. A *pair* of colors from a web page is of the form *(fg,bg)*, where *fg* is a foreground text color and *bg* is a background color. There can be many such *(fg,bg)* color pairs.

- **Perceptual naturalness[SPRweb]:** It is a measure of how close is set **R** to set **O**. Ideally, if our algorithm is able to find a set such that **R** = **O** then the *perceptual naturalness* will be maximum and the *cost of naturalness* would be 0. *Cost of naturalness* quantifies naturalness factor. As defined in SPRWeb[], *cost of naturalness* can be quantified using the following expression:

$$pn = \frac{1}{N} * \sum_{i=1}^{N} \Delta_{L_{ab}}(O_i, R_i)$$

N Size of **O**.

$\Delta_{L_{ab}}$ Euclidean distance in Lab color space.

Lab color space is explained in the next section. A *low* value of *pn* is highly desirable. A *low* value of *pn* as compared to a high value, essentially tells us that the

recolored web page having *low* value is closer to original web page in terms of colors. Optimizing the value of *pn* while computing the recolored set **R** makes sure that the recolored version does not have abrupt colors as compared to the original version.

- **Perceptual differentiability[SPRweb]:** This factor maintains the differentiability among colors belonging to a pair in the original web page. Suppose the original web page has only one color pair, i.e size of **P** is 1. And the pair is *(Red,White)*, then the recolored pair should have colors such that the difference in original pair should be maintained to its best possible extent. This factor can be quantified using the following equation:

$$pd = \frac{1}{size(\mathbf{P})} * \sum_{(X,Y) \in P_O, P_R} |\Delta_{L_{ab}}(X_f, X_g) - \Delta_{L_{ab}}(Y_f, Y_g)|$$

P_O	Set of pairs in original web page.
P_R	Set of pairs in recolored web page.
X	A pair of color in original web page thus a member of P_O.
Y	Replacement of color pair X thus a member of P_R.
$X_f and Y_f$	*fg* in pair X and Y respectively.
$X_g and Y_g$	*bg* in pair X and Y respectively.

- **Subjective naturalness[SPRweb]:** We need this factor in optimizing function to keep the subjective responses of users as is. For example, using this in optimization we can keep *warm* colors *warm* or *heavy* colors *heavy*. We are using subjective response model developed by [Ou et all] to compute the subjectivity factor. To preserve subjectivity of colors in original web page, we use *subjective response space*.

We discuss *subjective response space* in detail in future sections. Subjective naturalness cost can be quantified as follows:

$$srn = \frac{1}{N} * \sum_{i=1}^{N} \emptyset_u(O_i, R_i)$$

N Size of **O**.

$\emptyset_u$ Euclidean distance in subjective response space.

- **Subjective differentiability[SPRweb]:** Similar to perceptual differentiability, subjective differentiability should be maintained among pairs of colors parsed from original web page. We again use *subjective response space* to quantify this factor as follows:

$$spd = \frac{1}{size(\mathbf{P})} * \sum_{(X,Y) \in P_O, P_R} |\emptyset_u(X_f, X_g) - \emptyset_u(Y_f, Y_g)|$$

P_O Set of pairs in original web page.

P_R Set of pairs in recolored web page.

X A pair of color in original web page thus a member of P_O.

Y Replacement of color pair X thus a member of P_R.

$X_f and Y_f$ *fg* in pair X and Y respectively.

$X_g and Y_g$ *bg* in pair X and Y respectively.

- **Cost function:** $= W_{pn} * pn + W_{pd} * pd + W_{spn} * spn + W_{spd} * spd$

Where $W'_{ii}s$ are weights corresponding to the factors. We can choose any value for these factors depending upon the requirement. We keep a value 1 for all of them to treat all factors equally.

2.5   Color spaces

- **Lab color space:**   A Lab color space is a color-opponent space with dimension L for lightness and a and b for the color-opponent dimensions, based on nonlinearly compressed CIE XYZ color space coordinates. The three coordinates of CIELAB represent the lightness of the color ($L^* = 0$ yields black and $L^* = 100$ indicates diffuse white; specular white may be higher), its position between red/magenta and green (a^*, negative values indicate green while positive values indicate magenta) and its position between yellow and blue (b^*, negative values indicate blue and positive values indicate yellow)[wikipedia]. The range of the three coordinates is as follows:

$$
\begin{aligned}
L^* &\in \quad [0,100] \\
a^* &\in \quad [-127,128] \\
b^* &\in \quad [-127,128]
\end{aligned}
$$

Why did we use Lab color space?

- Lab color is designed to approximate human vision. It aspires to perceptual uniformity, and its L^* component closely matches human perception of lightness.

- The $L^*a^*b^*$ color space includes all perceivable colors which means that its

gamut exceeds those of the RGB and CMYK color models (for example, ProPhoto RGB includes about 90% all perceivable colors).

– One of the most important attributes of the $L^*a^*b^*$-model is device independence. This means that the colors are defined independent of their nature of creation or the device they are displayed on. The $L^*a^*b^*$ color space is used e.g. in Adobe Photoshop when graphics for print have to be converted from RGB to CMYK, as the $L^*a^*b^*$ gamut includes both the RGB and CMYK gamut.

- **Subjective response space:** *Subjective response space* consists of three dimensions which can be formulated as follows using [Ou et all] model:

Activity(Active, Passive):

$$AP = -2.1 + 0.06\sqrt{(L^* - 50)^2 + (a^* - 3)^2 + (\frac{b^* - 17}{1.4})^2}$$

Temperature(Warm, Cool):

$$WC = -0.5 + 0.02(C^*)^{1.07}cos(H^* - 50°)$$

Weight(Heavy, Light):

$$HL = -1.8 + 0.04(100 - L^*) + 0.45cos(H^* - 100°)$$

Chroma:

$$C^* = \sqrt{a^{*2} + b^{*2}}$$

Hue:

$$H^* = arctan(b^*, a^*)$$

$L^*, a^* \, and \; b^*$ Dimensions of Lab color space

Chapter 3

SYSTEM ARCHITECTURE

In this chapter, we are going to present the approaches that we took to solve the problem in hand and then we will finally present our algorithm to recolor web pages.

3.1 Initial thoughts and approaches

We started of with a very intuitive idea. We first figured out that which colors are conflicting with rest of the color schema of the web page. After we figure that out, we replace those colors such that there are no more conflicts. In this idea, we need an initial knowledge of what two colors would be conflicting. And in addition to this, we also need to know what would be the safe colors which we can put in place of conflicting colors. To get an idea of this algorithm lets see the following example:

For simplicity, lets assume that we are recoloring web pages which are developed using only the following set of colors. Lets call this set as **U**(Universal Set):

1. Black(#000000)

2. Blue(#003366)

3. Orange(#FF9900)

4. Yellow(#FFCC00)

5. Red(#FF0000)

6. Green(#00FF00)

7. White(#FFFFFF)

To recolor web pages for a particular type of CVD, lets say Protanopia, we need a kind of table like this:

	Black	Blue	Orange	Yellow	Red	Green	White
Black	✗	✓	✓	✓	✓	✓	✓
Blue	✓	✗	✓	✓	✓	✓	✓
Orange	✓	✓	✗	✓	✗	✗	✓
Yellow	✓	✓	✓	✗	✓	✓	✗
Red	✓	✓	✗	✓	✗	✗	✓
Green	✓	✓	✗	✓	✗	✗	✓
White	✓	✓	✓	✗	✓	✓	✗

FIG. 3.1: Sample Look Up Table (LUT)

This look up table can help us determine the *conflict* of colors in a webpage. A *conflict* is defined as a situation when two or more colors are seen as similar color by a CVD person, leading to very low differentiability amongst the colors. And since the differentiability is low, one of them should be replaced with a color that can relatively increase the differentiability.

As we can see in Fig 3.2(a), two colors having hue properties as Orange and Green map to colors of similar hues, leading to conflict for a CVD person. LUT in Fig 3.1 lists all such conflicts for colors in **U** which can be experimentally determined.

Safe colors

(a) As seen by a normal person

(b) As seen by CVD person (Protanopia)

FIG. 3.2: Example of conflict

Safe colors are the colors which can replace the *conflicting* colors, removing the existing conflict and causing no more additional conflicts. For the given set **U**, some of the safe colors can be:

1. ColorA #CC00FF

2. ColorB #669999

3. ColorC #003300

So our table in Fig 2.1 would look something like this now:

	Black	Blue	Orange	Yellow	Red	Green	White	ColorA	ColorB	ColorC
Black	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Blue	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Orange	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓
Yellow	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓
Red	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓
Green	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓
White	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓
ColorA	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
ColorB	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
ColorC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗

FIG. 3.3: Sample Look Up Table (LUT)

Recoloring algorithm using LUT and safe colors

An algorithm such as Algorithm 1 can be implemented to recolor web pages.

Another way of recoloring web pages could be to recolor them entirely using a predefined set of non-conflicting colors. Information on how to obtain these safe sets is provided in next section. We can perform Algorithm 2 utilizing these existing sets to recolor web pages:

3.1.1 How to obtain safe colors?

Most accurate way of obtaining these safe colors would be to show a set of colors to CVD users and then classify the colors as *safe* or *conflicting*. Although degree of CVD may slightly vary from user to user but we assume in our study that a safe set developed for some of the CVD users is applicable to most of them.

A similar study was done by [5]. They obtained a color palette as seen by a CVD users.

| (a) As seen by a normal person | (b) As seen by CVD person (Protanopia) |

FIG. 3.4: Color Palette

As we can observe, a CVD person can only see a subset of all possible colors. In other words, an entire set of colors(Fig 3.4(a)) is mapped to only a few different colors(Fig 3.4(b)). It is very probable that if two random colors are put into a single web page, they map to two colors that have very little differentiability between them. If they are used in

background and text formatting, a color blind person won't be able to read the text, which might be a crucial aspect from a website designers point of view. To resolve this, we need a particular set of colors, which when used together, map to colors having significant differentiability for a CVD user.

Lets say there are two colors in a web page *C1* and *C2* as seen by a normal person. They will be seen as *C1'* and *C2'* by a CVD user. *C1'* and *C2'* should be differentiable enough for a CVD person to read the content on the web page. Idea is to replace *C1* and *C2* with *D1* and *D2* such that *D1'* and *D2'*, as seen by a CVD user, are differentiable enough.

Limitations of this approach

- We will have to manually generate a finite number of sets from the palette available at [5], so that we can choose a particular set while recoloring.

- Still, there can be a case in which none of the colors in *ColorHex* belongs to any of the sets, which restricts the scope of this approach.

- This approach leads to bad *perceptual naturalness*, as the complete set of original CSS colors will be replaced with a newer one.

3.1.2 SPRWeb

SPRweb[] by David et. al. is a tool that recolors websites to preserve subjective responses and improve color differentiability thus enabling users with CVD to have similar on-line experiences. To recolor web pages, it uses a mapping available in DTEP(Dichromacy Trichromacy Equivalence Plane) set.

To find the DTEP set previous researches, have used findings from unilateral dichromats (individuals who are dichromatic in one eye, but are trichromatic in the other) to identify

the set of colors that are perceived identically in dichromatic color vision and typical color vision.

DTEP set contains all the colors which are perceived same as by a normal person and a CVD person. It also lists all the colors which are perceived by a CVD person.

The key difference in our initial approach and that of SPRweb is the availability of a bigger set of colors and the inclusion of optimization of cost corresponding to *perceptual naturalness*, *perceptual differentiability*, *subjective naturalness* and *subjective differentiability*.

We developed following algorithm implementing approach mentioned in SPRWeb to compare its performance with our final algorithm.

Performance Improvement in SPRWeb   While implementing SPRWeb to do performance comparison, we came across some redundant calculations which were happening at every step. By storing the result of those calculations once and then reusing them gave us a significant performance improvement.

- Improvement Naturalness cost calculation: Procedure *PercepNaturalness* in Algorithm 6 computes a summation and has a time complexity of **O(n)** where n is the number of colors parsed from CSS file. And *PercepNaturalness* is called every time we make a change in our *ColorRep* set during optimization.

  Instead of computing whole summation every time, we can compute the summation only once and then make changes in that for further replacement (because at each step only one color is being replaced and we can just change the sum according to that rather than re-computing whole summation), thus reducing computations.

  We can rewrite our *PercepNaturalness* procedure as shown in Algorithm 8. As we can see that the for loop for summation will get executes only once. Thus the time complexity of one call to *PercepNaturalness* has an average cost of **O(1)** instead of

O(n).

Some other changes such as passing the value of of the index where the colors is getting replaced and the value of color which is getting replaced will have to be passed as well to the *CalculateCost* procedure. But they all can be done without any added complexity.

- Improvement differentiability cost calculation: Procedure *PercepDifferentiability* in Algorithm 6 computes a double summation and has a time complexity of $O(n^2)$ where n is the number of colors parsed from CSS file. And *PercepDifferentiability* is called every time we make a change in our *ColorRep* set during optimization process.

  Instead of computing the two summations every time, we can compute the inner summation only once and then make changes in that for further replacement (because at each step only one color is being replaced and we can just change the sum according to that change rather than re-computing the double summation), thus reducing computations.

  We can rewrite our *PercepDifferentiability* procedure as shown in Algorithm 9. As we can see that the inner for loop for summation will get executes only once. Thus the time complexity of one call to *PercepNaturalness* has an average cost of O(n).

3.1.3   SECTION-TITLE

Algorithm 1 Recoloring 1.1

1: **procedure** RECOLORWEB(CSSFILE)

2:     $ColorsHex, ColorMap \leftarrow$ *CSSParser(CSSFile)*

3:     **for** $i \leftarrow 1$ to $ColorsHex.length$ **do**

4:         **for** $j \leftarrow 1$ to $ColorsHex.length$ **do**

5:             **if** LUT$[i][j] == 0$ **then**

6:                 $ColorsHex[i] \leftarrow S[S.length]$

7:                 $S.length \leftarrow S.length - 1$

8:     $NewCSSFile \leftarrow ReplaceColors(CSSFile, ColorsHex, ColorMap)$

9:     **return** $NewCSSFile$

10: **procedure** CSSPARSER(CSSFILE)

11:     $a \leftarrow readLine(CSSFile)$

12:     $j \leftarrow 1$

13:     **while** $a \neq EOF$ **do**

14:         **if** $re.find(\#xxxxxx || rgb(r, g, b) || hsl(h, s, l)) ==$ **true then**

15:             $Colors[j] \leftarrow a[StartOfFind : EndOfFind]$

16:             $ColorMap[j + +] \leftarrow (StartOfFind, EndOfFind)$

17:         $a \leftarrow readLine(CSSFile)$

18:     **for** $i \leftarrow 1$ to $Colors.length$ **do**

19:         $ColorsHex[\mathbf{i}] \leftarrow ToHex(Colors[i])$

20:     **return** $ColorsHex, ColorMap$

21: **procedure** REPLACECOLORS(CSSFILE, COLORSHEX, COLORMAP)

22:     $a \leftarrow readLine(CSSFile)$

23:     $j \leftarrow 1$

24:     **while** $a \neq EOF$ **do**

25:         **if** $re.find(\#xxxxxx || rgb(r, g, b) || hsl(h, s, l)) ==$ **true then**

26:             $a[ColorMap[j][0] : ColorMap[j][1]] \leftarrow ColorsHex[j + +]$

27:         $a \leftarrow readLine(CSSFile)$

28:     **return** $NewCSSFile$

Algorithm 2 Recoloring 1.2

1: **procedure** RECOLORWEB(CSSFILE)

2:     $ColorsHex, ColorMap \leftarrow$ *CSSParser(CSSFile)*

3:     **for** $i \leftarrow 1$ to $ColorsHex.length$ **do**

4:         **for** $j \leftarrow 1$ to $SETS.length$ **do**   ▷ SETS contains the predefined set of non-conflicting colors

5:             **if** $ColorsHex[i] \in SETS[j]$ **then**

6:                 $Replaced \leftarrow i$ ▷ Helps in keeping the original color and thus naturalness

7:                 $SetChosen \leftarrow j$

8:                 **break**

9:     **for** $i \leftarrow 1$ to $ColorsHex.length$ **do**

10:         **if** $i == Replaced$ **then**

11:             **continue**

12:         $ColorsHex[i] \leftarrow SETS[j][i]$

13:     $NewCSSFile \leftarrow ReplaceColors(CSSFile, ColorsHex, ColorMap)$

14:     **return** $NewCSSFile$

15: **procedure** CSSPARSER(CSSFILE)

16:     *Defined in Algorithm 1*

17: **procedure** REPLACECOLORS(CSSFILE, COLORSHEX, COLORMAP)

18:     *Defined in Algorithm 1*

Algorithm 3 SPRWeb

1: **procedure** SPRWEB(CSSFILE)

2:     $ColorsHex, ColorMap \leftarrow$ *CSSParser(CSSFile)*

3:     $ColorsOrig \leftarrow HexToCIELab(ColorsHex)$

4:     $DTEPSampled \leftarrow UniformSample(DTEP, 900)$ $\triangleright$ gets 900 uniform samples from DTEP

5:     $ColorsRep \leftarrow RAND(DTEPSampled, ColorsOrig.length)$ $\triangleright$ Initializing Replaced color set with random colors from DTEPSampled

6:     $FirstOuput \leftarrow FirstPass(ColorsOrig, ColorsRep, DTEPSampled)$

7:     **for** $i \leftarrow 1$ to $FirstOuput.length$ **do**

8:         **for** $j \leftarrow 1$ to $DTEP$ **do**

9:             **if** $distance(FirstOuput[i], DTEP[j]) < 5$ **then** $\triangleright$ Looks for all the points at 5 Lab units

10:                 $DTEPFiltered[i].append(DTEP[j])$

11:     $FinalOutput \leftarrow SecondPass(ColorsOrig, FirstOuput, DTEPFiltered)$

12:     $ColorsHex \leftarrow CIELabToHex(FinalOutput)$

13:     $NewCSSFile \leftarrow ReplaceColors(CSSFile, ColorsHex, ColorMap)$

14:     **return** $NewCSSFile$

Algorithm 4 SPRWeb:continue

15: **procedure** FIRSTPASS(COLORSORIG,COLORSREP,DTEPSAMPLED)

16:     $cost \leftarrow CalculateCost(ColorsOrig, ColorsRep)$

17:     $Color Rep1 \leftarrow ColorsRep$ ▷ Initializing a variable to check optimization

18:     **while true do**

19:         **for** $i \leftarrow 1$ to $ColorsRep.length$ **do**

20:             **for** $j \leftarrow 1$ to $DTEPSampled$ **do**

21:                 $Value \leftarrow ColorsRep[i]$

22:                 $ColorsRep[i] \leftarrow DTEPSampled[j]$ ▷ trying a new color from DTEPSampled

23:                 $NewCost \leftarrow CalculateCost(ColorsOrig, ColorsRep)$ ▷ Calculating new cost after the change

24:                 **if** $NewCost > cost$ **then**

25:                     $ColorsRep[i] \leftarrow Value$ ▷ increased cost, Rejecting the color

26:         $ColorsRep2 \leftarrow ColorsRep1$

27:         $ColorsRep1 \leftarrow ColorRep$

28:         **if** $ColorsRep1 == ColorRep2$ **then**

29:             **break**

30:     **return**$ColorRep1$

31: **procedure** CSSPARSER(CSSFILE)

32:     *Defined in Algorithm 1*

33: **procedure** REPLACECOLORS(CSSFILE, COLORSHEX, COLORMAP)

34:     *Defined in Algorithm 1*

Algorithm 5 SPRWeb:continue

35: **procedure** SECONDPASS(COLORSORIG,COLORSREP,DTEPFILTERED)

36:    $cost \leftarrow CalculateCost(ColorsOrig, ColorsRep)$

37:    $ColorRep1 \leftarrow ColorsRep$ ▷ Initializing a variable to check optimization

38:    **while true do**

39:        **for** $i \leftarrow 1$ to $ColorsRep.length$ **do**

40:            **for** $j \leftarrow 1$ to $DTEPFiltered[i]$ **do**

41:                $Value \leftarrow ColorsRep[i]$

42:                $ColorsRep[i] \leftarrow DTEPFiltered[i][j]$ ▷ trying a new color from DTEPSampled

43:                $NewCost \leftarrow CalculateCost(ColorsOrig, ColorsRep)$ ▷ Calculating new cost after the change

44:                **if** $NewCost > cost$ **then**

45:                    $ColorsRep[i] \leftarrow Value$ ▷ increased cost, Rejecting the color

46:        $ColorsRep2 \leftarrow ColorsRep1$

47:        $ColorsRep1 \leftarrow ColorRep$

48:        **if** $ColorsRep1 == ColorRep2$ **then**

49:            **break**

50:    **return**$ColorRep1$

Algorithm 6 SPRWeb:continue

51: **procedure** CALCULATECOST(COLORSORIG,COLORSREP)

52:     $pn \leftarrow PercepNaturalness(ColorsOrig, ColorsRep)$

53:     $pd \leftarrow PercepDifferentiability(ColorsOrig, ColorsRep)$

54:     $spn \leftarrow SubNaturalness(ColorsOrig, ColorsRep)$

55:     $spd \leftarrow SubDifferentiability(ColorsOrig, ColorsRep)$

56:     $cost \leftarrow pn + pd + spn + spd$

57:     **return** $cost$

58: **procedure** PERCEPNATURALNESS(COLORSORIG,COLORSREP)

59:     $pn \leftarrow 0$

60:     **for** $i \leftarrow 1$ to $ColorsOrig.length$ **do**

61:         $pn \leftarrow pn + distance(ColorsOrig[i], ColorsRep[i])$

62:     $pn \leftarrow pn/ColorsOrig.length$

63:     **return** pn

64: **procedure** PERCEPDIFFERENTIABILITY(COLORSORIG,COLORSREP)

65:     $pd \leftarrow 0$

66:     **for** $i \leftarrow 1$ to $ColorsOrig.length$ **do**

67:         **for** $j \leftarrow i + 1$ to $ColorsRep.length$ **do**

68:             $pd \leftarrow pd + abs(distance(ColorsOrig[i], ColorsOrig[j]) - distance(ColorsRep[i], ColorsRep[j]))$

69:     $pd \leftarrow pd/((ColorsOrig.length) * (ColorsOrig.length - 1))$

70:     **return** pd

Algorithm 7 SPRWeb:continue

71: **procedure** SUBNATURALNESS(COLORSORIG,COLORSREP)

72:     **return** $PercepNaturalness(Sub(ColorsOrig), Sub(ColorsRep))$

73: **procedure** SUBDIFFERENTIABILITY(COLORSORIG,COLORSREP)

74:     **return** $PercepDifferentiability(Sub(ColorsOrig), Sub(ColorsRep))$

75: **procedure** SUB(COLOR)   ▷ Transforms a color to the subjective space

76:     $L \leftarrow Color[0]$

77:     $a \leftarrow Color[1]$

78:     $b \leftarrow Color[2]$

79:     **if** $a == 90$ **then**

80:         $H \leftarrow 90$

81:     **else**

82:         $H \leftarrow tan^{-1}b/a$

83:     $C \leftarrow sqrt(a^2 + b^2)$

84:     $activity \leftarrow -2.1 + 0.06 * sqrt((l - 50)^2 + (a - 3)^2 + ((b - 17)/1.4)^2)$

85:     $temp \leftarrow -0.5 + 0.02 * C^{1.07} * cos(H - 50)$

86:     $weight \leftarrow -1.8 + 0.04 * (100 - l) + 0.45 * cos(H - 100)$

87:     **return** $(activity, temp, weight)$

Algorithm 8 Improvements in SPRWeb

1: **procedure** PERCEPNATURALNESS(COLORSORIG,COLORSREP,k,OLDCOLOR)   ▷
   k=index of replacement, OldColor = Color being replaced

2:   **if** $pn == InitialzedValue$ **then**   ▷ is true only once - during first computation of
   cost

3:      **for** $i \leftarrow 1$ to $ColorsOrig.length$ **do**

4:         $pn \leftarrow pn + distance(ColorsOrig[i], ColorsRep[i])$

5:      $pn \leftarrow pn/ColorsOrig.length$

6:   **else**

7:      $pn \leftarrow pn * ColorsOrig.length$

8:      $pn \leftarrow pn - distance(ColorsOrig[k], OldColor) + distance(ColorsOrig[k], ColorRep[k])$

9:   **return** pn

Algorithm 9 Improvements in SPRWeb

1: **procedure** PERCEPDIFFERENTIABILITY(COLORSORIG,COLORSREP,k,OLDCOLOR)

   $\triangleright$ k=index of replacement, OldColor = Color being replaced

2:    **if** $pd == InitialzedValue$ **then** $\triangleright$ is true only once - during first computation of cost

3:        **for** $i \leftarrow 1$ to $ColorsOrig.length$ **do**

4:            **for** $j \leftarrow i + 1$ to $ColorsRep.length$ **do**

5:                $pd \leftarrow pd + abs(distance(ColorsOrig[i], ColorsOrig[j]) - distance(ColorsRep[i], ColorsRep[j]))$

6:        $pd \leftarrow pd/((ColorsOrig.length) * (ColorsOrig.length - 1))$

7:    **else**

8:        $pd \leftarrow pd * ((ColorsOrig.length) * (ColorsOrig.length - 1))$

9:        **for** $i \leftarrow 1$ to $ColorsOrig.length$ **do**

10:            $pd \leftarrow pd - abs(distance(ColorsOrig[i], ColorsOrig[k]) - distance(ColorsRep[i], OldColor))$

11:            $pd \leftarrow pd + abs(distance(ColorsOrig[i], ColorsOrig[k]) - distance(ColorsRep[i], ColorRep[k]))$

12:        $pd \leftarrow pd/((ColorsOrig.length) * (ColorsOrig.length - 1))$

13:    **return** pd

APPENDIX-TITLE

INSERT-APPENDIX-TEXT-HERE

A.1 SECTION-TITLE

A.2 SECTION-TITLE

A.2.1 SECTION-TITLE

A.2.2 SECTION-TITLE