EN3160 – Assignment 1 on Intensity Transformations and Neighborhood Filtering

Name – De Silva A.L.U.P.

Index – 200105F

GitHub Link –

Question 01

```
c = np.array([(50,50),(50,100),(150,255),(150,150)])
t1 = np.linspace(0, c[0,1], c[0,0]+1-0).astype('uint8')
t2 = np.linspace(c[1,1], c[2,1], c[2,0] - c[1,0]).astype('uint8')
t3 = np.linspace(c[3,1], 255, 255 - c[3,0]).astype('uint8')

transform = np.concatenate((t1, t2), axis=0).astype('uint8')
transform = np.concatenate((transform, t3), axis=0).astype('uint8')
assert len(transform) == 256
```

Figure 1 - Generating the Transform

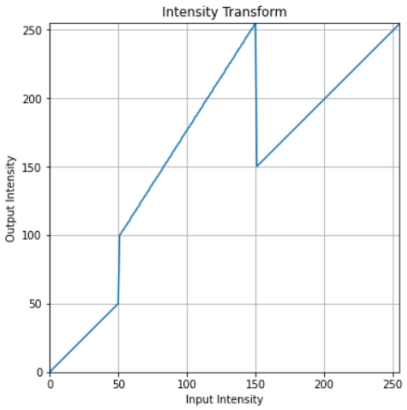

Figure 2 - Intensity Transformation

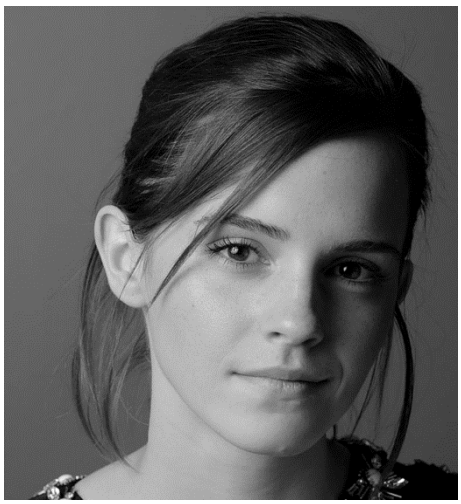

Figure 3(a) - Image before applying transformation

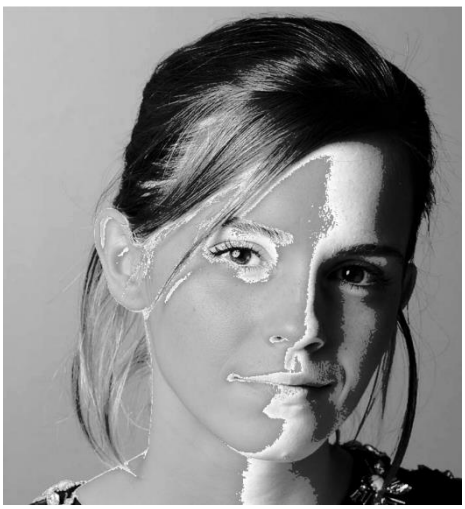

Figure 3(b) - Image after applying transformation

Question 02

```python
lower_thresh = 180
upper_thresh = 255
c = np.array([(0,lower_thresh),(lower_thresh,upper_thresh)])
t1 = np.linspace(0, c[0,0], c[0,1]+1-0).astype('uint8')
t2 = np.linspace(c[1,1], 255, 255 - c[1,0]).astype('uint8')

transform = np.concatenate((t1, t2), axis=0).astype('uint8')
assert len(transform) == 256
```

Figure 4(a) - Generating transformation for white matter

```python
lower_thresh = 100
upper_thresh = 180
c = np.array([(0,lower_thresh),(lower_thresh,255),(upper_thresh,255)])
t1 = np.linspace(0, c[0,0], c[0,1]+1-0).astype('uint8')
t2 = np.linspace(c[1,1],c[2,1],c[2,0] - c[1,0] ).astype('uint8')
t3 = np.linspace(0, 0, 255 - c[2,0]).astype('uint8')

transform = np.concatenate((t1, t2), axis=0).astype('uint8')
transform = np.concatenate((transform, t3), axis=0).astype('uint8')
assert len(transform) == 256
```

Figure 4(b) - Generating transformation for gray matter

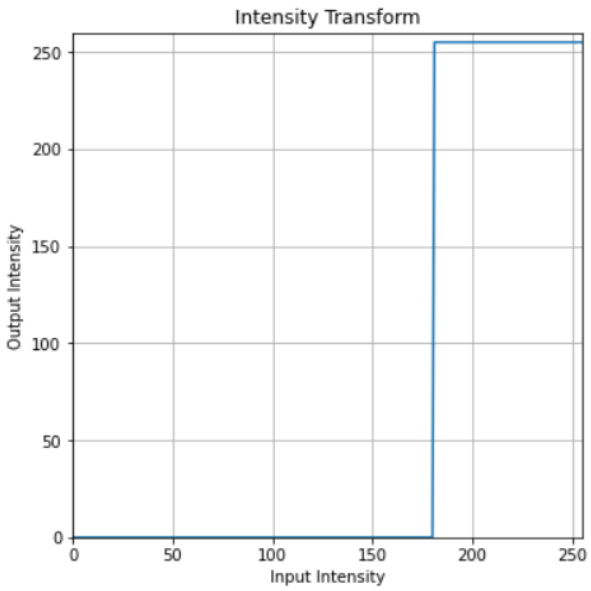

Figure 5(a) - Intensity transformation for white matter

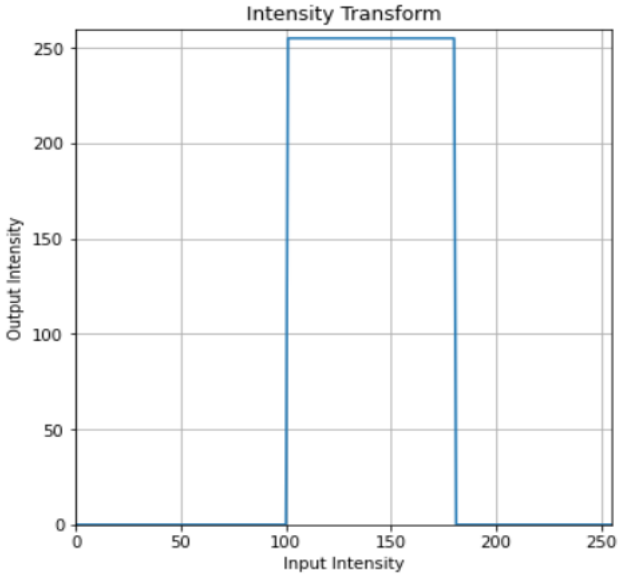

Figure 5(b) - Intensity transformation for gray matter

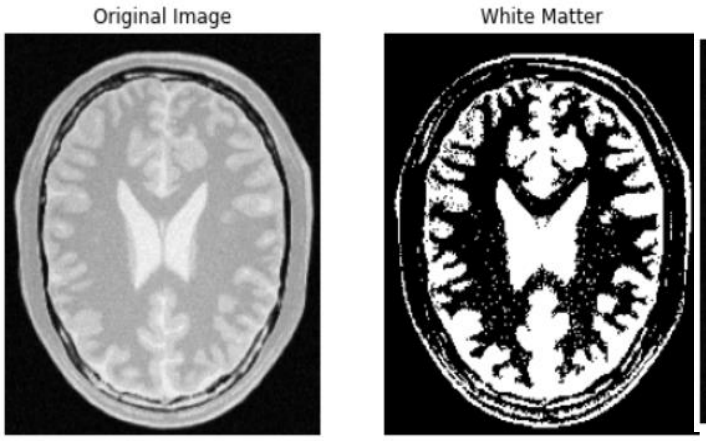

Figure 7(a) - Accentuated white matter

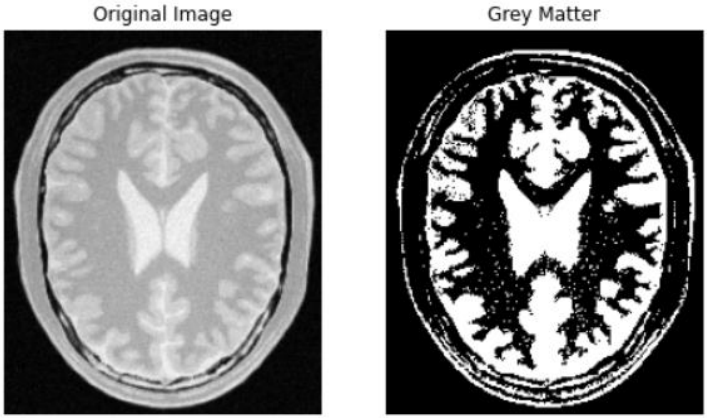

Figure 6(b) - Accentuated gray matter

Question 03

```
L, a, b = cv.split(cv.cvtColor(img, cv.COLOR_BGR2LAB))
gamma = [0.2, 0.8, 1.2, 2]

for i in gamma:
    t = np.array([(p/255)**i*255 for p in range(0,256)]).astype(np.uint8)
    g = cv.LUT(L, t)

    corrected_img = cv.merge([g, a, b])

    hist1 = cv.calcHist([img], [0], None, [256], [0, 256])
    hist2 = cv.calcHist([corrected_img], [0], None, [256], [0, 256])
```

Figure 8 - Gamma correction and generating histograms

Figure 9(a) - Gamma = 0.2

Figure 9(b) - Gamma = 0.8

Figure 10(c) - Gamma = 1.2

Figure 9(d) - Gamma = 2

Question 04

(a) image_hsv = cv.cvtColor(image_bgr, cv.COLOR_BGR2HSV)

h, saturation_plane, v = cv.split(image_hsv)

(b) def intensity_transform(x, a, sigma=70):
f_x = np.clip(x+a*128*np.exp(-(x-128)**2/(2*sigma**2)), 0, 255)
return f_x

Apply the intensity transform function

transforemed_saturation_plane = intensity_transform(saturation_plane, 0.4)

- Visually Pleasing results were obtained for values in the [0.3, 0.4] range.

(d) image_copy = image_hsv.copy()

image_copy[:, :, 1] = transforemed_saturation_plane

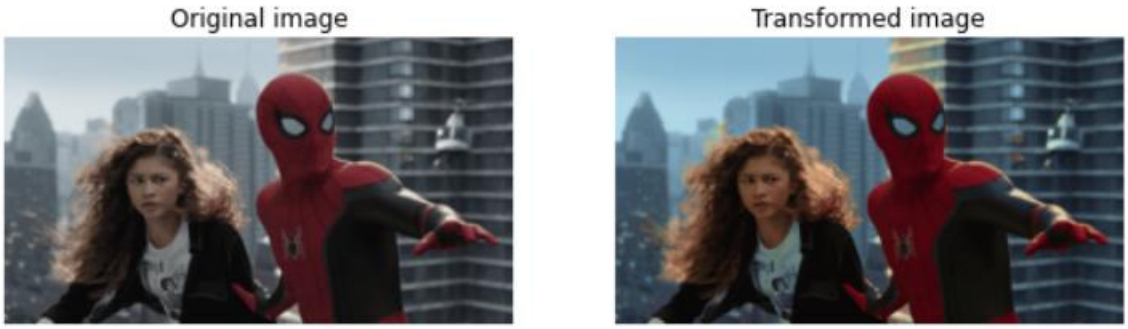

Figure 11 - Original and Enhanced images

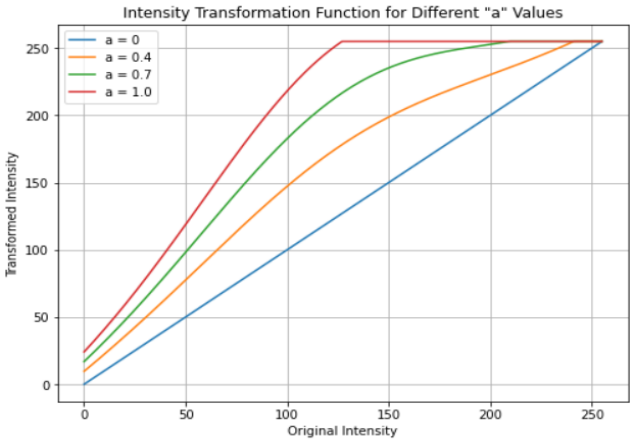

Figure 12 - Intensity Transformations

intensity_values = np.arange(256)

a_values = [0, 0.4, 0.7, 1.0]

for a in a_values:

transformation_values = intensity_transform(intensity_values, a)

Question 05

```
def histogram_equalization(im):
    cv.imread(im,cv.IMREAD_GRAYSCALE)

    # Calculate the histogram of the image
    histogram = np.zeros(256, dtype=int)
    for pixel_value in img.flat:
        histogram[pixel_value] += 1

    # Calculate the cumulative distribution function (CDF)
    cdf = np.zeros(256, dtype=int)
    cdf[0] = histogram[0]
    for i in range(1, 256):
        cdf[i] = cdf[i - 1] + histogram[i]

    # Perform histogram equalization
    num_pixels = img.size
    equalized_image = np.zeros_like(img)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            pixel_value = img[i, j]
            equalized_pixel = int((cdf[pixel_value] / num_pixels) * 255)
            equalized_image[i, j] = equalized_pixel

    # Calculate the histogram of the equalized image
    histogram_equalized = np.zeros(256, dtype=int)
    for pixel_value in equalized_image.flat:
        histogram_equalized[pixel_value] += 1
```

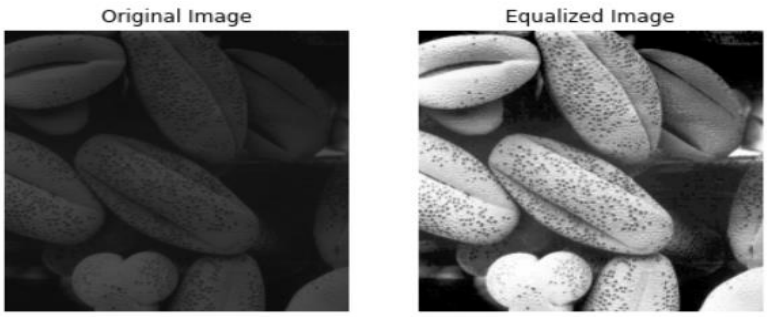

Figure 13 - Images before and after equalization

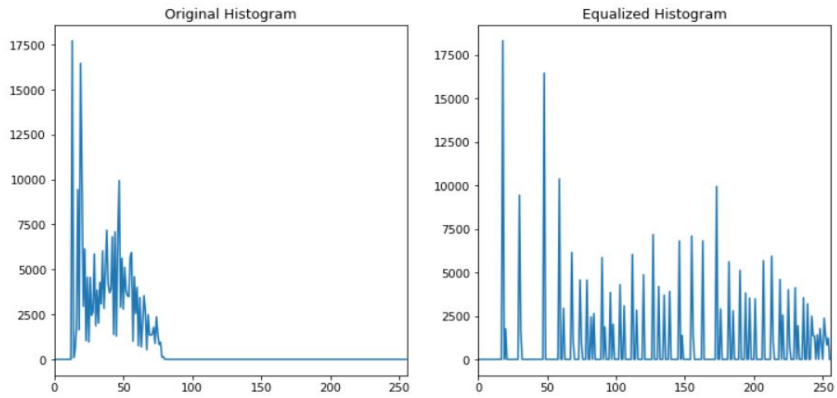

Figure 14 - Histogram before and after equalization

Question 06

(a) hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)
hue, saturation, value = cv.split(hsv_image)

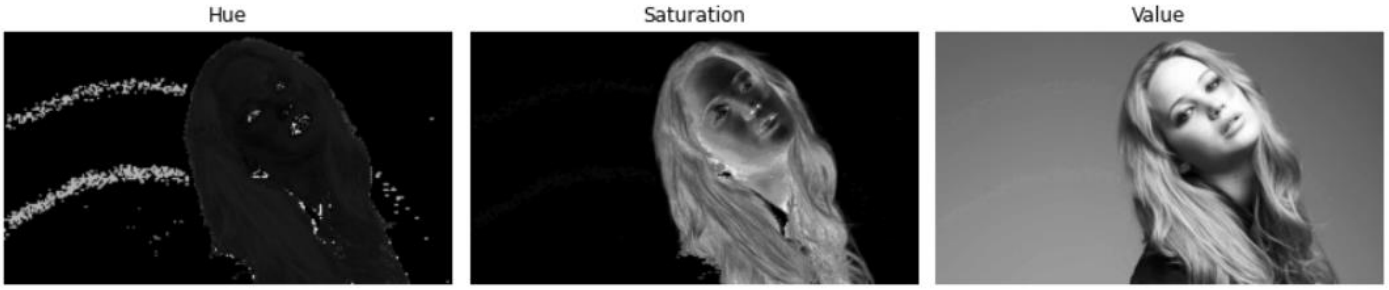

Figure 15

(b) The foreground object is highly saturated compared to the background. Therefore, the Saturation plane is the appropriate plane for the threshold in extracting the foreground mask.

```
(c)saturation_min = 15
   saturation_max = 255
   foreground_mask = cv.inRange(saturation, saturation_min, saturation_max)
   foreground_mask = cv.morphologyEx(foreground_mask, cv.MORPH_CLOSE,
cv.getStructuringElement(cv.MORPH_ELLIPSE,(80, 80)))
   foreground = cv.bitwise_and(image, image, mask=foreground_mask)
   histogram = cv.calcHist([foreground], [0], foreground_mask, [256], [0, 256])
```