Assignment 2: Image Histograms and Point Operations Matthew McCaughan Due March 2nd 2025

Problem 1

To determine the height of the column, we'll to use the focal length and the column's dimensions from the camera to construct a ratio between its appearance in the world and in the camera.

12000mm/95000mm = x/50mm = **6.316 mm**

The column has an image height of 6.316 milimeters, and with a camera resolution of 300 pixels per inch which equals 300/25.4 = 11.81 pixels per milimeter.

With a height of 6.316 milimeters, the column will take up 6.316 * 11.81 = 74.59 = **approximately 75 pixels**

Problem 2

[Q2 – 20 pts] Write an ImageJ plugin or Python function that counts the number of pixels of a given color in an image. Use this plugin to count the number of pixels with pure red (255,0,0), pure green (0,255,0), pure blue (0,0,255), white (255,255,255), and black (0,0,0) in an input image (generate such images at size 640x480 to test your code). Test your code on a selfie photo of you and report the result. Test your code on the image shapes.png and report the result.

```
import numpy as np
import matplotlib.pyplot as plt
import PIL
```

Testing with selfie

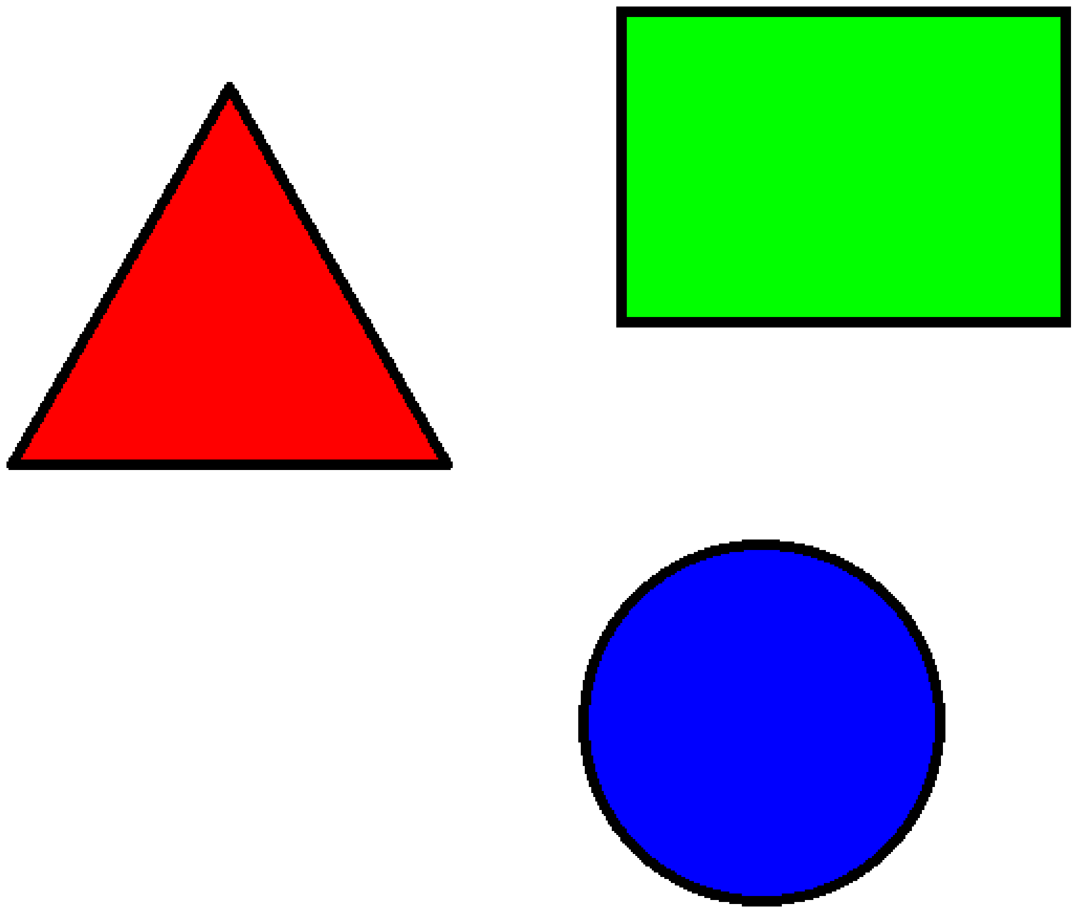
```
import math
selfImage = PIL.Image.open('/content/self.jpg')
img_np = np.array(selfImage)
height,width, z = img_np.shape
```

```
toDisplay = selfImage.resize((math.floor((height/10)),(math.floor((width/6)))))
display(toDisplay)
#print(type(selfImage))
#print(type(img_np))
#print(img_np[1][1] == [229,180,87])
#print(np.array_equal, img_np[1][1] == [229,180,87])
#list(selfImage.getdata())
Rcount = 0
Gcount = 0
Bcount = 0
Wcount = 0
Blcount = 0

for pixel in selfImage.getdata():
  if pixel == (255,0,0):
    Rcount += 1
  if pixel == (0,255,0):
    Gcount +=1
  if pixel == (0,0,255):
    Bcount +=1
  if pixel == (255,255,255):
    Wcount +=1
  if pixel == (0,0,0):
    Blcount +=1
print("self.jpg pixel count:")
print("Number of Red Pixels: " , Rcount)
print("Number of Green Pixels: ", Gcount)
print("Number of Blue Pixels: ", Bcount)
print("Number of White Pixels: ", Wcount)
print("Number of Black Pixels: ", Blcount)
```

```
self.jpg pixel count:
Number of Red Pixels:  0
Number of Green Pixels:  0
Number of Blue Pixels:  0
Number of White Pixels:  7
Number of Black Pixels:  55
```

Testing on shapes.png

```
ShapeImage = PIL.Image.open('/content/Shapes.png')
display(ShapeImage)
img_np = np.array(ShapeImage)
height,width, z = img_np.shape
#print(type(currentImage))
#print(type(img_np))
#print(img_np[1][1] == [229,180,87])
#print(np.array_equal, img_np[1][1] == [229,180,87])
#list(currentImage.getdata())
Rcount = 0
Gcount = 0
Bcount = 0
Wcount = 0
Blcount = 0

for pixel in ShapeImage.getdata():
  if pixel == (255,0,0):
    Rcount += 1
  if pixel == (0,255,0):
    Gcount +=1
  if pixel == (0,0,255):
    Bcount +=1
  if pixel == (255,255,255):
    Wcount +=1
  if pixel == (0,0,0):
    Blcount +=1
print("shapes.png pixel count:")
print("Number of Red Pixels: " , Rcount)
print("Number of Green Pixels: ", Gcount)
print("Number of Blue Pixels: ", Bcount)
print("Number of White Pixels: ", Wcount)
print("Number of Black Pixels: ", Blcount)
```

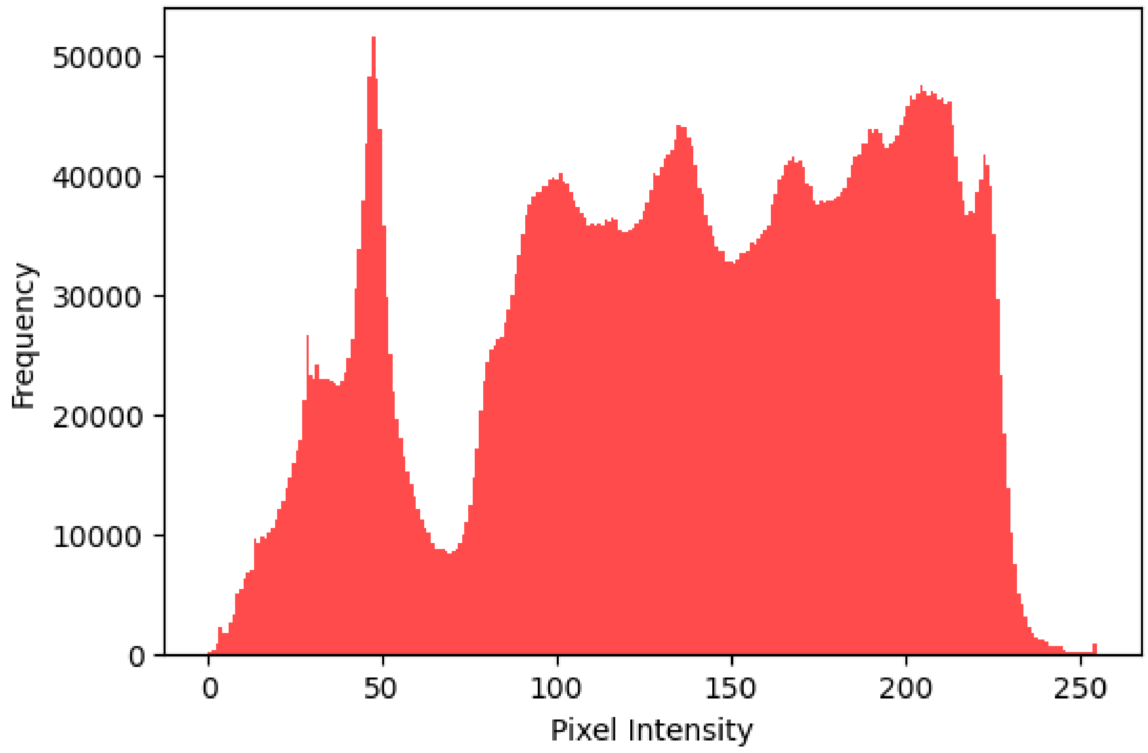


```
shapes.png pixel count:
Number of Red Pixels:  17197
Number of Green Pixels:  29601
Number of Blue Pixels:  21377
Number of White Pixels:  229662
Number of Black Pixels:  9363
```

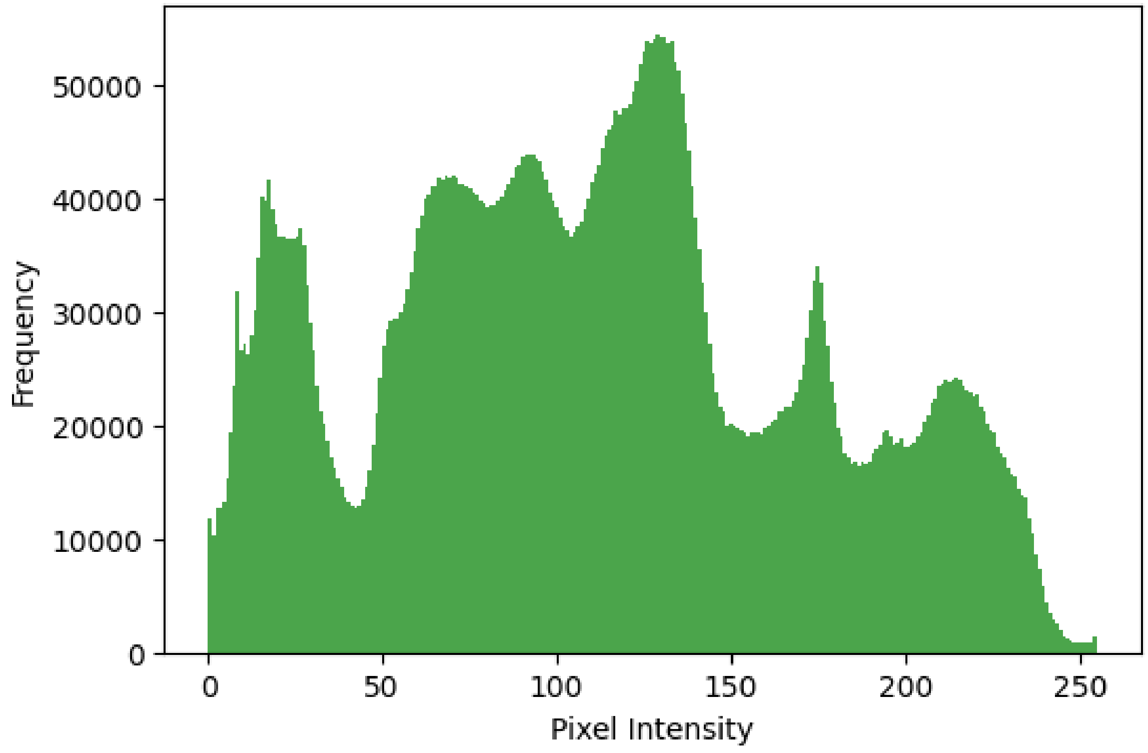
Problem 3

```
color_array = np.array(selfImage)

# Extract RGB channels
r, g, b = color_array[:,:,0], color_array[:,:,1], color_array[:,:,2]

# Plot Individual Color Histograms
colors = ['red', 'green', 'blue']
for i, channel in enumerate([r, g, b]):
    plt.figure(figsize=(6, 4))
    plt.hist(channel.flatten(), bins=256, color=colors[i], alpha=0.7)
    plt.title(f"{i+4}. {colors[i].capitalize()} Histogram")
    plt.xlabel("Pixel Intensity")
    plt.ylabel("Frequency")
    plt.show()
grayScale = (r+g+b)/3
grayScale = grayScale.astype(np.uint8)
plt.figure(figsize=(6, 4))
plt.hist(grayScale.flatten(), bins=256, color='gray', alpha=0.7)
plt.title("Grayscale (Luminance) Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()
```

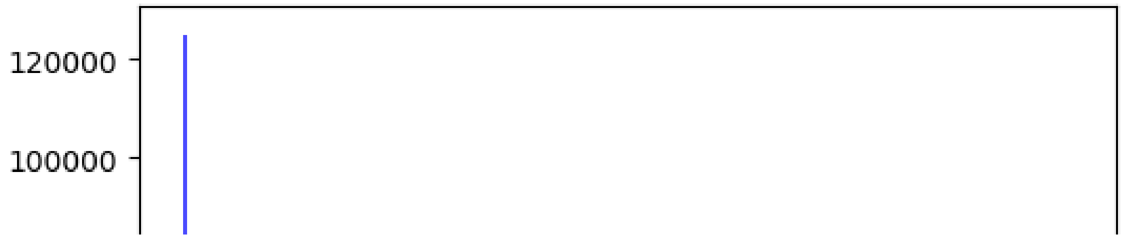
4. Red Histogram

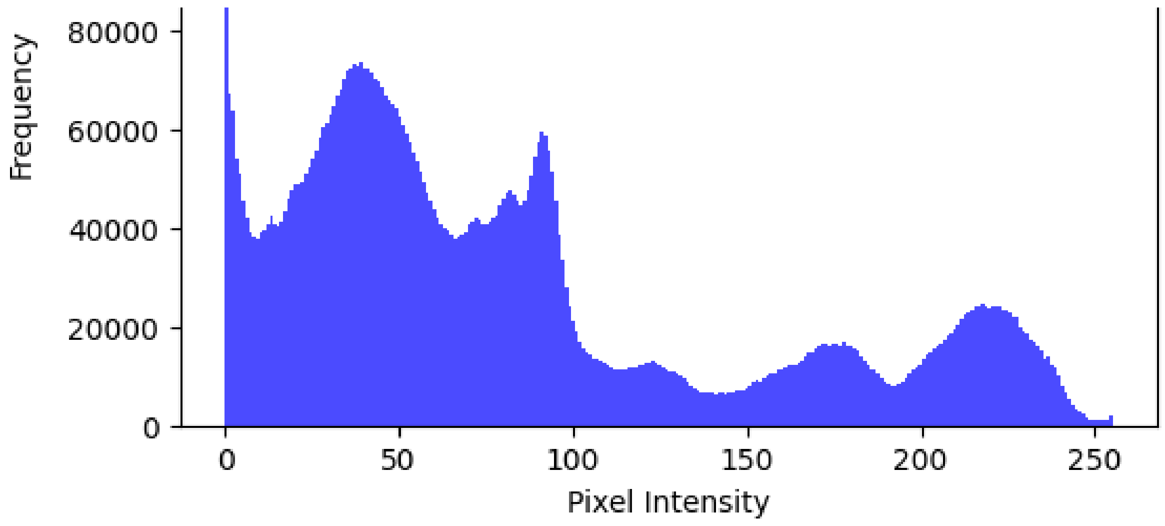


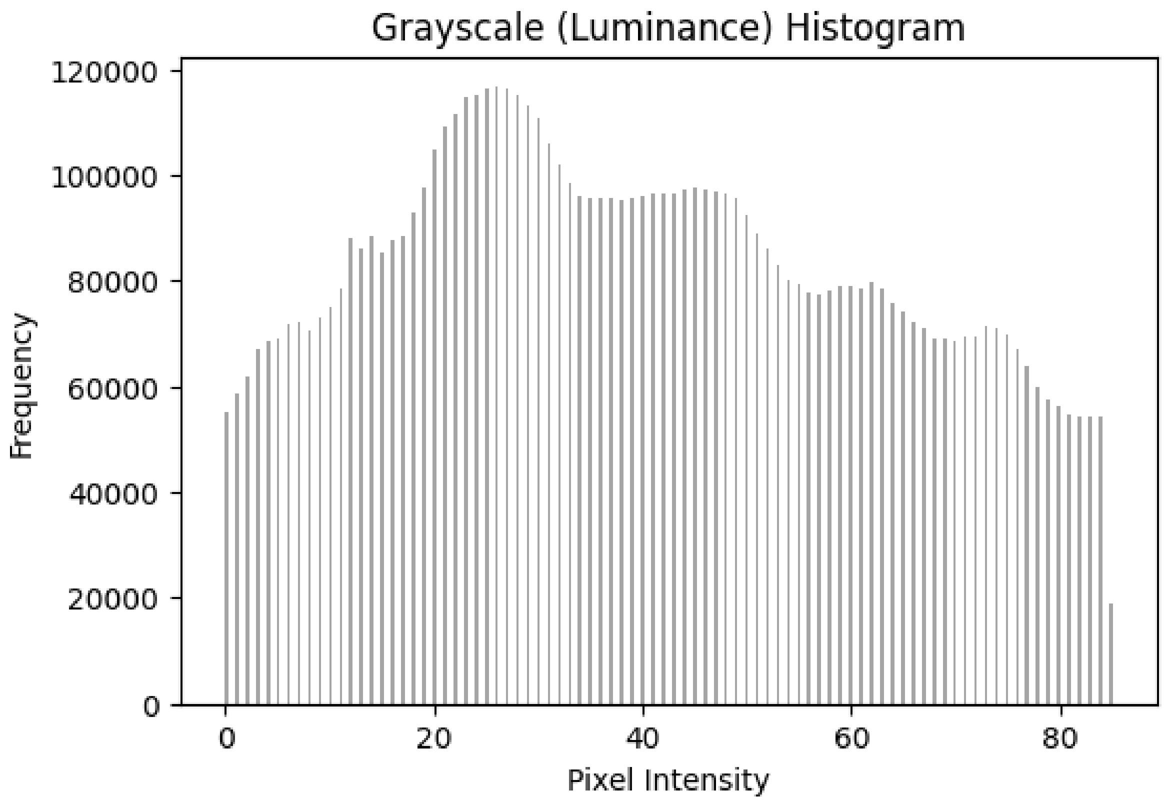
5. Green Histogram



6. Blue Histogram




Frequency
0
20000
40000
60000
80000
0
50
100
150
200
250
Pixel Intensity


Grayscale (Luminance) Histogram
Frequency
0
20000
40000
60000
80000
100000
120000
0
20
40
60
80
Pixel Intensity

Based on the resultant Grayscale Histogram, the dynamic range of the image has been reduced, with spikes at certain pixel intensities instead of a continuous curve. This is likely due to then compression of the image before performing these operations

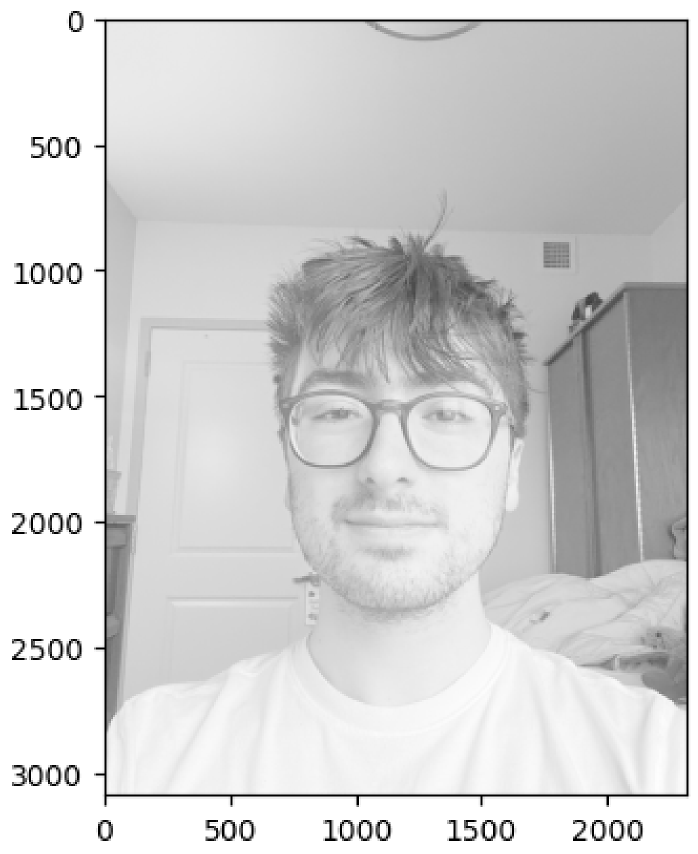
Problem 4

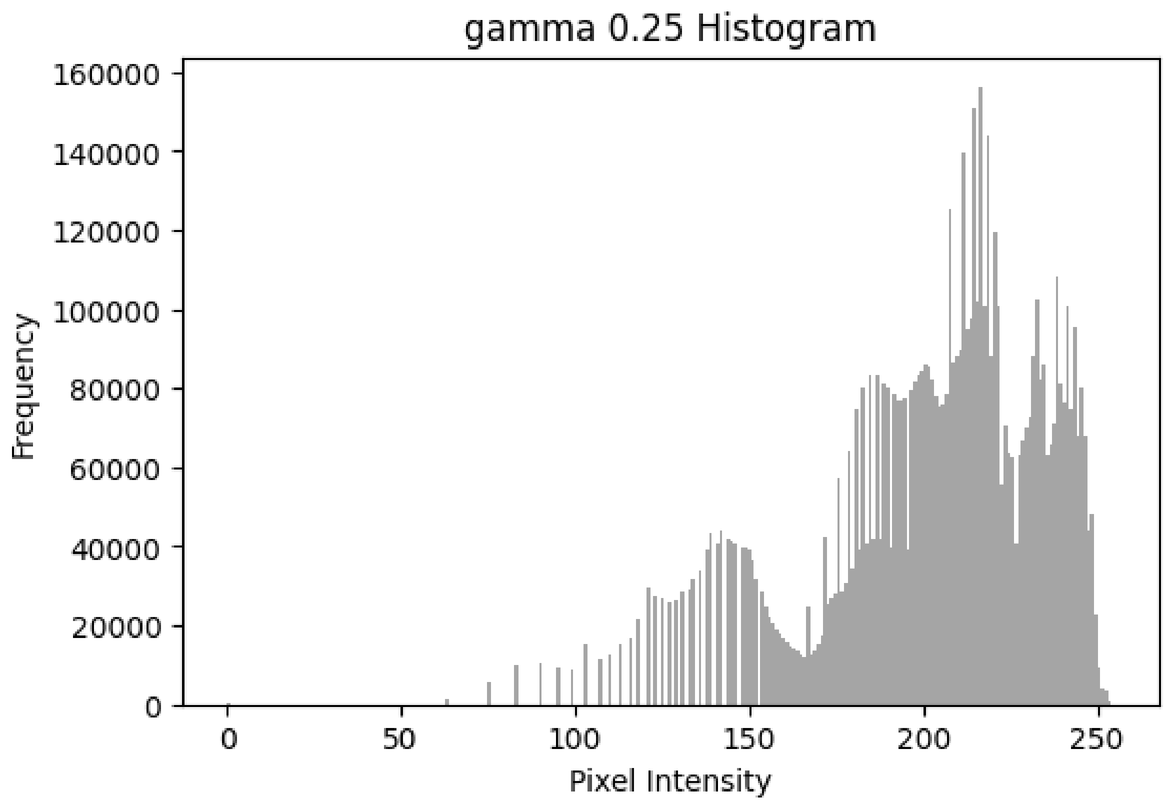
```
GraySelf = PIL.Image.open("/content/self.jpg").convert("L")
GrayArray = np.array(GraySelf)
#display(GraySelf)
```

Gamma Correction Function

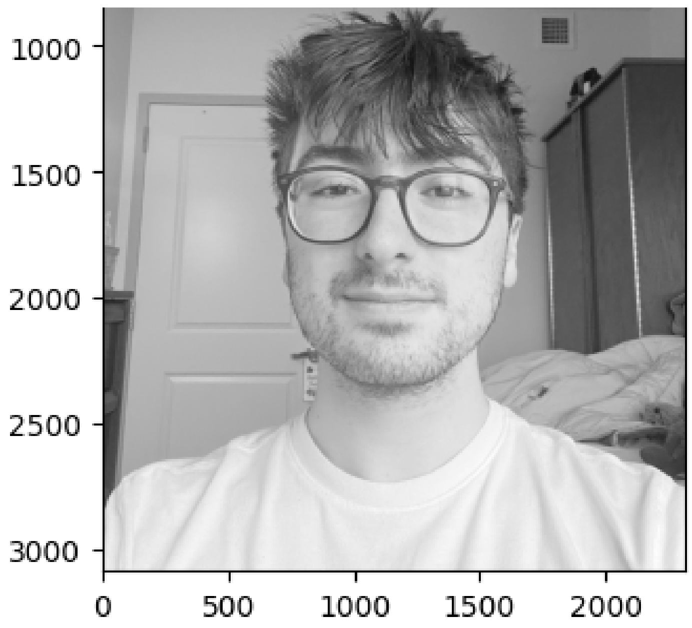
```
def gammaCorrect(imageToChange, gammaValue):
  GammInvert = 1.0 / gammaValue
  imageToChange = np.power(imageToChange.astype(np.float32)/255,gammaValue) * 255.0
  return imageToChange.astype(np.uint8)
```

```
import matplotlib.pyplot as plt
#print(GrayArray.shape)
result1 = gammaCorrect(GrayArray,.25)
result2 = gammaCorrect(GrayArray,.5)
result3 = gammaCorrect(GrayArray,2)
result4 = gammaCorrect(GrayArray,4)

plt.imshow(result1,cmap="gray")
plt.show()
plt.figure(figsize=(6, 4))
plt.hist(result1.flatten(), bins=256, color='gray', alpha=0.7)
plt.title("gamma 0.25 Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()

plt.imshow(result2,cmap="gray")
plt.show()
plt.figure(figsize=(6, 4))
plt.hist(result2.flatten(), bins=256, color='gray', alpha=0.7)
plt.title("gamma 0.5 Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()

plt.imshow(result3,cmap="gray")
plt.show()
plt.figure(figsize=(6, 4))
plt.hist(result3.flatten(), bins=256, color='gray', alpha=0.7)
plt.title("gamma 2 Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()

plt.imshow(result4,cmap="gray")
plt.show()
plt.figure(figsize=(6, 4))
plt.hist(result4.flatten(), bins=256, color='gray', alpha=0.7)
plt.title("gamma 4 Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()
```

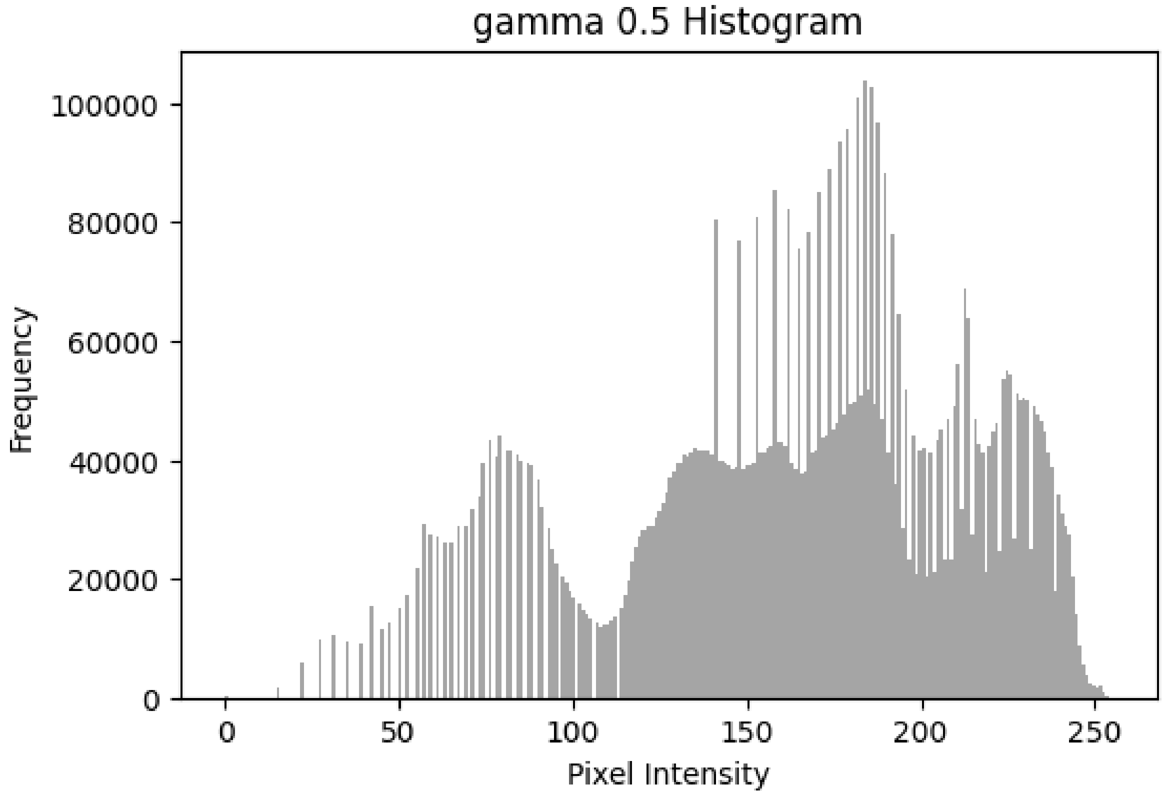

0
500
1000
1500
2000
2500
3000
0
500
1000
1500
2000


gamma 0.25 Histogram
Frequency
Pixel Intensity
0
20000
40000
60000
80000
100000
120000
140000
160000
0
50
100
150
200
250

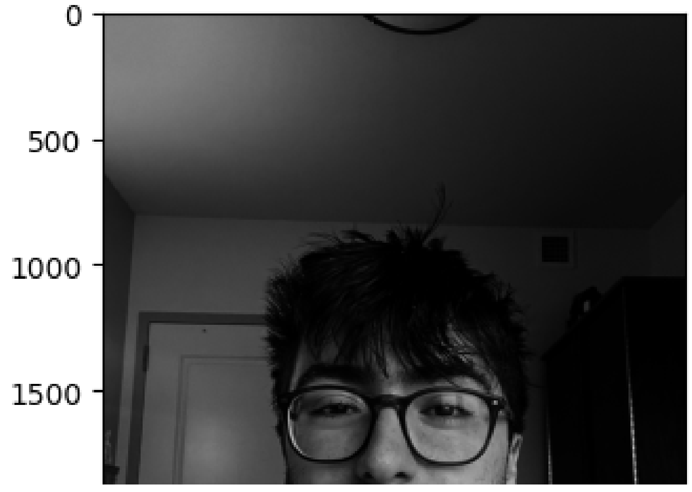
0
500



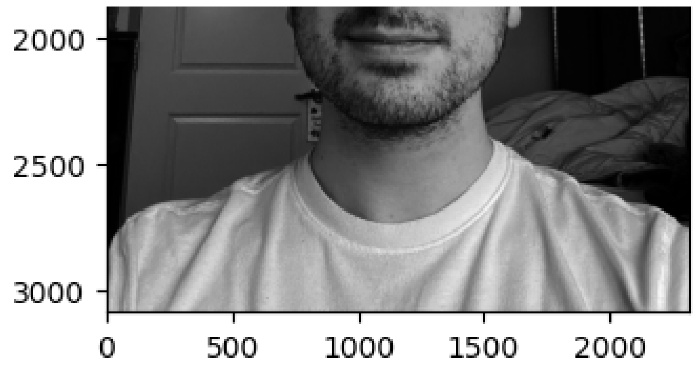
1000
1500
2000
2500
3000
0
500
1000
1500
2000

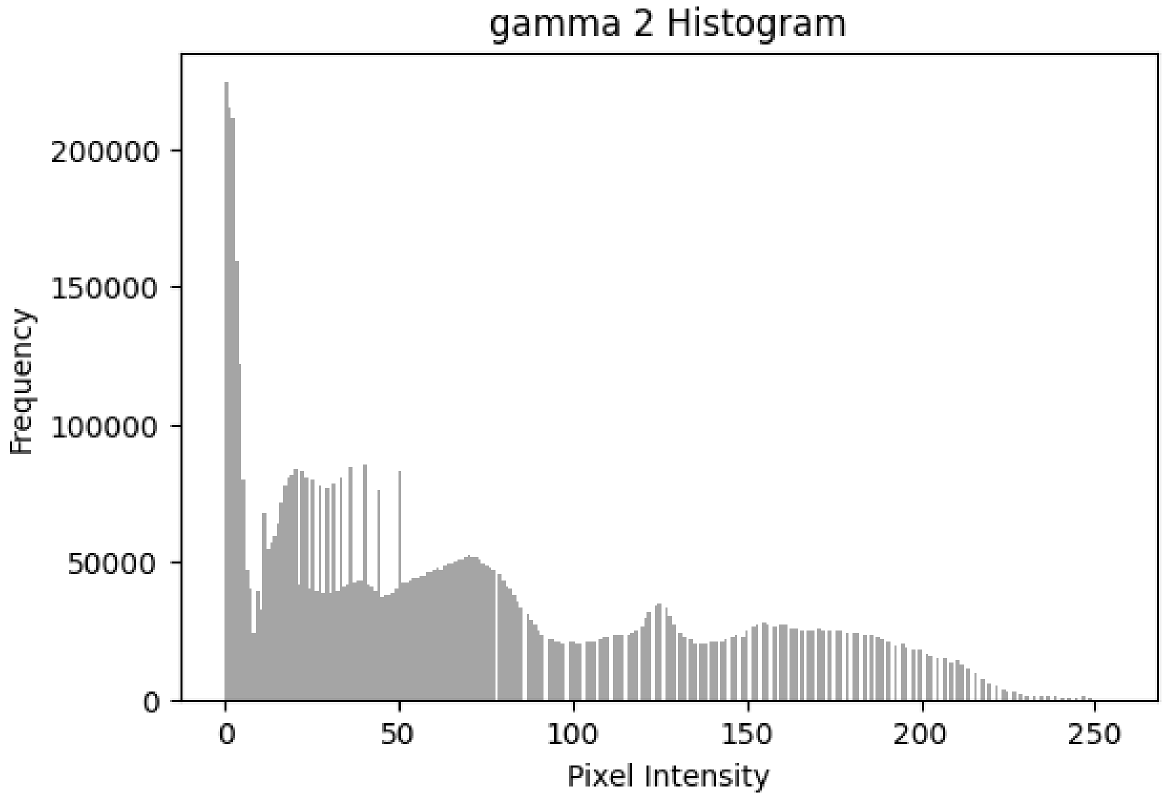


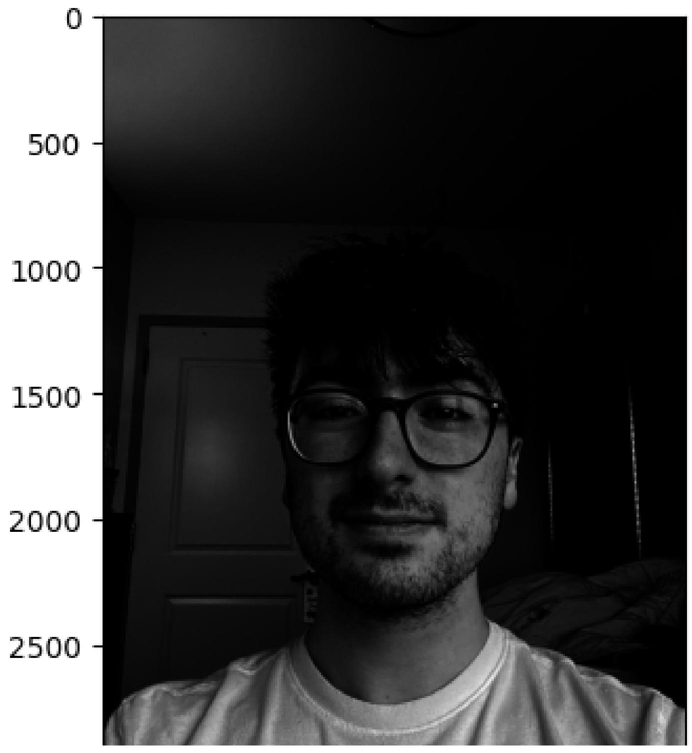
gamma 0.5 Histogram
Frequency
Pixel Intensity
0
20000
40000
60000
80000
100000
0
50
100
150
200
250

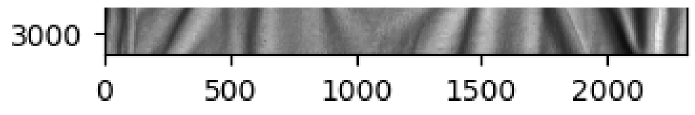


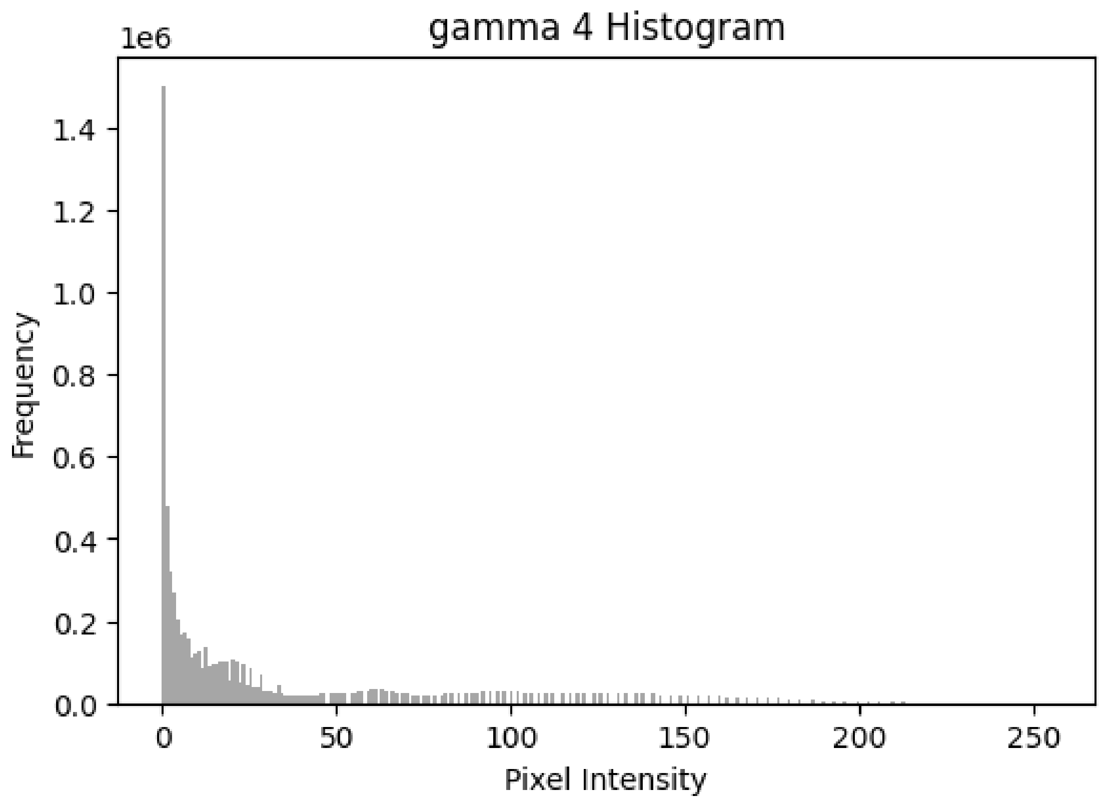
0
500
1000
1500


2000
2500
3000
0
500
1000
1500
2000


gamma 2 Histogram
Frequency
Pixel Intensity
0
50000
100000
150000
200000
0
50
100
150
200
250


0
500
1000
1500
2000
2500





Based on the results of changing the gamma for these four images, I believe the original downloaded image gives the most faithful representation of what i see on my phone. These gammas chosen produce an image that is too bright or too dark relative to the original image on my phone.

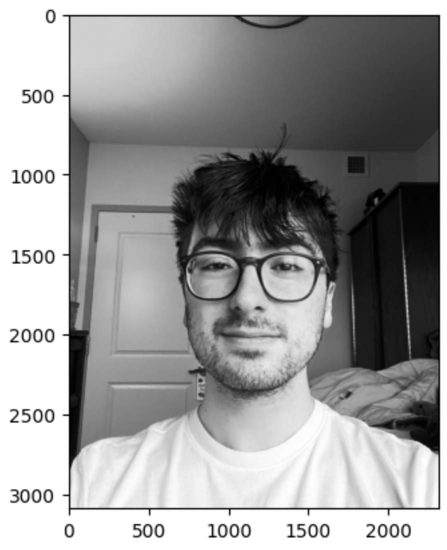
```
plt.title("gamma 4 Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()
```

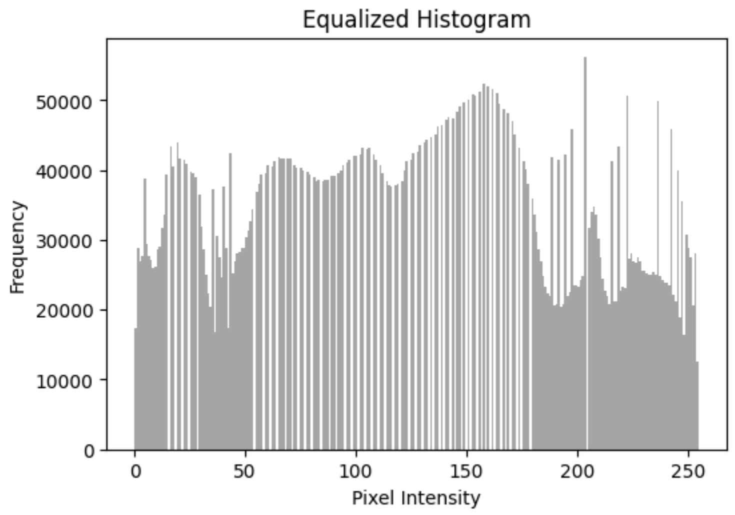
Show hidden output

Based on the results of changing the gamma for these four images, I believe the original downloaded image gives the most faithful representation of what i see on my phone. These gammas chosen produce an image that is too bright or too dark relative to the original image on my phone.

Histogram Equalization Function:

```
Equalized = PIL.ImageOps.equalize(GraySelf)
Arr = np.array(Equalized)
plt.imshow(Equalized,cmap="gray")
plt.show()

plt.figure(figsize=(6, 4))
plt.hist(Arr.flatten(), bins=256, color='gray', alpha=0.7)
plt.title("Equalized Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()
```





```
Start coding or generate with AI.
```