Fall 2020

Testing and Improving an Optimization-Based Digital Colorblindness Corrective Filter

Zachary Kenneth McIntyre
Bard College

Testing and Improving an Optimization-Based Digital Colorblindness Corrective Filter

Senior Project Submitted to
The Division of Science, Math, and Computing
of Bard College

by

Zachary McIntyre

Annandale-on-Hudson, New York
December 2020

Acknowledgements

I would like to thank my advisor, Kerri-Ann Norton, for the guidance and mentorship she has given me during this project. I would also like to thank the wonderful faculty I've had the pleasure to work with at Bard: Keith O'Hara, Sven Anderson, and Nathan Shockey, all of whom have taught me so much. I would also like to thank my horn teachers, Julia Pilant, Barbara Jöstlein Curie and Jeff Lang for their support and patience as I completed this project. Thank you also to Glenn Wagner for encouraging me to pursue computer science when I was not confident in myself. Finally, I would like to thank my friends, family, and girlfriend for their unfaltering love and support.

Table of Contents

Abstract **–** Computers often communicate essential information via color which is lost to colorblind users. In order to address this information loss, designers and computer scientists have created a variety of different correction methods to improve computer accessibility. One such method was created by Luke Jefferson and Richard Harvey in their 2006 paper, "Accommodating Color Blind Computer Users" which consists of a difference histogram, differences of key colors, optimization and interpolation to adjust images for specific types of congenital colorblindness. I have recreated their algorithm as well as their original test images. I then conducted extensive tests on challenging images to examine the strengths and weaknesses of the algorithm. Finally, I further improved the algorithm in an effort to determine if this method might be useful for a real-time corrective filter.

Introduction

Section 1: Project Summary

As the world grows more aware of the issue of accessibility (or perhaps as the voices demanding equality of access grow louder) it has become apparent that nearly all of the common features of design are not accessible. When it comes to computer design, the user often relies on color to help them discern what links have been activated or whether something is green for “go” and “no problems here”, or red for “stop” and “we have an issue.” While these color indications work for some users, these cues may not be discernible for colorblind users.

For these colorblind users, there are a few solutions to this loss of color information. There is no shortage of augmented reality apps that can help the user see more clearly in real life. Of course, this does not solve the problems that users might encounter while browsing the web or playing a game on their smart phone or computer. Operating systems such as Windows often build in colorblind corrective filters for their users. However, this information is often difficult to find, especially for users who are not familiar with computers. For instance, there is no mention of a colorblindness corrective filter in the Windows 10 guide (Microsoft n.d.). There are also apps that users can install, such as Visolve, which can be downloaded and used as a colorblindness corrective filter or as a tool to change the saturation of the user’s screen, thus making certain colors more discernable (Visolve n.d.).

Another such filter was produced by Luke Jefferson and Richard Harvey in their papers “Accommodating Color Blind Computer Users” (2006) and “An Interface to Support Color Blind Computer Users” (2007). I intend to recreate the results produced by Jefferson and Harvey and then experiment with further testing. Then, I will explore different methods for improving

the algorithm. In addition to the papers mentioned above, I am basing my work largely on research done by Hans Brettel, Françoise Viénot and John D. Mollon who worked on a daltonization method of simulating colorblindness (1997) and Onur Fidaner, Poliang Lin and Nevran Ozguven who made a python daltonizer (2014).

Section 1: What is Colorblindness?

To understand what colorblindness is, we must first lay the groundwork for how the human eye works without defects. In a normal human eye, there are three different cone photopigments from each of the three spectral classes, which is to say that the eye contains three different cones whose purpose is to detect light of different wavelengths (Neitz et al. 1998, 101). One cone is sensitive to long wavelengths, which are red colors, another is sensitive to medium wavelengths of light, which is green, and then a third cone is sensitive to blue light, which has the shortest wavelength. These cones are often abbreviated to L, M and S cones, respectively. When a person has three fully functioning cones, they have trichromatic vision.

There are a few main types of colorblindness that we can observe. The least severe form of colorblindness is called anomalous trichromacy, which occurs when two cones' sensitive ranges overlap more than they should. For example, in a normal eye there is a small range of light wavelengths that can be perceived by both the S and M cones. In the eye of someone with anomalous trichromacy, however, that range is much larger than it should be. The intensity of this disorder can widely vary. Although some anomalous trichromats essentially have full color vision, others are severe enough to be nearly dichromatic. This kind of colorblindness is by far the most common, affecting about 8% of the population in total, with different demographics being affected at different rates (Neitz et al. 1998, 101).

The next type of colorblindness is called dichromacy. People with dichromacy can only see two colors, meaning that one of the cones in their eye is either not present or defective. Protanopia (loss of function in the L cone) and deuteranopia (loss of function in the M cone) are about equally common in people who are assigned male at birth (AMAB). Deuteranopia is slightly more common, occurring in 1.1% of AMAB people and 0.1% of people who are assigned female at birth (AFAB). Protanopia occurs in 1.0% of AMAB people and 0.02% of AFAB people (Jefferson et al. 2006). Since the L and M cones correspond to red and green vision, respectively, the two conditions are often grouped into one diagnosis, called red-green colorblindness. A defect involving the S cone, which is called tritanopia, is also possible and it results in the person being unable to tell the difference between blue and yellow. It affects both AMAB and AFAB people equally because of process by which it is inherited (Neitz et al. 1998, 103).

The final type of colorblindness is called monochromacy. This is the rarest form of colorblindness and it is marked by the complete absence of the cone photopigments in the eye. People with monochromacy cannot see color at all and are completely colorblind. This condition can be further categorized into two conditions, the first of which is called rod monochromacy. This condition occurs when the person has no function in any of the three types of cones. The second condition is called blue cone monochromacy, which is when only the S cones remain functional (Neitz et al. 1998, 101).

In the same way that there are many kinds of colorblindness, there are also many ways of acquiring colorblindness. The most common of which is congenital color-blindness, which is widely known to be much more common in AMAB people than AFAB people because it is a recessive trait carried on the X chromosome. Congenital colorblindness is probably one of the

most famous congenital defects because of this tendency in AMAB people. The human genome has 23 pairs of chromosomes: 22 pairs of autosomes and one pair of sex chromosomes which differ based on sex (Neitz et al. 1998, 101). People who are assigned male at birth have an X and Y chromosome, while people who are assigned female at birth have two X chromosomes. For AFAB people, a given gene on the X chromosome is called heterozygous if the gene is different on each X chromosome, and homozygous if the gene is the same on both X chromosomes (102).

In somatic cells, the members of each autosome are commonly expressed. In sex chromosomes, however, the X chromosomes contains much more genetic information, and so many more genes are only on the X chromosome. This is the case for most types of congenital colorblindness. In AFAB people, one X chromosome is randomly selected. Red-green colorblindness is inherited as X-linked traits because the genes that have the instructions on making L and M cones are on the X chromosome. Take for example an AFAB person who is heterozygous for protanopia, meaning one of their X chromosomes contains the defect and the other does not. These people are called carriers because they have the capabilities of passing the defect on to their children but they do not present any signs of the defect. This occurs because the X that does not contain the defect will activate the correct genes. If this person has a child with someone who has normal vision, their sons have a 50% chance of having protanopia and their daughters have a 50% chance of becoming carriers like their mother. If, however, a carrier has a daughter with a protanope, their daughter has a 50% chance of having protanopia and a 50% chance of being a carrier. This scenario also applies to the other X linked defects like deuteranopia and blue cone monochromacy (Neitz et al. 1998, 102). This does not apply to tritanopia, however, because it is not an X linked defect. In fact, tritanopia is carried in one of the

22 autosomal chromosome pairs, and that is why it presents itself in both AMAB and AFAB people equally.

In total, colorblindness affects about 1 in 12 AMAB people and 1 in 200 AFAB, which means there are about 300 million people in the world with colorblindness (Colour Blind Awareness n.d.).

Section 2: History of Color Vision

An understanding of colorblindness was impossible before scientists could understand the way that light works and what happens physiologically in our eyes and brains that allows us to see color. For centuries in the West, all colors were thought to be a series of intermediates between black and white. It was believed that black and white were the extremes and colors like red, blue and yellow were on a spectrum in between. It was not until the 17th century that these old theories around light were disproved, largely thanks to Sir Isaac Newton's famous Cambridge experiments with prisms that he wrote about in *Opticks* (Lee 2008, 4). However, Newton was not a proponent of the wave theory of light which we know to be true today, but instead believed in the corpuscular theory of light which held that light is made up of tiny particles (corpuscles). These two theories, corpuscular and wave, would continue to compete until Jean Foucault disproved the corpuscular theory by measuring the speed of light (3). The wave theory predicted that light would move faster in air than in water, while the corpuscular theory predicted the opposite. Foucault found that the wave theories predictions were correct, and so the corpuscular theory fell out of favor.

Theories of light become relevant when examining the psychophysiological theories of how we understand color. One of the first academic proponents of the trichromatic theory of

color perception was Dr. Thomas Young, who proposed the existence of three color-receptors that can perceive a range of wavelengths in his lecture, *On the Theory of Light and Colours*. In one section of the lecture discussing how the eyes sense color, he notes the following:

> It is almost impossible to conceive each sensitive point of the retina to contain an infinite number of particles, each capable of vibrating in perfect unison with every possible undulation, it becomes necessary to suppose the number limited, for instance, to the three principal colours, red, yellow, and blue, of which the undulations are related in magnitude nearly as the numbers 8, 7, and 6.[1]

Young is essentially stating that it would be impossible for the eye to be tuned to every possible frequency of light, and instead it must have bodies that are only sensitive to certain wavelengths of light. His theory of the existence of only three receptors in the eye was not widely believed for some time, however (Lee 2008, 6).

Section 3: Colorblindness History and Treatments

The English chemist John Dalton, famous for his work on atomic theory, is often credited as the first member of academia to take an interest in colorblindness and attempt to analyze it, spurred on by his own colorblindness. In some languages the term "colorblindness" is called "Daltonism" in his honor (Hunt et al. 1995, 984). In fact, one method for correcting colorblindness is also named after Dalton and called "daltonization". Dalton first took an interest in color vision when he noticed inconsistencies in the ways that he viewed colors and gave lectures on the subject in 1794. He believed that the cause of his inability to see color was that "the vitreous humor of his eye was tinted blue" (984) and thus absorbed the longer wavelengths

[1] Thomas Young, "The Bakerian Lecture. On the Theory of Light and Colours," in *Philisophical Transactions of the Royal Society*, (London: G. and G. Nicol, 1802), 21.

of light, such as red and orange. He requested that after he died his eyes be autopsied to confirm his hypothesis. However, the doctor who performed the autopsy found that his colorblindness was not caused by a filter in the eye that prevented him from seeing color. He did note "deficient development" in the part of his frontal lobe that processes color, which coincided with the other popular explanation of the time which was that colorblindness is caused by a defect in the brain (984). Young, however, predicted that it was possible that people with colorblindness are missing one of the three receptors in the eyes (Lee 2008, 5).

There were two main obstacles that stood in the way of a better understanding of colorblindness. The first, explored above, was the competing theories around the psychophysical perception of color. The second was a lack of subjects and equipment. The lack of interest in colorblindness and subjects with colorblindness that persisted for a majority of human history is puzzling, considering how critical it was for our primate ancestors to perceive colors on the red-green spectrum. Think, for instance, of trying to pick raspberries from a bush. Regardless, there were not many colorblind subjects that physicists and doctors could examine. And even when they could find subjects, they were unable to truly test color perception until James Clerk Maxwell devised a test in 1865 (8). The test was based on the Young's theory that hues can be created by the three primary colors. Maxwell took a disc and placed different colored paper on it. He then took a smaller disc with black and white paper and placed it on top of the larger disc, then placed them both on a spinning table. By overlapping the pieces of colored paper and spinning the top, he could show that you can create different hues by mixing colors (Maxwell 1865, 275-6). Maxwell was able to test the apparatus on colorblind subjects and definitively show what colors they could and could not see.

Historically, there have not been many attempts to correct colorblindness until the 20th century. The earliest example of colorblindness corrective lenses I could find was a patent filed in 1970, which was an improvement on a patent from 1965 (Zeltzer 1970). These glasses are now the most popular product to remedy colorblindness. The most popular producer of these glasses is EnChroma, whose glasses are targeted at people with mild to medium intensity red-green colorblindness (EnChroma, n.d.). This form of correction has grown in popularity as of late and there is no shortage of viral videos of colorblind people tearfully trying them on for the first time. The glasses work by increasing the contrasts between red wavelengths and green wavelengths, making red objects redder and green objects greener (EnChroma n.d.). However, since dichromats are missing a cone entirely, they do not work for them as well as they do for anomalous trichromats. They also do not work for any variation of yellow-blue colorblindness. In one 2018 study, researchers found that subjects' ability to recognize and arrange colors did not improve with the use of the corrective lenses, i.e. their ability to complete an Ishihara or Fansworth-Munsell test (Gómez-Robledo et al. 2018, 28,689-91).

One promising path for colorblindness treatment is gene therapy. Experiments in mice and primates have been very successful. In 2017, the U.S. Food and Drug Administration approved a new drug called Luxturne (voretigene neparvovec-ryzl) to treat a rare congenital condition that causes vision loss (FDA 2017). In April 2020, a new study on gene therapy to treat achromatopsia was published in *Jama Ophthalmology*. The 9 subjects in the study had two injections of AAV8.CNGA3 in one eye. The virus would target the CNGA3 gene which is responsible for making part of the cone. The patients had no immune response to the treatment and exhibited in increase in visual acuity and contrast sensitivity (Fischer et al. 2020, 644-7). Hopefully, more human trials will be safely conducted in the future, though it is unlikely that

gene therapy will become a common treatment for more common forms of colorblindness due to the costly and invasive nature of gene therapy.

Section 4: Methods of correcting colorblindness digitally

There are various methods for digitally correcting colorblindness that have been attempted. One of the simplest methods is to simply make an image monochromatic. When testing on Ishihara plates, it makes them virtually unreadable to normal users, but makes them just slightly clearer for colorblind users (Kulshrestha et al. 2013, 19). The problem, of course, is that this method does virtually nothing to give the user an accurate representation of the image, and so the method is best left untouched.

Another method for correcting colorblindness digitally is with the use of augmented reality. This method has been gaining popularity as augmented reality technology grows more accessible. Currently, there are smart phone apps that colorblind users can download to help them better see the world. One such application is Chroma, an extension for Google Glass that allows users to better see the world in real time (Tanuwidjaja et al. 2014). These solutions differ from colorblind glasses like EnChroma because rather than refracting the light to allow users to better distinguish between colors, these AR solutions merely filter the video that they receive similarly to filters that can be downloaded on one's computer. That being said, they are a cheaper alternative to pricy colorblindness glasses like EnChroma which, as previously mentioned, may not significantly increase color vision (Gómez-Robledo et al. 2018, 28,691).

I decided to replicate the correction method used by Jefferson and Harvey in their papers "Accommodating Color Blind Computer Users" (2006) and "An Interface to Support Color Blind Computer Users" (2007). Their method used optimization in order to create a mapping of

key colors of an image. This mapping would be optimized in order to match the color differences of the original image. After reading their papers, I was left wondering what would happen to colors that are unaffected by colorblindness, like the color blue in a red-green colorblindness correction. Furthermore, I wanted to see how their method would handle more complex images. These questions, along with my interest in improving the algorithm to make it more suited for real-time corrections, motivated my decision to replicate Jefferson and Harvey's algorithm.

Methods

Before explaining the methodology of the correction process used by Jefferson and Harvey in their papers “Accommodating Color Blind Computer Users” (2006) and “An Interface to Support Color Blind Users” (2007), it is important to define the terminology used in this algorithm. When referring to what a trichromatic user would see, I use the term “wild type vision,” whereas I use “colorblind vision” or “protanopia vision,” “deuteranopia vision,” etc. to denote what a colorblind user would see.

Jefferson and Harvey’s algorithm can be broken down into five steps. First, simulate the effects of the type of colorblindness on which the correction will be focused. Second, select N key colors from the image based on the different colors that would be missing due to that kind of colorblindness. Third, calculate the key color distances which are determined by the color and brightness of each of the colors. Fourth, minimize the key color distances to new mappings for the key colors which would be more appropriate for a given type of colorblindness. Finally, interpolate the key colors’ new mappings in the place of the colors unavailable due to the given type of colorblindness.

Section 1: Simulating colorblindness

Jefferson and Harvey’s algorithm to correct colorblindness works by simulating one of the three most common types of congenital colorblindness and then essentially altering the colors of the image to maintain color and brightness differences. Thus, it is important to find a good method for simulating colorblindness.

The colorblindness simulation itself is achieved via a series of matrix operations. First, I convert the image I wish to correct from a PIL Image (Clark 2015) object into a *numpy* array of size [*M*, *N*, 3] with the *asarray()* method from the *numpy* library (Harris et al. 2020, 257-62). *M* and *N* are the length and width of the image, respectively, and the final value, 3, represents the RGB values of each pixel. Next, I take the implicit Einstein sum (using *np.einsum()*) of the RGB matrix and an LMS (Large, Medium, Small) conversion matrix. The LMS colorspace is critical because it is meant to mimic the three types of cones that are in the human eye. This means the image can be altered specifically based on the type of colorblindness on which the correction will focus. The Einstein sum is meant to preserve the shape of the original RGB matrix, which is the same shape as the original image, and is therefore mostly useful for notational purposes (Harris et al. 2020, 257-62). Now, a colorblindness simulation matrix is multiplied by the LMS image matrix. Since the simulation's focus is the three most common types of congenital colorblindness, there are three different simulation matrices that might be used. The simulation matrices are as follows:

Protanopia: $$\begin{bmatrix} 0 & 2.02344 & -2.52581 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Deuteranopia: $$\begin{bmatrix} 1 & 0 & 0 \\ 0.494207 & 0 & 1.24827 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Tritanopia: $$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -0.395913 & 0.801109 & 0 \end{bmatrix} \quad (3)$$

Then the Einstein sum of the LMS image matrix and the colorblindness simulation matrix is taken. Finally, this new LMS matrix is converted back to an RGB image matrix, which is converted to a PIL Image using the *fromarray()* PIL function.

Section 2: Selecting Key Colors

In order to select the most appropriate colors on which to focus their correction, Jefferson and Harvey created what they called a "difference histogram" (2006). A difference histogram is a histogram of all the pixels that are in the wild type vision image and not the colorblind vision image. Thus, the difference histogram shows the colors that are lost in colorblind vision, as well as what proportion of the wild type vision image those colors make up. To create this difference histogram, histograms of both the wild type vision image and the colorblind vision image must be created first. To histogram an image, each RGB value is put in one of 1000 bins (10 for R, 10 for G, 10 for B), resulting in a three-dimensional histogram of size (10,10,10). Each bin's value is the proportion of pixels of the entire image that are in that bin. To create the difference histogram, simply subtract all the bins of the colorblind vision histogram from the wild type vision histogram. From the bins of the difference histogram, the key colors are selected via a weighted random choice. Thus, the bins with a higher proportion of pixels are more likely to be chosen. The average colors of each selected bin are used as the key colors.

Section 3: Calculating Key Color Differences

The next step entails calculating the differences of each of the key colors in relation to each other in the wild type vision image. This produces a difference matrix which will be used in the optimization process. It will be critical to the correction that these differences are preserved so that the new mappings found during optimization contrast each other in the same way as the key colorsin the wild type vision image. Jefferson and Harvey refer to a W3C manual to determine what chiefly denotes the difference between two colors (Jefferson et al. 2006). The difference between two colors is two-fold; the difference in brightness and the difference in color (Ridpath et al. 2000).

For colors C_i and C_j, the difference in brightness is:

$$D_b(i,j) = |Y(C_i) - Y(C_j)| \tag{4}$$

Such that *Y(C)* is the luminance component *Y* of the color *C* in the *YIQ* colorspace. The transpose of the RGB components of the color *C* must be multiplied by this luminance component, *Y* as follows:

$$Y(C) = [0.299, 0.587, 0.114] \bullet [C_R, C_G, C_B]^T \tag{5}$$

Next, the color difference, D_c, is found:

$$D_c(i,j) = \sum_{p=1}^{3} |C_i[p] - C_j[p]| \tag{6}$$

Now, the difference between two colors, $D(i,j)$ can be found by taking the sum of (4) and (6) as follows:

$$D(i,j) = \alpha_1 D_c(i,j) + \alpha_2 D_b(i,j) \tag{7}$$

where α_1 and α_2 are weights so that the effect of the brightness difference and color difference on the final recoloring can be better controlled. The differences of all the key colors are represented as a matrix, *D*. These distances will be used to create mappings that attempt to match these differences for a colorblind user.

Section 4: Optimization

Next, a mapping for the key colors based on their differences must be created. This is achieved by minimizing an error function using the *fmin()* function from the *scipy.optimize()* module (Virtanen et al. 2020), which uses a Nelder-Mead simplex algorithm. This error function is multifaceted. Part 1 is a modified Sammon error (Jefferson et al. 2006):

$$E_1 = \frac{1}{\sum_{i<j} D(i,j)} \sum_{i<j}^{N} (D(i,j) - d(i,j))^2 \quad (8)$$

$D(i,j)$ is the difference between color C_i and C_j, and $d(i,j)$ is the distance between colors C'_i and C'_j, where C' is the mapped color. It is critical that $d(i,j)$ is the difference between the mapped colors after having been simulated for the type of colorblindness that is being corrected. Without this step, the luminance and color differences will matter little since the correction will not be focused for a colorblind viewer.

Part 2 of the error function exists to prevent RGB values from going out of the [0,255] range. Rather than truncating values to be in range, it instead punishes the optimization by adding extra error, making it much less likely that it will converge with values that are out of gambit.

$$E_2 = \sum_{p=1}^{3} \sum_{i \in G_1[p]} (C'_i)^2 + \sum_{i \in G_2[p]} (C'_i[p] - 1)^2 \quad (9)$$

Where $G_1[p]$ and $G_2[p]$ are sets of colors that have the p^{th} RGB value out of range. Thus,

$$G_1[p] = \{ i | C'_i[p] \in (-\infty, 0) \} \quad (10)$$

$$G_2[p] = \{ i | C'_i[p] \in (1, \infty) \} \quad (11)$$

Put more simply, $G_1[p]$ and $G_2[p]$ are the sets containing colors that have an RGB value that is out of gambit. The specific out of range R.G or B values are then selected to add to the total error of the optimization function.

The final error, E, is:

$$E = E_1 + E_2 \quad (12)$$

The parameters for the optimization are the RGB values of the key colors. So, for N key colors, there will be N*3 variables to optimize over, leading to a lengthy optimization process.

Thus, it is important to select an *N* that is high enough to properly correct the image, but low enough so the optimization does not take too long. The optimization runs from 10 random starting points. By starting point, I mean a random set of RGB values that will be used as the initial guess for the key colors' mappings. The set of mapped keys with the lowest final error is selected as the final set of mappings. As the optimization runs, the mapped RGB values will slowly shift to find the best mapping.

Section 5: Interpolation

Finally, the new mapped key colors must be interpolated back into the image. Only the colors that are in the difference histogram should be affected since colors that were not in the difference histogram are unchanged by the effects of colorblindness. To interpolate a given color, Jefferson and Harvey use the inverse square of its distance from the key colors, which is the Shepard method (2006). The new RGB value assigned to some color, *C*, is:

$$C = \frac{\sum_{i=1}^{N} w_i C'_i}{\sum_{i=1}^{N} w_i} \tag{13}$$

With C_i being the key colors, C'_i being the key colors' mapped equivalents. w_i is a weight value determined by:

$$w_i = \frac{1}{\|C - C_i\|^2} \tag{14}$$

If *C* is a key color, then:

$$C_i = C'_i \tag{15}$$

This is the algorithm that is used by Jefferson and Harvey. However, there are a few issues with this method of colorblindness correction. To start with, runtime is a major concern.

With only 25 key colors, the algorithm takes between 5-7 minutes to run. However, with 50 key colors the algorithm can run for over an hour. This makes real-time implementation impossible in the algorithm's current state. The parts of the algorithm that take the most time to run are the optimization and interpolation steps. The optimization in particular can be quite lengthy, especially as more key colors are used on more complex images.

Another issue with the algorithm is the randomness of key colors. The algorithm only optimizes a single image, which is to say that even if the algorithm could run in real time, it could potentially pick new key colors on every iteration. Furthermore, as colors come and go in real time, the algorithm would find new, potentially radically different, mappings for the same color based on the prevalence of other colors in the image. For all these reasons, it would be beneficial to create a more generalized set of mappings based on this algorithm.

Section 3: Mapping Improvements

The first method I used to improve the runtime was to create a generalized mapping by caching the mappings for an image that has a lot of common colors, and then using those mappings for a different image. First, I found a mapping for 50 key colors on an image that had a lot of different colors in it. For these mapping tests, I used a slightly different key color selection process wherein I pick the most common colors instead of a partially random assortment of colors. My reasoning was twofold; I wanted to more easily recreate the results that my tests produced, which would be much more difficult with random key color selection, and I wanted to target colors that occurred the most often in this image. My hopes were that the image on which the mapping would be based would be a decent representation of the potential colors that would need to be corrected for a colorblind user.

Although this method was partially successful, I tried to further refine my process. I created a new method that involved comparing the color differences of the generalized mapping and a sample of the key colors from the image I aimed to correct. My thought process was that an image with fewer colors will have many similar key colors, thus producing a lower average color difference among the key colors. For instance, if an image being corrected for a protanope only has the colors red and green, then the image's key colors would mainly be different shades of red and green. On the other hand, if an image being corrected for a protanope has a large swathe of colors, it would still likely select reds and greens, but also shades of pink, purple and orange. Thus, the average difference of the key colors of the first image would be lower than that of the second image.

With this in mind, I sought to make the differences of the generalized mapping more similar to the differences of the image specific key colors. First, I removed colors from the generalized mapping that are absent from the image I wished to correct. Next, I found the key colors of the image that was being corrected. Then, I created difference matrices for both sets of key colors. At first, I only removed the color that had the largest total difference from the generalized mapping. However, when that had little effect on the final recoloring, I continued to remove key colors from the generalized mapping until the average differences of the generalized mapping and the key colors of the image being corrected were nearly equal. Unfortunately, this made the final recoloring worse, not better.

My next attempt to correct the mappings involved trying out a bunch of different mappings and finding the most common mapping of a given color. Since there are over 16 million different colors that can be created with RGB coordinates, it is important to find common colors for my correction since the goal is to create a set of mapped colors that can be applied to

multiple different images. In an effort to select the colors that would be best used in a generalized mapping, I used Paul Hebert's "Colors of the Web" website where he isolates the most common colors from the most popular sites on the web (2016). I put all these colors in a histogram and picked average color from the 25 most populous bins to use in my correction. Instead of just making one mapping for all 25 colors, I made many mappings for each color. I wanted to create multiple mappings for each color, and then find the median mapping as it were. In other words, I wanted to find the mapping that was the most common.

In order to find as many possible mappings as possible, each color was mapped with just a few other colors, as well as with all the other colors. Since it would take very, very long to make a mapping for every single combination of colors, I decided to at first focus on colors that are close to each other. It happened that the first two most common colors were both shades of red, for example, so one of their mappings is a result of running the optimization with just those two colors. From there, I would make a mapping with the first three colors, then the first four colors, and so on until I made a mapping with all the colors. Then, I would start over again, except this time it would be just colors 2 and 3, then 2, 3 and 4, etc. As a result of this methodology, the colors in the middle of the list, namely colors 13 and 14 ended up having many more mappings than the colors in the beginning and the end of the list of colors. To compensate, and also to try out mappings of more dissimilar colors, I strategically shuffled the list of colors such that the colors in the middle of the list would be at the ends, and vis versa. In the end, each color had roughly the same number of mappings in total.

With these new mappings, I had to now choose one mapping for each color. At first, naively, I tried to just get the mean RGB value for a given key color. Of course, this only resulted in many shades of brown. Instead, I found that a much more effective strategy was to

find the median color of all the mappings to use as a generalized mapping. To do so, I binned each of the mapped colors and then selected the bin with the most mapped colors. Then, I took the mean of the colors that were placed in the bin. I was not just taking the average color of the bin, but instead taking the average color of the mappings that were in the bin. That way, if the mappings were all very similar and on the edge of two bins, I would get a color that reflected those mappings, not just the bin itself.

Section 4: Interpolation Improvements

The optimization is not the only aspect of the algorithm that can be improved. The interpolation can also be improved upon. The method I used to improve the interpolation was to speed up the process itself by caching the interpolations of the average colors in the difference histogram. When testing the interpolation, I found that small changes in the original color did not translate to equal changes in the mapped color. For example, color 1 was mapped to color 2. Then, when I reduced the R value by 10, it was not interpolated to color 2 with less R. Instead, the R, G and B values were all slightly adjusted. This occurs because in the original interpolation process, we use a weighting function that treats the distance in the RG and B values equally. So, our change in the R value does not result in a one-to-one change in the R value of the final interpolated color, instead the change in the R value is distributed across all three values.

What I attempted to do by caching the interpolated colors of the difference histogram was to make a one-to-one mapping, which would take less time. Rather than having to run the interpolation equation on every color in the image, I would have the interpolation done in advance by pre-interpolating all the colors that are lost by a given kind of colorblindness. Thus, the lengthy interpolation was complete in advance. Then I would perform these steps for each color in the image: find the bin of the color so I could look up the interpolation of the average

color in its bin. Then, I would adjust the interpolated average color based on the distance of the average color in the current pixel's bin from the color of the current pixel, creating a re-coloring that is faster than the original method of running the interpolation algorithm on every single pixel.

Results

Section 1: Histograms

For my first experiment, I tried to recreate the images that are found in Jefferson and Harvey's paper. The first step in doing this was to create my difference histogram. In the following figures, I will compare the results of Jefferson and Harvey with my own. First, I set out to recreate the histograms that Jefferson and Harvey included in their paper based on their correction of an image for a protanope. I made a histogram with 1000 bins; 10 bins for each RGB value. The image for which Jefferson and Harvey made histograms is found in Figure 3, with their resultant histograms in Figure 1 and my recreated histograms in Figure 2.

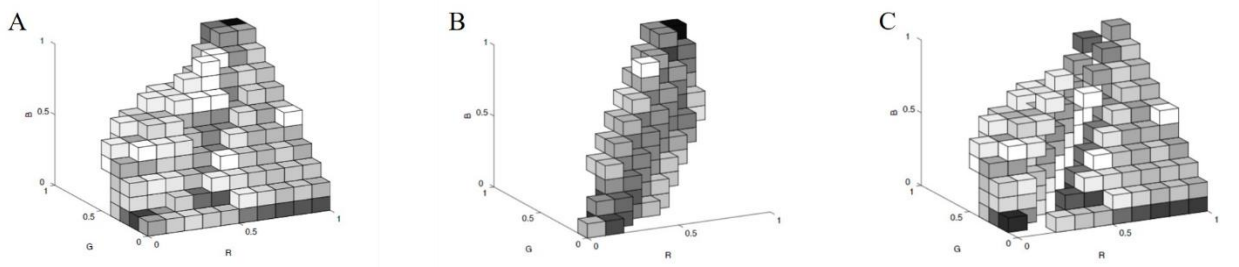

Figure 1: Jefferson and Harvey's histograms created from Figure 3A. (A) is the histogram of the wild type vision image; (B) is the histogram of the protanopia vision image, and (C) is the difference histogram created by removing (B) from (A)

Figure 1A is Jefferson and Harvey's histogram of the wild type vision image, whereas 1B is the histogram of the protanopia vision image. Finally, 1C is the difference histogram which is the result of removing the bins in 1B from 1A.

As you can see in Figure 1A, the wild-type vision image contains a wide breadth of color information, despite seemingly containing about 6 colors: the white background, black shadows, the red pencil, the brown pencil, the green pencil, and the reddish-brown wood of each pencil. Then, see how all of that information is compressed in Figure 1B, which is the histogram of the

protanopia-vision image. Since protanopia is a type of red-green colorblindness, you can see how the red and green portion of the histogram is empty, and all of the pixels that would be red or green in the wild type vision become brown in the protanopia vision. The difference histogram in 1C emphasizes this issue by featuring just those reds and greens that are lost with protanopia vision. These colors are what the correction must address.

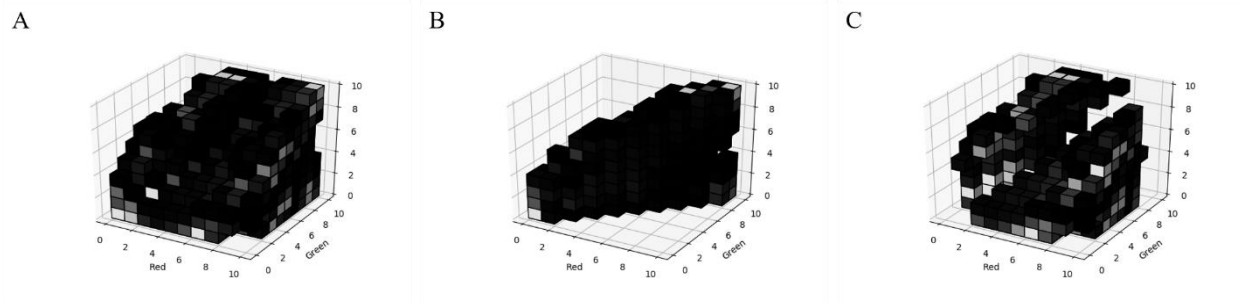

Figure 2: My histograms created from Figure 3A. (A) is the histogram of the wild type vision image; (B) is the histogram of the protanopia vision image, and (C) is the difference histogram created by removing (B) from (A)

Figure 2 features my histograms which are, from left to right, the wild type vision histogram of the image from Figure 3, the protanopia vision histogram of the image, and the difference histogram. One major difference between the Figure 1 histograms and the Figure 2 histograms is that I switched the x and y axes to provide a different view of the compression caused by protanopia in Figure 2B. You can get a better sense of the color information that is lost by protanopia vision by seeing it at this angle.

Section 2: Recreating Jefferson and Harvey's Results

For their tests, Jefferson and Harvey used 25 key colors and weight values of α_1=1.0 and α_2=0.5, so I did the same in my initial tests. Furthermore, I ran the optimization from 10 random points, and then used the mappings that came from the optimization that had the lowest final error. Figures 3, 4 and 5 contain the results of both Jefferson and Harvey's algorithm as well as

my algorithm on a series of images. I include the original wild type vision and colorblind vision images in each figure, followed by Jefferson and Harvey's results and then my own.

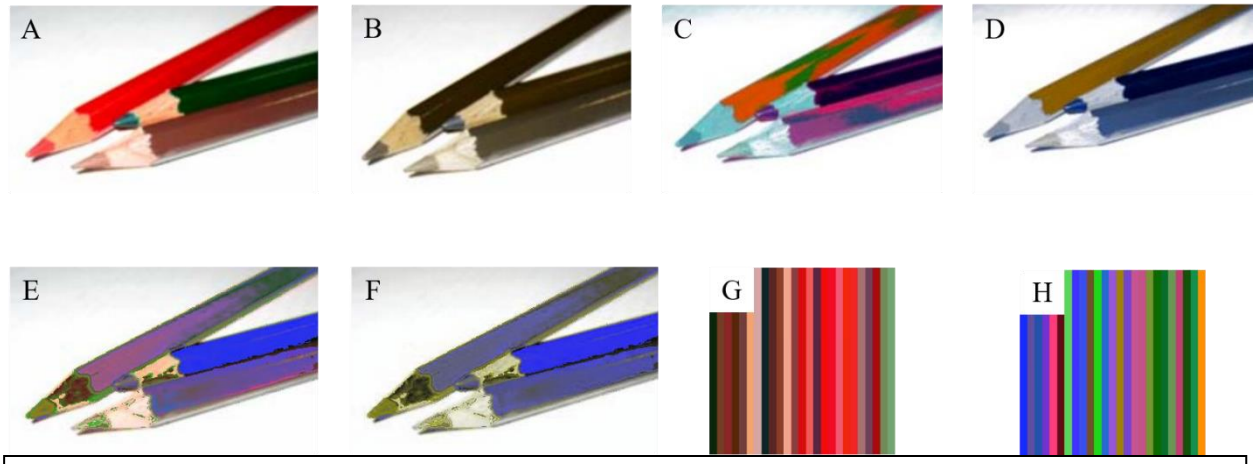

Figure 3: A comparison of Jefferson and Harvey's correction algorithm and my recreation for a protanopic user. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is Jefferson and Harvey's wild type vision recoloring; (D) is Jefferson and Harvey's protanopia vision recoloring; (E) is my wild type vision recoloring; (F) is my protanopia vision recoloring; (G) are the 25 key colors I used in my correction; and (H) are the key colors mappings that I used in my correction

Figure 3 is the result of the algorithm on a protanopia vision image. In this scenario, the user would have missing or damaged L cones, making the unable to distinguish red and green. As a result, the red and green pencils from Figure 3A end up becoming virtually indistinguishable from the brown pencil when viewed with protanopia vision, as you can see in Figure 3B. Jefferson and Harvey's mapping sent the red colors to green and orange, and made the green pencil a deep shade of purple. It also changed the brown pencil to purple and pink. These colors are seemingly random and look out of place in the wild type vision image, 3C, but they give a clear sense of contrast of color in the protanopia vision of the mapping in 3D. 3E and 3F are the wild type vision and protanopia vision of my attempt at a recoloring. You can see that an area my algorithm struggled with was handling the edges of objects. You can see that parts of the wood on the tips of the pencils gets incorrectly colored, but only a portion of it. While my algorithm does make the three pencils more differentiable than the original protanopia vision, it makes the red and brown pencils a bit too similar. Figure 3G features the key colors that my

algorithm selected from 3A, and Figure 3H shows the new colors to which the key colors were mapped. You will notice that the colors seem random, but are still rather tame compared to the mappings that Jefferson and Harvey used. You will also notice that many of those colors become lost in the recoloring because of the interpolation step.

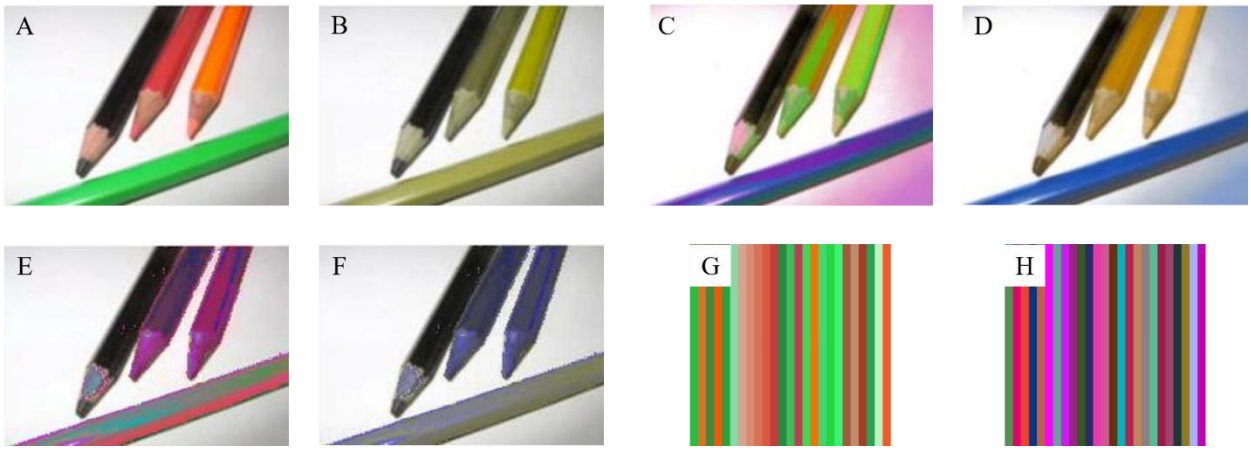

Figure 4: A comparison of Jefferson and Harvey's correction algorithm and my recreation for a deuteranopic user. (A) is the wild type vision image; (B) is the deuteranopia vision simulation of (A); (C) is Jefferson and Harvey's wild type vision recoloring; (D) is Jefferson and Harvey's deuteranopia vision recoloring; (E) is my wild type vision recoloring; (F) is my deuteranopia vision recoloring; (G) are the 25 key colors I used in my correction; and (H) are the key colors mappings that I used in my correction

Figure 4 is a run of the algorithm on a deuteranope vision image, which means the viewer would lack an M cone and thus also suffer from red-green colorblindness. Once again, A-D are Jefferson and Harvey's images, E and F are my mapped images, and G and H are my key colors and their mappings. Again, you will notice that red, orange and green are the colors that I had to correct because deuteranopia is a type of red-green colorblindness. Interestingly, both Jefferson and Harvey's algorithm and my own algorithm made the orange and red colored pencils very similar in the mapping, as you can see in 4D and 4F, even though in the wild type vision image the two pencils are markedly different colors.

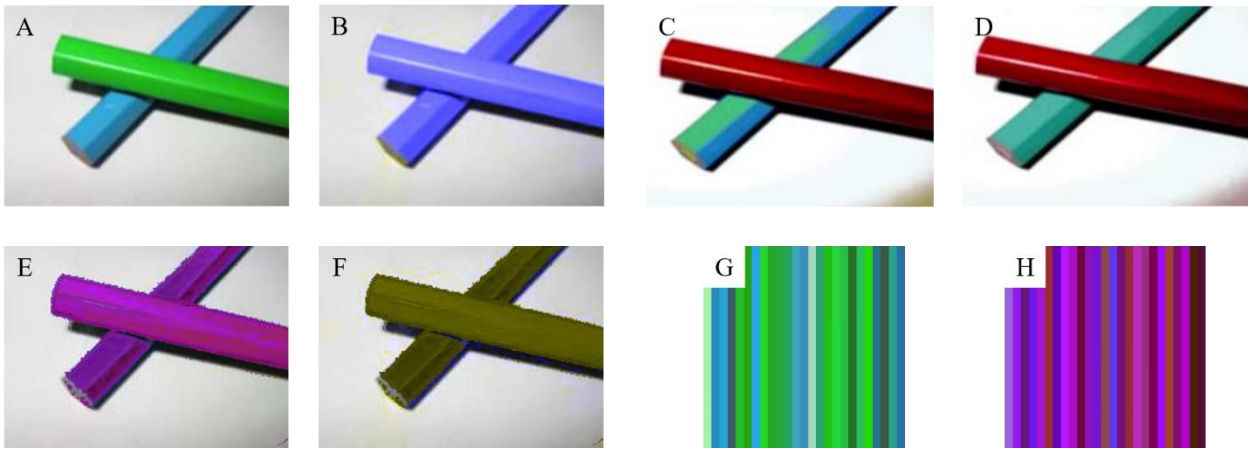

Figure 5: A comparison of Jefferson and Harvey's correction algorithm and my recreation for a tritanopic user. (A) is the wild type vision image; (B) is the tritanopia vision simulation of (A); (C) is Jefferson and Harvey's wild type vision recoloring; (D) is Jefferson and Harvey's tritanopia vision recoloring; (E) is my wild type vision recoloring; (F) is my tritanopia vision recoloring; (G) are the 25 key colors I used in my correction; and (H) are the key colors mappings that I used in my correction

Figure 5 are the results for a tritanope image, which is the rarest type of colorblindness associated with a missing S cone and trouble discerning blues and yellows. Again, A-B are the original images, C-D are Jefferson and Harvey's mappings, E-F are my mappings and G-H are the key colors I selected. In this image, both of the colored pencils are affected by the tritanopia vision, and so Jefferson and Harvey made the pencils green and red instead of blue and green. This mapping gave my algorithm quite a bit of trouble, as you can see. It seems that the lack of variety in the key color selection process caused my mapping to make all of the colors a dark purple, which did not preserve the differences in color or brightness in the final remapping. The final mappings are also suspiciously lacking in terms of color variety. Compared to the mappings of Figures 4h and 3H, the mappings of 5H are tame. If the algorithm selected a more vibrant array of colors, I believe the final recoloring would have improved.

Section 3: Further Testing

When reading Jefferson and Harvey's paper on this algorithm, I found their testing insufficient. An important motivation of this project was to further test the results of this

algorithm on more complex images. In my first set of tests, I wanted to continue to use colored pencils since I already knew that small sets of pencils could produce decent results. My main focus during these tests was to examine how the algorithm handles colors that are unaffected by the alterations made by a given type of colorblindness. For example, I will refer back to figure 3. Recall that in both Jefferson and Harvey's recoloring and my own, at least one of the colored pencils were recolored to be blue. In that case, the color blue is completely absent from the original image. I was curious what would happen if blue was present in the original image. Would the algorithm select a different color for the recoloring? Would it change the blue pencil even though it would be unaffected by protanopia? More generally, is the algorithm still effective on images with more than just a few colors? These were my main concerns when running these tests.

The next three figures will show the results of my tests on a single image of colored pencils. I once again used 25 key colors and weight values of α_1=1.0 and α_2=0.5. I also started the optimization from 10 random points and used the optimization with the lowest error for my final mapping, as I did when recreating Jefferson and Harvey's results. These tests will not only evaluate the effectiveness of the algorithm on more varied images, but they will also serve as excellent exhibitions of how each type of colorblindness affects an image, and the unique challenges that they pose while attempting correction.

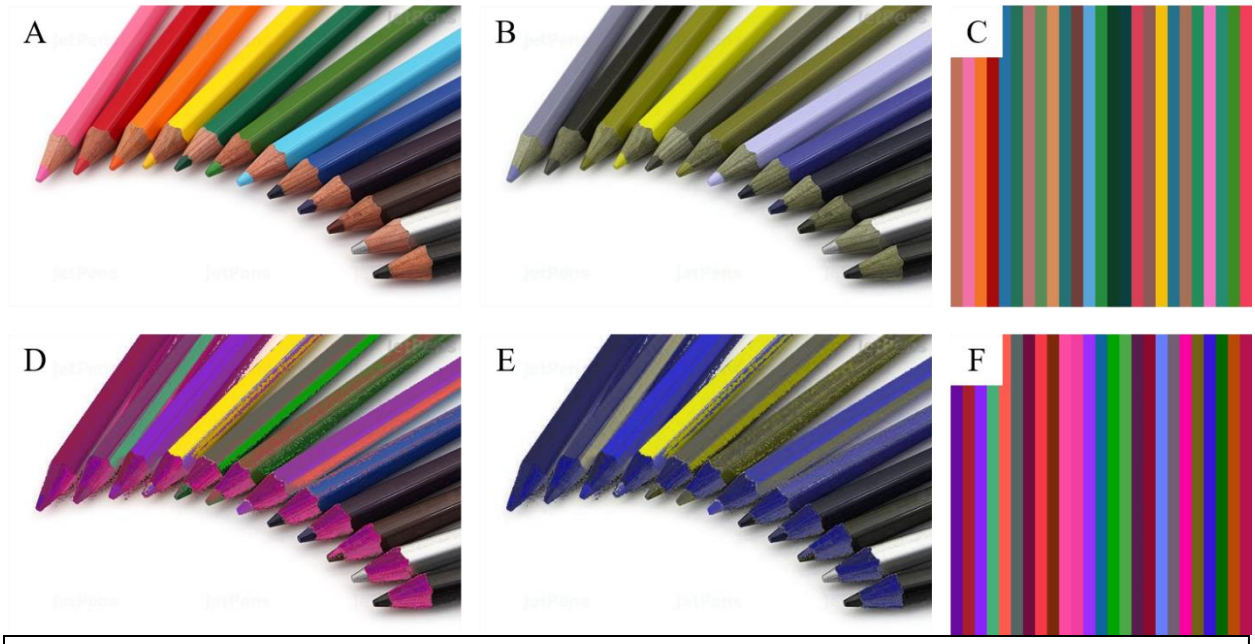

Figure 6: Protanopic correction of an image of colored pencils. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the protanopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

Figure 6 shows the results of the algorithm on the wild type vision image, Figure 6A. 6B is the protanopia vision of 6A. The 25 key colors that the algorithm selected are featured in 6C, while the mappings that the algorithm selected are in 6F. The wild type vision of the recoloring is in 6D, and the protanopia vision of the recoloring is in 6E. As you can see in 6B, a lot of the colors in the image are affected by protanopia. Thus, the key colors that are selected vary by quite a bit. Unfortunately, the recoloring greatly favors magenta and pink over other colors, sending most of the pencils that are problem colors to these colors. This leads to an unimpressive, and frankly more confusing, protanopia vision recoloring. The reddish-brown pencil wood becomes a dark blue in the protanopia vision recoloring, which throws the viewer off from the beginning. Add to that the overabundance of blue in the 6E overall, and it makes the recoloring worse than the original image in a lot of ways. This is the beginning of the answer to

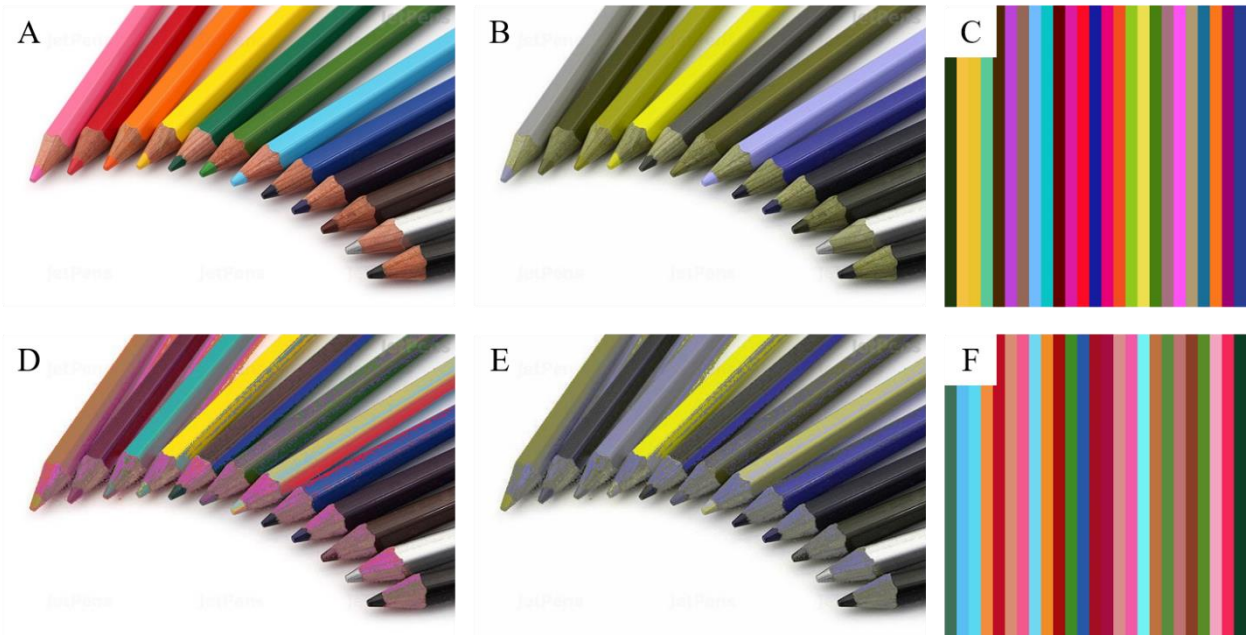

Figure 7: Deuteranopic correction of an image of colored pencils. (A) is the wild type vision image; (B) is the deuteranopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the deuteranopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

my earlier question regarding how colors that are normally unaffected by colorblindness are handled by the algorithm; poorly.

Although the protanopia correction was disappointing, the deuteranopia correction in Figure 7 gives cause for hope. The images in Figure 7 are organized in the same way as in Figure 6, the only difference being that the correction was made for deuteranopia instead of protanopia. Despite the similarity of these types of colorblindness (they are both categorized as kinds of red-green colorblindness), deuteranopia is handled far better by the algorithm. When comparing 7A, 7B and 7E, you will notice a number of small changes that lead to easier color differentiation. The orange pencil is made light blue in the remapping, which leads to a clearer distinction between the orange and yellow pencils. Furthermore, the pink pencil is made darker to make it more different from the red pencil. This is not to say the recoloring is perfect. In fact, the two

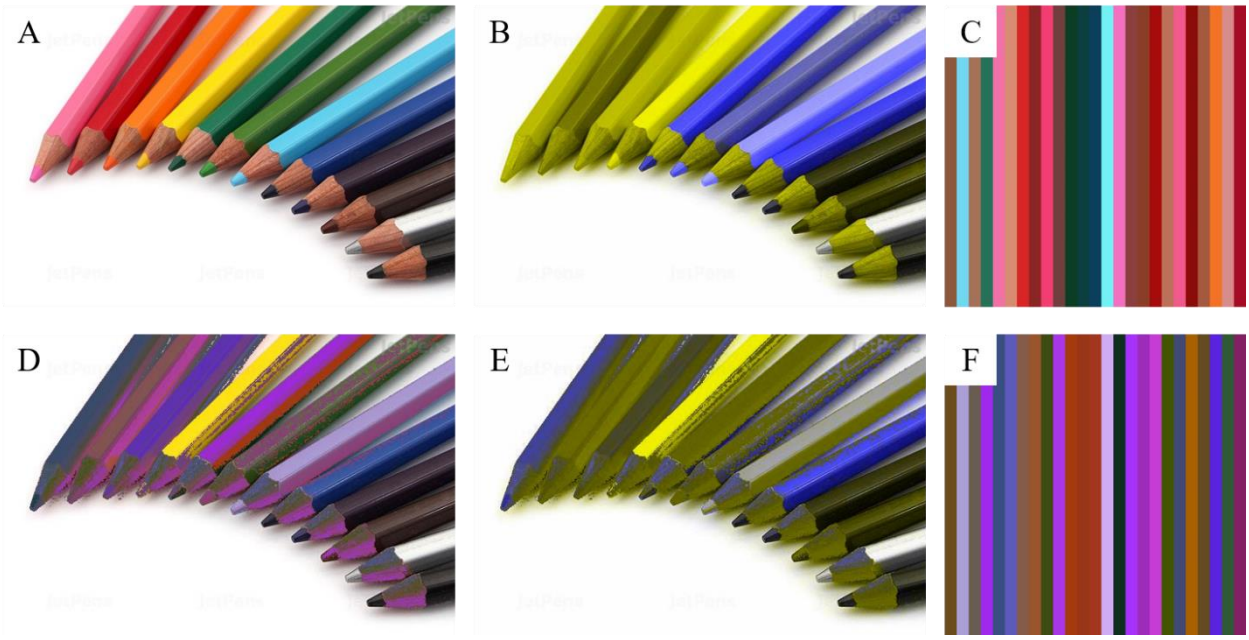

Figure 8: Tritanopic correction of an image of colored pencils. (A) is the wild type vision image; (B) is the tritanopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the tritanopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

green pencils become harder to tell apart from each other, while remaining equally similar to the red pencil.

Finally, Figure 8 shows the results of the algorithm on the same image, this time correcting for tritanopia. You can see in the tritanopia vision image, 8B, that this type of colorblindness has the least amount of variation out of the three. Clearly, a lot of work needs to be done to make these pencils distinguishable. The key color selection in 8C is indeed promising, but unfortunately the mappings in 8F are less so. Once again, the mappings are not very varied, which leads to a fairly monochrome tritanopia vision recoloring in 8E. Even in the wild type vision recoloring a lot of the colors that were clear before become muddled. It is worth noting that instead of making the pencils that all appear to be a similar shade of yellow in 6B more varied, the algorithm actually adds more pencils to the muddle of dark yellow by recoloring

previously blue pencils to grey and the green pencils to yellow in 8E. Needless to say, this recoloring is also quite frustrating.

All of these tests begin to show us the cracks in the algorithm. The addition of multiple shades of the same color, as well as colors that are unaffected by one kind of colorblindness but affected by another challenges the algorithm.

Section 4: Complex Images

My next tests involved trying the algorithm on images that are not colored pencils. Jefferson and Harvey do not show us how the algorithm does on images that do not contain colored pencils, so I tried it on three different, each catering to different weaknesses I identified in the algorithm tailored to each type of colorblindness it can correct. In these tests, I used the same parameters as my earlier tests; 25 key colors, weight values of α_1=1.0 and α_2=0.5, and 10 random starting points for the optimization.

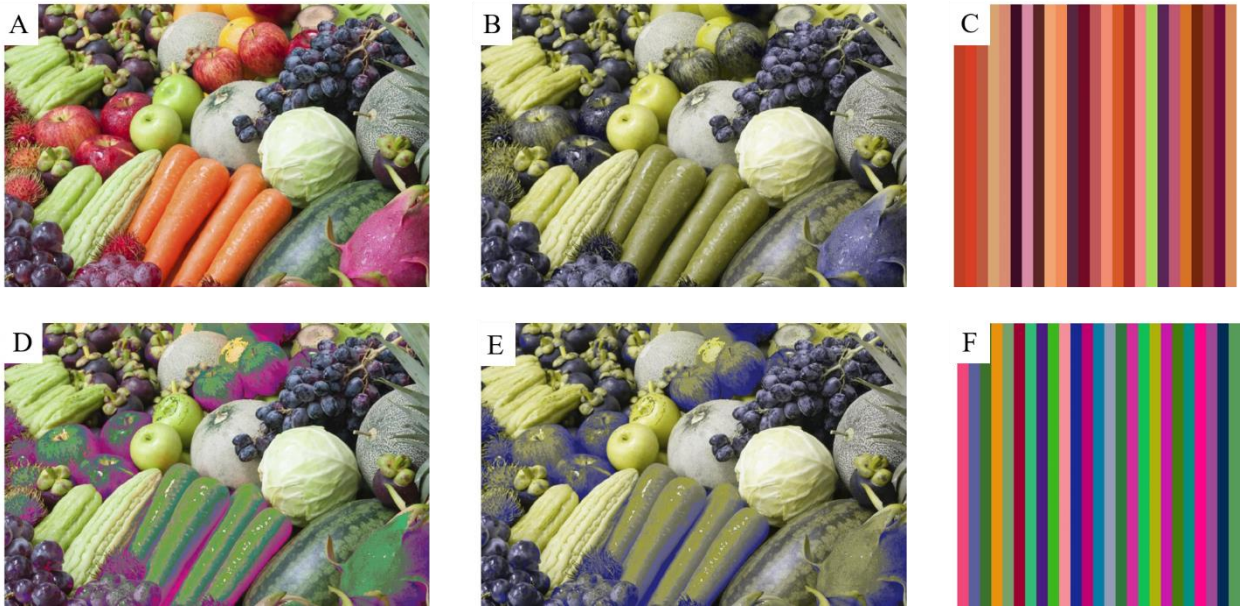

Figure 9: Protanopic correction of an image of fruit and vegetables. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the protanopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

Figure 9 is the results of the algorithm correcting for protanopia. 9A is the wild type image of fruit of varying shapes and colors. 9B is the protanopia vision image, and 9C features the 25 key colors which the algorithm will focus the correction on. Right away, you will notice that the key colors disproportionately feature dark purples and oranges, with not a single red. This is certainly troubling considering how the red apples, the dragon fruit, and the red fruit on the left are greatly affected by protanopia. This does not inspire hope for the recoloring. Sure enough, you can see in 9D and 9E that although the recoloring did change a lot of colors (compare 9D, the wild type vision of the recoloring to 9A) these changes do not translate well in the protanopia vision recoloring, 9E. Intriguingly, the mappings of the key colors in 9F are very promising. There is a lot of variety in the mappings, but all of that is lost in the recoloring. In past runs of the algorithm for protanopia, I noticed that the edges of objects caused issues, but that does not appear to be the case here. All of the fruit are still easy to pick out, or at the very least, equally difficult in both 9B and 9E. Surprisingly, it seems that the issue in this image is that the algorithm did not make enough changes, or at least that the changes it did make were not drastic enough to translate into a better image in 9E.

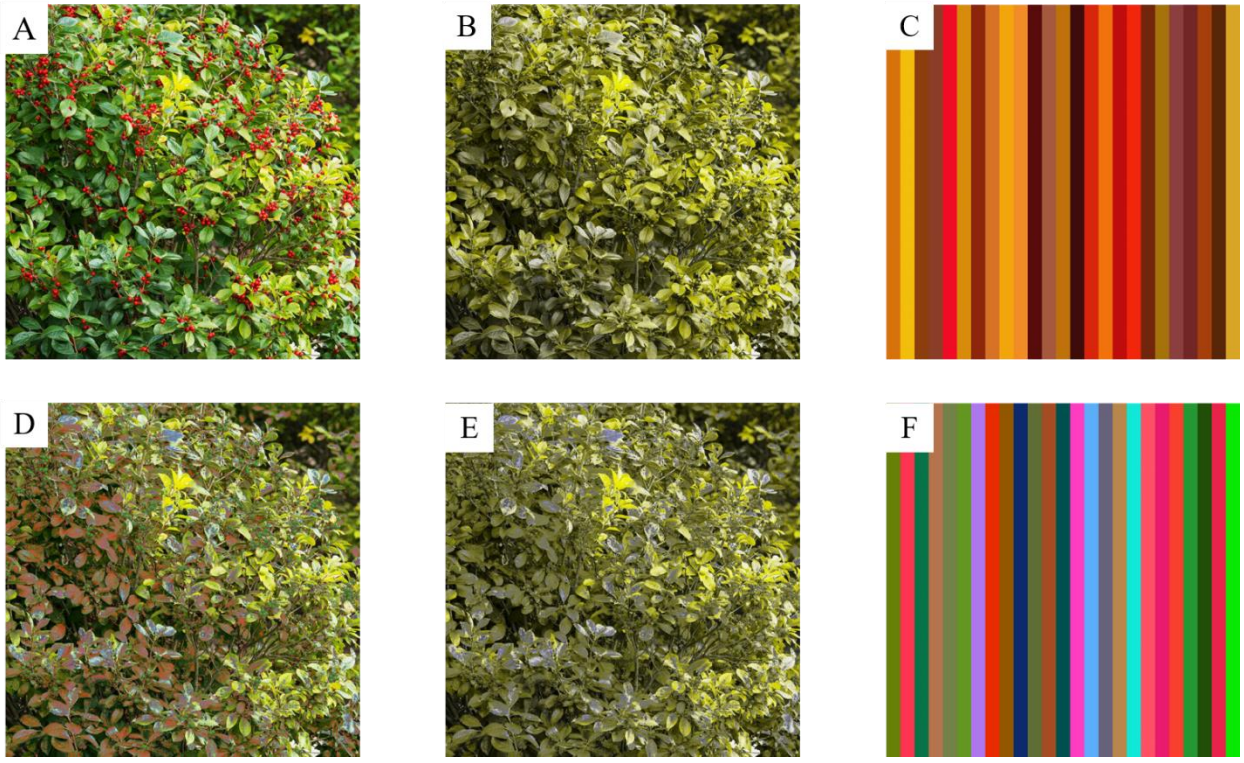

Figure 10: Deuteranopic correction of an image of red berries on a green bush. (A) is the wild type vision image; (B) is the deuteranopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the deuteranopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

Figure 10 shows the algorithm on an image of red berries and a green bush that has been corrected for a deuteranope. Unfortunately, these results are very disappointing, especially compared to how well the algorithm handled the deuteranope correction in Figure 7. 10B, the deuteranopia vision of the original image, is indeed troubling. The berries completely disappear into the green of the bushes. Once again, the key colors selected in 10C are a cause for worry because the algorithm seems to focus on brown from the undergrowth of the bushes and the yellow of some of the leaves. Even though the mapped colors in 10F are varied and look like they actually have the opposite effect. Instead of changing the red colors to a different, more visible color, they instead map the few red colors that are captured by the key colors to green! This makes for an absolutely awful mapping, even with wild type vision in 10D. Finally, 10E, the deuteranopia vision recoloring, makes almost no changes compared to 10B. The only change

is that the shadow of the bush in the bottom left of the image is slightly darker. The berries remain impossible to discern from the bushes.

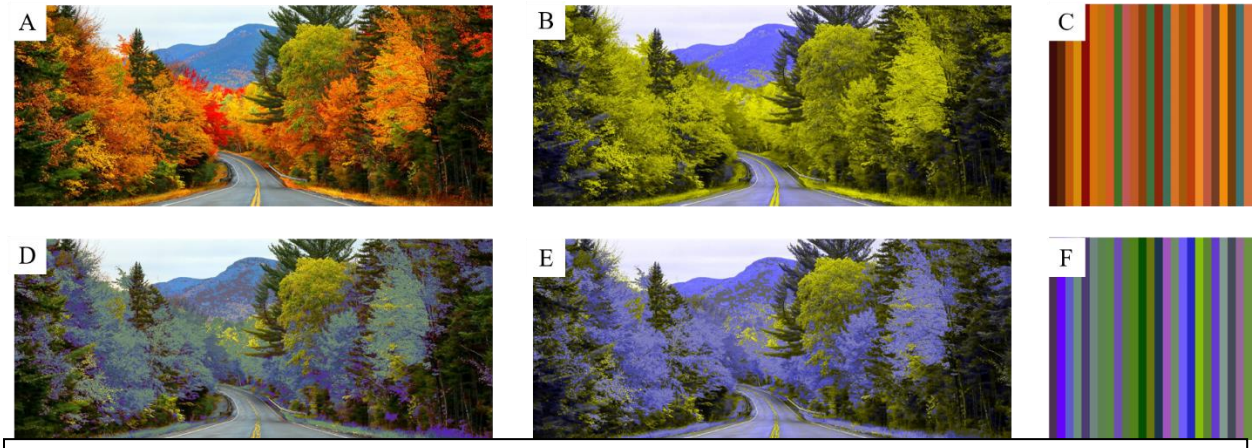

Figure 11: Tritanopic correction of fall foliage on a road with a mountain in the background. (A) is the wild type vision image; (B) is the tritanopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the tritanopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

Finally, Figure 11 is an image of fall foliage corrected for tritanopia vision. I selected this image because in past iterations of the algorithm on tritanope images, it struggled with two groups of colors being affected by tritanopia, so I wanted to see how it would handle the green, red and orange trees with the blue-green tinted mountain in the background. Interestingly, the algorithm handles this issue fairly well. The 25 key colors in 11C seem appropriate for the image, but their mappings in 11F seemed a bit too similar to me. However, the remapping is surprisingly good, especially considering how much the algorithm struggled with Figure 8. However, the algorithm does well with the orange trees in the front of the image, making them clearly different from the green trees. It struggled a bit with the bright red trees in the distance, and unfortunately did not make much of an adjustment to the mountain in the background to make it stand out from the bright foliage. However, the latter result was not surprising considering how the algorithm has performed on past images. It almost never adjusts colors that are not its original target.

Section 5: Weight Experiments

Another component of the algorithm that Jefferson and Harvey neglect is the weights. When I first started working on this algorithm, I thought the weights could be used to adjust the intensity of the correction. Thus, I thought that one could correct an image for a user with anomalous trichromacy, which causes highly individualistic abnormalities in the user's vision. However, the weights are actually used to affect the impact that brightness and color differences have on the optimization. Before I begin the optimization, I create a matrix of the differences between all the key colors. For each pair of key colors, I find the brightness difference and the color difference. To find the total difference between the two colors, I multiply each difference component by a weight value and then add the two components together. In Jefferson and Harvey's experiments, they used weight values of α_1=1.0 and α_2=0.5, which means that the brightness difference was unchanged, and the color difference was multiplied by 0.5. To see the effect of the weights on the final recoloring, I ran the algorithm on the same image 100 times with the same key colors each time. Each run of the algorithm used a different set of weight values. I found that with the exception of weight values α_1=0.1 and α_2=0.1, the weights did not affect the final recoloring.

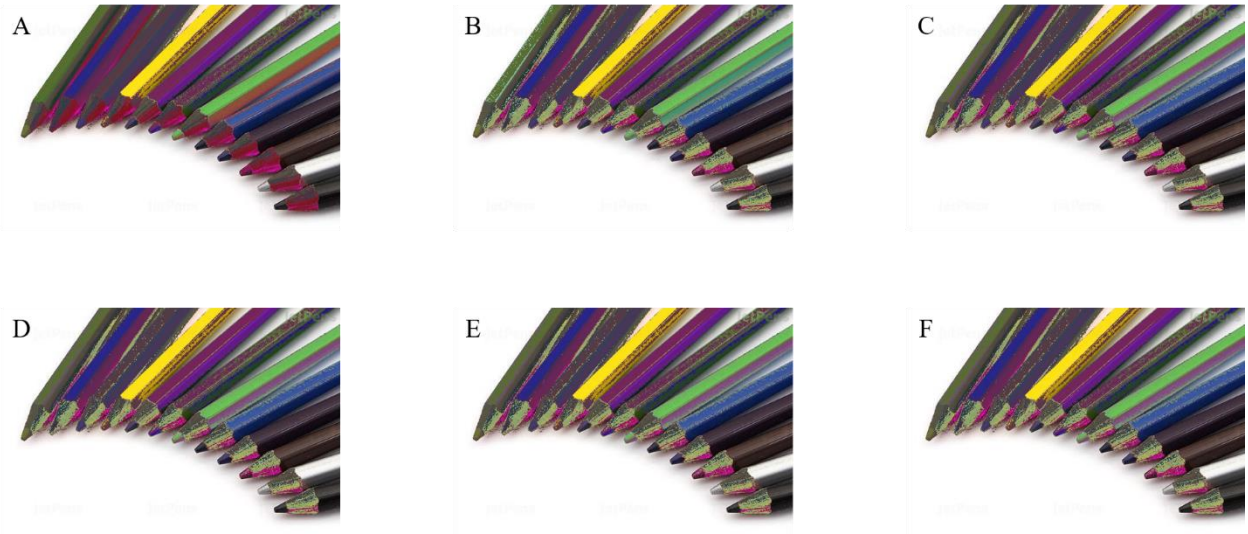

Figure 12: A series of wild type vision recolorings with varying weights of the image from Figure 6 for a protanopic user. (A) is a recoloring using α_1=0.1, α_2=0.1; (B) is a recoloring using α_1=0.1, α_2=0.2; (C) is a recoloring using α_1=0.1, α_2=0.9; (D) is a recoloring using α_1=0.5, α_2=0.5; (E) is a recoloring using α_1=0.7, α_2=0.1; and (F) is a recoloring using α_1=0.9, α_2=0.9

Figure 12 shows the results of my weights experiment. Figure 12A is the wild type recoloring of Figure 6A with weight values α_1=0.1 and α_2=0.1. You will notice that this iteration is the only one that is different from the other images. 12B is the recoloring with weight values α_1=0.1 and α_2=0.2, 12C has weights α_1=0.1 and α_2=0.9, Figure 12D has weight values α_1=0.5 and α_2=0.5, 12E has weight values α_1=0.7 and α_2=0.1, and 12F has weight values α_1=0.9 and α_2=0.9. Oddly enough, the only recoloring that is different from the rest is 12A with weights α_1=0.1 and α_2=0.1. I included a variety of weight ratios because I thought that the ratio of the weights might affect the final recoloring. However, all of the recolorings except for 12A are exactly the same.

Section 6: Mapping Generalization

My first attempt to make a generalized mapping was rather unsuccessful. I attempted to generalize a mapping of 50 key colors from one image to use for other images. First, I chose to use the same image I used from Figures 6-8 since it featured many colors. I hoped it could select

a good range of colors that it could use for a recoloring. I selected the 50 most common colors from the image to be the key colors from that image, aiming to correct for protanopia. I then ran the optimization to get the mappings for those key colors. I once again used weight values 1 and 0.5, and ran the optimization ten times from random starting points.

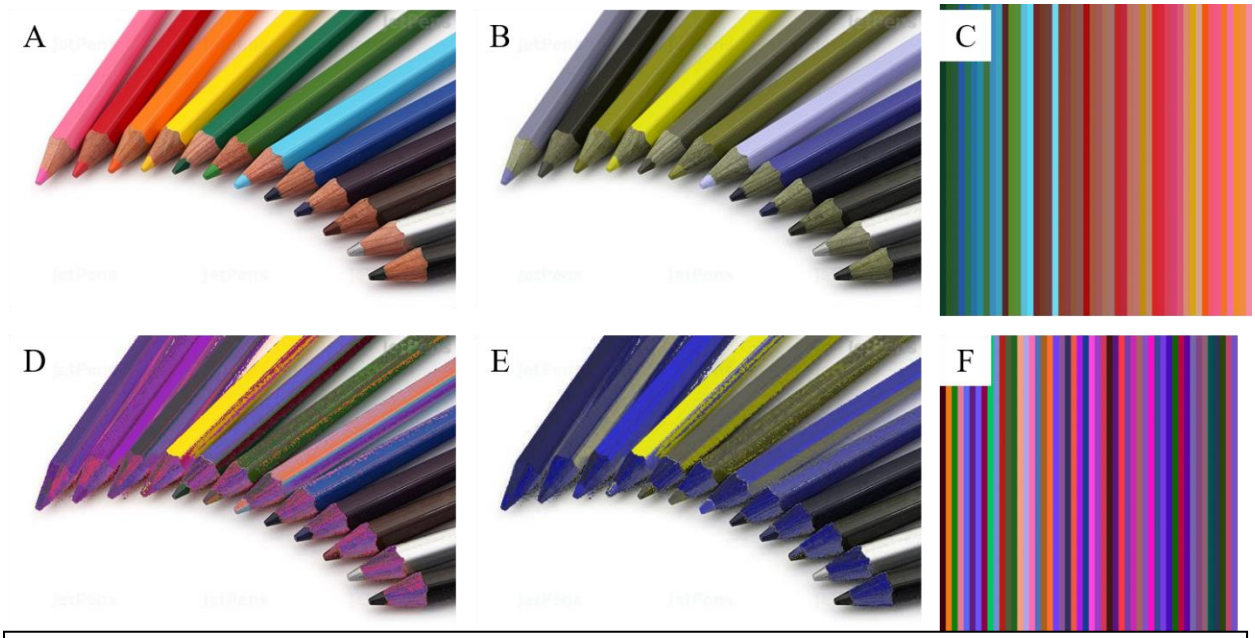

Figure 13: Protanopic recoloring of an image of colored pencils (the same from Figure 6). Corrected using 50 key colors instead of 25 key colors. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the protanopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

First, I tested the 50 key colors on the original image, the results of which can be found in Figure 13. 13A and 13B are the wild type vision and protanopia vision images, respectively. 13C is the 50 key colors that I used. Instead of randomly selecting colors, I chose the 50 most prominent problem colors by picking the 50 most populous bins from the difference histogram. I decided to use colors based on their prominence rather than a semi-random fashion because I thought it important that colors that were not prominent in the original image not be featured in the generalized mapping. Figure 13D and 13E are the wild type vision and protanopia vision recolorings, the results of which are unsurprising based on the results from Figure 6. Figure 13F shows the mappings for our key colors, which I found to be acceptable in terms of variety.

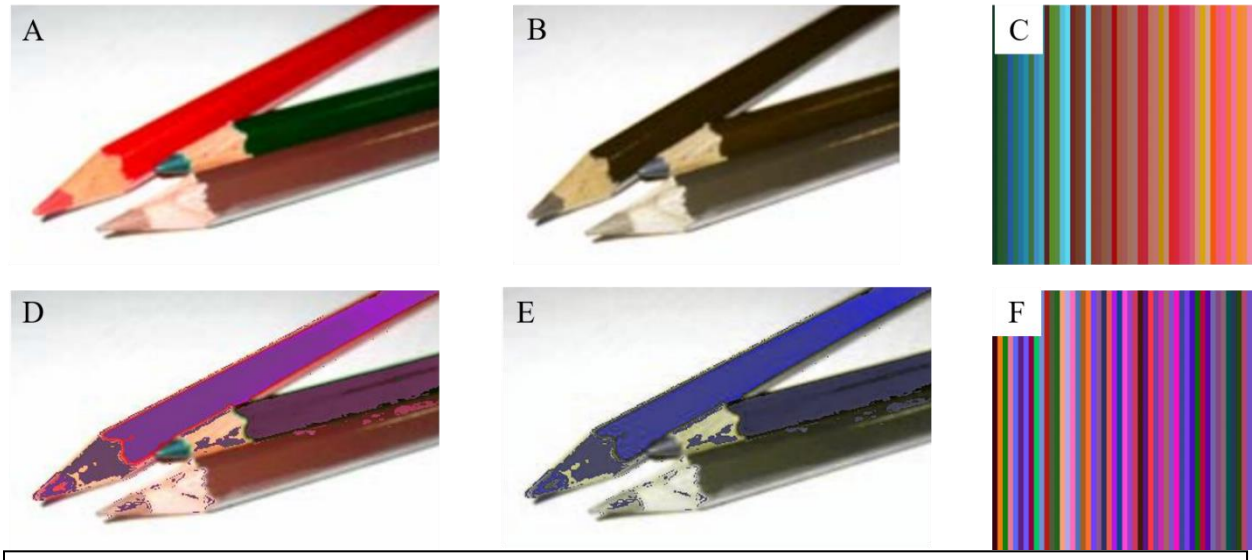

Figure 14: Protanopic recoloring of the image from Figure 3 using the key colors from Figure 13. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the protanopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

The next step was to test this generalized recoloring on different images and see how it performed. Figure 14 shows the results from this attempt. Figure 14A is the wild type vision image from Figure 3 on which I planned to use the generalized mapping, with 14B being the protanopia vision of 14A. 14C and 14F are the same as 13C and 13F since I used the exact same key colors. Figure 14D and 14E are the wild type vision and protanopia vision recolorings of the original image, using the 50 key colors shown in 14C and 14F. In some ways, this mapping is actually better than the mapping that I made from Figure 3. Notice that the three pencils in 14E are three distinct colors. Compare these to the pencils in 3F, in which two of the pencils are a bit too similar to easily tell them apart. Unfortunately, this recoloring is still not perfect. For one, there is quite a bit more incorrect recoloring in 14D and 14E than there was in Figure 3. By incorrect recoloring, I am referring to areas of the pencil that should all be the same color, but instead become a noticeably different color from the rest of the image. For example, look at the wood on the leftmost pencil in the recolorings. See also the boundary between the two pencils on the right, as well as the wood of the middle pencil and the lead of the right pencil. All of these

areas became a dark purple in the wild type vision recoloring and a dark blue-grey in the protanopia

Next, I tried to make the generalized key colors better match the differences between the key colors that the algorithm would normally select for a given image. I did this by adjusting the number of key colors based on the differences between those colors and the most prominent colors in the image I aimed to correct.

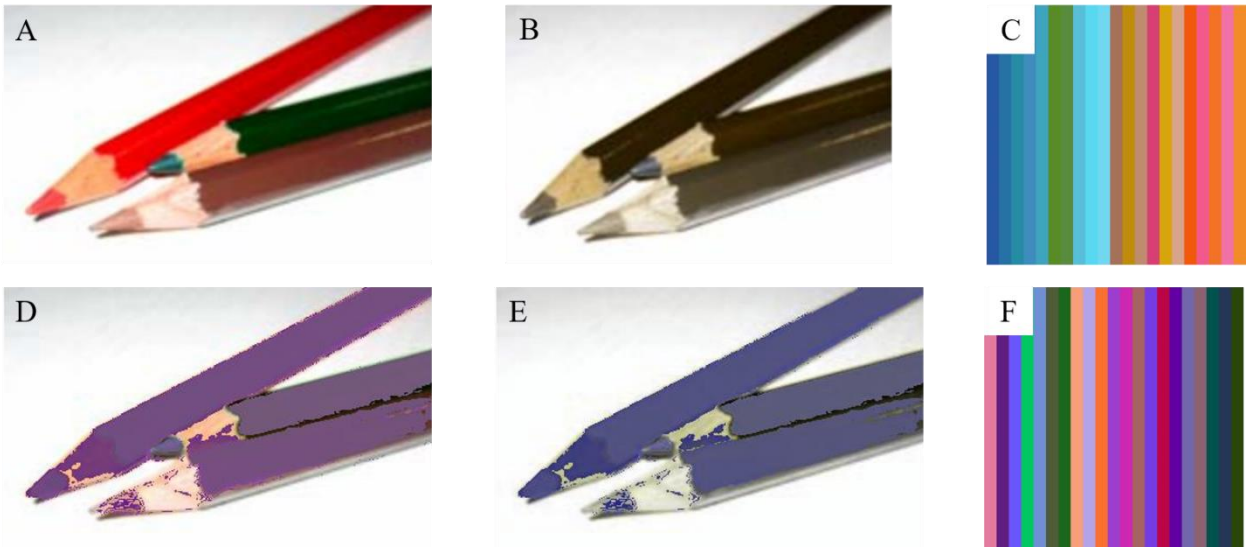

Figure 15: Protanopic recoloring of the image from Figure 3 using the parred key colors from Figure 13. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is the 25 key colors which the correction will target; (D) is the wild type vision recoloring; (E) is the protanopia vision recoloring; and (F) is the mapping of the 25 key colors used in the correction

This was the correction method I used to produce Figure 15. 15A and 15B are the wild type vision and protanopia vision of the original image. 15C is the pared down key colors that I used for the recoloring, with their associated mappings in 15F. I arrived at those specific colors as the key colors by examining the difference matrix of the 15A's own key colors. First, I selected 50 key colors from 15A. Then, I made a difference matrix containing the differences between all the key colors for both 15A's key colors and the 50 generalized key colors. I compared the sum of those matrices and removed colors from the generalized key colors until

the difference between the two sums was within 100. To select a color to remove from the generalized mapping, I would check to see which color had the largest set of differences from the other key colors, then removed that color and checked the total sums again. Doing this left us with the key colors of 15C. I then used these keys and their mappings to recolor 15A and produce 15D and 15E, which are the wild type vision and the protanopia vision recolorings, respectively. Clearly, this method was unsuccessful. The three pencils all became a dull blue color in the protanopia vision recoloring, and they become virtually identical in color. By removing key colors, I inadvertently reduced the specificity of the correction, which translated to a recoloring that was all one or two colors. This method of applying a generalized mapping is a regression from my previous method, not an advancement.

The hitch I encountered in my previous attempt to make a generalized mapping led me to take a different approach. Instead of picking out 25 colors from a random image, I based this generalized mapping on the most common colors on the web.

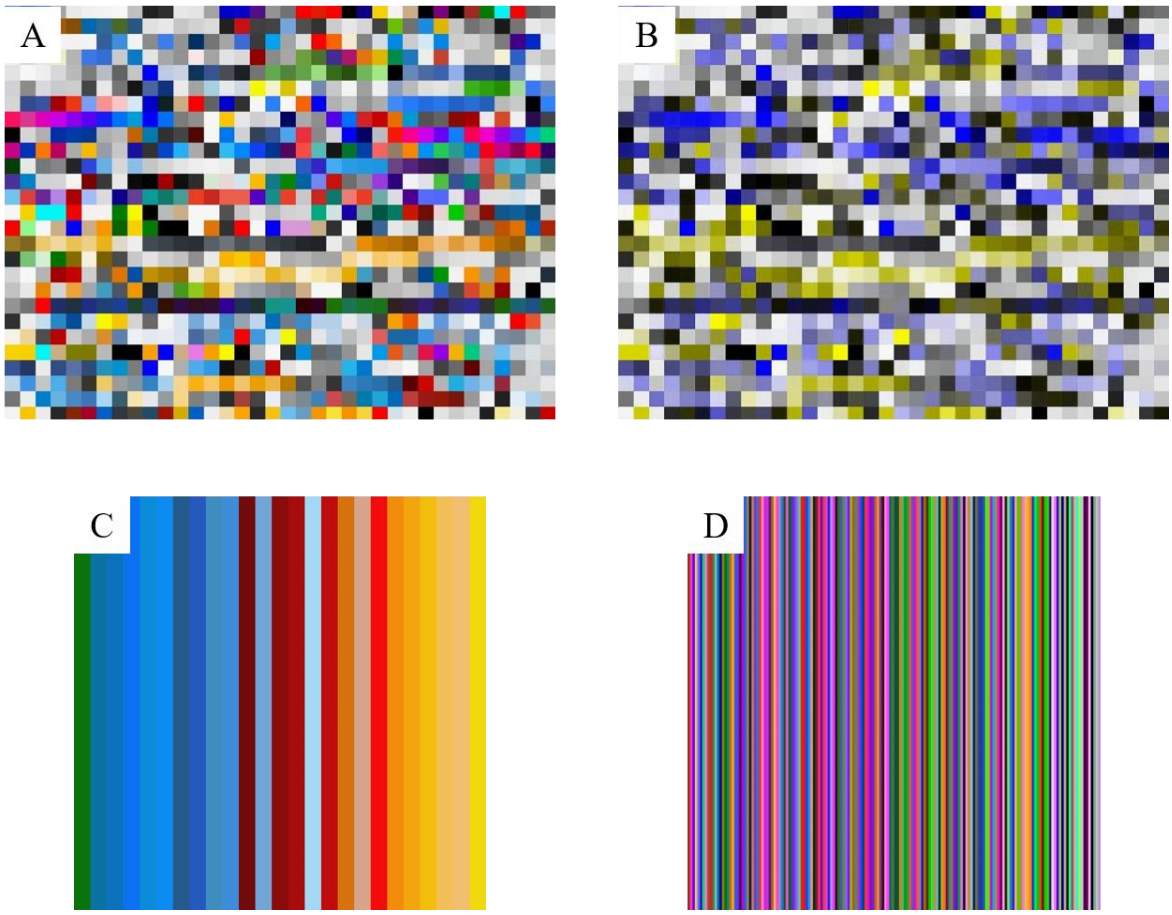

Figure 16: The most common colors of the web and key colors selected for a protanopic correction. (A) is the wild type vision image of the most common colors of the web; (B) is the protanopia vision simulation of (A); (C) is the 25 colors representing the 25 most populous bins of the difference histogram of (A) and (B); and (D) is a sample of the potential mappings for the leftmost color from (C)

Figure 16 features the most common colors on the web. The wild type vision and protanopia vision images are in 16A and 16B. 16C is the key colors that I selected from 16A. To make this selection, I used the same method of taking colors from the most populous bins in the difference histogram that I used to get the 50 key colors from Figure 13. I then created a series of mappings using different combinations of key colors to get a little under 200 potential mappings for each of the key colors. Figure 16D features a sample of potential mappings for the first of the 25 key colors, found on the far left of 16C. This serves to exemplify how reliant the mappings are on what colors are used during the optimization process.

From this set of potential mappings, I selected one color to be the mapping for each of the key colors. I did this by binning all the mappings into a 5x5x5 histogram, and then selecting the bin that had the most mappings. I then took the average of the mappings in that bin to use as the official mapping for that key color. Finally, I used the resultant mappings to recolor images for protanopes.

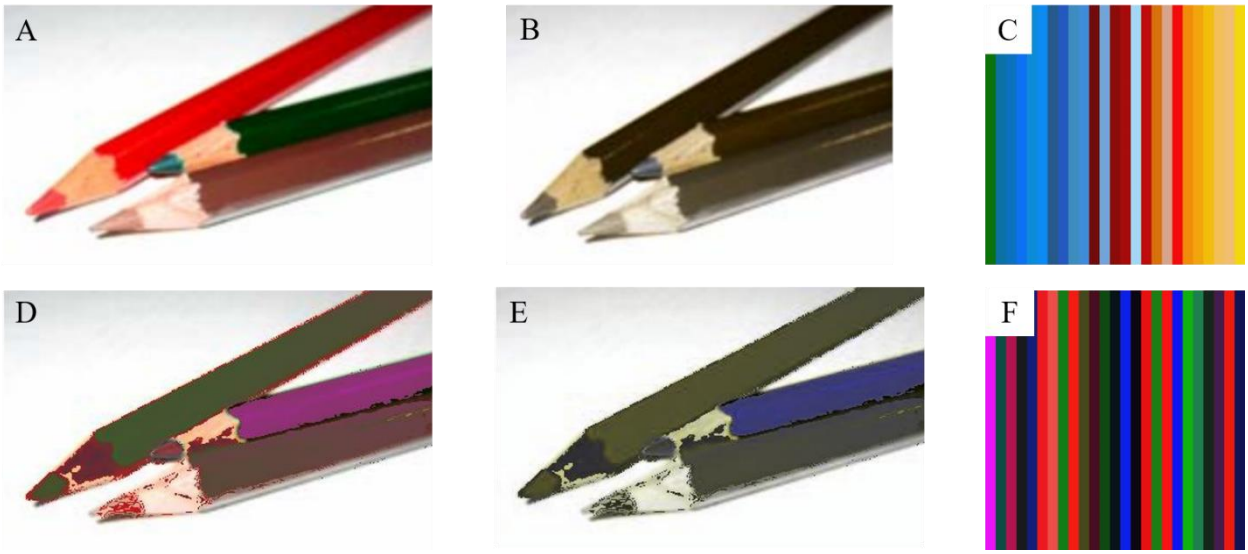

Figure 17: Protanopic correction of the image from Figure 3 using the key colors from the most common colors of the web. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is the 25 key colors from Figure 16; (D) is the wild type vision recoloring; (E) is the protanopia vision recoloring; and (F) is the mappings of the key colors used for the correction

The results of this generalized mapping can be found in Figure 17. 17A and 17B are the wild type vision and protanopia vision images from Figure 3. Figure 17C is the key colors that were selected from Figure 16A, with 17F being their respective mappings. 17D and 17E are the wild type vision and protanopia vision recolorings that I made using the key colors and their mappings. Unfortunately, despite the amount of work that this set of mappings took to create, they do not perform as well as the 50 key colors that I selected from Figure 13. You can see that the leftmost and rightmost pencils in 17E are far too similar compared to 17A. What's more, I still get the incorrect recoloring of the pencil wood that I struggled with in Figure 14.

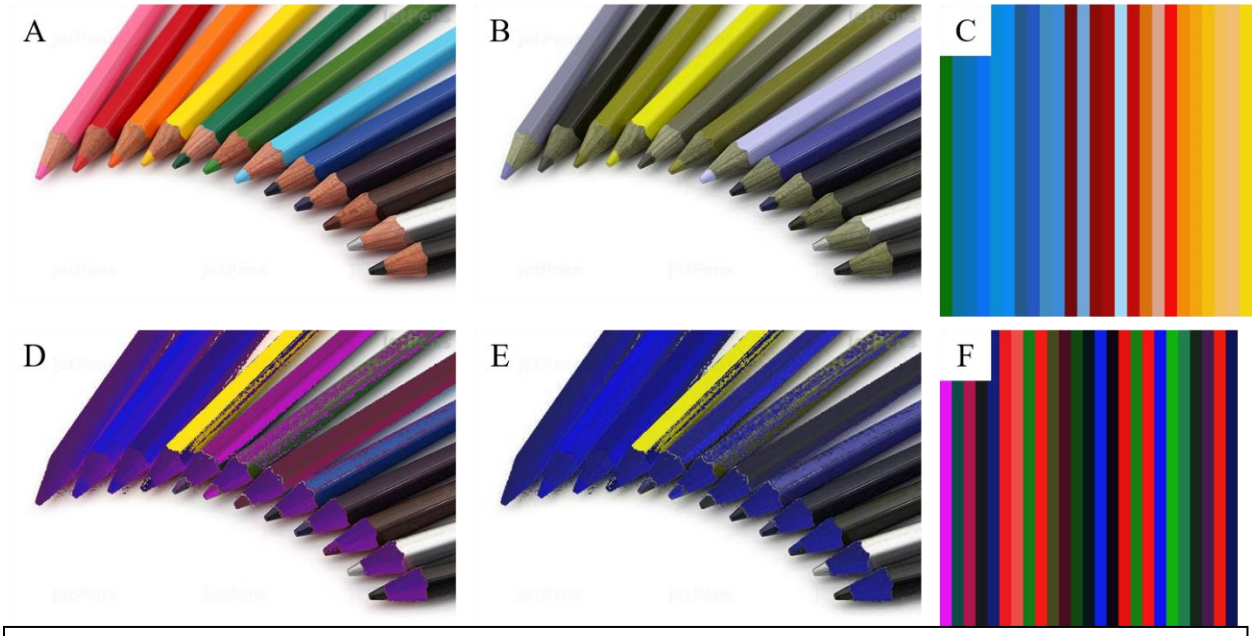

Figure 18: Protanopic correction of the image from Figure 6 using the key colors from the most common colors of the web. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is the 25 key colors from Figure 16; (D) is the wild type vision recoloring; (E) is the protanopia vision recoloring; and (F) is the mappings of the key colors used for the correction

As you can see in Figure 19, the generalized mapping from the most common colors on the web does not fare much better on an image with a larger variety of colors. 19A and 19B are the wild type vision and protanopia vision images from Figure 6. 18C and 18F are the same keys and mappings from 17E and 17F because these keys are meant to be used in more general cases. 18D and 18E are the wild type vision and protanopia vision recolorings using those keys and mappings. This process does produce varied key colors, but does not provide an adequate recoloring of the image. Although this process works better on images that have fewer colors present, it does not work as well as getting new key color mappings for each new image.

Section 7: Interpolation Caching

For the interpolation step, I took some liberties with the Shepard's algorithm that Jefferson and Harvey originally used. The Shepard's interpolation takes a lot of time to run because for each color there is a unique weight value that incorporates every color-key color

relationship, and then these must be summed. In order to make simple cachings, I ran the interpolation normally for the average color of each bin in that had at least one pixel in the difference histogram since those are the colors the interpolation actually runs on. Then, each time I wanted to recolor a certain color I would directly map its distance from the average color in its bin to the bin's mapping. So, if a pixel had a difference of 10 in the R value of its RGB from the average color in its bin, it's final recoloring's R value would also be 10 away from the average color's mapping. That way, I can run the actual interpolation once, and then just use these cached interpolations to use in the recoloring.

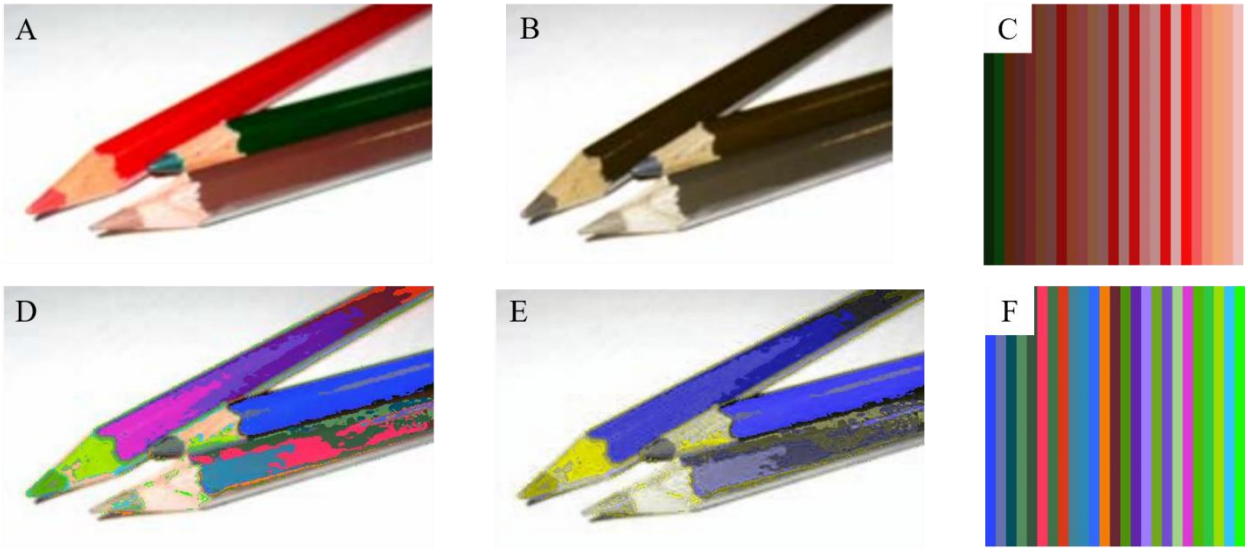

Figure 29: Protanopic correction of the image from Figure 3 using the caching interpolation method. (A) is the wild type vision image; (B) is the protanopia vision simulation of (A); (C) is the 25 most common colors from the difference histogram of (A) and (B) to be used in the correction; (D) is the wild type vision recoloring; (E) is the protanopia vision recoloring; (F) is the mapping of the key colors used to create cachings of the average color of each bin in the difference histogram

Figure 20 shows the results of this recoloring. Figure 19A and 19B are the wild type vision and protanopia vision images from Figure 3. 19C is the 25 most common colors from the difference histogram made from 19A and 19B, which I used as the key colors. Figure 19F is the mappings of the key colors in 19C. I made these mappings by running the normal optimization

process with weight values α_1=1.0 and α_2=0.5 from 10 random starting points. 19D and 19F are the wild type vision and protanopia vision images of the recoloring made with the caching process I previously described. You will note that Figures 19D and 19E are not as accurate in their recoloring as Figure 3G and 3H, but are still better than 19B in showing the color differences of the pencils, and could be made much faster because of the caching process.

Discussion

Section 1: What Did and Did Not Work

The version of the algorithm that I implemented had strengths and weaknesses that I was able to expose with my testing. In general, I found that images with a more diverse range of colors were better corrected by the algorithm. This is likely caused by the more varied key color selection. This hypothesis is further proved by the generalized mapping. In those tests, my first attempt—the attempt with the most key colors—performed the best.

This can account for many of the problematic recolorings I found. The deuteranopia recoloring from Figure 7 seemed to do very well, but the deuteranopia recoloring from Figure 10 failed miserable. However, Figure 7 had much more varied key colors, with pink, red, yellow and blue all well represented. Compare those keys to that of Figure 10, which is entirely comprised of shades of orange and brown. The same holds true for the tritanopic corrections from Figure 5 and Figure 11.

Finally, as stated above, this might explain why the first generalized mapping performed the best. Whenever I removed key colors from the correction, it made that correction worse. The best examples are Figures 14 and 15. By removing the variation that came with the presence of more key colors in Figure 15, I made the subsequent correction much worse than that of Figure 14.

Section 2: What Didn't Work

In the course of implementing the original algorithm and my attempts to improve it, I encountered a variety of issues. One point of contention was in the optimization step of the

original algorithm. The optimization I implemented was not the exact same version described by Jefferson and Harvey, as I was unable to implement a functional E_3 error equation. In their paper, they emphasize the importance of including this error to guarantee the proper mappings of certain colors, i.e. ensuring black and white get mapped to themselves since they are not affected by colorblindness. However, since black and white are not affected by colorblindness, they would not get changed by my interpolation. This section of the optimization step left me puzzled.

Ultimately, I was unable to match Jefferson and Harvey's results. My corrections were decent overall, but still inferior to the original. I the problem was in part that Jefferson and Harvey left crucial steps of their algorithm ambiguous, as I will detail in the next section. Another explanation of my occasionally unimpressive results was my more extensive testing. I was willing to test and show the strengths, and more importantly, the weaknesses of this algorithm.

The generalized mapping was also imperfect. The first test that I ran using the key colors from one image to make a recoloring for an image with less colors was the most successful of all my attempts. It would seem that further edits to the key colors either removed vital colors from the correction or added extra colors that muddied the correction. I still believe that a generalized mapping might be achievable, perhaps by creating a color palette using a set of generalized mappings.

Section 3: Research Obstacles

I encountered a few unexpected obstacles while attempting to recreate the results of Jefferson and Harvey's paper. During my initial readings of the paper, their algorithm seemed straightforward. However, as I began to recreate the algorithm, I realized that they leave many of

the finer points of the algorithm completely ambiguous, especially in the optimization and interpolation steps. Furthermore, their tests left a lot to be desired.

The ambiguity of their algorithm led to a few problems. For example, they do not make it clear just exactly the optimization is achieving the goal of matching the differences of the key colors. They tell us that $d(i, j)$ represents the difference between the mapped colors C_i and C_j. So, when I was recreating the optimization step, I gave the error function my set of RGB values that correspond to the key colors' new mappings and got the corresponding differences of those mapped colors. However, the program did not work as intended. I was taking the differences of my mapped colors as they were, when I was supposed to be simulating the effects of colorblindness on them, and then taking the differences. This is never so much as mentioned in Jefferson and Harvey's original paper.

My next problem with the Jefferson and Harvey paper is in how they describe the interpolation step. From my experience with this algorithm, the interpolation is extremely important since it determines what the final recoloring will look like. In spite of this, Jefferson and Harvey go into no detail in how they interpolate. All they tell us is that they used Shepard's method of interpolation, which takes the inverse distances of the points and squares them. That much is clear. However, they do not explain where they actually perform this interpolation. They preface their explanation of the interpolation step by saying that because of time constraints, they do not correct all the colors in an image. They then go on to give the formula for Shepard's interpolation for some given color, C, but we are never told what subset of colors they perform the interpolation on. For such a critical step for the success of the algorithm, we receive very little information on the process. Perhaps that is because it is self-explanatory, but despite my attempts at parsing the section and finding further reading to do on Shepard's interpolation, I

could not figure out what set of colors on which they performed the interpolation. As such, I only interpolated the colors that were in the difference histogram. My rationale was that since colors that were not in the difference histogram did not need to be re-colored since a colorblind viewer can see these colors just fine (what's more, the key colors explicitly ignore colors excluded from the difference histogram) it seemed logical to only adjust colors in the difference histogram. Although this method seems to work, I cannot be sure. After all, I could not perfectly recreate their tests with my algorithm.

Finally, I believe that the biggest oversight in the Jefferson and Harvey paper is the lack of tests. In their paper, they only show us images of colored pencils, with no more than 4 colors in any single image. I believe this is problematic in a few different ways. To start with, the use of only colored pencils does not give the reader a good idea of the effectiveness of their algorithm on images with colors that are not as simple as pencils. Although I think the colored pencils do serve their purpose as a simple yet effective way of showing how dramatically colors are affected by colorblindness, it is important to run the algorithm on other types of images as well. As you can see from my tests, the algorithm often gets confused on the borders of two colors. Since colored pencils have much more obvious color boundaries, I think that by only showing tests run on images of colored pencils, the reader does not get to see this weakness of the algorithm.

Putting aside the issue of only using one type of image in their testing, I also found the lack of colors in the testing images to be problematic. When I first decided to recreate Jefferson and Harvey's algorithm, I had a major question: what happens to colors that are unaffected by colorblindness? None of their test images show the user what happens when, say, the color blue is present in an image that is being corrected for a protanope or deuteranope. I was left

wondering what would happen to these colors. Frankly, I'm still not sure how they intended these unaffected colors to be handled since their interpolation explanation was problematic, as I explained above.

It is of note that the only published work from Jefferson and Harvey that is currently accessible are conference papers. Of course, conference papers can have a positive impact on the scientific community as they allow researchers to deliver information much faster than in scientific journals, they also do not undergo the rigorous review demanded of journal articles (Franceschet 2010, 129). In one 2010 study on the impact of conference papers and journal articles, conference papers were found that conference papers dominate popular topics, and yet journal articles are cited more often, meaning they might have a bigger impact on the community (131-2). This is not to discount the results of all conference papers, but I believe more research should be conducted to examine the vetting process of conference papers and their long-term effects on the scientific community.

Section 4: Further Work

If given more time, there are a few different avenues that I would want to explore to further improve the algorithm. First and foremost, I would want to explore more methods for improving the optimization step of the algorithm. There is the somewhat obvious path of creating more optimizations for more colors and testing those on different images. However, I would also want to explore tweaking the optimization. In my version of the optimization, the error function is minimized using the *fmin()* function from *scikit-learn*, which uses a simplex minimization. Likewise, Jefferson and Harvey used *fminunc()* from Matlab which also uses a simplex minimization (2006). One possibility for improvement would be experimenting with different optimization methods. Nelder-Mead optimization, which is what *fmin()* and *fminunc()* use, is an

easy to implement optimization method since it does not use derivatives. However, as Olsson and Nelson explain, there are some problems that the Nelder-Mead algorithm struggles with, such as problems with a large number of constraints (1975, 50). I might want to explore a different method of optimization that uses derivatives so that I could add more constraints to the error function, which could help prevent incorrect mappings.

Another improvement that can be made to the algorithm is to implement it in real time. Due to the current runtime of the algorithm, there would have to be some further improvements in order to make it work in real time. There might be a few other issues that may arise in a real-time implementation. For one, I'm concerned that despite generalized optimization that I used, colors might be assigned completely new mappings if they got slightly darker or lighter than they were originally. I would also be very wary of objects that change colors. I believe that it might be helpful to use some kind of object recognition to ensure that objects that change color are still mapped in such a way that a viewer can recognize that object as it moves.

Another avenue for further work would be to examine other methods of correcting colorblindness. My goal in this project was to improve upon an optimization-based approach, but there are other ways of correcting colorblindness. For example, Visolve is a software that helps ease the effects of colorblindness in real time by performing transformations on the colors that remain constant. Rather than making mappings for the colors in an image based on the problem colors, the Visolve algorithm transforms the problem colors in a constant way. Their Red-Green transform mode, for example, will make reds brighter and greens darker, thus making it easier to discern the two (Visolve n.d.). It produces much more consistent results than my algorithm, although it tends to not to preserve the distances of the colors, which was a key component of the algorithm I used.

An interesting niche that I would like to explore in the future is correcting rarer vision defects. In this project, I made corrections aimed chiefly at dichromatic users. I did not address the much rarer monochromatic variation of colorblindness. However, there are many other vision problems that I could explore. Retinal dystrophies, for example, are degenerative diseases that can lead to night blindness, partial blindness and eventually complete blindness (Nash et. al. 2015, FDA 2017). It could be fruitful to find ways to make computers easier to use for people with total colorblindness, partial vision loss or night/day blindness.

The final component that I would want to add to this project is feedback from the colorblind community. In many of the papers I examined in this project, there was a distinct lack of contact with the colorblind community. Although some of the research I found tested their results on people with colorblindness, none addressed what people with colorblindness actually want. I would like to gather information on what specifically is difficult about computer usage for colorblind users. Are there certain websites or activities that are more difficult than others? Are there quality of life problems that current filters do not address, such as filter intensity customization? Perhaps there is an important issue that faces colorblind computer users that I do not know about, and could not know about without either experiencing colorblindness myself or asking people in the community.

Conclusion

From my tests and findings, I do not believe an optimization-based approach is the best method of correcting colorblindness, especially if the aim is to make real-time corrections. The main obstacles to a real-time implementation are runtime and an inability to effectively handle color changes However, that is not to say that the method is entirely worthless. For instance, difference histograms are excellent illustrations of how colorblindness affects color information in images. In addition, I think the core motivation of the algorithm—that color differences should be maintained in a correction—is still worth pursuing. Further work should be done to combine the optimization-based filter's color contrasts with the speed of transformation-based filters, perhaps by creating a color palette derived from the optimization filter. Until then, the optimization method as it stands is not the most suited filter for colorblindness correction.

Bibliography

Brettel, Hans, Françoise Viénot, and John D. Mollon. 1997. “Computerized simulation of color appearance for dichromats.” *Journal of the Optical Society of America* 14, no. 10: 2647-2655. https://doi.org/10.1364/JOSAA.14.002647.

Clark, A. 2015. “Pillow (PIL Fork) Documentation.” Readthedocs. https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

Colour Blind Awareness. n.d. “Colour Blindness.” Colour Blind Awareness. Accessed 9 May 2020. https://www.colourblindawareness.org/colour-blindness/.

EnChroma. n.d. “How EnChroma Glasses Work.” EnChroma. Accessed 13 August 2020. https://enchroma.com/pages/how-enchroma-glasses-work.

Fidanger, Onur, Poliang Lin, and Nevran Ozguven. 2014. “Daltonize.” Github. Last modified 10 November 2020. https://github.com/joergdietrich/daltonize.

Fischer, M. Dominik, Stylianos Michalakis, Barbara Wilhelm, Ditta Zobor, Regine Muehlfriedel, Susanne Kohl, Nicole Weisschuh, et al. 2020. “Safety and Vision Outcomes of Subretinal Gene Therapy Targeting Cone Photoreceptors in Achromatopsia: A Nonrandomized Controlled Trial.” *JAMA Ophthalmology* 138, no. 6 (2020): 643-651. https://doi.org/10.1001/jamaophthalmol.2020.1032.

Food and Drug Administration. 2017. “FDA approves novel gene therapy to treat patients with a rare form of inherited vision loss.” Food and Drug Administration. Accessed 15 August 2020. https://www.fda.gov/news-events/press-announcements/fda-approves-novel-gene-therapy-treat-patients-rare-form-inherited-vision-loss.

Franceschet, Massimo. 2010. “The role of conference publications in computer science: a bibliometric view.” *Communications of the ACM* 53, no. 12: 129-132. https://dl.acm.org/doi/10.1145/1859204.1859234.

Gómez-Robledo, L., E. M. Valero, R. Huertas, M. A. Martínez-Domingo, and J. Hernández-Andrés. 2018. “Do EnChroma glasses improve color vision for colorblind subjects.” *Optics Express* 26, no. 22: 28682-28692. https://doi.org/10.1364/OE.26.028693.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. “Array programming with NumPy.” *Nature* 585: 357–362. https://doi.org/10.1038/s41586-020-2649-2.

Hebert, Paul. 2016. “The Colors Used by the Ten Most Popular Sites.” Paul Hebert Designs. Last modified 18 September 2016. http://paulhebertdesigns.com/web_colors/index.php.

Hunt, David M., Kanwaljit S. Dulai, James K. Bowmaker, and John D. Mollon. 1995. “The Chemistry of John Dalton’s Color Blindness.” *Science* 267, no. 5200: 984-988. https://doi.org/10.1126/science.7863342.

Hunter, John D. 2007. “Matplotlib: A 2D Graphics Environment.” *Computing in Science and Engineering* 9, no. 3: 90-95. https://doi.org/10.1109/MCSE.2007.55.

Jefferson, Luke and Richard Harvey. 2006. “Accommodating Color Blind Computer Users.” Paper presented at the *Conference on Computers and Accessibility, Portland, OR, 23-25 October 2006*. https://doi.org/10.1145/1168987.1168996.

Jefferson, Luke and Richard Harvey. 2007. “An Interface to Support Color Blind Computer Users.” Paper presented at the *Computer/Human Interaction Conference, San Jose, CA 28 April-3 May 2007*.

Kulshrestha, Ruchi and R. K. Bairwa. 2013. “Review of Color Blindness Removal Methods using Image Processing.” *International Journal of Recent Research and Review* 6: 18-21. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.410.3304.

Lee, Barry B. 2008. “The evolution of concepts of color vision.” *Neurociencias* 4, no.4: 1-23. Accessed 7 May 2020. https://www.ncbi.nlm.nih.gov/pubmed/21593994.

Maxwell, James Clerk. 1865. “Experiments on colour as perceived by the eye, with remarks on colour-blindness.” In *Transactions of the Royal Society of Edinburgh, Vol. XXI, Part II*, 275-297. Edinburgh: Neill.

Microsoft Corporation. n.d. “Windows 10 Manual.” Windows 10 Guide. Microsoft Corporation. Accessed November 26 2020. https://windows10-guide.com/windows-10-manual-pdf.

Nash, Benjamin M., Dale C. Wright, John R. Grigg, Bruce Bennetts, Robyn V. Jamieson. 2015. “Retinal dystrophies, genomic applications in diagnosis and prospects for therapy.” *Translational Pediatrics* 4, no. 2: 139-63. https://doi.org/10.3978/j.issn.2224-4336.2015.04.03.

Neitz, Maureen and Jay Neitz. 1998. “Molecular Genetics and the Biological Basis of Color Vision.” In *Color Vision: Perspectives from Different Disciplines*, edited by Werner G. K. Backhaus, Reinhold Kliegl and John. S. Werner, 101-117. Berlin: Walter de Gruyter.

Olsson, Donald M. and Lloyd S. Nelson. 1975. “The Nelder-Mead Simplex Procedure for Function Minimization.” *Technometrics* 17, no. 1: 45-51.

Ridpath, Chris and Wendy Chisholm. “Techniques for Accessibility Evaluation and Repair Tools.” World Wide Web Consortium. Published 26 April 2000. http://www.w3.org/TR/AERT.

Ryobi Systems Co., Ltd. n.d. “What is Visolve.” Visolve. Ryobi Systems Co., Ltd. Accessed August 13 2020. https://www.ryobi-sol.co.jp/visolve/en.

Tanuwidjaja, Enrico, Derek Huynh, Kirsten Koa, Calvin Nguyen, Churen Shao, Patrick Torbett, Colleen Emmenegger, and Nadir Weibel. 2014. “Chroma: A Wearable Augmented-Reality Solution for Color Blindness.” Paper presented at the *ACM International Joint Conference on Pervasive and Ubiquitous Computing, Seattle, WA, September 2014*. https://doi.org/10.1145/2632048.2632091.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski et al. 2020. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.” *Nature Methods* 17, no. 3: 261-272. https://doi.org/10.1038/s41592-019-0686-2.

Young, Thomas. 1802. “The Bakerian Lecture. On the Theory of Light and Colours.” In *Philisophical Transactions of the Royal Society*, 12-48. London: G. and G. Nicol.

Zeltzer, Harry I. 1970. Method of Improving Color Discrimination. US Patent 49,582, filed 24 June 1970, and issued 22 June 1971.