EN3160 – Assignment 1 on Intensity Transformations and Neighborhood Filtering

Name – De Silva A.L.U.P.

Index – 200105F

GitHub Link –

Question 01

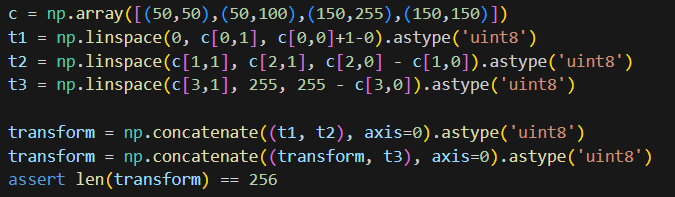


Figure 1 - Generating the Transform

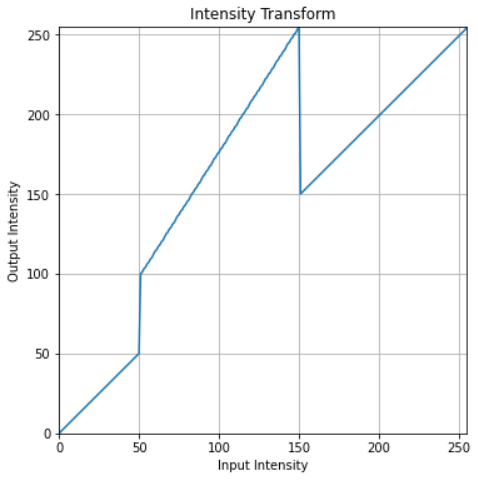


Figure 2 - Intensity Transformation



Figure 3(b) - Image after applying transformation



Figure 3(a) - Image before applying transformation

Question 02

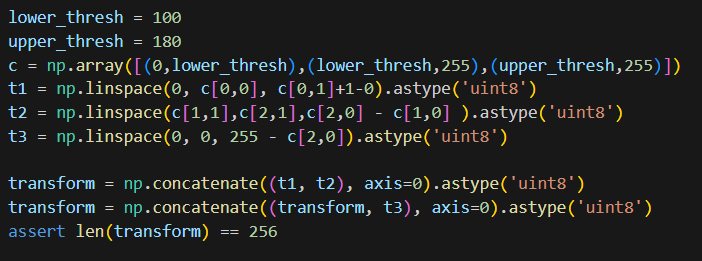


Figure 4(b) - Generating transformation for gray matter

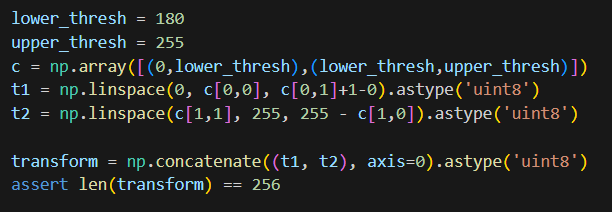


Figure 4(a) - Generating transformation for white matter

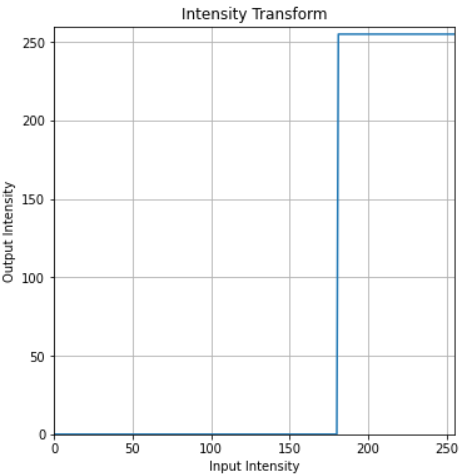


Figure 5(a) - Intensity transformation for white matter

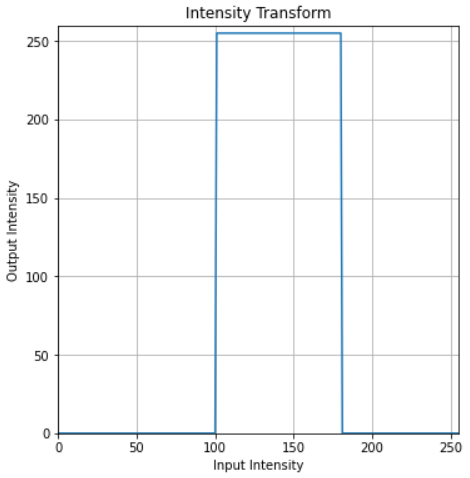


Figure 5(b) - Intensity transformation for gray matter

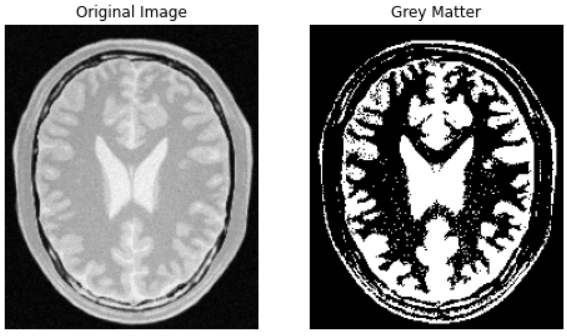


Figure 6(b) - Accentuated gray matter

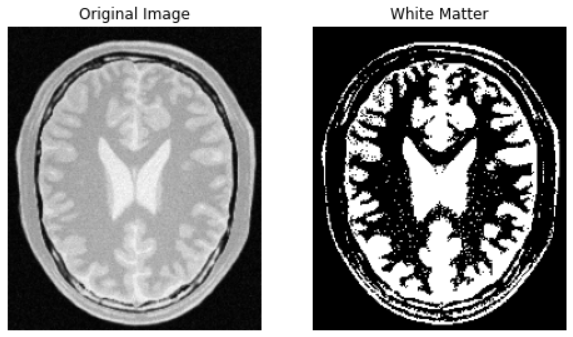


Figure 7(a) - Accentuated white matter

Question 03

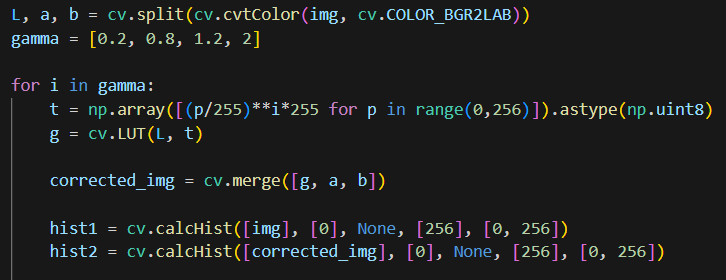


Figure 8 - Gamma correction and generating histograms

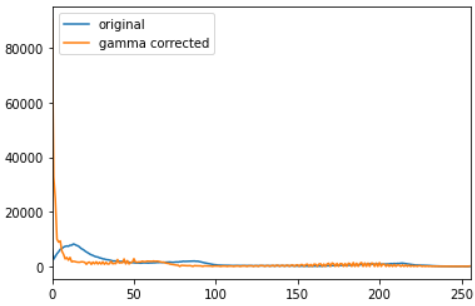
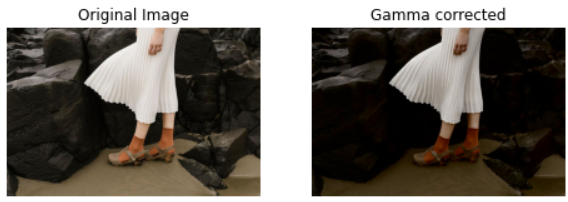


Figure 9(d) - Gamma = 2

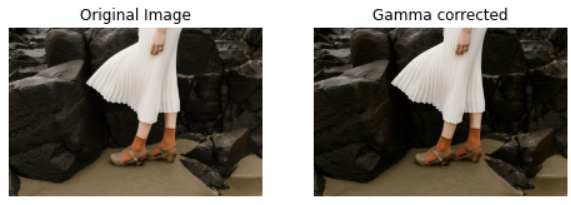
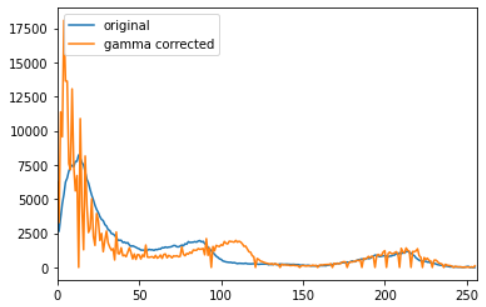


Figure 10(c) - Gamma = 1.2

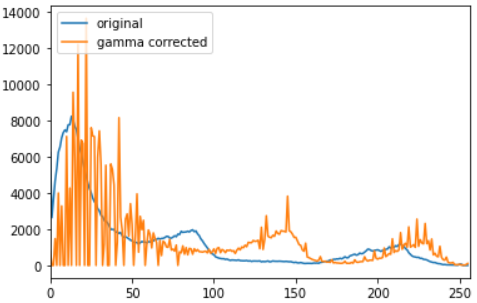
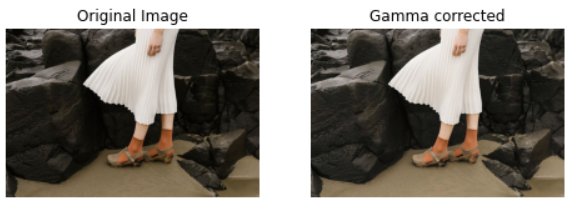


Figure 9(b) - Gamma = 0.8

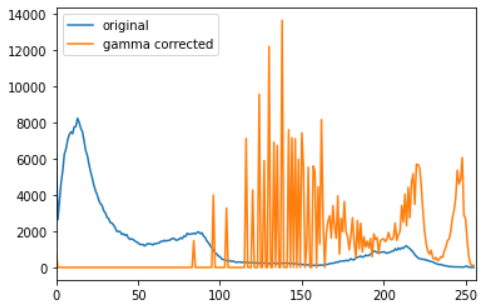
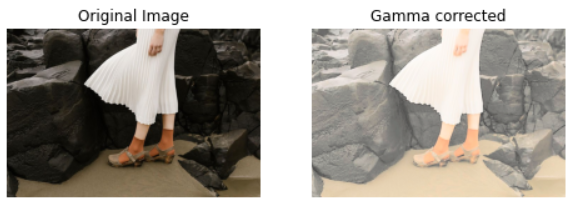


Figure 9(a) - Gamma = 0.2

Question 04

1. image\_hsv = cv.cvtColor(image\_bgr, cv.COLOR\_BGR2HSV)

h, saturation\_plane, v = cv.split(image\_hsv)

1. 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

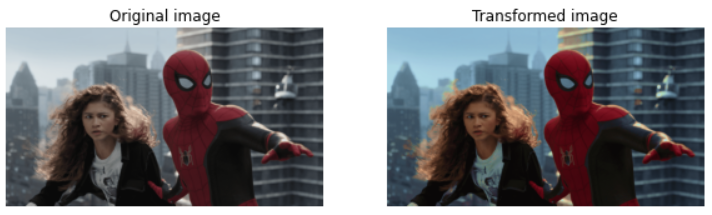


Figure 11 - Original and Enhanced images

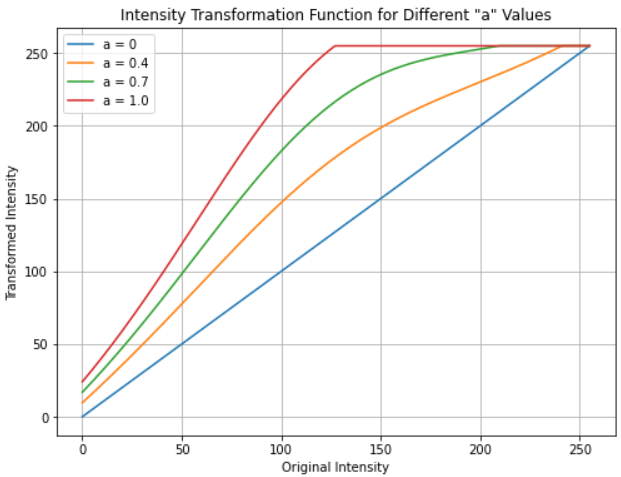


Figure 12 - Intensity Transformations

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)

    histogram = np.zeros(256, dtype=int)

    for pixel\_value in img.flat:

        histogram[pixel\_value] += 1

    cdf = np.zeros(256, dtype=int)

    cdf[0] = histogram[0]

    for i in range(1, 256):

        cdf[i] = cdf[i - 1] + histogram[i]

    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

    histogram\_equalized = np.zeros(256, dtype=int)

    for pixel\_value in equalized\_image.flat:

        histogram\_equalized[pixel\_value] += 1

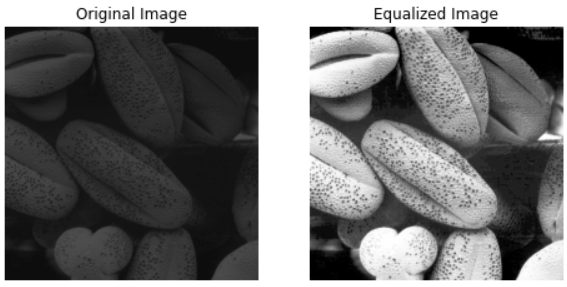


Figure 13 - Images before and after equalization

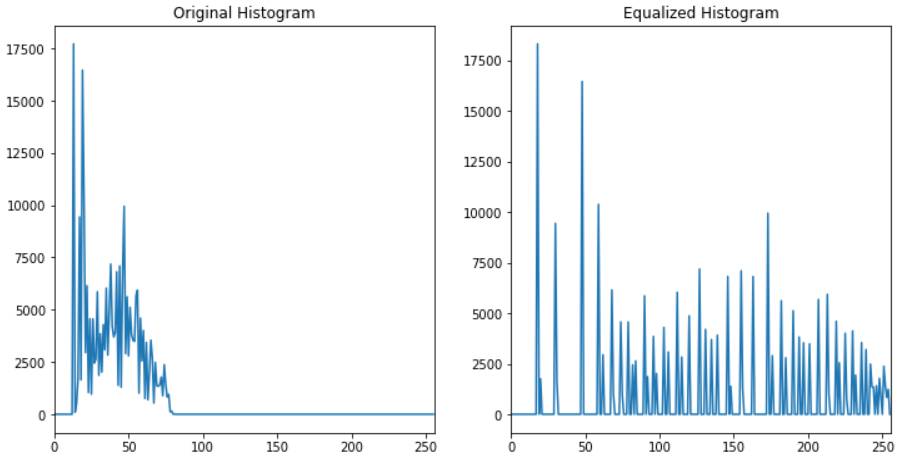


Figure 14 - Histogram before and after equalization

Question 06

1. hsv\_image = cv.cvtColor(image, cv.COLOR\_BGR2HSV)

hue, saturation, value = cv.split(hsv\_image)

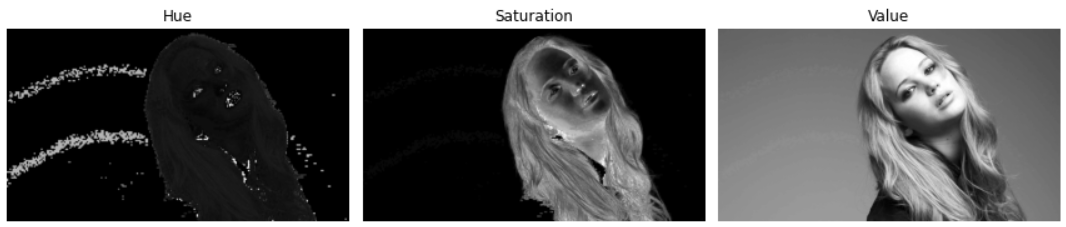


Figure 15

1. 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.
2. 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])

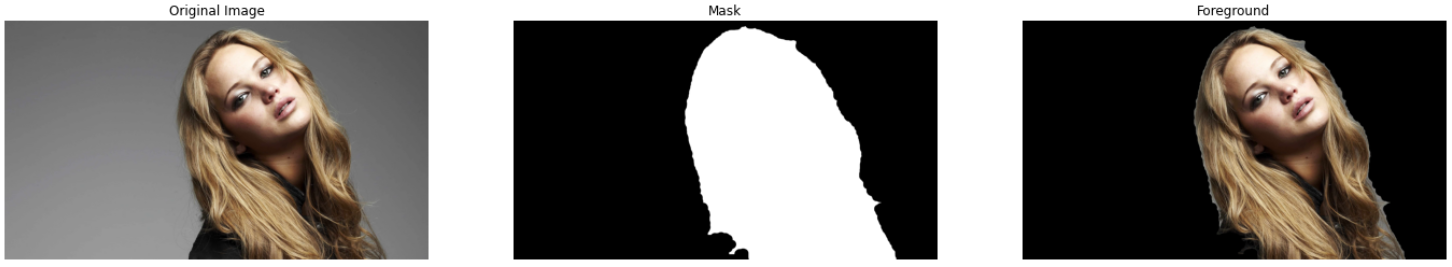


Figure 16

1. cumulative\_histogram = np.cumsum(histogram)
2. hsv\_foreground = cv.cvtColor(foreground, cv.COLOR\_BGR2HSV)

value\_foreground = hsv\_foreground[:, :, 2]

equalized\_value\_foreground = cv.equalizeHist(value\_foreground)

hsv\_foreground[:,:,2] = equalized\_value\_foreground

1. background\_mask = cv.bitwise\_not(foreground\_mask)

extracted\_background = cv.bitwise\_and(image, image, mask=background\_mask)

result = cv.add(extracted\_background, cv.cvtColor(hsv\_foreground, cv.COLOR\_HSV2BGR))



Figure 17

Question 07

1. sobel\_v = np.array([(-1, -2, -1), (0, 0, 0), (1, 2, 1)], dtype='float32')

sobel\_h = np.array([(-1, 0, 1), (-2, 0, 2), (-1, 0, 1)], dtype='float32')

imv = cv.filter2D(im, -1, sobel\_v)

imh = cv.filter2D(im, -1, sobel\_h)

grad\_mag = np.sqrt(imv\*\*2 + imh\*\*2)

1. rows, columns = im.shape

kernal\_size = 3

imv = np.zeros((rows-kernal\_size+1, columns-kernal\_size+1), dtype='float32')

imh = np.zeros((rows-kernal\_size+1, columns-kernal\_size+1), dtype='float32')

for row in range(rows-kernal\_size+1):

for column in range(columns-kernal\_size+1):

imv[row, column] = np.sum(im[row:row+kernal\_size, column:column+kernal\_size] \* sobel\_v)

imh[row, column] = np.sum(im[row:row+kernal\_size, column:column+kernal\_size] \* sobel\_h)

grad\_mag = np.sqrt(imv\*\*2 + imh\*\*2)

1. sobel\_h\_kernel = np.array([1, 2, 1], dtype=np.float32)

sobel\_v\_kernel = np.array([1, 0, -1], dtype=np.float32)

im1 = cv.sepFilter2D(im, -1, sobel\_h\_kernel, sobel\_v\_kernel)

im2 = cv.sepFilter2D(im, -1, sobel\_v\_kernel, sobel\_h\_kernel)

grad\_mag = np.sqrt(imv\*\*2 + imh\*\*2)

In all the methods, similar results were obtained.

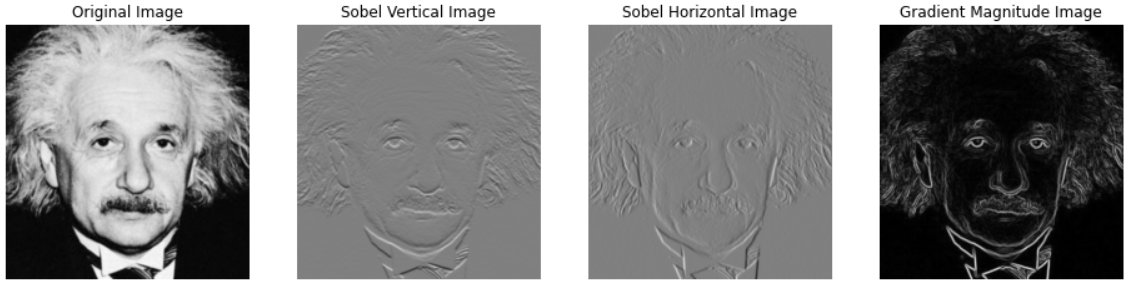


Figure 18

Question 08