



# **EN2550 Fundamentals of Image Processing and Machine Visions**

Intensity Transformations and Neighborhood Filtering

## **Author**

Sadith W.M.L. 190538N  
link for GitHub repository

## Question 1

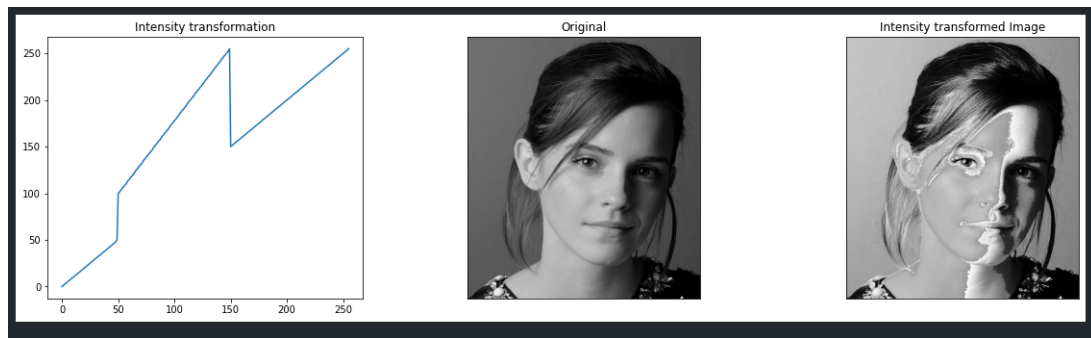


Figure 1: Intensity Transformation

```

1 import numpy as np
2 import cv2 as cv
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5
6 #Q1
7 t1 = np.linspace(0,50,50)
8 t2 = np.linspace(100,255,100)
9 t3 = np.linspace(150,255,106)
10
11 t = np.concatenate((t1,t2,t3),axis = 0).astype(np.uint8)
12 print(len(t))
13
14 img = cv.imread('emma-gray.jpg',cv.IMREAD_GRAYSCALE)
15 assert img is not None, "Image Not Found!!"
16
17 new_img = cv.LUT(img,t)
18
19 fig,ax = plt.subplots(1,3,figsize = [20,5])
20 ax[0].plot(t)
21 ax[1].imshow(img,cmap='gray',vmin=0,vmax=255)
22 ax[1].axes.xaxis.set_visible(False)
23 ax[1].axes.yaxis.set_visible(False)
24 ax[0].set_title('Intensity transformation')
25 ax[1].set_title('Original')
26 ax[2].set_title('Intensity transformed Image')
27 ax[2].imshow(new_img,cmap='gray',vmin=0,vmax=255)
28 ax[2].axes.xaxis.set_visible(False)
29 ax[2].axes.yaxis.set_visible(False)
30 plt.show()

```

Source Code 1: Intensity Transformation

## Question 2

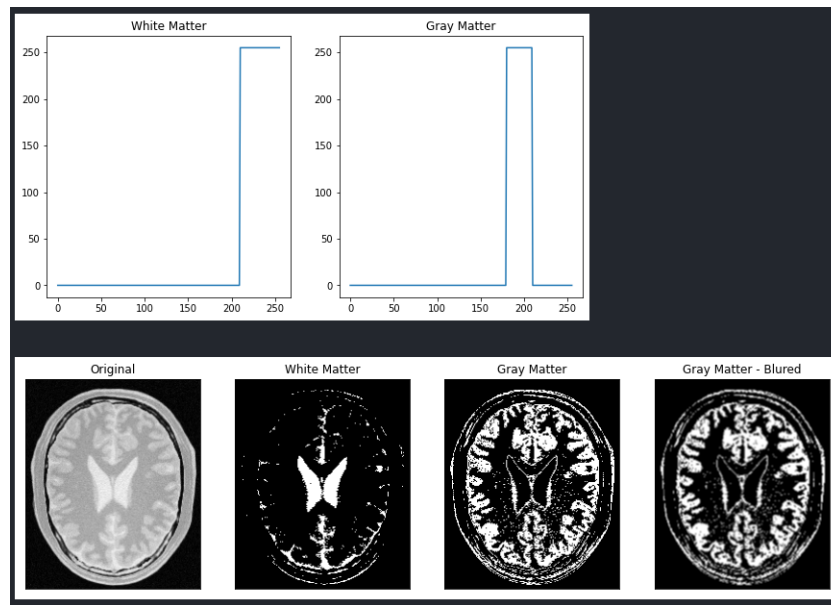


Figure 2: Separating White and Dark Matter

```

1 #Q2
2 x = 210
3 t1 = np.linspace(0,0,x)
4 t2 = np.linspace(255,255,256-x)
5
6 y = 180
7 y_size = 30
8 t3 = np.linspace(0,0,y)
9 t4 = np.linspace(255,255,y_size)
10 t5 = np.linspace(0,0,256-y_size-y)
11
12
13 t_white = np.concatenate((t1,t2),axis = 0).astype(np.uint8)
14 t_gray = np.concatenate((t3,t4,t5),axis = 0).astype(np.uint8)
15 assert len(t_white)== 256, "Transformation Incorrect"
16 assert len(t_gray)== 256, "Transformation Incorrect"
17
18
19 img = cv.imread('brain-proton-density-slice.png',cv.IMREAD_GRAYSCALE).astype(np.uint8)
20 assert img is not None, "Image Not Found!!"
21
22 white_matter = cv.LUT(img,t_white)
23 gray_matter = cv.LUT(img,t_gray)
24 gray_matter_filtered = cv.GaussianBlur(gray_matter,(3,3),0)
25 # gray_matter_filtered = cv.medianBlur(gray_matter,3)
26
27
28 fig,ax = plt.subplots(1,2,figsize = [10,5])
29 ax[0].plot(t_white)
30 ax[1].plot(t_gray)
31
32 ax[0].set_title('White Matter')
33 ax[1].set_title('Gray Matter')
34
35 fig,ax = plt.subplots(1,4,figsize = [15,5])
36
37 ax[0].imshow(img,cmap='gray',vmin=0,vmax=255)
38 ax[0].axes.xaxis.set_visible(False)
39 ax[0].axes.yaxis.set_visible(False)
40
41 ax[1].imshow(white_matter,cmap='gray',vmin=0,vmax=255)
42 ax[1].axes.xaxis.set_visible(False)
43 ax[1].axes.yaxis.set_visible(False)
44
45 ax[2].imshow(gray_matter,cmap='gray',vmin=0,vmax=255)
46 ax[2].axes.xaxis.set_visible(False)
47 ax[2].axes.yaxis.set_visible(False)
48
49 ax[3].imshow(gray_matter_filtered,cmap='gray',vmin=0,vmax=255)
50 ax[3].axes.xaxis.set_visible(False)
51 ax[3].axes.yaxis.set_visible(False)
52
53 ax[0].set_title('Original')
54 ax[1].set_title('White Matter')
55 ax[2].set_title('Gray Matter')
56 ax[3].set_title('Gray Matter - Blurred')
57 plt.show()

```

Source Code 2: Separating White and Dark Matter

## Question 3

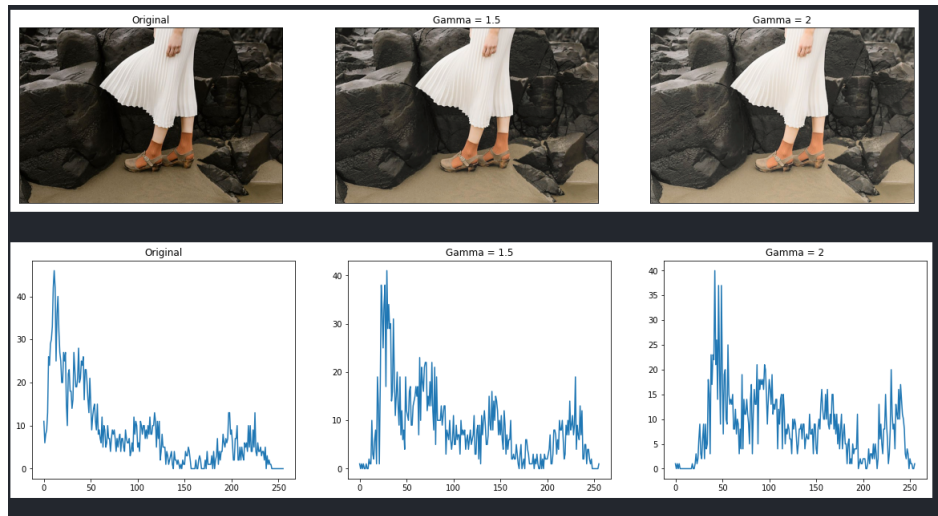


Figure 3: Gamma correction

```

1 #Q3
2 def gammaCorrection(img,gamma): #gamma correction only to the L plane
3     temp_img = img.copy()
4     invGamma = 1/gamma
5     for i in range(len(img)):
6         for j in range(len(img[0])):
7             temp_img[i][j][0] = ((img[i][j][0] / 255) ** invGamma) * 255
8     return temp_img
9
10 img = cv.imread('highlights-and-shadows.jpg').astype(np.uint8)
11 assert img is not None, "Image Not Found!!"
12
13 gamma_1 = 1.5
14 gamma_2 = 2
15
16 img_RGB = cv.cvtColor(img,cv.COLOR_BGR2RGB)
17 img_Lab = cv.cvtColor(img,cv.COLOR_BGR2Lab)
18
19 img_corrected_1 = gammaCorrection(img_Lab,gamma_1)
20 img_corrected_2 = gammaCorrection(img_Lab,gamma_2)
21
22 img_corrected_RGB_1 = cv.cvtColor(img_corrected_1,cv.COLOR_LAB2RGB)
23 img_corrected_RGB_2 = cv.cvtColor(img_corrected_2,cv.COLOR_LAB2RGB)
24
25 fig,ax = plt.subplots(1,3,figsize = [20,8])
26
27 ax[0].imshow(cv.cvtColor(img,cv.COLOR_BGR2RGB))
28 ax[0].axes.xaxis.set_visible(False)
29 ax[0].axes.yaxis.set_visible(False)
30
31
32 ax[1].imshow(img_corrected_RGB_1)
33 ax[1].axes.xaxis.set_visible(False)
34 ax[1].axes.yaxis.set_visible(False)
35
36 ax[2].imshow(img_corrected_RGB_2)
37 ax[2].axes.xaxis.set_visible(False)
38 ax[2].axes.yaxis.set_visible(False)
39
40 ax[0].set_title('Original')
41 ax[1].set_title('Gamma = {}'.format(gamma_1))
42 ax[2].set_title('Gamma = {}'.format(gamma_2))
43 plt.show()
44
45 hist_original = cv.calcHist(cv.cvtColor(img_RGB,cv.COLOR_BGR2RGB),[0],None,[256],[0,256])
46 hist_corrected_1 = cv.calcHist(img_corrected_RGB_1,[0],None,[256],[0,256])
47 hist_corrected_2 = cv.calcHist(img_corrected_RGB_2,[0],None,[256],[0,256])
48
49 fig,ax = plt.subplots(1,3,figsize = [20,5])
50
51 ax[0].plot(hist_original)
52 ax[1].plot(hist_corrected_1)
53 ax[2].plot(hist_corrected_2)
54
55 ax[0].set_title('Original')
56 ax[1].set_title('Gamma = {}'.format(gamma_1))
57 ax[2].set_title('Gamma = {}'.format(gamma_2))
58
59 # plt.legend(loc=1, prop={'size': 10})
60 plt.show()

```

Source Code 3: Gamma correction

## Question 4

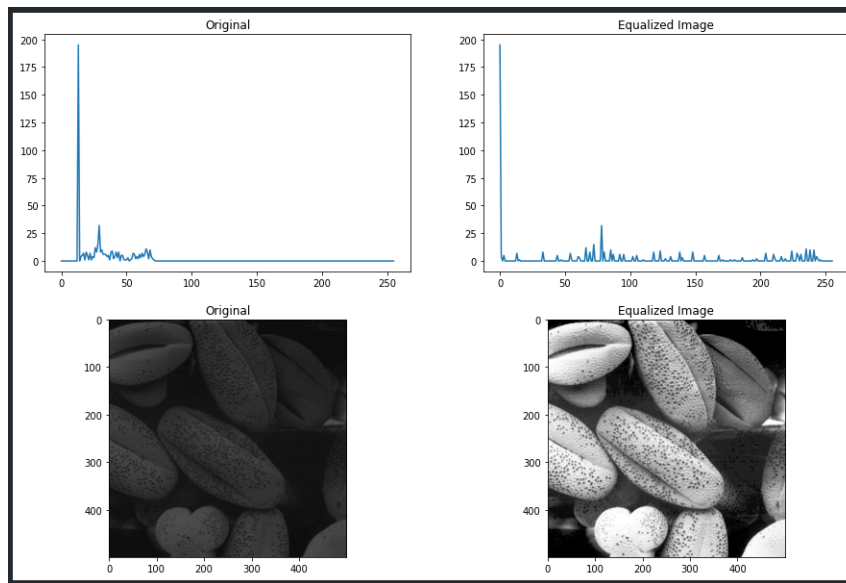


Figure 4: Histogram equalization

```

1 #Q4
2 img = cv.imread('shells.png',cv.IMREAD_GRAYSCALE).astype(np.uint8)
3 assert img is not None, "Image Not Found!!"
4
5 eq_img = cv.equalizeHist(img)
6
7 hist_original = cv.calcHist(img,[0],None,[256],[0,256])
8 hist_corrected_1 = cv.calcHist(eq_img,[0],None,[256],[0,256])
9 # hist_corrected_2 = cv.calcHist(img_corrected_RGB_2,[0],None,[256],[0,256])
10
11 fig,ax = plt.subplots(2,2,figsize = [15,10])
12
13 ax[0][0].plot(hist_original)
14 ax[0][1].plot(hist_corrected_1)
15 ax[1][0].imshow(img,cmap = 'gray',vmin =0,vmax =255)
16 ax[1][1].imshow(eq_img,cmap = 'gray',vmin =0,vmax =255)
17
18 ax[0][0].set_title('Original')
19 ax[0][1].set_title('Equalized Image')
20 ax[1][0].set_title('Original')
21 ax[1][1].set_title('Equalized Image')
22
23 plt.show()

```

Source Code 4: Histogram equalization

## Question 5 - (a)

```

1 #Q5 - zoom images using nearest-neighbor method
2 import numpy as np
3 import cv2 as cv
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6
7 def convertIndex(i,j,scale):
8     x = int(i/scale)
9     y = int(j/scale)
10    return x,y
11
12 def zoomImg(img,scale):
13    rows = int(img.shape[0]*scale)
14    columns = int(img.shape[1]*scale)
15
16    new_img = np.zeros((rows,columns,3),img.dtype)
17
18    for i in range(rows):
19        for j in range(columns):
20            x,y = convertIndex(i,j,scale)
21            new_img[i][j] = img[x][y]
22
23    return new_img
24
25 def displayImages(image):
26    img = cv.imread(image)
27    assert img is not None, "Image Not Found!!"
28

```

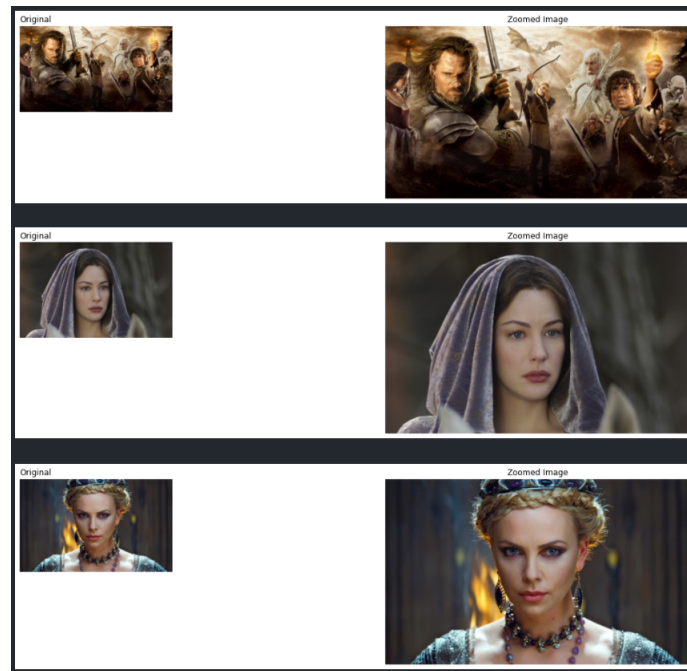


Figure 5: Zoom using nearest-neighbor method

```

29 img = cv.cvtColor(img,cv.COLOR_BGR2RGB)
30 zoomed_img = zoomImg(img,2)
31
32 fig,ax = plt.subplots(1,2,figsize =[18, 6],sharey=True,sharex=True)
33
34 ax[0].imshow(img)
35 ax[1].imshow(zoomed_img)
36 ax[0].axis('off')
37 ax[1].axis('off')
38 ax[0].set_title('Original',loc='left')
39 ax[1].set_title('Zoomed Image')
40
41 displayImages('alq5images/im01small.png')
42 displayImages('alq5images/im02small.png')
43 displayImages('alq5images/im03small.png')
44
45 plt.show()

```

Source Code 5: Zoom using nearest-neighbor method

## Question 5 - (b)

```

1 #Q5 - zoom images using bilinear interpolation method
2 def zoomImg2(img,scale):
3     rows = int(img.shape[0]*scale)
4     cols = int(img.shape[1]*scale)
5     zoomed = cv.resize(img,(cols,rows),interpolation = cv.INTER_LINEAR)
6     return zoomed
7
8 def displayImages(image):
9     img = cv.imread(image)
10    assert img is not None, "Image Not Found!!"
11
12    img = cv.cvtColor(img,cv.COLOR_BGR2RGB)
13    zoomed_img = zoomImg2(img,2)
14
15    fig,ax = plt.subplots(1,2,figsize =[18, 6],sharey=True,sharex=True)
16
17    ax[0].imshow(img)
18    ax[1].imshow(zoomed_img)
19    ax[0].axis('off')
20    ax[1].axis('off')
21    ax[0].set_title('Original',loc='left')
22    ax[1].set_title('Zoomed Image')
23
24 displayImages('alq5images/im01small.png')
25 displayImages('alq5images/im02small.png')
26 displayImages('alq5images/im03small.png')
27
28 plt.show()

```

Source Code 6: Zoom using Bilinear Interpolation

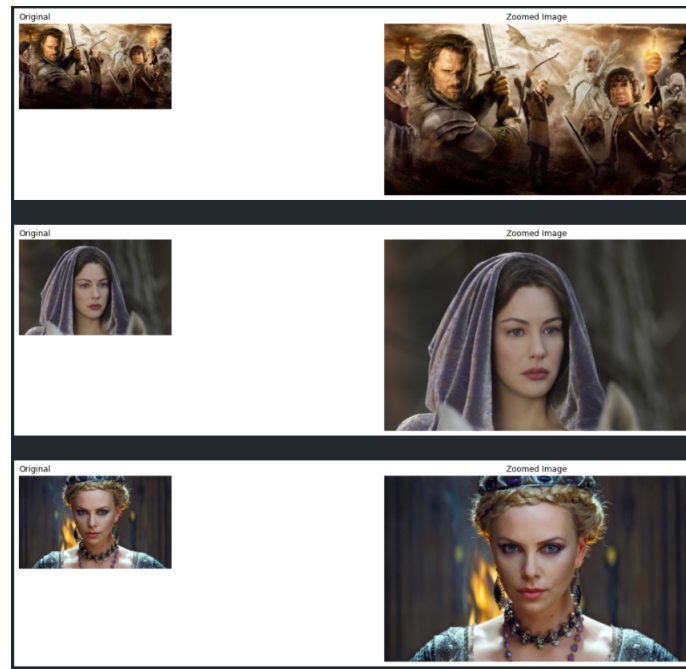


Figure 6: Zoom using Bilinear Interpolation

## Question 6 - (a)

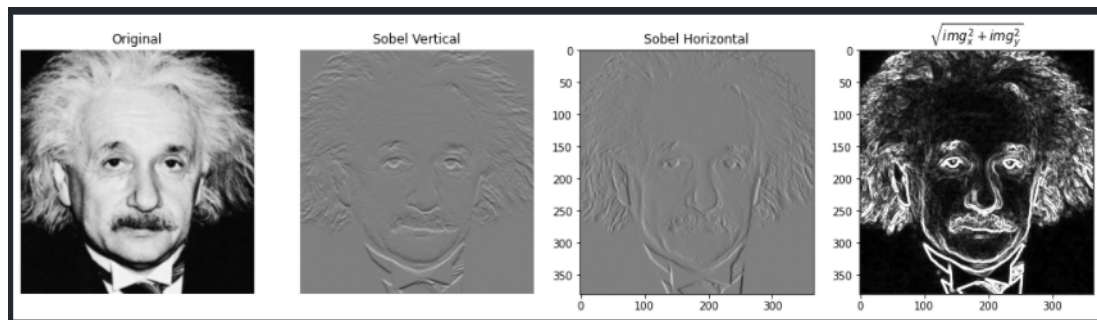


Figure 7: Sobel Filtering - 1

```

1 #Q6 - part A
2 img = cv.imread('einstein.png', cv.IMREAD_GRAYSCALE).astype('float32')
3 assert img is not None, "Image Not Found!!"
4
5 sobel_v = np.array([[-1,-2,-1],[0,0,0],[1,2,1]], dtype=np.float32)
6 sobel_h = np.array([[-1,0,1],[-2,0,2],[-1,0,1]], dtype=np.float32)
7
8 img_x = cv.filter2D(img,-1,sobel_v)
9 img_y = cv.filter2D(img,-1,sobel_h)
10
11 grad_mag = np.sqrt(img_x**2 +img_y**2)
12
13
14 fig,ax = plt.subplots(1,4,figsize=(18,6))
15 ax[0].imshow(img,cmap='gray',vmin=0,vmax=255)
16 ax[1].imshow(img_x,cmap='gray',vmin=-1020,vmax=1020)
17 ax[2].imshow(img_y,cmap='gray',vmin=-1020,vmax=1020)
18 ax[3].imshow(grad_mag,cmap='gray',vmin=0,vmax=255)
19
20 ax[0].axis('off')
21 ax[1].axis('off')
22
23 ax[0].set_title("Original")
24 ax[1].set_title("Sobel Vertical")
25 ax[2].set_title("Sobel Horizontal")
26 ax[3].set_title("$\sqrt{img_x^2 + img_y^2}$")
27
28 plt.show()

```

Source Code 7: Sobel Filtering - 1



## Question 6 - (b)

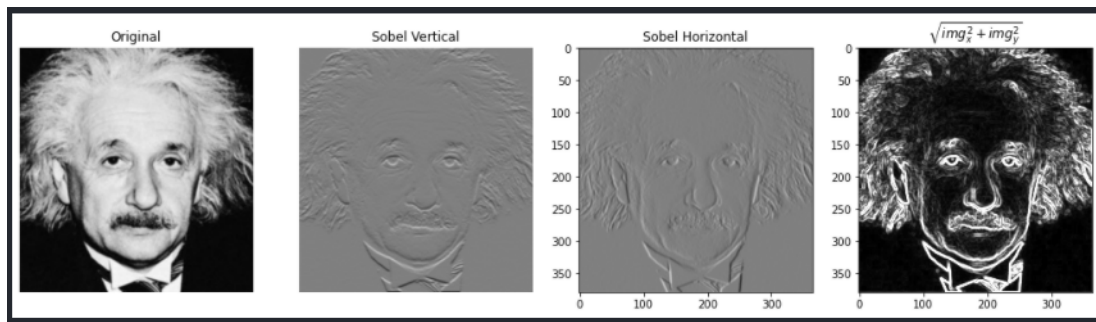


Figure 8: Sobel Filtering - 2

```

1 #Q6 - part B
2
3 def convolve2D(img, kernel):
4     row, col = np.shape(img)
5     x, y = np.shape(kernel)
6     img_pad = np.pad(img, ((x//2, x//2), (y//2, y//2)), 'constant', constant_values=(0,0))
7     img_con = np.zeros((np.shape(img)))
8     #convolution
9     for i in range(row):
10         for j in range(col):
11             img_con[i][j] = np.sum(img_pad[i:i+x, j:j+y]*kernel)
12
13     return img_con
14
15 img = cv.imread('einstein.png',cv.IMREAD_GRAYSCALE).astype('float32')
16 assert img is not None, "img Not Found!!"
17
18 sobel_v = np.array([[-1,-2,-1],[0,0,0],[1,2,1]],dtype=np.float32)
19 sobel_h = np.array([[-1,0,1],[-2,0,2],[-1,0,1]],dtype=np.float32)
20
21 img_x = convolve2D(img,sobel_v)
22 img_y = convolve2D(img,sobel_h)
23
24 grad_mag = np.sqrt(img_x**2 +img_y**2)
25
26
27 fig,ax = plt.subplots(1,4,figsize=(18,6))
28 ax[0].imshow(img,cmap='gray',vmin=0,vmax=255)
29 ax[1].imshow(img_x,cmap='gray',vmin=-1020,vmax=1020)
30 ax[2].imshow(img_y,cmap='gray',vmin=-1020,vmax=1020)
31 ax[3].imshow(grad_mag,cmap='gray',vmin=0,vmax=255)
32
33 ax[0].axis('off')
34 ax[1].axis('off')
35
36 ax[0].set_title("Original")
37 ax[1].set_title("Sobel Vertical")
38 ax[2].set_title("Sobel Horizontal")
39 ax[3].set_title("$\sqrt{img\_x^2 + img\_y^2}$")
40
41 plt.show()

```

Source Code 8: Sobel Filtering - 2

## Question 6 - (c)

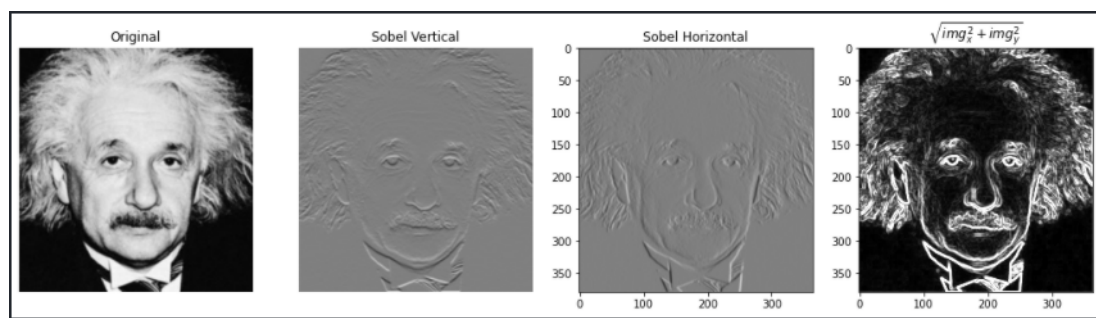


Figure 9: Sobel Filtering - 3



```

1 #Q6 - part C
2 img = cv.imread('einstein.png',cv.IMREAD_GRAYSCALE).astype('float32')
3 assert img is not None, "img Not Found!!"
4
5 sobel_h1 = np.array([[1],[2],[1]])
6 sobel_h2 = np.array([[1,0,-1]])
7
8 sobel_v1 = np.array([[1],[0],[-1]])
9 sobel_v2 = np.array([[1,2,1]])
10
11 img_yy= convolve2D(img,sobel_h1)
12 img_yy= convolve2D(img_yy , sobel_h2)
13
14 img_xx= convolve2D(img,sobel_v1)
15 img_xx= convolve2D(img_xx , sobel_v2)
16
17 grad_mag = np.sqrt(img_xx**2 + img_yy**2)
18
19 fig,ax = plt.subplots(1,4,figsize=(18,6))
20 ax[0].imshow(img,cmap='gray',vmin=0,vmax=255)
21 ax[1].imshow(img_xx,cmap='gray',vmin=-1020,vmax=1020)
22 ax[2].imshow(img_yy,cmap='gray',vmin=-1020,vmax=1020)
23 ax[3].imshow(grad_mag,cmap='gray',vmin=0,vmax=255)
24
25 ax[0].axis('off')
26 ax[1].axis('off')
27
28 ax[0].set_title("Original")
29 ax[1].set_title("Sobel Vertical")
30 ax[2].set_title("Sobel Horizontal")
31 ax[3].set_title(" $\sqrt{\text{img\_xx}^2 + \text{img\_yy}^2}$ ")
32
33 plt.show()

```

Source Code 9: Sobel Filtering - 3

## Question 7



Figure 10: GrabCut segmentation

```

1 #Q7
2 import numpy as np
3 import cv2 as cv
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6
7 img = cv.imread('daisy.jpg')
8 assert img is not None, "Image Not Found!!"
9
10 img_RGB = cv.cvtColor(img,cv.COLOR_BGR2RGB)
11
12 mask = np.zeros(img_RGB.shape[:2],np.uint8)
13 bgdModel = np.zeros((1,65),np.float64)
14 fgdModel = np.zeros((1,65),np.float64)
15 rect = (50,50,500,500)
16
17 (mask, bgdModel, fgdModel) = cv.grabCut(img_RGB,mask,rect,bgdModel,fgdModel,5,cv.GC_INIT_WITH_RECT)
18 mask2 = np.where((mask==cv.GC_BGD)|(mask==cv.GC_PR_BGD),0,1).astype('uint8')
19 fmask = (mask == cv.GC_PR_FGD).astype("uint8") * 255
20 flower = img_RGB*mask2[:, :, np.newaxis]
21
22 print("a: ",cv.GC_FGD,"b: ",cv.GC_PR_FGD)
23 bmask = (mask == cv.GC_PR_BGD).astype("uint8") * 255
24 outMask = (np.where((mask == cv.GC_FGD) | (mask == cv.GC_PR_FGD), 0, 1)*255).astype(np.uint8)
25 background = cv.bitwise_and(img, img, mask=outMask) # Background Image
26
27 fig,ax = plt.subplots(1,4,figsize=(15,6))
28 ax[0].imshow(img_RGB)
29 ax[1].imshow(flower)
30 ax[2].imshow(cv.cvtColor(background,cv.COLOR_BGR2RGB))
31 ax[3].imshow(cv.cvtColor(fmask,cv.COLOR_BGR2RGB))
32
33 ax[0].axis('off')
34 ax[1].axis('off')
35 ax[2].axis('off')
36 ax[3].axis('off')
37
38 ax[0].set_title("Original")
39 ax[1].set_title("Flower")
40 ax[2].set_title("Background")
41 ax[3].set_title("Mask")
42
43 background_blur =cv.GaussianBlur(background, (9,9), 4)
44 re_created = cv.add(cv.cvtColor(flower,cv.COLOR_RGB2BGR), background)
45 enhanced = cv.add(cv.cvtColor(flower,cv.COLOR_RGB2BGR), background_blur)
46
47 fig,ax = plt.subplots(1,4,figsize=(15,7))
48
49 ax[0].imshow(cv.cvtColor(img,cv.COLOR_BGR2RGB))
50 ax[1].imshow(cv.cvtColor(re_created,cv.COLOR_BGR2RGB))
51 ax[2].imshow(cv.cvtColor(enhanced,cv.COLOR_BGR2RGB))
52 ax[3].imshow(cv.cvtColor(background_blur,cv.COLOR_BGR2RGB))
53
54 ax[0].axis('off')
55 ax[1].axis('off')
56 ax[2].axis('off')
57 ax[3].axis('off')
58
59 ax[0].set_title("Original")
60 ax[1].set_title("Re-created Image")
61 ax[2].set_title("Enhanced Image")
62 ax[3].set_title("Gaussian Blurred Background")
63
64 plt.show()

```

Source Code 10: GrabCut segmentation

When we are going to enhance the image, we blurred the background using gaussianBlur with a 9,9 kernel. Because of that the rough edges of the flower shape are smoothened. Therefore, background just beyond the edge of the being flower quite dark in the enhanced image. We can clearly see the effect in the above pictures.

Link for GitHub Repository : - <https://github.com/limalkasadith/S4-EN2550-Workspace.git>