Assignment 1 :-
Intensity Transformation

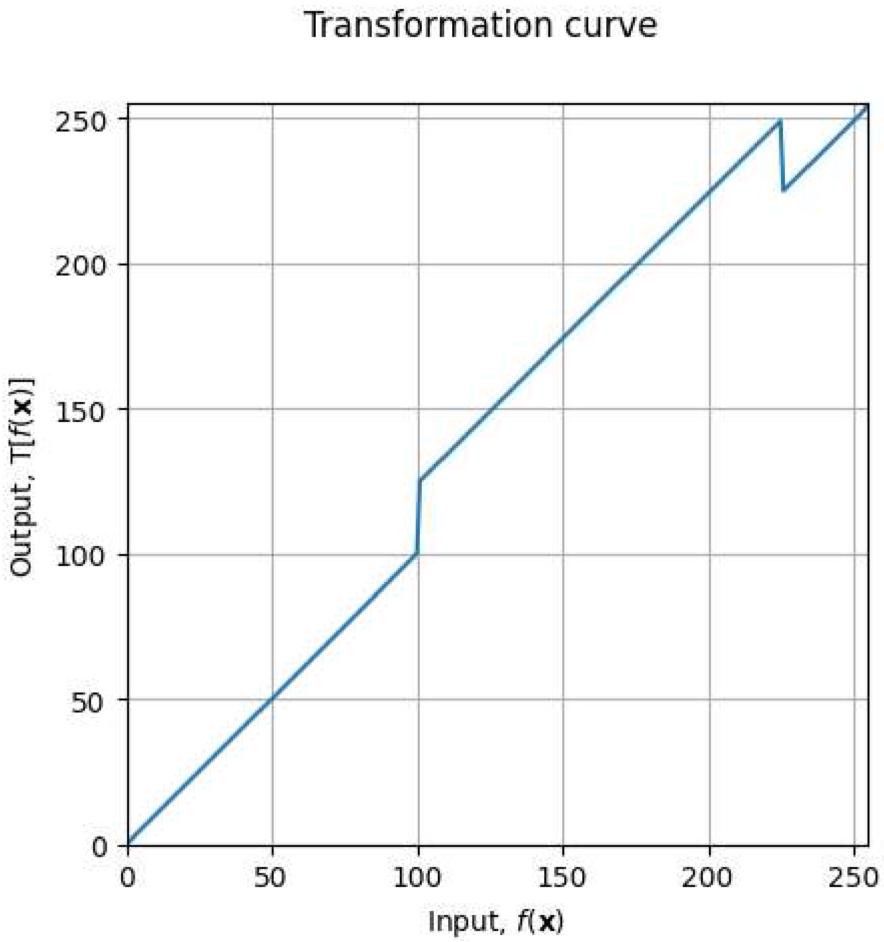
A.U.C PERERA
D/ENG/21/0079/EE

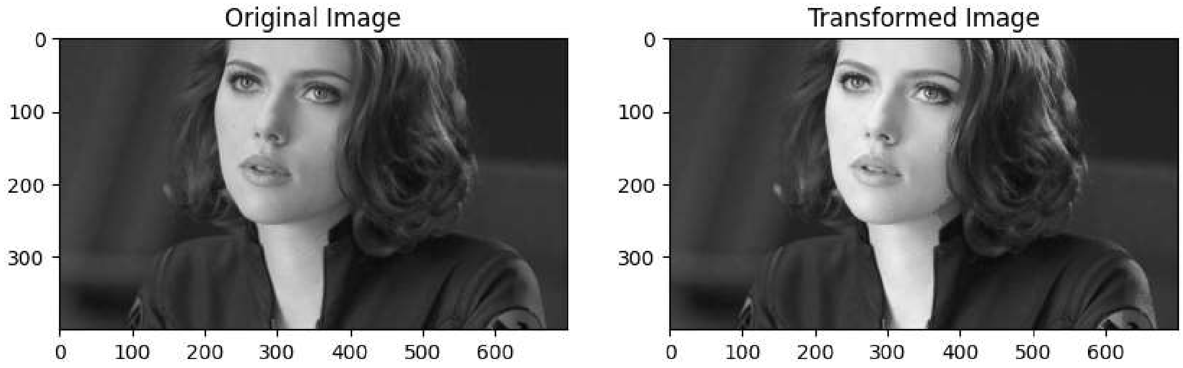
QUESTION 01

In [ ]:

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

im = cv.imread('images/natasha_grayscale.jpg', cv.IMREAD_GRAYSCALE)
assert im is not None

            #Creating regions
#print(len(t1))
t1 = np.linspace(0, 100, 101).astype('uint8')
#print(len(t2))
t2 = np.linspace(125, 249, 125).astype('uint8')
#print(len(t3))
t3 = np.linspace(225, 255,30).astype('uint8')

            #transformation
transform = np.concatenate((t1,t2,t3), axis=0).astype('uint8')
#transform = np.concatenate((transform, t3), axis=0).astype('uint8')
#print(len(transform))

#fig
fig, ax = plt.subplots()
fig.suptitle("Transformation curve")
ax.plot(transform)
ax.set_xlabel(r'Input, $f(\mathbf{x})$')
ax.set_ylabel(r'Output, $\mathrm{T}[f(\mathbf{x})]$')
ax.set_xlim(0,255)
ax.set_ylim(0,255)
ax.set_aspect('equal')
ax.grid()
plt.show()

            #showing image
#cv.namedWindow('Image', cv.WINDOW_NORMAL)
#cv.imshow('Image', im)
#cv.waitKey(0)
image_transform = cv.LUT(im, transform)
#cv.imshow('Image', image_transform)
#cv.waitKey(0)
#cv.destroyAllWindows()

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

```python
ax[1].imshow(image_transform, cmap="gray")
ax[1].set_title("Transformed Image")
plt.show()
```



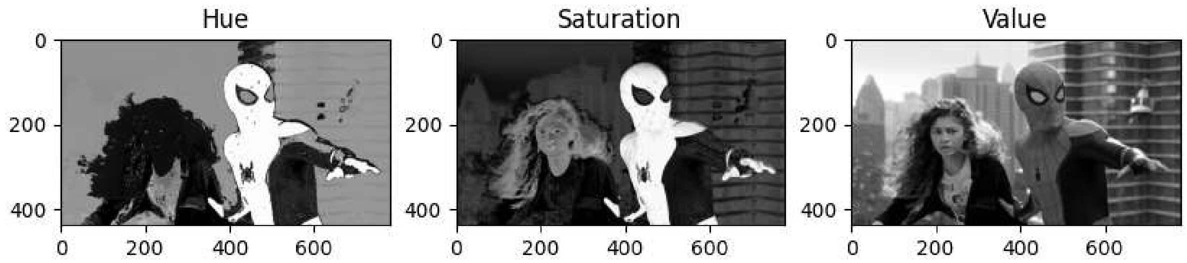


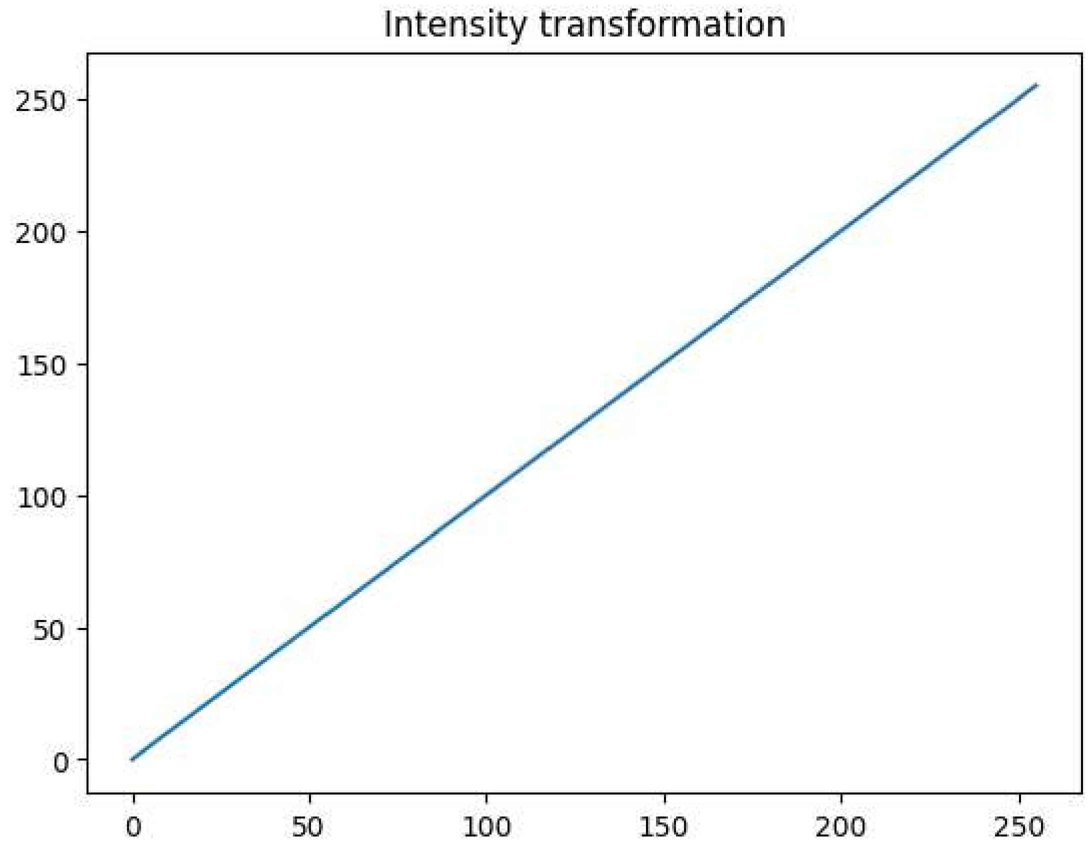
QUESTION 02

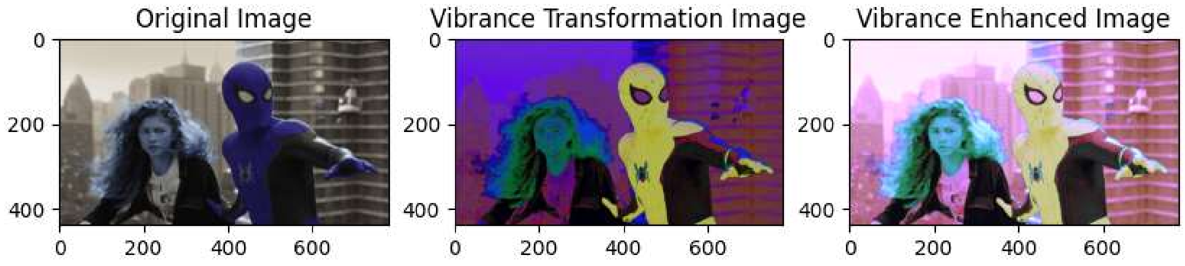
In [ ]:

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

im = cv.imread('images/spider.png', cv.IMREAD_COLOR)
assert im is not None

im1 = cv.cvtColor(im, cv.COLOR_BGR2HSV)
h_img,s_img,v_img = cv.split(im1)
```

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

x= np.arange(0, 256).astype('uint8')
a = .1
sigma = 70
Y = np.minimum((((x)+(a*(np.exp(-(x-128)**2/(2*sigma**2))))/128), 255).astype('uint8
image_transform = cv.LUT(s_img, Y)
plt.title('Intensity transformation')
plt.plot(Y)
plt.show()

newHSV = cv.merge([h_img,image_transform,v_img])
result =  cv.cvtColor(newHSV,  cv.COLOR_HSV2BGR)
added_img = cv.add(newHSV, im)

#Y2 = (a/128)*Y1
#Y = np.add(Y, Y1)
fig, ax= plt.subplots(1,3, figsize=(10,20))
ax[0].imshow(im, cmap="gray")
ax[0].set_title('Original Image')
ax[1].imshow(newHSV, cmap="gray")
ax[1].set_title('Vibrance Transformation Image')
ax[2].imshow(added_img, cmap="gray")
ax[2].set_title('Vibrance Enhanced Image')
plt.show()
```







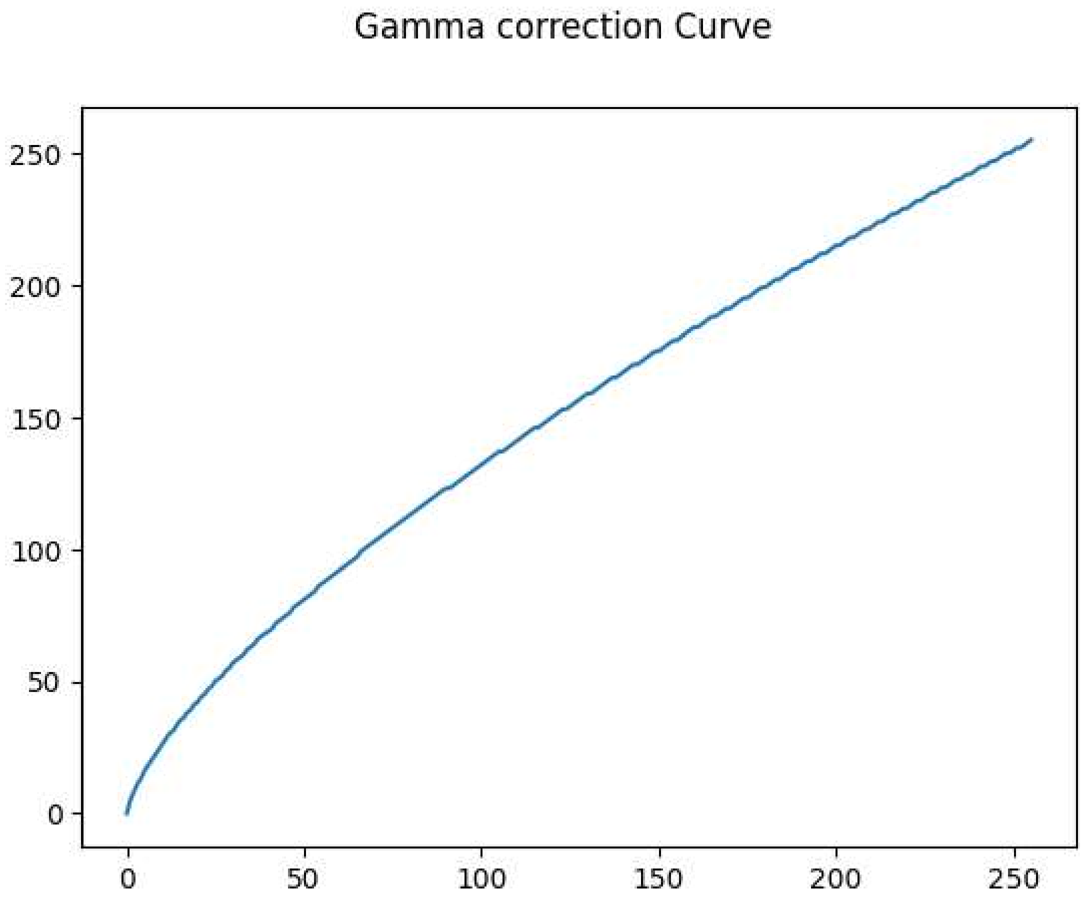
QUESTION 03

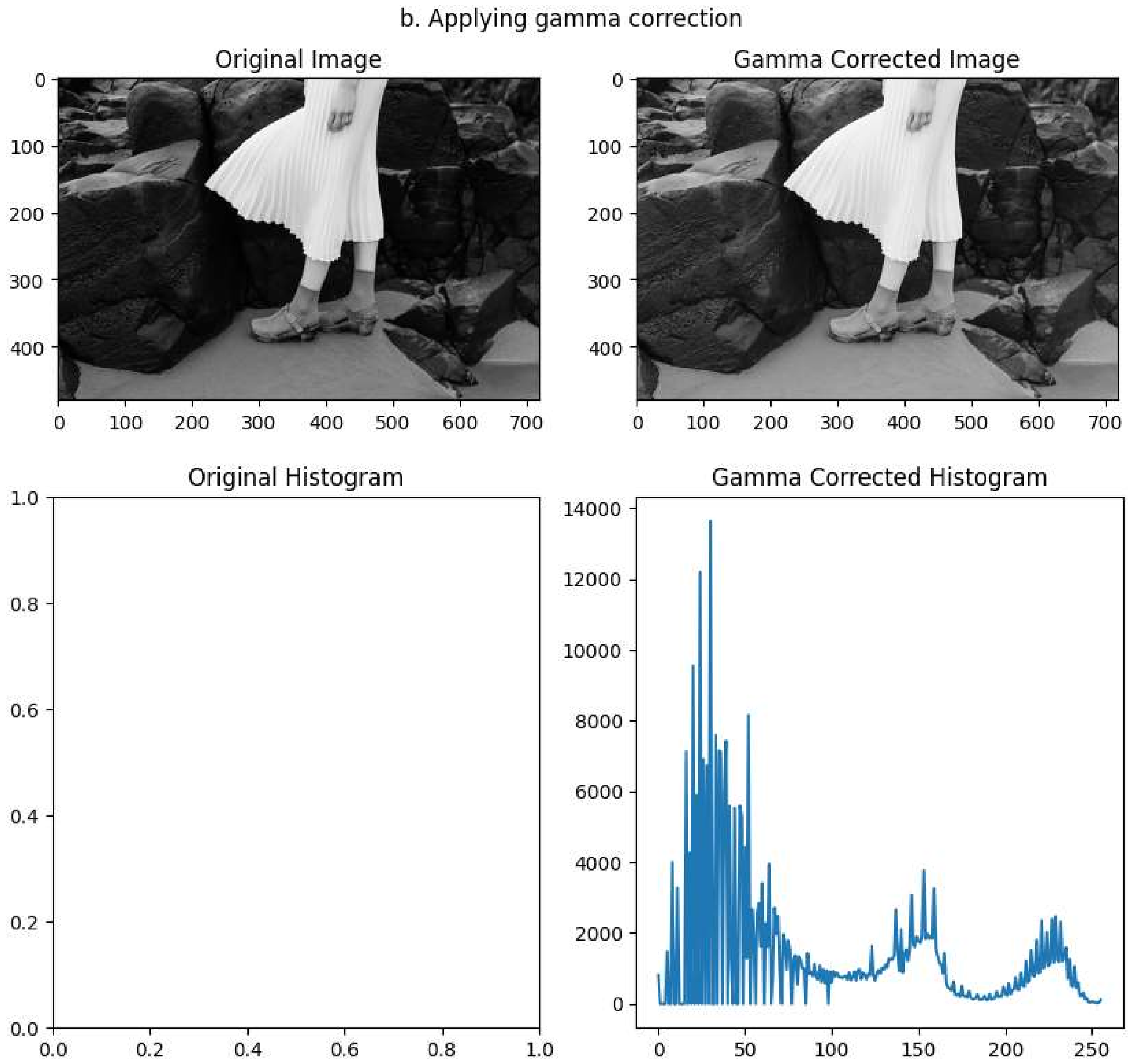
The value of γ = 0.6

In [ ]:

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

im = cv.imread ('images/highlights_and_shadows.jpg', cv.IMREAD_COLOR)
assert im is not None

im_LAB = cv.cvtColor(im, cv.COLOR_BGR2LAB)
L_im, q_im , r_im = cv.split(im_LAB)

gamma = .7
t = np.array([(i/255.)**gamma*255 for i in range (256)], np.uint8)
g = t[L_im]
plt.suptitle("Gamma correction Curve")
plt.plot(t)
plt.show()
```

```
fig, ax = plt.subplots(1,2, figsize=(10,3.5))
fig.suptitle("b. Applying gamma correction")
ax[0].imshow(L_im, cmap="gray")
ax[0].set_title("Original Image")
ax[1].imshow(g,cmap="gray")
ax[1].set_title("Gamma Corrected Image")
plt.show()

plt.figure(figsize = [10, 5])
#Histogram of the original image
plt.subplot(1, 2, 1)
plt.gca().set_title('Original Histogram')
im_h = cv.calcHist([L_im],[0],None,[256],[0,256])
#Histogram of the corrected image
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()
```



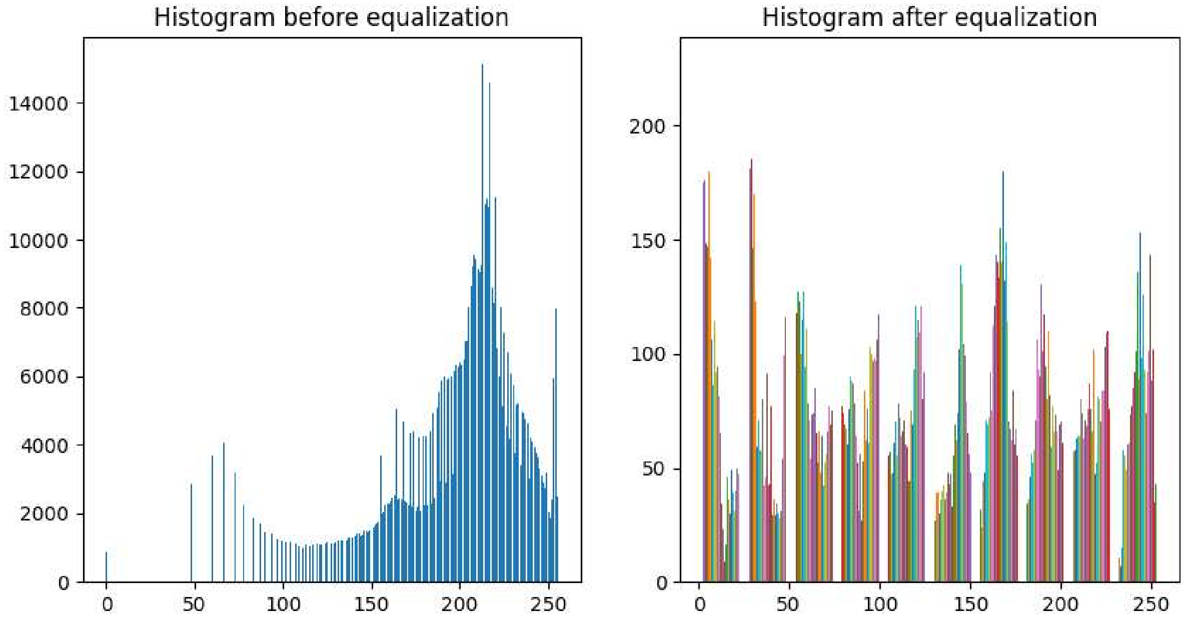


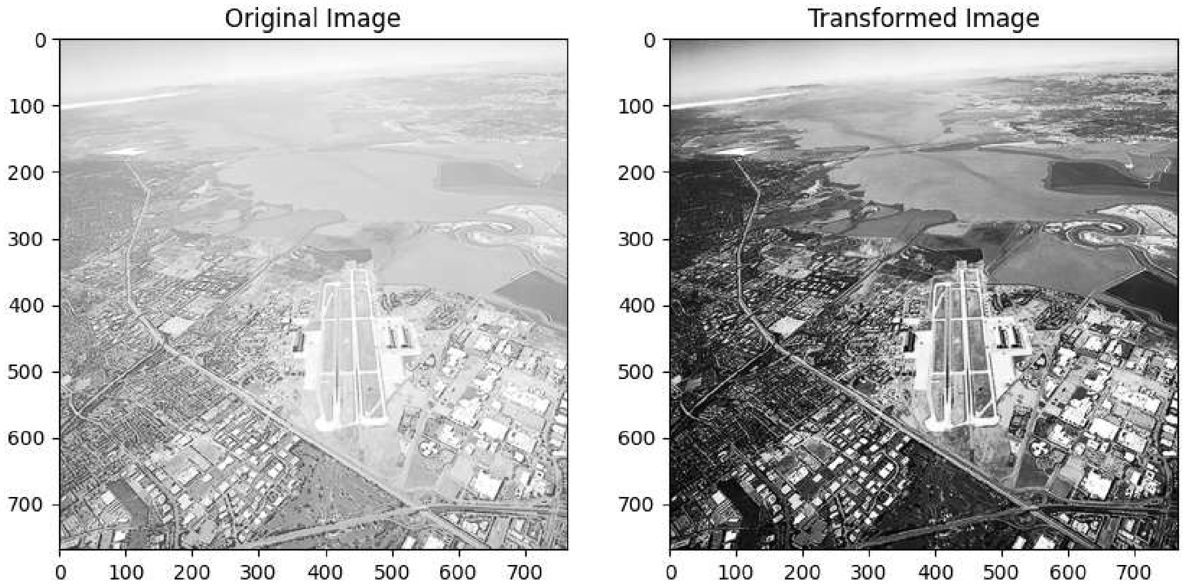
QUESTION 04

In [ ]:

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

im = cv.imread('images/washed_out_aerial_image.png', cv.IMREAD_GRAYSCALE)
assert im is not None

plt.figure(figsize = [10, 5])
plt.subplot(1, 2, 1)
plt.gca().set_title('Histogram before equalization')
h = np.zeros(256)
h = [np.sum(im==i) for i in range (256)]
plt.bar(range(256), h)

plt.subplot(1, 2, 2)
plt.gca().set_title('Histogram after equalization')
eh = cv.equalizeHist(im)
plt.hist(eh)
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(eh, cmap="gray")
ax[1].set_title('Transformed Image')
plt.show()
```





QUESTION 05

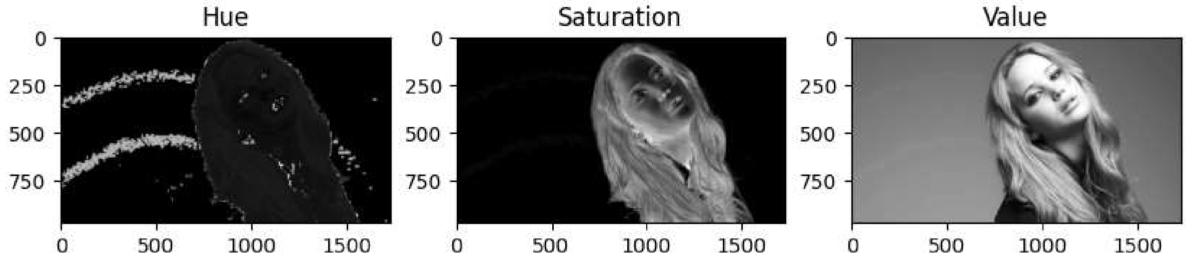
In [ ]:

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

im = cv.imread('images/jeniffer.jpg', cv.IMREAD_COLOR)
assert im is not None

im1 = cv.cvtColor(im, cv.COLOR_BGR2HSV)
```

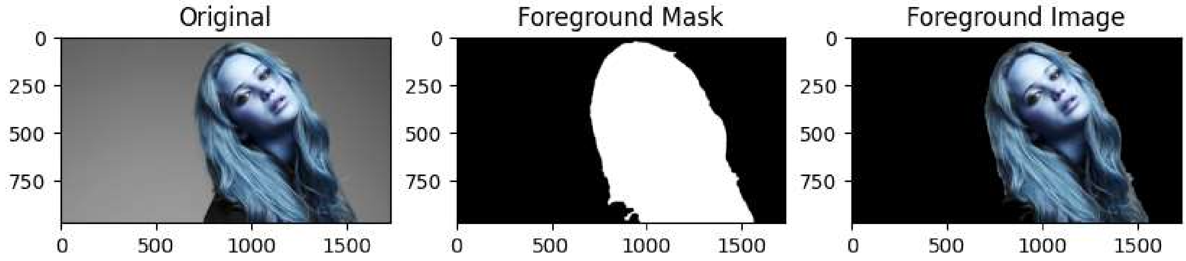
```
h_img,s_img,v_img = cv.split(im1)

fig, ax= plt.subplots(1,3, figsize=(10,2.5))
fig.suptitle("a. Splitting into HSV regions", fontsize=12)
ax[0].imshow(h_img, cmap="gray")
ax[0].set_title('Hue')
ax[1].imshow(s_img, cmap="gray")
ax[1].set_title('Saturation')
ax[2].imshow(v_img, cmap="gray")
ax[2].set_title('Value')
plt.show()

            #Foreground image extractor
#lower = np.array([200, 200, 200])
#upper = np.array([255, 255, 255])
thresh = cv.inRange(s_img, 15, 230)
kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE, (20,20))
morph = cv.morphologyEx(thresh, cv.MORPH_CLOSE, kernel)
mask = morph
result = cv.bitwise_and(im, im, mask=mask)

fig, ax = plt.subplots(1,3, figsize=(10,2.5))
fig.suptitle("b. Extracting Foreground mask")
ax[0].imshow(im, cmap="gray")
ax[0].set_title("Original")
ax[1].imshow(mask, cmap="gray")
ax[1].set_title("Foreground Mask")
ax[2].imshow(result, cmap="gray")
ax[2].set_title("Foreground Image")
plt.show()

            #Histogram equalizing the fore ground
plt.figure(figsize = [10, 2.5])
plt.subplot(1, 2, 1)
plt.gca().set_title('Original Histogram of foreground')
fg_h = cv.calcHist([result],[0],None,[256],[0,256]) #histogram of image
plt.plot(fg_h)

plt.subplot(1, 2, 2)
plt.gca().set_title('Corrected Histogram')
result1 = cv.cvtColor(result, cv.COLOR_BGR2GRAY)
eh = cv.equalizeHist(result1) #Equalizing histogram
eh1 = cv.calcHist([eh], [0], None, [256], [0,256])
plt.plot(eh1)
plt.show()

            #cumulative sum
cumulative_sum = np.cumsum(eh)
plt.plot(cumulative_sum)
plt.title("cumulative_sum")
plt.show()

            #background image
mask1 = 255 - morph
bg_img = cv.bitwise_and(im, im, mask=mask1)
bg_img1 = cv.cvtColor(bg_img, cv.COLOR_BGR2GRAY);
```

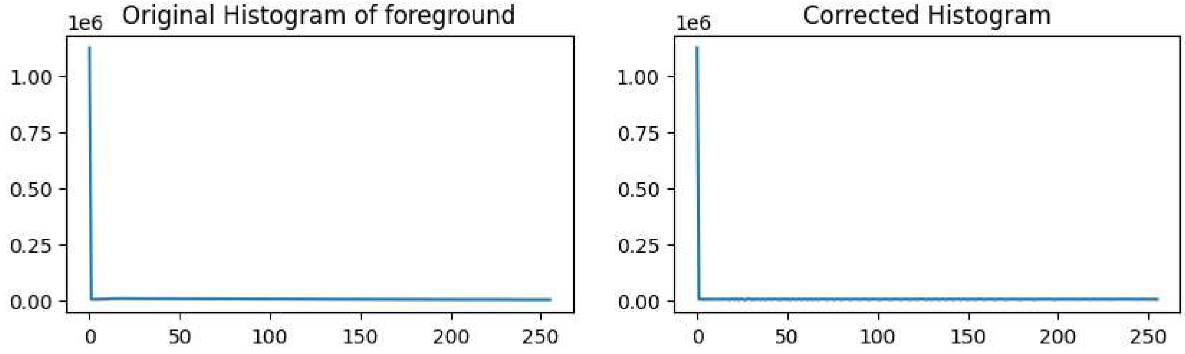
```
            #added image
img1 = cv.add(bg_img1,eh)

fig, ax = plt.subplots(1,3, figsize=(10,2.5))
fig.suptitle("f. Adding background with equalized")
ax[0].imshow(bg_img, cmap="gray")
ax[0].set_title("Background")
ax[1].imshow(eh, cmap="gray")
ax[1].set_title("Foreground")
ax[2].imshow(img1, cmap="gray")
ax[2].set_title("Added Image")
plt.show()
```

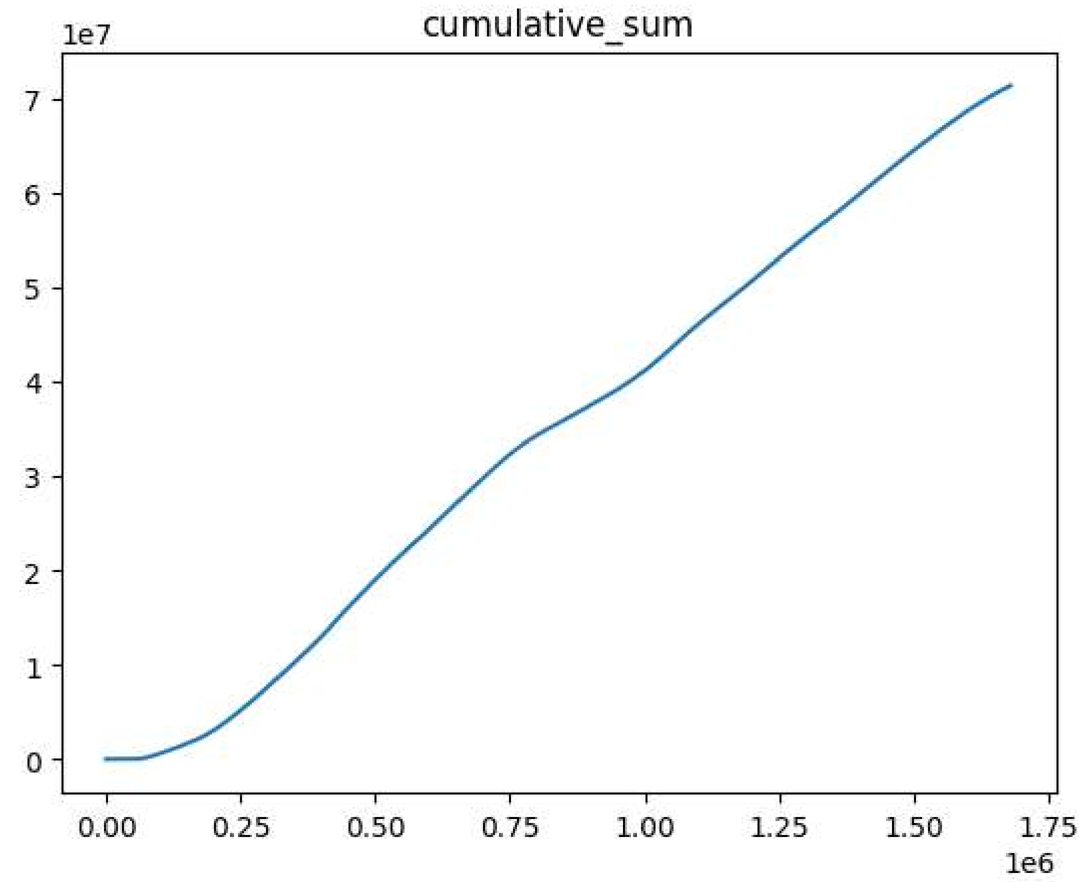
a. Splitting into HSV regions



b. Extracting Foreground mask







f. Adding background with equalized