Department of Electronic and Telecommunication Engineering
University of Moratuwa

Assignment I

190610E - Sumanasekara W.K.G.G.

This report is submitted as a partial fulfillment of module EN2550

March 1, 2022

Note: All python codes relevant to this assignment can be found in https://github.com/GevinduGanganath/EN2550/tree/main/Assignment%201

Question 1

In the given intensity transformation, pixel values lie within the range 50 to 150 has been increased while other pixels remain same. Figure 1 shows the original image, intensity transformation function and output image respectively.

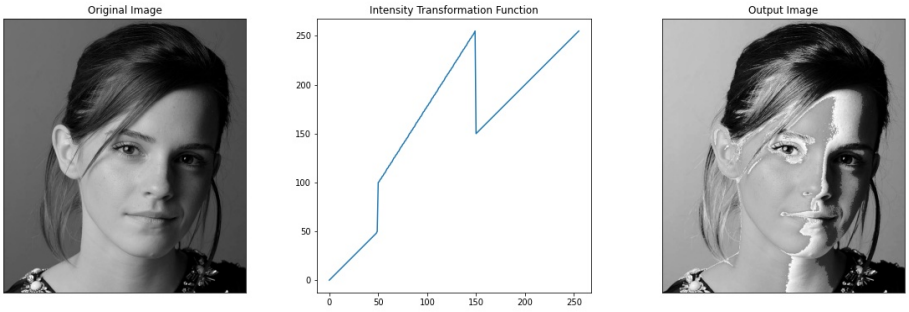

Figure 1: Question 1

Question 2

(a) In this part the white matter of brain proton density image has been accentuated. Applied intensity transformation is shown in figure 2. Since both the white matter and gray matter have closer pixel values it is very important to select the correct cut-off value. Here 175 was selected as cut-off value and range between 150 and 200 has transformed linearly while others are shifted to pure white or black.

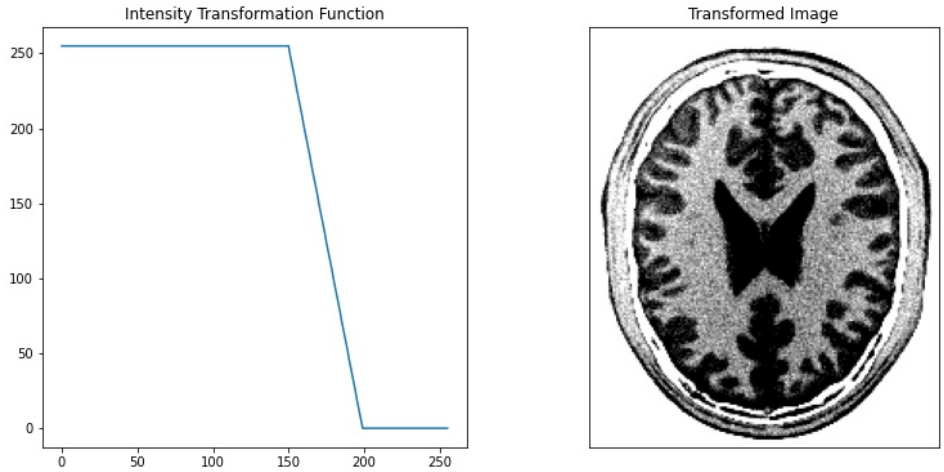

Figure 2: Question 2-a

(b) As the second part, gray matter of the image has been accentuated. Here the transformation is different from the previous one because if a transformation of the same shape

is applied white matter also accentuated and then it is difficult to figure out the features of gray matter. Corresponding transformation function and accentuated image is shown in figure 3.

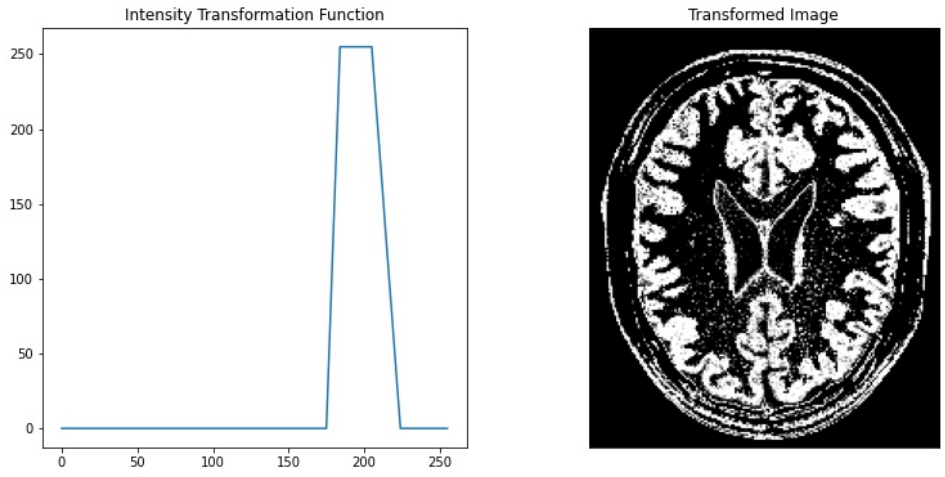

Figure 3: Question 2-b

Question 3

In this exercise a gamma correction ($\gamma = 0.8$) has been performed on the L place of the given image after converting it to the L*a*b colour space. Results are shown in figure 4.

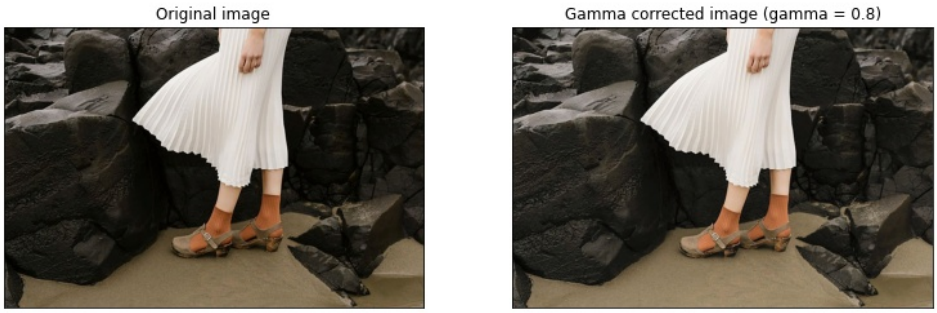

Figure 4: Question 3 input and gamma corrected images

In the L*a*b colour space L represents the lightness of the pixel. According to the equation 1 applying a gamma value less than 1 always produces a new L value which is grater than the previous. Therefore, after the gamma correction output image is lighter than before giving a nice appearance to dark places like rock hallows.

$$\text{new L value} = 255 \left(\frac{\text{current L value}}{255} \right)^{0.8} \tag{1}$$

This can also be represented using the histograms of the two versions of image. As you can observe in figure 5, after the gamma correction histogram has moved right slightly while storing more pixels in right most bins.

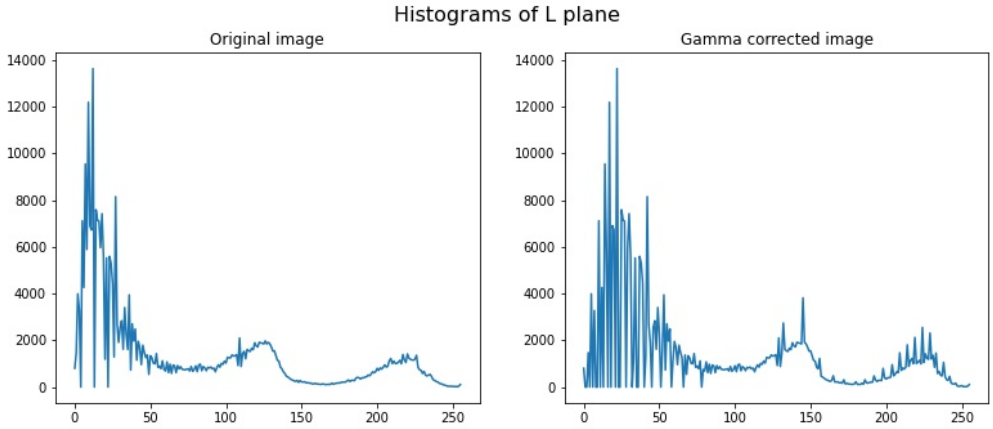

Figure 5: Question 3 histograms

Question 4

Images may have histograms confined into some region, but a histogram of a good image have the values in all regions. Therefore, we need to distribute the pixel values throughout the region. That is what histogram equalization does.

In this exercise a python function was written (figure 6) to carry out the histogram equalization on a given image. As the first step histogram of the given image was obtained using the numpy histogram function and the cumulative summation was calculated. Then the histogram equalization equation can be applied resulting a transformation function. Finally, it can be used as a look-up table to generate the equalized image. Resulting histograms are shown in figure 7 and the proper operation of the implemented function can be verified by comparing it with the output of openCV in-built histogram equalization function. Equalized image is shown in figure 8.

```
def equalizeHist(img):
    hist = np.histogram(img.flatten(),256,[0,256])[0]
    cdf = hist.cumsum()
    cdf = np.ma.masked_equal(cdf,0)
    cdf = (cdf - cdf.min())*255/(cdf.max()-cdf.min())
    cdf = np.ma.filled(cdf,0).astype('uint8')
    img_equatized = cv.LUT(img, cdf)

    return img_equatized
```

Figure 6: Histogram Equalization Function

Question 5

Two functions were implemented to carry out the zooming and calculate the normalized sum of squared difference (SSD). The zooming function is capable of handling the both nearest-neighbor and bilinear interpolation techniques. By observing the SSD values in table 1, we can conclude that bilinear interpolation provides the better results.

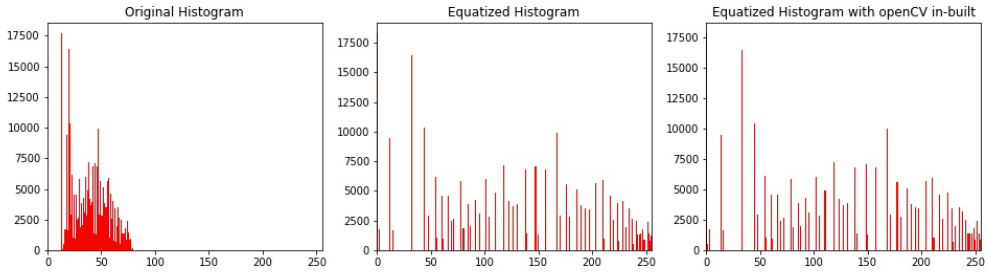

Figure 7: Histograms of shell image

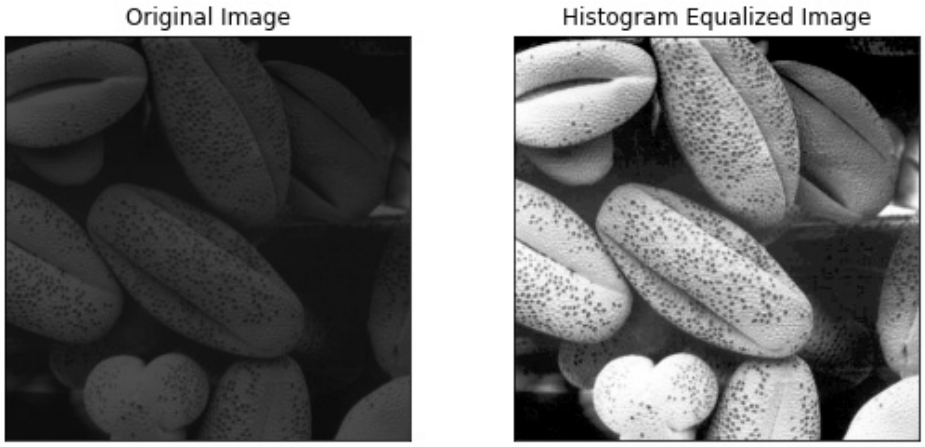

Figure 8: Equalized image

Table 1: SSD value comparison

Image	SSD with nearest-neighbor interpolation	bilinear interpolation
Image 1	641.788	628.113
Image 2	268.688	259.388
Image 3	426.194	404.029

Question 6

(a) Here openCV filter2D function was used to carry out the sobel filter on Einstein image. Sobel kernels detect the edges of a given image. As can be observed in the figure 9 Sobel vertical kernel detects the horizontal edges while Sobel horizontal kernel detects the vertical edges.

(b) A manual python function was written to carry out the sobel vertical and horizontal filtering. The function is shown in figure 10. Outputs are almost same as the in-built filter2D function (figure 11)

(c) Here the sobel filtering was carried out with associative property. First the image is filtered with $[-1, 0, 1]^T$ and then with $[1, 2, 1]$ for vertical. Transposes of the above matrices are applied for the horizontal. Again the results are same as in-built function (figure 12).

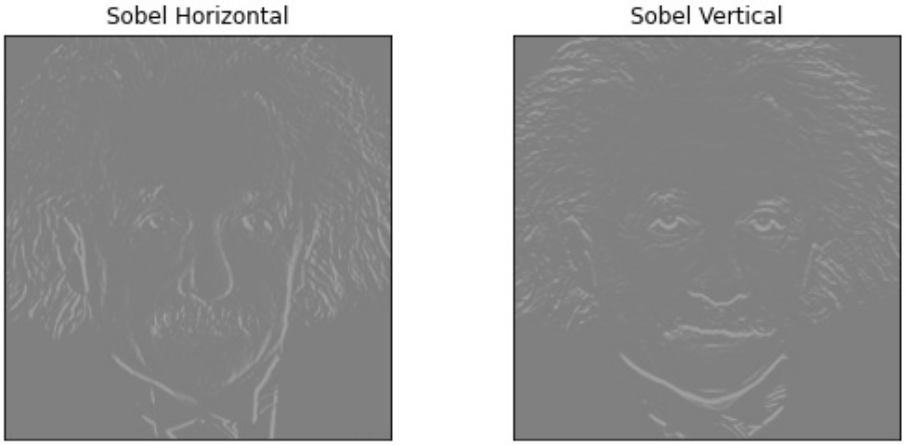

Figure 9: Question 6-a

```python
def sobel(img, mode):
    if mode == "vertical": kernel = np.array(((-1, -2, -1), (0, 0, 0), (1, 2, 1)), dtype=np.float32)
    elif mode == "horizontal": kernel = np.array(((-1, 0, 1), (-2, 0, 2), (-1, 0, 1)), dtype=np.float32)
    else: raise TypeError

    h, w = img.shape
    result = np.zeros((h, w), "float")

    for i in range(1, h-1):
        for j in range(1, w-1):
            result[i, j] = np.dot(img[i-1 :i+2 , j-1: j+2].flatten(), kernel.flatten())

    return result
```

Figure 10: Manual sobel filtering function

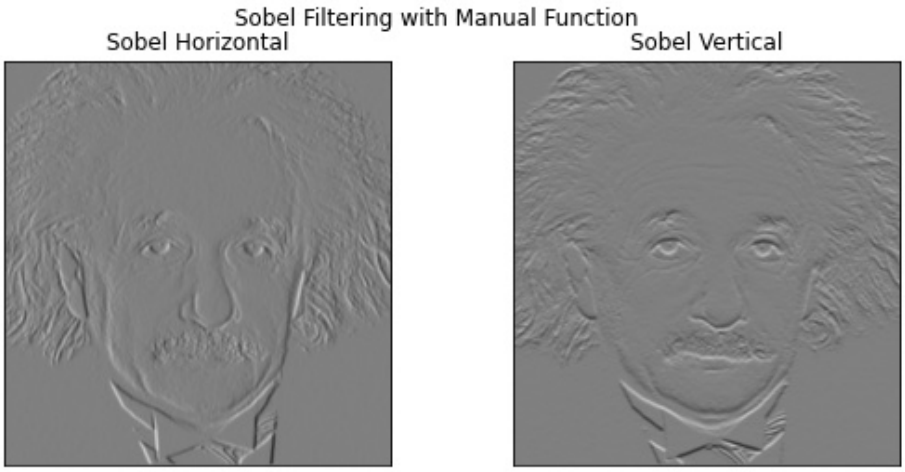

Figure 11: Question 6-b

Question 7

(a) Here a grabCut segmentation was carried out to extract the background from the given image. Initially, I tried the segmentation with bounding boxes, but it did not produce the

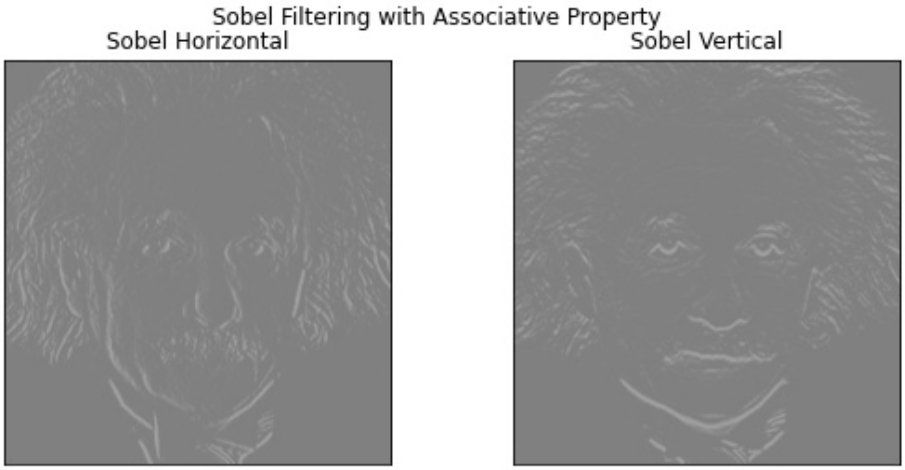

Figure 12: Question 6-c

expected output, specially closer to the flower bud and to the petals of flower in question. Then a mask was generated by doing few modifications to the binary thresholded image and grabCut was carried out with mask approximation. In this case the outputs were better than the previous. Results are shown in figure 13.

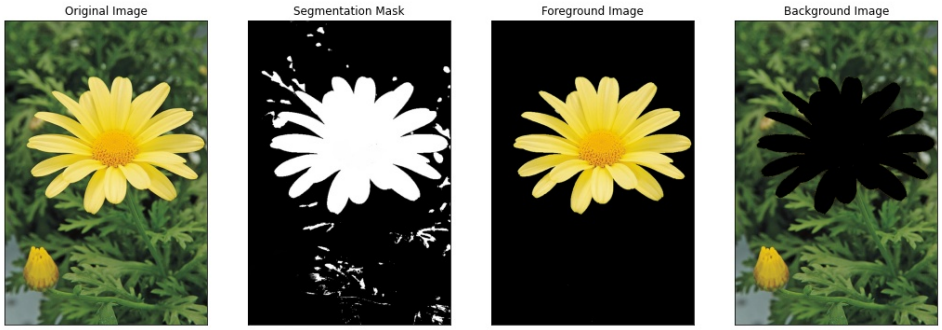

Figure 13: Question 7-a

(b) In the second part an enhanced image was produced by summing up the foreground image (saturation increased) and background image (Gaussian blurred). Output is shown in figure 14.

(c) When the Gaussian blurring is carried out on the background image, it generates some values for the pixels where initially we had extracted as the foreground. Then after the summing up this edge seems to be quite dark. This effect can be reduced by reducing the Gaussian kernel size (figure 15).

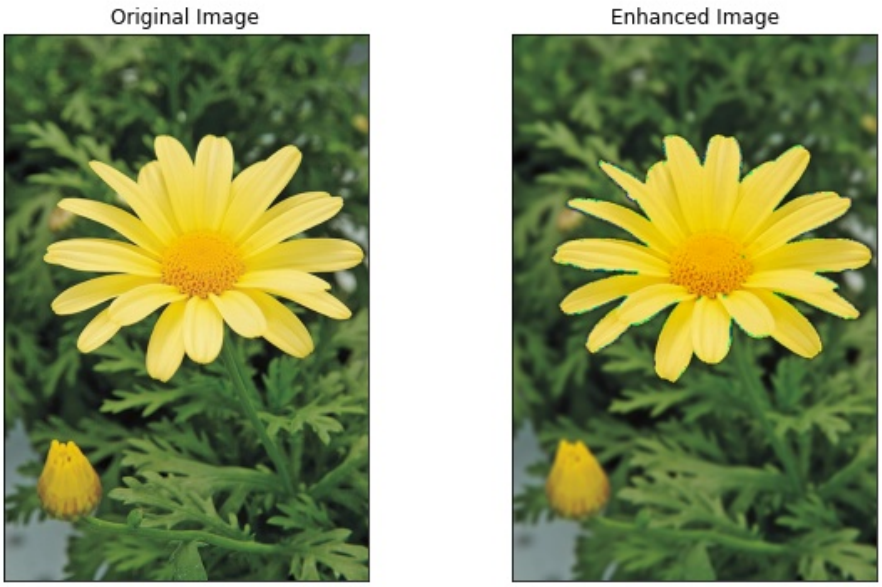

Figure 14: Question 7-b

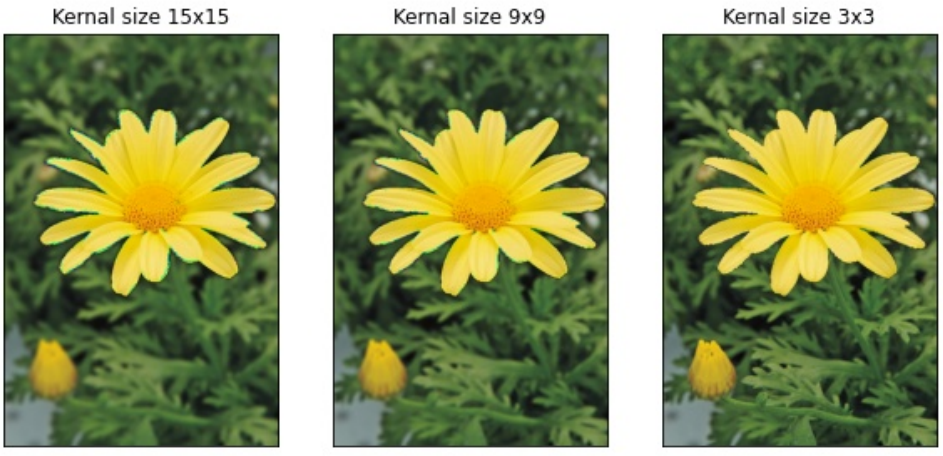

Figure 15: Question 7-c