11A 12

Activity 1: Digital Image Formation and Enhancement

Julian Christopher L. Maypa
2020-07587
Applied Physics 157 WFY-FX-1

11A 12

Objectives

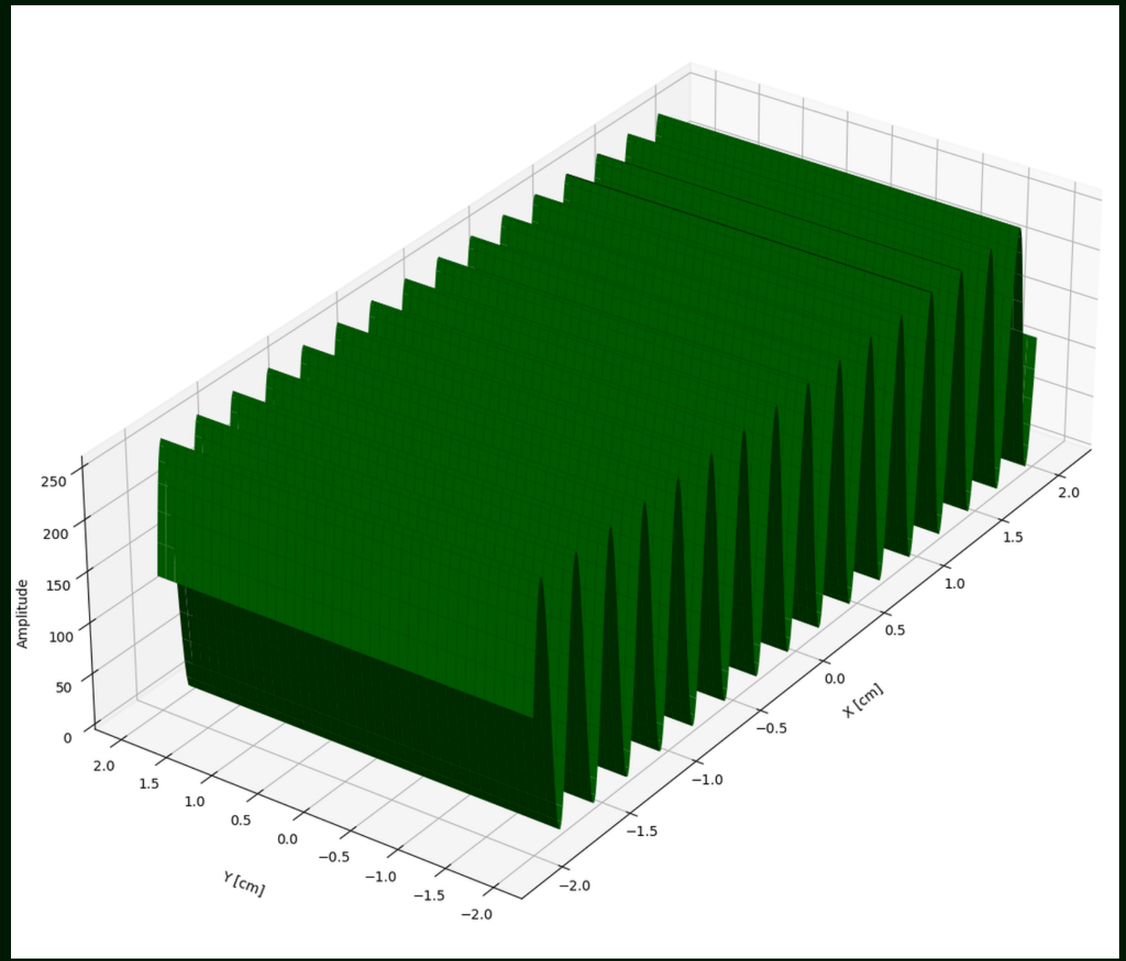
- Mathematically create images.
- Save an image in an appropriate file format.
- Open and capture images using Python or Matlab
- Improve the appearance of graylevel and color images
- Use the backprojection technique to transform the histogram of an image to a desired distribution

These objectives wer directly lifted from the Applied Physics 157 laboratory manual

1.1 Image DIY

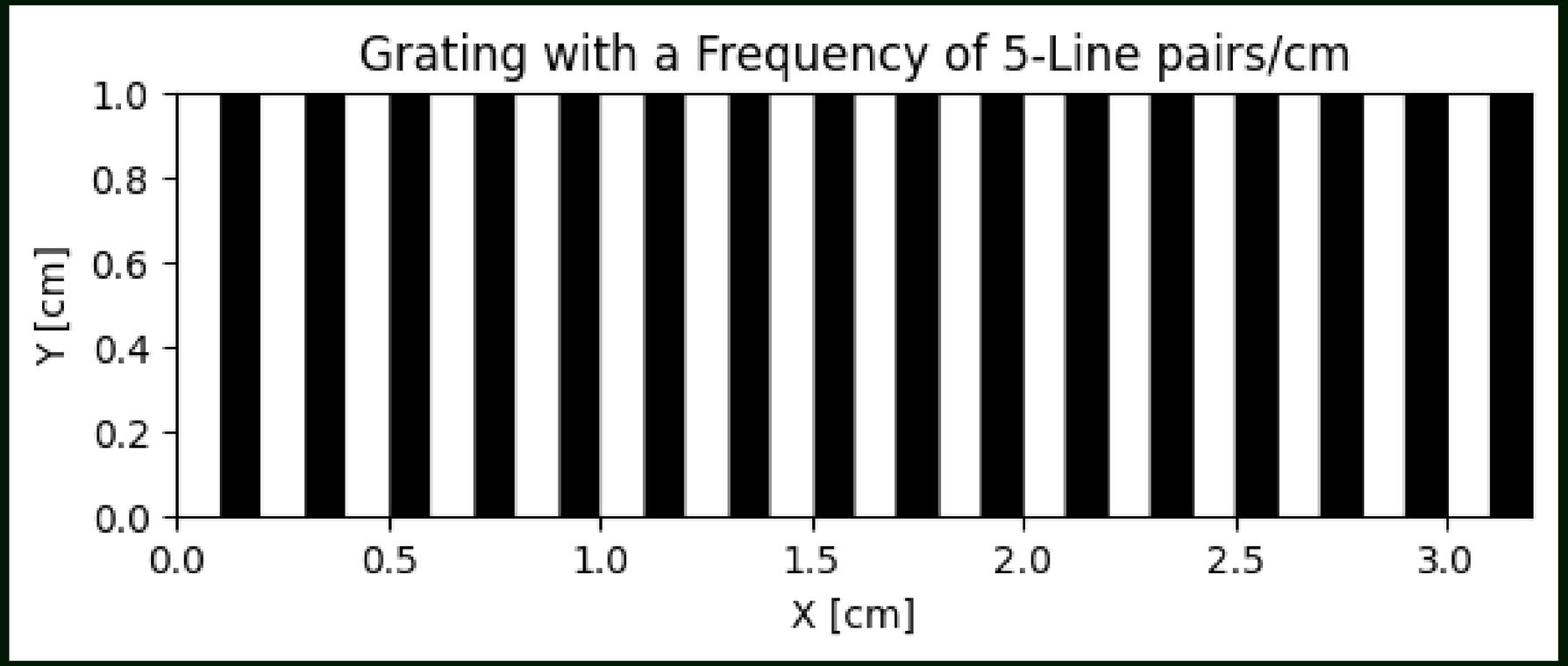
In this part of the activity, I created a 3D sinusoid along the x-direction with a frequency of 4 cycles/cm, a grating with 5 line-pairs/cm, the Hubble's Primary Mirror, and a hexagon array simulating the James Webb telescope.

Starting with the sinusoid, I made a sine wave that propagates along the x-axis with x values ranging from -2 cm to 2 cm with 400 data points. I used the equation $\sin(\omega x)$ where ω is the frequency of the sine wave. The value I assigned to ω is 2π*4 to achieve the 4 cycles/cm. After getting the Z values of the sine wave, I then proceeded to normalized the values from 0 to 255 by using the formula (Z - min(Z))/(max(Z) - min(Z)). My output can be seen on the right.



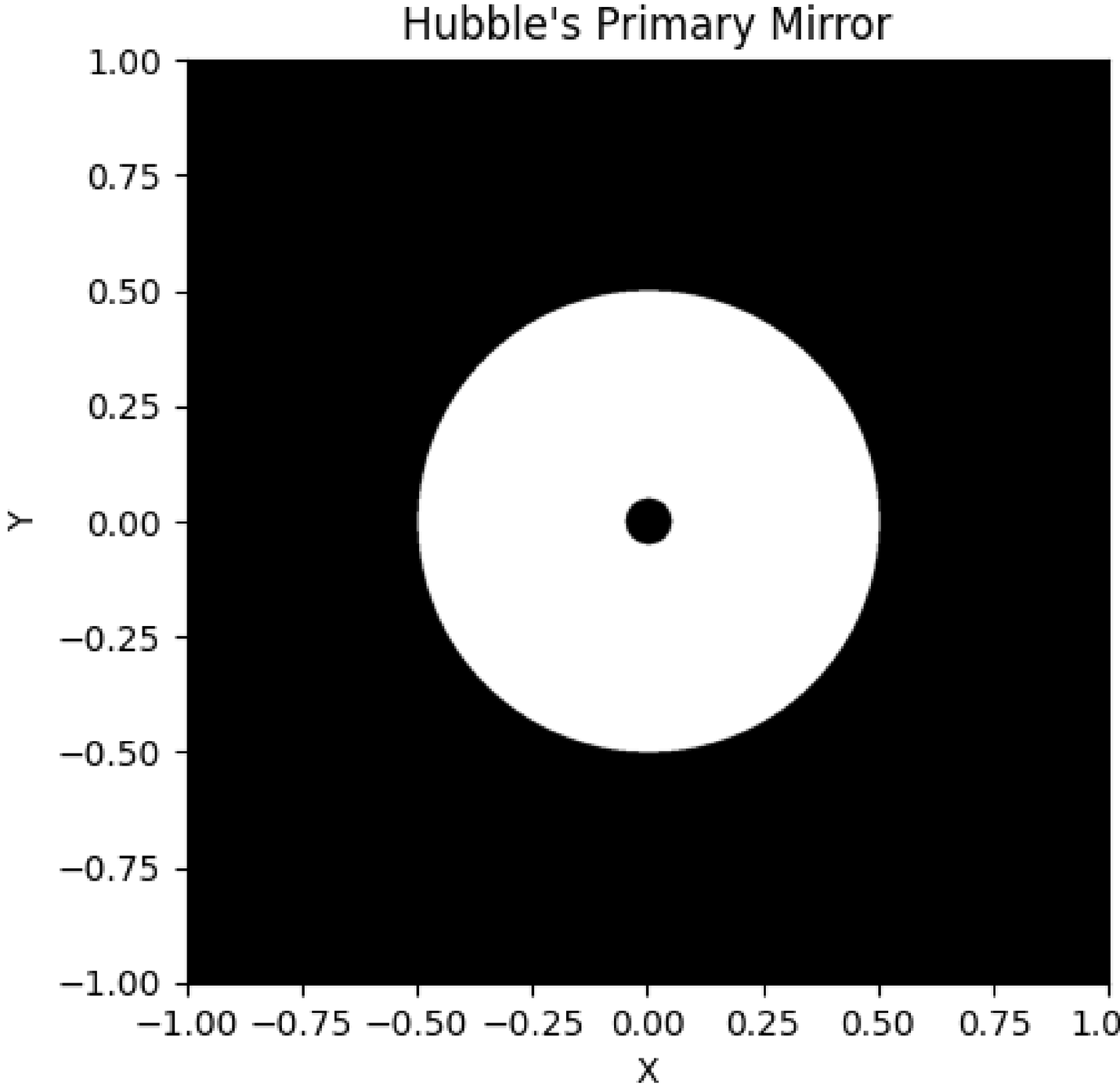
1.1 Image DIY

To make the grating, I used the same code that I used for creating the sinusoid wave but I changed the frequency to 5 cycles/cm. This is because each line pair represents one wave cycle of the sinusoid. I then made a zero matrix named A with the same shape as R. For each element in R that is greater than 127.5 (half of the sinusoid's amplitude), I set the corresponding element in A to be equal to 1 (i.e. if the element in (12,55) in R is greater than 127.5, I set the element in (12,55) in A be equal to 1). My output can be seen below.



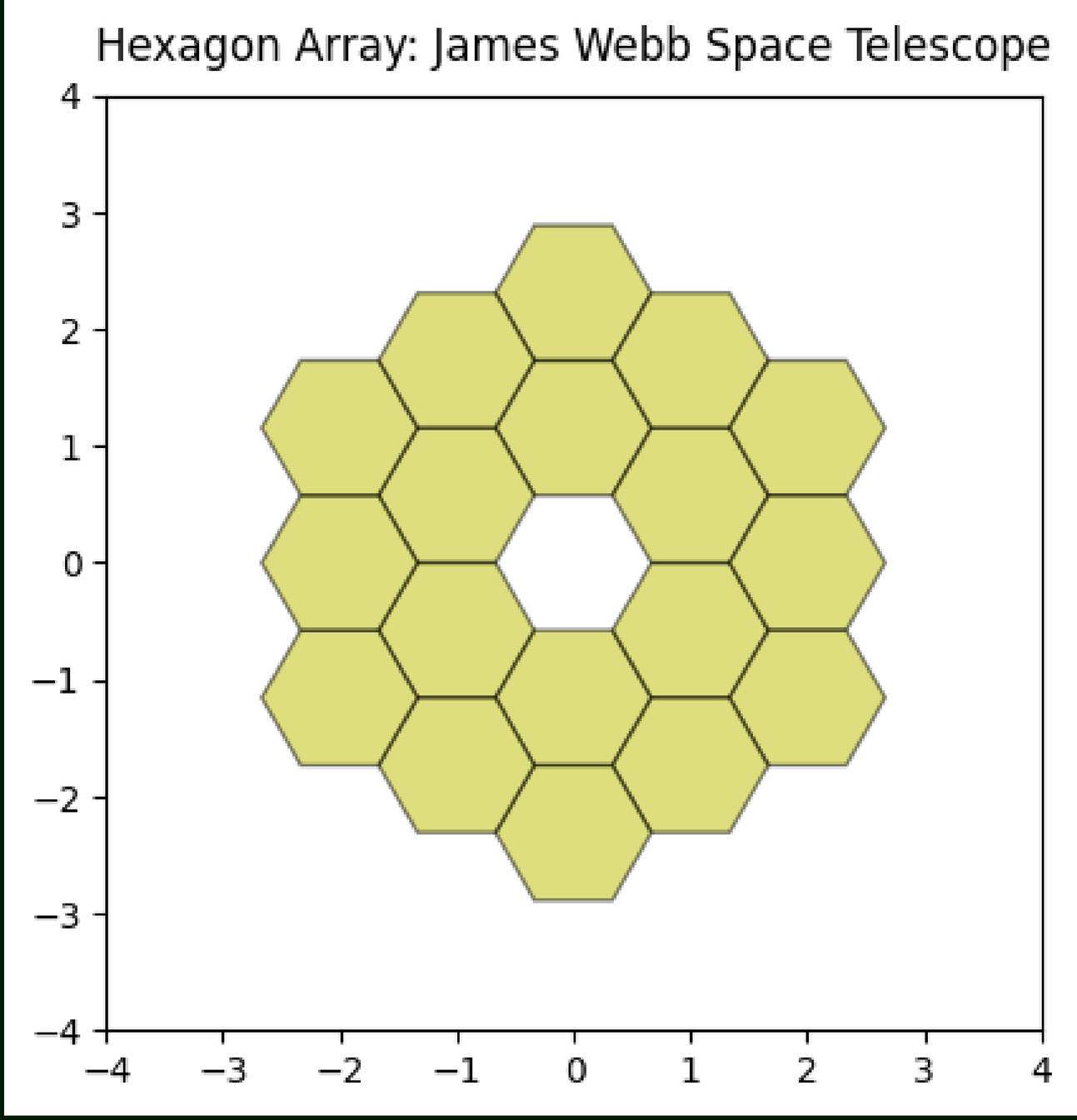
1.1 Image DIY

I made the Hubble's Primary Mirror by first making a grid with x and y values ranging from -1 to 1 with 1024 data points. I then made a zero matrix with a shape of 1024 × 1024. With the grid and matrix set up, I first made a circle with a radius of 0.5 and centered it at the origin. All the values inside the circle are set to 1. I then made a concentric circle with a radius of 0.05 and made all the values inside this smaller circle equal to 0. My output can be seen on the right.



1.1 Image DIY

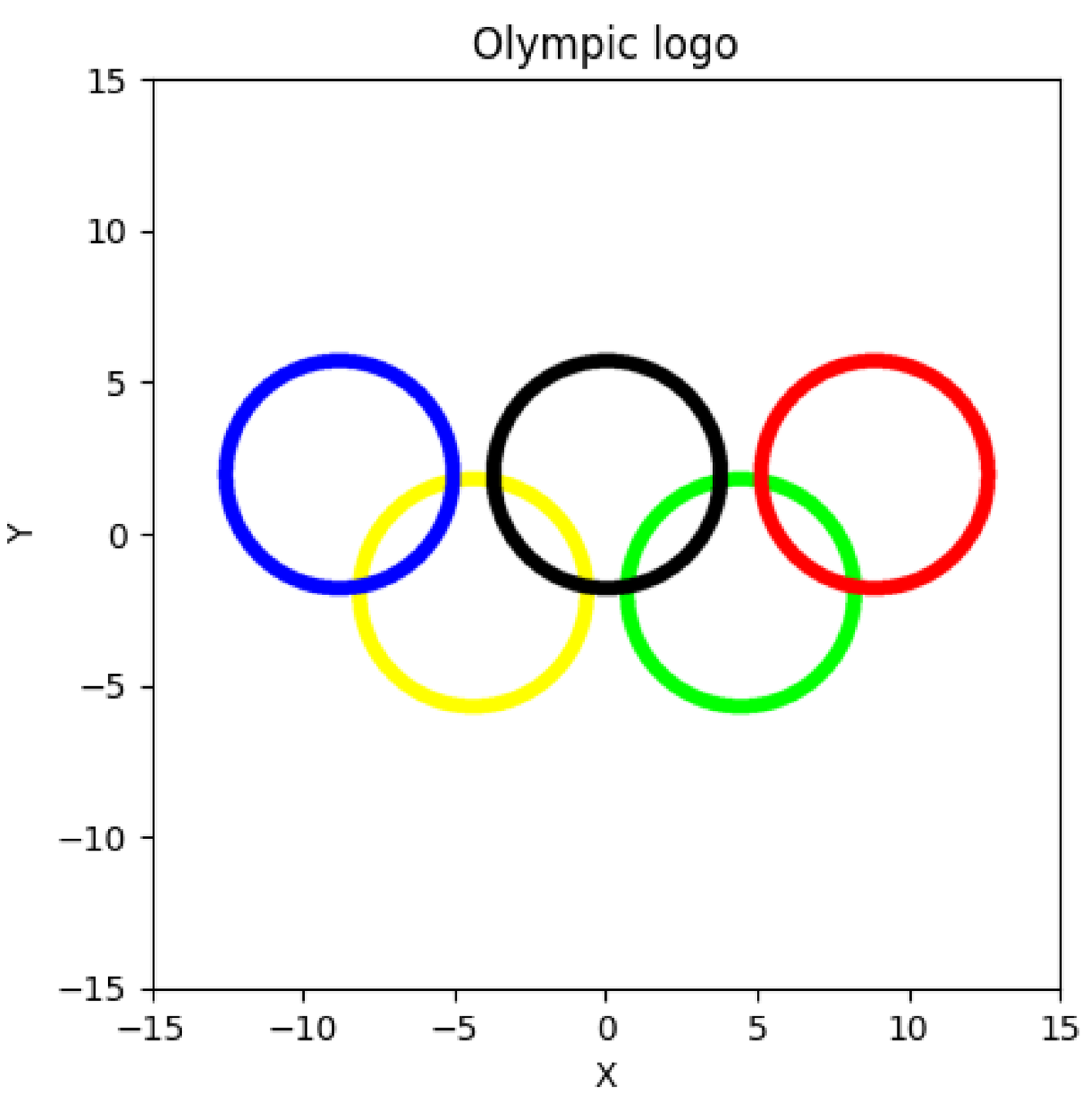
The hexagon array code that I used was taken from an article published on GeeksforGeeks by RajuKumar19 [2]. Their code used the matplotlib.patches RegularPolygon function to create the polygons. Although their code contained errors, I was able to fix it and I modified their code to make the hexagon array look like the James Webb Space Telescope. My output can be seen on the right.



link to the GeeksforGeeks article: https://www.geeksforgeeks.org/matplotlib-patches-regularpolygon-class-in-python/

1.2 Color Image

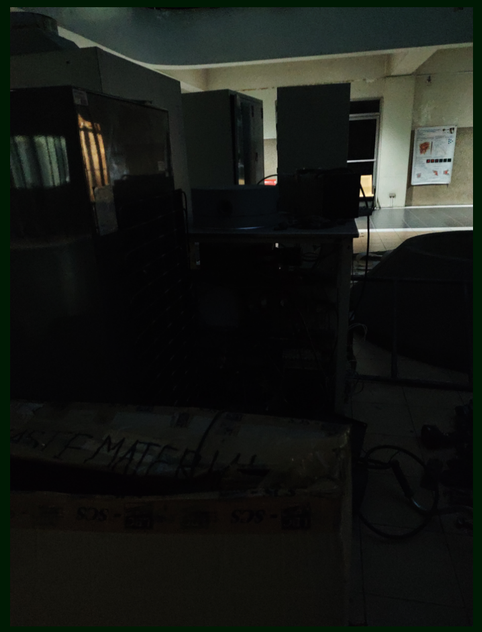
For this part of the activity, I made the Olympic logo. I did this by first making a grid of X and Y values ranging from -15 to 15. I then made a 1024 × 1024 × 3 matrix with each element having a value of 255. I first made the lower row of circles and then the upper row of circles. I did it in this order becasue I did not want the circles to overlap in color. By making the lower row of circles first, the upper row of circles would appear to be on top of the lower row. For the positioning of the circles, I used the sample code given in the Applied Physics 157 laboratory manual. My output can be seen on the right.

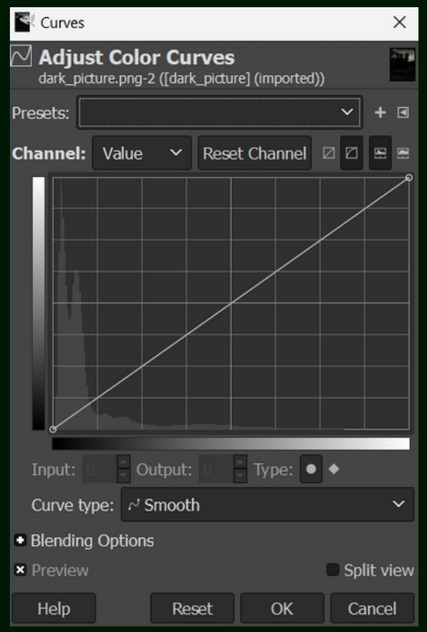


1.3 Altering the Input-Output Curve

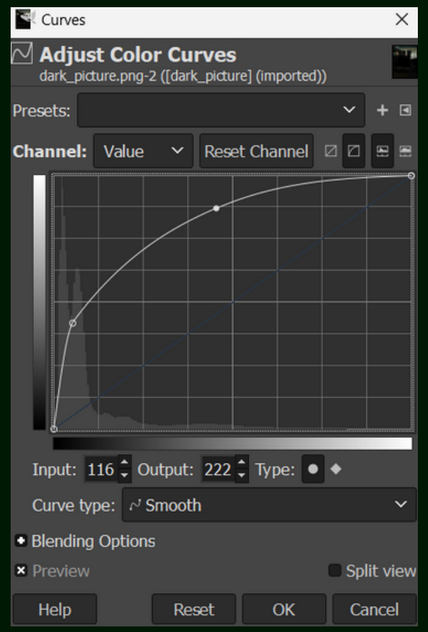
For this part of the activity, I took a picture of some machine in the research wing in NIP at night. I then used GIMP to adjust the brightness of the picture. By increasing the values of the left side of the IO curve, the brightness of the image increased. This is because increasing or decreasing the leftt side of the IO curve brightens or darkens the dark pixels respectively. The same is true for the right side of the IO curve but it brightens or darkens the bright pixels instead [3].

Original Image





Adjusted Image



1.4 Histogram Backprojection on Grayscale Images

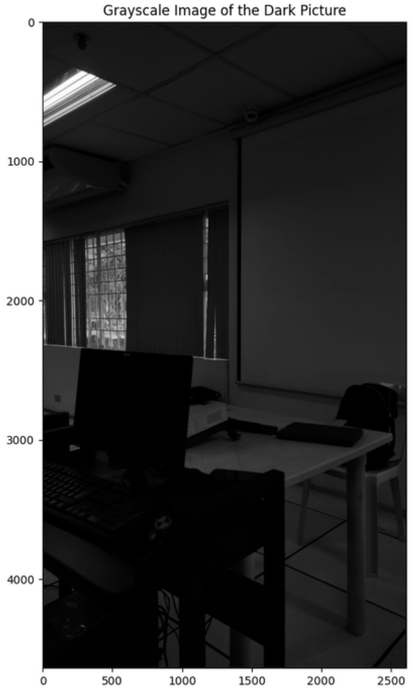
For this part of the activity, I took an artificially dark image inside Computer Lab 1 in CSRC. The image is artificially darkened by lowering the aperture and shutter speed of my camera. With this image, I did the following steps:

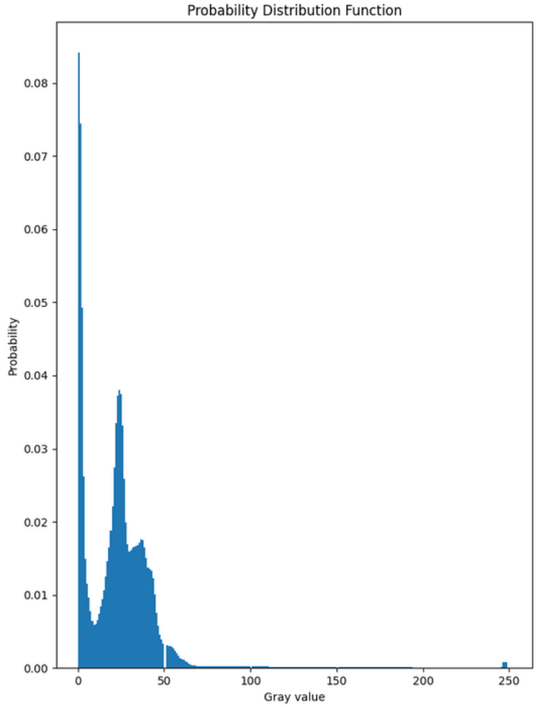
1. Converted the image to grayscale and got its PDF and CDF
2. Created an ideal CDF [linear from 0 to 1]
3. Applied histogram backprojection by using numpy's interp function. I used the x and y values of the desired CDF as the basis of my interpolate function. Then the x values I used to evaluate the interpolated function is CDF[flattened grayscale image] (what I did here is that I used the grayscale image pixel values as the indices to call the corresponding values of the CDF. So for each pixel in the grayscale image, I got its corresponding CDF value, and used it to evaluate my interpolated function to create my new image).
4. Normalize the values of the new image back to the range of 0 to 255 because the values of the image after interpolation is in the range of 0 to 1. Normalizing the values back to 0 to 255 makes the comparison between the before and after PDFs easier.
5. Reshape the image back into the original image's shape
6. I repeated steps 2 through 5 but instead used a sigmoid CDF.

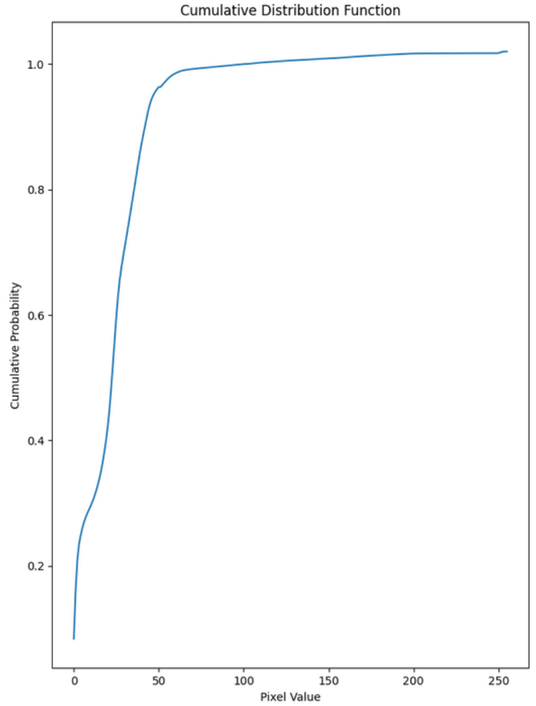
My output can be seen in the succeeding slides.

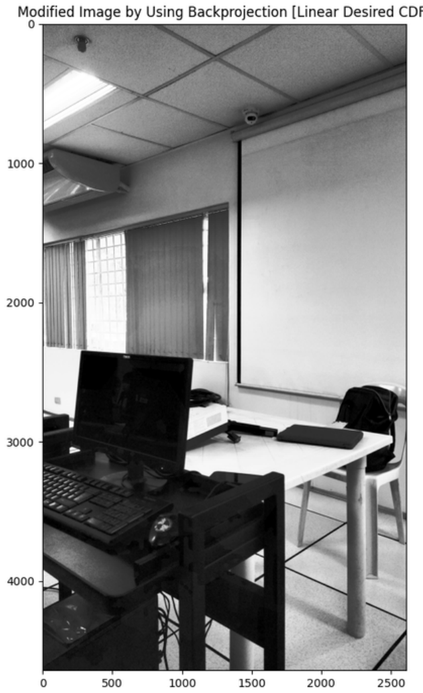
1.4 Output of a linear desired CDF

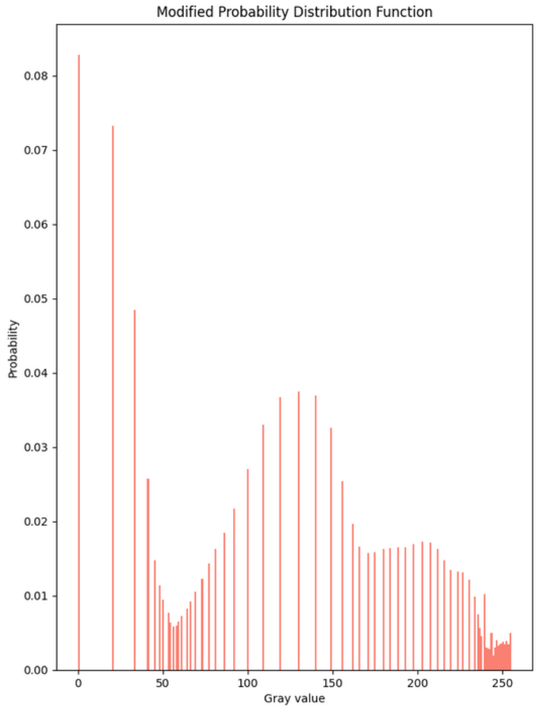
By applying histogram backprojection with a linear desired CDF, it can be seen that the image becomes brighter. Looking at the output image's PDF, its values are "spread out" across the histogram, which consequently brightens up the image. The output image's CDF also has a linear behavior. This is a result of the backprojection because the original image's CDF was interpolated into the desired CDF. Although the output CDF is not exactly like the desired CDF, it clearly has a linear trend. Thus showing that I have successfully applied histogram backprojection on my darkened grayscale image.

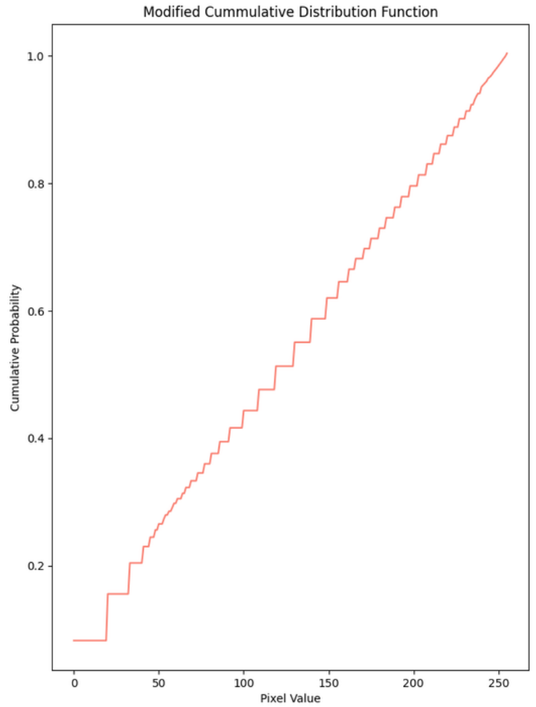






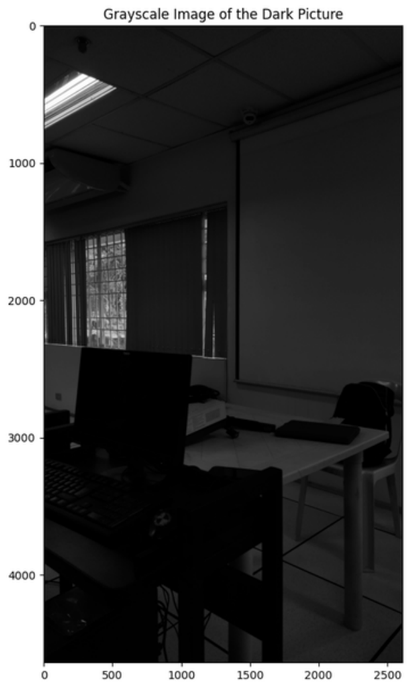


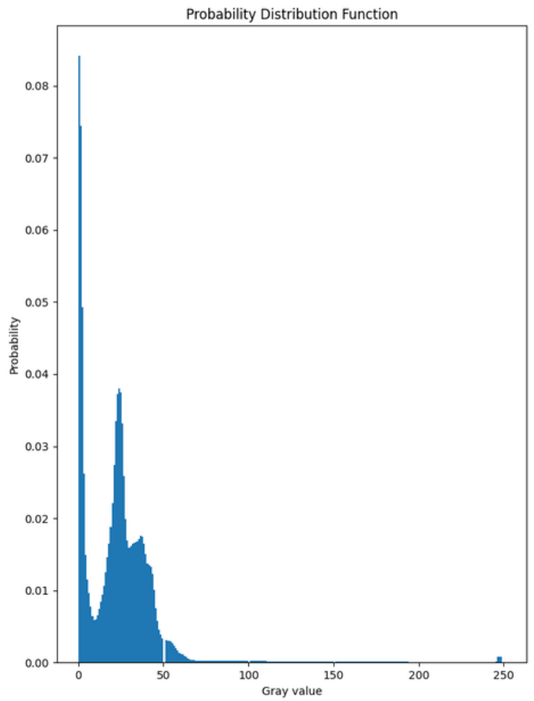


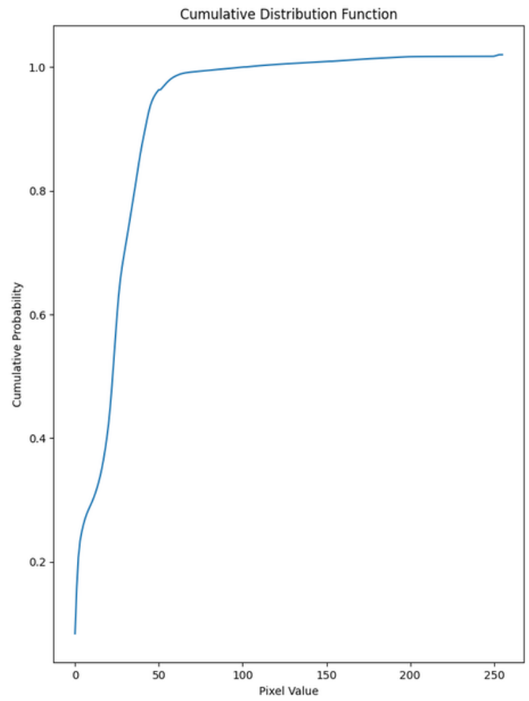


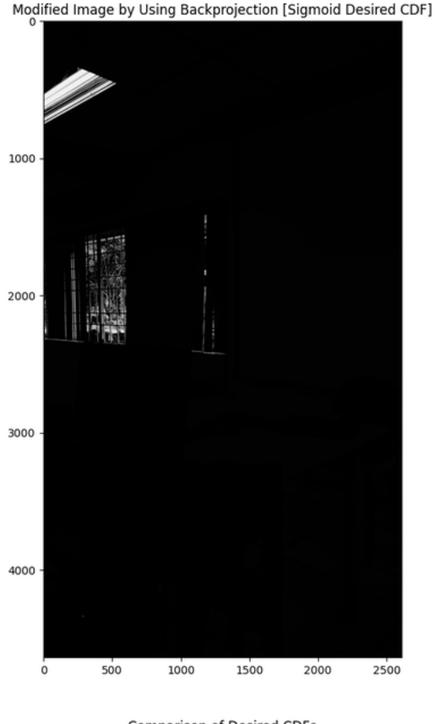
1.4 Output of a sigmoid desired CDF

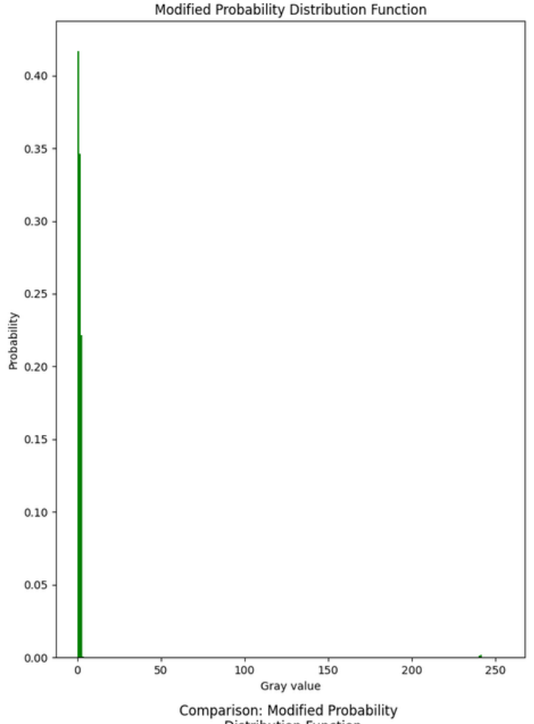
By applying histogram backprojection with a sigmoid desired CDF, it can be seen that the image becomes darker. Looking at the output image's PDF, its values look like they are compressed to the left side of the histogram, which consequently darkens the image. This condensation of values in the histogram can also be seen in the CDF where the output CDF "jumps" from ~0.45 to ~0.9 at low pixel values.

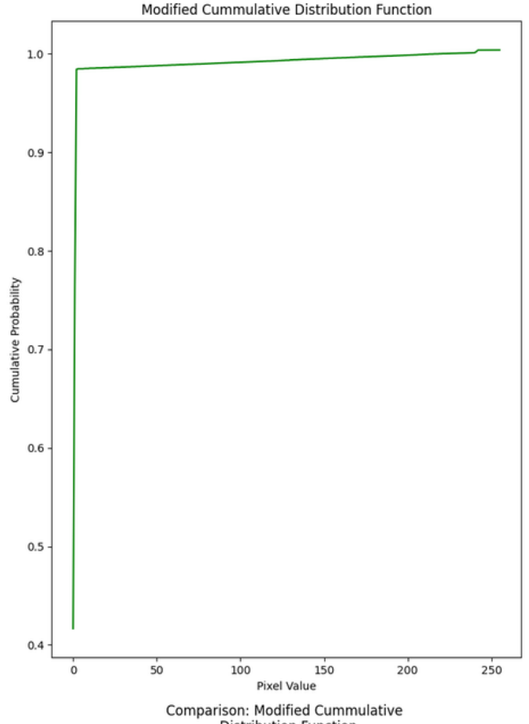




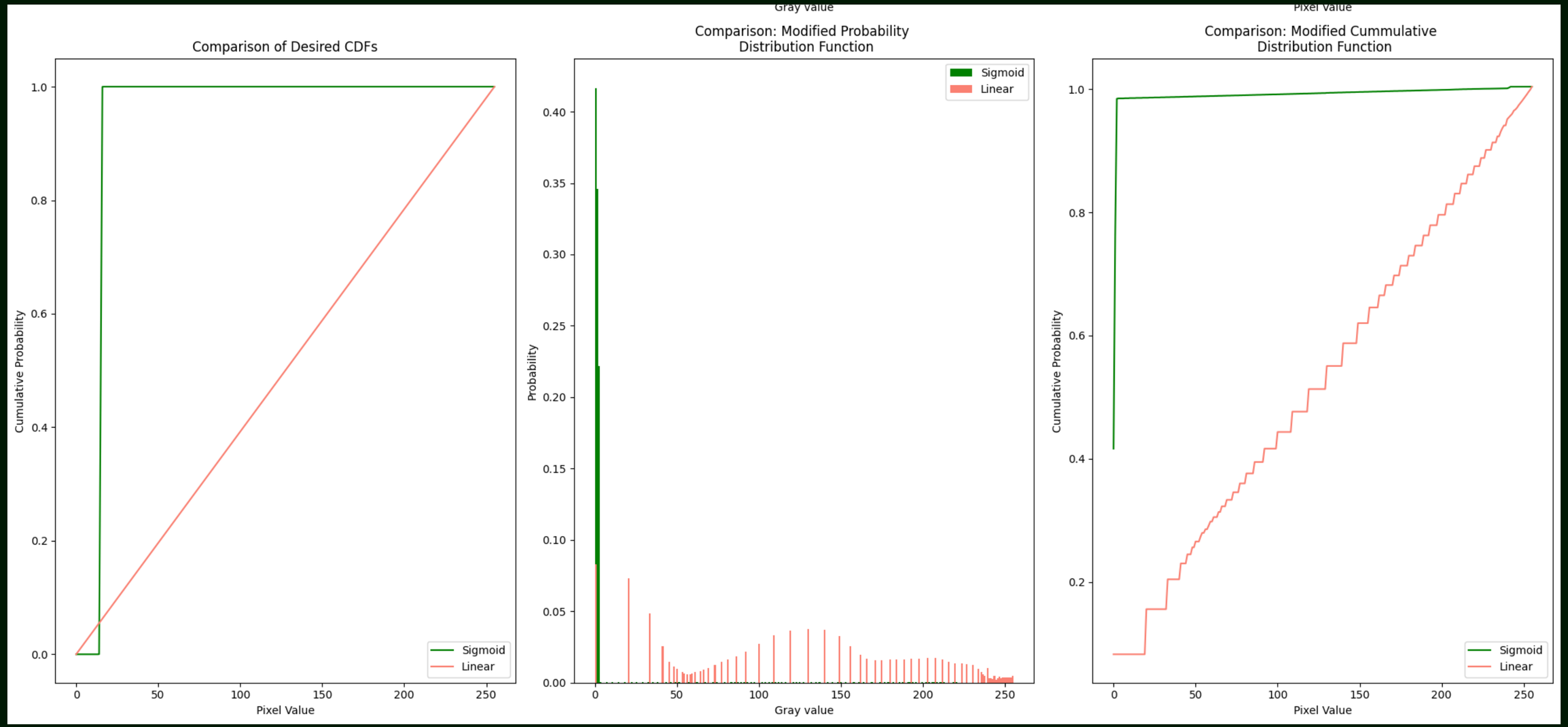








1.4 Comparison of different desired CDFs



linear = x/255 ; sigmoid = 1/(1 + exp(-(8.5*x-(255/2))))

1.5 Contrast Enhancement

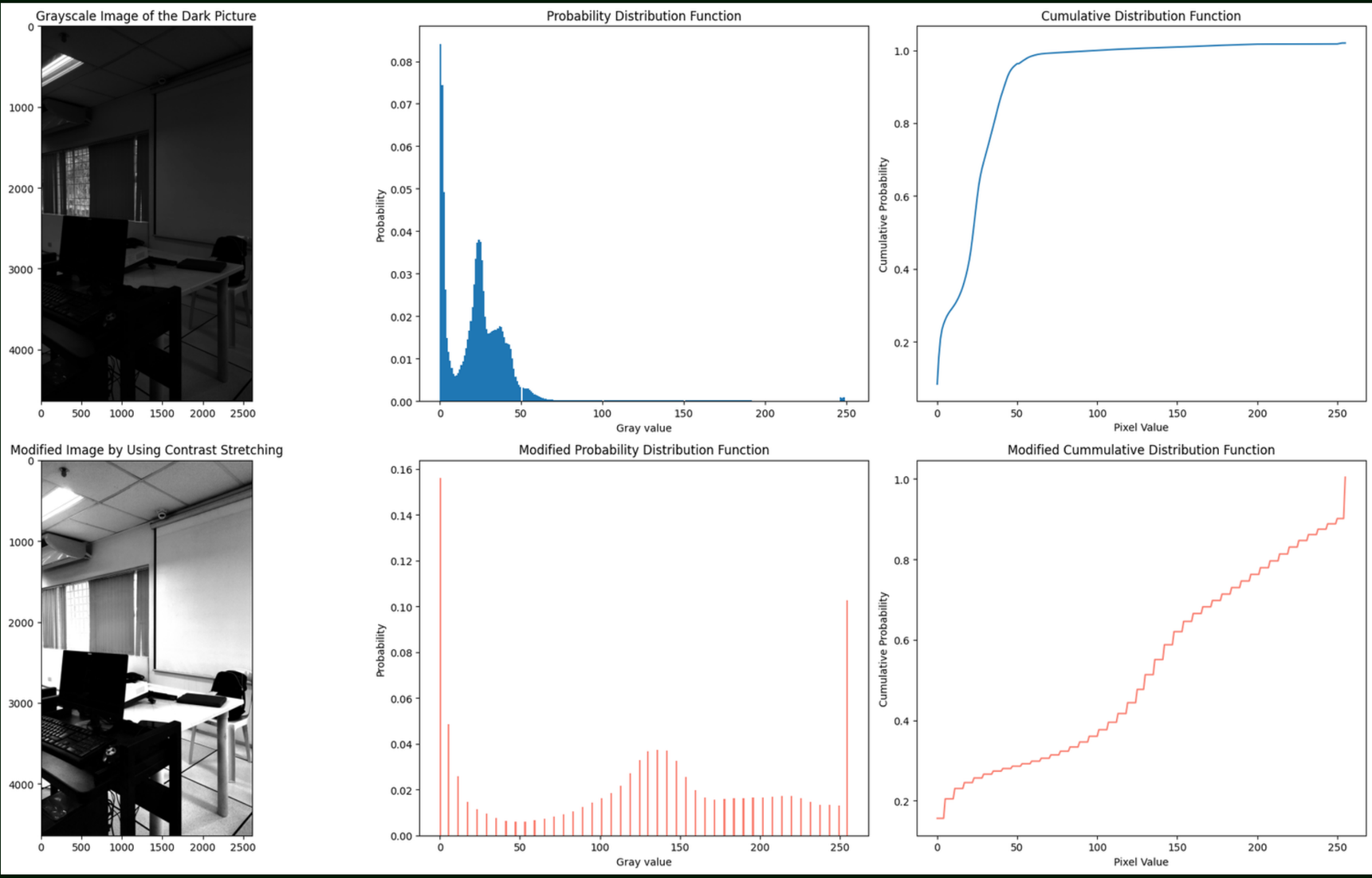
For contrast enhancement, I used the same picture in 1.4. The steps that I did are as follows:

1. Got the 90th percentile and 10th percentile of the image values and set them as the maximum and minimum values respectively (as suggested by the laboratory manual).
2. I then normalized the values by using the following formula: I_new = 255*(I_old - maximum)/(maximum - minimum). This formula was given in the laboratory manual.
3. I then clipped the values above 255 by settting them to 255. I did the same for values below 0 by setting them to 0.

My output for the contrast enhancement are in the succeeding slides

1.5 Output for Contrast Enhancement

Comparing the two images, it can be seen that the brighter parts became brighter and the dark parts became darker in the output image. This can be explained in the output's PDF. Its histogram values are "stretched", which means its histogram values are more spread out across the pixel value range. As a result, the contrast between dark and light objects in the picture is enhanced [3].

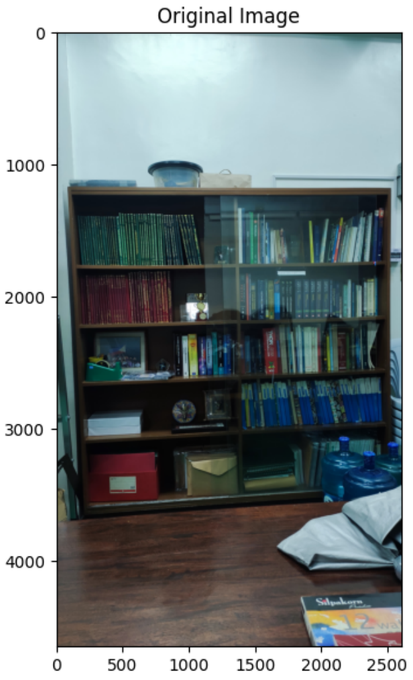


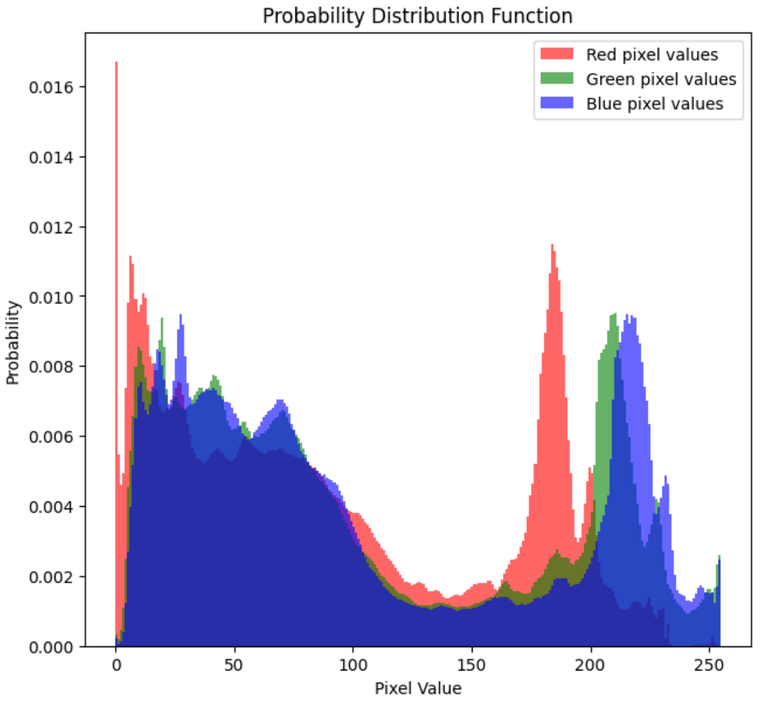
1.6 Restoring Faded Colored Photographs: Contrast Stretching

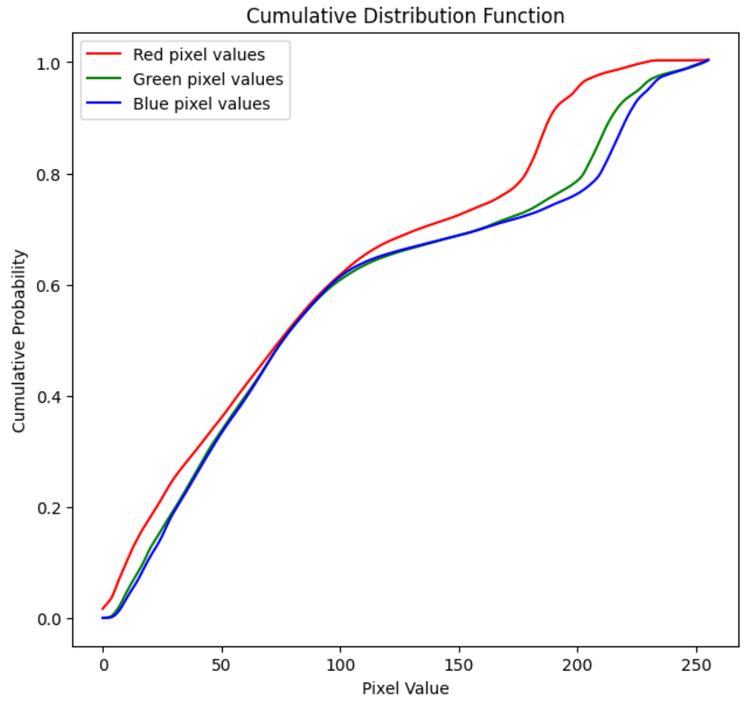
The picture that I used for Activity 1.6 is the bookshelf in IPL-VIP R202. The image contained a faint blue hue, which is due to my phone's attempt to adjust the image's color based on the lighting. For the contrast stretching, I did the same steps that I used in 1.5, but I no longer converted the picture to a grayscale image. I also did not normalize the pixel values back to 0 to 255. After contrast stretching, I let the image values be in the range of 0 to 1 to minimize conversion errors. I also clipped values that are greater than 1 and less than 0 accoridingly. After taking account of these adjustments, I then applied the steps to each RGB channel of the picture. By doing this, the faint blue hue of the original image was removed and the colors of the books and other objects in the photo were corrected (i.e. red books became more red, white walls that looked blue became whiter, brown shelf and table became more brown). My output can be seen in the succeeding slide.

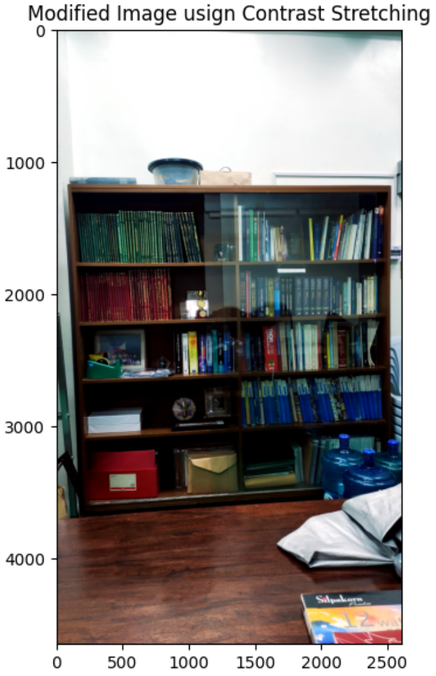
1.6 Restoring Faded Colored Photographs: Contrast Stretching

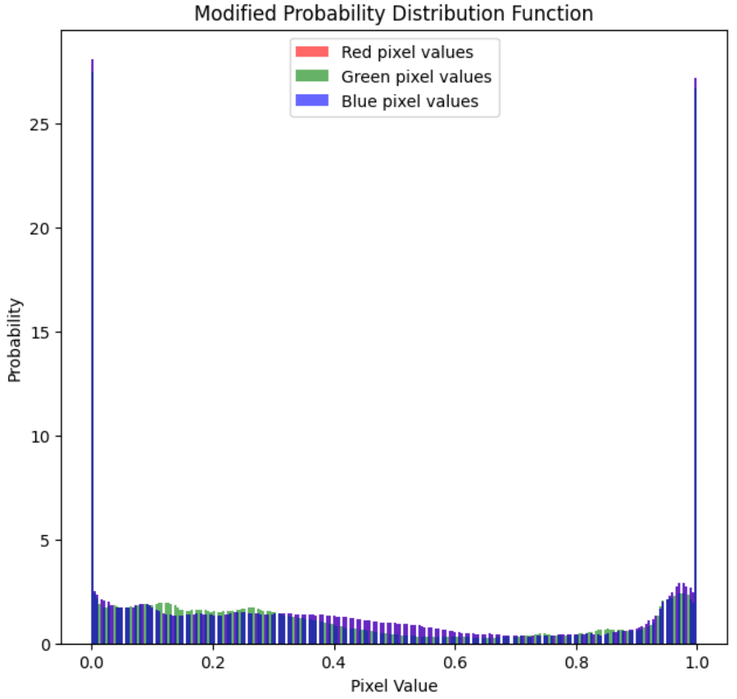
Comparing the two images, it can be seen that before contrast stretching, the image has a faint blue hue. The intensity of its RGB values are also not uniformly distributed. This is evident in how the RGB channels overlap in the image's PDF and CDF. But after contrast stretching, it can be seen that the white, red, green, and blue colored objects are more vibrant. It can also be seen that the distribution of pixel values in the output's PDF is more uniformly distributed. This is evident in the output's CDF as the RGB channels have many overlaps.

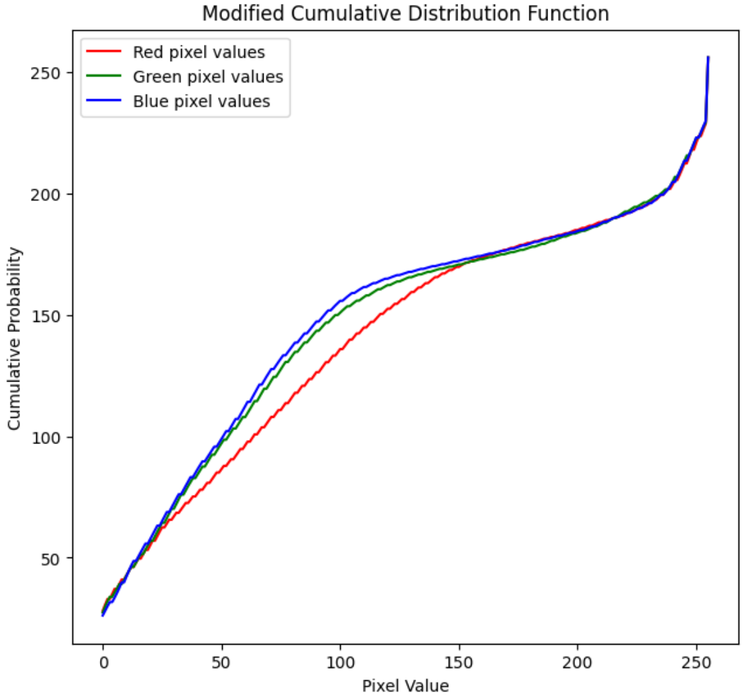












1.6 Restoring Faded Colored Photographs: Gray World Algorithm

For the Gray Worl Algorithm, I did the following steps:

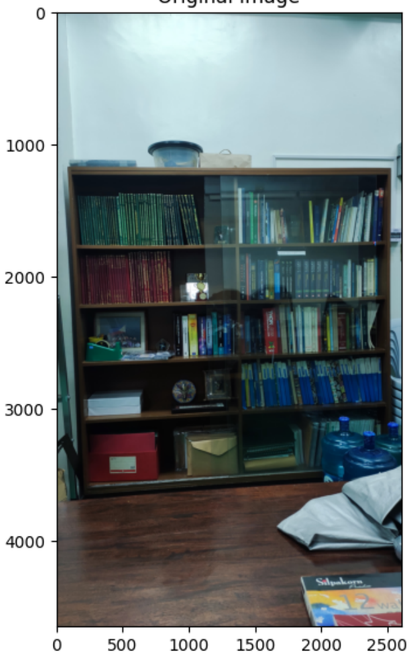
1. Divided each RGB channel with their respective averages
2. Multiplied all the elements in the photo by a factor of 0.6 (as suggested by Ma'am Jing)
3. Normalized the image values from 0 to 1
4. I then tried different multiplying factors to see its effects on the Gray World Algorithm.

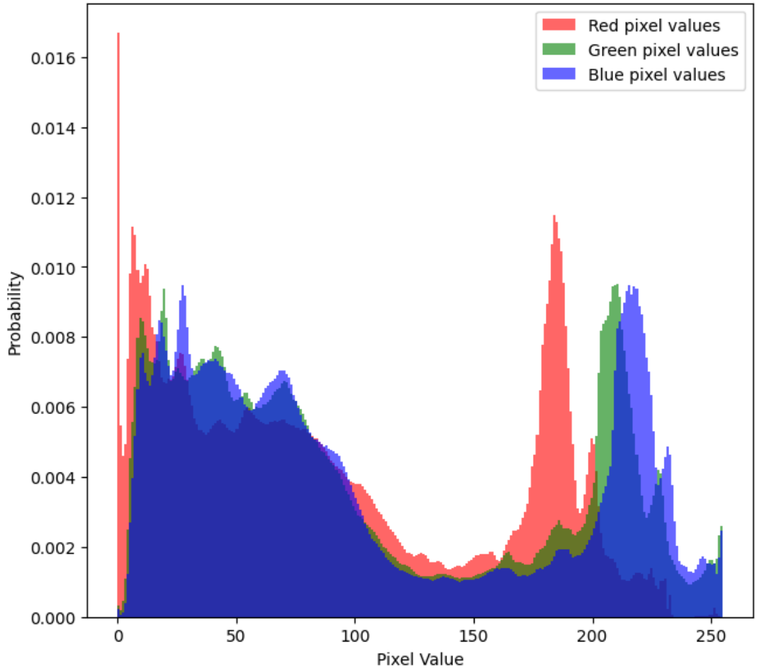
My outputs can be seen in the succeeding slides.

1.6 Restoring Faded Colored Photographs: Gray World Algorithm

Looking at the output image, it is similar to the output image from contrast stretching. The faint blue hue in the original photo is removed and the colors of the objects are more vibrant. Looking at the output PDF, it may look the same as the original PDF but it is actually stretched. This is because its x values are from 0 to 1.5, this means that its histogram bins are more dispersed than the original photo's histogram.

Original Image





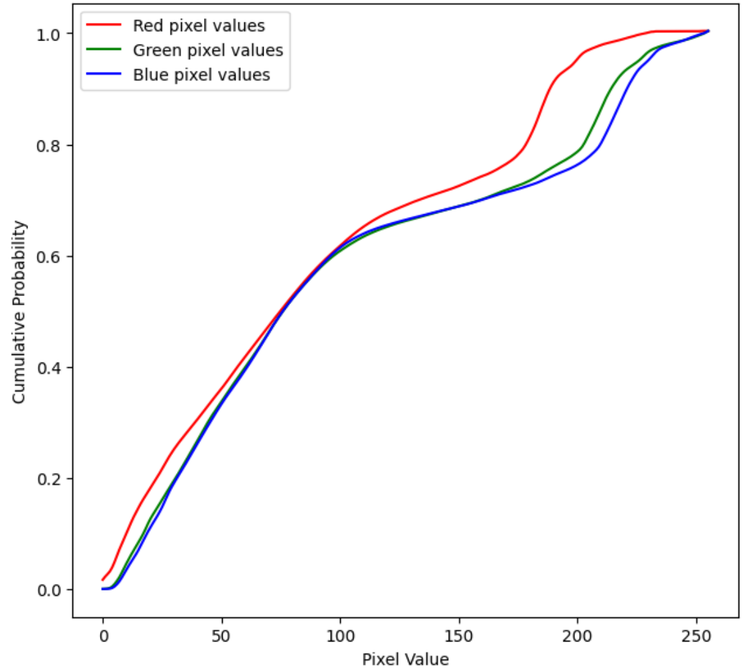
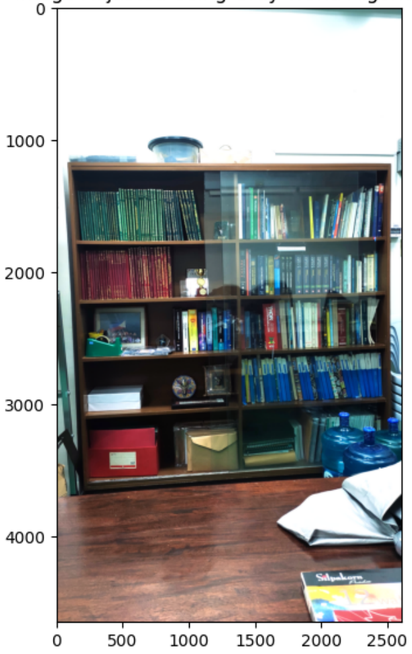
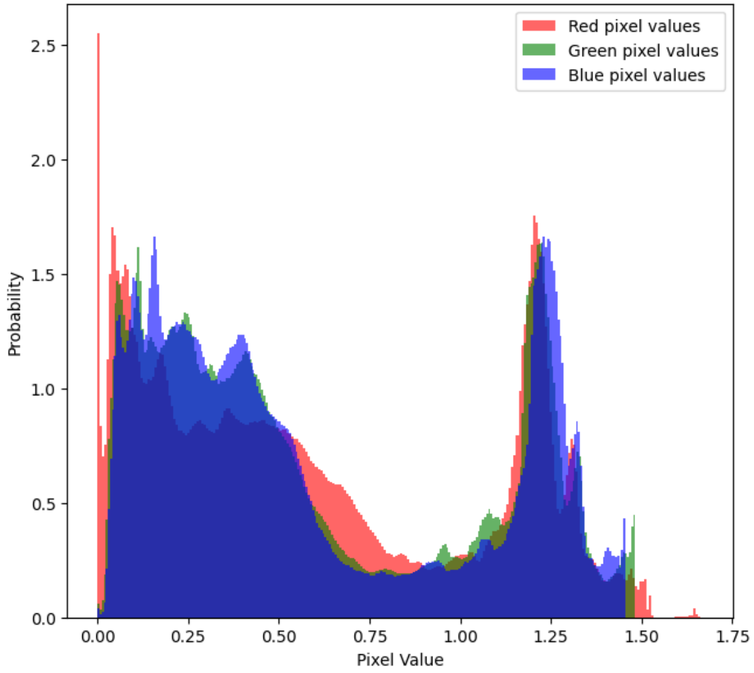
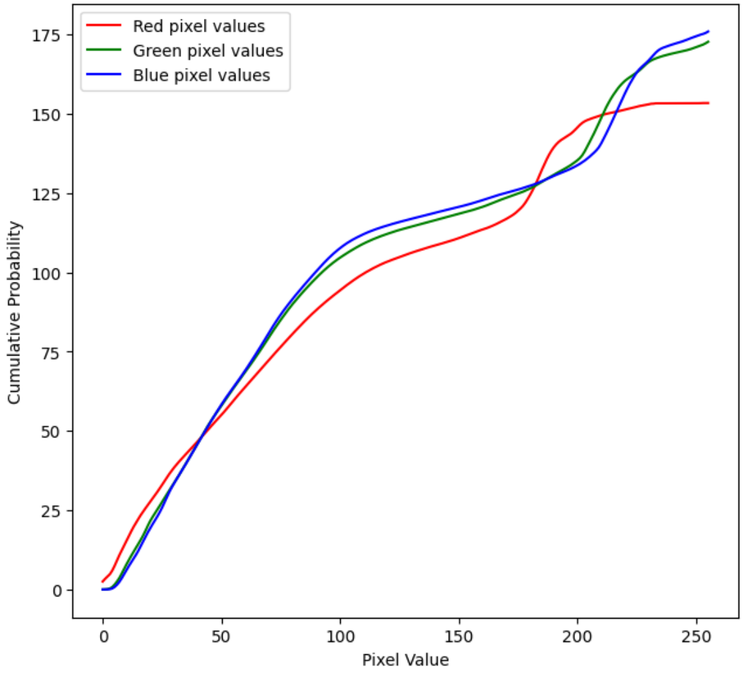


Image Adjusted Using Gray World Algorithm

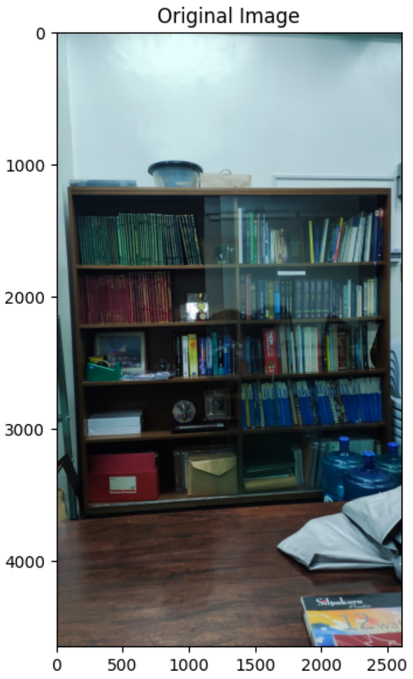


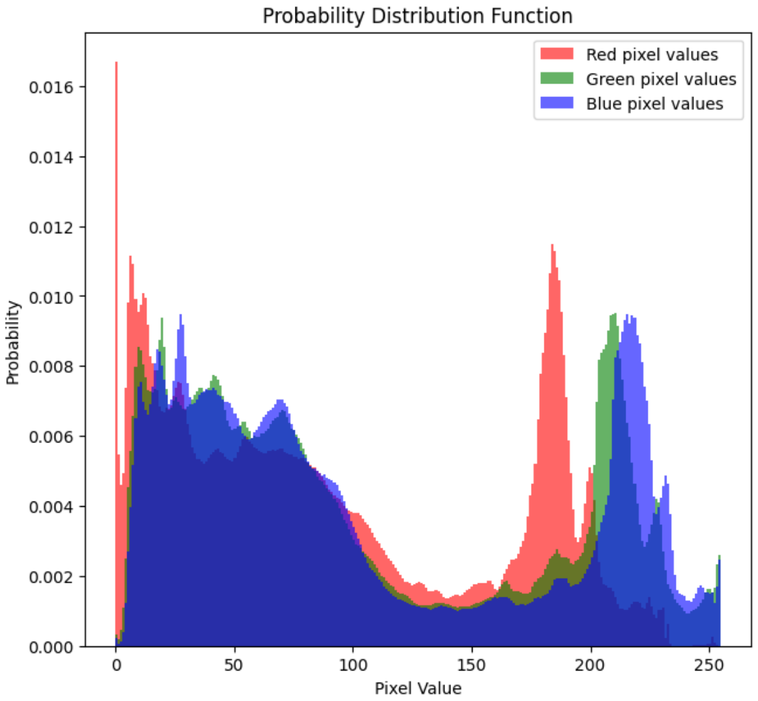


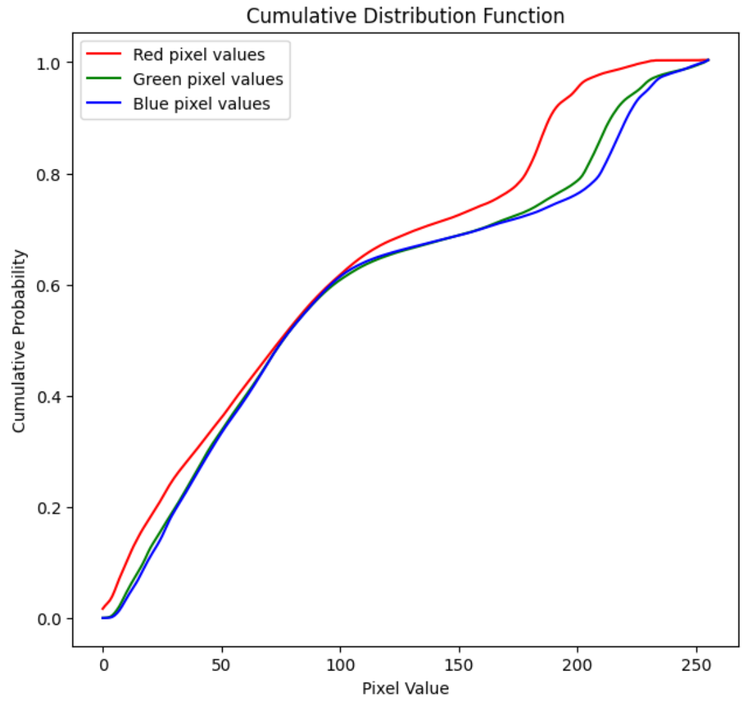


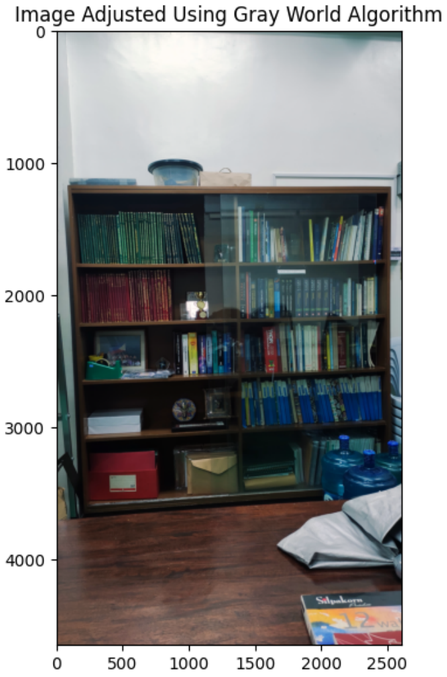
1.6 Restoring Faded Colored Photographs: Gray World Algorithm

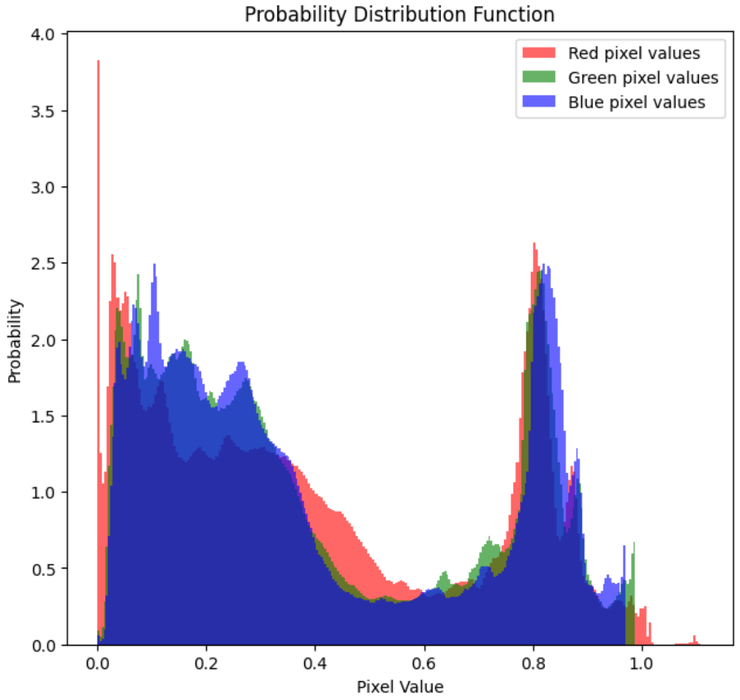
By setting the multiplier equal to 0.4, the overall image is darker. The blue hue is removed and it appears that the objects are not as vibrant. Looking at the output PDF, its x-axis spans from 0 to ~1.2. This means that decreasing the multiplying factor decreases the amount at which the PDF is stretched.

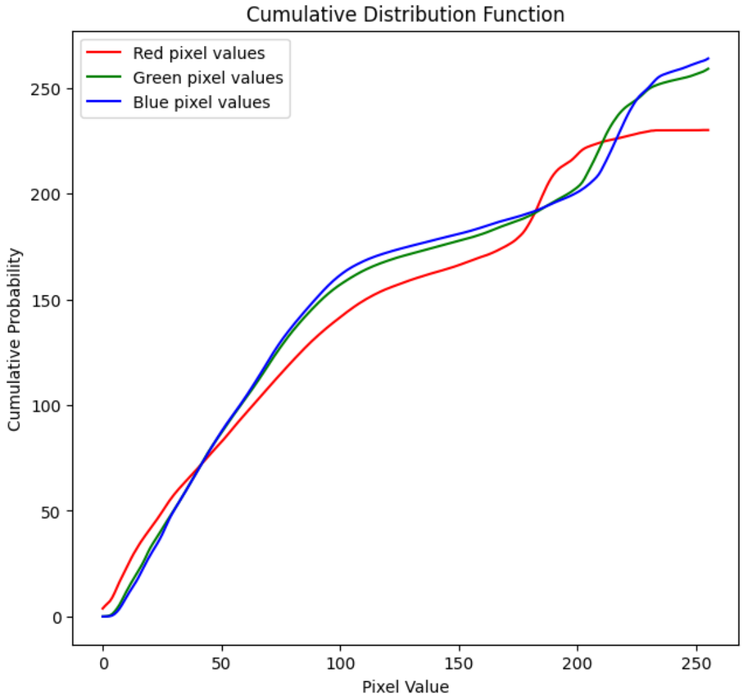








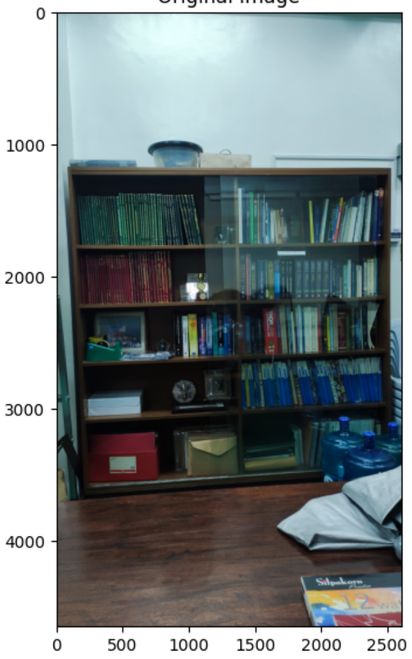


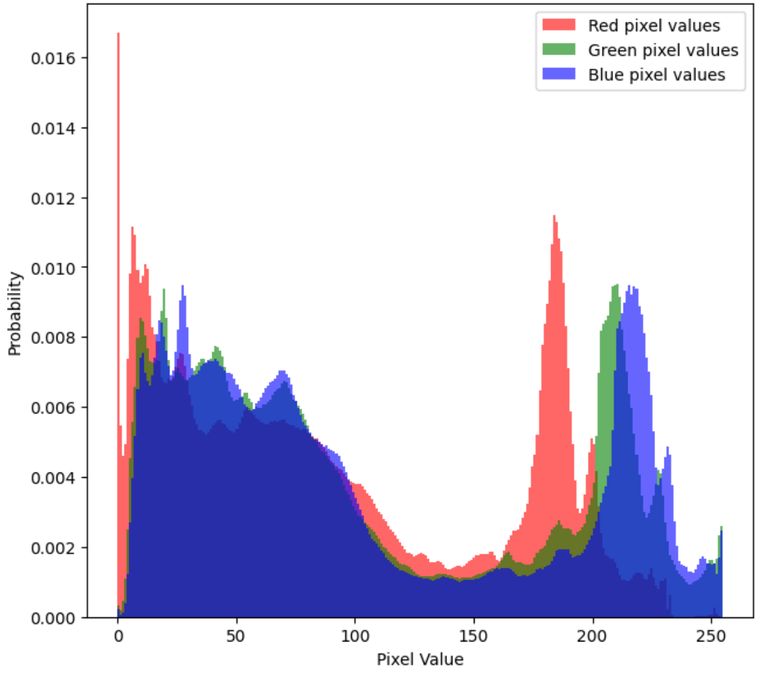


1.6 Restoring Faded Colored Photographs: Gray World Algorithm

By setting the multiplier equal to 1, the overall image is brighter, and its PDF now spans from 0 to ~3. This means that increasing the multiplier stretches out the image's PDF even more and assigns higher pixel values to the entire image.

Original Image





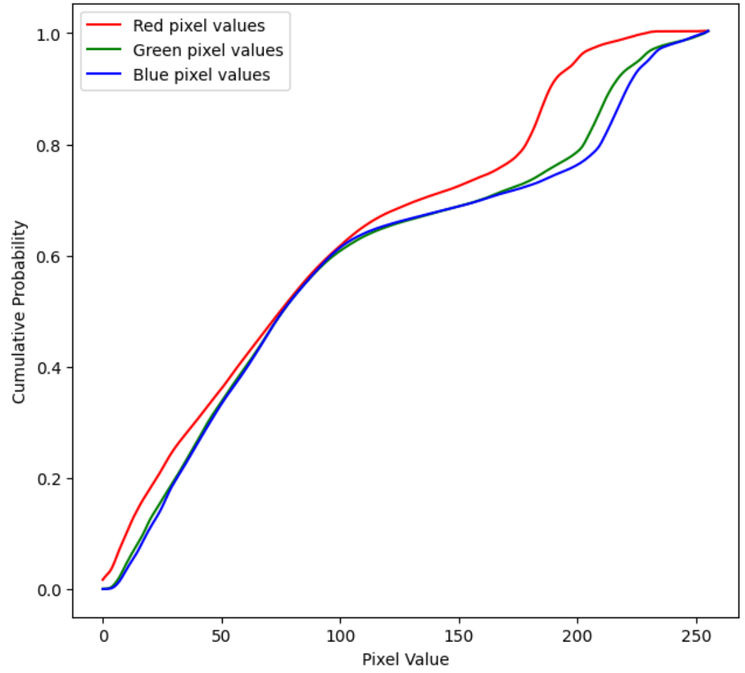
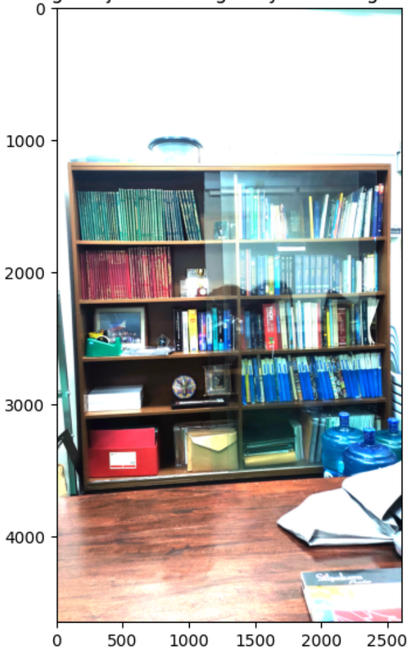
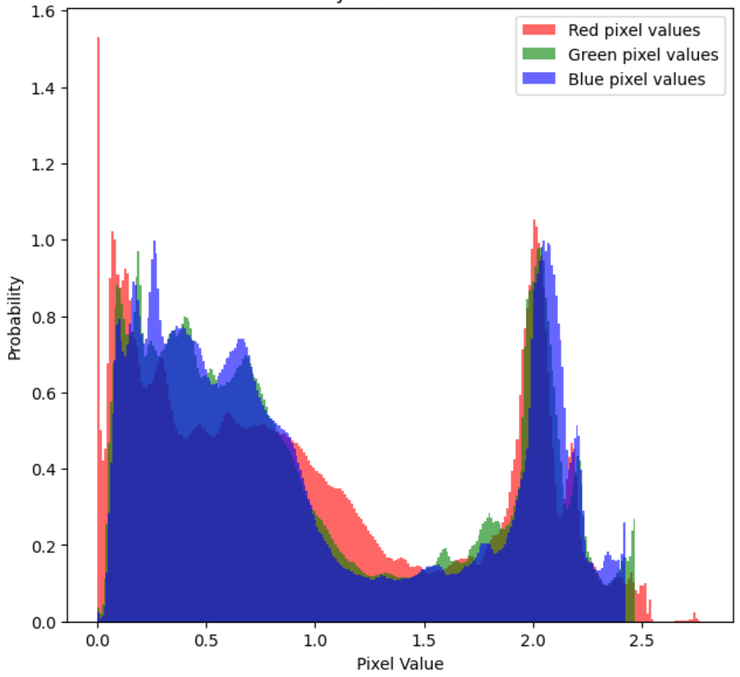
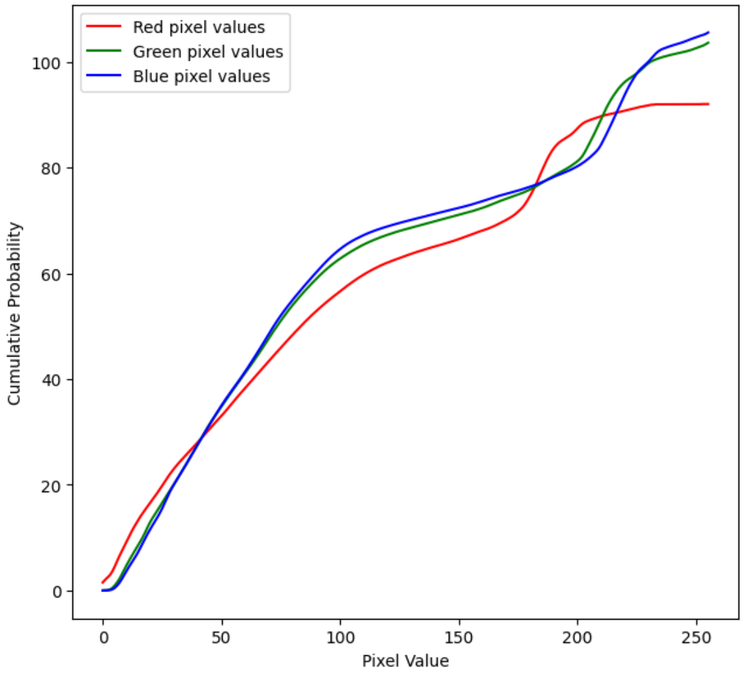


Image Adjusted Using Gray World Algorithm







1.6 Restoring Faded Colored Photographs: White Patch Algorithm

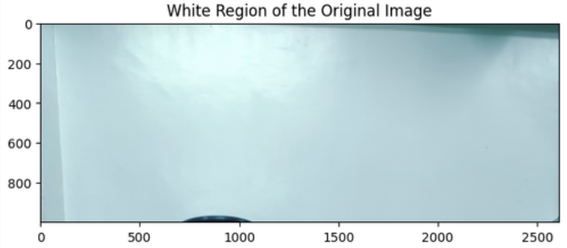
For the White Patch Algorithm, I only used the top portion of my image. This is because the top portion is known to be a white wall, but it has a blueish color in the image. The steps I used in implementing the White Patch Algorithm is as follows:

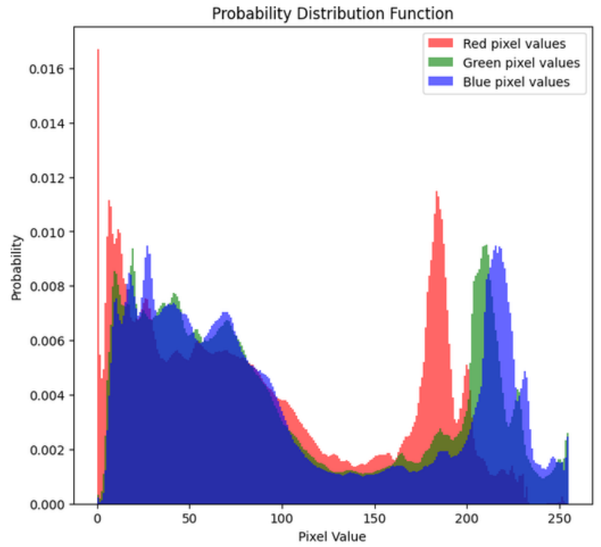
1. Get the average of each RGB channel in the white region.
2. Divide each RGB channel with their respective averages.

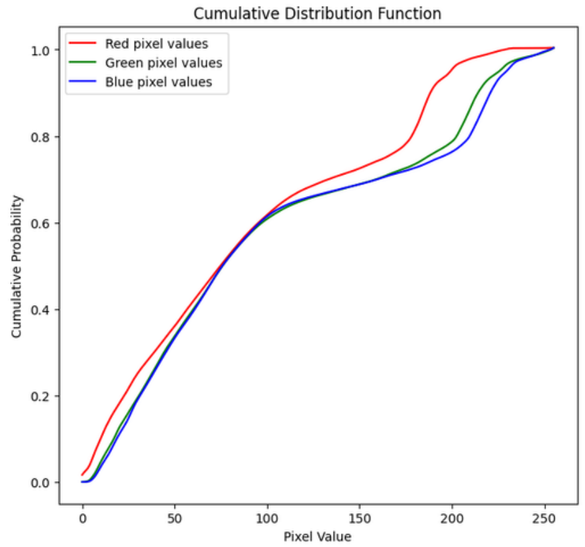
My outputs can be seen in the succeeding slides.

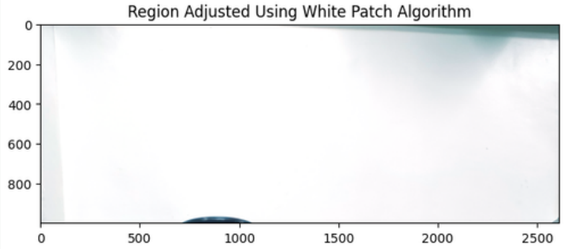
1.6 Restoring Faded Colored Photographs: White Patch Algorithm

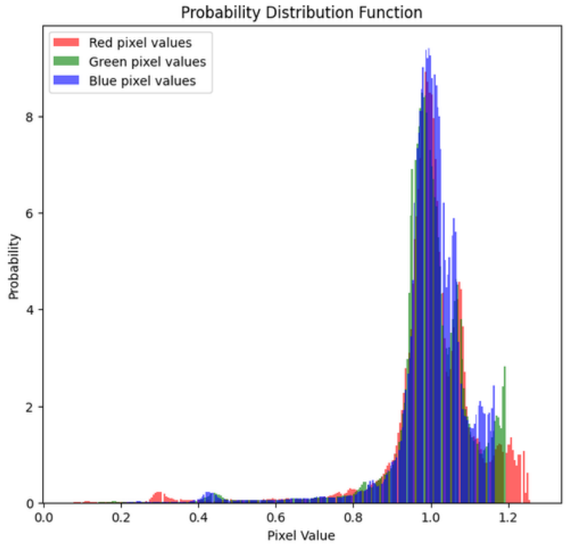
Comparing the two images, It can be seen that the White Patch Algorithm removed the faint blue hue and it shows that the wall is really white. It Also shows that the algorithm increased the pixel values of the image. This is evident in the output's PDF, where the bins are shifted to the right side of the graph. It is also evident in the output's CDF where it suddenly increases at around the ~200 pixel value mark

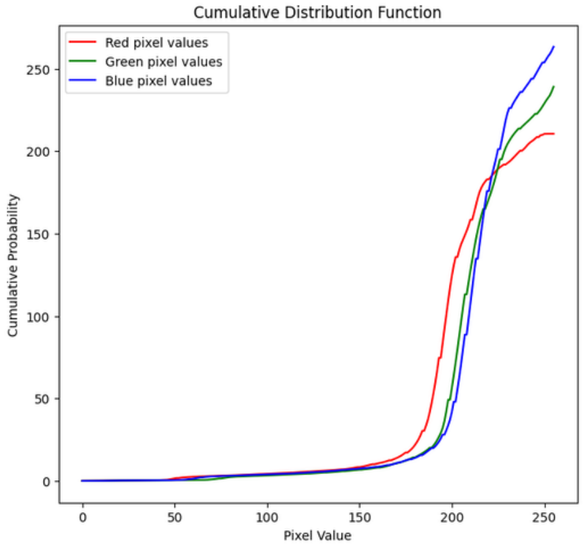












Reflection

I really love this first activity, mostly because I was able to collaborate with my classmates, instructors, and professor. For this activity, I think all of my output images are visually correct. However, I am not that confident on how I processed the images. I am still not sure if I handled the data correctly, especially for the contrast stretching parts because I get weird outputs when I tried to normalize the range of the pixel values back to 0 to 255. I am also not content with my explanations, they seem a little bit too bland for me and I don't have enough time to make them better.

Acknowledgements

I would like to acknowledge the people who helped me in understanding and completing this activity. I am very thankful to my professor and instructors: Ma'am Jing, Sir Rene, and Sir Kenneth for clearly answering all of my inquiries about the activity. I would also like to thank my friends and classmates namely: Ron, Edneil, Ja, Jonabel, Johnenn, and Lovely for helping me out with my code and helping me understand the activity.

Self Grade

Technical Correctness	I understood the lesson and met all the objectives. However, I do not think my results are verifiably correct. I think they can still be improved along with my analysis.	**25**
Quality of Presentation	The images I added to this report are of good quality and all the graphs are properly labelled. But I didn't expect that my code will also be submitted so its explanations are lacking.	30
Self Reflection	I got the expected results, and acknowledged the contributions of my peers while doing this activity. I also properly cited onlien references.	30
Initiative	I mad improvements to the code that I found online and fixed their bugs and errors.	5
	Total	90

References

[1] Applied Physics 157 Digital Image Formation and Enhancement Laboratory Manual

[2] RajuKumar19. (2022, October 7). Matplotlib.patches.RegularPolygon class in Python. GeeksforGeeks. Retrieved March 10, 2023, from https://www.geeksforgeeks.org/matplotlib-patches-regularpolygon-class-in-python/

[3] Boone, J. (2020, September 14). A beginner's guide to color curves for powerful correction. Frame.io Insider. Retrieved March 10, 2023, from https://blog.frame.io/2017/09/20/beginners-guide-to-color-curves/

[4] Kang, & Atul. (2019, April 19). Contrast stretching. TheAILearner. Retrieved March 10, 2023, from https://theailearner.com/2019/01/30/contrast-stretching/