APPLIED PHYSICS 157

Activity 1

DIGITAL IMAGE FORMATION AND ENHANCEMENT

JONABEL ELEANOR BALDRES

WFY-FX-2

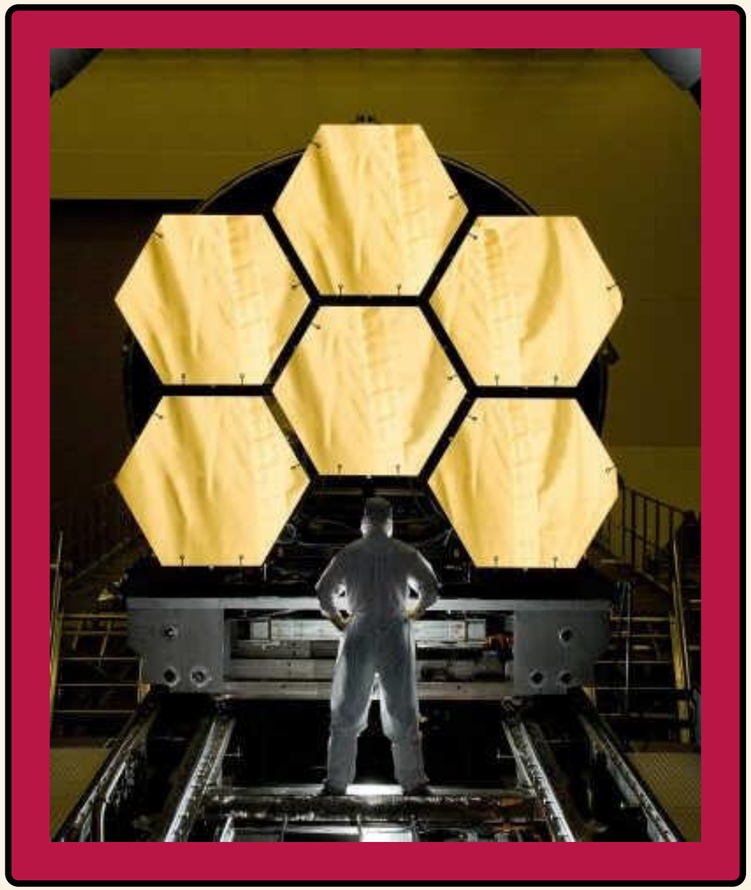
table of CONTENTS

The outputs presented in the succeeding pages are created using MATLAB. Moreover, the codes are uploaded in Github.

IMAGE DIY

Objectives:

- Mathematically create images of the follow
  - Sinusoid
  - Grating
  - Hubble's Primary Mirror
  - Hexagon Array



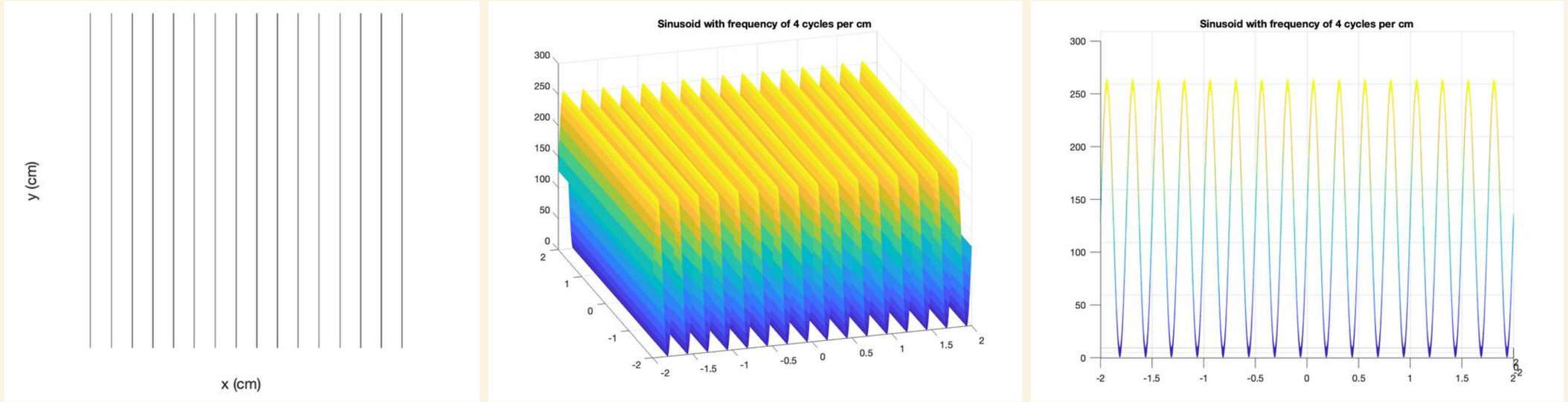


Figure 1. Sinusoid along the x-direction with frequency of 4 cycles per centimeter. The X and Y ranges from -2 cm to 2 cm. From left to right: (a) two-dimensional view, (b) three-dimensional mesh view, (c) three-dimensional mesh view with Z values ranging from 0 to 250 cm.

In Figure 1, I created a sinusoid with an equation of sin(8Xπ) with X being the linear space from -2 to 2. Since physical images do not have negative values, I used the rescale function to rescale the entire array from 0 to 255. I then created the figure for its 2-dimensional and 3-dimensional planes as shown above. From the image you can see that X and Y are within -2 and 2 while the Z value has no negative values.

Sinusoid

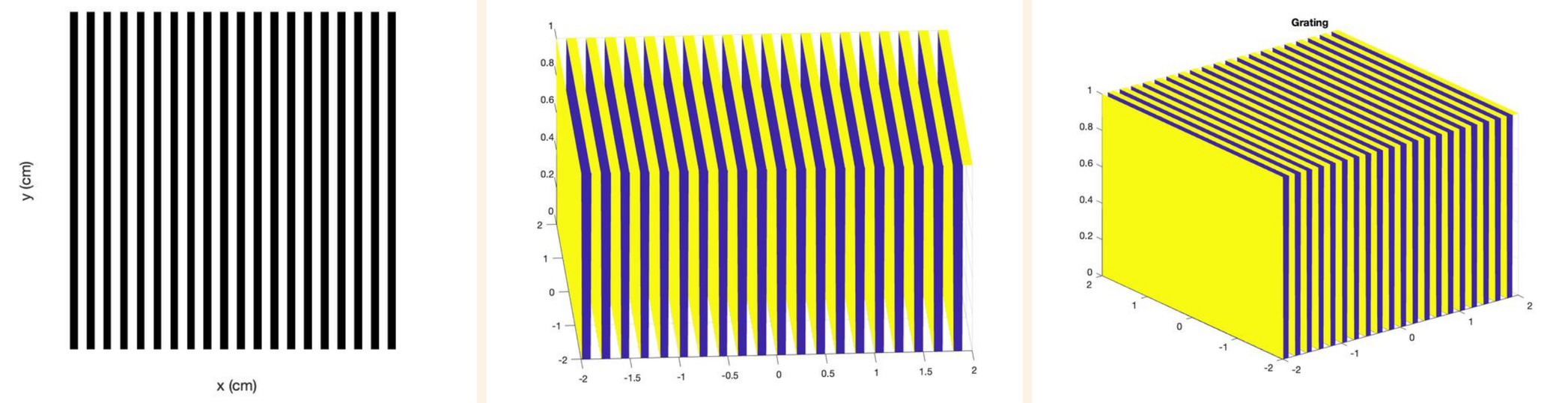


Figure 2. Grating with frequency of 5 line pairs per centimeter. The X and Y ranges from -2 cm to 2 cm. From left to right: (a) two-dimensional view of grating, (b) three-dimensional mesh view, (c) three-dimensional mesh view with Z values ranging from 0 to 1 cm.

Figure 2 shows grating created using the equation R = sin(10Xπ) with X (Y) from range -2 to 2. Here I created variable A which contains zeros with size R. I then manipulated its contents such that when R is less than 0.1, A is equal to 1. After that, I created the figure for its 2-dimensional and 3-dimensional planes as shown above. From the image you can see that X and Y are within -2 and 2 while the Z is from 0 to 1.

Grating

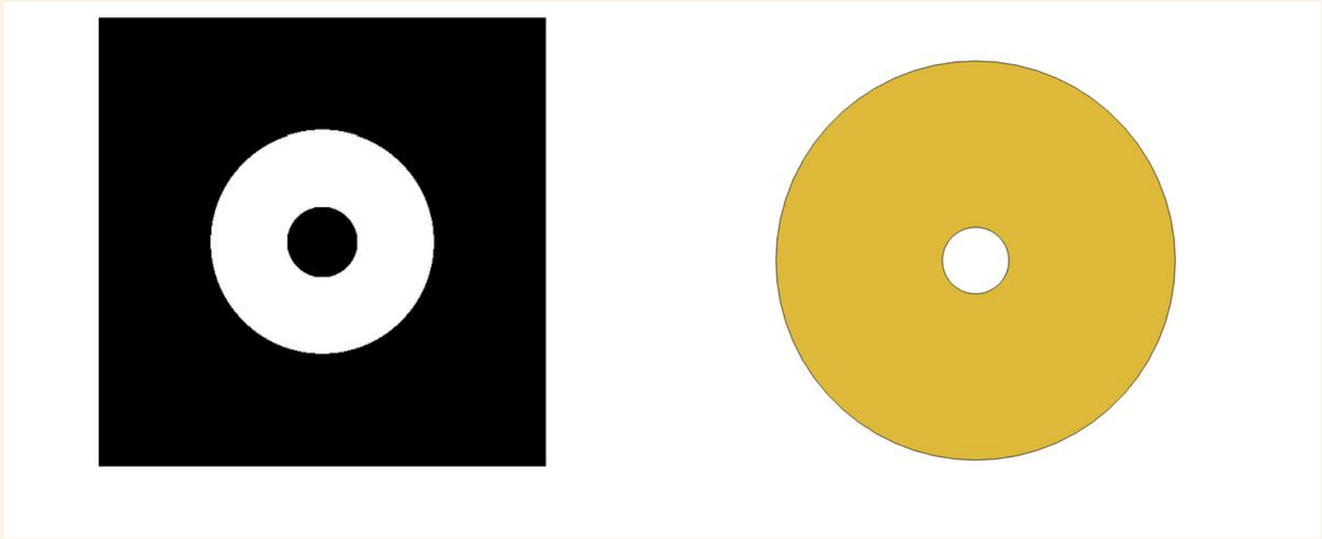


Figure 3. Annulus: circle with a hole in the middle. From left to right: (a) annulus created using mathematical equations, and (b) using ringAnnulus function.

Here, I devised two ways to create the annulus. The left image was created using mathematical equation R = X^2 + Y^2. I then created variable A which initially contains an array of zeros with the size similar to R. I then manipulated A that when R is less than 1, the zeros become 1 (white background) and when R is less than 0.1, the zeros remain as is (black background). The only drawback I encountered using this method was the inability to remove the black background surrounding the annulus. With that, I created another with the use of ringAnnular() function. The syntax of the function is:
ringAnnular(InnerRadius = x, Width= y). This created an image with no dark background. However, the drawback of this method is it does not follow the objective which is to mathematically create images.

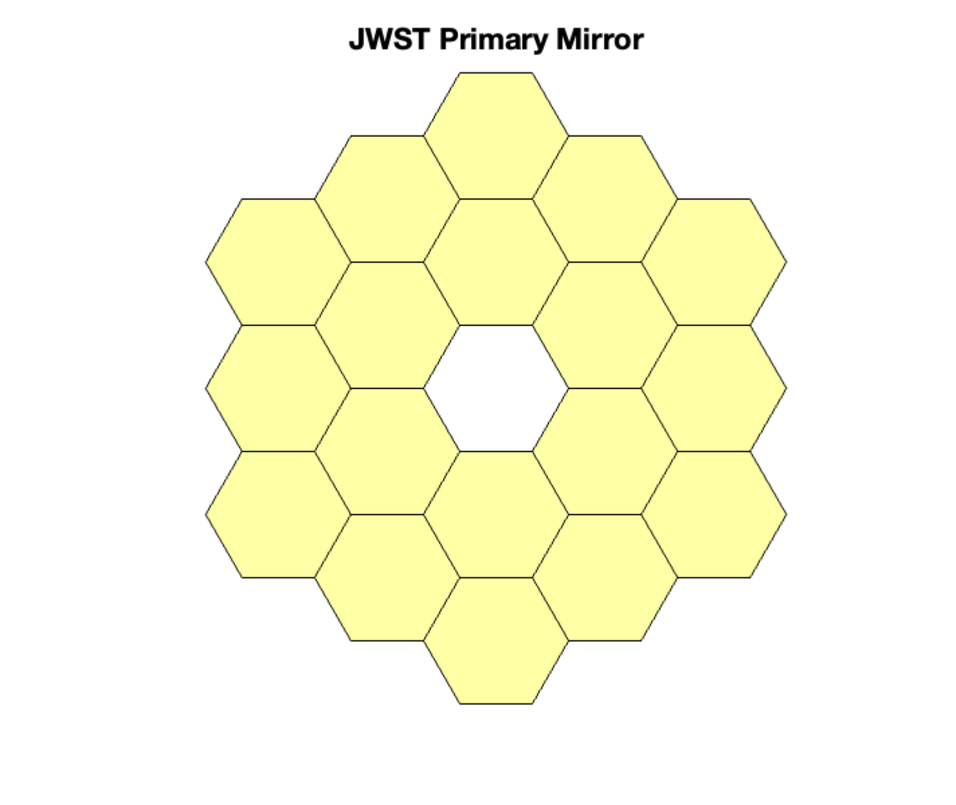


Figure 4. Hexagon Array created using nsidedpolygon() function.

In this activity, I took advantage of Matlab's nsidedpoly() function. The syntax is as follows: nsidedpoly(number_of_equal_sides, 'Center', [h k], 'SideLength', n). The tedious part of this activity is to make sure that the hexagons are clumped together with no space in between their edges and sides. To resolve this, I manually solved the points needed for the hexagons to be side by side and I also manually plotted their centers.

Overall, Activity 1.1 showed the importance of accuracy in scientific pursuits. Being able to create images helps in modeling and understanding real-life equipment without exposure to the said equipment that may cause a lot of money, time, and resources.

COLOR IMAGE

Objective:

- Mathematically recreate the Olympics logo as an image
- Compare 4 different file formats

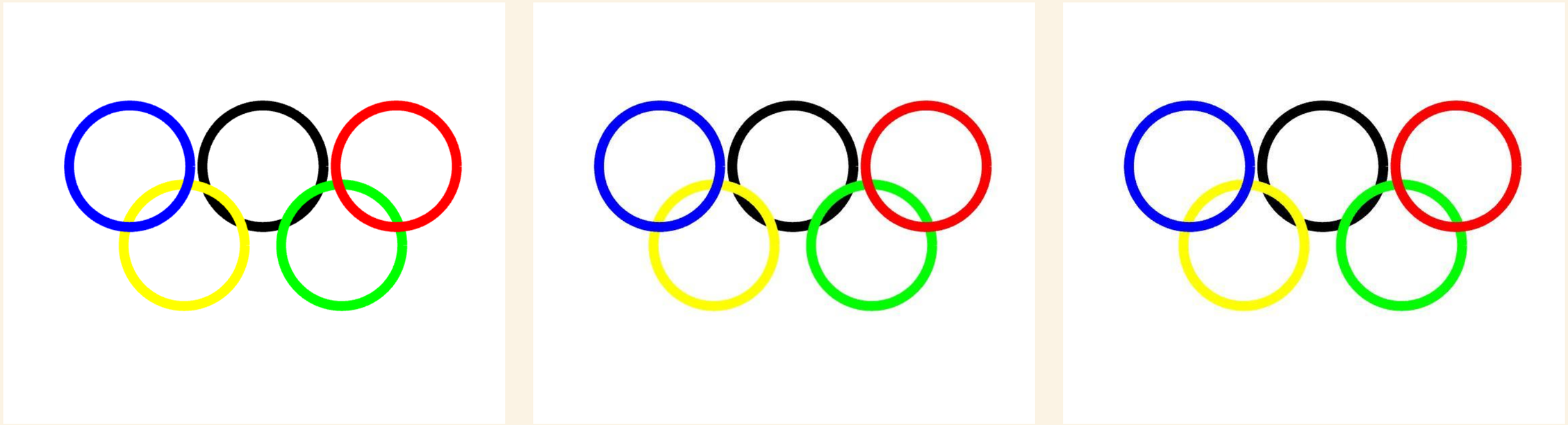


Figure 5. Mathematically recreating the Olympics logo without removing the overlapping circles. From left to right: (a) PNG format, (b) JPEG/JPG format, (c) TIFF format.

To do the Olympics logo, I created circles using the equations: x = cos(a) and y = sin(a). I added ± 1.3 and ± 2.2 to create another circles that form the Olympics logo. To add colors I use the syntax: plot(x,y, "LineWidth", n , "Color", y/g/r/b/k). Where n is the size of the line width and y/g/r/b/k represents the color of the circles.

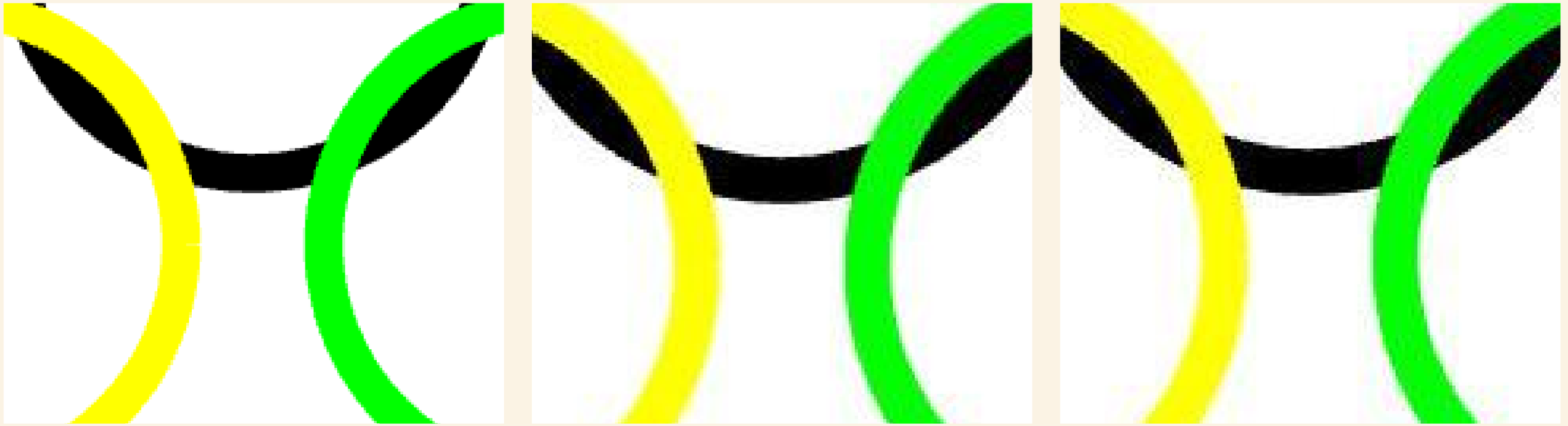


Figure 6. Enlarged view of Mathematically recreated Olympics logo without removing the overlapping circles. From left to right: (a) PNG format, (b) JPEG/JPG format, (c) TIFF format.

I saved the Olympics logo in three different formats: PNG, JPEG, and TIFF. When enlarged, we can identify from the above that the JPEG format has more edges and more grainy than the rest of the file formats. Comparing PNG and TIFF, we can see that TIFF has smoother edges in the curves than PNG.

Olympics Logo Enlarged

FOUR DIFFERENT FILE FORMATS

JPEG

Joint Photographic Expert Group image. Used for lossy compression. Significantly reduces file size of an image. Not suitable for using images in need of accuracy.

PNG

Portable Network Graphics. Used when transparency is needed. Preferred over JPEG for its better precision.

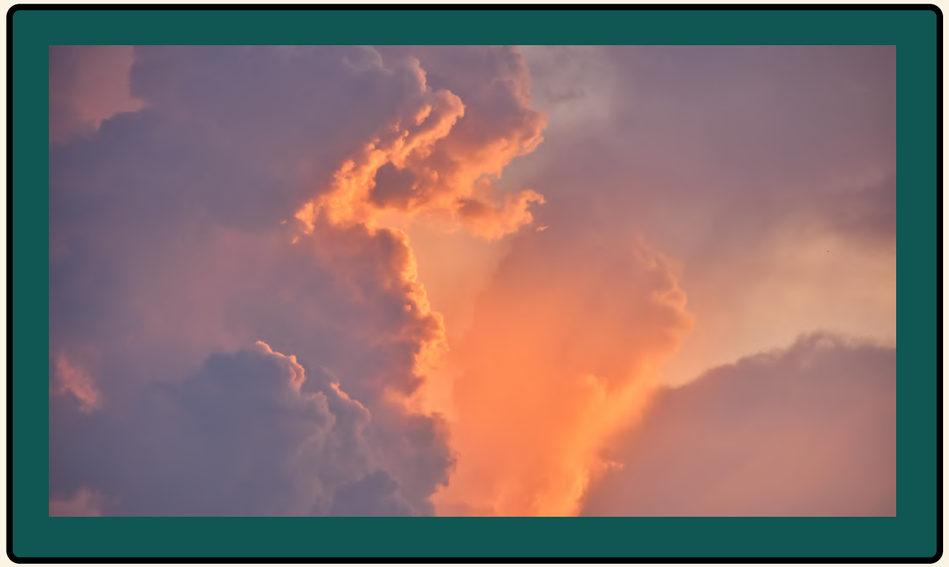
BMP

Bitmap File. Uncompressed raster image resulting to larger file size compared to JPEG and PNG. Not widely used.

TIFF

Tagged Image File Format. Lossless form of file compression creating larger sizes than those lossy compression image formats.

INPUT-OUTPUT CURVE ALTERATION



Objective:

- Enhance image by altering the input-output curve using GIMP

INPUT-OUTPUT CURVE ALTERATION

Figure 7. Input-Output curve alteration using GIMP. From left to right: (a) original image, (b) altered image

From Figure 7, we can see the difference between the original image and the altered image. In the original, the bushes in front can not be seen since the lower part of the image is dark. Aside from that, the colors of the wheels can not be identified but upon altering the image, we can see their various colors. However, looking at the altered image, there seems to be a white color overlay that makes the color of the skies and clouds more muted.

I used GIMP to alter the shape of the curve of the image. I did not have a specific curve in mind so I experimented on the best curve that can show the hidden objects in the image.

INPUT-OUTPUT CURVE ALTERATION

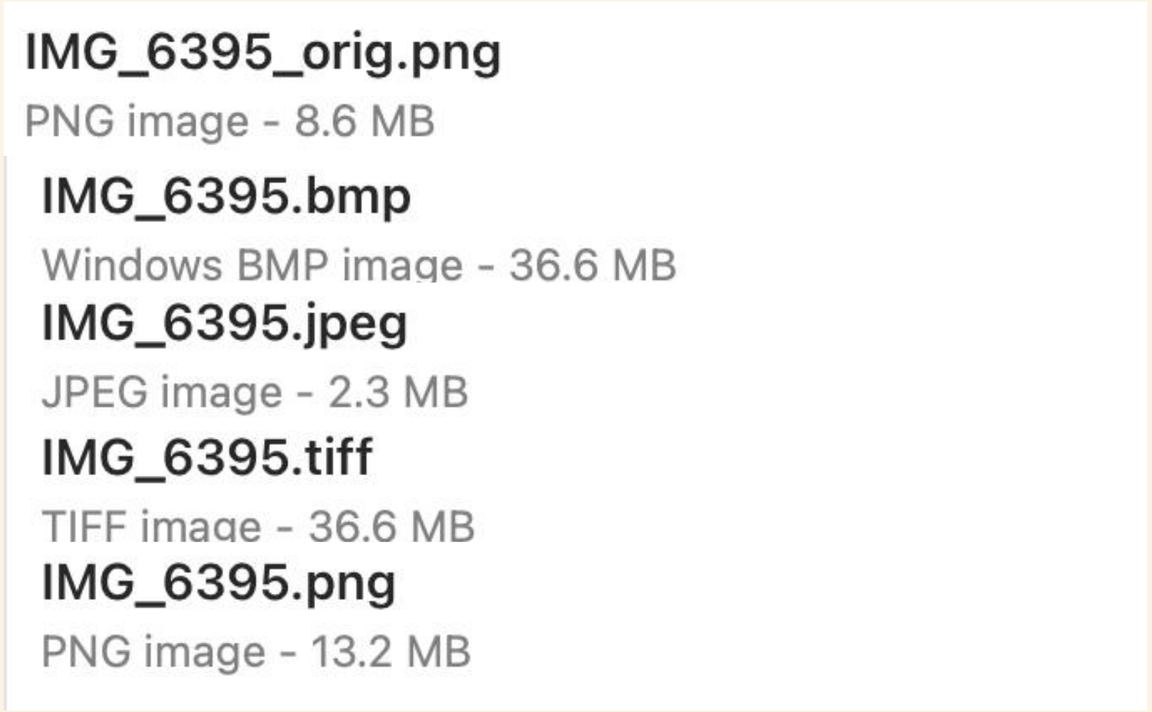


Figure 8. File sizes of different image formats with BMP and TIFF having the largest image size and JPEG format having the smallest image size.

I tried saving the exported image into different file formats: PNG, BMP, JPEG, and TIFF. From Figure 8, we can see that TIFF and BMP have the highest image file size and the JPEG has the lowest image file size.

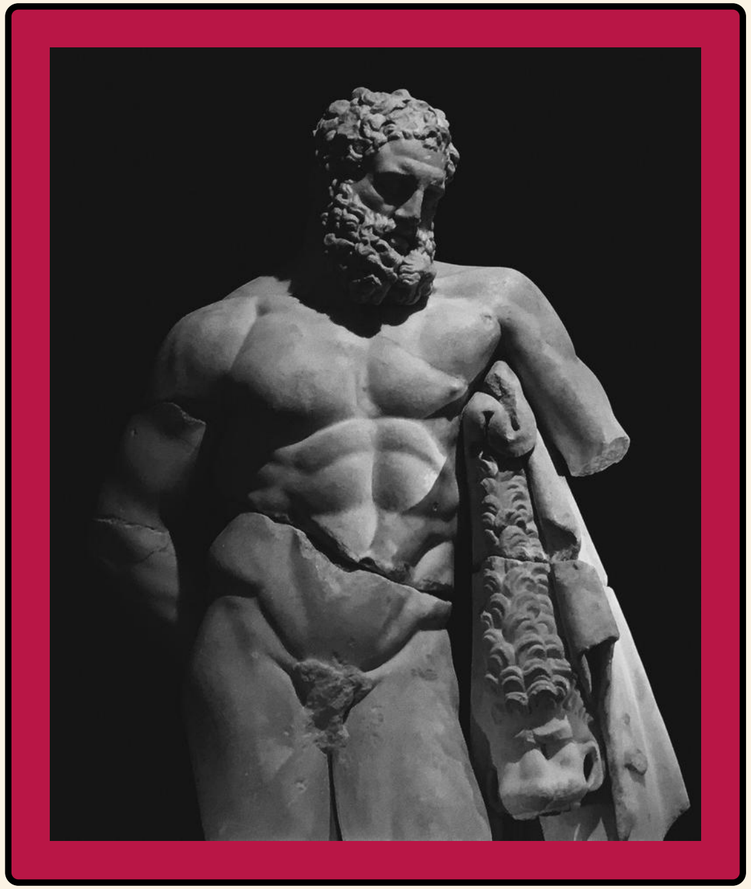
This is because TIFF and BMP incorporates lossless compression which saves the image as original as possible. Moreover, the JPEG incorporates lossy compression since it tries to reduce the file size as much as possible.

File size of images type

HISTOGRAM BACKPROJECTION

Objectives:

- Use *backprojection* technique to transform histogram of an image into desired histogram distribution



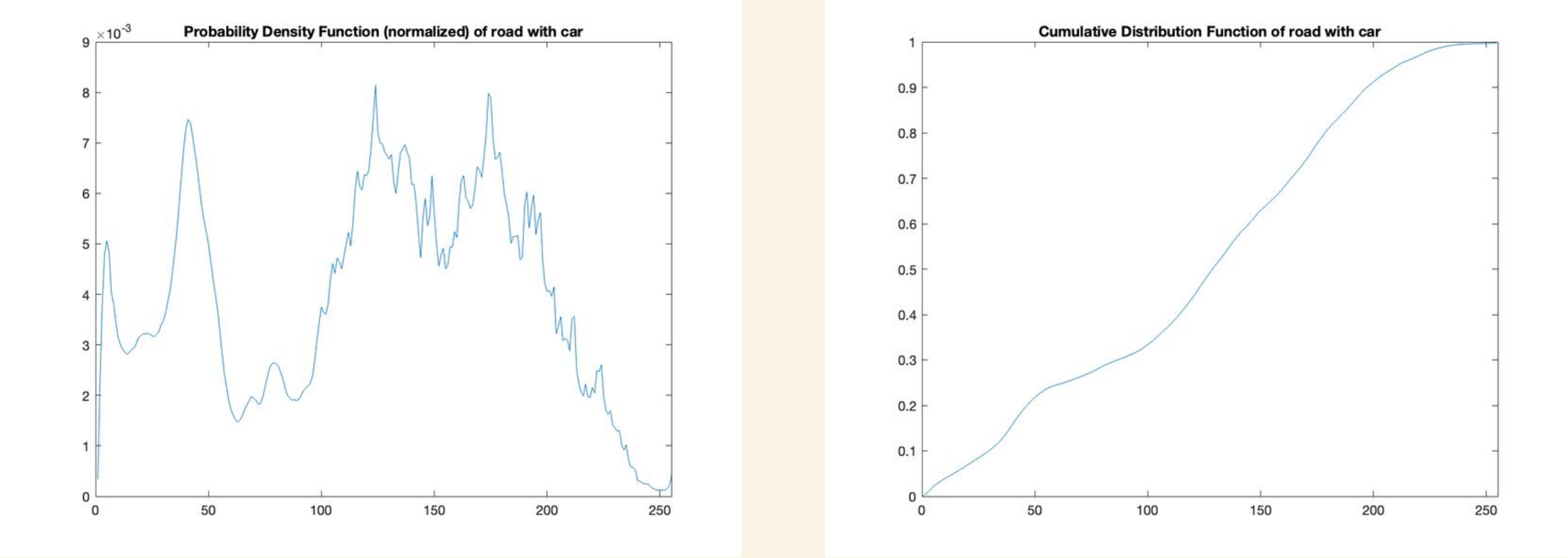


Figure 9. From left to right: (a) Probability Density Function of the original grayscale image, and (b) Cumulative Distribution Function of the original grayscale image

To do the graphs in Figure 9, I used the hist() and cumsum() functions to create the PDF and CDF respectively. Before doing so, I I used im2double() function in the image used to convert it to double precision. It rescales the output from integer data types to the range [0,1].

PDF and CDF of Original Grayscale Image

HISTOGRAM BACKPROJECTION

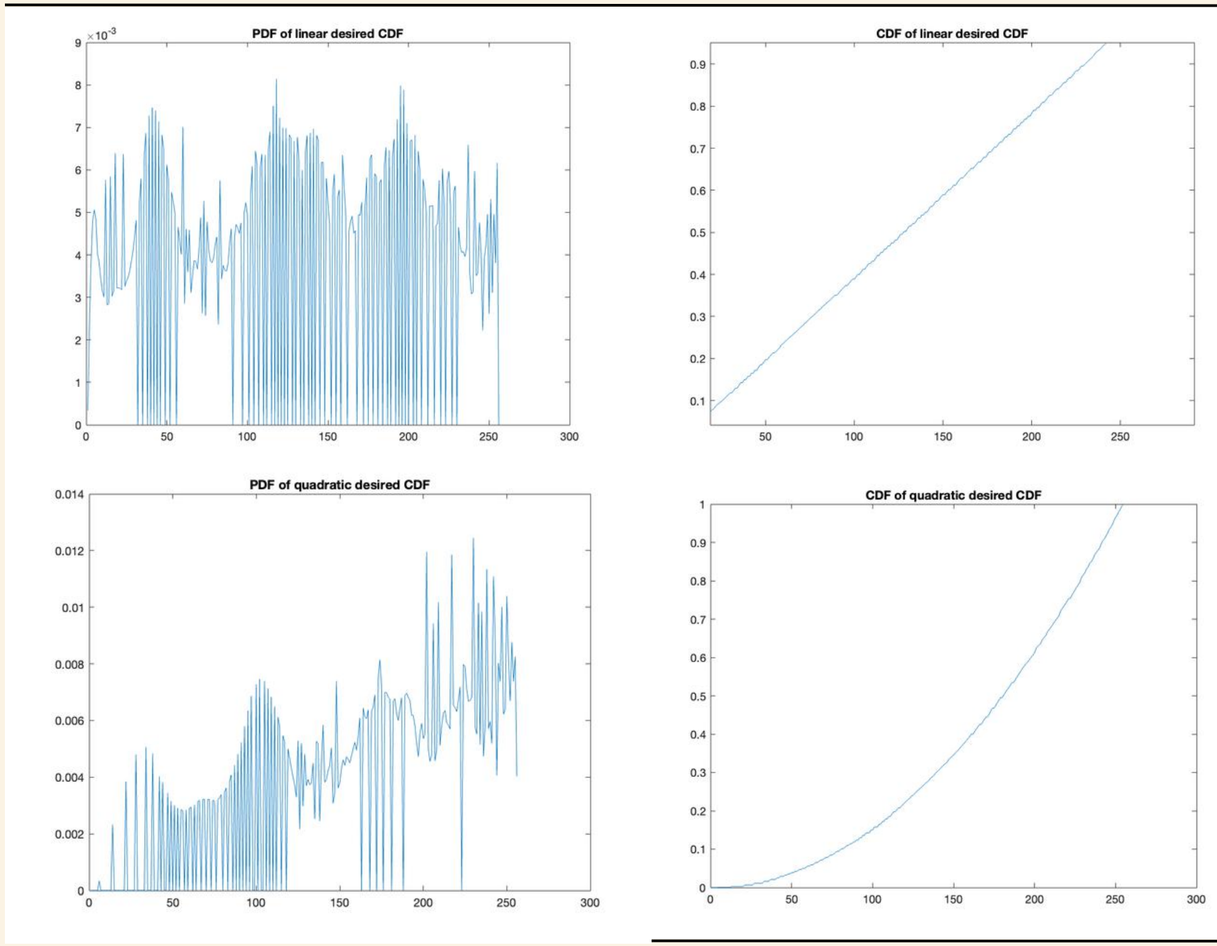


Figure 10. From left to right: (a) PDF of linear desired CDF, (b) CDF of linear desired CDF, (c) PDF of quadratic desired CDF and (d) CDF of quadratic desired CDF.

From the Cumulative Distribution Function produced, we can see that it follows the trend of the desired CDF (linear and quadratic). Though even if the noise can be seen when zoomed in, the overall trend follows the desired.

I used interpolation function interp1() for inverse mapping and rescale() function to make sure the image produced is of the same size as the original.

Aside from the following methods, I also used histogram equalization and adaptive histogram equalization functions which outputs are shown in the next page.

PDF and CDF of Altered Image

HISTOGRAM BACKPROJECTION

Figure 10. Histogram back-projection of a grayscale image. From left to right: (a) original grayscale image, (b) image with linear desired CDF, (c) image with quadratic desired CDF, (d) image formed using histogram equalization function and (e) image formed using adaptive histogram equalization function.

From the images above, we can see that the Figure 10 (b) and (d) are strikingly similar with the original image. Moreover, Figure 10(c),which uses quadratic desired CDF, have an output brighter than the rest and Figure 10(e) shows bold contrasts between the objects in the image.

Results of Histogram Backprojection

CONTRAST ENHANCEMENT

Objectives

- Use contrast stretching with percentiles of different minimum and maximum intensities

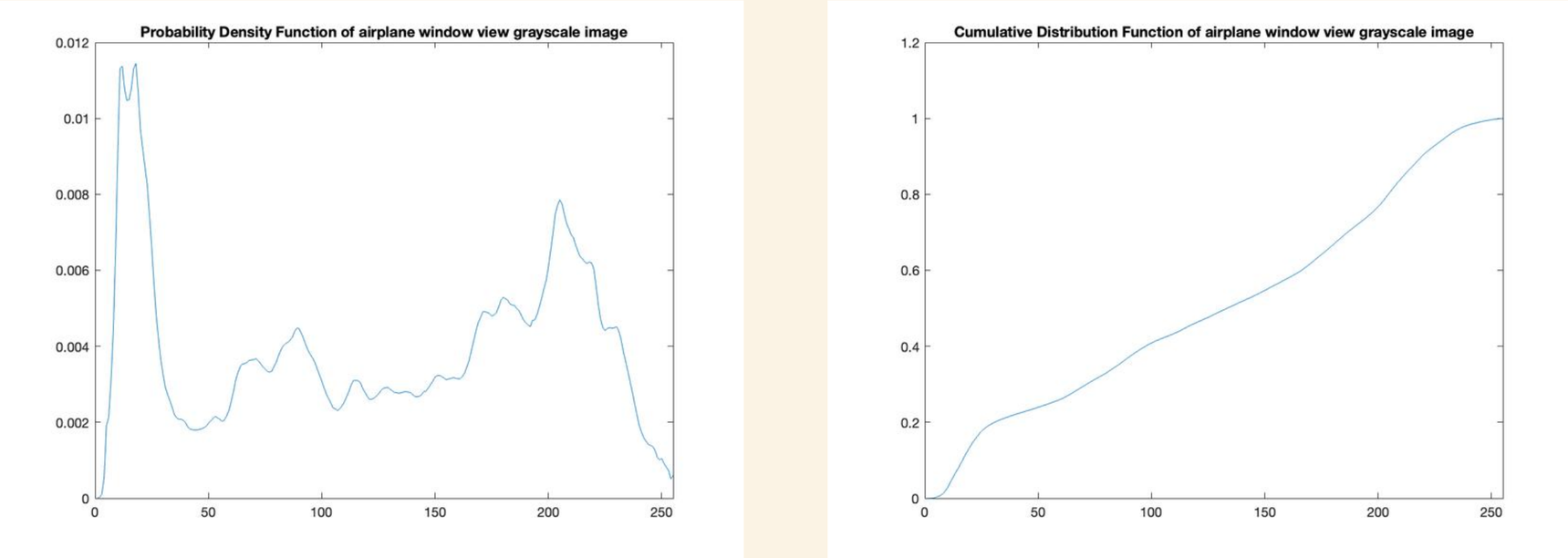


Figure 11. From left to right: (a) Probability Density Function of the original grayscale image, and (b) Cumulative Distribution Function of the original grayscale image

To do the graphs in Figure 11, same with Figure 9, I used the hist() and cumsum() functions to create the PDF and CDF respectively. Before doing so, I used im2double() function in the image used to convert it to double precision. It rescales the output from integer data types to the range [0,1].

PDF and CDF of Original Grayscale Image

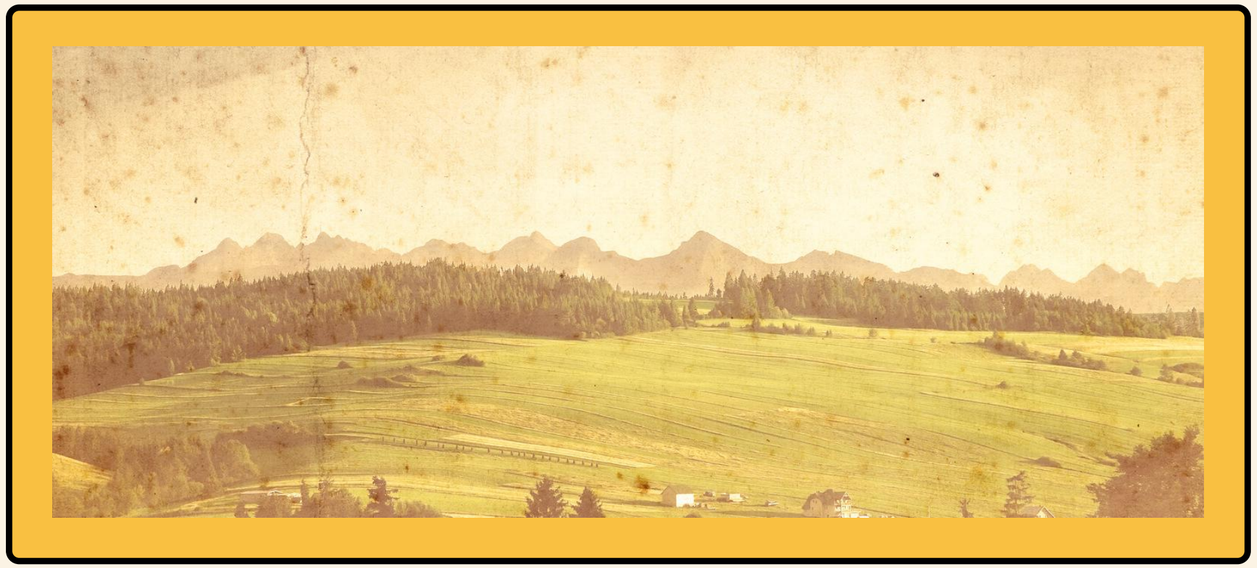
CONTRAST ENHANCEMENT

Figure 12. Contrast enhancement of a grayscale image. From left to right: (a) original grayscale image, (b) image using its minimum and maximum, (c) image using the 10th and 20th percentile as minimum and maximum respectively, (d) image using the 10th and 40th percentile as minimum and maximum respectively (e) image using the 10th and 60th percentile as minimum and maximum respectively and (f) image using the 10th and 80th percentile as minimum and maximum respectively.

To do the contrast enhancement, I used equation (2) from the Digital Image Formation and Enhancement module. I varied the values for minimum and maximum values using the percentile function. From the figure, we can see that as the maximum and minimum percentile moves away from each other, the altered image becomes more and more similar with the original grayscale image. Before solving the equation, I made sure to convert the image class into double so the range is only from 0 to 1.

Results of Contrast Enhancement

RESTORING FADED PHOTOGRAPH



Objectives:

- Use white balancing to restore faded or imperfect photographs
  - Contrast Stretching
  - Gray World Algorithm
  - White Patch Algorithm

Figure 13. RGB distribution of original image. From left to right: (a) Red (b) Green and (c) Blue

Figure 14. RGB distribution of contrast-stretched image. From left to right: (a) Red (b) Green and (c) Blue

Contrast Stretching

Figure 15. From left to right: (a) original image and (b) image formed using contrast stretching.

I first split the image into red, green, and blue values using imsplit() function. I then used equation (2) of the module to create the new values of the image. Figure 15 (b) showed the contrast-stretched version of it. One problem that arose in this activity is the manual inputting of percentile values of RGB to the new image. I was unsure of the appropriate minimum and maximum percentiles that's why I decided to experiment with it and see if what range produces better image.

Contrast Stretching

Figure 16. From left to right: (a) original image and (b) image formed using gray world algorithm.

To do the new image, I averaged the values of red, green, and blue channels of the original image. I then divided the original RGB values of the image by the average values of RGB. The result became the new values of RGB as seen in Figure 16 (b) creating a somewhat overly bright output than the original.

Figure 17. From left to right: (a) original image and (b) image formed using white patch algorithm.

I used imcrop() function to crop a white portion of the image. I then saved it and used it as my reference for my white patch. After that, I averaged the RGB values of the white pixel chosen. I used it to divide the original RGB values and create the new values for RGB. The image at the right is the output of such method.

Figure 18. Histogram back-projection of a grayscale image. From left to right: (a) original image, (b) image formed using contrast stretching, (c) image formed using gray world algorithm, (d) image formed using white patch algorithm and (e) image formed using imlocalbrighten() function.

The image above shows the comparison of the different white balancing algorithms. In the last image, I used imlocalbrighten() to show how the algorithms fared with respect to the MATLAB's built-in function. From my perspective, all algorithms were able to enhance the original image. However, I think the white patch algorithm is the most appropriate for this image in restoring it as it is not as dull nor as bright as Figure17 (b) and (c).

REFLECTION

RATING: 100 / 100

I give myself a perfect rating since I was able to satisfy the objectives of this activity. Before starting, I do not have any background knowledge regarding MATLAB since I used Python for my previous classes. Despite this weakness, I still decided to push through with creating this activity using MATLAB. My intention for this was to add another skill that will be helpful for my future career. So during the first week of classes, I enrolled myself to their Introductory courses and I was amazed by how more efficient this software was compared to Python. I was able to familiarize myself with the basics and, with the help of my classmates (Johnenn Manalang, Abdel Sinapilo, and Jarod Ordonio), the MatLab's documentation, and the module provided, I was able to accomplish the required tasks. Moreover, I was also able to add additional codes that made my output more presentable.

I actually really enjoyed the image processing part since I was able to see the results of my codes. In fact, when I was doing that part, time passed by swiftly and I realized I was doing image processing tasks up until 2 am. I would have not realized that if I didn't check the clock.

References

Help Center. (n.d). Mathworks. Retrieved March 10, 2023. https://www.mathworks.com/help/matlab/ref/im2double.html

Image file type and format guide. (n.d). MDN web docs. Retrieved March 10, 2023. https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types

Soriano, M. (2023). *Digital Image Formation and Enhancement*. UVLe. https://uvle.upd.edu.ph/course/view.php?id=11940