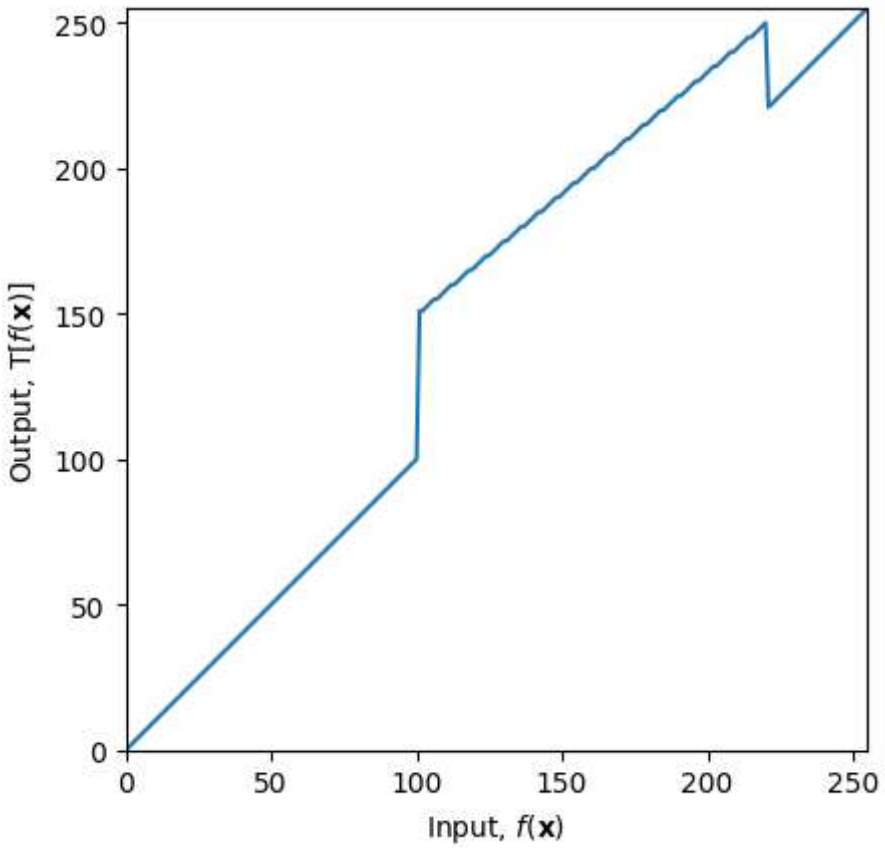
In [ ]:
```
%matplotlib inline
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np

c = np.array([(100,100), (100,150), (220,250), (220,220)])

t1 = np.linspace(0, c[0,1], c[0,0]+1 - 0 ).astype('uint8')
print(len(t1))
t2 = np.linspace(c[0,1]+1, c[1,1], c[1,0]-c[0,0]).astype('uint8')
print(len(t2))
t3 = np.linspace(c[1,1]+1, c[2,1], c[2,0] - c[1,0] ).astype('uint8')
print(len(t3))
t4 = np.linspace(c[2,1]+1, c[3,1], c[3,0] - c[2,0]).astype('uint8')
print(len(t4))
t5 = np.linspace(c[3,1]+1, 255, 255- c[3,0] ).astype('uint8')
print(len(t5))

transform = np.concatenate((t1,t2), axis=0).astype('uint8')
transform = np.concatenate((transform,t3), axis=0).astype('uint8')
transform = np.concatenate((transform,t4), axis=0).astype('uint8')
transform = np.concatenate((transform,t5), axis=0).astype('uint8')

print(len(transform))

fig , ax = plt.subplots()
ax.plot(transform)
ax.set_xlabel(r'Input, $f(\mathbf{x})$')
ax.set_ylabel('Output, $\mathrm{T}[f(\mathbf{x})]$')
ax.set_xlim(0,255)
ax.set_ylim(0,255)
ax.set_aspect('equal')
# plt.savefig('../transform.png')

plt.show()


img_org = cv.imread('../Resources/natasha_grayscale.jpg', cv.IMREAD_GRAYSCALE)
print(img_org.shape)
cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', img_org)
cv.waitKey(0)
image_transformed = cv.LUT(img_org,transform)
cv.imshow('Image', image_transformed)
cv.waitKey(0)
cv.destroyAllWindows()
```

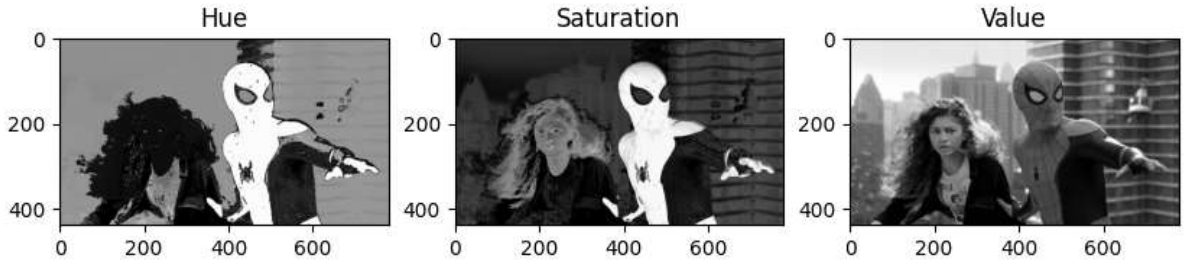
```
101
0
120
0
35
256
```

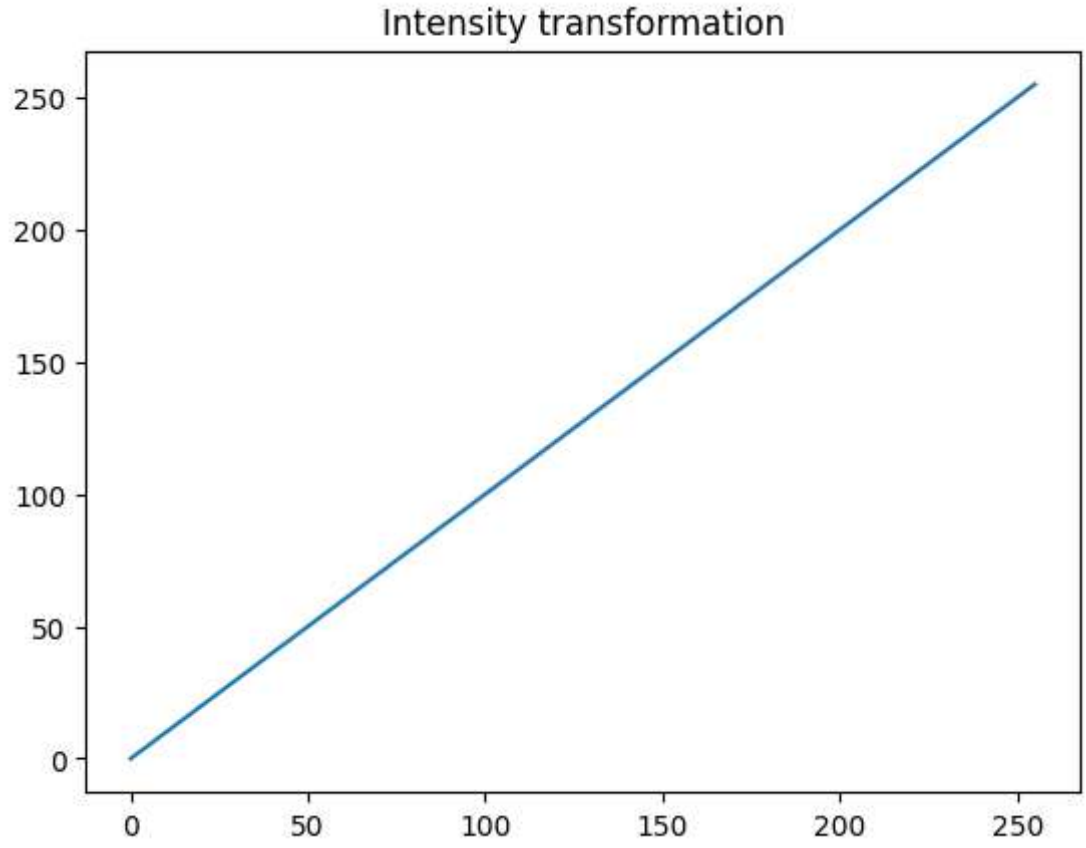


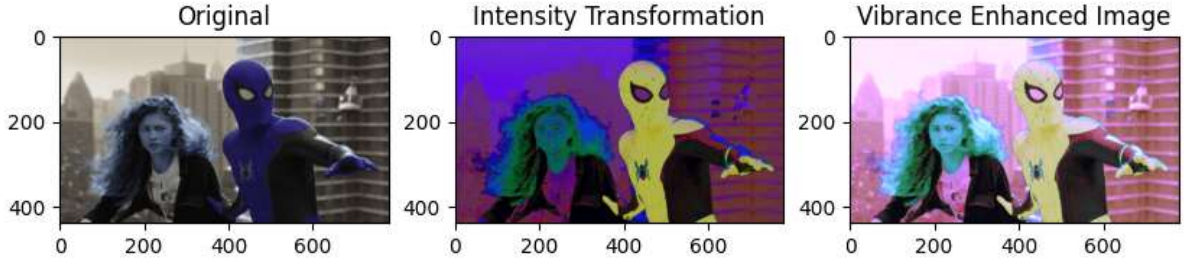

```
(400, 700)
```

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np
img = cv.imread('../Resources/spider.png')
if img is None:
    print('Image could not be read')
    assert False
img1 = cv.cvtColor(img, cv.COLOR_BGR2HSV)
hue, saturation, value = cv.split(img1)
fig, ax= plt.subplots(1,3, figsize=(10,20))
ax[0].imshow(hue, cmap="gray")
ax[0].set_title('Hue')
ax[1].imshow(saturation, cmap="gray")
ax[1].set_title('Saturation')
ax[2].imshow(value, cmap="gray")
ax[2].set_title('Value')
plt.show() # result up to here is in answer a

x= np.arange(0, 256).astype('uint8') # define range for x variable
a = 0.1
sigma = 70
T = np.minimum(((x)+(a*(np.exp(-(x-128)**2/(2*sigma**2))))/128), 255).astype('uint8
# given intensity transformation
image_transform = cv.LUT(saturation, T) # adding transformation to saturation plane
plt.title('Intensity transformation')
plt.plot(T)
plt.show() # result here in in answer b
```

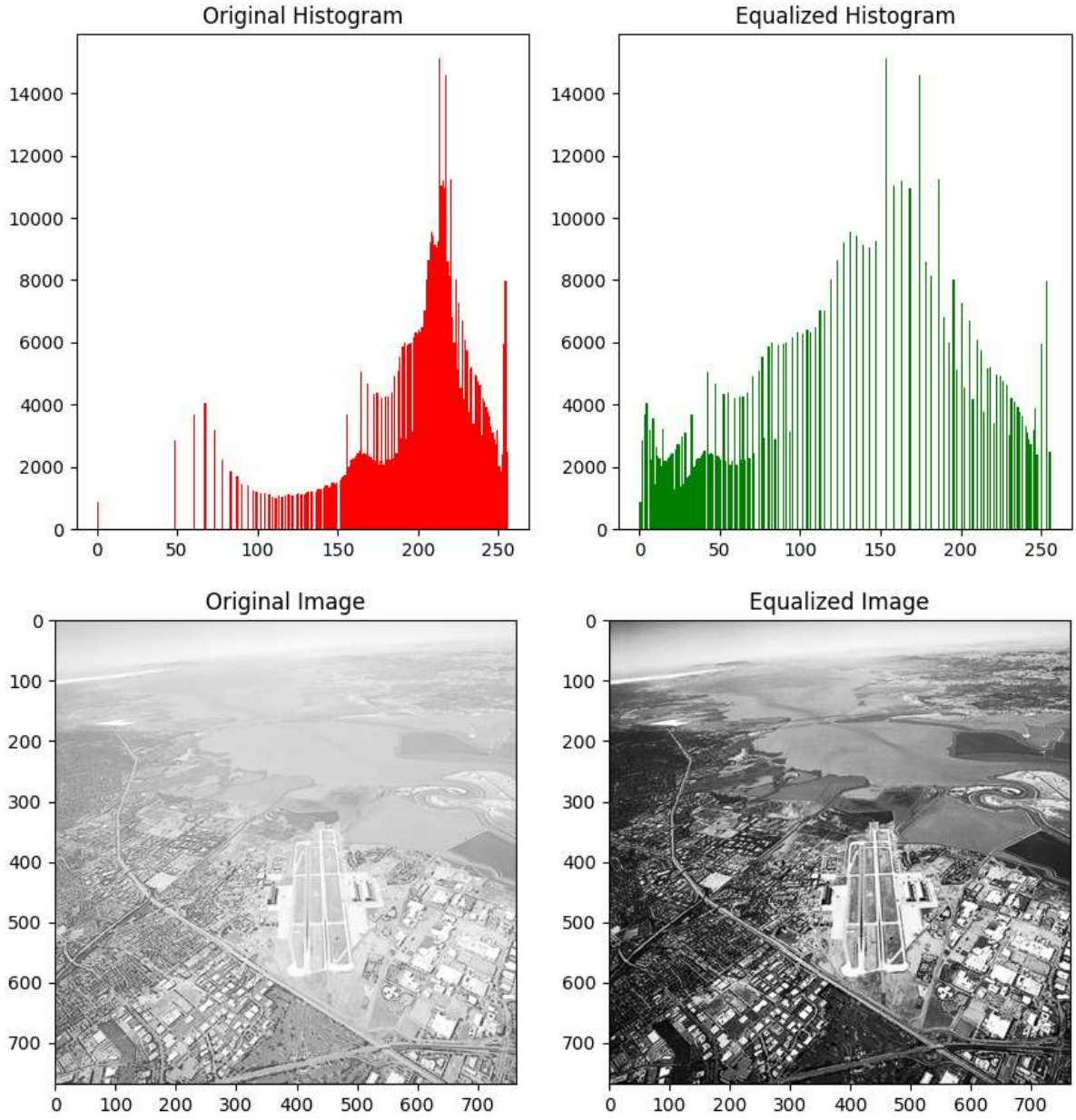
```
new_HSV = cv.merge([hue,image_transform,value])
result =  cv.cvtColor(new_HSV,  cv.COLOR_HSV2BGR)
added_img = cv.add(new_HSV, img)
fig, ax= plt.subplots(1,3, figsize=(10,20))
ax[0].imshow(img, cmap="gray")
ax[0].set_title('Original')
ax[1].imshow(new_HSV, cmap="gray")
ax[1].set_title('Intensity Transformation')
ax[2].imshow(added_img, cmap="gray")
ax[2].set_title('Vibrance Enhanced Image')
plt.show() # result of here is given in answer d
```





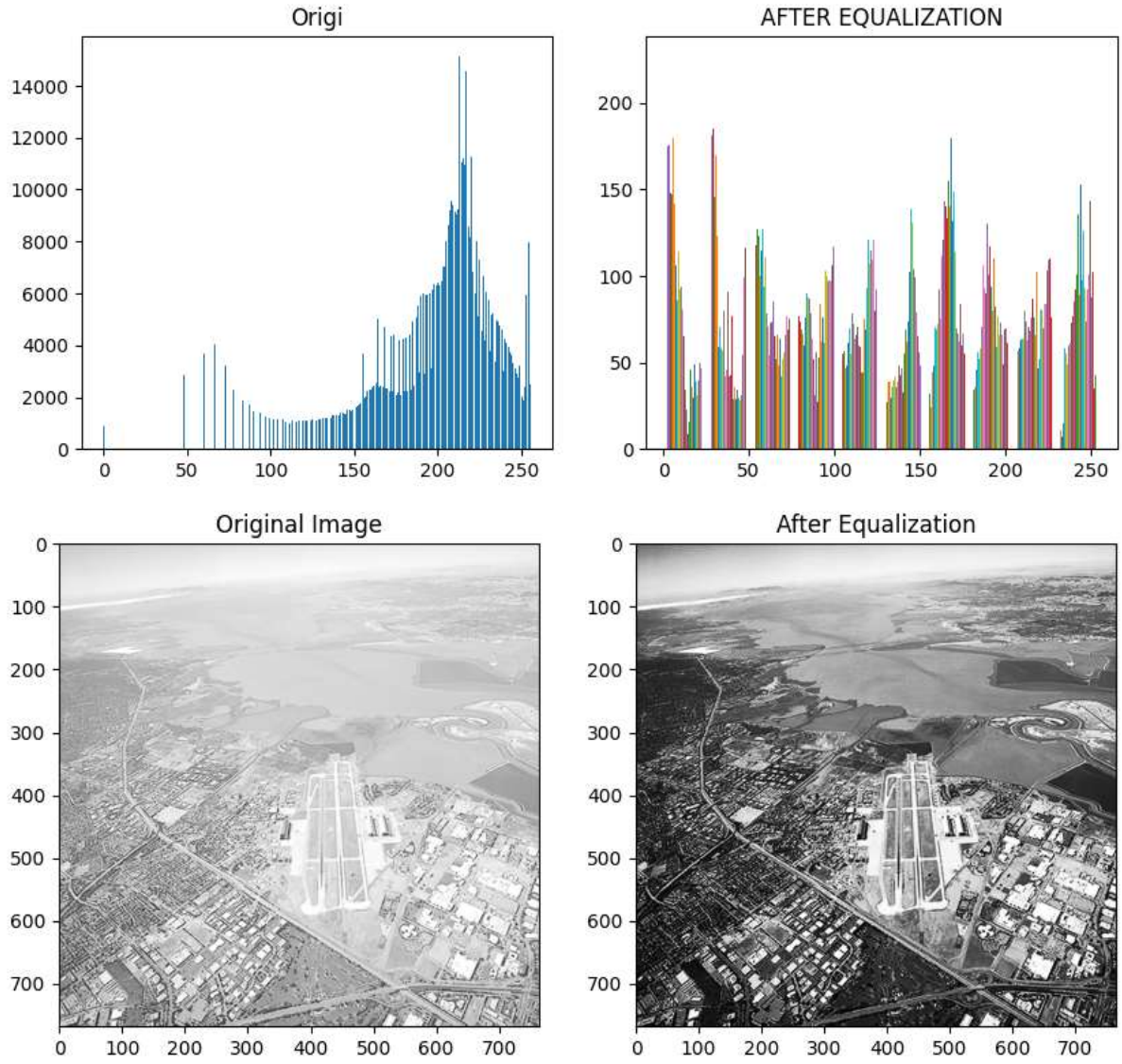


In [ ]:
```python
# Question 4
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

def Hist_function(img):
    gray_img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    hist, bins = np.histogram(gray_img.flatten(), 256, [0, 256]) #normal histogram
    cdf = hist.cumsum()  # Cumulative Distribution Function is calculated
    cdf_normalized = cdf / cdf.max()
    lut = np.interp(np.arange(256), bins[:-1], cdf_normalized * 255)
    equalized_img = cv.LUT(gray_img, lut) #map the pixels to their equalized values
    hist_equalized, bins_equalized = np.histogram(equalized_img.flatten(), 256, [0,
    plt.figure(figsize=(10,5))
    plt.subplot(1,2,1)
    plt.hist(gray_img.flatten(), 256, [0, 256], color='r')
    plt.title('Original Histogram')
    plt.subplot(1,2,2)
    plt.hist(equalized_img.flatten(), 256, [0, 256], color='g')
    plt.title('Equalized Histogram')
    plt.show()

    return equalized_img

img = cv.imread('../Resources/washed_out_aerial_image.png')
if img is None:
    print('Image could not be read')
    assert False
equalized_img = Hist_function(img)
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))
plt.title('Original Image')
plt.subplot(1,2,2)
plt.imshow(equalized_img, cmap='gray')
plt.title('Equalized Image')
plt.show()
```



In [ ]:
```
#histogram manual method
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
im = cv.imread('../Resources/washed_out_aerial_image.png', cv.IMREAD_GRAYSCALE)
if im is None:
    print('Image could not be read')
    assert False
plt.figure(figsize = [10, 4])
plt.subplot(1, 2, 1)
plt.gca().set_title('Origi')
f = np.zeros(256)
f = [np.sum(im==i) for i in range (256)]
plt.bar(range(256), f)
plt.subplot(1, 2, 2)
plt.gca().set_title('AFTER EQUALIZATION')
im2= cv.equalizeHist(im)
plt.hist(im2)
```

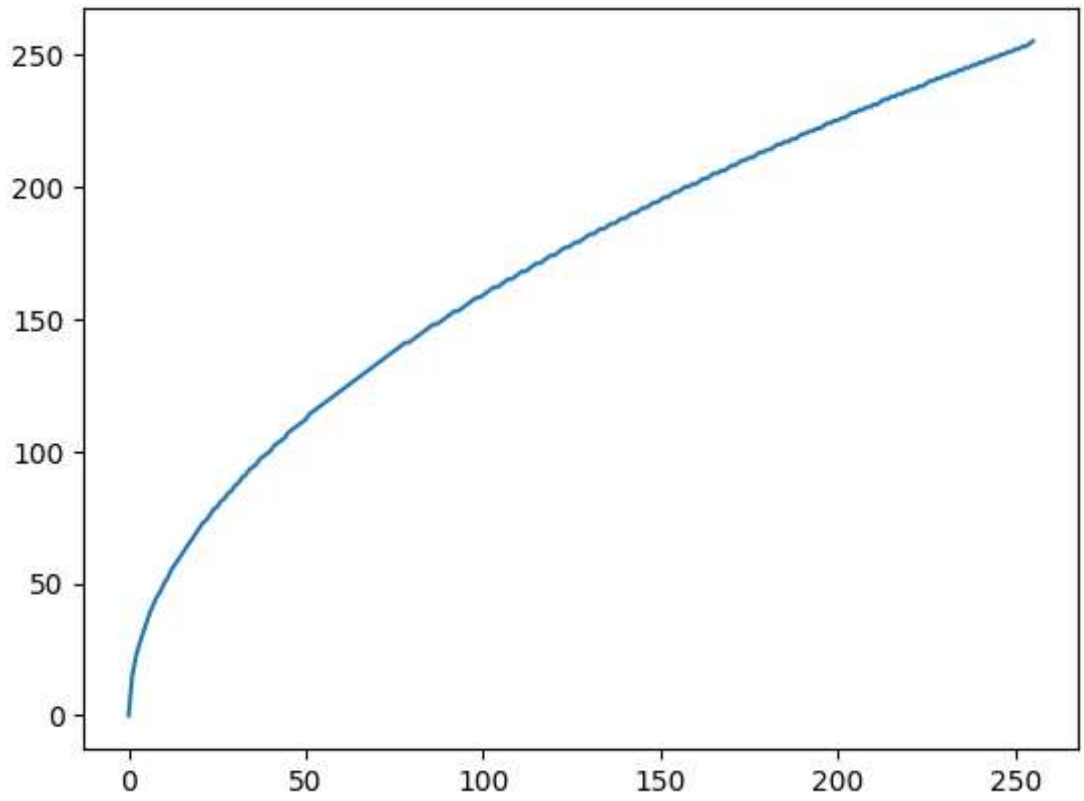
```
plt.show()
fig, ax= plt.subplots(1,2, figsize=(10,20))
ax[0].imshow(im, cmap="gray")
ax[0].set_title('Original Image')
ax[1].imshow(im2, cmap="gray")
ax[1].set_title('After Equalization')
plt.show()
```

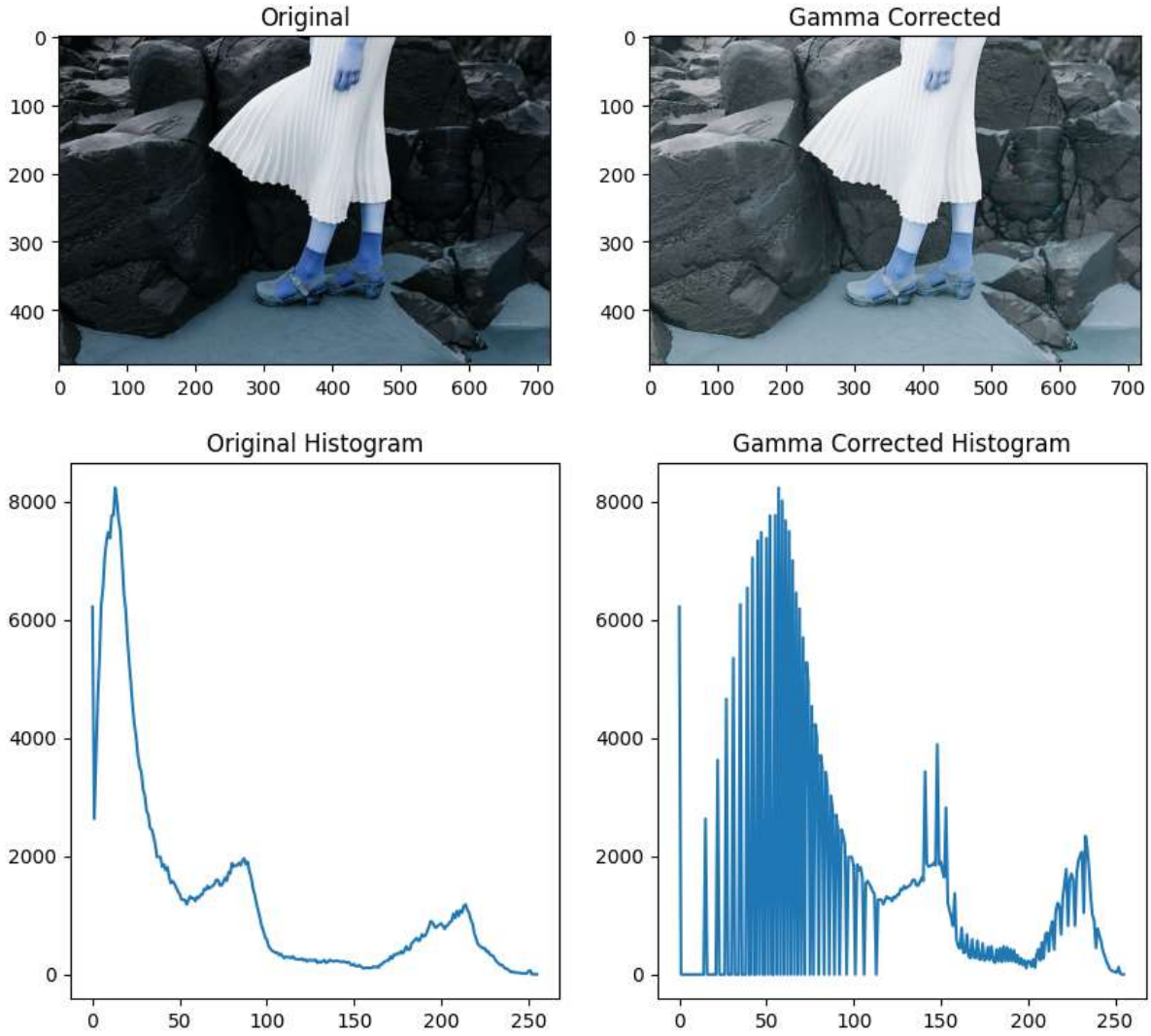


In [ ]:

```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np

im = cv.imread ('../Resources/highlights_and_shadows.jpg', cv.IMREAD_COLOR)
if im is None:
    print('Image could not be read')
    assert False

im_LAB = cv.cvtColor(im, cv.COLOR_BGR2LAB)  # converting image into LAB planes

gamma = 0.5
t = np.array([(i/255.)**gamma*255 for i in range (256)], np.uint8)
g = t[im]
```

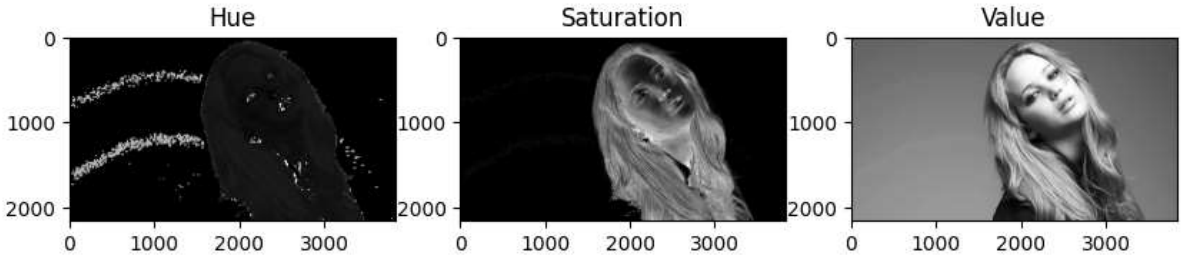
```
plt.plot(t)
plt.show()

fig, ax = plt.subplots(1,2, figsize=(10,20))
ax[0].imshow(im, cmap="gray")
ax[0].set_title("Original")
ax[1].imshow(g,cmap="gray")
ax[1].set_title("Gamma Corrected")
plt.show()

plt.figure(figsize = [10, 5])
plt.subplot(1, 2, 1)
plt.gca().set_title('Original Histogram')
im_h = cv.calcHist([im],[0],None,[256],[0,256])
plt.plot(im_h)
plt.subplot(1, 2, 2)
plt.gca().set_title('Gamma Corrected Histogram')
g_h = cv.calcHist([g],[0],None,[256],[0,256])
plt.plot(g_h)
plt.show()
```

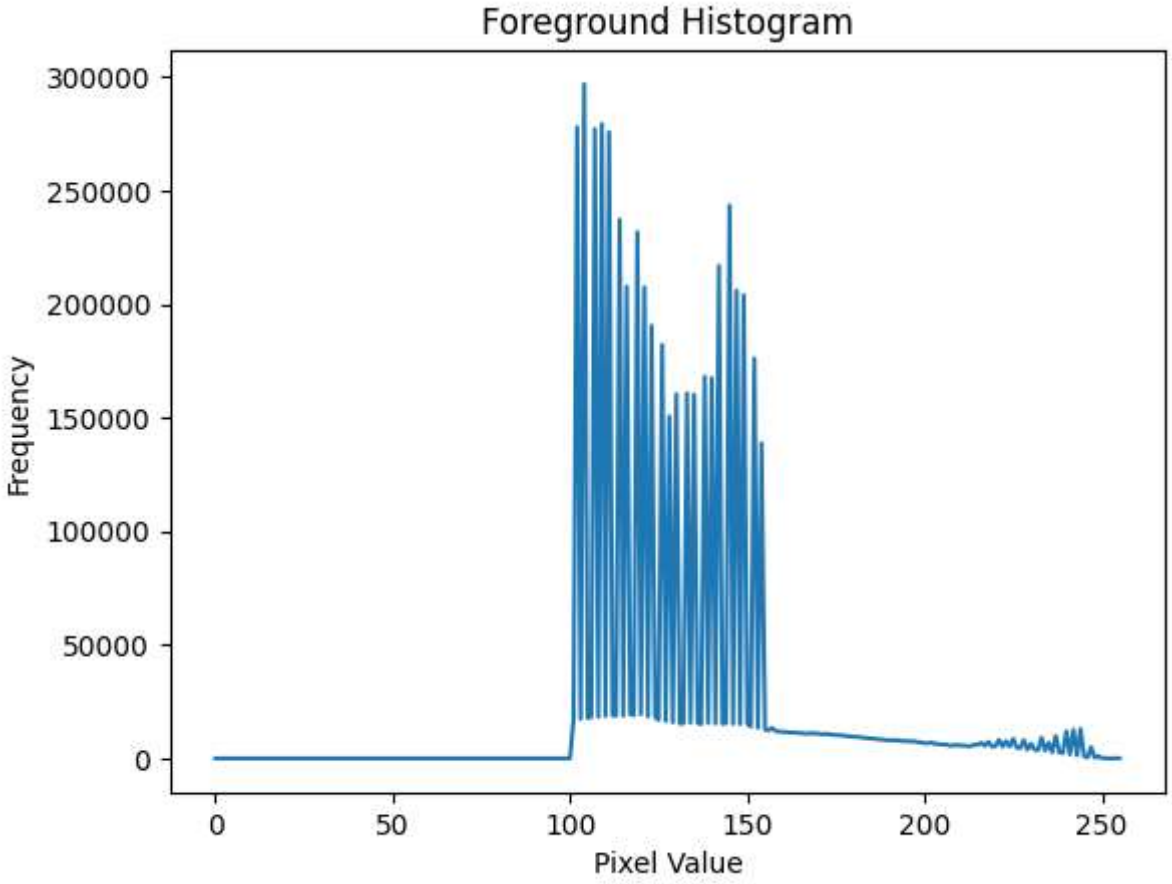


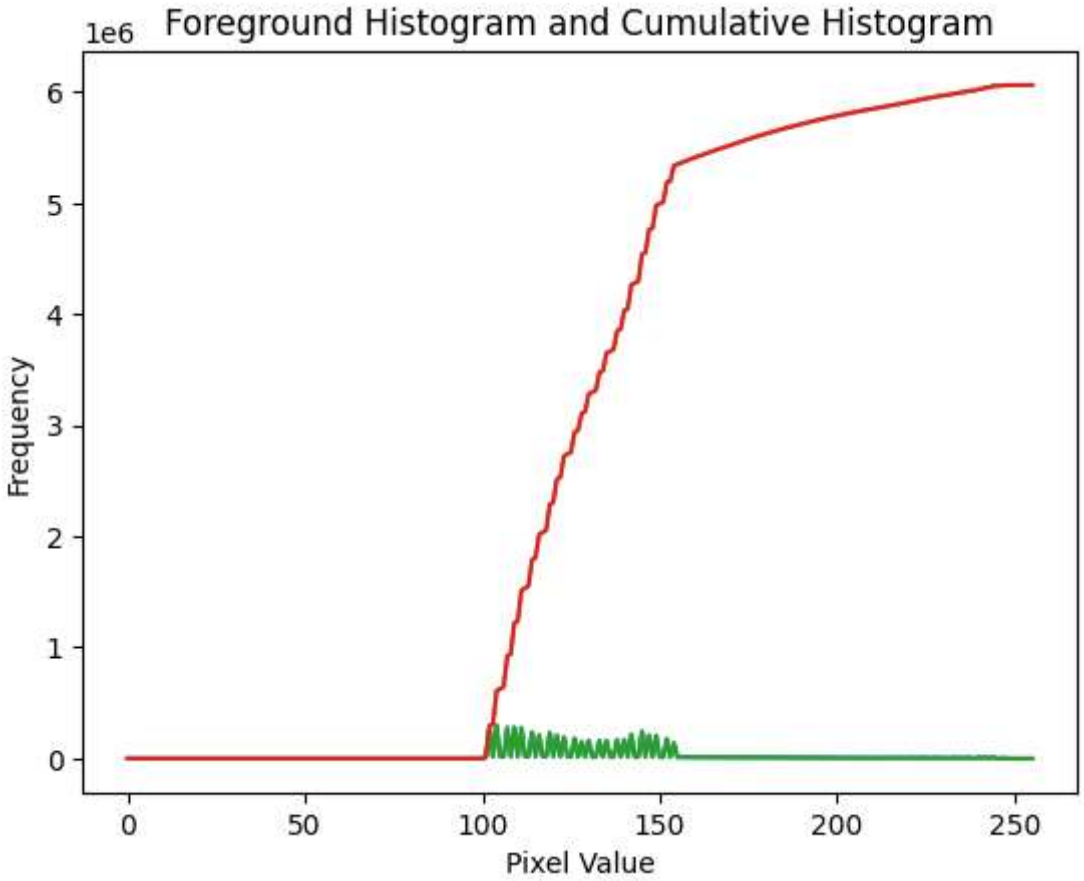


In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np

img = cv.imread('../Resources/jeniffer.jpg')
if img is None:
    print('Image could not be read')
    assert False

img1 = cv.cvtColor(img, cv.COLOR_BGR2HSV)
hue, saturation, value = cv.split(img1)

fig, ax= plt.subplots(1,3, figsize=(10,20))
ax[0].imshow(hue, cmap="gray")
ax[0].set_title('Hue')
ax[1].imshow(saturation, cmap="gray")
ax[1].set_title('Saturation')
ax[2].imshow(value, cmap="gray")
ax[2].set_title('Value')
plt.show()

# value plane can be taken to threshold in extract the foreground mask
```

```
img = cv.imread('../Resources/jeniffer.jpg', cv.IMREAD_GRAYSCALE)
thresh_value = 100 # set a threshold value
_, mask = cv.threshold(img, thresh_value, 255, cv.THRESH_BINARY)
foreground = cv.bitwise_and(img, img, mask=mask) # Apply the mask to the original i
histogram = cv.calcHist([foreground], [0], mask, [256], [0, 256]) # compute the His

# Plot the histogram
plt.plot(histogram)
plt.title('Foreground Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()


cumulative_histogram = np.cumsum(histogram) #obtaining the cumulative sum of the hi
plt.plot(histogram)
plt.plot(cumulative_histogram) # plots both the original histogram and the cumulati

# Plot original and cumulative histogram
cumulative_histogram = np.cumsum(histogram)
plt.plot(histogram)
plt.plot(cumulative_histogram)
plt.title('Foreground Histogram and Cumulative Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()
```

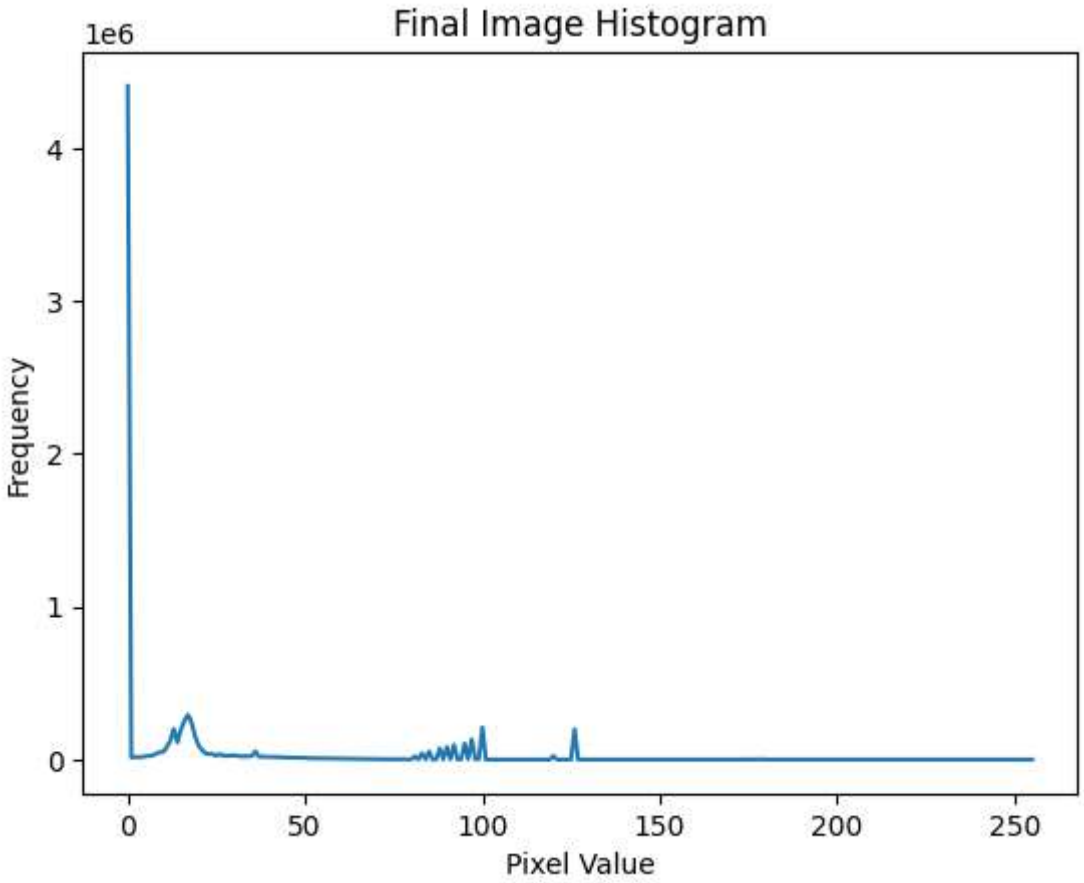


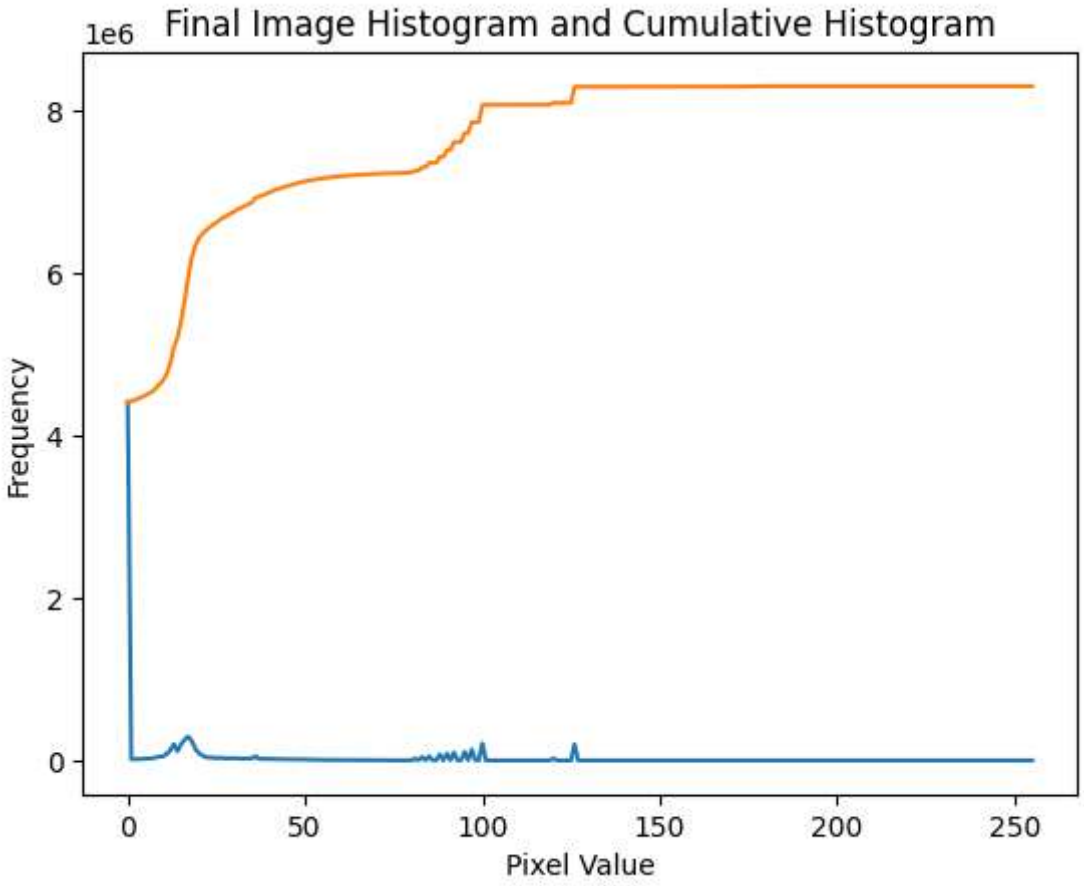

Foreground Histogram
300000
250000
200000
150000
100000
50000
0
Frequency
0
50
100
150
200
250
Pixel Value


Foreground Histogram and Cumulative Histogram
1e6
6
5
4
3
2
1
0
Frequency
0
50
100
150
200
250
Pixel Value

In [ ]:
```
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np

img = cv.imread('../Resources/jeniffer.jpg')
if img is None:
    print('Image could not be read')
    assert False

# Convert to HSV color space and split channels
img_hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)
hue, saturation, value = cv.split(img_hsv)

# Threshold value channel to get foreground mask
thresh_value = 100
_, mask = cv.threshold(value, thresh_value, 255, cv.THRESH_BINARY)

# Invert mask to get background mask
background_mask = cv.bitwise_not(mask)

# Apply mask to original image to get background
background = cv.bitwise_and(img, img, mask=background_mask)

# Equalize value channel of foreground
equalized_value = cv.equalizeHist(value)

# Merge equalized value channel with original hue and saturation channels
equalized_foreground = cv.merge([hue, saturation, equalized_value])

# Apply mask to equalized foreground to get equalized foreground
equalized_foreground = cv.bitwise_and(equalized_foreground, equalized_foreground, m

# Add background and equalized foreground together to get final image
final_image = cv.add(background, equalized_foreground)

# Compute histogram of final image
histogram = cv.calcHist([final_image], [0], None, [256], [0, 256])

# Plot histogram
plt.plot(histogram)
plt.title('Final Image Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()

# Plot original and cumulative histogram
cumulative_histogram = np.cumsum(histogram)
plt.plot(histogram)
plt.plot(cumulative_histogram)
plt.title('Final Image Histogram and Cumulative Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()

# Show final image
```

```
cv.imshow('Final Image', final_image)
cv.waitKey(0)
cv.destroyAllWindows()
```




Final Image Histogram and Cumulative Histogram
1e6
8
6
4
2
0
Frequency
0
50
100
150
200
250
Pixel Value