Activity 1: Digital Image Formation and Enhancement

Applied Physics 157 AY22-23 2nd Sem
Mary Franczine Tan
March 13, 2023

Objectives

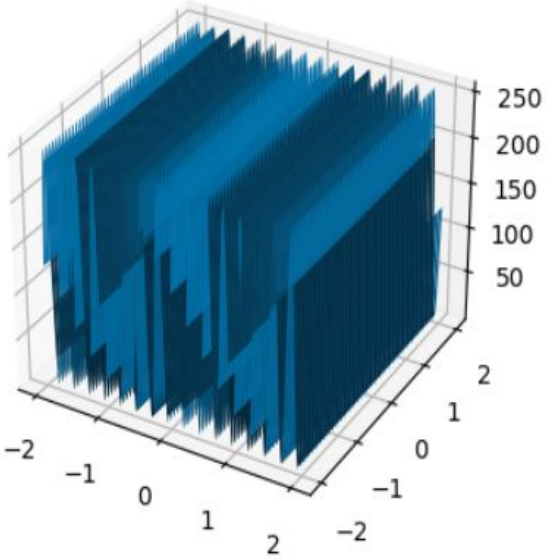
1. Learn how to create images on a programming software; in my case, Python
2. Use backprojection to alter the distribution of a grayscale image
3. Apply contrast stretching to a grayscale image
4. Apply the three white balancing algorithms (contrast stretching, gray world algorithm, and white patch algorithm) on faded colored images

Results and Analysis

1.1 Image DIY

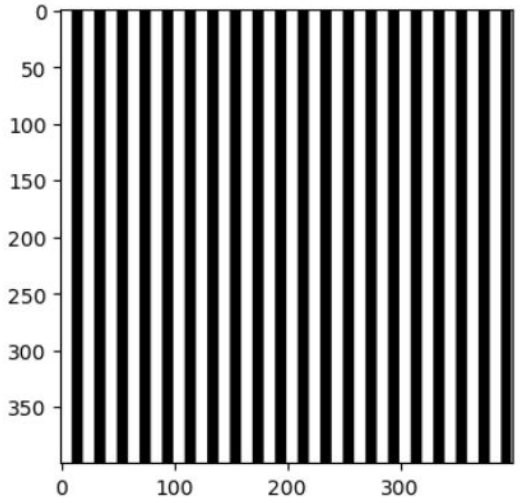
The sinusoidal wave is a regular sine wave, making the plotting of its surface plot straightforward. The grating makes use of a square wave. The grating itself represents the z-values of the wave, showing the binary values of a square wave.

An important thing to take note of would be how to alter the frequency of the wave. This is simply done by multiplying the angle (in radians) by the frequency.



Sinusoid along the x-direction, frequency is 4 cycles/cm

$z = np.sin(8\pi x)$



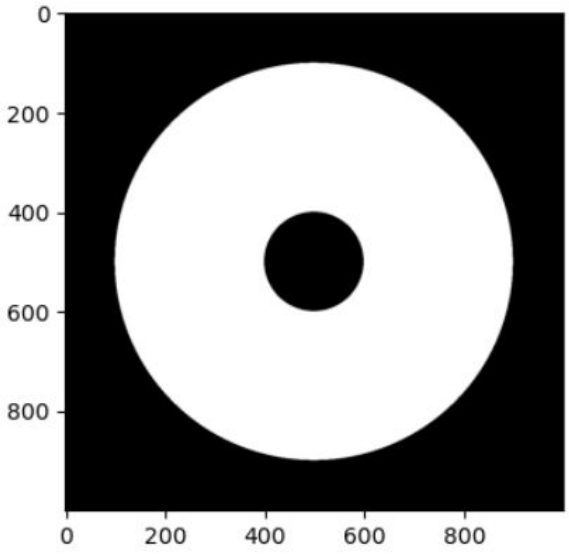
Grating, frequency is 5 line pairs/cm

$z = signal.square(10\pi x)$

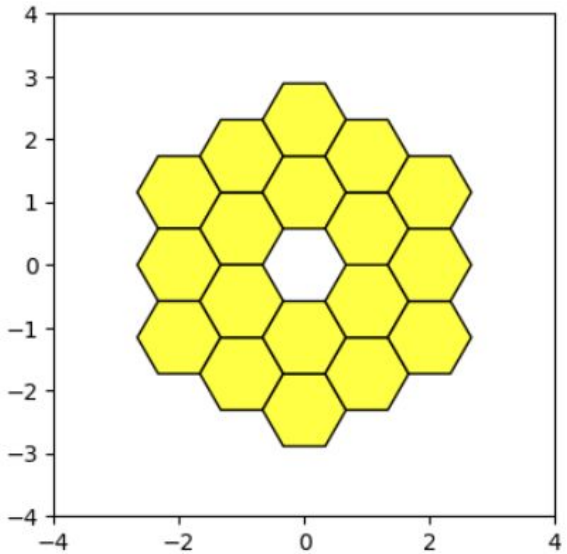
1.1 Image DIY

The simulation of the hubble space telescope lens was created based from the code of the circle given in the module. It was constructed by first creating a big circle, then creating a smaller circle that represents the “hole”, and subtracting these two arrays.

The JWST lens is constructed pretty much the same way the real lens was, by combining different hexagons and tiling them to form the desired shape.



Hubble Space Telescope Lens

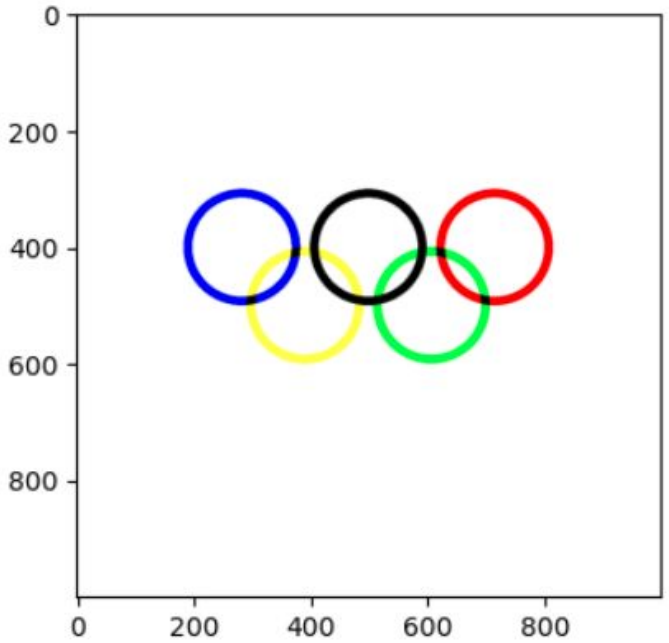


James Webb Space Telescope Lens

1.2 Color Image

The olympics logo was created by making different loops and assigning the appropriate color and location for each one.

An interesting point here is that since the logo's background has to be white, the way to give color for each loop wasn't straightforward. It worked more like cancellation, where the loops I created actually represented the pixels that are turned off in the image array.



Recreation of Olympics Logo

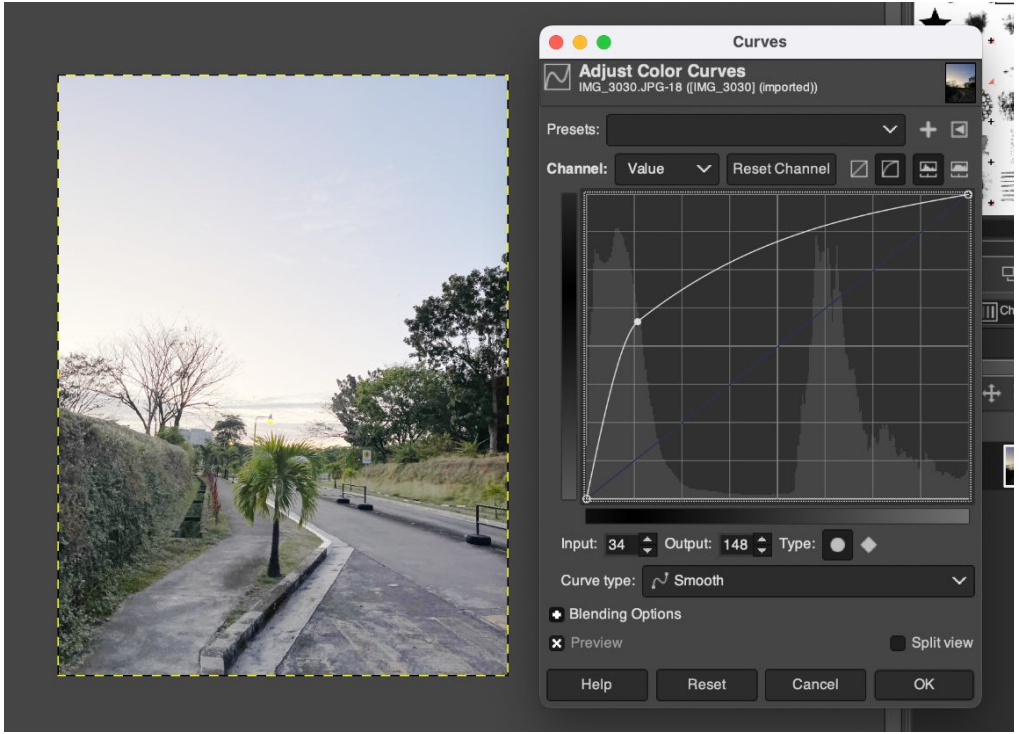
1.3 Altering the Input-Output Curve

The curve of the image represents the brightness of each pixel in the image, and altering it changes the brightness of each pixel. What I've noticed while playing around with the curve is that as it is shifted to the left, the image gets brighter. If it is shifted to the right, its gets darker overall.

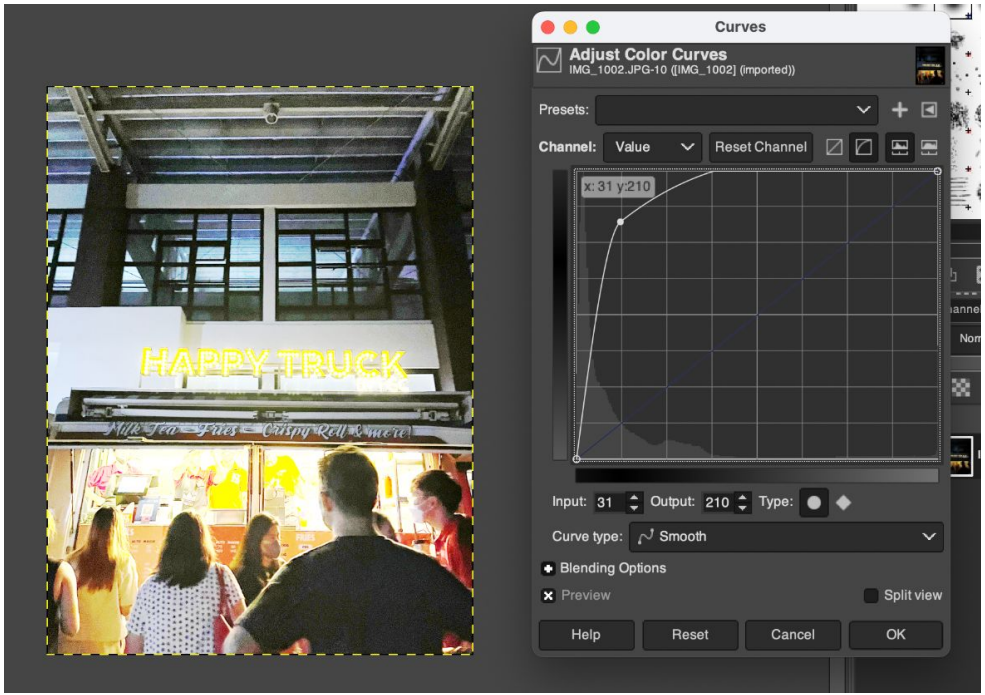
I've also tested the method in different light settings. I've noticed that the less darker or less light source there is, the grainier the image is when I make the image brighter by altering the curve.

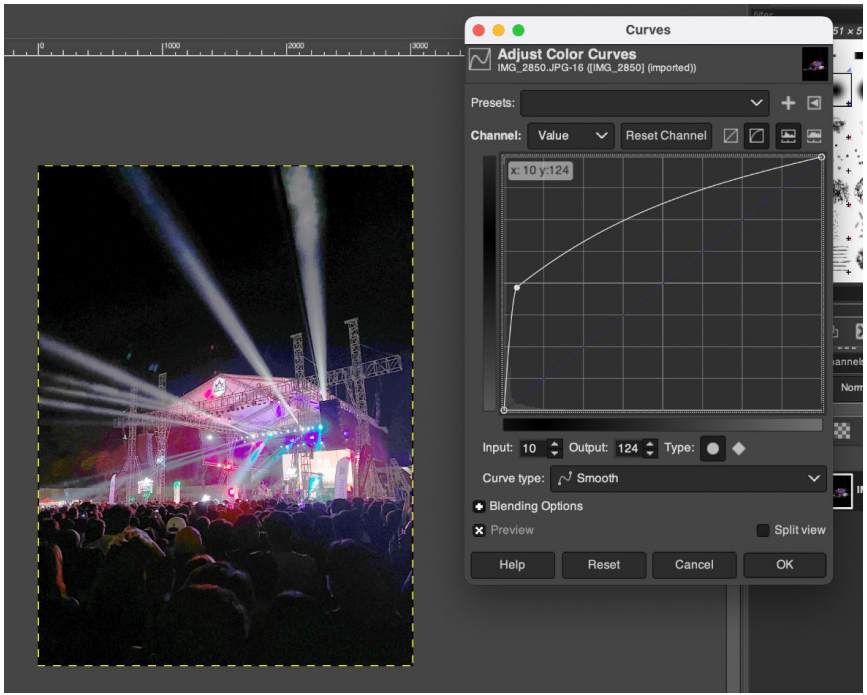
The following slides show the colors curves and their results.

All photos used for this part are mine.

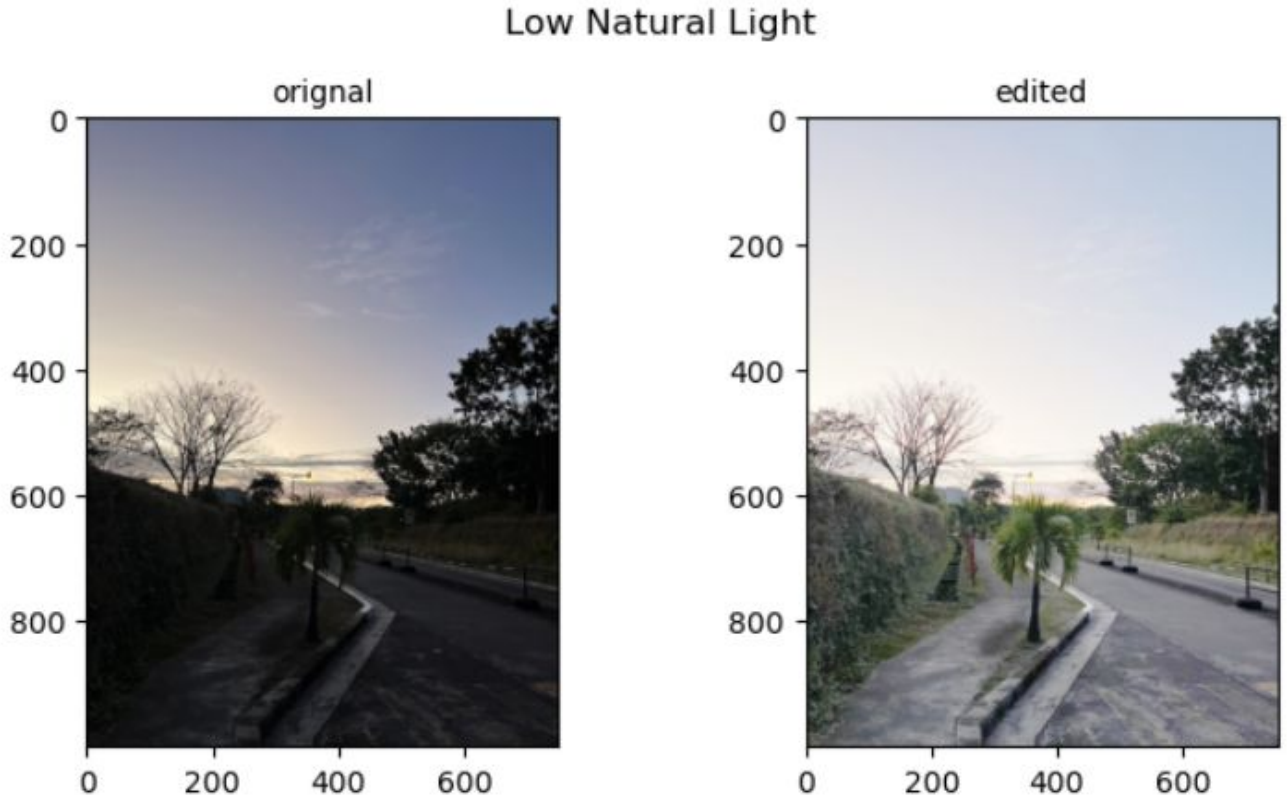


1.3 Altering the Input-Output Curve

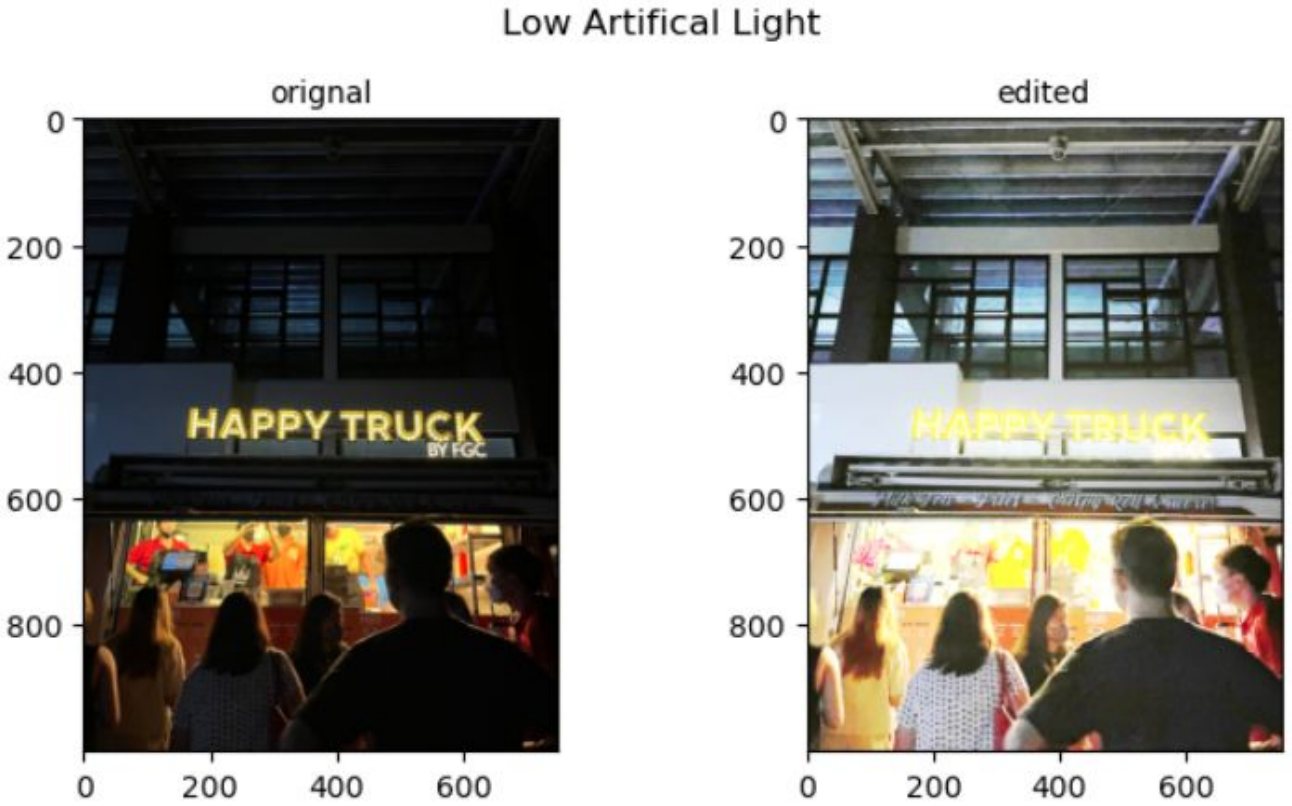




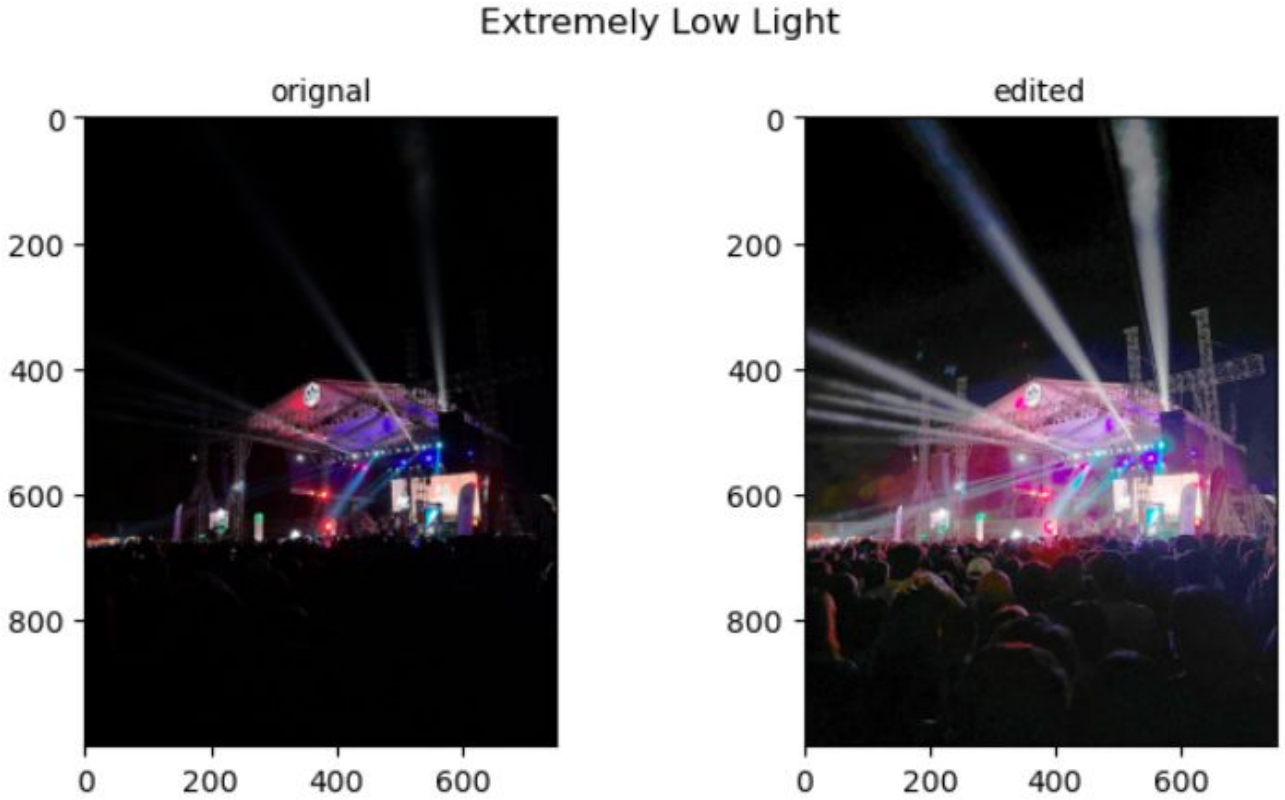
1.3 Altering the Input-Output Curve



1.3 Altering the Input-Output Curve



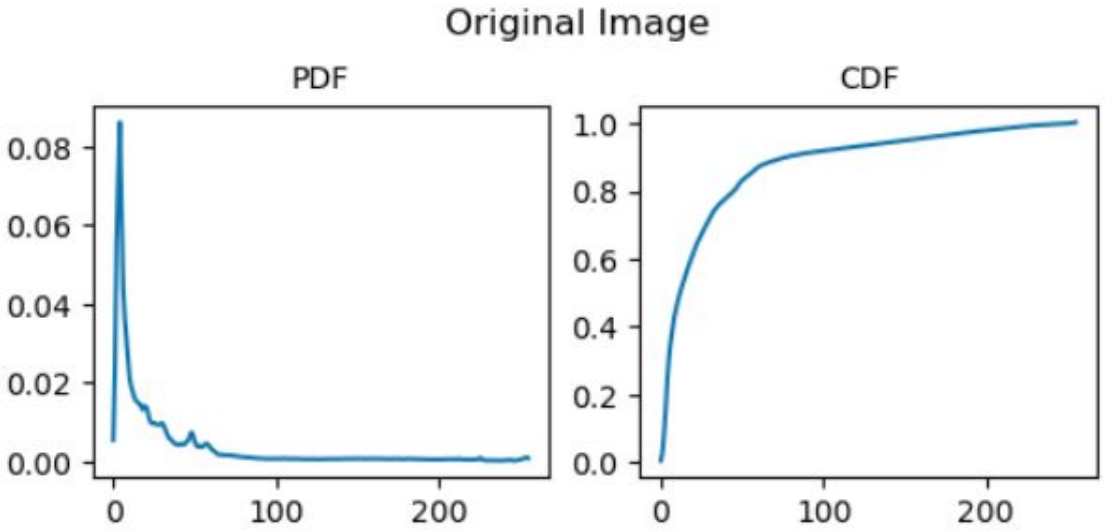
1.3 Altering the Input-Output Curve



1.4 Histogram Backprojection on Grayscale Images

Histogram backprojection is the process of altering an image's Cumulative Distributive Function (CDF) to a desired shape, thereby altering the brightness of each pixel in the image.

We can see that the original image has mostly dark pixels.

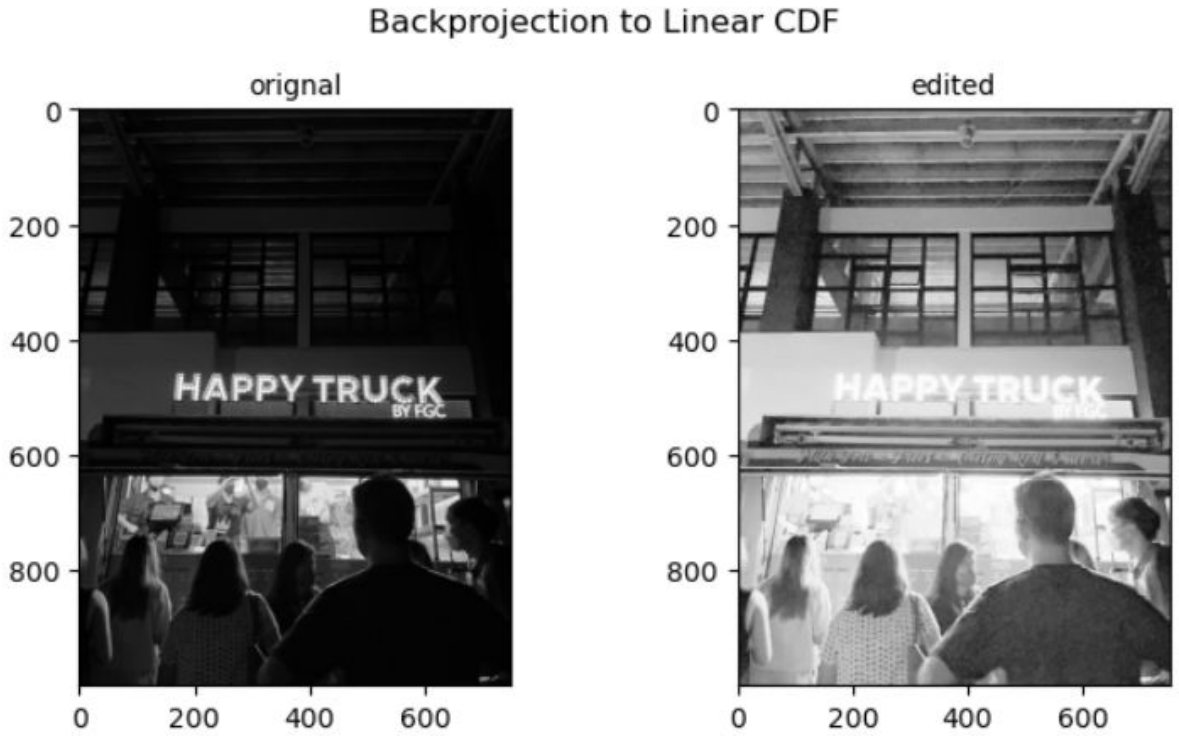


1.4 Histogram Backprojection on Grayscale Images

The resulting image of a histogram backprojection to a linear CDF has a brighter image overall.

This matches the theoretical results, given that the desired CDF is linear which means that all range of pixel values should be represented.

One effect of the backprojection, though, is that the words in the photograph are harder to read because they lack contrast with the rest of the image.



1.4 Histogram Backprojection on Grayscale Images

The resulting image of a backprojection to a sigmoid function CDF has a brighter image overall, but still darker compared to the linear CDF.

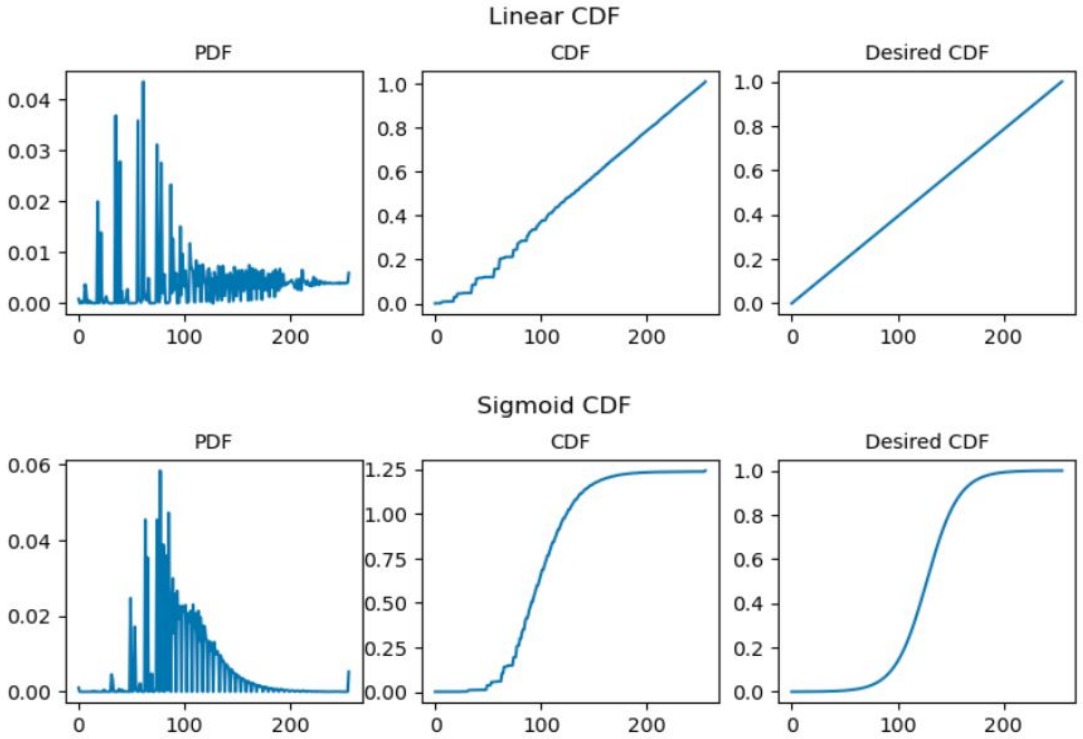
This is expected because the sigmoid function places most of the pixels in the middle range. This gives us an image that is bright enough to reveal details, but still evokes the lack of light in the original image. Notice also that the image has enough contrast that the text is still readable.

1.4 Histogram Backprojection on Grayscale Images

Shown are the PDFs, CDFs, and desired CDFs of the backprojections.

Notice that the CDFs of the backprojected images closely resemble the shape of the desired CDF, if not for the imperfections in its graph. This shows that the backprojection worked in imitating the shape of the desired CDF.

The PDF of the image is also more distributed, unlike the original one which has most values in the low values. This also shows that there are more brighter pixels.



1.5 Contrast Enhancement

Contrast enhancement is done by normalizing the brightness of each pixel in the image.

But, there is the issue of an image having fully occupied the entire possible range of values for the pixel brightness, which means normalization has no effect.

For my code, I've written a function that automatically does the contrast stretching, with it a built-in if-statement to decide whether it will normalize as normal or use a minimum and maximum based off its CDF>

```
def stretchContrast(image, imageCDF):
    """
    applies contrast stretching to image
    """

    roundCDF = list(np.round(imageCDF, 1)) # rounds CDF to integers

    ## chooses Imin and Imax depending on the min and max value of the image

    if (image.max() > 245) and (image.min() < 10):
        Imax = roundCDF.index(0.9)
        Imin = roundCDF.index(0.1)
    else:
        Imax = image.max()
        Imin = image.min()

    imageCS = 255 * ((image - Imin) / (Imax - Imin)) # normalization

    return imageCS
```

If min and max is close to 0 and 255, it sets the Imax and Imin to be at the grayscale value at CDF equals 0.1 and 0.9 respectively

1.5 Contrast Enhancement

For the original image, the min and max values were actually 0 and 255.

This is to be expected, since even from a quick view of the image, we can see that the subject (the text “HAPPY TRUCK”) is in high contrast to the rest of the image.

When contrast stretching was applied, it lightened the image but at the cost of the “HAPPY TRUCK” text being unreadable.

1.5 Contrast Enhancement

The contrast enhancement was also applied to the backprojected images. This one is for the Linear Desired CDF.

The thing that immediately stands out to me is that the text is also unreadable here, and arguably more indistinguishable from the surroundings compared to the contrast stretched original image.

We can also observe that the blacks on the contrast enhanced image are deeper (as shown by the black shirt of the guy).

1.5 Contrast Enhancement

This is the contrast enhancement for the Sigmoid Function Desired CDF.

I think this one has the most desirable results of the contrast enhancement. The text is still readable, but the higher contrast is still obvious in the black shirt and the shadows.

I think perhaps the other images will have more desirable results if the contrast wasn't stretched to the absolute maximum. Though the method of applying that code to python is not something that I was able to do.

1.6 White Balancing of Color Images

White balancing is the process of correcting an image such that the whites in the image actually appear like whites. There will be three methods tested for this part: Contrast Stretching, Gray World Algorithm, and White Patch Algorithm.

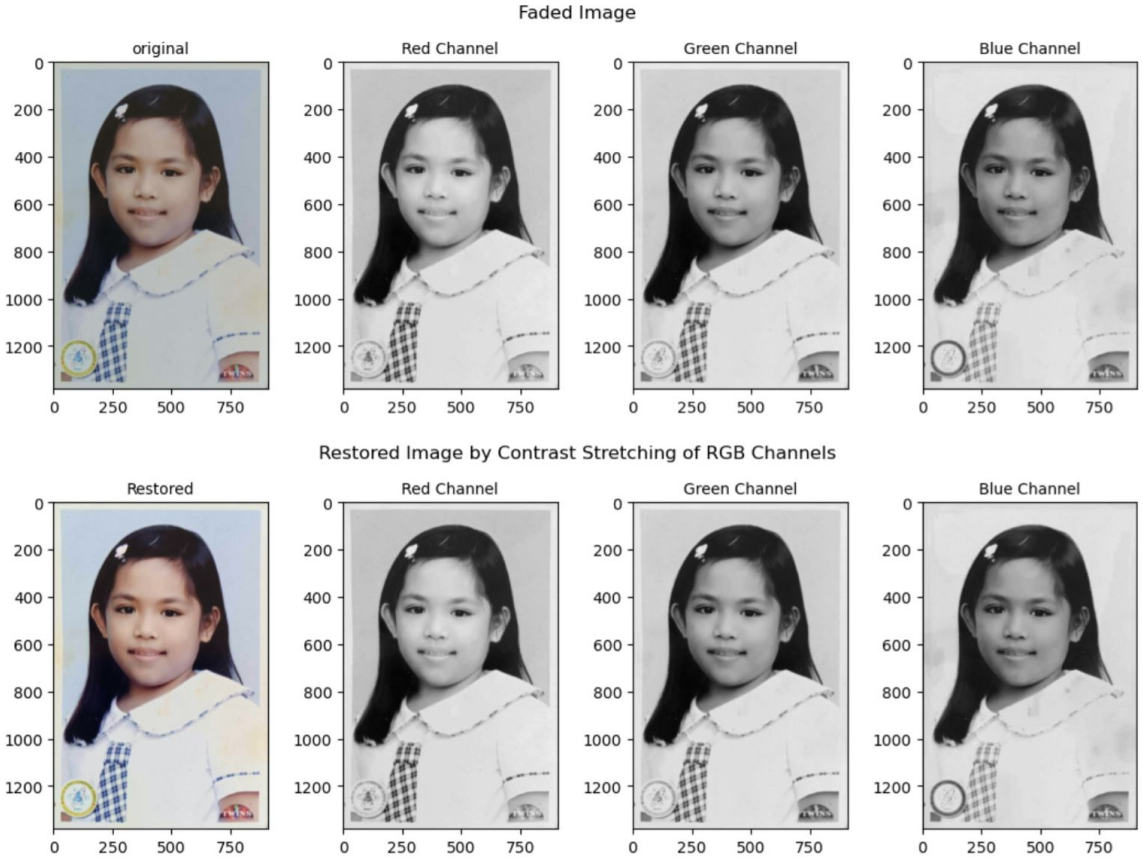
The different methods have worked as expected, and their results will be shown and discussed in the following slides. I also created functions for each white balancing method so that I can easily apply them on their images.

Here, I've used old family images that have their colors faded already. The first one (shown to the right) is a picture of me taken and developed around 2011.

1.6 White Balancing of Color Images

We can see that the original image has its colors already faded. For reference, the border around the image was originally white.

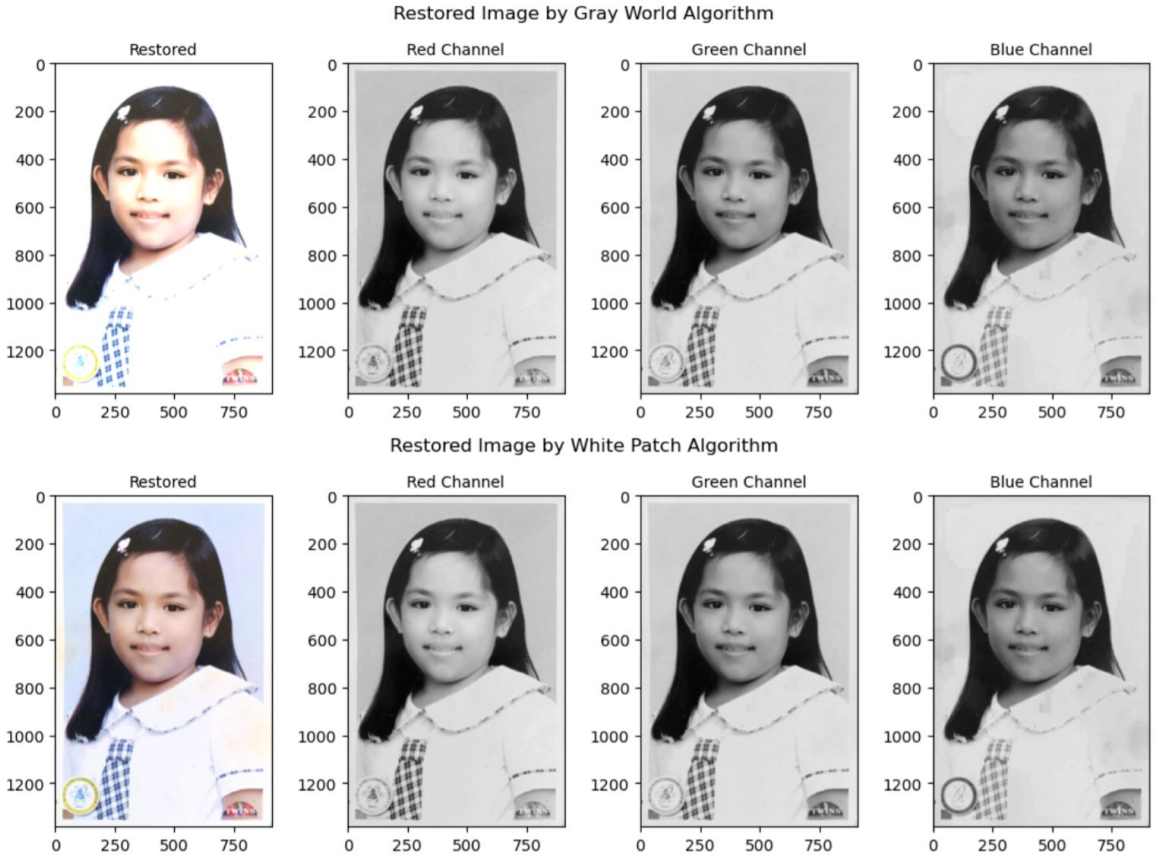
The results of the white balancing by contrast stretching is a photo that has more vibrant and saturated colors compared to the original. This shows that the color restoration by this method works.



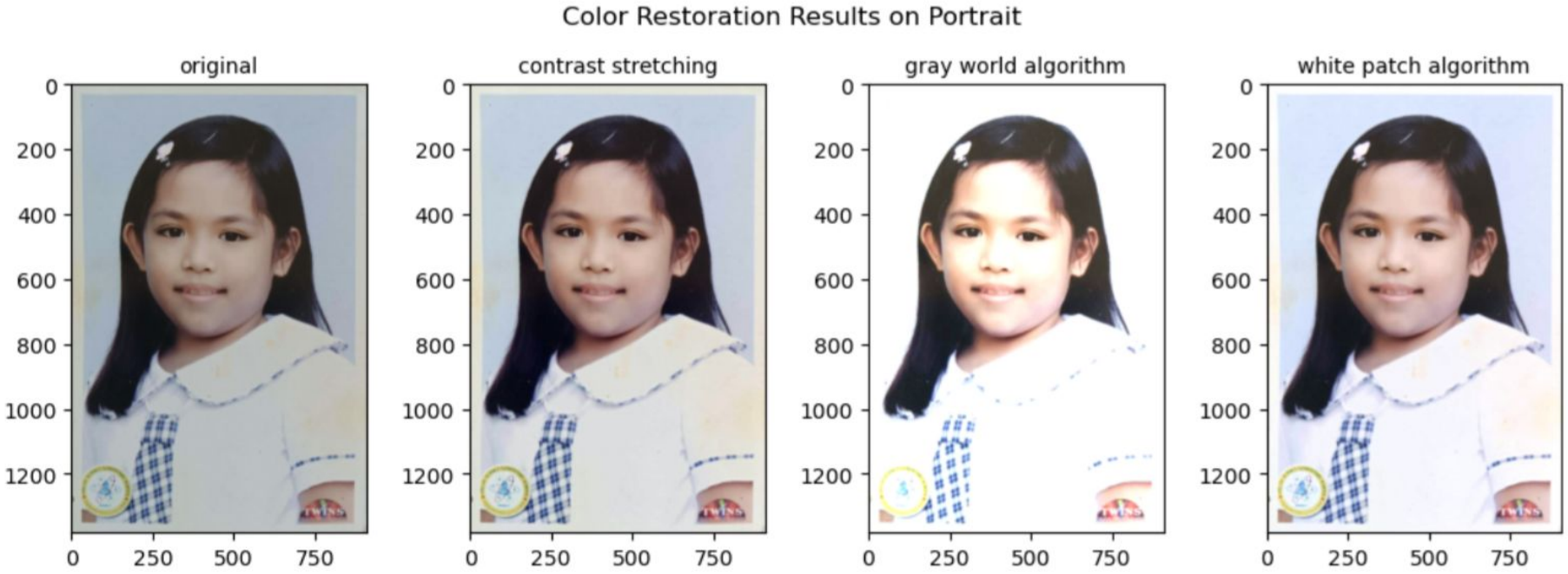
1.6 White Balancing of Color Images

The gray world algorithm was also successful in returning colors to the image, but the subject and the background is completely blown out. The border and the backdrop color became indistinguishable because of it.

The white patch algorithm is also successful in returning the colors while also retaining the features of the subject and the entire image.



1.6 White Balancing of Color Images



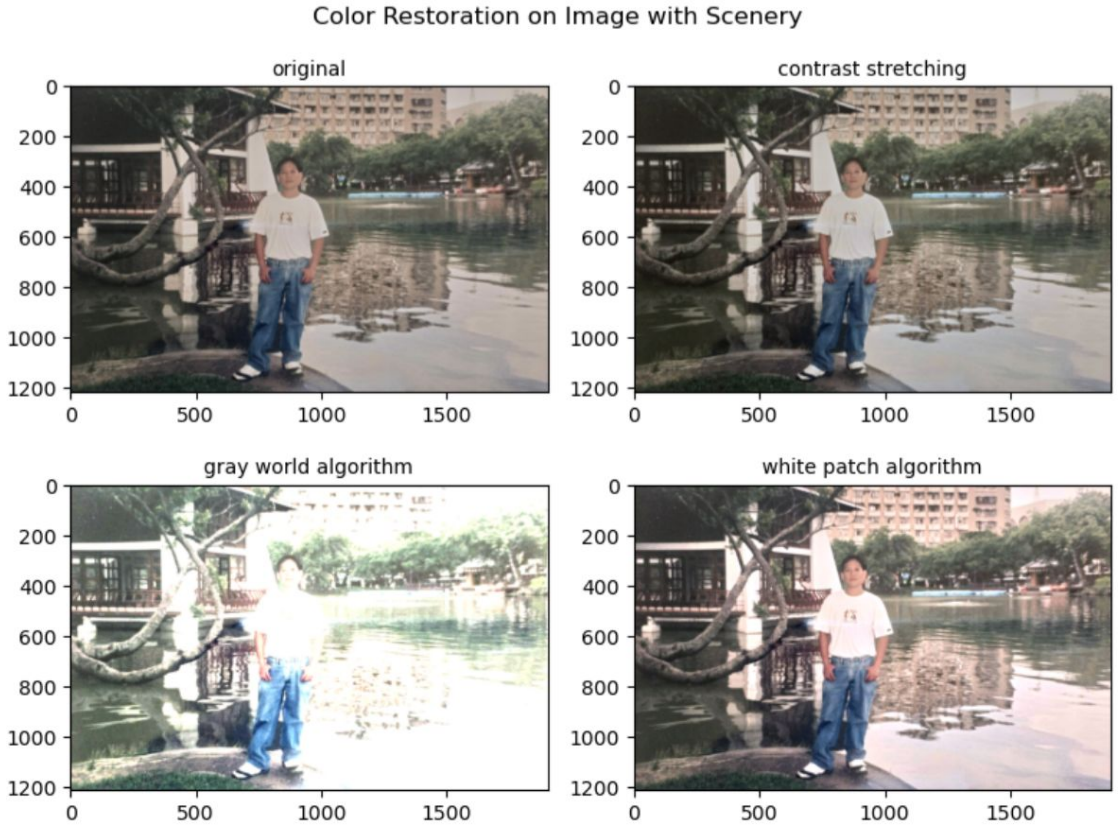
The white patch algorithm has done the best job of color restoration in this image. The whites in the image are very clearly white, unlike in contrast stretching where they still look a bit gray. It also has the most vibrant coliors.

1.6 White Balancing of Color Images

I wanted to try the algorithms in more photos, and chose this an image that contained a scenery. This makes the image more complicated because it has more features than a portrait.

This photo is of my father from the 1990s.

The results are pretty consistent with the first image. The gray world algorithm is still too blown out, and the white patch algorithm does the best white balancing and color restoration.

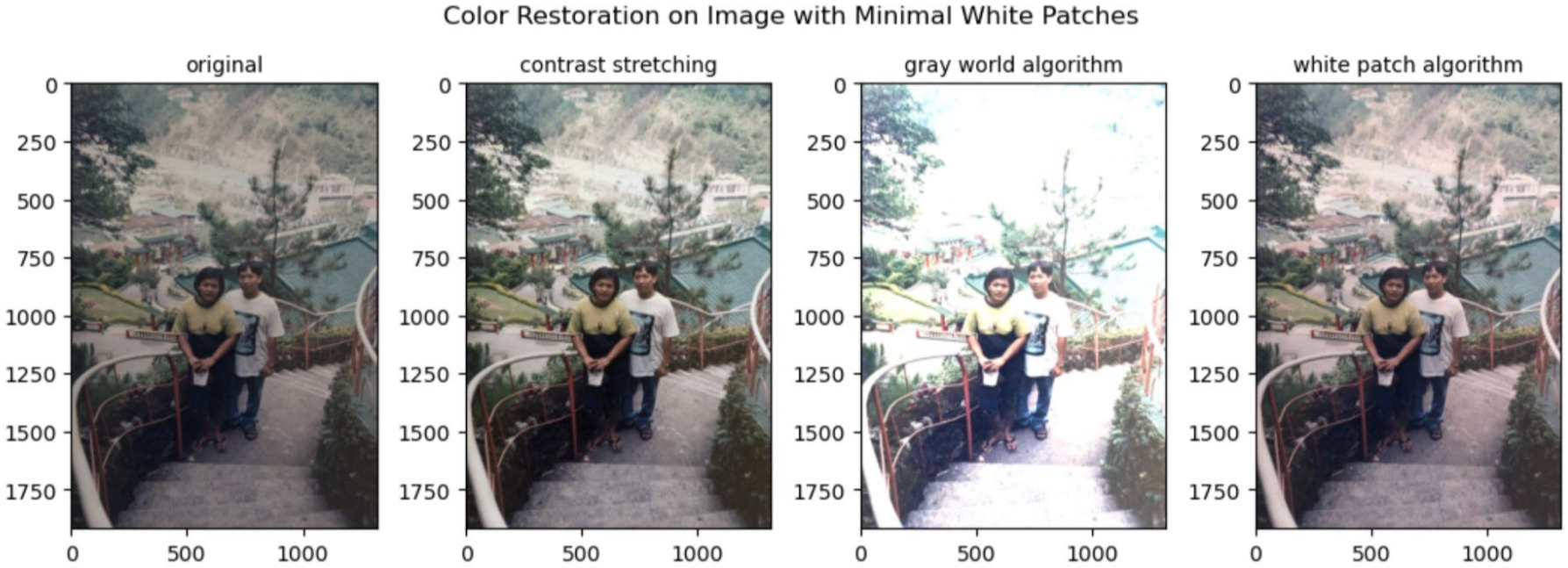


1.6 White Balancing of Color Images

Since the white patch algorithm was the most successful so far, I wanted to see how it would fair in an image that had very minimal white patches.

This is an image of my parents from the 1990s. I chose the white shirt of my father as the white patch.

1.6 White Balancing of Color Images



Here, we can see that the white patch algorithm performed poorly and its result doesn't differ much to the original. The gray world algorithm is still too blown out, but did the best in restoring the colors. In terms of overall performance, the contrast stretching did the best because it restored the vibrancy of the colors, albeit very minimal, and was able to keep the details of the background.

Reflection

1.6 White Balancing of Color Images

I actually enjoyed the activity a lot, although I did get super frustrated at the histogram backprojection. I encountered a lot of problems there, from my code not working to struggling the entire time with applying the sigmoid function (turns out I needed to shift it). There was also a part where the 3d plots suddenly weren't showing up. Thanks to Sir Kenneth, I was able to resolve them. Shout out also to my seatmate Zach Hizon for helping me with the histogram backprojection.

The part that I enjoyed the most would probably be the white balancing and color restoration. I was kind of sad that the white patch algorithm didn't work on the last image, but it was expected given that the image had very little white patches. I wanted to do it on more photos, and a part of me is very excited to go back home since we have an entire drawer of old photos that I can use this on. I also really enjoyed making functions for the process that I repeated a lot. It was my first time creating functions that had a lot of customizability as well as ones that automatically set values/methods depending on the parameters (like the function I made for the contrast stretching).

The results of my code meets the expectations for the activities, and I believe the presentation also does a good job of explaining my results and my thought process during the activity. That's why I chose to give myself a perfect score. I also gave myself 5 extra points for the initiative, since I think my creation of the functions for the histogram backprojection, contrast stretching, and the three white balancing algorithms were significant improvements to the code since it made applying them to different images extremely easy and repeatable.

1.1 Image DIY

The plotting of hexagons was one that I found very hard to wrap my head around.