APPROVAL SHEET

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

Name of Candidate: Vikas Bansal
Masters in Science, 2014

Thesis and Abstract Approved: _____

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

_____

Dr. Lina Zhou
Associate Professor
Department of Information Systems

Date Approved: _____

Curriculum Vitae

Name: Vikas Bansal.

Permanent Address: 130 Forest St, Stamford, CT 06901.

Degree and date to be conferred: Masters in Science, May 2014.

Date of Birth: 01/27/1991.

Place of Birth: India.

Secondary Education: Columbia Foundation Senior Secondary School, New Delhi, New Delhi.

Collegiate institutions attended:

University of Maryland Baltimore County, Masters in Science Computer Science, 2014.

LNMIIT, Bachelor of Technology (Hons.) Electronics and Communication Engineering, 2012

Major: Computer Science.

Minor: Computer Science.

Professional positions held:

Intern at Xerox Corporation. (May 28, 2013 – August 23, 2013).

Research Assistant at UMBC. (January 13, 2013 – June 30, 2014).

Teaching Assistant at UMBC. (January 20, 2014 – May 21, 2014).

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

Color vision begins with the activation cone cells. When one of the cone cells dysfunction, color vision deficiency (CVD) ensues. Due to CVD, users become unable to differentiate as many colors a normal person can. Lack of this ability results in less rich web experience, incomprehension of basic information and thus frustration. Solutions such as carefully choosing colors while designing or recolor web pages for CVD users exist. We first present the improvement in the time complexity of an existing tool SPRWeb[6] to recolor web pages. After that we present our tool, FBRecolor, which explores the foreground-background relationship between colors in a web page. Using this relationship we propose FBRecolor, which preserves naturalness, pair-differentiability and subjectivity. In the last part, we add an additional step in to FBRecolor to ensure that the contrast in the parsed color pairs meets the required W3C guidelines[5]. In evaluation, we found that FBRecolor does significantly better in preserving pair-differentiability and produces lower total cost solutions than SPRWeb. Quantitative experimentation of extension to FBRecolor shows that contrast ratio in each replacement pair is more than 4.5 as required for readability.

Recoloring Web Pages For Color Vision Deficiency Users.

by

Vikas Bansal

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Masters in Science
2014

I dedicate this thesis to mom, dad, Rajesh chacha and me.

ACKNOWLEDGMENTS

I am very thankful to Dr. Lina Zhou, Dr. Dongsong Zhang and Dr. Tim Finin for their time and contribution. They are really good professors and were helpful to me through my entire research time. Not only did they teach me how to be a good researcher, but also they showed me how to use time effectively and efficiently to get the expected goal.

Besides, I am very grateful to my friends, specially, Primal Pappachan, Manik Budhiraja and Akshay Grover for their support.

Finally, I would like to thank my family for all kinds of supports. Life would have been very difficult without them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Normal color vision begins with the activation of three different types of cone cells, each sensitive to a different range of wavelength. These are often called S cones, M cones, and L cones, for short, medium, and long wavelength; but they are also often referred to as blue cones, green cones, and red cones, respectively. Color vision deficiency (CVD) is a result of missing either one of the cones of a shift in sensitivity. Color perception of human can described by two orthogonal axis a red-green axis and a blue-yellow axis. Most common type of deficiency is along red-green axis.

CVD is very common in humans affecting more than 5% of the world's population [2]. Colors convey important information in our daily lives. Colors, specially in web sites, influence users' experience significantly. Colors fulfill the basic need of providing legibility between foreground text and background, for example [5]. In addition to this, they also influence the subjective responses [3,4,6]. That is, depending on the color, a website may seem heavy or light, high or low in temperature and may tell about its busyness as well.

CVD users are not able to differentiate between as many colors as a normal person can. As a result, they often are unable to understand the content on a web page designed keeping normal users in mind.

One solution, a preventive measure, to the problem of illegibility could be to educate

web designers so that they only use a set of colors which are differentiable to both CVD and normal users. The second solution, a curative one, is to recolor existing web pages (designed for normal users) keeping CVD users in mind.

Several color schemes have been developed to implement the first solution. The obvious disadvantage of the first solution is that we are restricting the web designer to stick to a particular scheme while designing. Several recoloring tools exist to implement the second solution. They mainly focus on replacing original colors with similar colors to preserve naturalness and does well in preserving differentiability too. A recent tool, SPRWeb[6], preserves subjective responses as well. But none of the tools explore the foreground-background relationships of colors in a web page to preserve the mentioned factors. None of the tools makes sure that the recolored version of the web page has sufficient contrast.

To address these issues, we first present an improvement on SPRWeb which significantly reduces the time complexity of finding the replacement colors while preserving naturalness, differentiability and subjectivity factors. After that we present our approach of parsing a web page to obtain foreground-background color pairs. We use this parsing approach and present a new 3 step algorithm,FBRecolor, to reach to the replacement color set. In the later section, we present a modified FBRecolor which makes sure that the replacement colors have the required contrast threshold.

In evaluation we compared FBRecolor with SPRWeb on naturalness, pair differentiability, subjective responses and total cost. Results show that FBRecolor performs better in 37 of the 50 test pages and performs slightly less in 7 pages. Testing of modified FBRecolor shows that modified FBRecolor ensures that the required contrast is present in recolored pages, while SPRWeb failing to make sure this in every case.

Chapter 2

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 [1], 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 [2]. 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

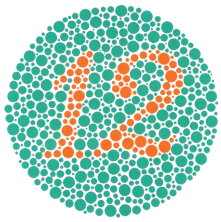
2.2 Diagnosis

The Ishihara color test [14], 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.

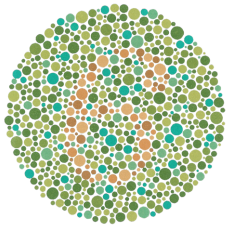
To test the type of color blindness users have, we did the Ishihara color test on each of them during our user study of the algorithm.

2.3 Related Work

Quite a work has been done to both simulate and correct CVD. Ichikawa et al. [7] were the first publishers of a website recolouring tool that improves accessibility for people with CVD. Iaccarino et al. [21] proposed the use of edge services to recolour website images in transit to the user. ColorBlindnessSimulateCorrect by Seewald Solutions [8] is one of the examples of an application developed to both simulate and correct CVD effect. It uses



(a) Plate No. 1 (12)



(b) Plate No. 13 (6)

FIG. 2.1: Ishihara Test

a linear transformation on rgb input. Daltonize developed by www.daltonize.org [9] 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 [20] 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 [19] 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 naturalness, differentiability of the original web page.

One such tool which preserves naturalness and improves color differentiability is kuhns [10]tool. One improvement on naturalness preservation was done by David et al. in SPRWeb [6]. 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 foreground 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[6]:** 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 [15][16] 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[6]:** 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[6]:** 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 [11] 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[6]:** 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 [15][16]:** 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). 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 [11] :** *Subjective response space* consists of three dimensions which can be formulated as follows using Ou's model [11]:

Activity(Active, Passive):

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

Temperature(Warm, Cool):

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

Weight(Heavy, Light):

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

Chroma:

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

Hue:

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

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

2.6 Range of value distribution for the four types of costs

- **Perceptual Naturalness** - Cost depends on number of colors N. We also scale it by N thus we can write the range as $[0, Max_distance]$. Where $Max_distance$ is 372.86. (The maximum distance possible in two points in CIELAB space)

- **Perceptual Differentiability** - Again the cost depends on number of colors N. We also scale it by $N * (N - 1)$ thus we can write the range as $[0, Max_distance]$. Where $Max_distance$ is 372.86. (The maximum distance possible in two points in CIELAB space)

- **Pair Differentiability** - The cost depends on the number of pair-colors in the web page. As shown earlier in the document, number of pairs were similar to number of colors in web page. Thus we can write range as $[0, N * Max_distance]$. Where $Max_distance$ is 372.86. (The maximum distance possible in two points in CIELAB space)

- **Activity** $[-2.1, 8.28]$

- **Temperature** $[-5.66, 4.66]$

- **Weight** $[-2.25, 2.65]$

- **Subjective naturalness** Cost depends on number of colors N. We also scale it by N thus we can write the range as $[0, Max_sdistance]$. Where $Max_sdistance$ is 15.43. (The maximum distance possible in two points in subjective response space)

Contrast	Pair Differentiability
1. Contrast is ratio of two colors in sRGB space.	1. Distance between two colors in CIELab color space.
2. More the contrast between colors, easier it is to read the content developed using the same two colors.	2. However, more differentiability between two colors does not ensure more ease in understanding the content.
3. There is a specific numerical value of contrast suggested by W3C to ensure a comfortable level of readability in text.	3. There is no such specific numerical suggested by W3C for pair differentiability.
4. We make sure in our algorithm that the minimum required threshold is present when we recolor pairs to improve readability.	4. We make sure that the pair differentiability in recolored version is as close to original version as possible to retain the original color scheme.

FIG. 2.2: Contrast vs Pair-differentiability

- **Subjective differentiability** Again the cost depends on number of colors N. We also scale it by $N * (N - 1)$ thus we can write the range as $[0, Max_s distance]$. Where $Max_s distance$ is 15.43. (The maximum distance possible in two points in subjective response space)

- **Subjective pair differentiability** The cost depends on the number of $pair - colors$ in the web page. As shown earlier in the document, number of pairs were similar to number of colors in web page. Thus we can write range as $[0, N * Max_s distance]$. Where $Max_s distance$ is 15.43. (The maximum distance possible in two points in subjective response space)

2.7 Difference between contrast and pair-differentiability

REFERENCES

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, FBRecolor, 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.

(a) As seen by a normal person

(b) As seen by CVD person (Protanopia)

FIG. 3.2: Example of conflict

Safe colors

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.

As we can observe, a CVD person can only see a subset of all possible colors. In other

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

FIG. 3.4: Color Palette

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.

3.1.3 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).

3.2 FBRecolor

While browsing the content on-line, specially reading, color of the text and the background are of key significance. Contrast between those two colors is one of the several important factors among other factors as mentioned in Definitions sections. SPRWeb tries to optimize the cost corresponding to perceptual naturalness, perceptual differentiability, subjective naturalness and subjective differentiability for the entire parsed set of colors. But at the very instant when user is reading a particular element of a web page, he/she may not be concerned about the other elements of the page. Therefore, we can optimize the cost corresponding to various factors for a particular element of the web page first and then repeat the same process for all the remaining elements.

3.2.1 Key idea

The idea is that while optimizing the various factors for a particular element rather than the whole web page, we will have more color options to choose from and thus the possibility of doing quantitatively better.

We divide a web page in elements based on the foreground-background color combination. For example, if a web page has two division elements, each having one color for background and one for the foreground text color, then the web page has two elements to be

processed. Each element having one foreground-background(fg-bg) color pair. We define, a color pair as a combination of foreground and background for a particular element on a page.

Initial step of our algorithm parses out such fg-bg color pairs. Such fg-bg color pairs can be extracted by analysis of HTML Document Object Model(DOM) structure and CSS files. One of the possible implementations is shown later in the section. Once we have all the the color pairs parsed out, we perform our algorithm on each of the color pair. Pseudo code of the algorithm is given in Algorithm 10. A textual description is given in next section.

3.2.2 Implementation

FBRecolor defines our tool. The input is the CSS file corresponding to the web page and the Document Object Model(DOM) of the web page. Output is a CSS file which is recolored using our tool. **FBRecolor** calls **NewCSSParser** whose input is also a CSS file and the DOM structure. The output is the pairs of foreground-background colors present in the CSS file. Procedure like **NewCSSParser** can be implemented by analyzing the DOM model and the CSS file both.

In the CSS file, we can read the contents as a string and use regular expressions to parse the IDs and classes which it has and their corresponding colors in block.A mapping can be maintained which has IDs and .class as keys and a list as value. List contains the color properties(foreground and background) in that ID or class.

To analyze the DOM structure of HTML web page, we can use tools such as BeautifulSoup. The Default pair is foreground and background colors of the body tag. We can take a default value if they are not defined. Using such modules, we can traverse the DOM tree and check the class or ID which they are using. If no class or IDs or font color attributes are present that means they simply inherit the color of the parent and no new color is added to the system. But if there is some ID or class or color or font color defined, we retrieve its color

property from dictionary maintained and report a new pair. Pairs are stored in separate lists like foreground color in one list and background color in other list.

After we obtain a **Foreground** and **Background** color list, we pass one pair at once to procedure **FindRecolor**. **FindRecolor** returns the recolored pair corresponding to the input color. It optimizes the naturalness and differentiability among pairs. The three major steps which are present in **FindRecolor** can be explained as follows:

- **Preserving naturalness:** In the first step, we find a replacement fg' of a color fg from a pair fg-bg in DTEP space. The replacement is such that the distance between fg and fg' is as minimum as possible. This optimization maintains one of the perceptual naturalness as defined in SPRWeb[]. This ensures that the replacement color is as close to the original color as possible and there is no dramatic shift in replacement colors from original colors.

- **Preserving differentiability:** In the second step, we draw a shell around fg' with internal radius as $d - e$ and outer radius as $d + e$, to obtain a possible set of colors for choosing a replacement bg' for bg. d is the distance between original fg-bg color and e is the factor introduced to ensure that we reach a solution. This step optimizes the perceptual differentiability property. This perceptual differentiability is different from what defined in SPRWeb. According to this property we try to maintain the contrast as it was present in original pair.

- **Preserving naturalness:** In the third step, we choose a replacement bg' for bg in possible set bg such that the distance between bg and bg' is minimum, again maintaining the perceptual naturalness property.

After we obtain the list of replacements for **Foreground** and **Background**, we need a list of replacement colors of size equal to original colors present CSS file. Such a list is needed

to replace colors back in original CSS and also to make sure that if a color is present in both background and foreground of different elements, then no conflict should arise. To achieve this, we first search for a color, present in original color list, in Foreground and Background color list. If the same color is present both in Foreground and Background then that means we have two candidate replacements for the same color. We choose a replacment for both the colors such that we get minimum cost of naturalness and differentiability in the pairs that contain those two colors.

If a color is present only in Foreground and Background then we just put replacement of Foreground or Background in the replacement color list.

After obtaining the replacement color list, we rewrite the CSS files reflecting new colors.

3.3 Maintaining minimum contrast

Contrast is one of the major factors which is important in making a web page readable. In all the techniques discussed, measures are taken to ensure that the original colors and the contrast among them are preserved as much as possible. But there can be a case in which a color pair of replaced foreground and background exist which doesn't meeting the contrast criteria for reading.

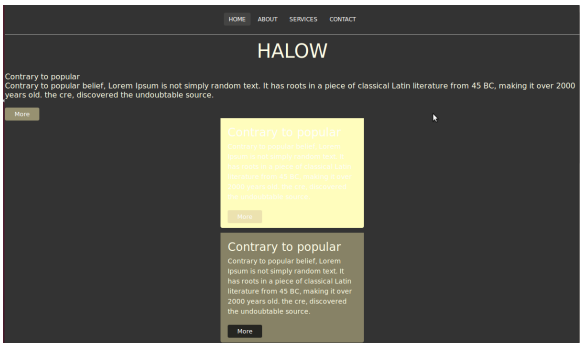


FIG. 3.5: Recoloring leading to bad contrast

To resolve such conflicts, we can follow the W3C guidelines of contrast threshold as provided in WCAG 2.0.

3.3.1 W3C guidelines

- this technique is needed to make sure that users can read text that is presented over a background.
- If the background is a solid color (or all black or all white) then the relative luminance of the text can be maintained by making sure that each of the text letters have 4.5:1 contrast ratio (CR) with the background.
- If the background or the letters vary in relative luminance (or are patterned) then the background around the letters can be chosen or shaded so that the letters maintain a 4.5:1 contrast ratio with the background behind them even if they do not have that contrast ratio with the entire background.
- For example, if a letter is lighter at the top than it is a the bottom, it may be difficult to maintain the contrast ratio between the letter and the background over the full letter. In this case, the designer might darken the background behind the letter, or add a thin black outline (at least one pixel wide) around the letter in order to keep the contrast ratio between the letter and the background above 4.5:1.
- The contrast ratio can sometimes be maintained by changing the relative luminance of the letters as the relative luminance of the background changes across the page.

The procedure to implement the method as suggested by W3C is follows:

- **Measure the relative luminance:** of foreground text using the formula:

$$L = 0.2126 * R + 0.7152 * G + 0.0722 * B$$

R if $R_{sRGB} <= 0.03928$ then $R = R_{sRGB}/12.92$ else $R = ((R_{sRGB} + 0.055)/1.055)^{2.4}$

G if $G_{sRGB} <= 0.03928$ then $G = G_{sRGB}/12.92$ else $G = ((G_{sRGB} + 0.055)/1.055)^{2.4}$

B if $B_{sRGB} <= 0.03928$ then $B = B_{sRGB}/12.92$ else $B = ((B_{sRGB} + 0.055)/1.055)^{2.4}$

R_{sRGB} $R_{8bit}/255$

G_{sRGB} $G_{8bit}/255$

B_{sRGB} $B_{8bit}/255$

- Measure the relative luminance of the background pixels immediately next to the letter using same formula.
- Calculate the contrast ratio using the following formula.

$$contrast = (L1 + 0.05)/(L2 + 0.05)$$

L1 Relative luminance of the lighter of the foreground or background colors

L2 Relative luminance of the darker of the foreground or background colors.

- Check that the contrast ratio is equal to or greater than 4.5:1

3.3.2 Including W3C guidelines in our approach

To include W3C guidelines in our approach, we analyzed the luminance values of the colors present in DTEP space. Following is the plot which was obtained: X-axis is index of DTEP color. Y-axis is luminance value. The graph is oscillating in nature.

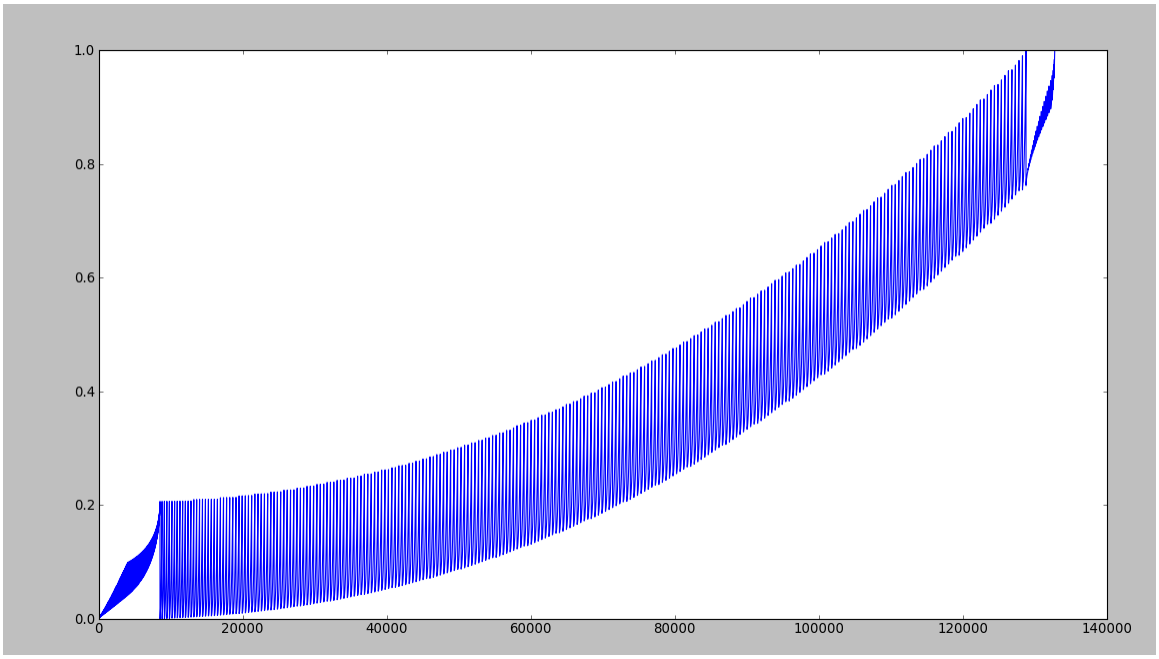


FIG. 3.6: Luminance plot for DTEP

If we see the plot of contrast ratio of a color with the rest of DTEP colors, it would be as follows: If we observe we can see that the graph is particularly divided in two regions, one has a decreasing contrast ratio values followed by a unity point and then we have a region where contrast ratio is somewhat increasing.

If we arrange the DTEP set according to the luminance values, we can obtain a graph like in Fig 3.8:

And using this same sorted DTEP set, we can obtain the contrast ratio graph of one of the colors, lets call A, and the rest of the DTEP members as shown in Fig 3.9.

As we can observe in Fig 3.9, value of the contrast ratio starts from a value, lets call it a starting point and then plunges to 0. Lets call it a unity point. This unity point is essentially the index of occurrence of color A in DTEP set. After that, the contrast ratio increases and reaches its maximum value, lets call it a ending point.

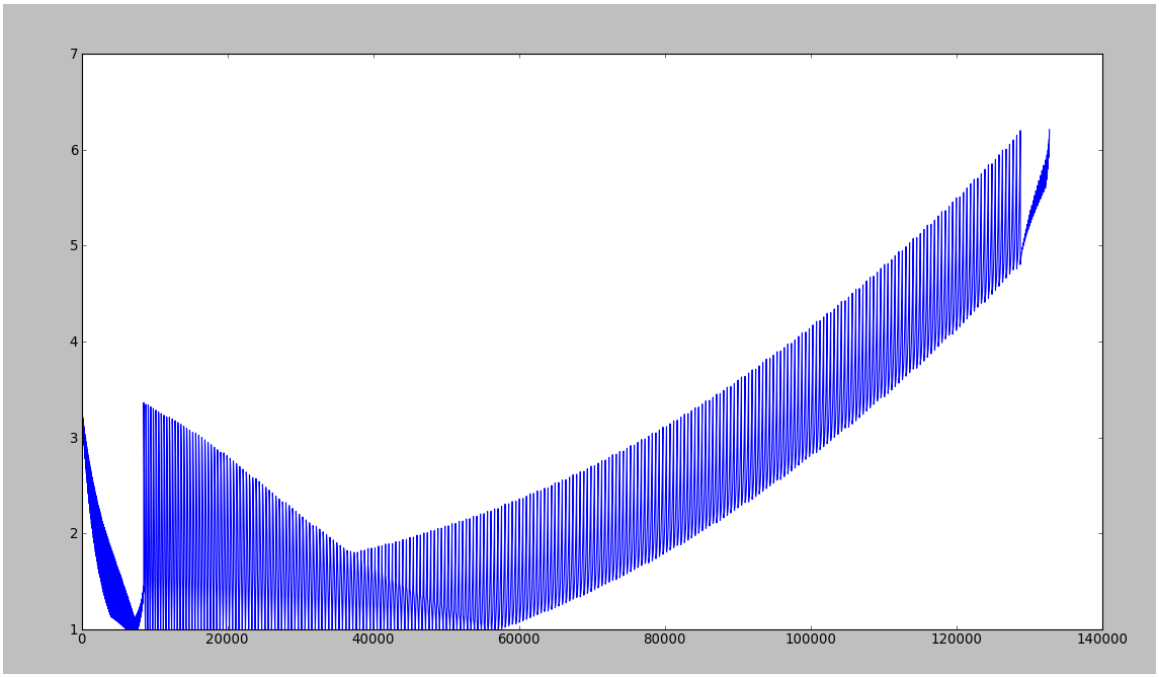


FIG. 3.7: contrast plot for DTEP

Following observations can be made about starting point, unity point and ending point:

- **Starting point can have a contrast value of [0,21])** : As per the arrangement in DTEP set (increasing order of luminance), it is evident that maximum value of contrast ratio can occur at starting point when it is paired up with the last value of DTEP.

- **Contrast ratio strictly declines from starting point till unity point** : At unity point, the color is paired up with itself thus giving a contrast ratio of 1. If we move slightly to the left of unity point, we will see colors which have less luminance value as compared to the color A under consideration. And since the colors to the left have a luminance value less than that of A, their luminance value will be in the denominator of the CR formula. And as we move backwards towards the starting point, the denominator will keep on decreasing (due to the sorted luminance order in DTEP set), increasing the net CR value.

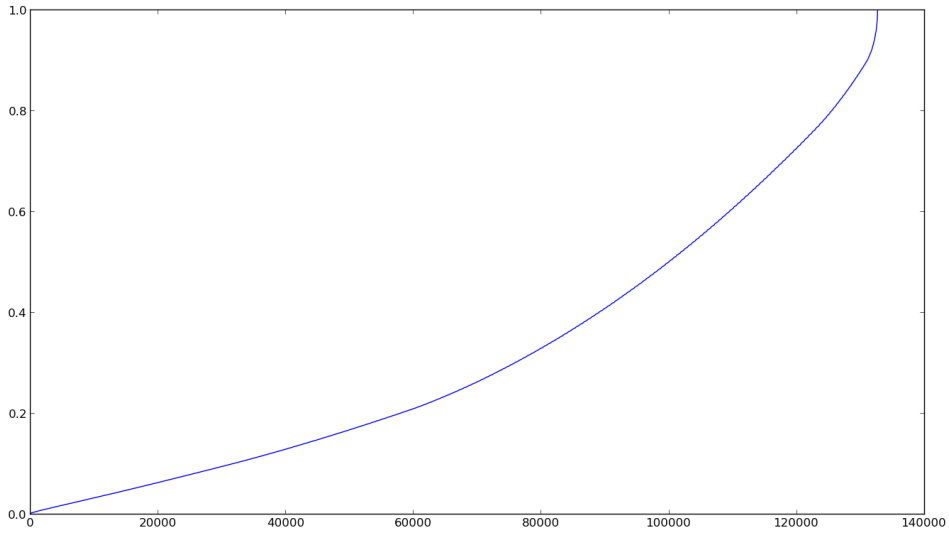


FIG. 3.8: SortedLuminancePlot

- **Contrast ratio strictly increases from unity point till ending point** : If we move slightly rightwards towards the ending point, the value of luminance in colors would increase. In the CR formula, luminance value of color on the right would be on top and that of color A, under consideration, would be in denominator. As we move right, numerator would increase, thus increasing net CR value.

To make sure that the colors we choose satisfy the minimum contrast threshold criteria, we draw a vertical line $Y = 4.5$ in Fig 3.9. The possible plots that can be obtained after that are shown in Fig. 3.10, 3.11 and 3.12:

The gray area in Fig 3.10,3.11 and 3.12 show the region from where we can take the colors such that the W3C guideline of minimum threshold is met.

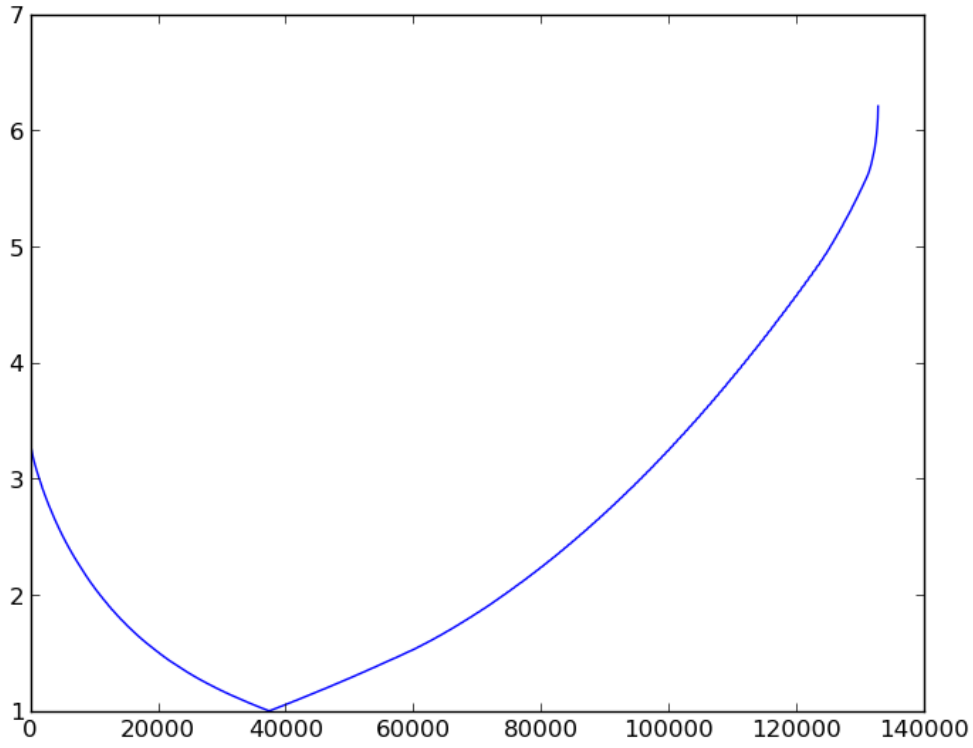


FIG. 3.9: Sorted contrast plot

3.3.3 Implementation

Since we are aiming to make sure that the recolored web pages have color pairs such that the contrast ratio between them meets the 4.5:1 contrast threshold criteria, we need to restrict our possible color search space from DTEP set to a shorter set. Length and the properties of the possible set depends on the color being considered. As shown in Fig 3.10, 3.11, 3.12, the number of possibilities and their location depends on the color. A general approach to include this new criteria in recoloring of a pair, lets say fg-bg, would be to first find a replacement fg' for fg using the **FindRecolor** procedure in algorithm 11. And then obtain a curve like shown in Fig 3.10,11 and 12. Followed by finding the points which

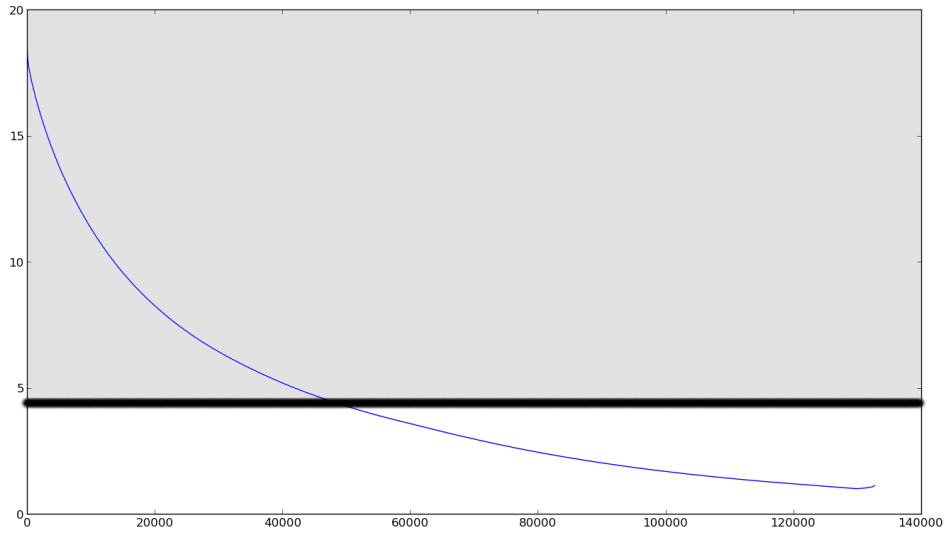


FIG. 3.10: Only starting point included

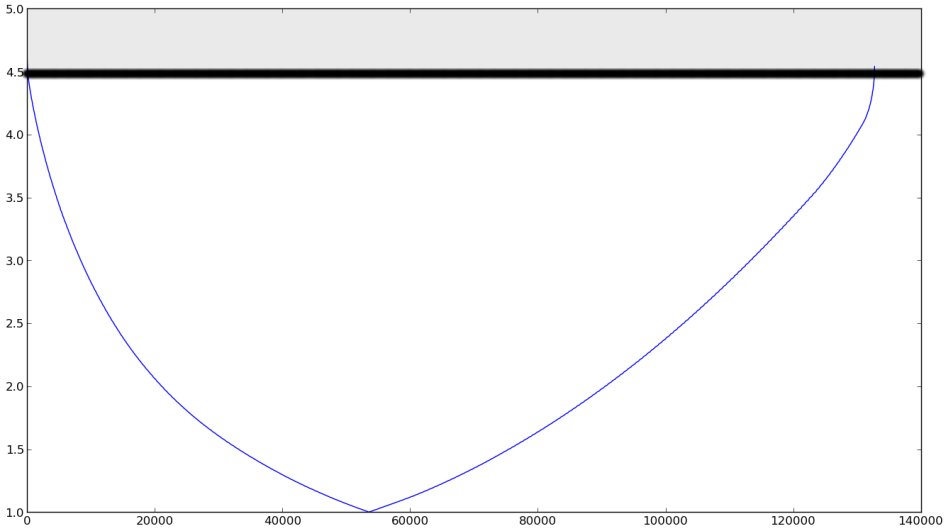


FIG. 3.11: Both starting and ending points included

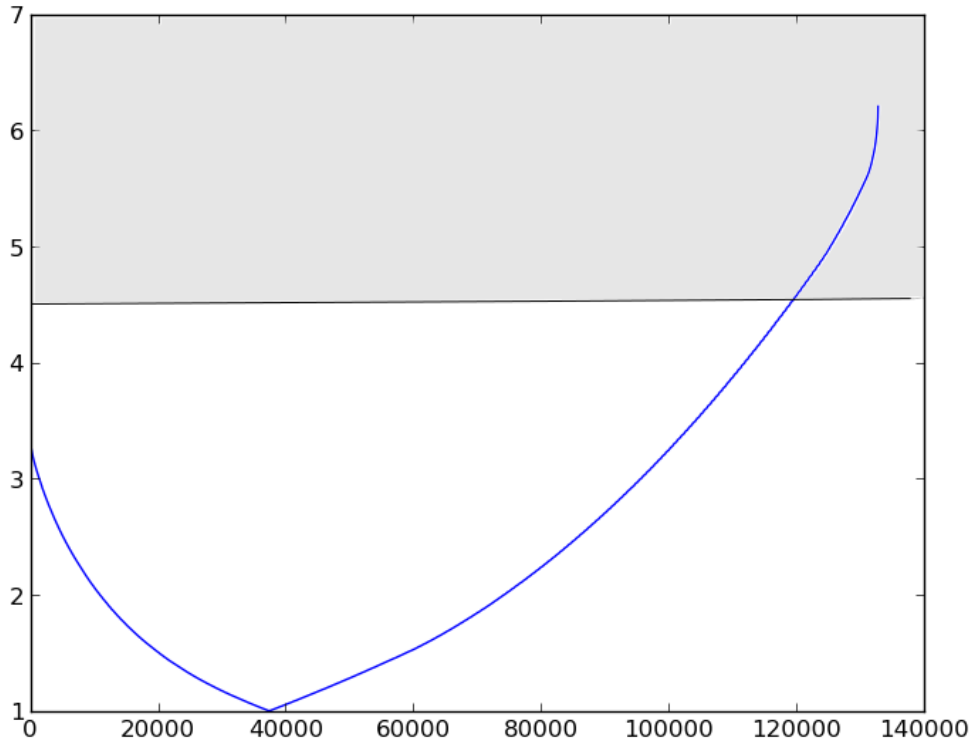


FIG. 3.12: Only ending point included

lie in the gray region in DTEP set and putting them in new *DTEPReduced* set. And then finally searching a possible replacement bg' in the *DTEPReduced* search space using the same approach as mentioned in **FindRecolor** procedure in algorithm 11. But since there are many pairs possible in a web page, obtaining a curve at run time for each and every pair adds to computational complexity.

To resolve the problem of having to calculate the curves and find a *DTEPReduced* set corresponding to a color in color pair, we pre-compute a mapping of possible values. This mapping has each color as a key and *DTEPReduced* as value. A color, lets say x, has a *DTEPReduced* set such that all colors in *DTEPReduced* set when paired with x, produce a

contrast ratio of more than 4.5. Thus, if in a pair fg-bg, where replacement for fg is fg', if we choose a replacement for bg in *DTEPReduced* set, then we would be sure that the recolored pair fg'-bg' has a contrast ratio of more than 4.5.

The mapping can be obtained using the procedure in Algorithm 12.

Using the mapping obtained from Algorithm 13, we can modify procedure **FindRecolor** in Algorithm 11 as shown in Algorithm 14.

Algorithm 14 describes a one possible heuristic to obtain a output color set using the mapping. The heuristic is such that when we recolor a pair lets say fg-bg, and we have many possible replacements for fg. Then out of those possible replacements of fg, we choose a a replacement such that the length of the list in *mapping* is maximum. In other words, we choose a value for fg such that we have a large number of option for bg. This gives us a possibility of reaching to a better solution in terms of naturalness and differentiability, since we have many options available. This is one possible heuristic to reach to the solution, several others heuristics can also be implemented.

Algorithm 1 Recoloring 1.1

```
1:  procedure RECOLORWEB(CSSFILE)
2:      ColorsHex, ColorMap ← CSSParser(CSSFile)
3:      for i ← 1 to ColorsHex.length do
4:          for j ← 1 to ColorsHex.length do
5:              if LUT[i][j] == 0 then
6:                  ColorsHex[i] ← S[S.length]
7:                  S.length ← S.length − 1
8:      NewCSSFile ← ReplaceColors(CSSFile, ColorsHex, ColorMap)
9:      return NewCSSFile
10: procedure CSSPARSER(CSSFILE)
11:     a ← readLine(CSSFile)
12:     j ← 1
13:     while a ≠ EOF do
14:         if re.find(#xxxxxx||rgb(r, g, b)||hsl(h, s, l)) == true then
15:             Colors[j] ← a[StartOfFind : EndOfFind]
16:             ColorMap[j + +] ← (StartOfFind, EndOfFind)
17:         a ← readLine(CSSFile)
18:     for i ← 1 to Colors.length do
19:         ColorsHex[i] ← ToHex(Colors[i])
20:     return ColorsHex, ColorMap
21: procedure REPLACECOLORS(CSSFILE, COLORSHEX, COLORMAP)
22:     a ← readLine(CSSFile)
23:     j ← 1
24:     while a ≠ 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]] ← ColorsHex[j + +]
27:         a ← readLine(CSSFile)
28:     return NewCSSFile
```

Algorithm 2 Recoloring 1.2

```
1: procedure RECOLORWEB(CSSFILE)
2:     ColorsHex, ColorMap ← CSSParser(CSSFile)
3:     for i ← 1 to ColorsHex.length do
4:         for j ← 1 to SETS.length do          ▷ SETS contains the predefined set of non-conflicting colors
5:             if ColorsHex[i] ∈ SETS[j] then
6:                 Replaced ← i                 ▷ Helps in keeping the original color and thus naturalness
7:                 SetChosen ← j
8:                 break
9:     for i ← 1 to ColorsHex.length do
10:        if i == Replaced then
11:            continue
12:        ColorsHex[i] ← SETS[j][i]
13:    NewCSSFile ← 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

procedure SPRWEB(CSSFILE)
  $ColorsHex, ColorMap \leftarrow$ *CSSParser(CSSFile)*
  $ColorsOrig \leftarrow HexToCIELab(ColorsHex)$
  $DTEPSampled \leftarrow UniformSample(DTEP, 900)$ ▷ gets 900 uniform samples from DTEP
  $ColorsRep \leftarrow RAND(DTEPSampled, ColorsOrig.length)$ ▷ Initializing Replaced color set with random colors from DTEPSampled
  $FirstOuput \leftarrow FirstPass(ColorsOrig, ColorsRep, DTEPSampled)$
  for $i \leftarrow 1$ to $FirstOuput.length$ **do**
    for $j \leftarrow 1$ to $DTEP$ **do**
      if $distance(FirstOuput[i], DTEP[j]) < 5$ **then** ▷ Looks for all the points at 5 Lab units
        $DTEPFiltered[i].append(DTEP[j])$
  $FinalOutput \leftarrow SecondPass(ColorsOrig, FirstOuput, DTEPFiltered)$
  $ColorsHex \leftarrow CIELabToHex(FinalOutput)$
  $NewCSSFile \leftarrow ReplaceColors(CSSFile, ColorsHex, ColorMap)$
  return $NewCSSFile$

Algorithm 4 SPRWeb:continue

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

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

17: $ColorRep1 \leftarrow ColorsRep$ ▷ 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)
▷ k=index of replacement, OldColor = Color being replaced
2: **if** $pd == InitialzedValue$ **then** ▷ 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

Algorithm 10 FBRecolor

1: **procedure** FBRECOLOR(CSSFILE,DOM)
2: $Foreground, Background, ColorMap \leftarrow$ *NewCSSParser(CSSFile,DOM)*
3: $ColorsOrig \leftarrow HexToCIELab(Foreground \cup Background)$
4: **for** $j \leftarrow 1$ to $Foreground.length$ **do**
5: $RecoloredForeground[i], RecoloredBackground[i] \leftarrow FindRecolor(HexToCIELAB(Foreground[i]), HexToCIELAB(Background[i]))$
6: **for** $j \leftarrow 1$ to $ColorsOrig.length$ **do**
7: **if** $Foreground.count(ColorsOrig[i])! = 0$ **then**
8: $FIndex \leftarrow Foreground.index(ColorsOrig[i])$
9: **if** $Background.count(ColorsOrig[i])! = 0$ **then**
10: $BIndex \leftarrow Background.index(ColorsOrig[i])$
11: $i \leftarrow xsuchmindist(RecoloredForeground[x], RecoloredBackground[x]) - dist($
12: $ColorRecolor[i] \leftarrow ColorsOrig[i]$
13: **else**
14: $ColorRecolor[i] \leftarrow RecoloredForeground[FIndex]$
15: $ColorRecolor[i] \leftarrow RecoloredBackground[Background.index(ColorsOrig[j])]$
16: $ColorsHex \leftarrow CIELabToHex(ColorRecolor)$
17: $NewCSSFile \leftarrow ReplaceColors(CSSFile, ColorsHex, ColorMap)$
18: **return** $NewCSSFile$

Algorithm 11 FBRecolor:continue

19: **procedure** FINDRECOLOR(FG,BG)

20: $minDist = MAXINF$

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

22: **if** $distance(bg, j) < minDist$ **then**

23: $minDist = distance(bg, DTEP[j])$

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

25: **if** $distance(bg, j) < minDist + 0.5$ **then**

26: $minDistCircle.append(distance(bg, DTEP[j]))$

27: $initialDifferentiability \leftarrow distance(fg, bg)$

28: $a \leftarrow 1$

29: $possibleSet \leftarrow 0$

30: **while** $possibleSet.length <= 0$ **do**

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

32: **if** $(distance(minDistCircle[0], DTEP[j]) > initialDiffrentiablity - a * 0.1 and distance(minDistCircle[0], DTEP[j]) < initialDiffrentiablity + a * 0.1)$: **then**

33: $possibeSet.append(DTEP[j])$

34: $a = a * 2$

35: $minDist \leftarrow MAXINF$

36: **for** $j \leftarrow 1$ to $possibleSet$ **do**

37: **if** $distance(fg, j) < minDist + 0.5$ **then**

38: $minDist = distance(fg, possibleSet[j])$

39: $replacementFg = possibleSet[j]$

40: **return** $replacementFg, minDistCircle[0]$

Algorithm 12 Find Mapping

procedure FINDMAPPING()
  for $j \leftarrow 1$ to $DTEP$ **do**
    $DTEPLuminance[j] \leftarrow luminance(DTEP[j])$
  $DTEPSorted \leftarrow [xfor(y,x)insorted(zip(DTEPLuminance, DTEP))]$ ▷ Sorting DTEP based on DTEPLuminance
  $DTEPLuminance \leftarrow Sorted(DTEPLuminance)$ ▷ sorting the list
  for $j \leftarrow 1$ to $DTEPSorted$ **do**
    if $ContrastRatioFinder(DTEPSorted[j], DTEPSorted[0]) < 4.5$ **then** ▷ Taking care of Fig. 3.12
      $Constant \leftarrow 4.5 * (DTEPLuminance[j] + 0.05) - 0.05$
      if $Constant > 1$ **then**
        $index \leftarrow -1$
        $Mapping[DTEPSorted[j]] \leftarrow [len(DTEPLuminance) - index - j - 1, (-1, -1)]$
      else
        $index \leftarrow findGe(DTEPLuminance[j + 1 : len(DTEPLuminance)], Constant)$
        $Mapping[DTEPSorted[j]] \leftarrow [len(DTEPLuminance) - index - j - 1, (index + j + 1, len(DTEPLuminance))]$
    else ▷ Taking care of Fig 3.10 and Fig 3.11
      $Constant \leftarrow (DTEPLuminance[j] + 0.05)/4.5 - 0.05$
      $index1 \leftarrow findGe(DTEPLuminance[0 : j], Constant)$
      $Constant \leftarrow 4.5 * (DTEPLuminance[j] + 0.05) - 0.05$

Algorithm 13 Find Mapping:continue

if $Constant > 1$ **then**

$index2 \leftarrow -1$

$Mapping[DTEPSorted[j]] \leftarrow [index1, (0, index1), (-1, -1)]$

else

$index2 \leftarrow findGe(DTEPLuminance[j + 1 : len(DTEPLuminance)], Constant)$

$Mapping[DTEPSorted[j]] \leftarrow [len(DTEPLuminance) - index2 - j - 1 + index1, (0, index1), (index2 + j + 1, len(DTEPLuminance))]$

return $Mapping$

procedure CONTRASTRATIOFINDER(COLOR1, COLOR2)

Can be implemented easily using the lightness and contrast formula described above.

procedure LUMINANCE(COLOR)

Can be implemented easily using the lightness formula described above.

procedure FINDGE(LIST, CONSTANT) ▷ Finds smallest item greater-than or equal to key. Raise ValueError if no such item exists. If multiple keys are equal, return the leftmost.

$i \leftarrow bisect.bisect_left(List, Constant)$

if $i == len(List)$ **then**

raise ValueError('No item found with Constant at or above)

return i

Algorithm 14 FBRecolor:Modified

1: **procedure** FINDRECOLOR(FG,BG)
2: $minDist = MAXINF$
3: **for** $j \leftarrow 1$ to $DTEP$ **do**
4: **if** $distance(bg, j) < minDist$ **then**
5: $minDist = distance(bg, DTEP[j])$
6: **for** $j \leftarrow 1$ to $DTEP$ **do**
7: **if** $distance(bg, j) < minDist + 0.5$ **then**
8: $minDistCircle.append(distance(bg, DTEP[j]))$
9: $initialDifferentiability \leftarrow distance(fg, bg)$
10: $maxOptions \leftarrow -1000$
11: **for** $j \leftarrow 1$ to $minDistCircle$ **do**
12: **if** $mapping[j].length > maxOptions$ **then** ▷ Mapping imported from Algorithm 13
13: $maxOptions \leftarrow mapping[j].length$
14: $MaxMinDist \leftarrow j$
15: $a \leftarrow 1$
16: $possibleSet \leftarrow 0$
17: **while** $possibleSet.length <= 0$ **do**
18: **for** $i \leftarrow 1$ to $mapping[j]$ **do**
19: **if** $(distance(minDistCircle[0], mapping[j][i]) > initialDiffrentiablity - a * 0.1 and distance(minDistCircle[0], mapping[j][i]) < initialDiffrentiablity + a * 0.1)$: **then**
20: $possibeSet.append(mapping[j][i])$
21: $a = a * 2$
22: $minDist \leftarrow MAXINF$

Algorithm 15 FBRecolor:Modified

23: **for** $j \leftarrow 1$ to $possibleSet$ **do**

24: **if** $distance(fg, j) < minDist + 0.5$ **then**

25: $minDist = distance(fg, possibleSet[j])$

26: $replacementFg = possibleSet[j]$

27: **return** $replacementFg, minDistCircle[0]$

Chapter 4

EVALUATION

To evaluate the approaches developed, we processed web pages using Kuhn's tool, SPRWeb and our tool and then compared the web pages on factors such as naturalness, differentiability, subjective naturalness and subjective differentiability.

4.1 Improvements in SPRWeb

We compared our tool with Kuhns[8] re-coloring tool. Metrics of comparison were perceptual naturalness, perceptual differentiability, subjective naturalness and subjective differentiability as defined in definitions section in Chapter 2.

First we converted the output colors obtained from both kuhn's[8] and our tool to CIELAB notation. To assess naturalness-preservation, we compared the L*a*b* Euclidean distance between each original color and its respective Kuhn[8] and our replacement colors. Lower distances represent better naturalness-preservation, i.e., the replacement color is more perceptually-similar to the original color.To test perceptual differentiability, we calculated the Euclidean distance between each pair of colors in each original website and determined the absolute difference between this distance and the equivalent color pair in the Kuhn[8] and our versions. Lower values represent better differentiability restoration, i.e., the replacement colors are as differentiable (not more or less differentiable) as the

Table 4.1: Evaluation: Improvements in SPRWeb (numbers in scaled CIELAB euclidean distance)

Web Page	N-k	N-t	D-k	D-t	SN-k	SN-t	SD-k	SD-t
Complex 1	34	16	18	2.64	0.87	0.5	0.05	0.2
Complex 2	23	10	12	2.11	0.84	0.6	0.02	0.3
Complex 3	30	11	14	0.7	0.88	0.4	0.05	0.1
Complex 4	50	3	50	0.5	4	0.08	3.5	0.02
Complex 5	50	16	50	4.5	4	0.6	1.3	0.2
Complex 6	33	8	20	0.3	1.23	1.04	0.14	0.3
Complex 7	47	16	17	2.15	0.87	1.01	0.018	0.4
Complex 8	50	27	20	4.25	1.73	1.5	1.37	0.4
Complex 9	40	21	23	4	0.99	1.1	0.1	0.5
Complex 10	50	21	50	2.6	3.07	1.1	4	0.3
Complex 11	18	6	15	1.6	0.78	0.4	0.05	0.2
Complex 12	44	17	17	3.8	1.29	0.6	0.02	0.2
Complex 13	27	12	23	1.7	0.8	0.7	0.15	0.3
Complex 14	14	4	10	0.5	0.3	0.1	0.104	0.1

k Kuhn's Tool

t Our tool developed by improvements in SPRWeb

N Naturalness

D Differentiability

SN Subjective Naturalness

SD Subjective Differentiability

original colors. Fig 4.1 gives a better view of comparison of the two tools in perceptual factors.

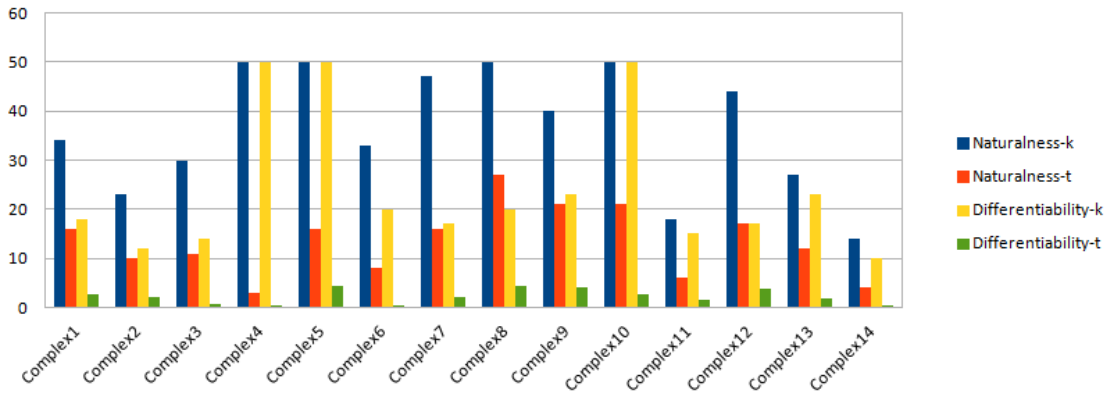


FIG. 4.1: Perceptual factors comparison

For theoretical subjective response, we followed the same procedure as in the case of perceptual factors. The major difference being that instead of using L^*,a^* and b^* as three dimensions, we used temperature, activity and weight as three dimensions to represent colors (as defined in chapter 2). For each of the color in original CSS, kuhn's[8] solution and our solution, we calculated the temperature, activity and weight factors. Then these three values are taken as one coordinate and same process as followed in perceptual factors was followed. Fig 4.2 compares kuhn's tool and our tool in subjectivity measures.

Based on this theoretical analysis, we show that our tool brings time complexity improvements in SPRWeb (derived theoretically), without a decline in its performance quantitatively.

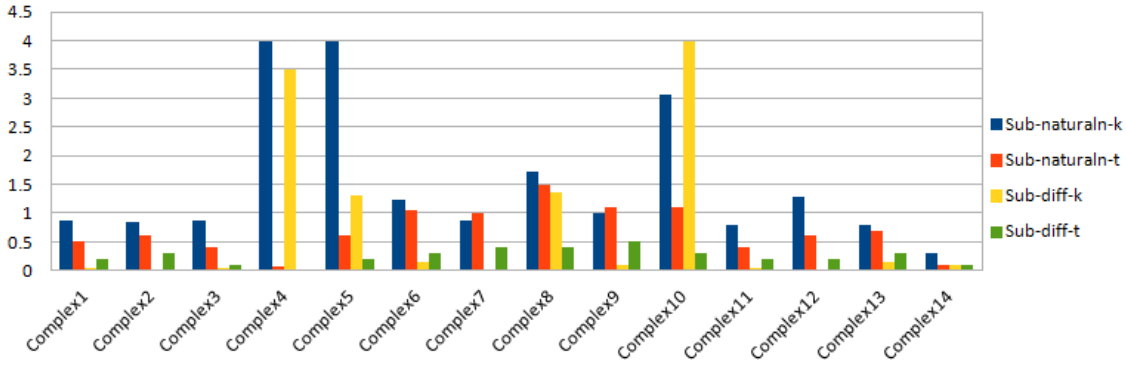


FIG. 4.2: Subjective factors comparison

4.2 FBRecolor

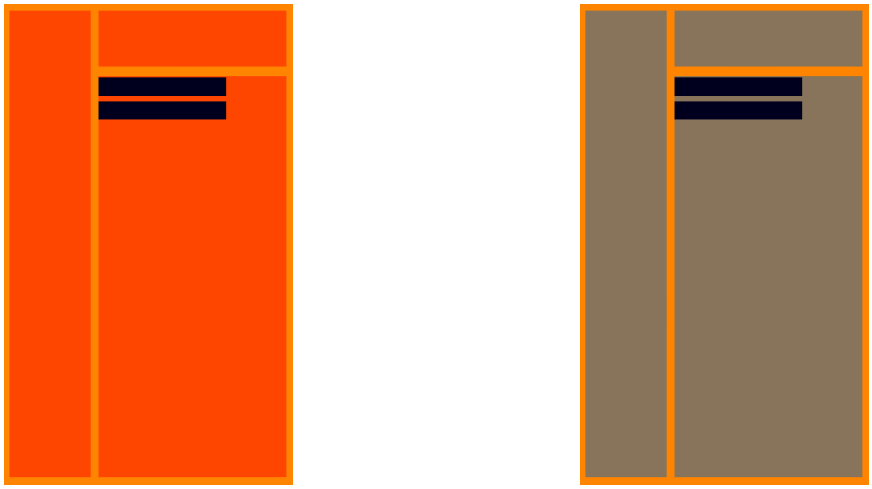
To evaluate our approach mentioned in section 3.2, we did analysis with two types of web pages - simple web pages and complex web pages.

4.2.1 Simple web pages

We generated simple web pages(Fig 4.3), having 3 colors in its layout with no text or images. Eight color schemes were used to generate 8 different web pages with similar structure. To test the subjectivity response preservation of tools being compared, we chose colors which were situated at extremities of Ou's subjective response model[]. As defined in chapter 2, colors are represented in three subjective response dimensions - activity, temperature and weight. And 8 color schemes were designed with colors at the extremities of subjective response dimensions.

Our tool and SPRWeb were compared this time on same metrics that were followed in section 4.1 and using the same method. Table 4.2 shows the testing results for simple web pages.

A comparison graph is given in Fig 4.4 (perceptual) and Fig 4.5(subjective).



(a) Simple 1 (b) Simple 2

FIG. 4.3: Simple web pages

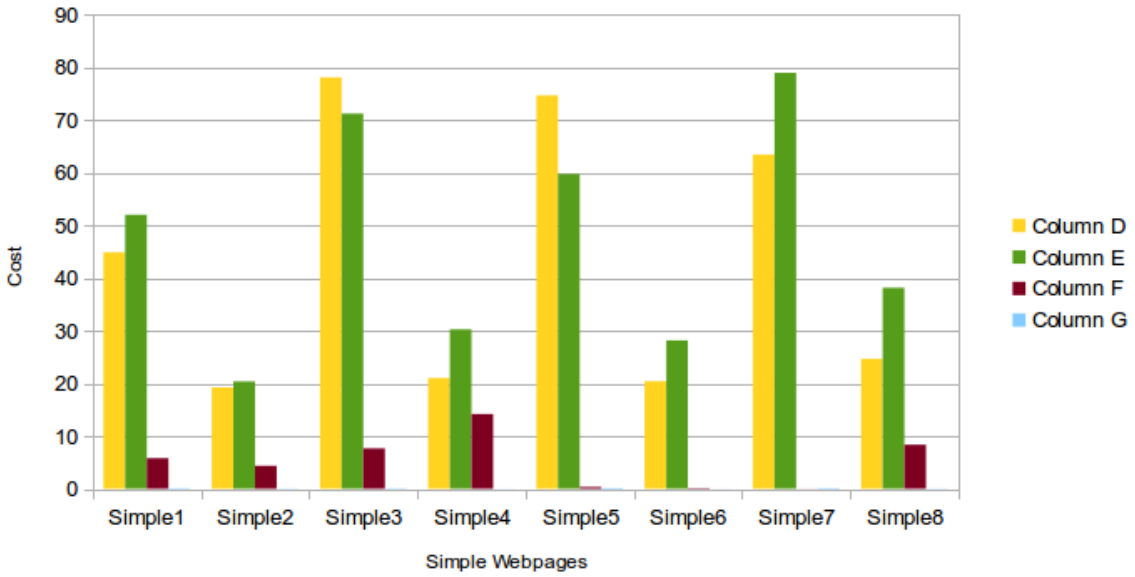


FIG. 4.4: Perceptual comparison

We also randomly generated 10 simple web pages to compare our tool with SPRWeb. The data is given in Table 4.3.

Table 4.2: Evaluation: FBRecolor (numbers in scaled CIELAB euclidean distance)

Web Page	C#	P#	N-S	N-t	D-S	D-t	SN-S	SN-t	SD-S	SD-t
Simple 1	3	2	44.89	52.02	5.78	0.11	0.06	0.29	2.84	3.07
Simple 2	3	2	19.23	20.37	4.35	0.04	1.02	0.63	1.74	1.82
Simple 3	3	2	78.11	71.24	7.66	0.09	1.08	0.76	2.03	2.54
Simple 4	3	2	21.03	30.27	14.19	0.006	1.87	1.92	0.95	1.25
Simple 5	3	2	74.7	59.76	0.33	0.15	1.1	2.05	3.38	3.26
Simple 6	3	2	20.41	28.16	0.11	0.009	1.43	1.66	1.8	1.94
Simple 7	3	2	63.46	78.98	0.003	0.13	2.31	2.52	2.57	2.73
Simple 8	3	2	24.67	38.17	8.35	0.04	2.44	2.95	1.29	1.37

S	SPRWeb
t	Our tool developed on parsing pairs of colors concept
N	Naturalness
D	Differentiability
SN	Subjective Naturalness
SD	Subjective Differentiability

A comparison plot is given in Fig 4.6 (perceptual) and Fig 4.7 (Subjective).

Comparison of total cost can be seen in Fig 4.8 and Fig 4.9 for general and randomly generated web pages respectively.

It can be observed from the comparison that our tool performs better in maintaining perceptual differentiability in every case. The overall cost of our tool is less than that of SPRWeb in 12 out of 18 simple web pages. In 2 pages the cost is marginally greater than SPRWeb. It can also be observed that perceptual naturalness cost is significantly higher in our tool as

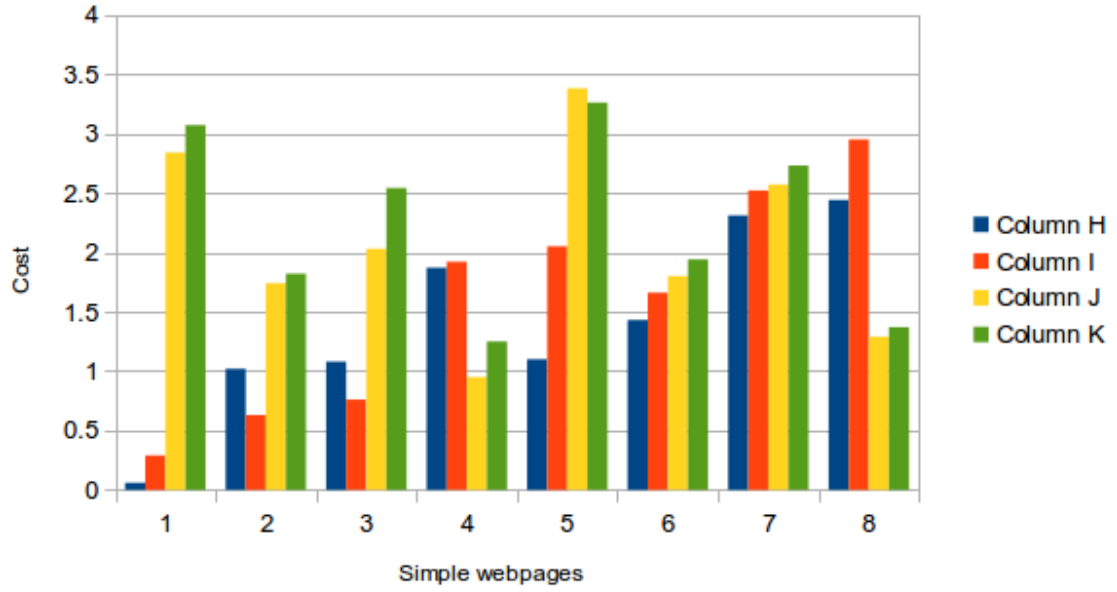


FIG. 4.5: Subjective comparison

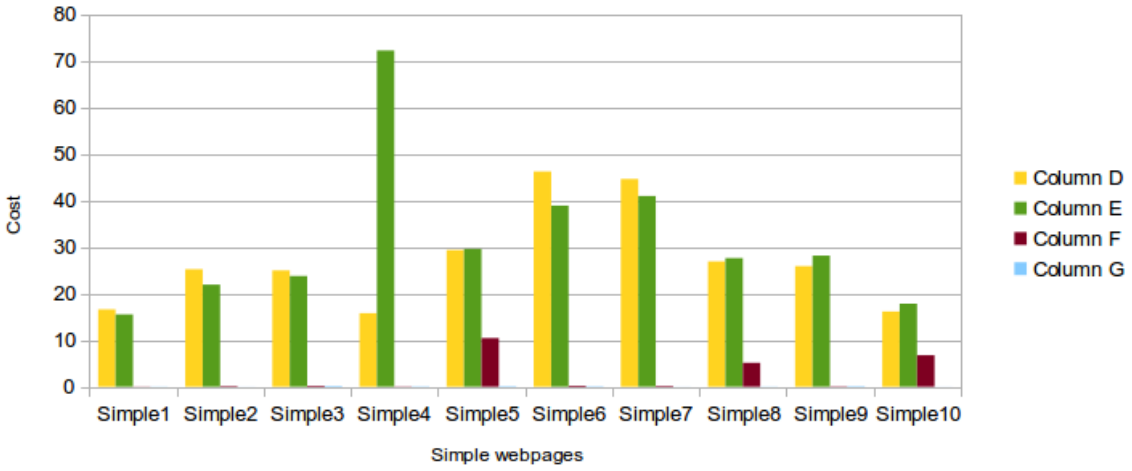


FIG. 4.6: Perceptual comparison

compared to SPRWeb. This can be attributed to the step in the algorithm where we restrict our possible replacement set in order to preserve perceptual differentiability.

Table 4.3: Evaluation: FBRecolor (numbers in scaled CIELAB euclidean distance)

Web Page	C#	P#	N-S	N-t	D-S	D-t	SN-S	SN-t	SD-S	SD-t
Simple 1	3	2	16.6	15.51	0.04	0.04	1.77	1.27	0.83	0.74
Simple 2	3	2	25.2	21.91	0.15	0.02	0.42	0.36	0.05	0.62
Simple 3	3	2	24.96	23.76	0.2	0.18	0.24	0.39	0.75	0.65
Simple 4	3	2	15.78	72.16	0.06	0.08	0.08	2.05	0.75	1.6
Simple 5	3	2	29.33	29.58	10.43	0.14	0.08	0.36	0.92	0.95
Simple 6	3	2	46.21	38.87	0.23	0.11	1.26	2.1	1.48	1.73
Simple 7	3	2	44.58	40.93	0.18	0.02	0.05	1.12	1.47	1.71
Simple 8	3	2	26.89	27.62	5.11	0.04	0.52	0.57	0.9	0.95
Simple 9	3	2	25.87	28.14	0.08	0.14	1.41	0.5	0.77	0.96
Simple 10	3	2	16.14	17.81	6.76	0.02	0.4	0.33	0.54	0.63

S	SPRWeb
t	Our tool developed on parsing pairs of colors concept
N	Naturalness
D	Differentiability
SN	Subjective Naturalness
SD	Subjective Differentiability

4.2.2 Complex web pages

In order to test our tool on real web pages, we selected 14 complex web pages. These web pages consisted of information style pages, forum-style pages and blog style pages. The evaluation data corresponding to these web pages is shown in Table 4.4. To test our tool further, we selected 18 more random complex pages and did analysis. Data corresponding

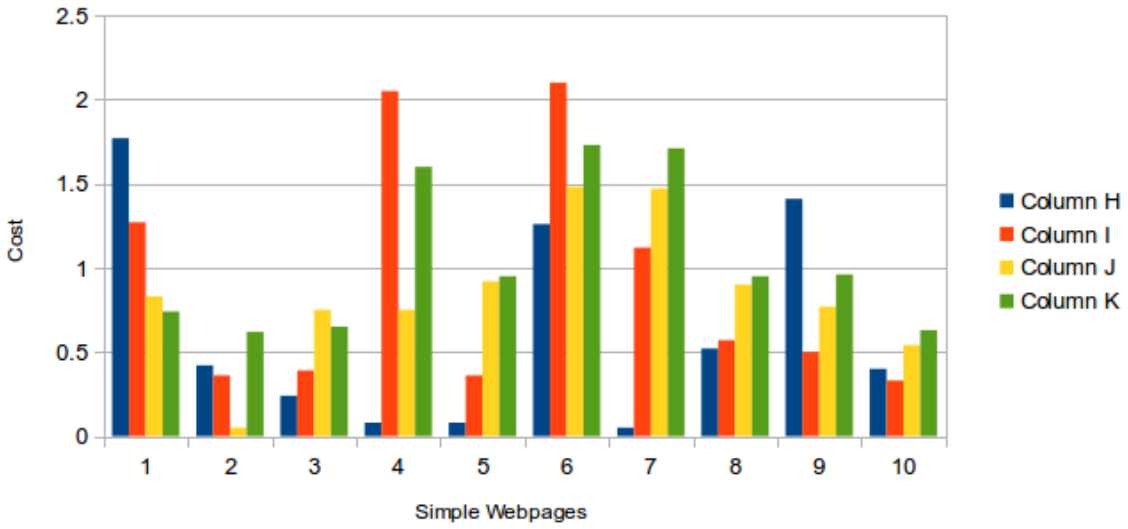


FIG. 4.7: Subjective comparison

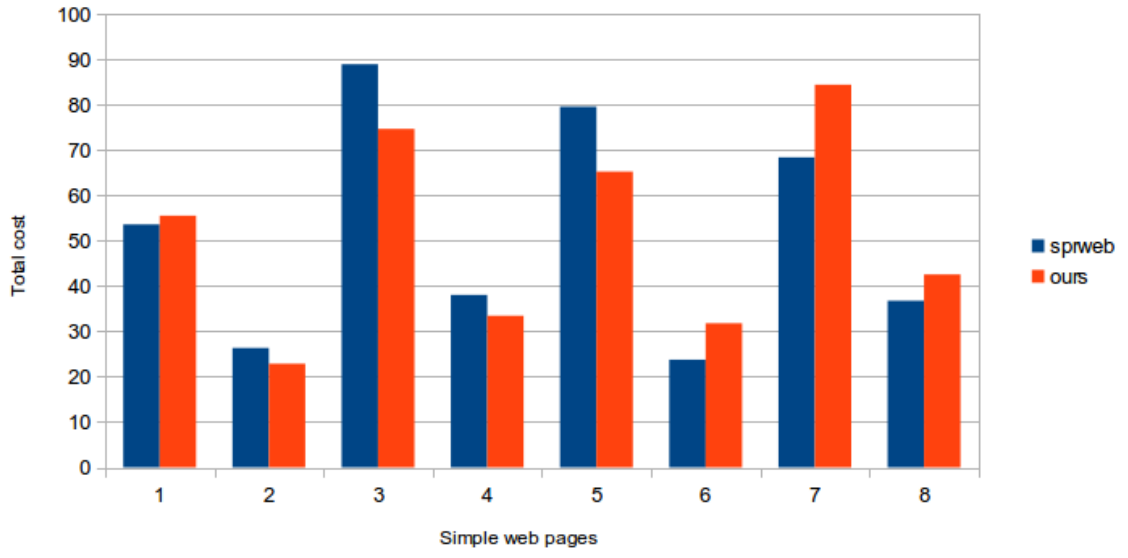


FIG. 4.8: Total cost comparison for general pages

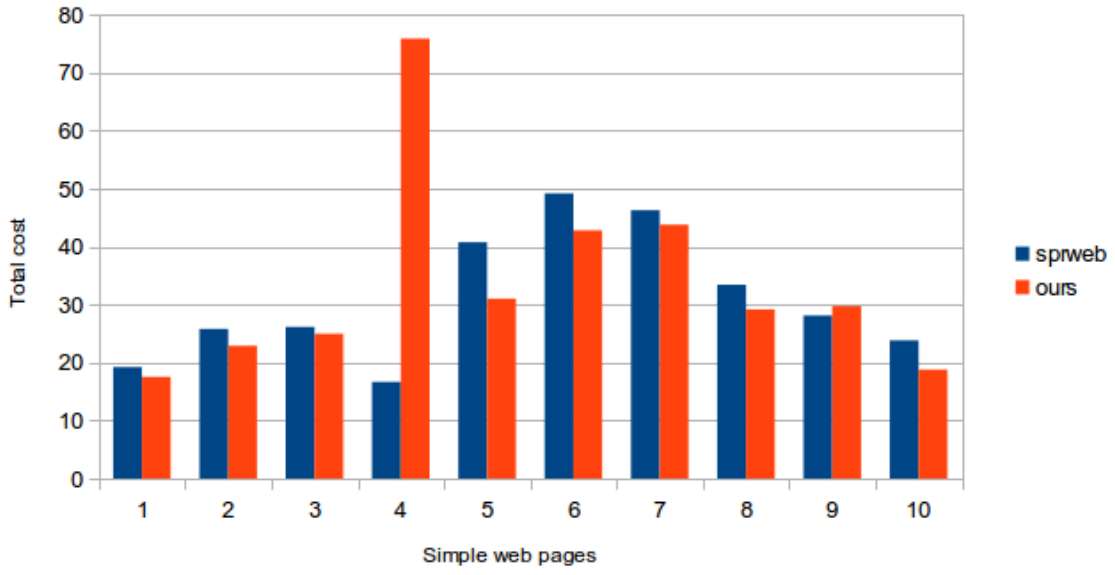


FIG. 4.9: Total cost comparison for random pages

to these 18 web pages is shown in Table 4.5.

The comparison graphs are provided in Fig 4.10 (general perceptual), Fig 4.11 (general subjective), Fig 4.12 (random perceptual) and Fig 4.13 (random subjective).

Comparison of total cost for both general and random complex pages is provided in Fig 4.14 and Fig 4.15 respectively.

By analyzing the total cost comparison for complex web pages, we found that out of 32 pages, our tool performs better in 25 pages. While performs slightly poorer than SPRWeb in 4 pages. It can be observed that while the perceptual differentiability is better in every case for our tool and naturalness suffers.

Table 4.4: Evaluation: FBRecolor (numbers in scaled CIELAB euclidean distance)

Web Page	C#	P#	N-S	N-t	D-S	D-t	SN-S	SN-t	SD-S	SD-t
Complex 1	12	12	29.05	29.05	8.22	0.05	1.08	2.11	1.21	1.26
Complex 2	12	9	10.28	11.18	5.08	0.11	1.29	0.94	0.73	0.72
Complex 3	4	3	9.15	7.77	0.24	0.1	0.41	0.65	0.28	0.32
Complex 4	6	4	7.88	8.66	5.12	0.02	0.31	0.11	0.3	0.39
Complex 5	6	4	10.78	10.49	0.079	0.015	0.35	0.71	0.28	0.34
Complex 6	7	4	10.19	9.73	2.63	0.05	1.38	2.16	0.92	0.92
Complex 7	22	22	21.57	23.99	67.46	0.01	7.35	8.89	1.39	1.48
Complex 8	5	4	34.81	46.3	22.7	0.02	1.1	2.32	1.85	2.5
Complex 9	7	5	32.5	34.83	28.22	0.08	2.25	0.705	1.57	1.61
Complex 10	6	5	13.09	17.29	17.25	0.05	1.68	0.54	0.71	0.649
Complex 11	15	15	25.12	27.79	9.88	0.61	0.87	1.15	3.28	4.52
Complex 12	13	9	24.45	30.98	15.38	0.24	0.8	1.1	0.9	0.4
Complex 13	9	7	13.56	21.33	1.62	0.002	0.77	1.1	1.21	0.62
Complex 14	13	10	16.76	17.6	41.62	0.09	0.78	0.82	0.82	0.06

S	SPRWeb
t	Our tool developed on parsing pairs of colors concept
N	Naturalness
D	Differentiability
SN	Subjective Naturalness
SD	Subjective Differentiability

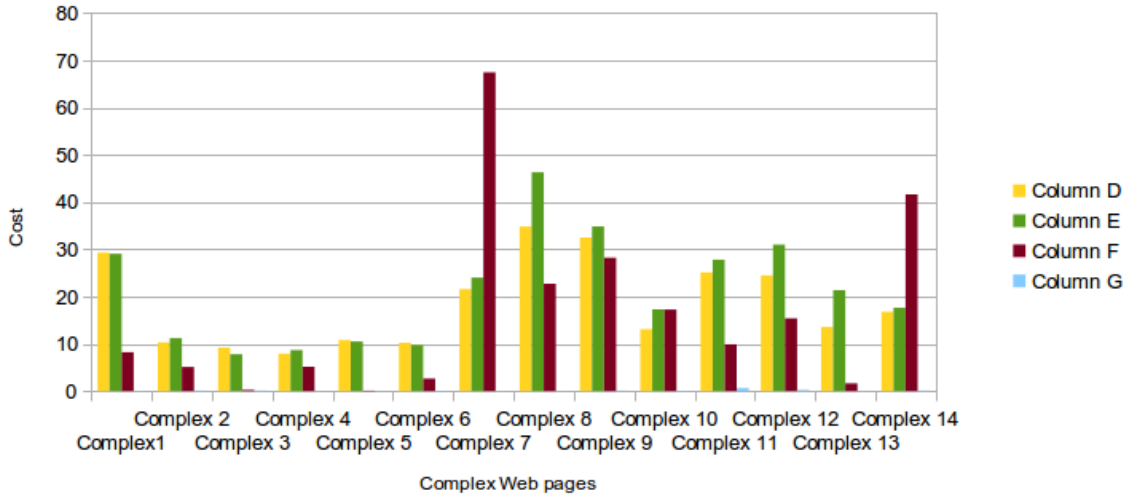


FIG. 4.10: Perceptual comparison for general complex pages

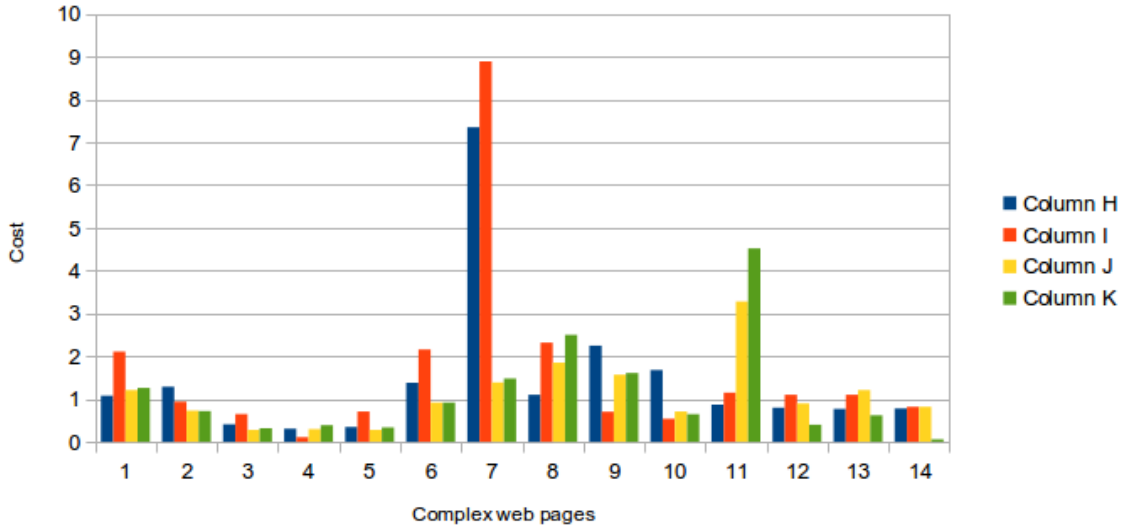


FIG. 4.11: Subjective comparison for general complex pages

4.3 Minimum Contrast

To evaluate the minimum contrast algorithm that we suggested, we took some simple web pages and recolored them using our tool and SPRWeb.

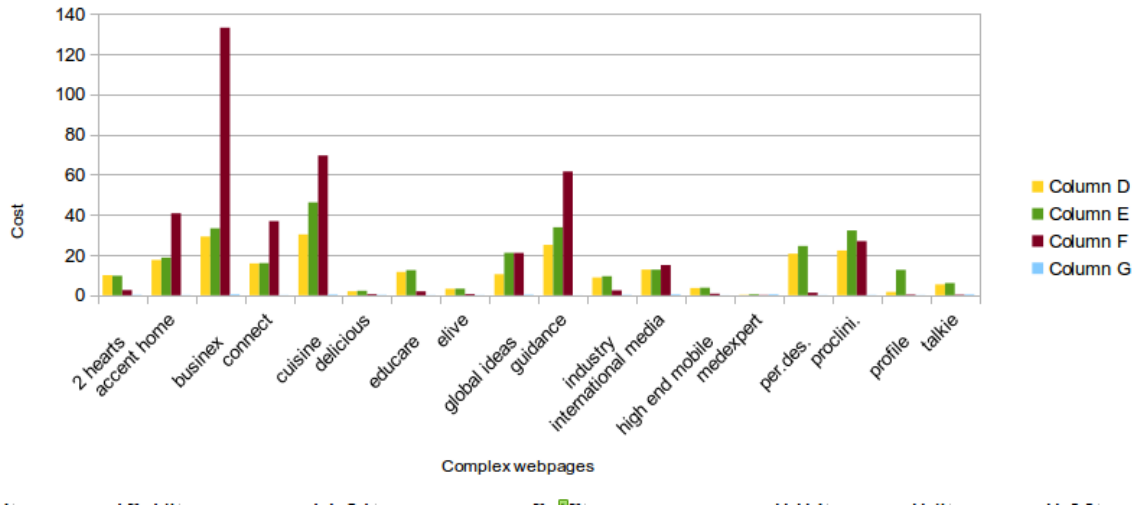


FIG. 4.12: Perceptual comparison for random complex pages

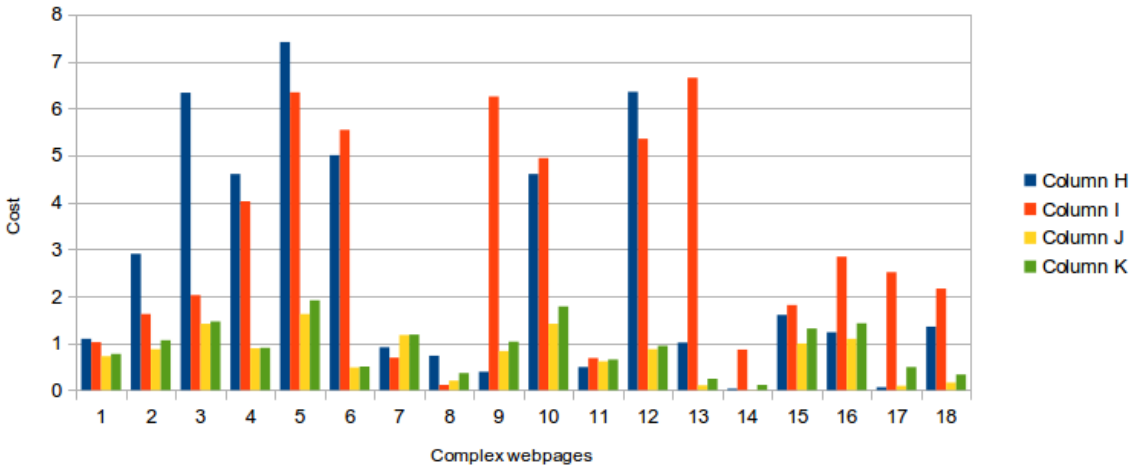


FIG. 4.13: Subjective comparison for random complex pages

Fig 4.16 shows the the simple web pages that were used to compare two tools and Fig 4.17, 4.18 shows the output of the two tools.

Results show that our tool successfully induces contrast in the color pairs. Though the costs corresponding to other factors worsens but this version of tool is able to make sure

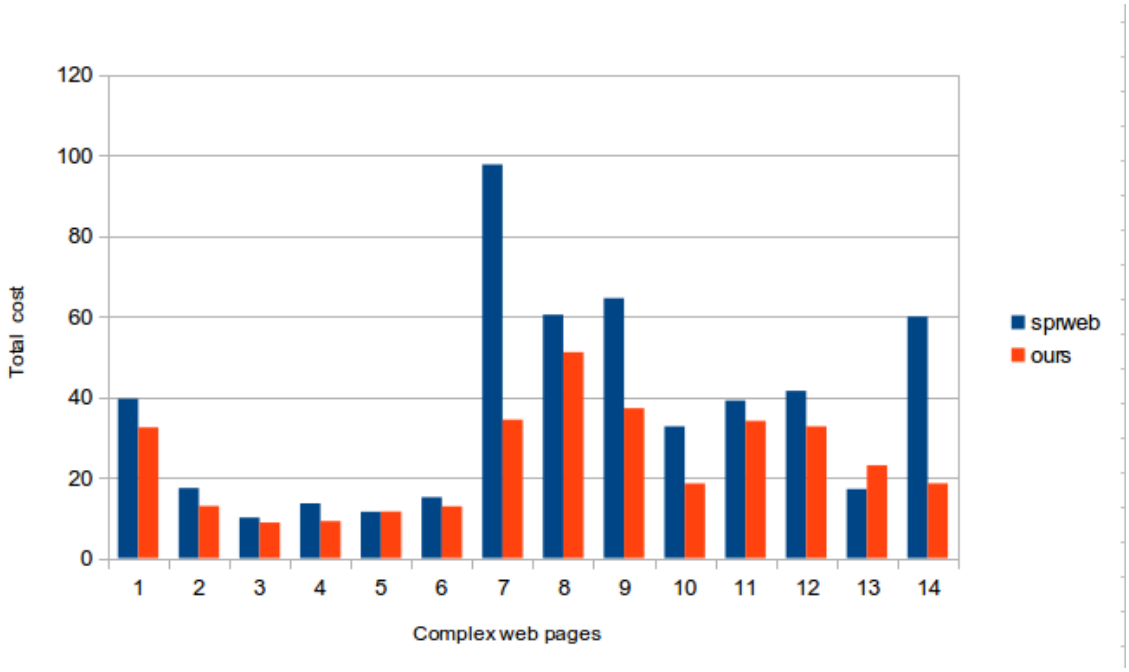


FIG. 4.14: Total cost comparison for general complex pages

that the W3C guideline of contrast threshold is maintained. In simple web pages, there are two pairs of colors - (top,background) and (middle, background). In recolored versions of simple 1, we can observe that the contrast in the pair (top,background) pair is better in our version than SPRWeb. There is no need to enhance contrast in top and middle as a pair as they don't exist as a pair on the web page.

Similarly in simple 2, the contrast in (middle,background) is better than that of SPRWeb. All these observations are also supported by numerical data present in Table 4.6.

We can observe that cost corresponding to all the factors is increasing. Perceptual differentiability, particularly, suffers a lot as compared to other factors. It can attributed to the fact that in our three step algorithm, we are reducing the size of search space for background color. Search space is being reduced because we are taking only those colors which have a contrast ratio of greater than 4.5 with foreground color.

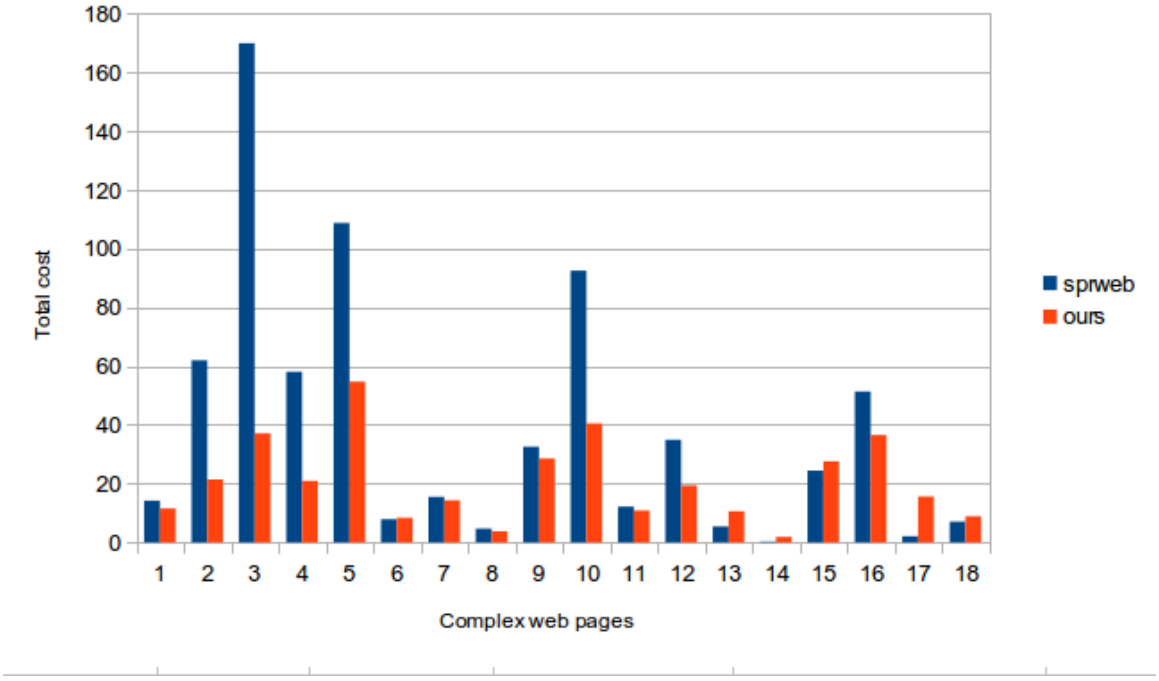
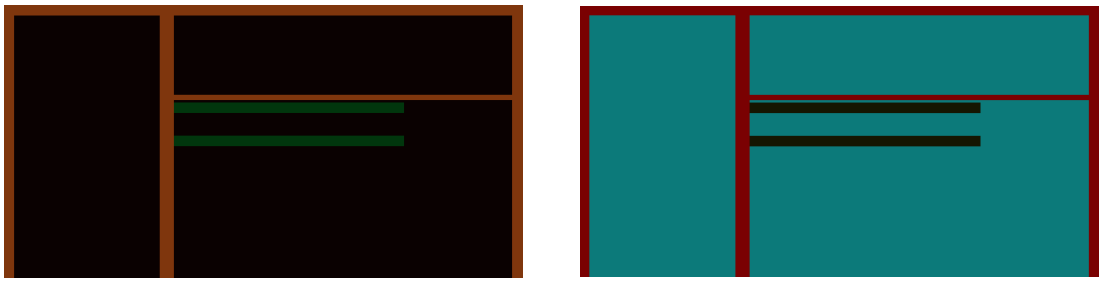


FIG. 4.15: Total cost comparison for random complex pages



(a) Simple 1

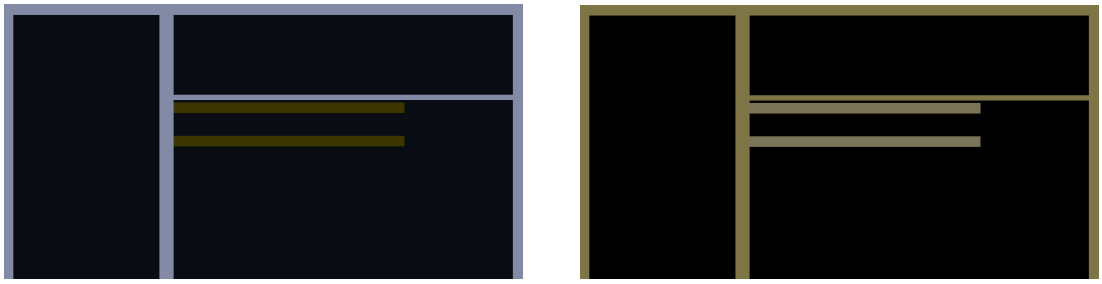
(b) Simple 2

FIG. 4.16: Simple web pages for testing

Table 4.5: Evaluation: FBRecolor (numbers in scaled CIELAB euclidean distance)

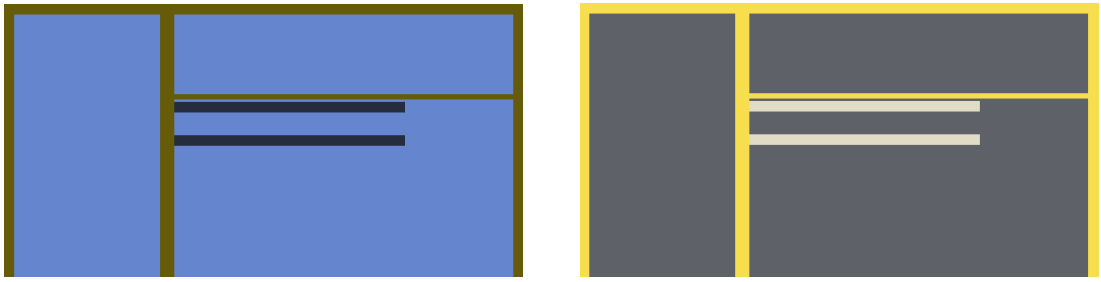
Web Page	C#	P#	N-S	N-t	D-S	D-t	SN-S	SN-t	SD-S	SD-t
Complex 1	5	3	9.92	9.65	2.49	0.14	1.09	1.02	0.72	0.77
Complex 2	10	9	17.51	18.68	40.7	0.08	2.9	1.62	0.87	1.06
Complex 3	12	11	29.15	33.21	133.05	0.44	6.33	2.02	1.41	1.46
Complex 4	16	17	15.79	15.95	36.8	0.06	4.6	4.02	0.89	0.9
Complex 5	8	7	30.22	46.14	69.48	0.35	7.41	6.34	1.62	1.91
Complex 6	11	9	1.92	2.19	0.48	0.14	5	5.54	0.48	0.5
Complex 7	15	8	11.57	12.44	1.89	0.03	0.91	0.69	1.17	1.18
Complex 8	11	12	3.2	3.26	0.56	0.05	0.73	0.11	0.2	0.36
Complex 9	8	10	10.39	21.05	20.98	0.24	0.39	6.25	0.83	1.03
Complex 10	15	11	24.99	33.76	61.53	0.01	4.6	4.94	1.41	1.78
Complex 11	8	9	8.76	9.38	2.34	0.15	0.49	0.68	0.61	0.65
Complex 12	16	17	12.73	12.69	14.98	0.4	6.35	5.35	0.87	0.94
Complex 13	13	15	3.57	3.7	0.71	0.005	1.01	6.65	0.1	0.24
Complex 14	16	12	0.14	0.41	0.089	0.45	0.029	0.86	0.003	0.11
Complex 15	10	9	20.62	24.46	1.21	0.03	1.6	1.81	0.99	1.31
Complex 16	16	10	22.21	32.2	26.85	0.1	1.23	2.84	1.09	1.42
Complex 17	3	3	1.61	12.57	0.32	0.042	0.06	2.51	0.09	0.49
Complex 18	16	11	5.33	5.99	0.24	0.4	1.35	2.16	0.16	0.33

S	SPRWeb
t	Our tool developed on parsing pairs of colors concept
N	Naturalness
D	Differentiability
SN	Subjective Naturalness
SD	Subjective Differentiability



(a) Recolored by SPRWeb (b) Recolored by our tool

FIG. 4.17: Recoloring simple web page 1(notice the contrast between top bar and background)



(a) Recolored by SPRWeb (b) Recolored by our tool

FIG. 4.18: Recoloring simple web page 2(notice the contrast between middle bars and the background)

Table 4.6: Evaluation: FBRecolor (numbers in scaled CIELAB euclidean distance)

Page	CRP1S	CRP1O	CRP2S	CRP2O	N-S	N-t	D-S	D-t	SN-S	SN-t	SD-S	SD-t
1	5.7	4.5	3.6	4.5	29.1	26.6	0.1	12.4	0.6	1.9	1.3	4.7
2	1.9	4.6	3.9	4.5	43	66.5	0.1	12.4	0.3	2.9	1.7	2.3
3	2.5	4.5	7	18.2	30.6	33.5	3.6	14	0.3	1	0.9	0.8
4	9	9.6	5.4	4.5	15.8	26.4	0.9	30.9	0.1	1.1	0.5	1.8

CRP1S	Contrast ratio in pair 1 for SPRWeb version
CRP1O	Contrast ratio in pair 1 for our version
N	Naturalness
D	Differentiability
SN	Subjective Naturalness
SD	Subjective Differentiability

Table 4.7: Evaluation: FBRecolor extension(numbers in scaled CIELAB euclidean distance)

Web Page	C#	P#	FP#S	FP#t	N-S	N-t	D-S	D-t	SN-S	SN-t	SD-S	SD-
Complex 1	5	3	2	0	38.7	22.59	2.49	38.79	1.09	2.91	0.72	1.1
Complex 2	10	9	5	0	31	22.7	40.7	31	2.9	3.45	0.87	0.9
Complex 3	12	11	3	0	44.2	56.7	133.05	44.21	6.33	6.63	1.41	2.1
Complex 4	16	17	9	0	51.8	22.2	36.8	51.8	4.6	8.49	0.89	1.1
Complex 5	8	7	5	0	0.5	52.68	69.48	0.54	7.41	3.09	1.62	1.7
Complex 6	11	9	4	0	34.1	14.82	0.48	34.14	5	6.77	0.48	1.1
Complex 7	15	8	7	0	38.9	23.14	1.89	38.91	0.91	3.77	1.17	1.4
Complex 8	11	12	3	0	47.9	7.19	0.56	47.94	0.73	4.88	0.2	0.3
Complex 9	8	10	6	0	0.7	11.01	20.98	0.79	0.39	4.37	0.83	0.8
Complex 10	15	11	7	0	0.6	38.12	61.53	0.69	4.6	8.63	1.41	1.6
Complex 11	8	9	2	0	27.6	19.38	2.34	27.61	0.49	4.45	0.61	1.1
Complex 12	16	17	14	0	142.6	36.64	14.98	142.64	6.35	15.5	0.87	1.7
Complex 13	13	15	4	0	17.9	14.27	0.71	17.93	1.01	7.35	0.1	0.6
Complex 14	16	12	6	0	153.4	10.6	0.089	153.41	0.029	9.95	0.003	0.5
Complex 15	10	9	7	0	50.7	28.06	1.21	50.73	1.6	6.79	0.99	1.4
Complex 16	16	10	2	0	52.1	54.79	26.85	52.11	1.23	8.39	1.09	1.5
Complex 17	3	3	1	0	14.6	5.34	0.32	14.66	0.06	1.28	0.09	0.4
Complex 18	16	11	8	0	147.2	16.6	0.24	147.29	1.35	13.5	0.16	0.9

FP#S#	number of pairs having contrast less than 4.5 in SPRWeb recolored version
FP#t#	number of pairs having contrast less than 4.5 in FBRecolor recolored version
S	SPRWeb
t	Our tool developed on parsing pairs of colors concept
N	Naturalness
D	Differentiability
SN	Subjective Naturalness
SD	Subjective Differentiability

Table 4.8: Evaluation: FBRecolor extension(numbers in scaled CIELAB euclidean distance)

Web Page	C#	P#	FP#S	FP#t	N-S	N-t	D-S	D-t	SN-S	SN-t	SD-S	SD-t
Complex 1	12	9	5	0	10.28	19.17	5.08	79.78	1.29	7.41	0.73	1.09
Complex 2	4	3	0	0	9.15	11.73	0.24	0.14	0.41	0.26	0.28	0.46
Complex 3	6	4	3	0	7.88	18.36	5.12	25.82	0.31	2.35	0.3	0.81
Complex 4	6	4	4	0	10.78	34.52	0.079	9.39	0.35	3.28	0.28	1.3
Complex 5	7	4	2	0	10.19	20.09	2.63	6.08	1.38	1.68	0.92	1.36
Complex 6	22	22	15	0	21.57	26.23	67.46	249.43	7.35	21.46	1.39	1.61
Complex 7	5	4	2	0	34.81	56.56	22.7	25.31	1.1	2.32	1.85	3.07
Complex 8	7	5	5	0	32.5	42.54	28.22	0.47	2.25	2.69	1.57	2.13
Complex 9	6	5	4	0	13.09	13.75	17.25	67.09	1.68	3.5	0.71	0.84
Complex 10	13	9	1	0	25.12	56	9.88	20.44	0.87	3.16	3.28	1.71
Complex 11	9	7	4	0	13.56	26.2	1.62	49.14	0.77	6.65	1.21	1.42
Complex 12	13	10	9	0	16.76	40.32	41.62	141.63	0.78	8.78	0.82	1.66

FP#S#	number of pairs having contrast less than 4.5 in SPRWeb recolored version
FP#t#	number of pairs having contrast less than 4.5 in FBRecolor recolored version
S	SPRWeb
t	Our tool developed on parsing pairs of colors concept
N	Naturalness
D	Differentiability
SN	Subjective Naturalness
SD	Subjective Differentiability

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Some limitations and possible ways to handle them

- Current method involves the optimization function, where equal weights are given to all the four factors. But as observed, values of perceptual factors are around 20 times greater than the value of subjective response factors. A little gain in cost corresponding to perceptual factors can overshadow the significant loss in subjective response factors.

  Addressing this problem depends on the user. If the person, trying to recolor web pages, wants to give more preference to subjective factors than the perceptual factors, then it can provide more weight to subjective response factors. Or in our case, it can search for solutions in subjective response space, preserving the subjective factors.

- Current method does significantly better than SPRweb in preserving differentiability across foreground-background color pairs. But suffers in preserving naturalness as compared to SPRWeb.

  One possible way to preserve naturalness could be to find replacements of both foreground and background color such that the replacement are as close to the originals as possible. But as tried and tested, this implementation results in lower preservation

of differentiability. An approach to optimize both naturalness and differentiability could be to try to find a replacement for background, in the third step, such that the summation of cost (naturalness and differentiability) is as minimum as possible.In FBRecolor, in the first step we find fg as a replacement of fg such that the distance between fg and fg is as minimum as possible. In the second step, we find a possible set for replacement of bg such that the differentiability is maintained. And in the final step we chose a replacement bg such that it is as close to bg as possible. In our new possibility, to improve naturalness and differentiability, we can include both the factors as constraint in third step of the algorithm to find a replacement for bg.

- In the modified algorithm, where we include the minimum contrast threshold, the cost corresponding to naturalness and differentiability increases as compared to the non-modified algorithm. In our modified algorithm, we use a mapping corresponding to a replacement fg of fg such that the elements of mapping have a contrast ratio of more than 4.5 with fg. In this we can modify the criteria to find the mapping, by including some distance metrics. And while finding replacements we can include those distance metrics in our equation. The distance metric could be something which could give a sense of location the point being considered. For example, consider a color fg, we create a mapping such that the set has all the colors which when paired with fg give a contrast of more than 4.5. The set that we develop can have one more characteristic per color. That characteristic can include some location sense of that color with respect to some point such as origin(L*=0, a=0, b=0). We can use this location information in our optimization equation such that we can choose points which have better cost values. How to use that location information is still a matter of discussion.

5.2 Future Work

To enable CVD users to have a similar web experience as normal users, we proposed an algorithm based on pair-wised parsing of colors from the web page. We also present a modification to our algorithm to ensure that the replacement color pairs have at least a contrast ratio of 4.5:1 as suggested by W3C guidelines.

In future, we plan to extend our tool to include image recoloring as well and develop a browser plug in for it. To deal with images, we will have to incorporate image processing techniques in our current tool. In our algorithm, we parse all the color pairs and find their replacement one by one. Instead, we can explore parallel computing concepts to save us some computation time. Saving computational time would positively support our algorithm in being a browser plug in. We can also extend our modification of algorithm, where we ensure the contrast of 4.5, to normal users. For example, if there are some web pages which have bad contrast in foreground-background color pairs, then our tool would be able to detect and correct the same.

Chapter 6

REFERENCES

- [1] - Wong, B (June 2011). ”Color blindness”. Nat. Methods 8 (6): 441.doi:10.1038/nmeth.1618. PMID 21774112.
- [2] - Cole, B. L. The Handicap of Abnormal Colour Vision. Clinical and Experimental Optometry 87, 4-5 (2004), 258275.
- [3] - Bonnardela, N., Piolata, A., and Bigot, L. L. The Impact of Colour on Website Appeal and Users Cognitive Processes. Displays 32, 2 (2011), 6980.
- [4] - Cyr, D., Head, M., and Larios, H. Colour Appeal in Website Design Within and Across Cultures: A Multi-Method Evaluation. IJHCI 68, 1(2010), 121.
- [5] - http://www.w3.org/TR/2014/NOTE-WCAG20-TECHS-20140408/G18
- [6] - SPRWeb: Preserving Subjective Responses to Website Colour Schemes through Automatic Recolouring. David R. Flatla1 , Katharina Reinecke2 , Carl Gutwin1 and Krzysztof Z. Gajos2
- [7] - Ichikawa, M., Tanaka, K., Kondo, S., Hiroshima, K., Ichikawa, K.,Tanabe, S., and Fukami, K. Web-Page Color Modification for Barrier-Free Color Vision with Genetic Algorithm. In Proc. GECCO (2003), 21342146.

- [8] - ColorBlindnessSimulateCorrect by seewald solutions.
- [9] - www.daltonize.org
- [10] - Kuhn, G., Oliveira, M., and Fernandes, L. An Efficient Naturalness-Preserving Image-Recoloring Method for Dichromats. IEEE Trans. Vis. & Comp. Graph. 14, 6 (2008), 17471754.
- [11] - Ou, L., Luo, M., Woodcock, A., and Wright, A. A Study of Colour Emotion and Colour Preference. Part I: Colour Emotions for Single Colors. Color Research & Application 29, 3 (2004), 232240.
- [12] - www.visibone.com
- [13] - http://colorlab.wickline.org/colorblind/colorlab/
- [14] - S. Ishihara, Tests for color-blindness (Handaya, Tokyo, Hongo Harukicho, 1917).
- [15]- Hunter, Richard Sewall (July 1948). ”Photoelectric Color-Difference Meter”.JOSA 38 (7): 661. (Proceedings of the Winter Meeting of the Optical Society of America)
- [16] - Hunter, Richard Sewall (December 1948). ”Accuracy, Precision, and Stability of New Photo- electric Color-Difference Meter”. JOSA 38 (12): 1094. (Proceedings of the Thirty-Third Annual Meeting of the Optical Society of America)
- [17] - Meyer, G. W., and Greenberg, D. P. Color-Defective Vision and Computer Graphics Displays. IEEE CG&A 8, 5 (1988), 2840.
- [18] - ”Document Object Model (DOM)”. http://www.w3.org/: W3C. Retrieved 2012-01-12. ”The Document Object Model is a platform- and language-neutral in-

terface that will allow programs and scripts to dynamically access and update the content, structure and style of documents."

- [19] - http://en.wikipedia.org/wiki/GNOME
- [20] - http://www.colorhelper.com/
- [21] - Iaccarino, G., Malandrino, D., and Scarano, V. Personalizable Edge Services for Web Accessibility. In Proc. W4A (2006), 2332.

REFERENCES