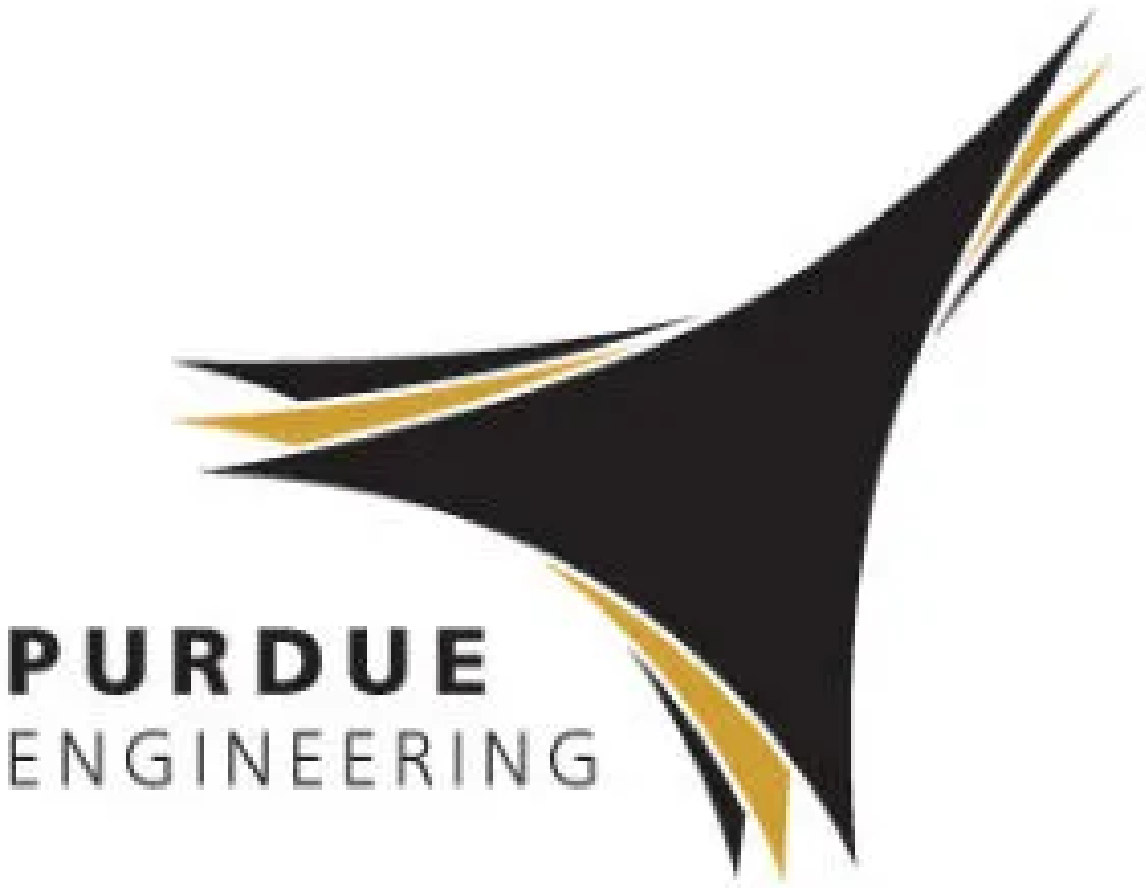


Colorblind Filter

Individual Project

ENGR 133 LC2-23

By Jamie Henson

Table of Contents

Abbreviation/Acronym	Definition
RGB	Red, blue, and green values of an image matrix
LMS	Long, medium, and short values of an image matrix

Section 1: Introduction to the Program

1.1 What does Colorblind Filter do?

Colorblind Filter aids those suffering from colorblindness and simulates the experience of colorblindness to those with normal vision. This program uses a color correction technique that rearranges colors so that one with color blindness can see colors more clearly, vibrantly, and discretely. This technique is known as Daltonization, named after John Dalton, a notable English chemist and physicist of the modern atomic theory. Colorblind as he is, he contributed to research and proposed an algorithm that shifts colors away from confusion lines towards colors visible to the colorblind individual. Thus, his study and algorithms on colorblindness will be used in his honor for this program.

1.2 Motivations for the project

According to the National Eye Institute, color blindness is the inability of one to distinguish specific colors. In anatomy, it could be defined as a genetic condition where one is missing the necessary cells in the retina that sense particular colors. Unfortunately, whether the symptoms of color deficiency can be mild or severe, there is no cure for color blindness. Thus, lacking the ability to differentiate between specific colors is a widespread problem worldwide. Colorblind people face difficulties in nearly every aspect of their lives. As someone with relatives with certain types of colorblindness, problems can arise from the simplest activities that anyone with normal vision often overlooks, such as cooking, driving a car, or simply deciding which clothes to wear. Thus, with the Colorblind filter, making an image or video more accessible for those with this genetic disability can improve their lives tremendously.

1.3 How is the Colorblind Filter related to my interests?

As a freshman engineering student at Purdue Univerisity, I have always wanted to be a pilot for nearly my entire life. Aviation and engineering have always fascinated me and are why I wanted to pursue aerospace engineering at Purdue. However, colorblind pilots can only fly under certain conditions, specifically with a copilot, and only during the day. With this program, Colorblind Filtering, one could see a converted image or live videos of what a color-blind person would see and help colorblind people, especially pilots, differentiate specific colors of an image.

Section 2: Description of inputs and outputs

The first thing the Color Blind Filter will do is greet the user. However, the program is split into two main tasks: recording a live video or processing an image. Thus, the first input the user will make in the program will be “video” or “image,” depending on whether the user wants a filtered video or image.

```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)?
```

Figure 2.0.1: The program will ask whether the user wants a live video or image.

2.1 Image Option Input

If the user selects the image option of the program, the program will give you three colorblind types to choose from (Deuteranopia. Protanopia, and Tritanopia). Then, the program will ask for an image, preferably a jpg, jpeg, or png, or it will return an error as stated in section 2.5, Error Handling.

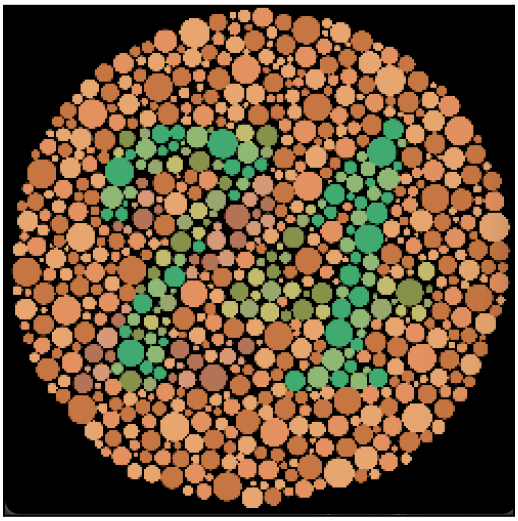
```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)? image
These are the options:
 Deuteranopia (Green absence)
 Protanopia (Red absence)
 Tritanopia (Blue-Yellow absence)

What type of Colorblindess are you looking for? protanopia
What is the name of your image? 74.png
```

Figure 2.1.1: The program will ask for an image after you input the type of color blindness.

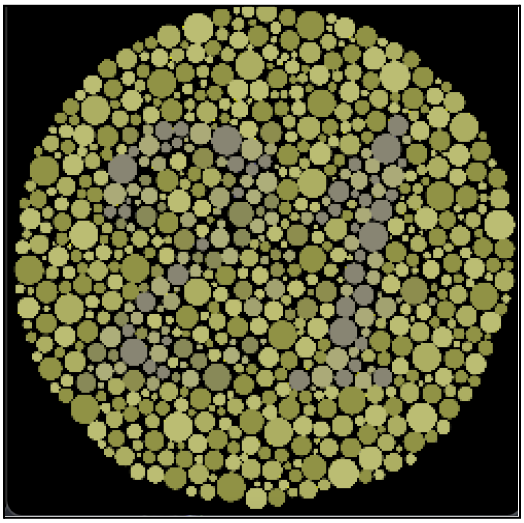
2.2 Image Option Output

Depending on the type of colorblind you choose, the program will display four images as the output using cv2—the original image, the simulated image, the daltonized image, and then the daltonized simulated image.

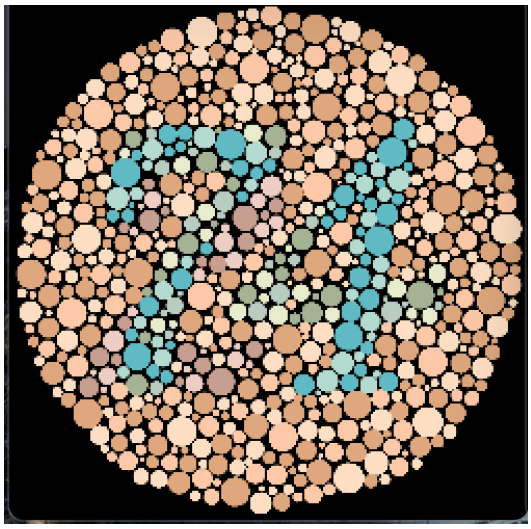
Original Image



Simulated Image



Daltonized Image



Daltonized then Simulated Image.

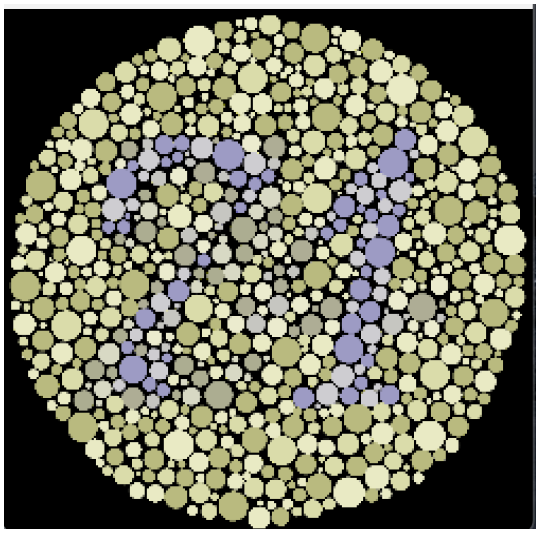


Figure 2.2.1: Image output for Colorblind filter of protanopia.

2.3 Video Option Input

If the user selects the video option of the program, the program will give you three options of which colorblind types to choose from as well (Deuteranopia. Protanopia, and Tritanopia). However, because the program can only access one loop on the webcam, it can only play one video at a time. Thus, it will have to give you the decision to display either a live simulated video or a daltonized video.

```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)? video
These are the options:
 Deuteranopia (Green absence)
 Protanopia (Red absence)
 Tritanopia (Blue-Yellow absence)

What type of Colorblindess are you looking for? tritanopia
Do you want a simulated or daltonized video? simulated
```

Figure 2.3.1: Video input of Colorblind filter for tritanopia.

2.4: Image Option Output

Then, regarding the type of colorblindness the user chooses, and if the user prefers seeing the simulated or daltonized colorblind filter, the program will enable the webcam to display a live video with a simulated or daltonized filter applied with the color blindness type respectively.

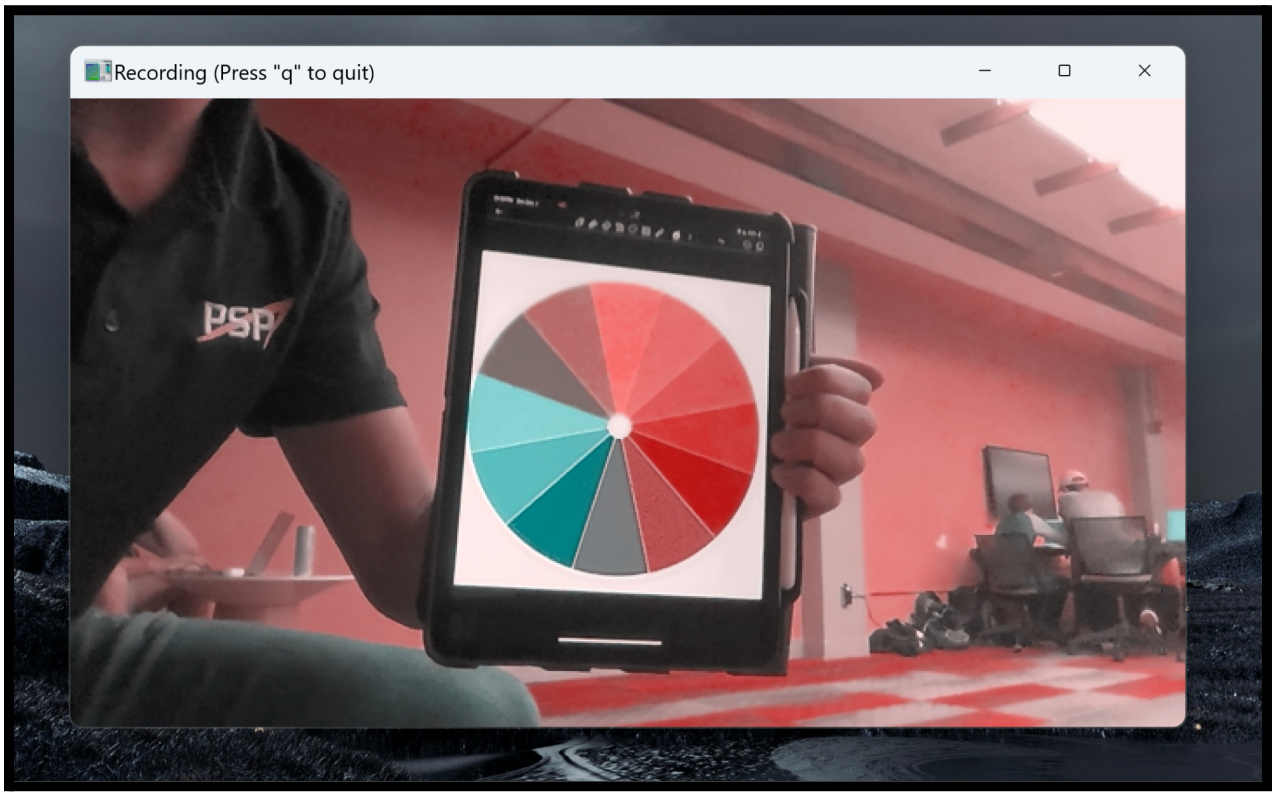


Figure 2.4.1: As depicted in the image above, a simulated depiction of tritanopia was processed in a live webcam video. The video was taken at Shreve Hall at Purdue University.

2.5 Error Handling

The program has several instances of error handling when the user enters an unintended input. For cases when the user inputs the wrong option or image choice, the program will call the restart function, which prompts the user if they want to rerun the program.

```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)? neither
Please enter either 'video' or 'image'.

Do you want to rerun the program (y or n)?
```

Figure 2.5.1: Case for entering incorrect input when prompted to choose either image or video demonstration.

```
These are the options:
 Deuteranopia (Green absence)
 Protanopia (Red absence)
 Tritanopia (Blue-Yellow absence)

What type of Colorblindess are you looking for? monochromatism
You must enter a type of blindness as listed above.
Do you want to rerun the program (y or n)?
```

Figure 2.5.2: Case where the user enters the wrong input of the type of color blindness.

```
What is the name of your image? nonexistant.jpg
[ WARN:0@232.519] global loadsave.cpp:248 cv::findDecoder imread_('nonexistant.jpg'):
ck file path/integrity
An unexpected error has occured while loading 'nonexistant.jpg'.

Do you want to rerun the program (y or n)?
```

Figure 2.5.3: Case where the user enters an invalid or nonexistent image file as the input.

```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)? video
These are the options:
 Deuteranopia (Green absence)
 Protanopia (Red absence)
 Tritanopia (Blue-Yellow absence)

What type of Colorblindess are you looking for? protanopia
Do you want a simulated or daltonized video? neither unfortunately
Please enter either 'simulated' or 'daltonized' as input here.

Do you want to rerun the program (y or n)?
```

Figure 2.5.4: Case where the user enters the incorrect input when prompted to choose a simulated or daltonized live video.

If the user fails to enter either a yes or no answer when asked if they want to restart the program, the program will prompt them to answer again until they enter the correct input.

```
Please enter either 'video' or 'image'.

Do you want to rerun the program (y or n)? a
Do you want to rerun the program (y or n)? b
Do you want to rerun the program (y or n)? c
Do you want to rerun the program (y or n)? d
Do you want to rerun the program (y or n)? e
Do you want to rerun the program (y or n)? n
Ending program, thank you for using Jamie's Colorblind filter
```

Figure 2.5.5: Case where the user enters an input that is not “y” or “n” when prompted to choose whether to rerun the program.

Section 3: User-Defined Functions

3.1 Simulate Function

The process of a colorblind simulator heavily relies on image transformations through the cv2 module. It is called with input as the original or daltonized image and then transformed into a simulated version of that image depending on which type of colorblindness was chosen. Using the kernels in Figure 3.1.1, the image is transformed using cv2. Then, the RGB values of the image are clipped to a certain range from 0 to 255 using *np.clip* and *astype(np.uint8)* and are returned to the main Python file.

Integrated Computer-Aided Engineers at the University of Pernambuco in Brazil researched the algorithms for altering the specific colors of an image in such a way that it can simulate real color blindness. A huge disclaimer is that color blindness simulations rely on imperfect theories and models. Nevertheless, the numpy matrixes depicted below were used in image transformation of certain types of colorblindness.

Deuteranopia Kernel

$$\begin{Bmatrix} 0.55 & 0.45 & 0 \\ 0 & 0.45 & 0.45 \\ 0 & 0.525 & 0.475 \end{Bmatrix}$$

Protanopia Kernel

$$\begin{Bmatrix} 0.95 & 0.05 & 0 \\ 0 & 0.65 & 0.35 \\ 0 & 0.675 & 0.325 \end{Bmatrix}$$

Tritanopia Kernel

$$\begin{Bmatrix} 0.567 & 0.433 & 0 \\ 0.558 & 0.442 & 0 \\ 0 & 0.242 & 0.758 \end{Bmatrix}$$

Figure 3.1.1: The three matrices for image transformation

The complex mathematics behind these matrixes are described in the article cited in Section 5.2: References.

3.2 Daltonize

The process of daltonizing an image in Python is more convoluted than simulating an image. In this function, daltonization.py attempts to adjust the colors of an image such that fewer color combinations would confuse a colorblind individual.

The human eye contains three types of cells responsible for interpreting different wavelengths of light. These cells, known as cones, consist of red, green, and blue. However, these wavelengths can be interpreted by their sensitivity to certain wavelengths of light, where short, medium, and long wavelengths correspond to blue, green, and red cones, respectively.

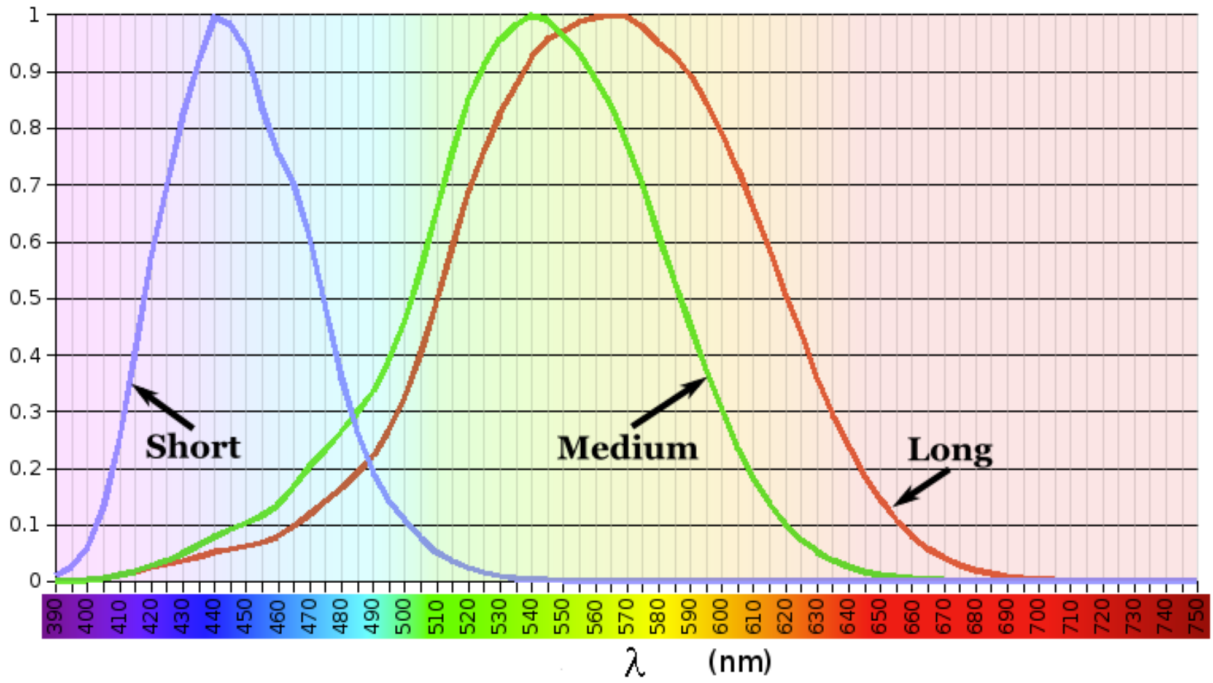


Figure 3.2.1: The above chart plots the simulation levels of cones exposed to different wavelengths.

Therefore, the first step is to remove gamma correction in the image. By doing that, we can rearrange the intensities of the RGB values from [0, 255] to [0, 1] to make the image more uniform to humans.

$$v' = \begin{cases} \left(\frac{\frac{v}{255}}{12.92}\right) & v <= 0.04045 \times 255 \\ \left(\frac{\frac{v}{255}+0.055}{1.055}\right)^{2.4} & v > 0.04045 \times 255 \end{cases}$$

Figure 3.2.2: Formula for removing gamma correction

Next, the image is transformed using *np.dot,* a function used for dot product multiplication in numpy arrays. Thus, the LMS image is transformed by this dot product multiplication. A color blindness daltonization process can be referred to as simply matrix multiplication to transform images.

$$T^{-1}ST\begin{bmatrix} r_c \\ g_c \\ b_c \end{bmatrix} = \begin{bmatrix} r_{c'} \\ g_{c'} \\ b_{c'} \end{bmatrix}$$

Figure 3.2.3: Image daltonization can be simply matrix multiplication to transform *c* to a daltonized color *c'.*

As depicted in Figure 3.2.3, $T^{-1}ST$ represents the transformation matrix, or kernel, necessary to daltonize a certain type of color blindness. James Schmitz, an expert on color science himself, has an in-depth explanation of the calculation behind these certain kernels that one can reference by going to section 5.2: References. The specific kernels used for the program are provided below:

Deuteranopia Kernel,

$$\begin{Bmatrix} 0.9 & 0.4 & 0 \\ 0 & 0.8 & 0.5 \\ -0.3 & 0.2 & 1 \end{Bmatrix}$$

Protanopia Kernel

$$\begin{Bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0.25 \\ -0.185 & 0.315 & 1 \end{Bmatrix}$$

Tritanopia Kernel

$$\begin{Bmatrix} 1 & 0.183 & 0 \\ 0 & 0.927 & 0.073 \\ 0 & 0 & 1 \end{Bmatrix}$$

Figure 3.2.4: The three matrices for image daltonization

Then, gamma correction can be later reapplied with the inverse of the formula used to remove gamma correction initially.

$$v = \begin{cases} 255\left(12.92v'\right) & v' <= 0.0031308 \\ 255\left(1.055v'^{0.41666} - 0.055\right) & v > 0.0031308 \end{cases}$$

Figure 3.2.5: Formula to reapplying gamma correction

Finally, just like simulate.py, the RGB values of the image are clipped to a certain range from 0 to 255 using *np.clip* and *astype(np.uint8).* The daltonized version of the original image is then returned back to the main Python file.

Section 4: User Manual

4.1 Procedure

Step 1). After Colorblind Filter welcomes you, you can pick the program to run a live video through the webcam or import an image. You need to input “video” or “image,” respectively, depending on your choice. If you choose to enter an image, you can skip steps 2 through 4.

```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)?
```

Step 2). If you choose to display a video, the program will prompt you to enter one of the three types of color blindness available (Deuteranopia, Protanopia, and Tritanopia). As an example, the deuteranopia was entered.

```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)? video
These are the options:
 Deuteranopia (Green absence)
 Protanopia (Red absence)
 Tritanopia (Blue-Yellow absence)

What type of Colorblindess are you looking for?
```

Step 3). The program will give you the option to choose if you want a simulated or daltonized video to display. For this example, simulation was entered.

```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)? video
These are the options:
 Deuteranopia (Green absence)
 Protanopia (Red absence)
 Tritanopia (Blue-Yellow absence)

What type of Colorblindess are you looking for? deuteranopia
Do you want a simulated or daltonized video?
```

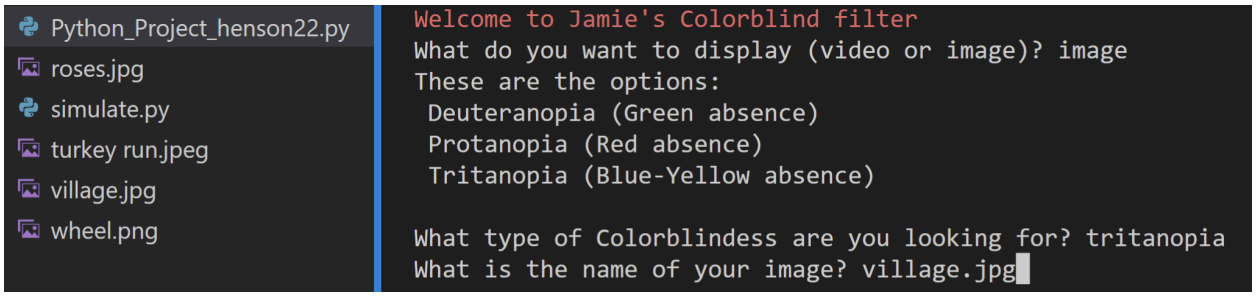
Step 4). Finally, your monitor should be able to display a live video with the Colorblind Filter applied. Once you press ‘q’ to end the video, the program will ask you if you want to rerun it.

Figure 4.1.1: Video taken at Wilmeth Active Learning Center (WALC) in Purdue Univerisity.

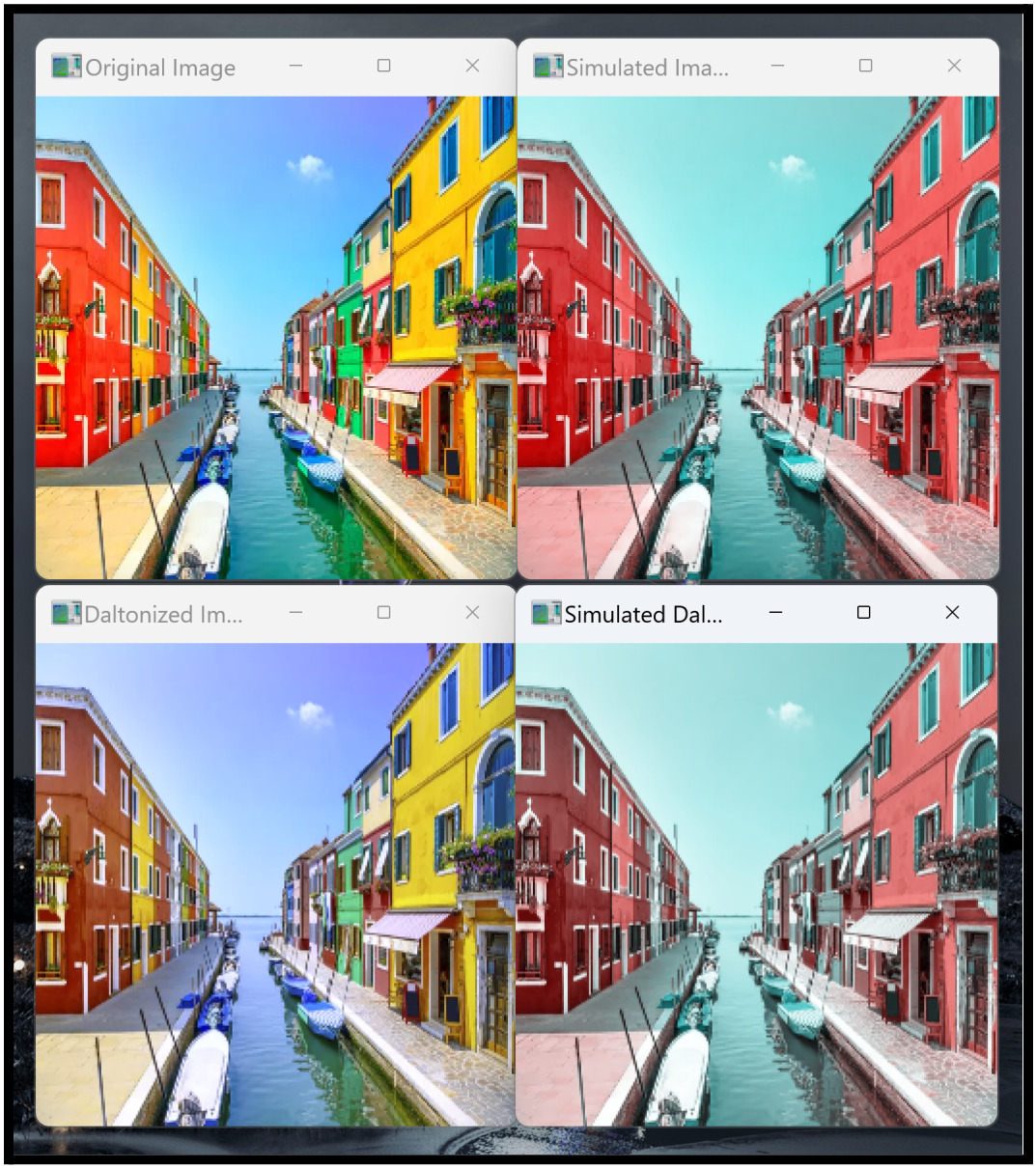
Step 5). If you prefer having an image as an input and output, then the first thing the program will prompt you to do is enter one of the three available types of colorblindness as well. For this example, we will use tritanopia.

```
Welcome to Jamie's Colorblind filter
What do you want to display (video or image)? image
These are the options:
 Deuteranopia (Green absence)
 Protanopia (Red absence)
 Tritanopia (Blue-Yellow absence)

What type of Colorblindess are you looking for?
```

Step 6). Then, the program will ask you to input the name of an image. It is essential to have the image file in the same folder where the main Python file is located. Avoid importing a file that isn't a jpg, jpeg, or pdf into the Colorblind Filter. One can save an image online from any platform, such as Google, and save it to the same folder where the main Python file is located. For this instance, *village.jpg* will be imported.



Step 7). Finally, the program will display four images. The original image, the simulated image, the daltonized image, and the daltonized then simulated image. Press any key to quit. You will also be given the choice to rerun the program if needed.

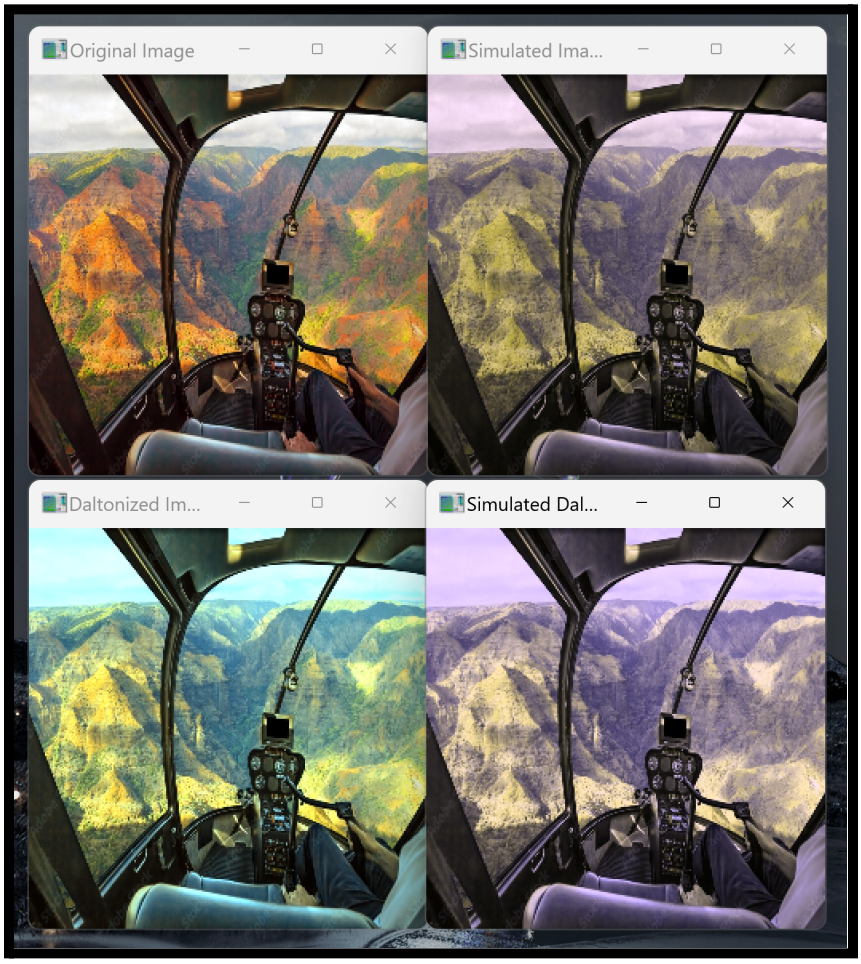


4.2 Demonstration

Disclaimer: The images provided in examples 1, 2, and 3 are provided online by Getty Images referenced in Section 5.2: References. The videos in examples 4 through 9 are taken in the WALC library.

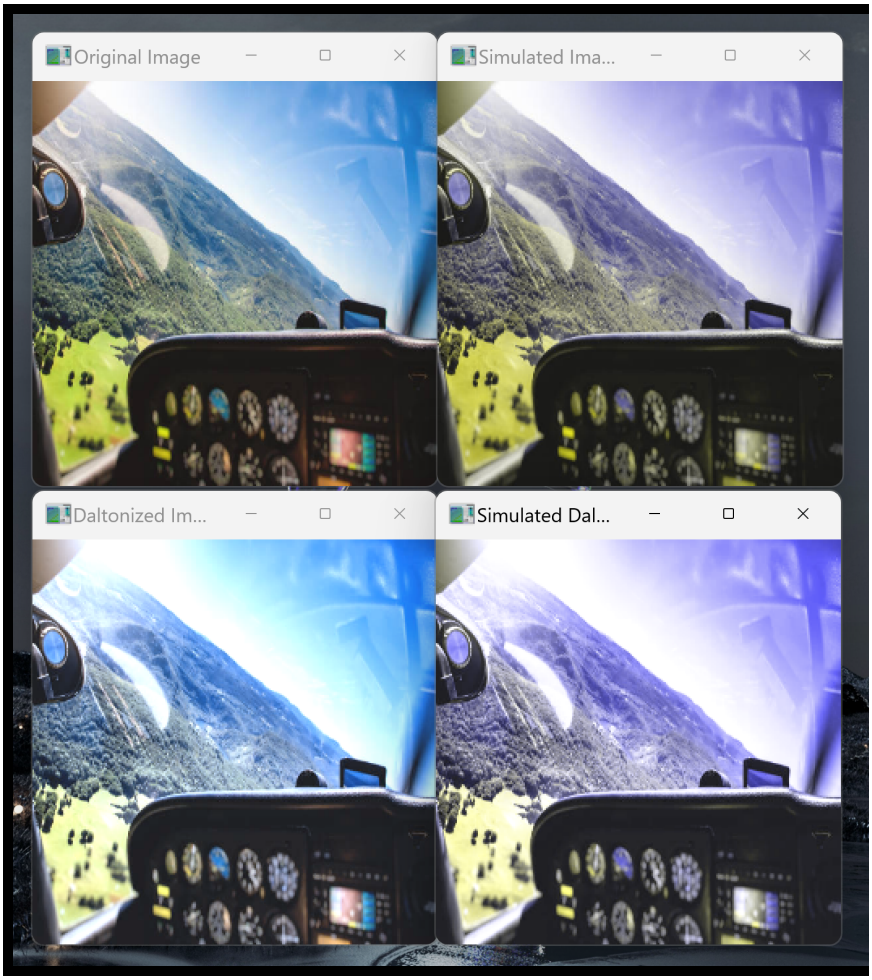
Example 1

Image > Deuteranopia > cockpit.jpg



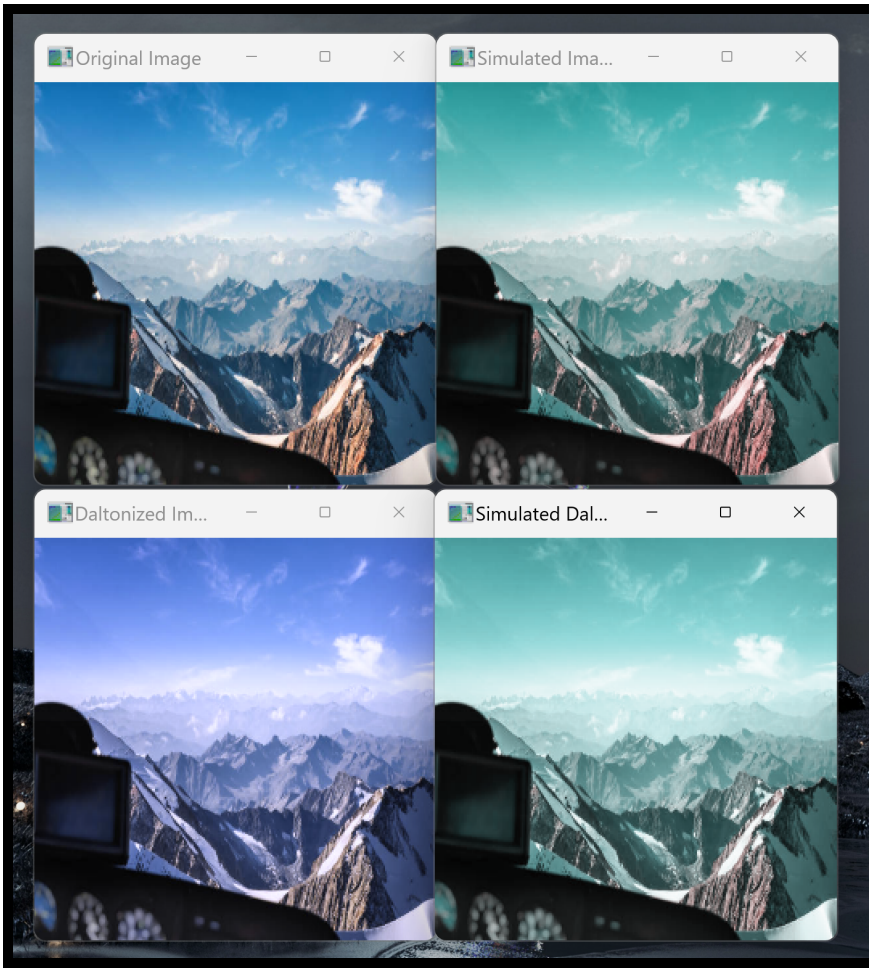
Example 2

Image > Protanopia > cockpit2.jpg



Example 3

Image > Tritanopia > cockpit3.jpg

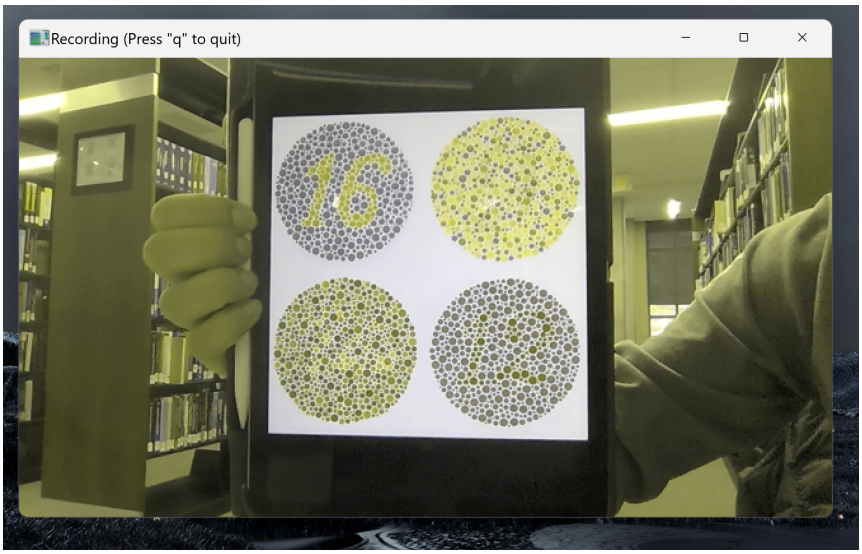


Example 4

Video > Deuteranopia > Simulate

Example 5

Video > Protanopia > Simulate

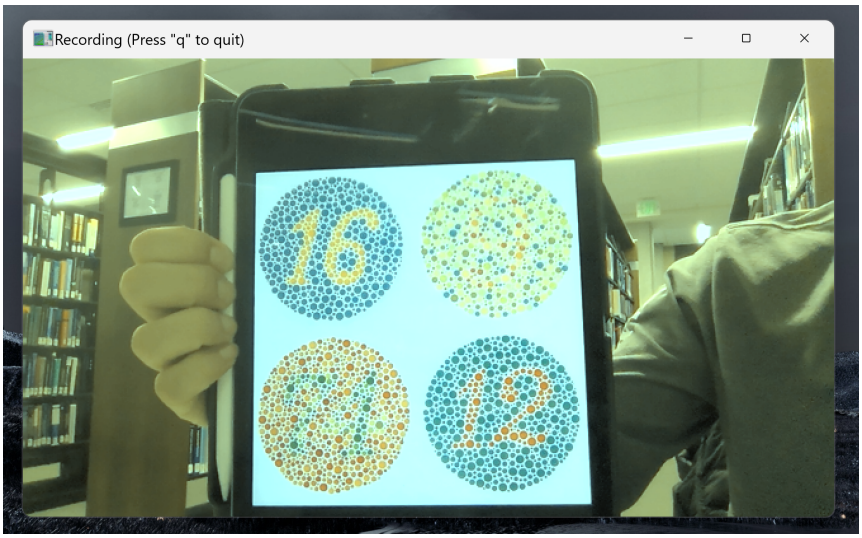


Example 6

Video > Tritanopia > Simulate

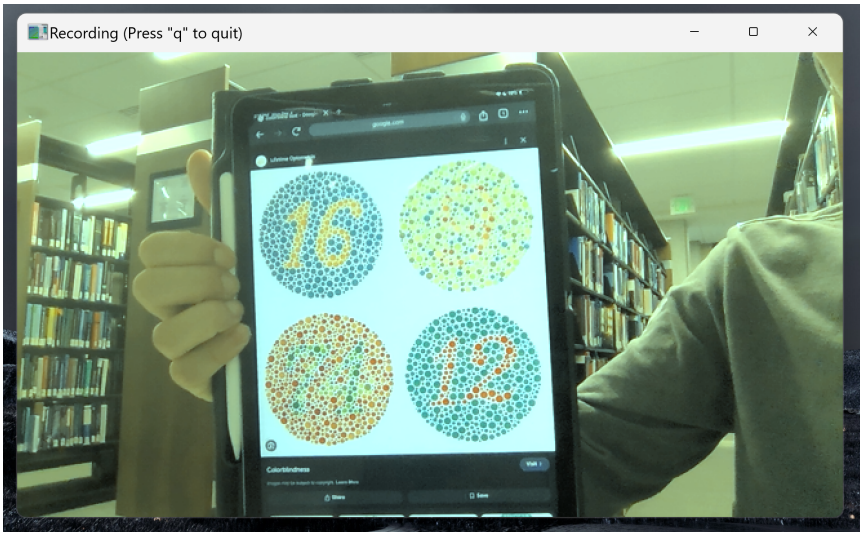
Example 7

Video > Deuteranopia > Daltonize



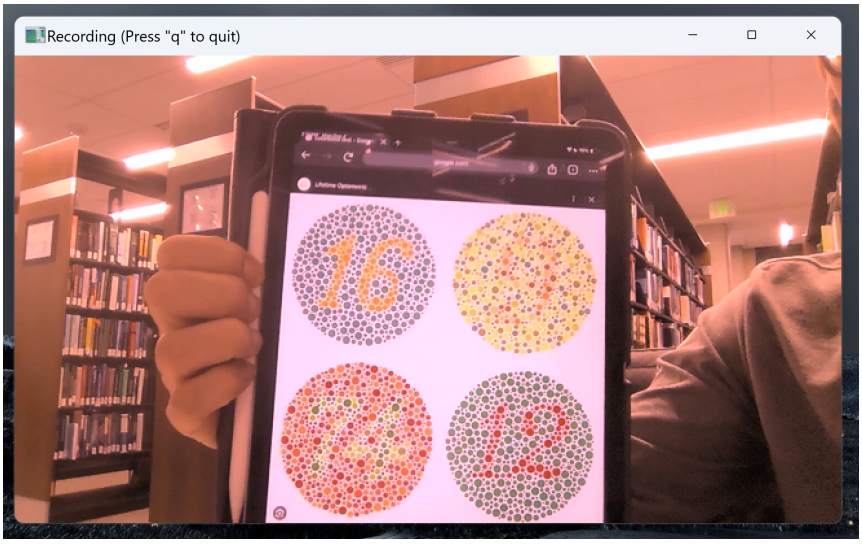
Example 8

Video > Protanopia > Daltonize



Example 9

Video > Tritanopia > Daltonize



4.3 Conclusion

Many color-blind people can live all their lives without knowing they have color vision disabilities and abnormalities. Nowadays, the evolution of information technology and computer science, specifically image processing techniques and computer graphics, can be fundamental in developing adaptive color blindness correction tools. Although the Colorblind filter isn't perfect in assisting color anomalies, hopefully, it will benefit the field of color science and the fight against colorblindness. I hope you enjoyed using my Colorblind filter.

Section 5: Appendix

5.1 Python Code

5.1.1 Python_Project_henson22.py

```
"""
================================================================================
=====
ENGR 13300 Fall 2023

Program Description
    Filtering images or live videos with different color contrast to make
them
    more colorblind friendly. Through a process known as daltonization,
converting
    frames of what a color-blind person would see and help colorblind
people,
    especially pilots, differentiate specific colors of an image or live
video.

Assignment Information
    Assignment:     Python Individual Project
    Author:         Jamie Henson, henson22@purdue.edu
    Team ID:        LC2 - 23 (e.g. LC1 - 01; for section LC1, team 01)

Contributor:    Name, login@purdue [repeat for each]
    My contributor(s) helped me:
    [ ] understand the assignment expectations without
        telling me how they will approach it.
    [ ] understand different ways to think about a solution
        without helping me plan my solution.
    [ ] think through the meaning of a specific error or
        bug present in my code without looking at my code.
    Note that if you helped somebody else with their code, you
    have to list that person as a contributor here as well.

ACADEMIC INTEGRITY STATEMENT
```

```
I have not used source code obtained from any other unauthorized
source, either modified or unmodified. Neither have I provided
access to my code to another. The project I am submitting
is my own original work.
================================================================================
=====
"""

# Import following modules
import numpy as np
import cv2
from simulate import simulate
from daltonize import daltonize


def main():
    # Greet the user
    print("\033[31mWelcome to Jamie's Colorblind filter\033[0m")

    # Ask the user if they want to display a video or image
    choice = input("What do you want to display (video or image)?
").lower()

    # Check if the user entered a valid choice
    if choice != "video" and choice != "image":
        print("Please enter either 'video' or 'image'.\n")
        restart()

    # Display the options ask what colorblind type they are looking for
    print("These are the options: \n Deuteranopia (Green absence) \n
Protanopia (Red absence) \n Tritanopia (Blue-Yellow absence)\n")
    blindtype = input("What type of Colorblindess are you looking for? ")

    # Check if the blindtype user inputed is valid and exists
    blindtype = blindtype.lower()
    if blindtype != "deuteranopia" and blindtype != "protanopia" and
blindtype != "tritanopia":
        print("You must enter a type of blindness as listed above.")
        restart()
```

```
    # Conditional depending if the user wants an image or a video
    if choice == "image":
        # Ask the user for an image
        image_name = input("What is the name of your image? ")

        # Try to import the name of the image file
        image = load(image_name)

        # Tell the user to wait
        print("\033[31m... Generating Images ...\033[0m")

        # Convert the image to a colorblind simulated version of the
image.
        simulated_image = simulate(image, blindtype)

        # Convert the image to a colorblind friendly version of the image.
        daltonized_image = daltonize(image, blindtype)

        # Convert the daltonized image into a colorblind simulated version
of the image.
        new_image = simulate(daltonized_image, blindtype)

        # Display the images
        print("\033[31mDone! Press any key to remove all the images.
\033[0m")
        display(image, simulated_image, daltonized_image, new_image)

    else:
        # Ask the user if they want a simulated or daltonized image
        choice2 = input("Do you want a simulated or daltonized video?
").lower()
        if choice2 != 'simulated' and choice2 != 'daltonized':
            print("Please enter either 'simulated' or 'daltonized' as
input here. \n")
            restart()

        try:
            # Try to open the default camera (camera index is 0)
            video = cv2.VideoCapture(0)
        except:
```

```
            print("Can't open webcam, please make sure you have a webcam
installed.")
            restart()

        # Loop through each frame in the video
        while(True):
            # Similar to cv2.read, capture each frame-by-frame
            ret, frame = video.read()

            # Convert the frame into either the simulated or daltonized
image
            if choice2 == "simulated":
                frame = simulate(frame, blindtype)
            else:
                frame = daltonize(frame, blindtype)

            # Display the resulting frame
            cv2.imshow('Recording (Press "q" to quit)', frame)

            # Break the loops if 'q' is pressed
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        # Release and end the video when recording is done
        video.release()
        cv2.destroyAllWindows()

    # Call restart function to ask user if they want to run the program
again
    restart()

def load(image_name):
    # Try to load the image to a numpy array with 255 bit values
    try:
        image = cv2.imread(image_name)
        return image.astype(np.uint8)
    # If the image doesn't exist, output an error statement
    except FileNotFoundError:
```

```
        print(f"Unfortunately, we run into an error opening
'{image_name}'.\nPlease make sure it exists or if you named it
correctly.\n")
        restart()
    # Handle any other unexpected exceptions.
    except:
        print(f"An unexpected error has occured while loading
'{image_name}'.\n")
        restart()

def display(image, simulated, daltonized, both):

    try:
        # Resize the images to be a quarter of the size because cv2 has a
display size limit
        # Display original image
        cv2.imshow('Original Image', cv2.resize(image, (250, 250), None))

        # Display simulated image
        cv2.imshow('Simulated Image', cv2.resize(simulated, (250, 250),
None))

        # Display daltonized image
        cv2.imshow('Daltonized Image', cv2.resize(daltonized, (250, 250),
None))

        # Display simulated daltonized image
        cv2.imshow('Simulated Daltonized Image', cv2.resize(both, (250,
250), None))

        # Tell user that the image are displaying
        print("Image is displaying, press any key to quit.")

        # Wait for a key event and close all windows when a key is pressed
        cv2.waitKey(0)
        cv2.destroyAllWindows()

    except:
        print("Something went wrong...")
        restart()
```

```
# In the case that the user inputs something wrong, give the user a chance
to restart the program
def restart():

    # Ask if user want to rerun the program
    user_input = str(input("Do you want to rerun the program (y or n)? "))

    if user_input == "y": # The user wants to restart the program
        print("\033[31mRestarting program...\n\033[0m")
        main()
    elif user_input == "n": # The user does not want to restart the
program
        print("\033[31mEnding program, thank you for using Jamie's
Colorblind filter\033[0m")
        exit()
    else:
        # If the user enters neither y or n, just simply exit the program.
        restart()

if __name__ == "__main__":
    main()
```

5.1.2 Daltonize.py

```
import numpy as np
import cv2

# Process described here:
https://ixora.io/projects/colorblindness/color-blindness-simulation-resear
ch/

def daltonize(image, blindtype):

    # Pick which transformation kernel to use
    if blindtype == "deuteranopia": # Kernel for deuteranopia
```

```
        kernel = np.array([[0.9, 0.4, 0.0],
                           [0.0, 0.8, 0.5],
                           [-0.3, 0.2, 1]])

    elif blindtype == "protanopia": # Kernel for protanopia
        kernel = np.array([[0.95, 0.35, 0.0],
                           [0, 0.9, 0.4],
                           [-0.2, 0.1, 1]])

    else: # Kernel for tritanopia
        kernel = np.array([[1, 0.183, 0],
                          [0, 0.927, 0.073],
                          [0, 0.5, 1]])

    # Remove gamma correction to processing colors ranging to [0, 1]
    gamma_removed = image / 255.0

    # Convert image colors using kernel colorspace
    transformed = np.dot(gamma_removed, kernel.T)

    # Set the transformed image range to [0, 1]
    transformed = np.clip(transformed, 0, 1)

    # Implement gamma correction into the image
    gamma_corrected = transformed * 255

    # For handling error, set color range to [0, 225] range
    daltonized_image = np.clip(gamma_corrected, 0, 255).astype(np.uint8)

    return daltonized_image
```

5.1.3 Simulate.py

```
import numpy as np
import cv2
```

```
# Process described here:
https://ixora.io/projects/colorblindness/color-blindness-simulation-resear
ch/

def simulate(image, blindtype):

    # Pick which transformation kernel to use based on user's inputted
blindtype
    if blindtype == "deuteranopia": # Kernel for deuteranopia
        kernel = np.array([[0.55, 0.45, 0],
                           [0, 0.45, 0.45],
                           [0, 0.525, 0.475]])

    elif blindtype == "protanopia": # Kernel for protanopia
        kernel = np.array([[0.95, 0.05, 0],
                           [0, 0.65, 0.35],
                           [0, 0.675, 0.325]])

    else: # Kernel for tritanopia
        kernel = np.array([[0.567, 0.433, 0],
                           [0.558, 0.442, 0],
                           [0, 0.242, 0.758]])

    # Transform the images
    simulated_image = cv2.transform(image, kernel)

    # Clip values to be in the valid range [0, 255]
    simulated_image = np.clip(simulated_image, 0, 255)

    # Convert back to uint8
    simulated_image = simulated_image.astype(np.uint8)

    return simulated_image
```

5.2 References

6,200+ Airplane Cockpit View Stock Photos, Pictures & Royalty-Free Images - iStock. (n.d.). Retrieved December 4, 2023, from www.istockphoto.com website: https://www.istockphoto.com/photos/airplane-cockpit-view

Foundation, P. (2017, October 13). Image Processing Library to Ease Differentiation of Colors for People with Colorblindness. Retrieved November 29, 2023, from Processing Foundation website: https://medium.com/processing-foundation/image-processing-library-to-ease-differentiation-of-colors-for-people-with-colorblindness-cc550f0670e0

Jim. (2016a, August 28). Color Blindness Simulation Research. Retrieved from ixora.io website: https://ixora.io/projects/colorblindness/color-blindness-simulation-research/

Jim. (2016b, August 28). ColorBlindess: Processing Library. Retrieved November 29, 2023, from ixora.io website: https://ixora.io/projects/colorblindness/

Jim. (2016c, September 13). Daltonization. Retrieved November 29, 2023, from ixora.io website: https://ixora.io/projects/colorblindness/daltonization/

Lee, J., & dos Santos, W. P. (2011). An adaptive fuzzy-based system to simulate, quantify and compensate color blindness. *Integrated Computer-Aided Engineering*, *18*(1), 29–40. https://doi.org/10.3233/ica-2011-0356

National Eye Institute. (2023, August 7). Types of Color Vision Deficiency | National Eye Institute. Retrieved from www.nei.nih.gov website: https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness/types-color-vision-deficiency

Wikipedia Contributors. (2019, March 11). Von Kries coefficient law. Retrieved January 5, 2020, from Wikipedia website: https://en.wikipedia.org/wiki/Von_Kries_Coefficient_Law