Department of Electronic & Telecommunication Engineering,
University of Moratuwa, Sri Lanka.

EN3160 - Image Processing and Machine Vision

E.M.A.R. Niroshan 210433R

Assignment 01

Intensity Transformations and Neighborhood Filtering

10th September 2024

GitHub Link

Question 1

Figure 1: Intensity transformation.

50 - 150 pixel intensity values : Increased
Other pixel intensity values : Same as original

Listing 1: Q1

```
c = np.array([(50, 50), (50, 100), (150, 255), (150, 150), (255, 255)])

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

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

Question 2

Wight matter enhanced: 80 to 180 pixel intensity values increased. Others are same.
Gray matter enhanced : Dark values further reduced. Bright values further increased.

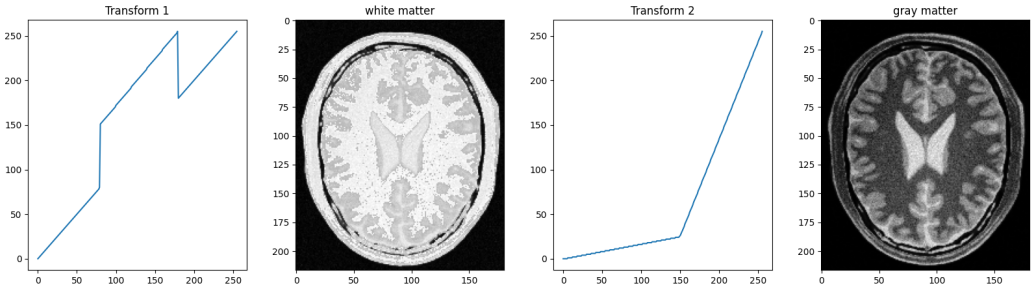

Figure 2

Listing 2: Q2

```
t1 = np.linspace(0,80,80)
t2 = np.linspace(81,150,0)
t3 = np.linspace(151,255,100)
t4 = np.linspace(180,255,76)
transform1 = np.concatenate((t1,t2,t3,t4), axis=0).astype(np.uint8)

t1 = np.linspace(0,25,150)
t2 = np.linspace(26,255,106)
transform2 = np.concatenate((t1,t2), axis=0).astype(np.uint8)
```

Question 3

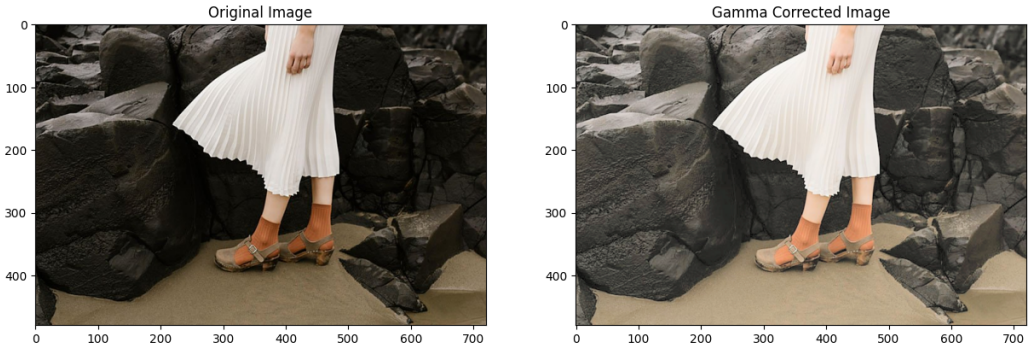

Figure 3: Image

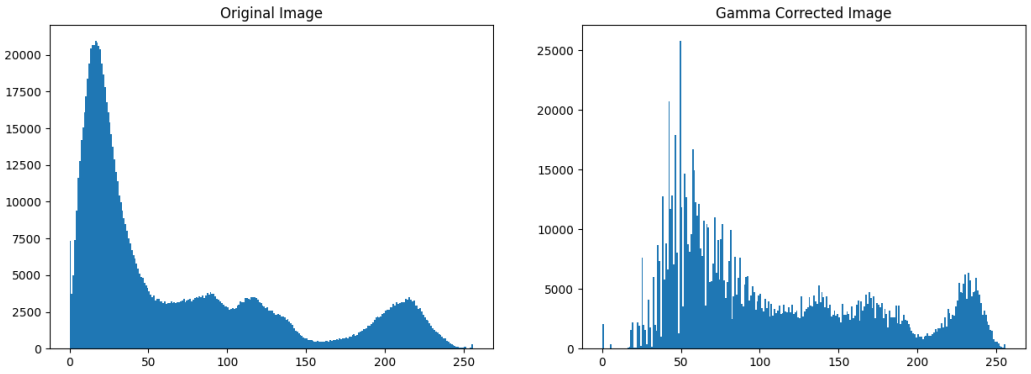

Figure 4: Histogram

When γ is set to 0.5, the narrow range of dark pixels expand to cover a wider range. This can be observed by comparing the histograms of the original and gamma corrected images, where the frequencies of the dark pixels in the original image are spread out over a larger range in the corrected image. As a result, the darker areas of the original image is becomes more visible, making the overall image appear clearer.

Listing 3: Q3

```
lab_image = cv.cvtColor(image, cv.COLOR_RGB2LAB)
L, a, b = cv.split(lab_image)

gamma = 0.5
transform = np.array([(i/255.0)**(gamma)*255 for i in np.arange(0, 256)]).astype(np.
    uint8)

L = cv.LUT(L, transform)

lab_image = cv.merge((L, a, b))
new_image = cv.cvtColor(lab_image, cv.COLOR_LAB2RGB)
```

Question 04

a = 0.5
By applying this intensity transformation to the saturation plane, the intensity of the image's pixel intensities can be effectively increased without significantly affecting the overall brightness.

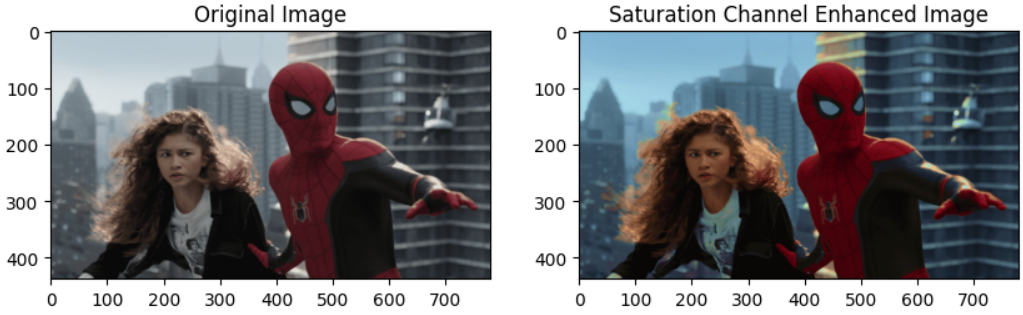

Figure 5

Listing 4: Q4

```
hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)
h, s, v = cv.split(hsv_image)

a = 0.5
sigma = 70
s_new = np.minimum(s + a*128*np.exp(-((s - 128) ** 2)/(2*(sigma**2))), 255).astype(np
    .uint8)

hsv_image = cv.merge((h, s_new, v))
```

Question 05

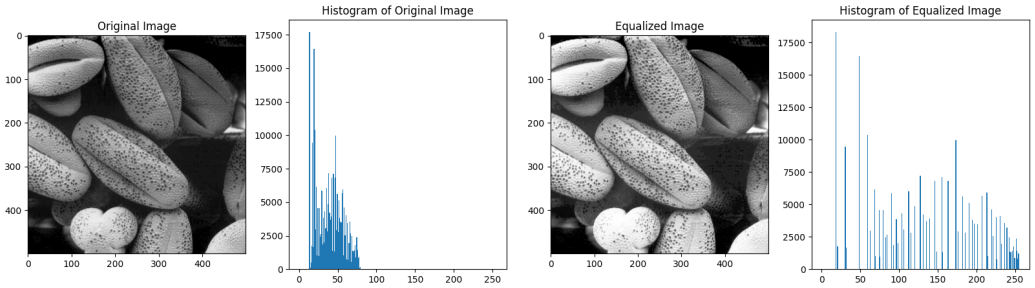

Figure 6: Histogram Equalization

After applying histogram equalization, the image appears more vivid. The histogram of the equalized image becomes nearly flat, distributing pixel intensities more evenly across the entire range.

0.1 Question 06

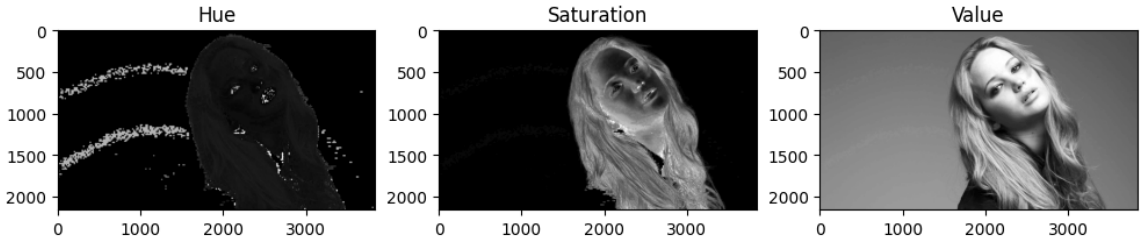

Figure 7: hsv Channels

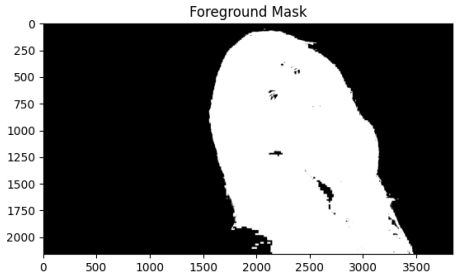

Figure 8: Mask

The face looks brighter than other places. Therefore histogram equalize the foreground only.The foreground mask was created by manually setting a threshold value of 15, using the saturation plane for mask extraction. The saturation plane is chosen because it effectively represents the color distribution of the image. Forground has higher saturation values than the background.

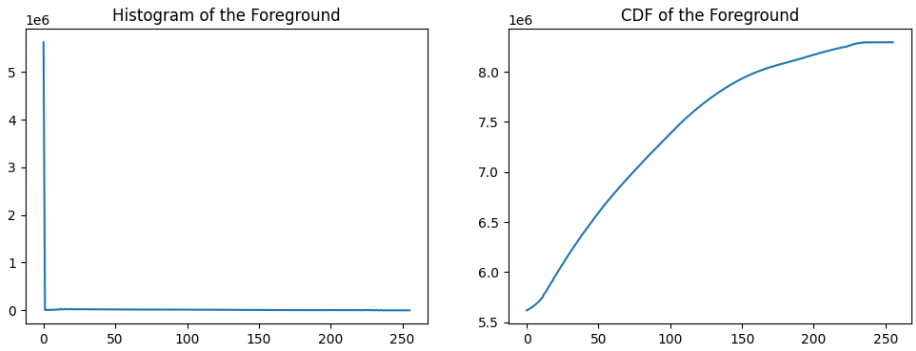

Figure 9

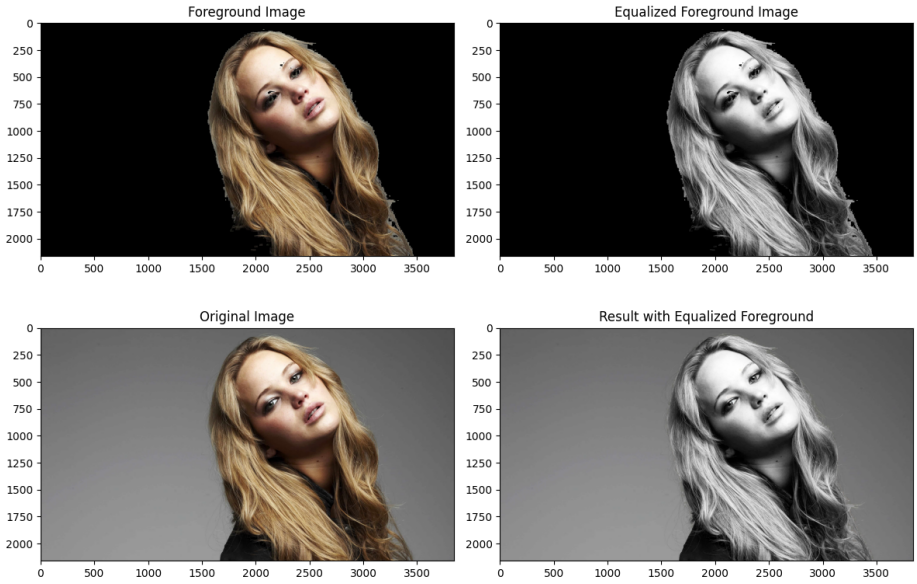

Figure 10

0.2 Question 07

Listing 5: Normal Sobel Filters

```
sobel_x = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]]) #Sobel Vertical
sobel_y = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]) #Sobel Horizontal

g_x = cv.filter2D(image, cv.CV_32F, sobel_x)
g_y = cv.filter2D(image, cv.CV_32F, sobel_y)
g_mag = np.sqrt(g_x**2 + g_y**2)
```

Listing 6: My Own Code

```
def convolution2d(image, kernel):
    m, n = kernel.shape
    if (m == n):
        y, x = image.shape
        y = y - m + 1
        x = x - m + 1
        new_image = np.zeros((y,x))
```

```
        for i in range(y):
            for j in range(x):
                new_image[i][j] = np.sum(image[i:i+m, j:j+m]*kernel)
    return new_image

# Convolution
g_x = convolution2d(padded_image, sobel_x)
g_y = convolution2d(padded_image, sobel_y)
g_mag = np.sqrt(g_x**2 + g_y**2)
```

Listing 7: Using the Properties of Separability

```
sobel_v = np.array([[1, 2, 1]], dtype=np.float32)
sobel_h = np.array([[1], [0], [-1]], dtype=np.float32)

g_x = cv.filter2D(image, cv.CV_32F, sobel_v)
g_x = cv.filter2D(g_x, cv.CV_32F, sobel_h)
g_y = cv.filter2D(image, cv.CV_32F, sobel_h)
g_y = cv.filter2D(g_y, cv.CV_32F, sobel_v)
g_mag = np.sqrt(g_x**2 + g_y**2)
```

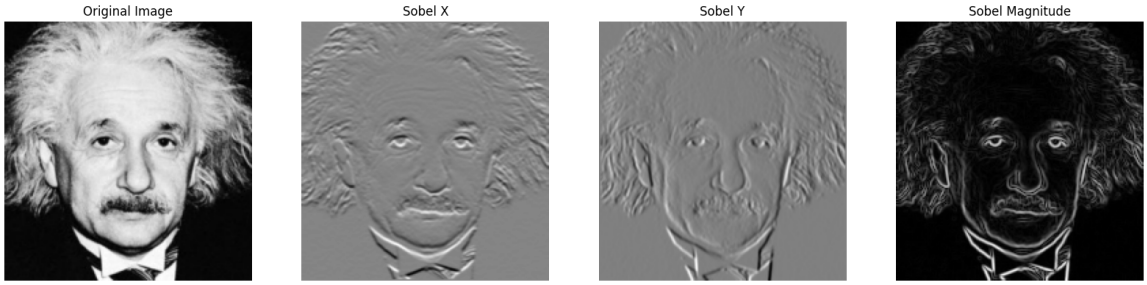

Figure 11: Result

The result is same in all 3 methods. But the method 3 (Using the property of separability of the Sobel operator) is computationally less complex than others. When using large sized images, this will be very useful.

0.3 Question 08

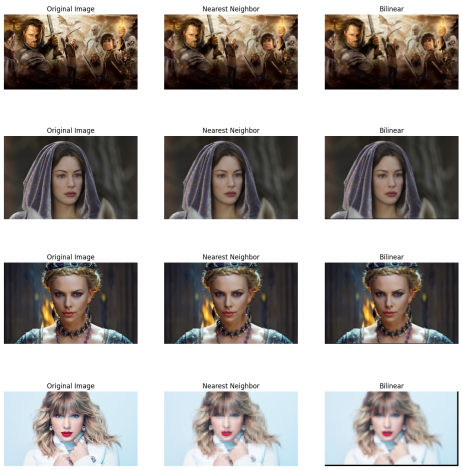

Figure 12

Normalized SSD for Nearest Neighbor and Bilinear Interpolation:
Image 1: NN = 0.03, BL = 0.03
Image 2: NN = 0.02, BL = 0.03
Image 3: NN = 0.03, BL = 0.03
Image 4: NN = 0.01, BL = 0.04
Nearest neighbour zooming is fast and maps pixels directly by rounding values, while bilinear interpolation calculates pixel values by averaging the four nearest neighbours for smoother results. Despite these differences, the SSD values for both methods are nearly same.

0.4 Question 09

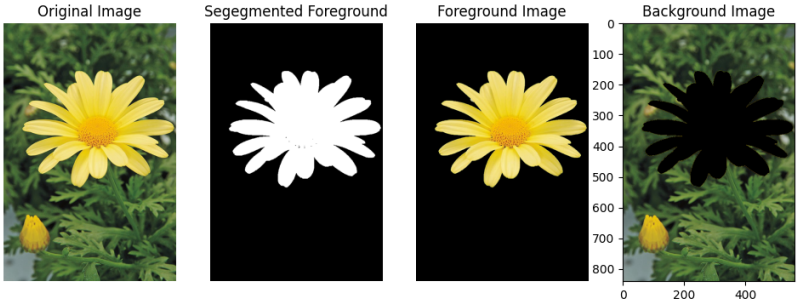

Figure 13

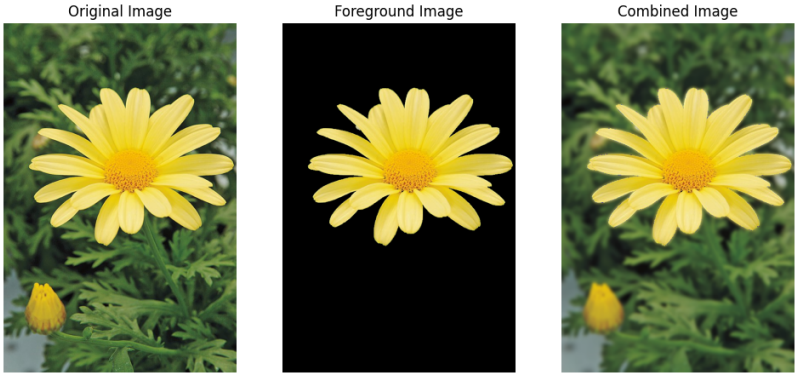

Figure 14

The GrabCut algorithm segments the forground from the background, but the mask may not perfectley capture the flower's boundary, causing some areas near the edge to be misclassified as background. When applying a Gaussian blur to the background, these regions blend with neighboring pixels, leading to a darkening effect, especially where the background contains dark areas. Misclassified flower details also contribute to this, making the background near the flower's edge appear darker.