Charmaine S. Tolledo's google chrome

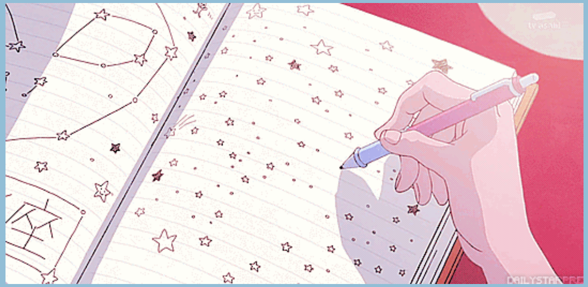
DIGITAL IMAGE FORMATION AND ENHANCEMENT

SEARCH

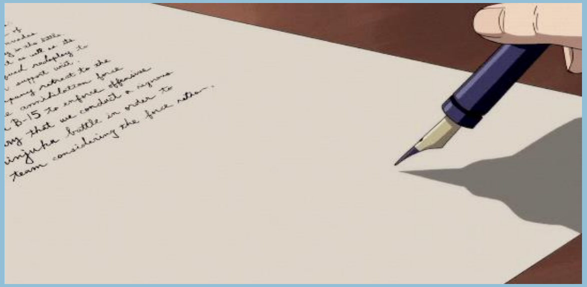
12:00 AM
03/11/2023

DIGITAL IMAGE FORMATION AND ENHANCEMENT

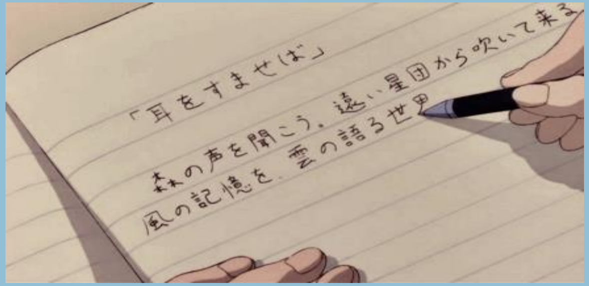
ACTIVITY 1.1



ACTIVITY 1.2

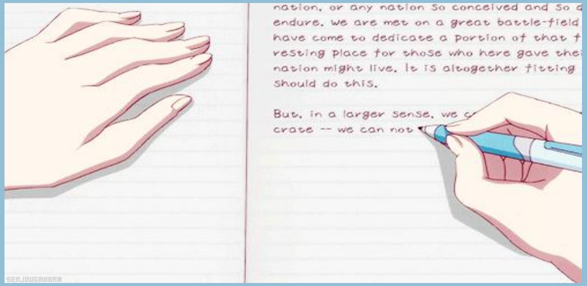


ACTIVITY 1.3



ACTIVITY 1.4

ACTIVITY 1.5



ACTIVITY 1.6

10:50 AM
04/03/2021

OBJECTIVES!

1 Mathematically create images.

2 Save an image in an appropriate file format.

3 Open and capture images using Python or Matlab.

4 Improve the appearance of gray level and color images.

5 Use the back projection technique to transform the histogram of an image to a desired distribution

12:00 AM
03/11/2023

Activity 1.1

Image DIY

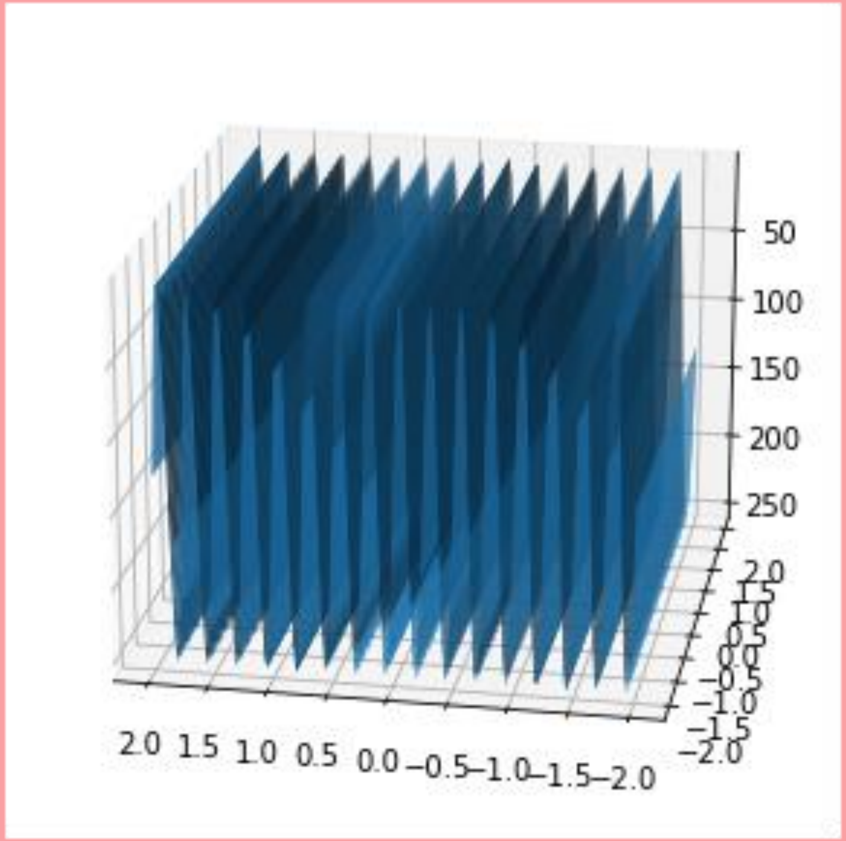
Created synthetic images of:

- ✓ Sinusoid along x-direction
- ✓ Grating of Sinusoid along x-direction
- ✓ Hubble Primary Mirror
- ✓ JWST Primary Mirror

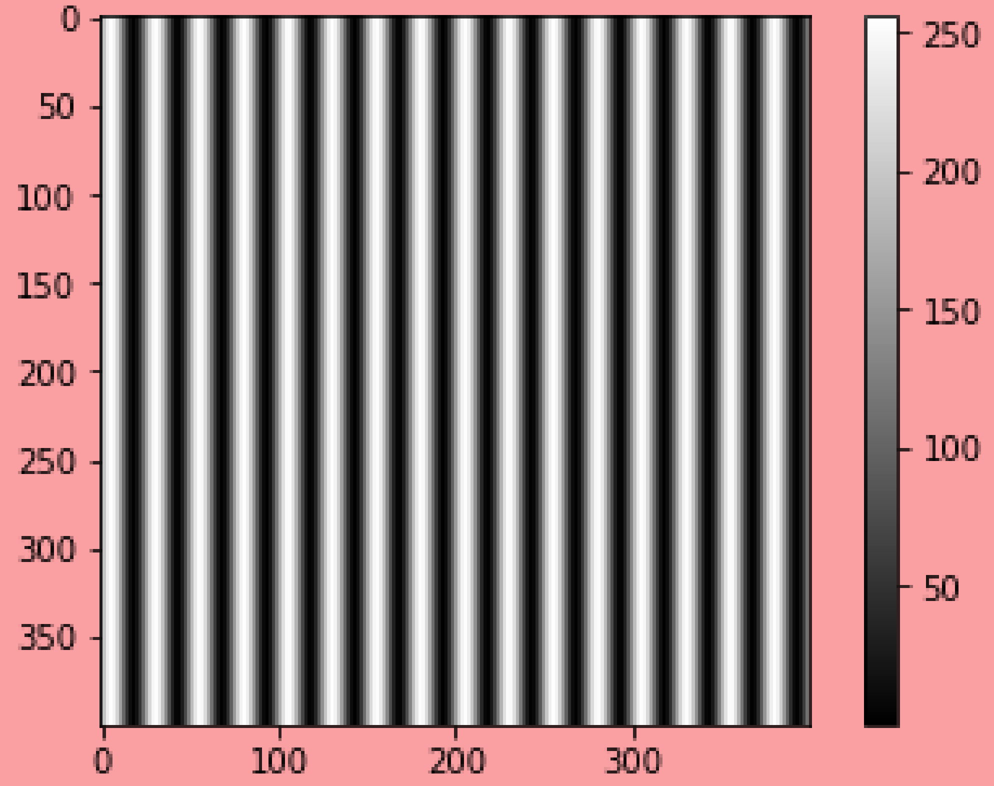
12:00 AM
03/11/2023

Sinusoid along x-direction

Using the Jupyter notebook, I made a 3D surface plot of a sinusoidal pattern along the x-axis with a frequency of 4 cycles per centimeter. In my code, I ensured that the values in my range were scaled by a factor of 256/2 such that they are between 0 and 256 and do not return negative values. The image was produced using the 'plot surface function,' which offers a comprehensive representation of a pattern in three-dimensional space. This synthetic image of a sinusoidal pattern along the x-axis is crucial in a variety of applications, such as image processing.



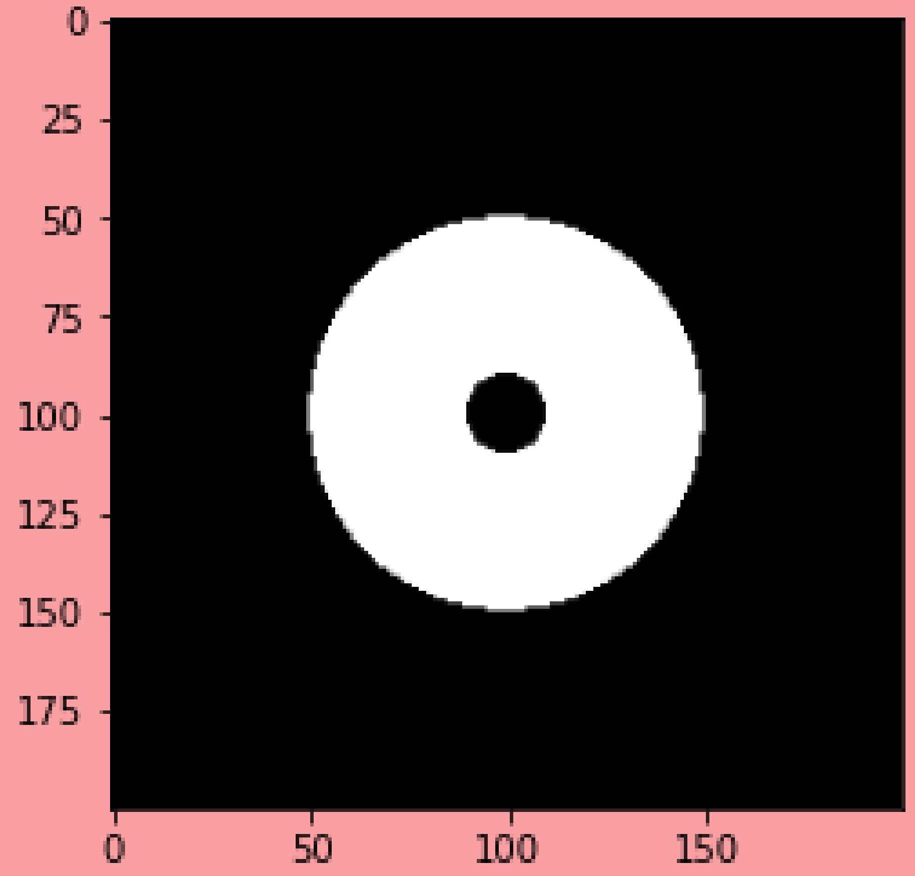
Sinusoidal Grating



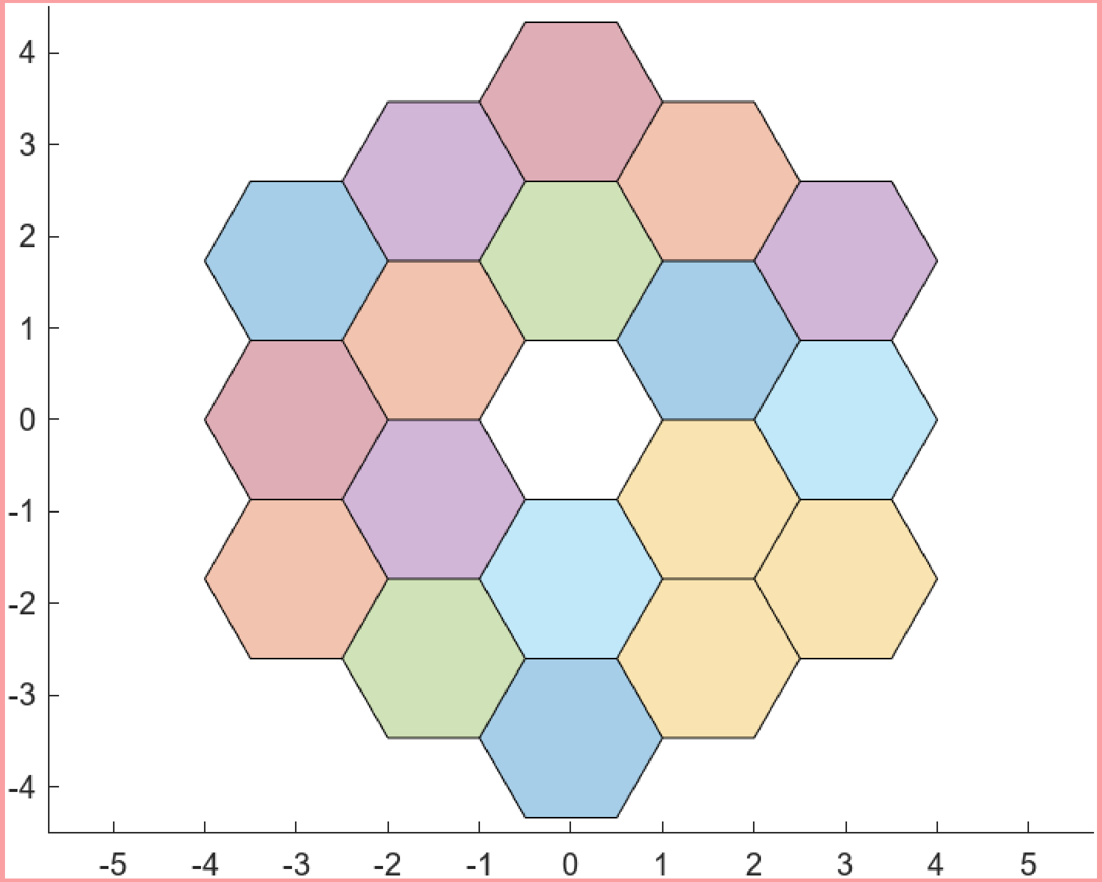
I replicated the sinusoidal pattern in both the horizontal and vertical directions using the following code: A = np.zeros(np.shape(R)), where A is an empty array with the same form as R. Also, I increased the frequency of the sinusoidal pattern by one to guarantee that the minimum value of the sinusoidal pattern matches to the background intensity of the grating, since we require a strip of black (0) and white (1). Hence, the produced grating has a frequency of 5 line pairs per centimeter (cm). This indicates that every 1 cm of the grating contains five line pairs consisting of alternating black and white stripes.

Hubble Primary Mirror

This is a synthetic image of the primary mirror of the Hubble space telescope, which is a sort of telescope mirror. I used the same code as in the lab manual, which formed a circle with a radius of 0.5 unit and whose points are set to 1.0, but I added another array A in which the points within this circle with a radius of 0.1 unit are set to 0.0. This additional array constituted the smaller circle. This produces a pattern consisting of a circular area bordered by a dark annulus. Similarly to the sinusoidal pattern, this is also incredibly beneficial for image processing algorithms.



JWST Primary Mirror



Finally, the James Webb Space Telescope has a synthetic main mirror (JWST). This is the most challenging to create of the four synthetic pictures. In contrast to the other three, I used Matlab here. I built 18 hexagonal segments using the MATLAB nsidepoly function. Then, I computed the center's specific coordinates to guarantee that they are in the correct location, and I included the axis equal command to ensure that the x and y axes are scaled equally. ((I really struggled making this, so a big shout out to my friend Mar Princer, who taught me which MATLAB function to use and how to compute the center of the hexagon))

ACTIVITY 1.2

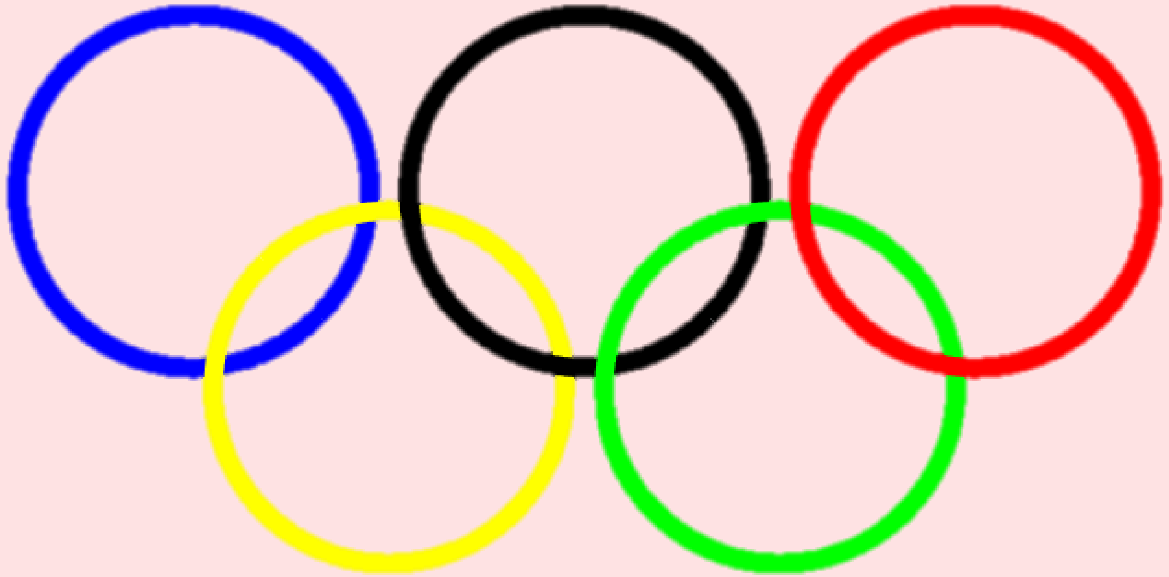
COLOR IMAGE

- ✓ Olympics Logo
- ✓ Basic Types of Image

12:00 AM
03/11/2023

OLYMPIC LOGO

This Olympic logo is a MATLAB-created color image. It is comprised of five overlapping rings of various hues, including blue, yellow, black, green, and red. In my code , I defined the x and y coordinates of each ring using the cos and sin functions, with a radius of 0.9, then shifted these points horizontally and vertically to the correct positions for each ring. This creation of the color image explains why it is so vital to many aspects of our everyday lives, including photography and scientific study. My code was inspired by the code I saw online, which can be accessed at https://blogs.mathworks.com/loren/2008/10/31/olympic-rings/. (To add, I first created a code that was comparable to the code in the lab manual, but I kept getting an error, particularly EError in images.internal.imageDisplayValidateParams so I just deleted the entire code.)



BINARY IMAGE

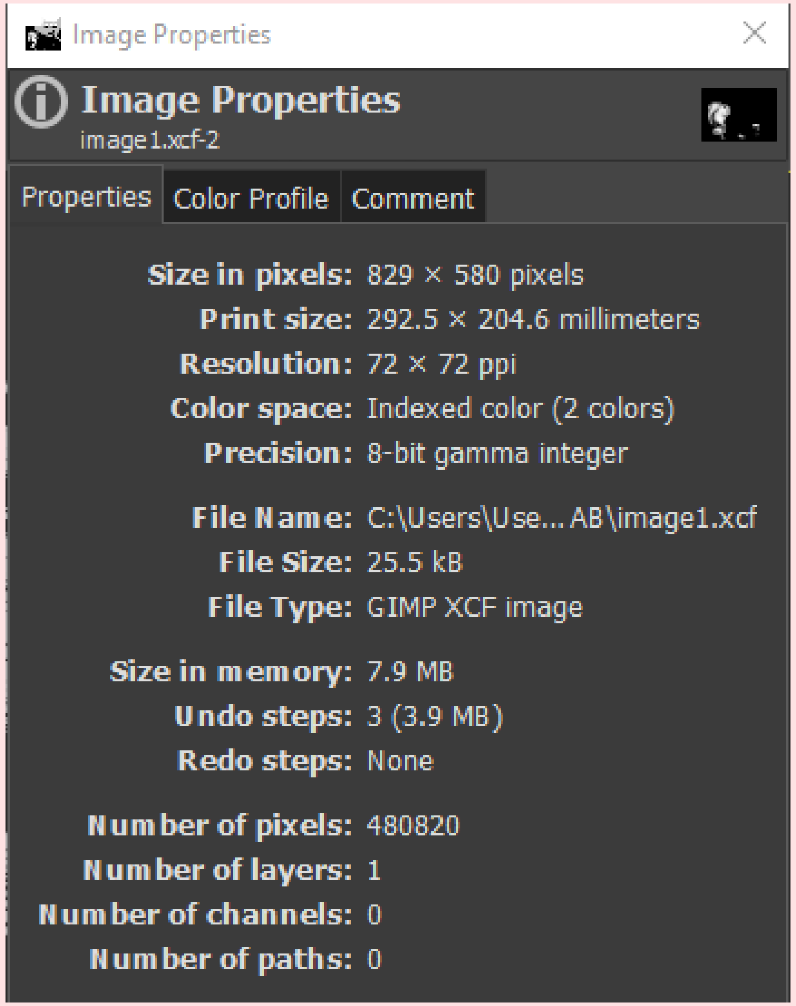


Image Properties

Image Properties
image1.xcf-2

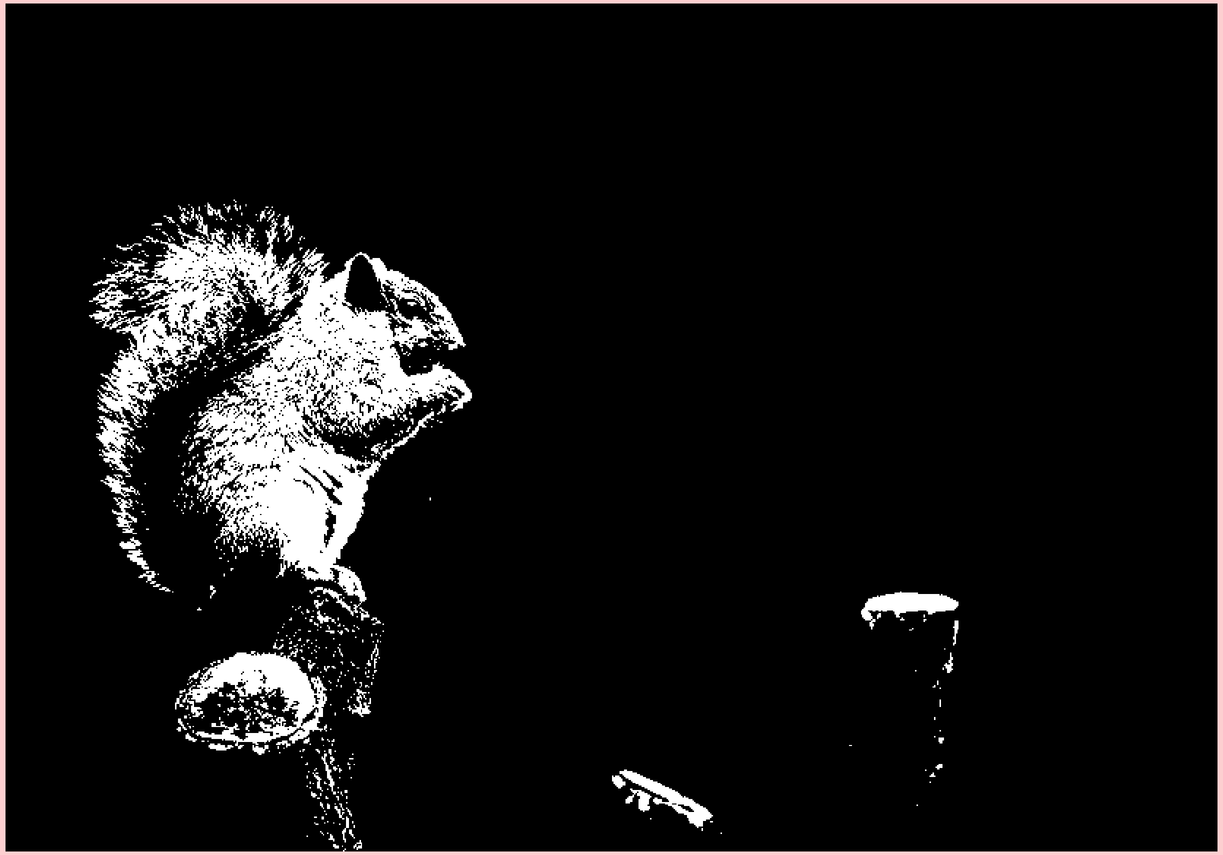
Properties | Color Profile | Comment

```
      Size in pixels: 829 × 580 pixels
          Print size: 292.5 × 204.6 millimeters
          Resolution: 72 × 72 ppi
         Color space: Indexed color (2 colors)
           Precision: 8-bit gamma integer

           File Name: C:\Users\Use... AB\image1.xcf
           File Size: 25.5 kB
           File Type: GIMP XCF image

      Size in memory: 7.9 MB
          Undo steps: 3 (3.9 MB)
          Redo steps: None

    Number of pixels: 480820
    Number of layers: 1
  Number of channels: 0
     Number of paths: 0
```


Source: https://images.unsplash.com/photo-1533651180995-3b8dcd33e834?ixlib=rb 4.0.3&ixid=MnwxMjA3fDB8MHxwaG9oby1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=870&q=80

The most basic image type which probably always mistaken as a grayscale image (which we will see later on) is the binary image. There are just two potential values for each pixel in this image: black and white, nothing in between. I turned this colored image I got from the internet into binary image using GIMP.

GRAYSCALE IMAGE

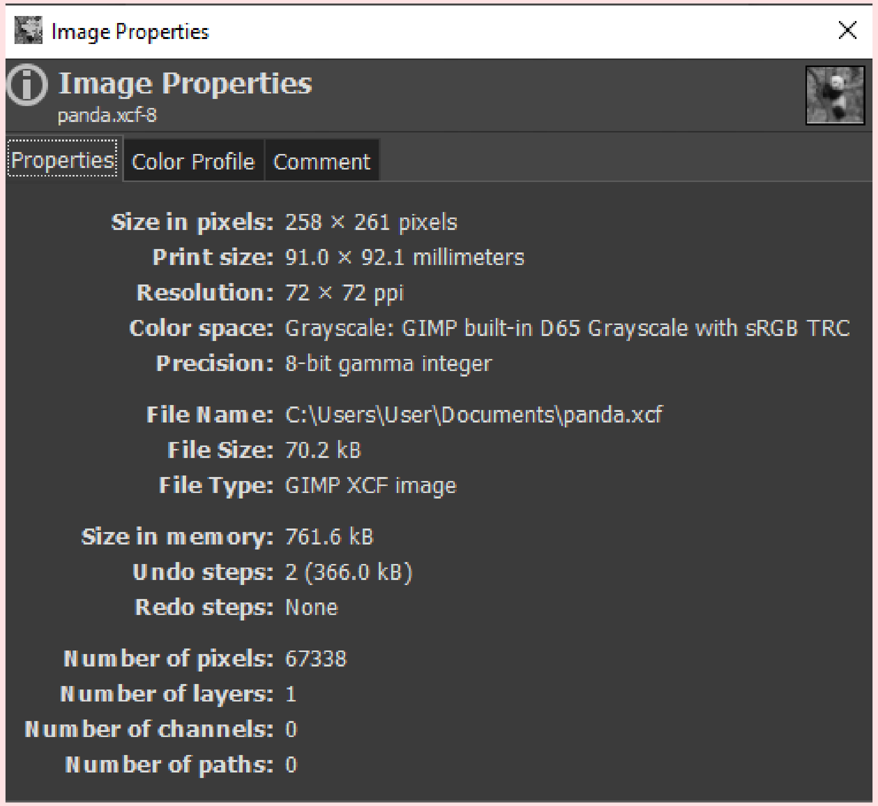


Image Properties

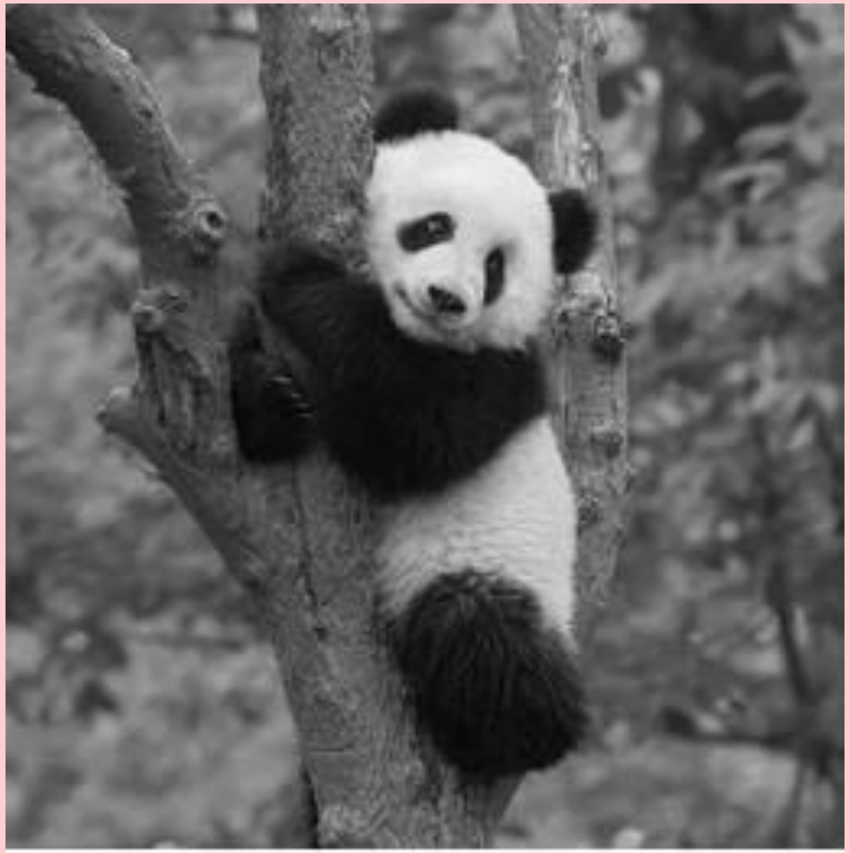
Image Properties
panda.xcf-8

```
    Size in pixels: 258 × 261 pixels
        Print size: 91.0 × 92.1 millimeters
        Resolution: 72 × 72 ppi
       Color space: Grayscale: GIMP built-in D65 Grayscale with sRGB TRC
         Precision: 8-bit gamma integer

         File Name: C:\Users\User\Documents\panda.xcf
         File Size: 70.2 kB
         File Type: GIMP XCF image

    Size in memory: 761.6 kB
        Undo steps: 2 (366.0 kB)
        Redo steps: None

  Number of pixels: 67338
  Number of layers: 1
Number of channels: 0
   Number of paths: 0
```


Source: https://www.shutterstock.com/image-photo/cute-panda-tree-201006434

The picture we referred to as Black and White is actually a grayscale image, not a binary one. This image, unlike the previous, is not absolutely black and white but may also be gray; in fact, its pixels contain values between 0 and 255. Also, comparing the property, its file size is smaller than that of true color and indexed image (which are in the next slides.) thus, this can be an advantage in situations where storage or transmission bandwidth is restricted.

TRUECOLOR IMAGE

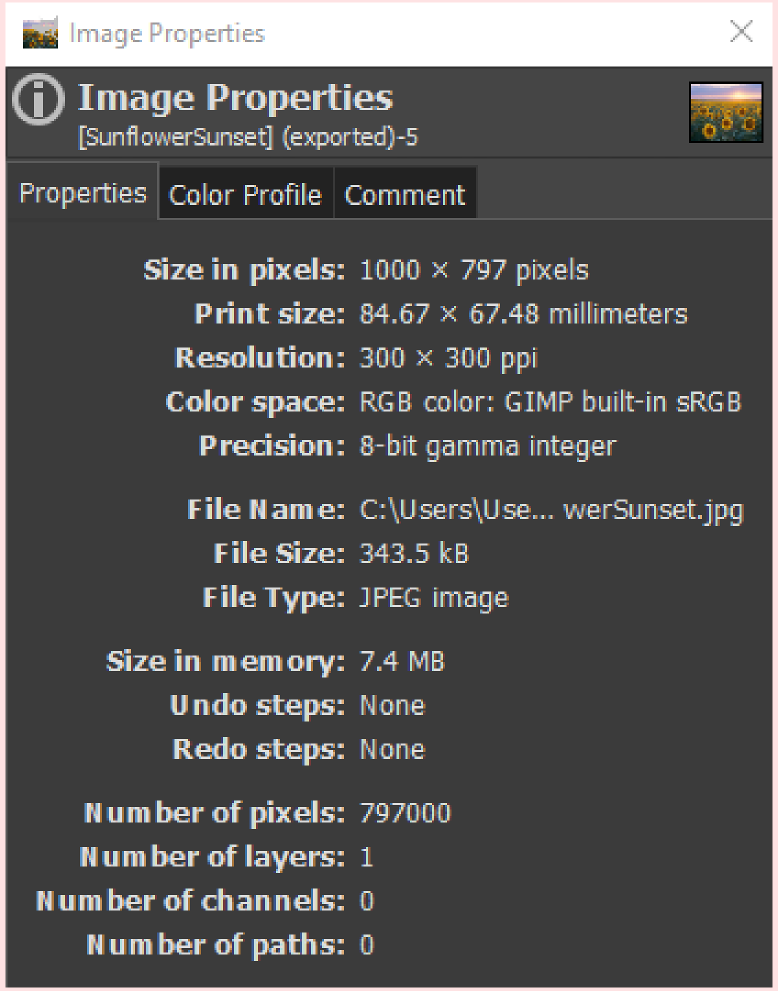


Image Properties

Image Properties
[SunflowerSunset] (exported)-5

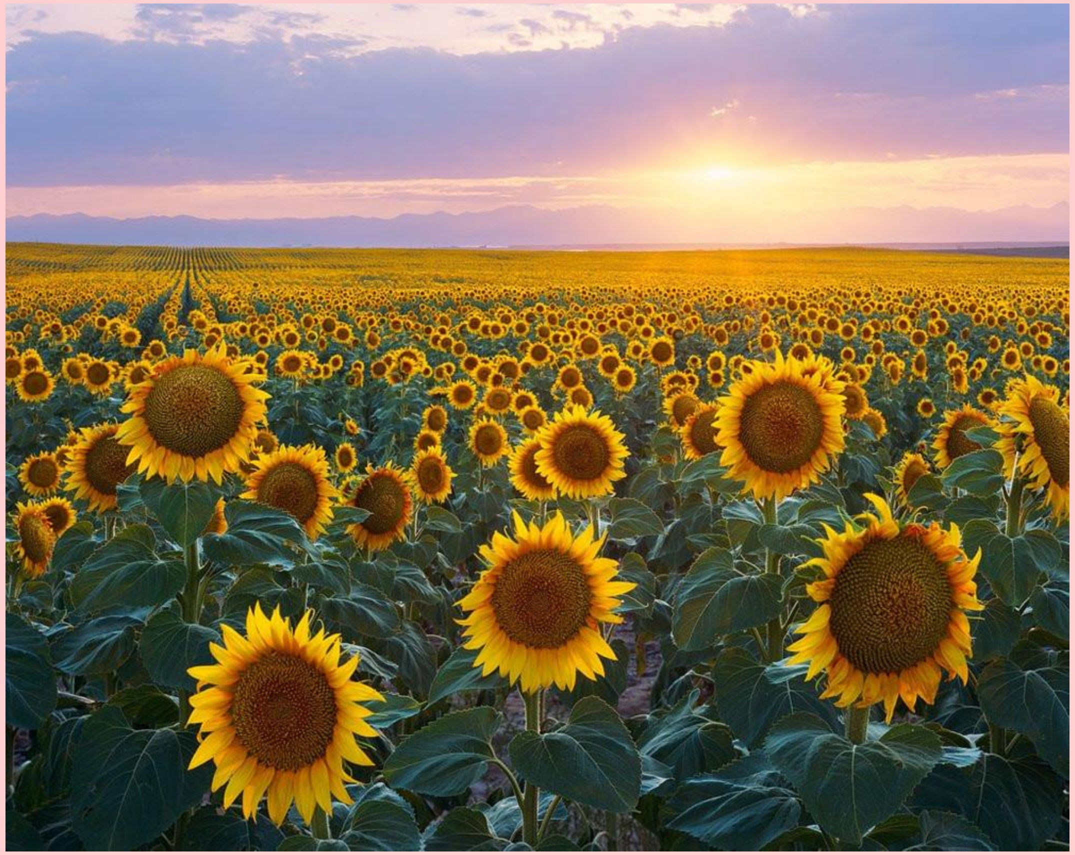
Properties | Color Profile | Comment

```
    Size in pixels: 1000 × 797 pixels
        Print size: 84.67 × 67.48 millimeters
        Resolution: 300 × 300 ppi
       Color space: RGB color: GIMP built-in sRGB
         Precision: 8-bit gamma integer

         File Name: C:\Users\Use... werSunset.jpg
         File Size: 343.5 kB
         File Type: JPEG image

    Size in memory: 7.4 MB
        Undo steps: None
        Redo steps: None

  Number of pixels: 797000
  Number of layers: 1
Number of channels: 0
   Number of paths: 0
```



Source: https://images.squarespace-cdn.com/content/v1/56bf55504c2f85a60a9b9fe5/1635897793784-OD3181KEQJ2AV5QTEEK6/SunflowerSunset.jpg?format=750w

Truecolor images, often known as RGB images, are the most prevalent type of image used today. Nearly all images we encounter nowadays, whether on social media sites or on the street, are likely of this sort. In accordance with its name, each pixel is made up of a combination of the three main colors: red, green, and blue. Combining these three will result in a vast variety of hues and shades.

INDEXED IMAGE

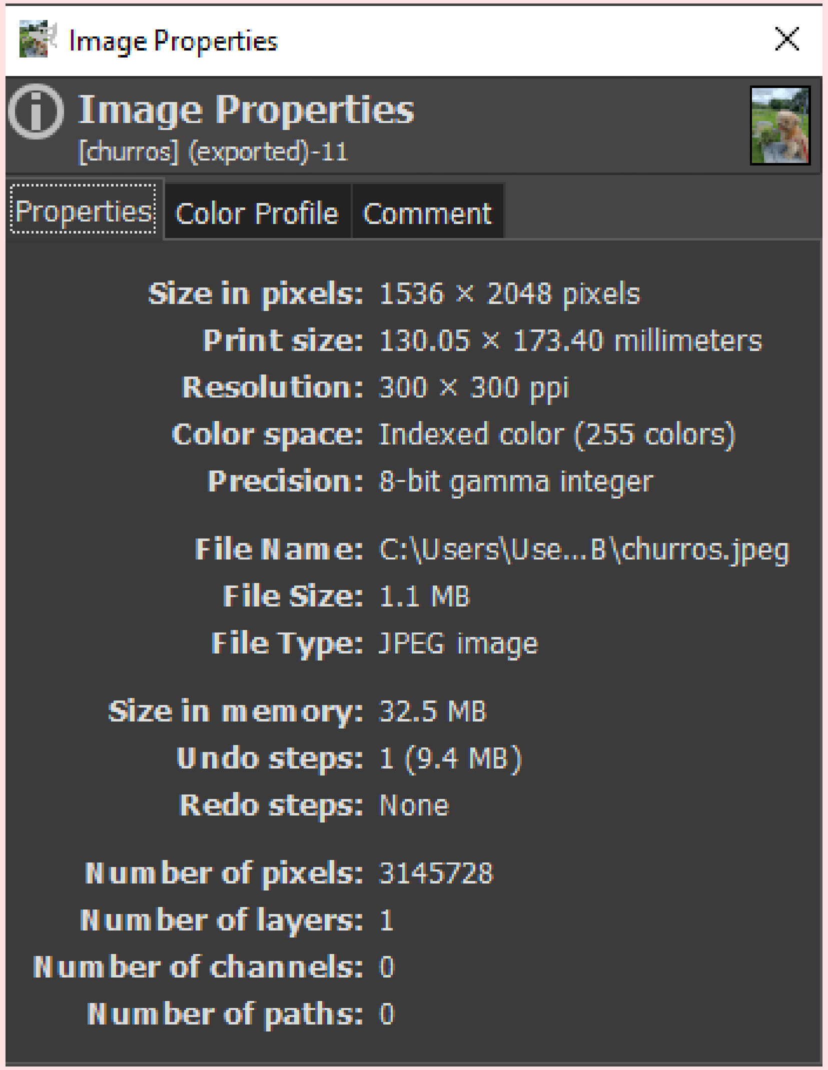


Image Properties

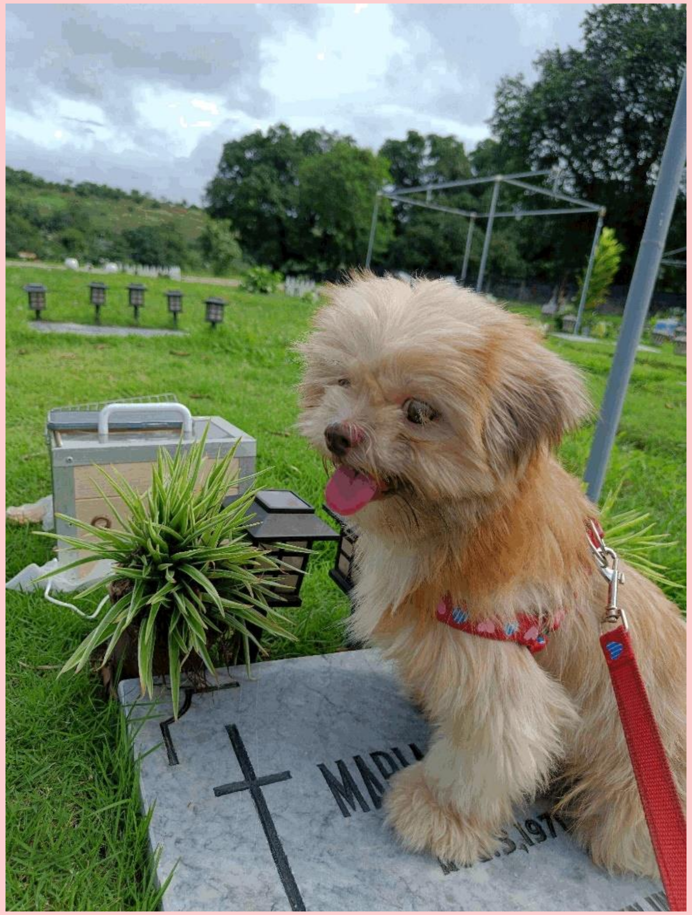
Image Properties
[churros] (exported)-11

```
     Size in pixels: 1536 × 2048 pixels
         Print size: 130.05 × 173.40 millimeters
         Resolution: 300 × 300 ppi
        Color space: Indexed color (255 colors)
          Precision: 8-bit gamma integer

          File Name: C:\Users\Use...B\churros.jpeg
          File Size: 1.1 MB
          File Type: JPEG image

     Size in memory: 32.5 MB
         Undo steps: 1 (9.4 MB)
         Redo steps: None

   Number of pixels: 3145728
   Number of layers: 1
 Number of channels: 0
    Number of paths: 0
```



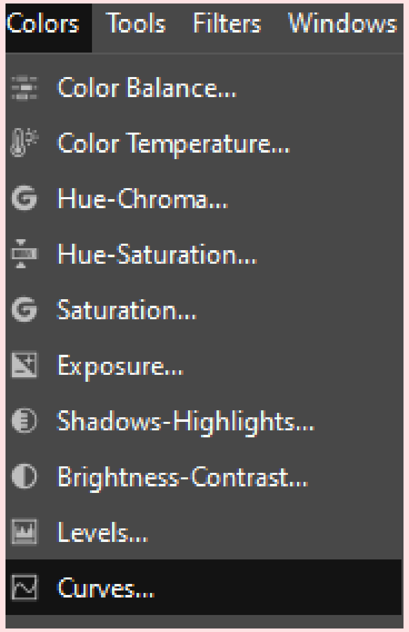
The last basic image type is the Indexed image. Instead of keeping the whole color information for each pixel, it assigns a unique index value to each color in the image using a color lookup table, as its name suggests. While it seems almost identical to a truecolor image to the human eye, they encode and store color information differently. Compared to truecolor photos, indexed images can be more space-efficient, making them useful for certain purposes.

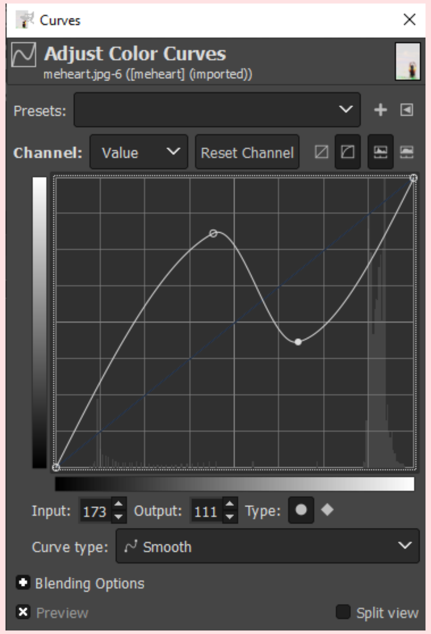
Activity 1.3

Altering the Input-Output Curve

12:00 AM
03/11/2023

Here, I manipulated histogram using the free picture editing software GIMP. In summary, based on the images below, I navigated to the "Colors" section and selected "Curves" to open the Curves dialog box (see second image), where I altered the straight line to establish a new control point. Here, I constructed an S-curve to alter the contrast of my image, and by sliding the point close to the center of the curve, I modified its brightness, thus giving me a brighter and clearer image.




Curves

Adjust Color Curves

meheart.jpg-6 ([meheart] (imported))

Presets:

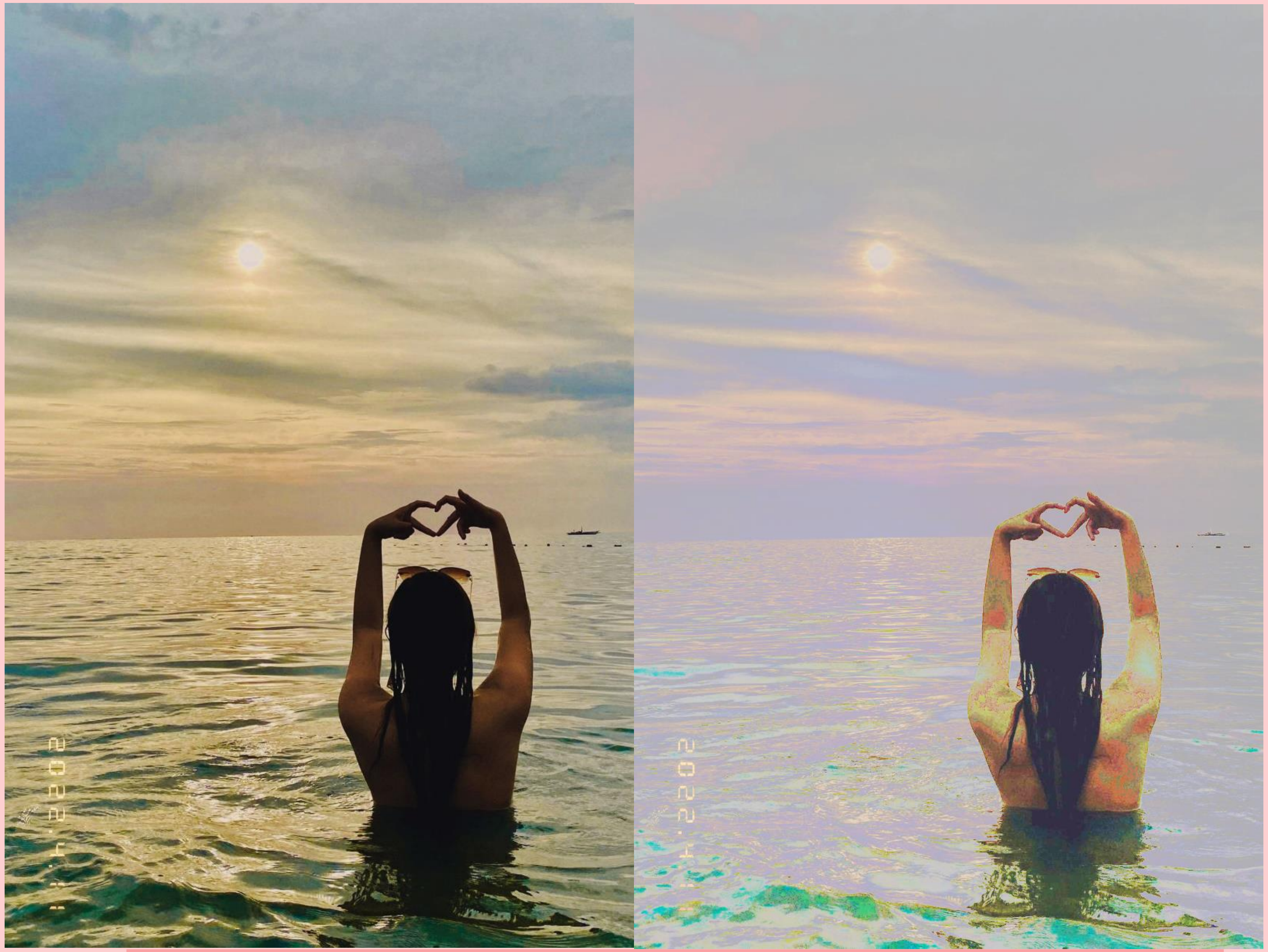
Channel: Value Reset Channel

Input: 173 Output: 111 Type:

Curve type: Smooth

Blending Options

Preview Split view

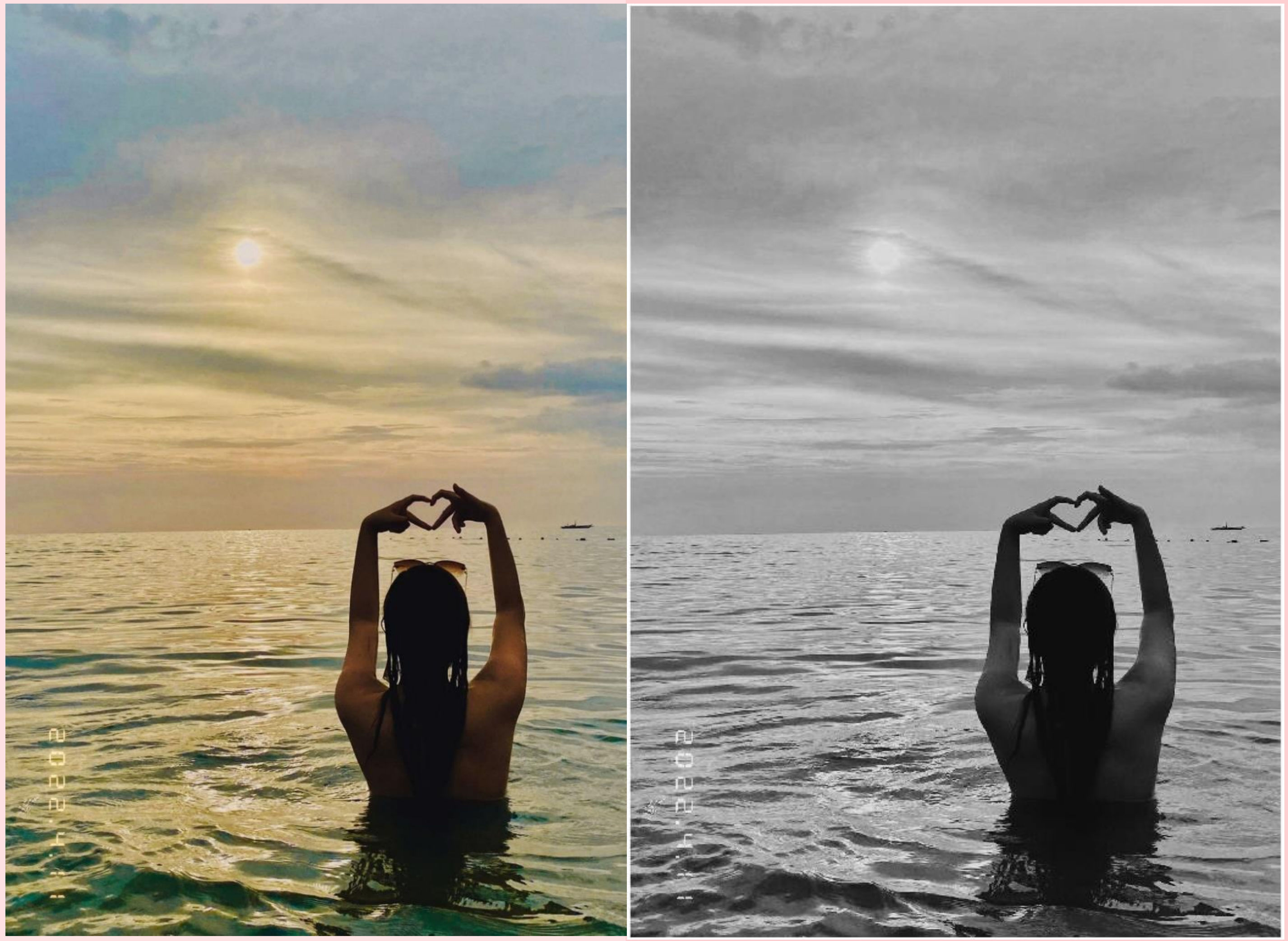

(a) Original Image (b) Enhanced image using GIMP

Activity 1.4

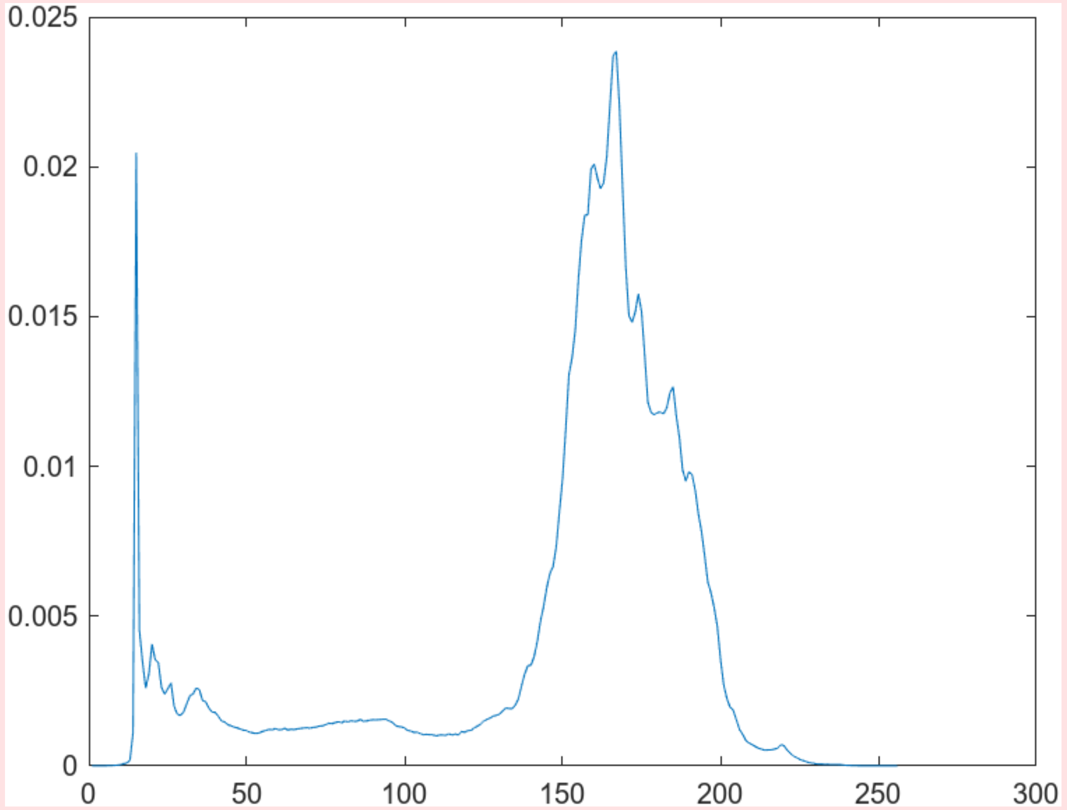
Histogram Backprojection on Grayscale Images

- ✓ Plotted the normalized PDF and CDF of the grayscale image
- ✓ Plotted the Linear, Quadratic, and Sigmoid shaped CDF
- ✓ Computed and plotted the desired CDF of the three shapes.
- ✓ Compared the original grayscale image to the enhanced images using the Linear, Quadratic, and Sigmoid CDF.

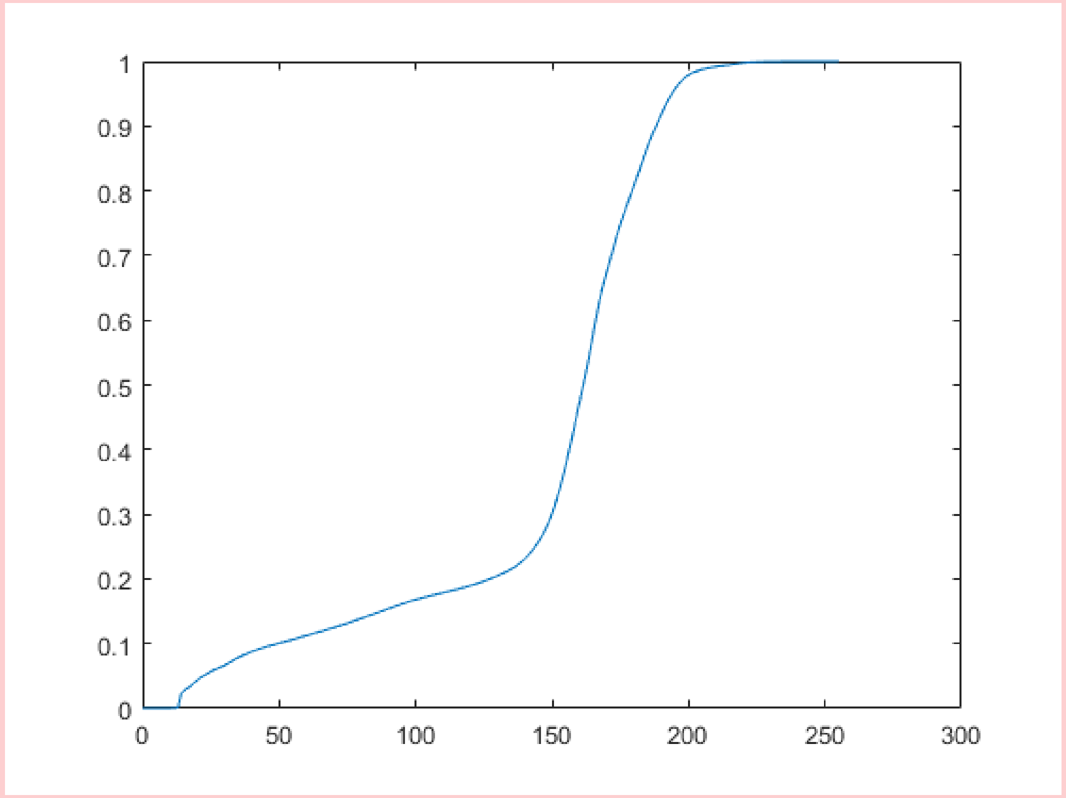
12:00 AM
03/11/2023


(a) Original Image (b) Grayscale image

The initial step required for this Activity was to transform the original image into a grayscale image, which I accomplished using MATLAB. I converted the RGB image to a grayscale image using the built-in function rgb2gray. Why then is conversion necessary? Because here we need the PDF of this picture, we transformed it to produce a single PDF that depicts the grayscale intensity distribution in the image, which is required for the subsequent slides.

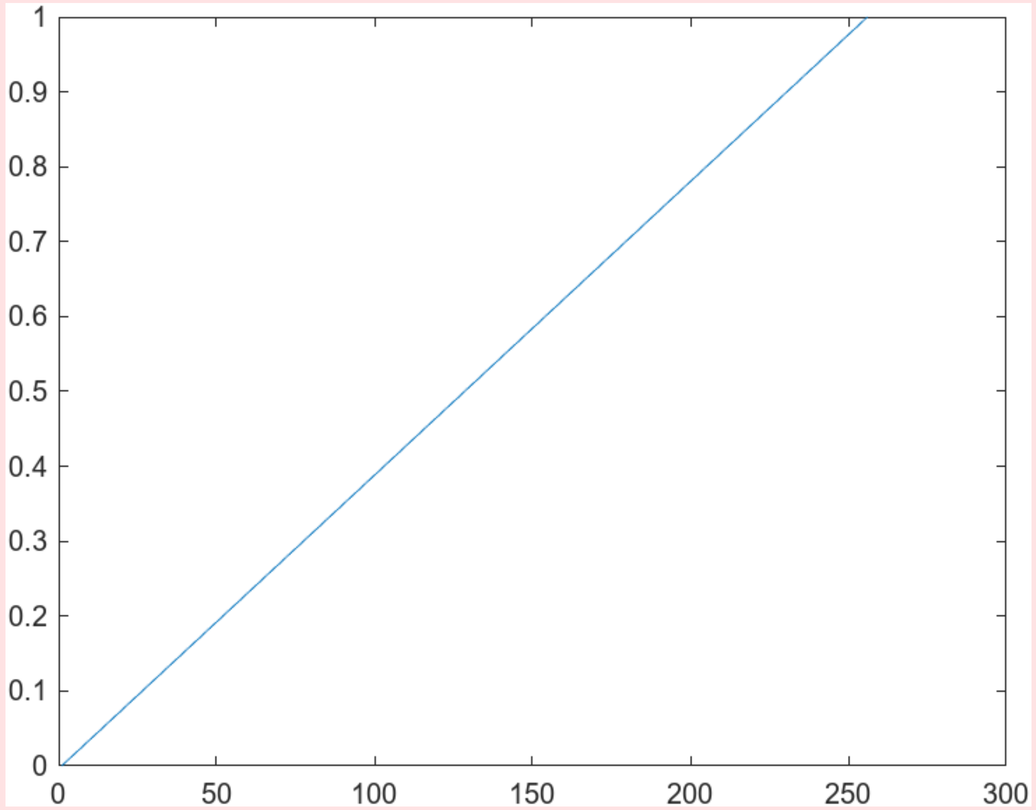


(a) PDF (normalized histogram) of the grayscale image

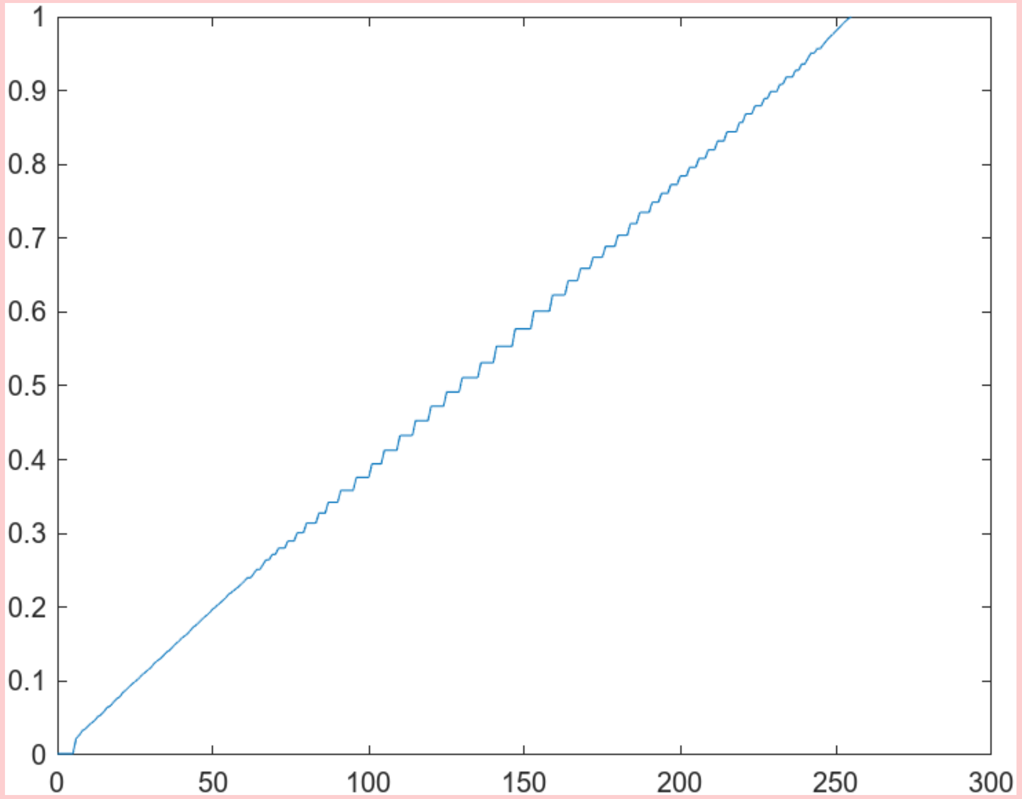


(b) Normalized CDF of the grayscale image

Here, I took the normalized PDF and normalized CDF of the grayscale image earlier. We see in the PDF that the peak falls between the values of 150 to 200, indicating that a considerable portion of the picture information falls within this range of grayscale intensities. In the CDF, it is evident that the values increased from 150 to 200, which corresponds to the PDF's peak.

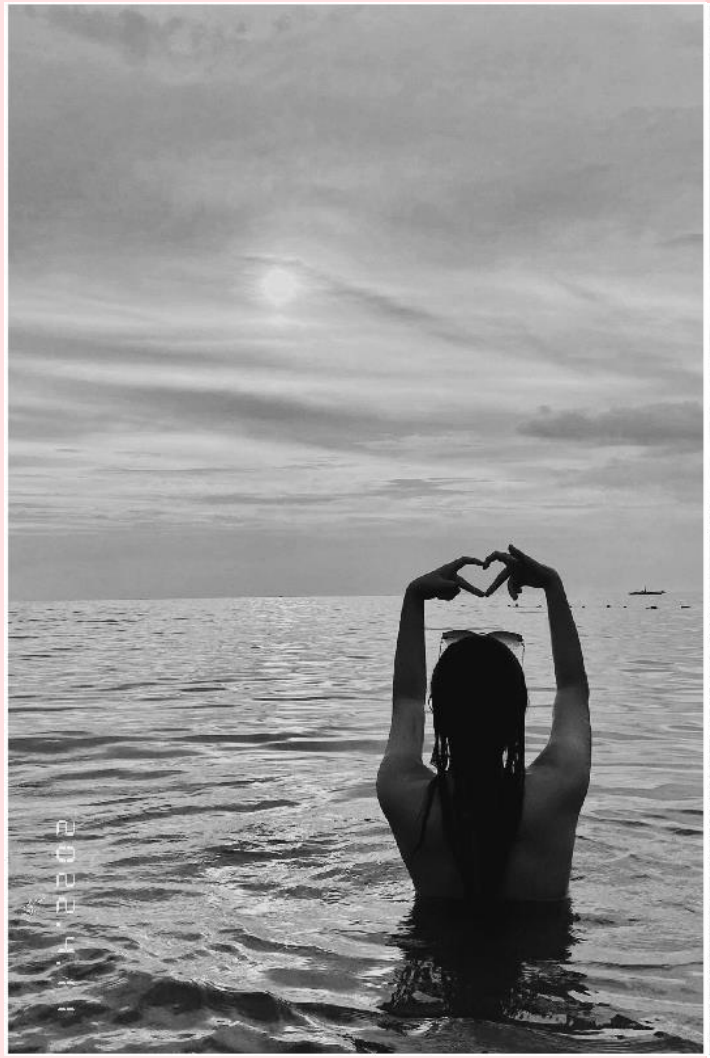


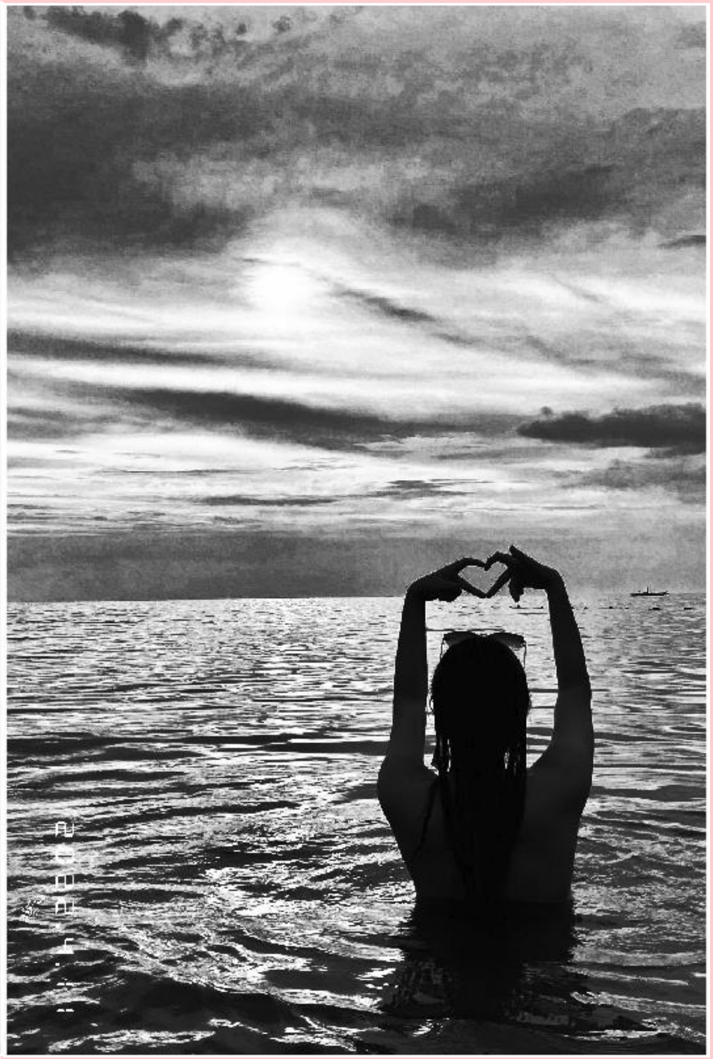
(a) Linear CDF



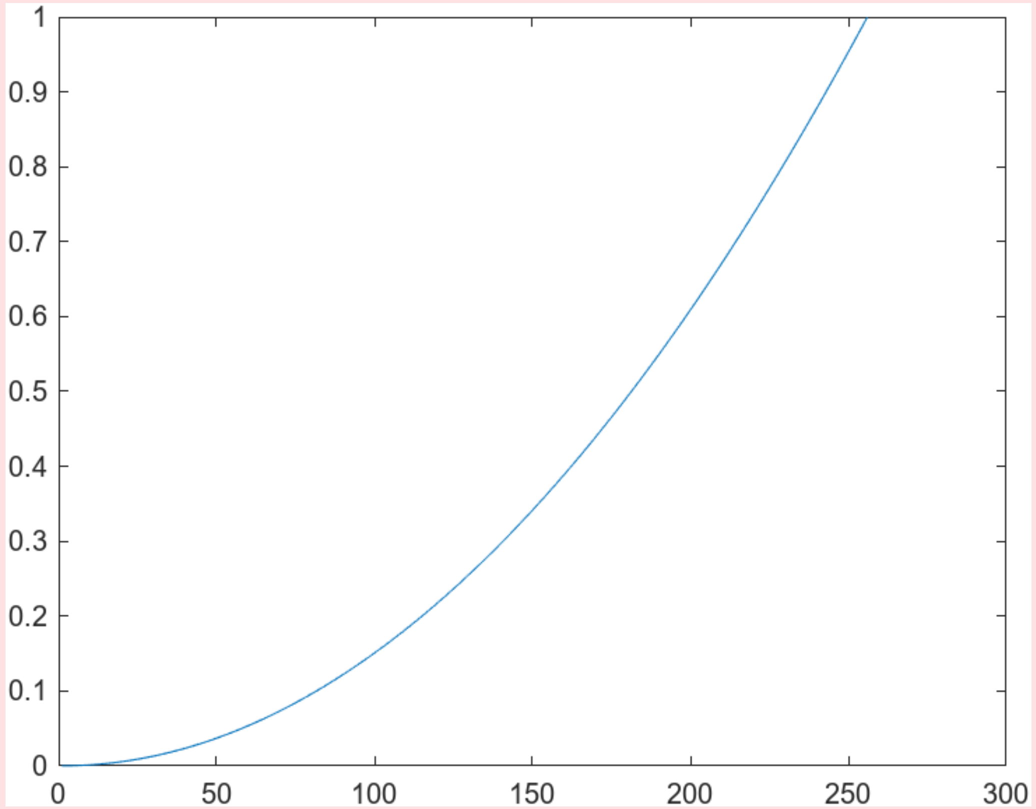
(b) Desired Linear CDF

So, we have constructed three distinct CDF shapes, one of which is the Linear CDF. The grayscale picture is contrast-stretched using a linear CDF, yielding an enhanced/back projected image (next slide), which was then used to compute and plot the desired linear CDF, as shown above.

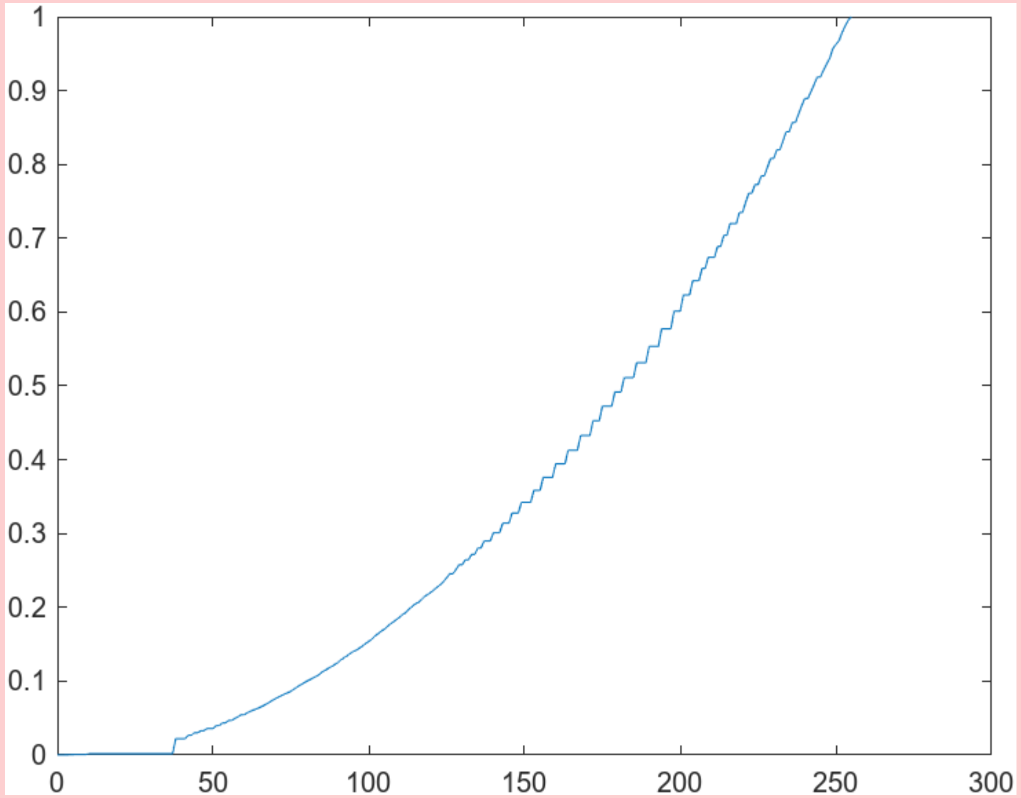

(a) Original grayscale image


(b) Enhanced grayscale image using Linear CDF

Upon comparing the two images, it is evident that the original grayscale image has low contrast and poor detail clarity. Meanwhile, the grayscale image enhanced using Linear CDF provided us with a more distinct and detailed view of the original image due to its improved contrast and brightness. Personally, I didn't liked this improved image over the enhanced grayscale images using quadratic and sigmoidal CDF on the next slides since the object (myself XD) was made even darker, but I did appreciate how the clouds and sun were accentuated in the backdrop.

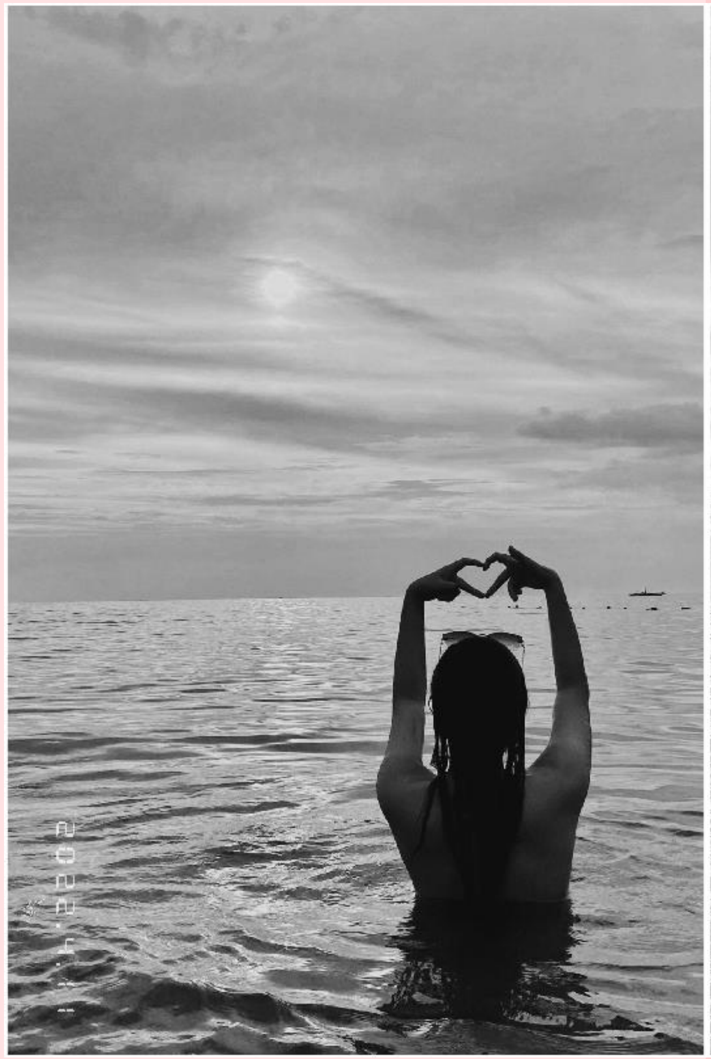


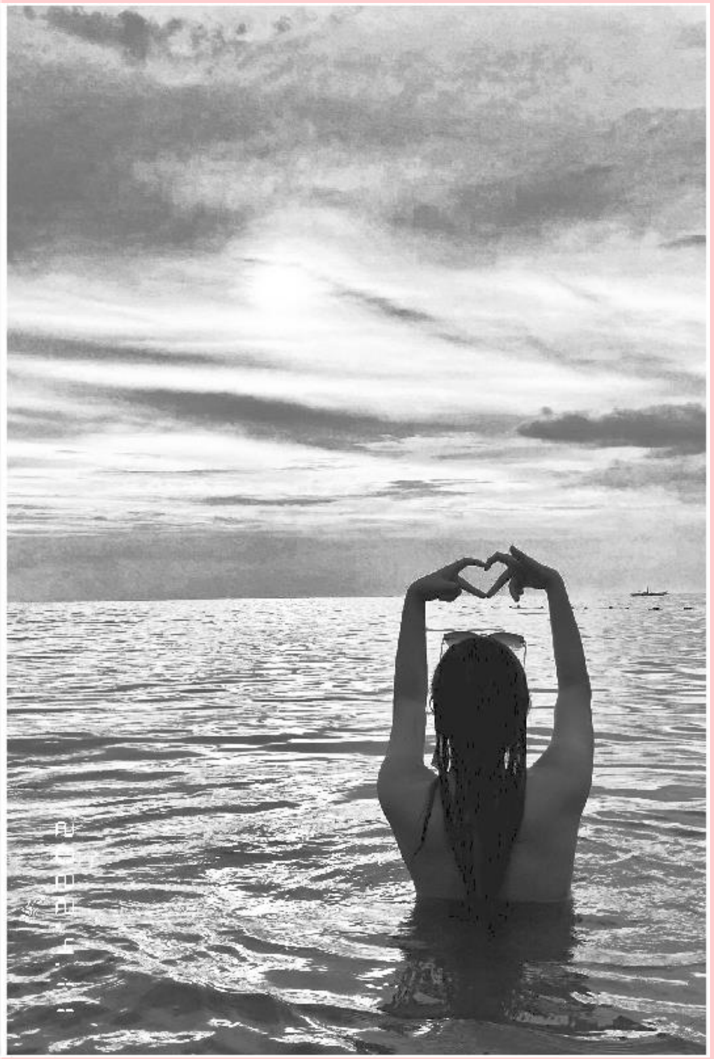
(a) Quadratic CDF



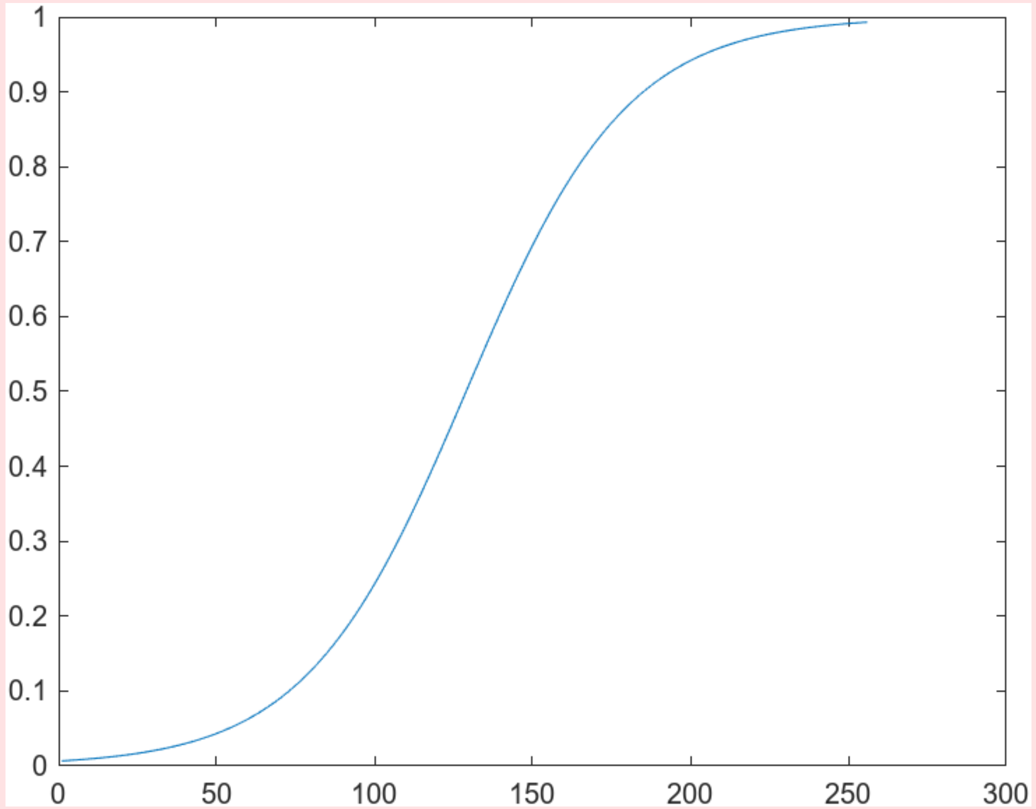
(b) Desired Quadratic CDF

In this case, we stretched the intensity values of an image using a quadratic cumulative distribution function (CDF), much like we did with the linear CDF approach. After we had the Quadratic CDF, we were able to use it to enhance the image, and then utilize that improved image to calculate and plot the Quadratic CDF we wanted.

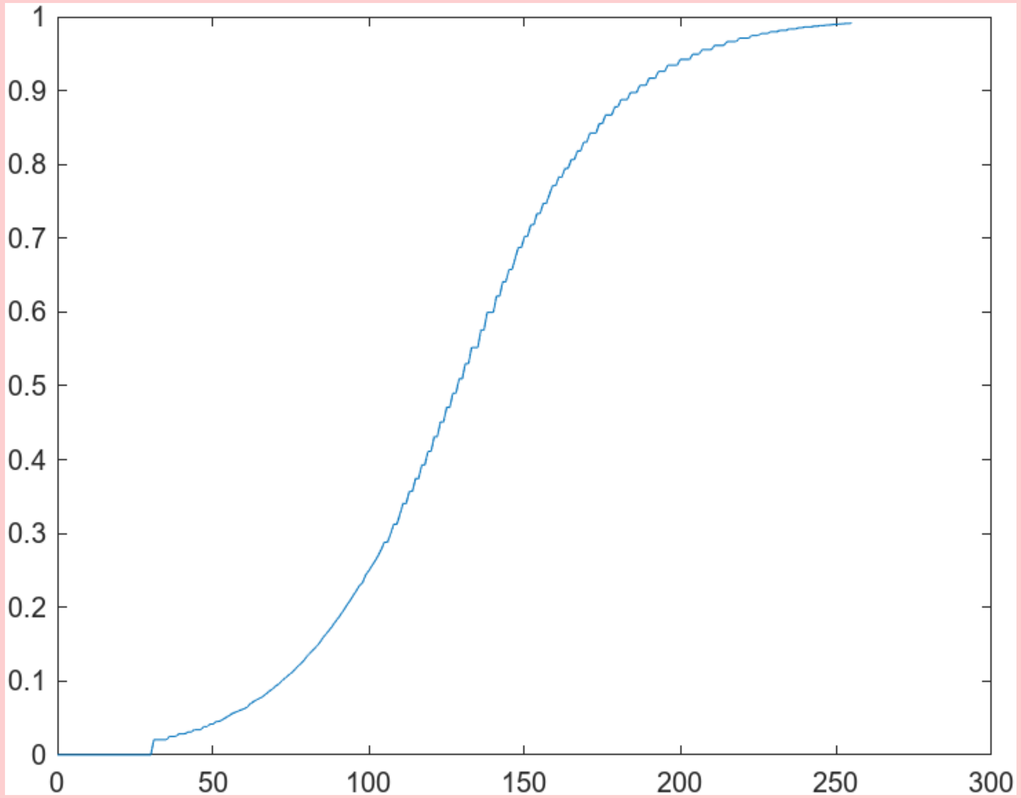

(a) Original grayscale image


(b) Enhanced grayscale image using Quadratic CDF

We can clearly see here that the areas that were too dark or too bright in the original grayscale image, making it impossible to detect the image's features, are now clear and distinct in the image that was improved using quadratic CDF. Moreover, the second image is brighter and has a greater contrast than the first. Unlike the enhanced image using linear CDF, I like the outcome of the object here since it is a lot brighter.

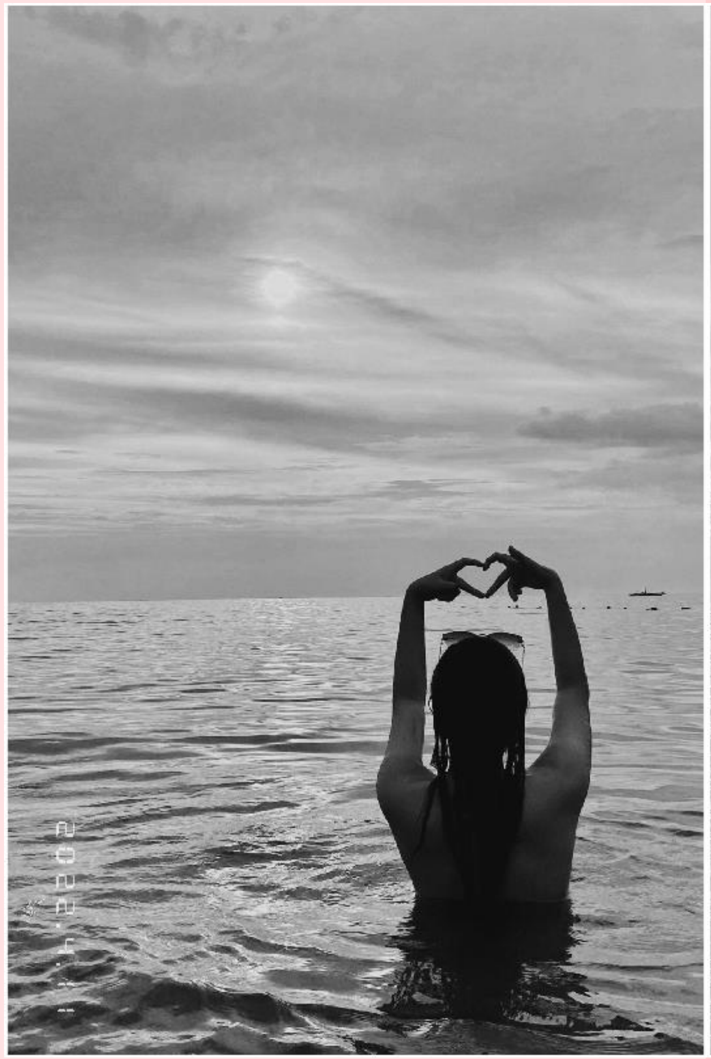


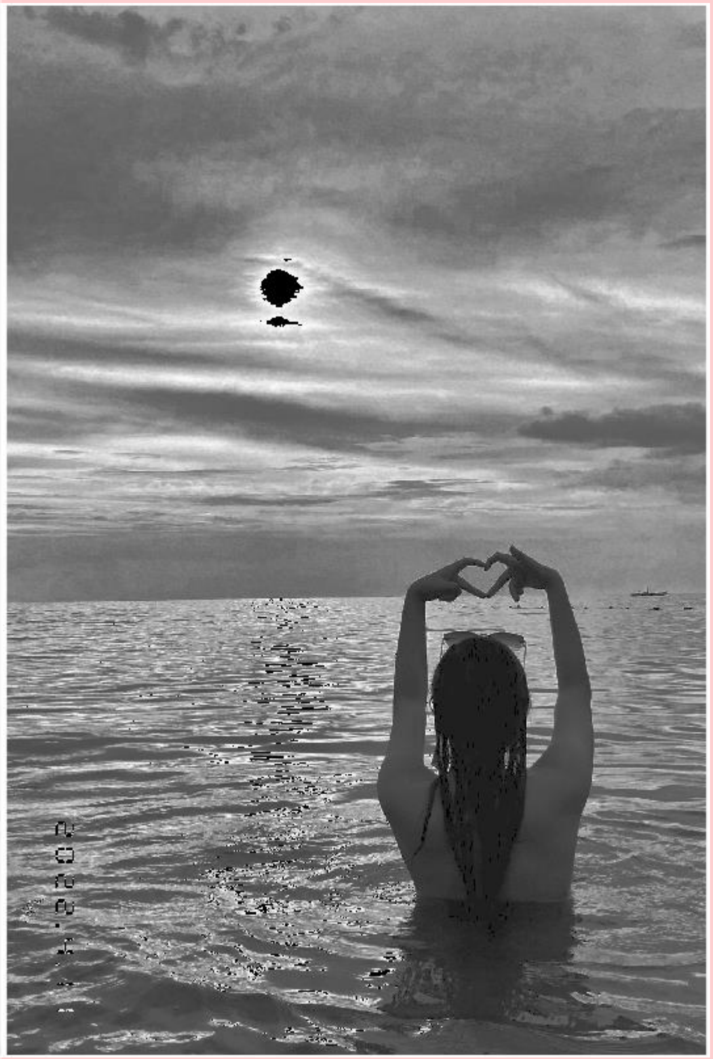
(a) Sigmoid CDF



(b) Desired Sigmoid CDF

The sigmoid function was the last CDF shape used in this activity. Same method to the other shapes, but here we let the variables a = 10 controlling the slope of the curve and b = 128 controlling the curve's midpoint. These variables are necessary to produce the sigmoid-shaped function where we used the code sigmoidCDF = 1 ./ (1 + exp(-a*(x-b)/255));. After producing the enhanced image, we computed and plotted the required sigmoid CDF, as seen above.


(a) Original grayscale image

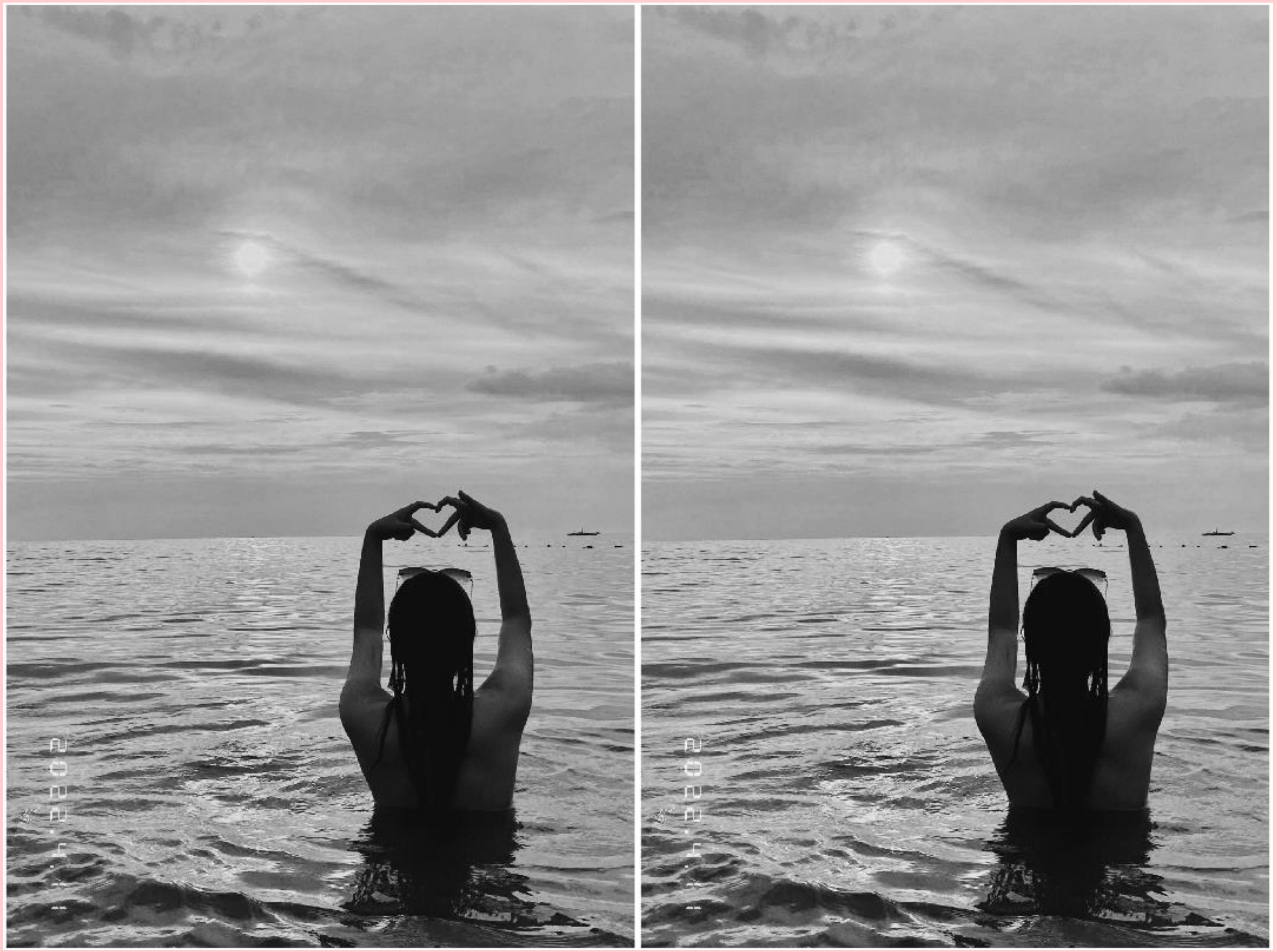

(b) Enhanced grayscale image using Sigmoid CDF

Examining the two images above, we can observe that their contrast and brightness are different. We can also see here that in the enhanced image, there is a certain prominent feature which is the sun. This is due to the fact that the sigmoid transformation can magnify certain image areas while reducing others. This is also considered the best method for image enhancement.

Activity 1.5
Contrast Enhancement

12:00 AM
03/11/2023

The contrast stretched image on the side was created using MATLAB. Using the min and max built-in functions, I determined the minimum and maximum values of the grayscale picture. Then, these values were normalized to the range [0,1] using the equation $I_{new} = \frac{I_{old} - I_{min}}{I_{maz} - I_{min}}$ or in code: I_new = double(I_old - I_min)/double(I_max - I_min); This was done to improve the image's visual quality by enhancing its contrast. Comparing the two, they are nearly identical, with only a minor difference in their contrast.



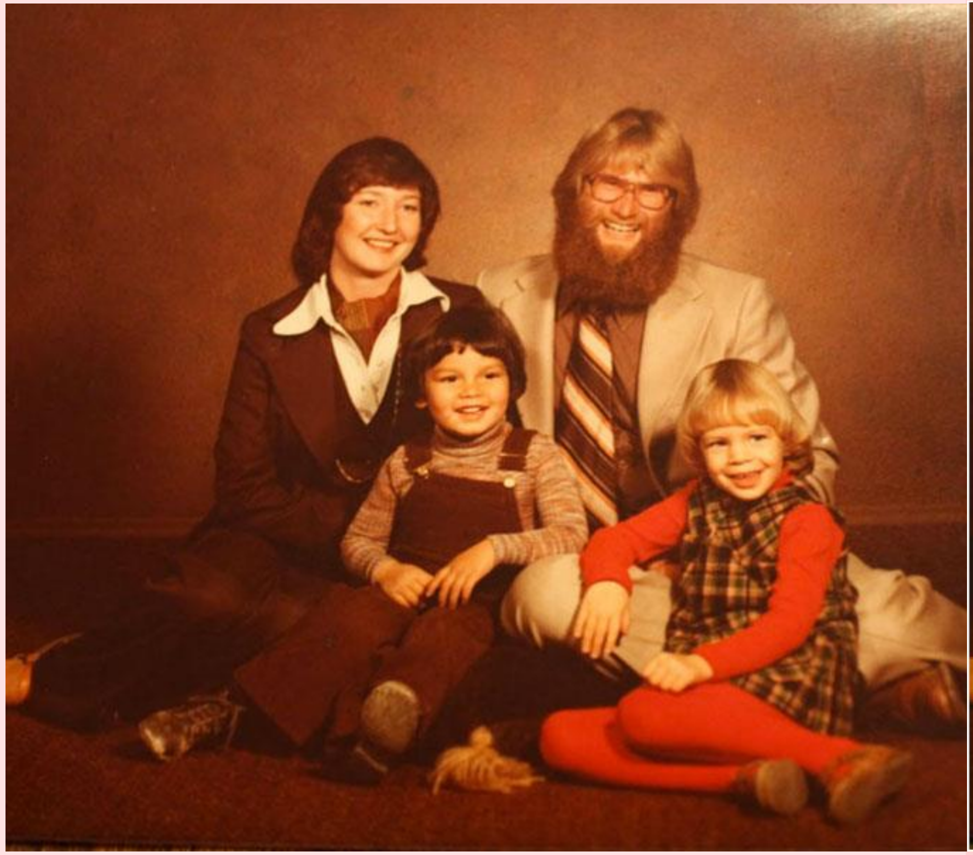
(a) Original grayscale Image (b) contrast-stretched image

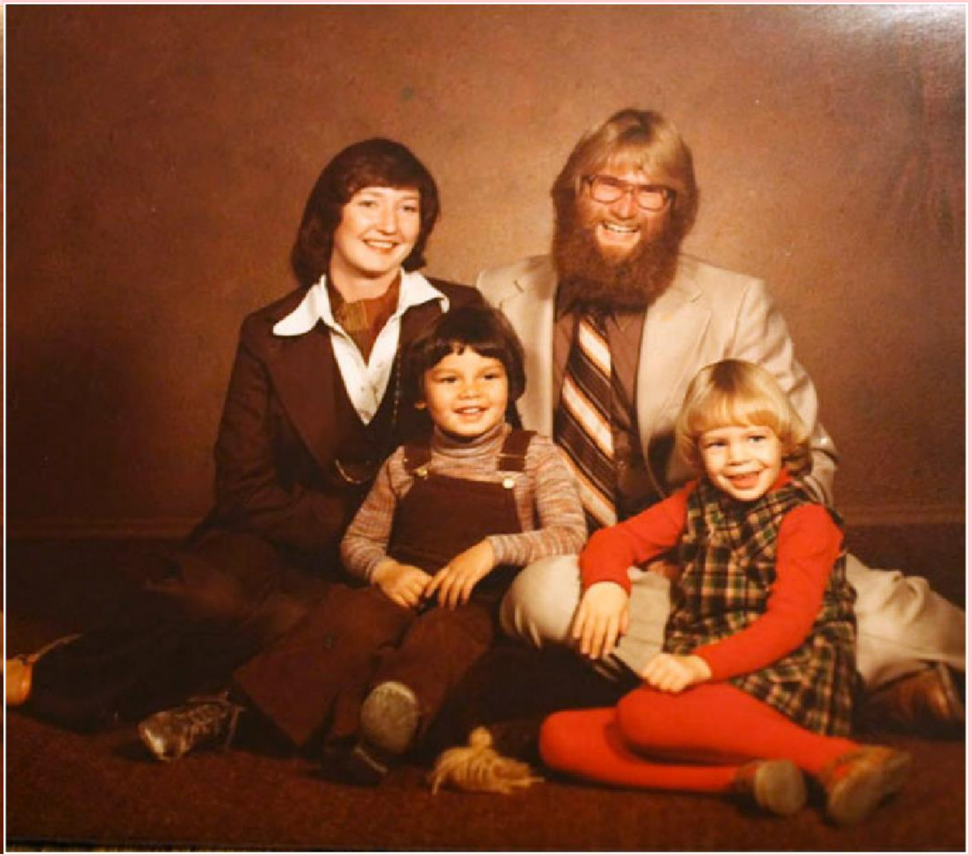
Activity 1.6

Restoring Faded colored Photographs

✓ Compared the original faded colored photograph to the enhanced images using the Contrast stretch, Gray World, and White Patch Algorithm.

12:00 AM
03/11/2023



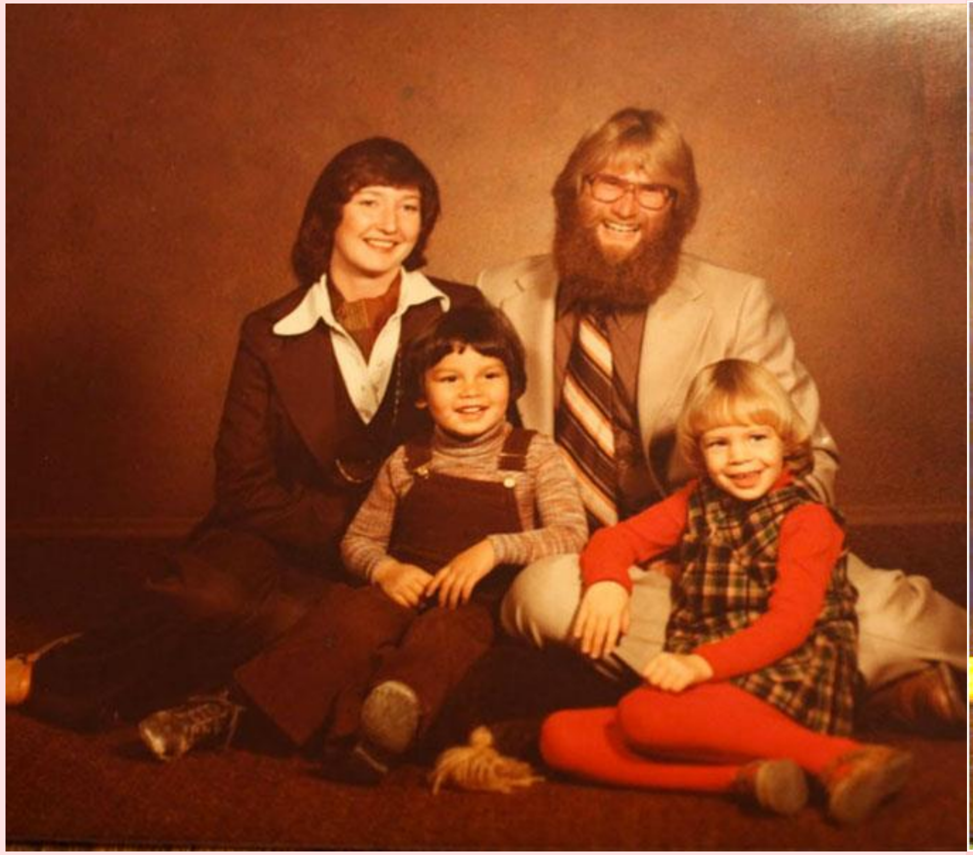


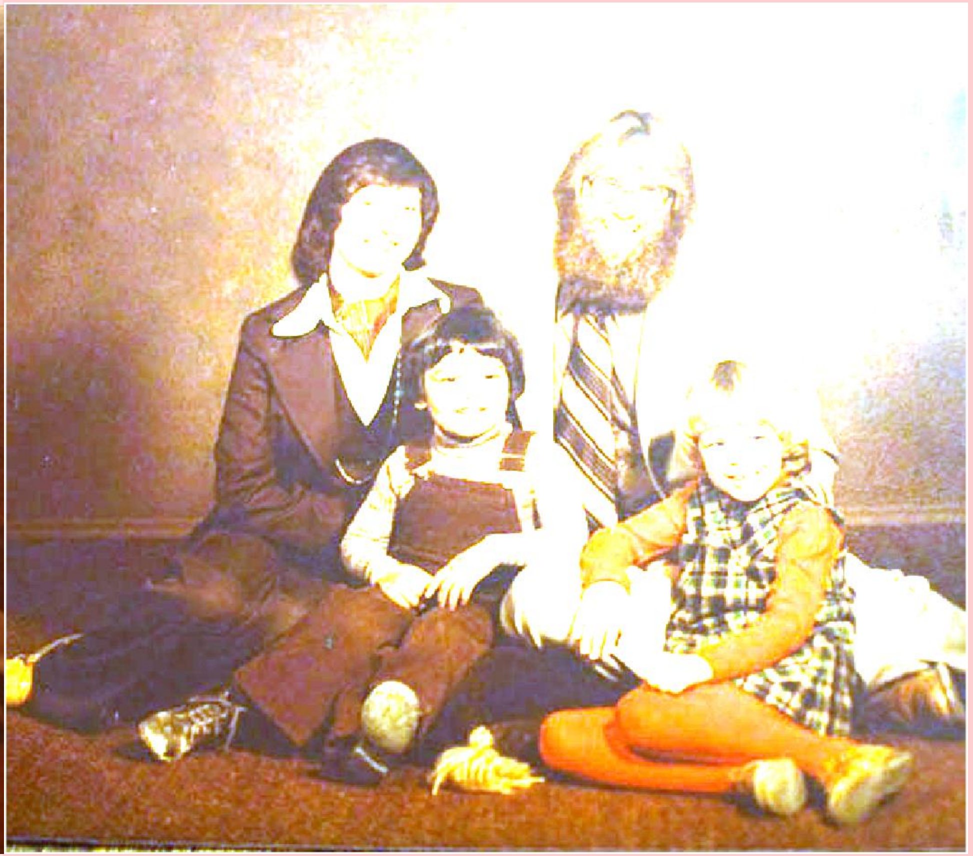
Source: https://d3ui957tjb5bqd.cloudfront.net/uploads/2017/03/Old-Photo.jpg

(a) Faded Colored Photograph

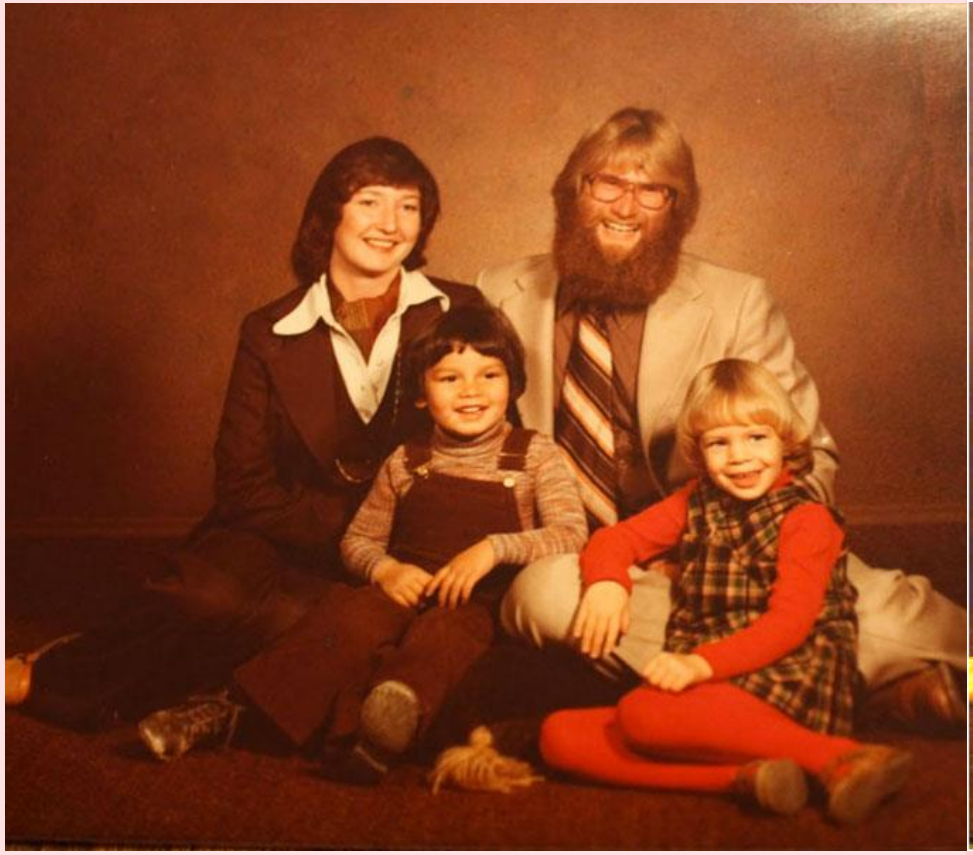
(b) restored faded colored photograph using color stretch

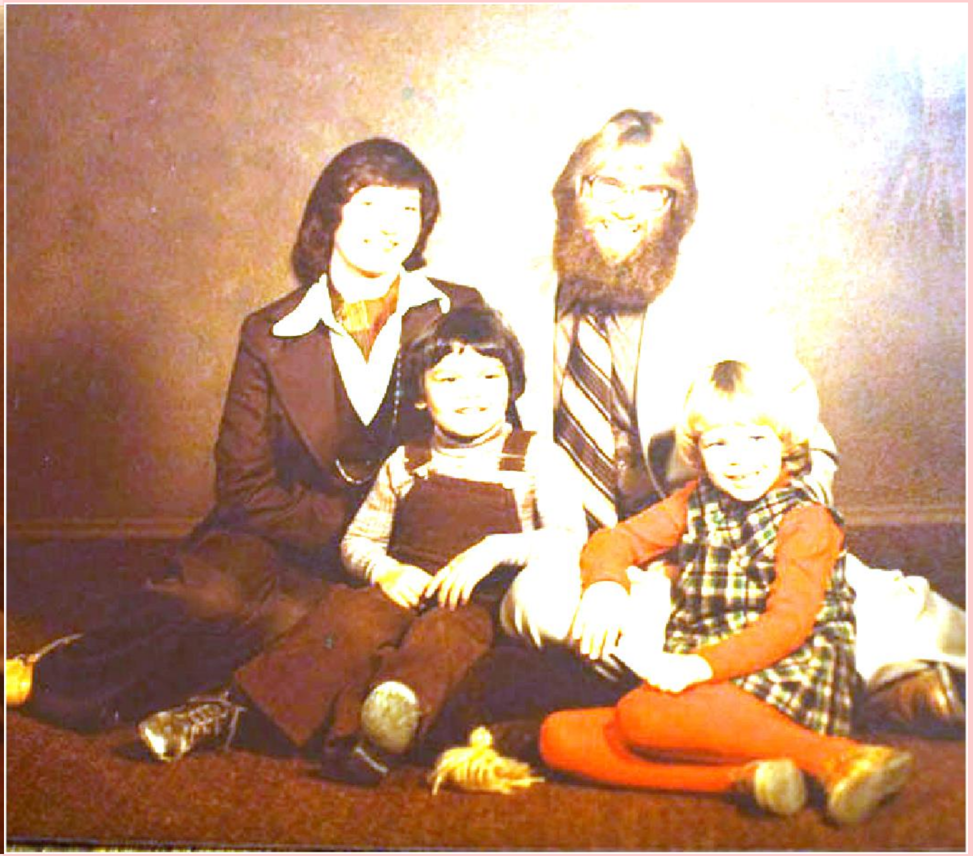
Here, I got a Google image of a fading photograph and restored it using three common algorithms, the first of which is contrast stretch. By the name itself, I performed contrast stretching on each color channel independently by obtaining the minimum and maximum pixel values for each channel. After this, I recombined the stretched channels into RGB using the 'cat' function, and the result is seen above. Of the three improved faded images, I prefer this one the most.


(a) Faded Colored Photograph


(b) restored faded colored photograph Gray World algorithm

The second algorithm in enhancing faded colored photograph is the Gray World algorithm which we performed here. To obtain the white-balanced image, I obtained the average values of each channel and then divided each RGB channel by their respective averages. Similar to before, I used the 'cat' function to merge the channels into a single RGB. Because of the brightness of the resulting image , I don't believe this approach is sufficient for recovering this image, since I can barely see the faces. The best faded photograph for this method is one that has an overall achromatic or gray color balance in order to accomplish its intended goal.


(a) Faded Colored Photograph


(b) restored faded colored photograph using White Patch algorithm

The White patch algorithm is the last algorithm performed in this task to enhance faded photographs. Below, I have selected the part of the image that I believe to be white. This area consists of pixels in the range of rows 250 to 300 pixels and columns 400 to 450 pixels. The color channel was then normalized by dividing each channel by the entire original image using the relevant white averages from each channel. Using the 'cat' function, the normalized channels were merged into a single RGB picture, as with the other two earlier. I believe this algorithm is inappropriate for this image. Note to self: choose the suitable white reference patch the next time I use this method.

REFLECTION (100/100)

As someone who lacks coding skills and barely passed Applied Physics 155, I felt quite overwhelmed by this task. I needed to learn python code and MATLAB code in just 3 weeks for me to be able to do this, which makes me really proud of myself. Looking back at my journey, It was very funny remembering that I was crying the whole time I was trying to make the JWTS primary mirror in Jupyter. This is why I moved to MATLAB, since a buddy who also taught me the gist of several codes recommended it (credits to Mar Princer). Plus, when I was working on this, especially this report, my health was deteriorating (lol), kidding aside, I am so unwell right now. I completed this despite having a fever, cold, headache, and body pain. I went above and beyond to learn how to finish these six sub-activities, which I think makes me a score deserving of 100/100. (please T.T)

10:50 AM
04/03/2021