EN2550 - Fundamentals of Image Processing and Machine Vision

Assignment 1

Submitted by

Chathumini B.G.D.T. - 190107T

Submitted on

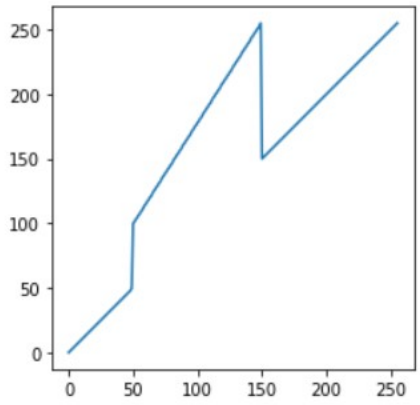
March 5, 2022

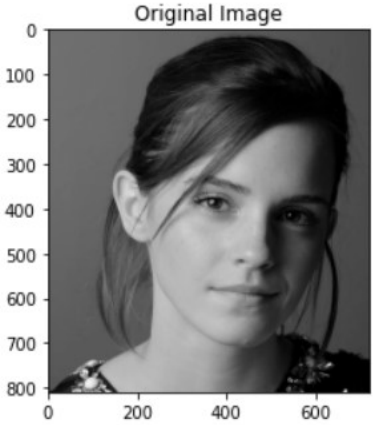
GitHub Repository link

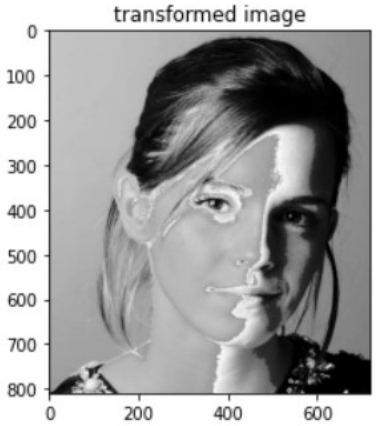
`https://github.com/dulmi-19/Image-Processing-and-Machine-Vision`

Question 1

```
# linearly spaced values
t1=np.linspace(0,50,50)
t2=np.linspace(100,255,100)
t3=np.linspace(150,255,106)

transform = np.concatenate((t1,t2,t3), axis = 0).astype(np.uint8)  #connect all 3 transformations
assert len(transform) == 256
emma_transformed= cv.LUT(emma,transform)
```



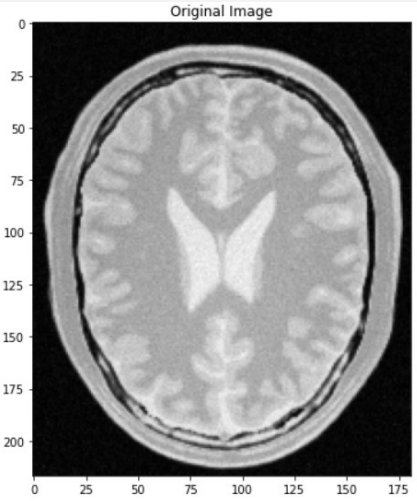


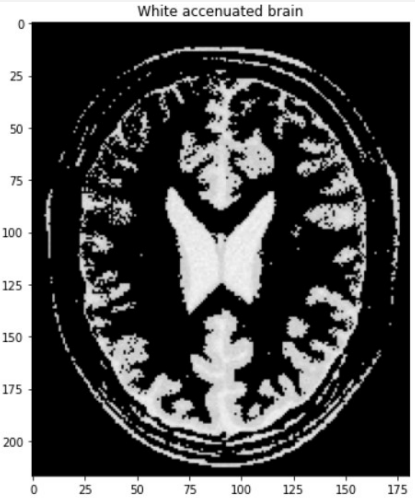


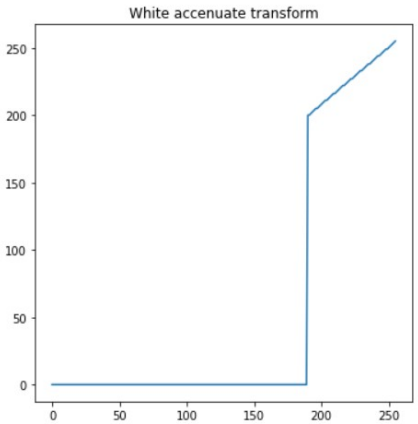
The intensity transformation is implemented using a lookup table. In this transformation the lower and higher grey values of input image are unchanged, but the middle values in (50-100) range are changed to (100-255) range

Question 2 (a) white matter

```
# accenuate white
p1=np.linspace(0,0,190)
p2=np.linspace(200,255,66)
accenuate_white = np.concatenate((p1,p2), axis = 0).astype(np.uint8)  #connect all transformations
assert len(accenuate_white) == 256
white_accenuated_brain = cv.LUT(brain,accenuate_white)
```



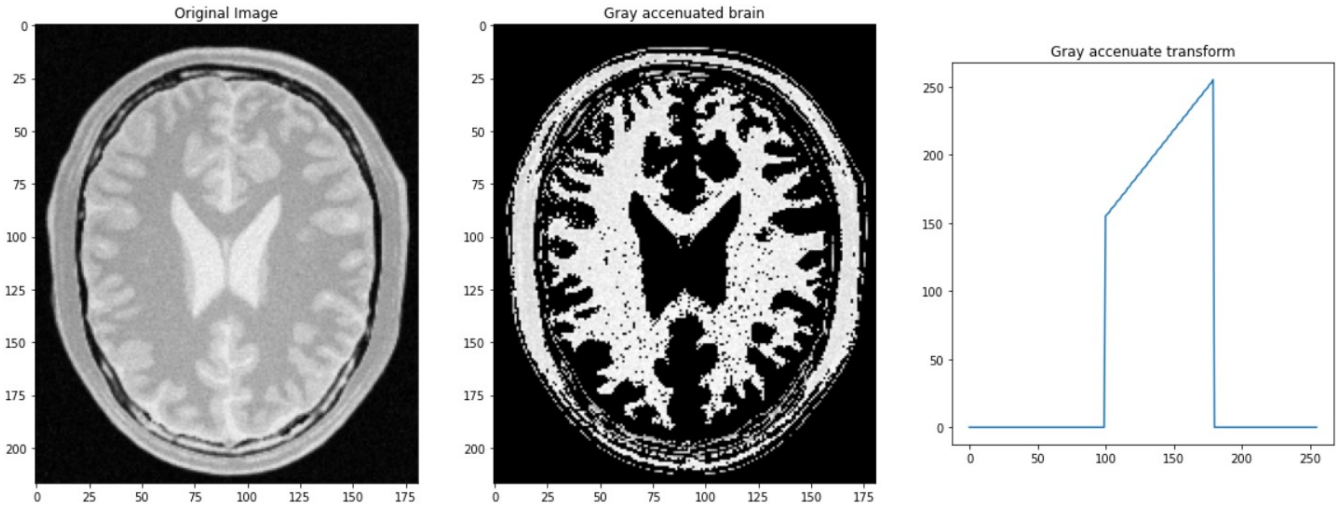




To accentuate white matter, the intensity of color values near white color(255) are increased. Because of the gradient, we can compare the intensity variation as well.

Question 2 (b) gray matter

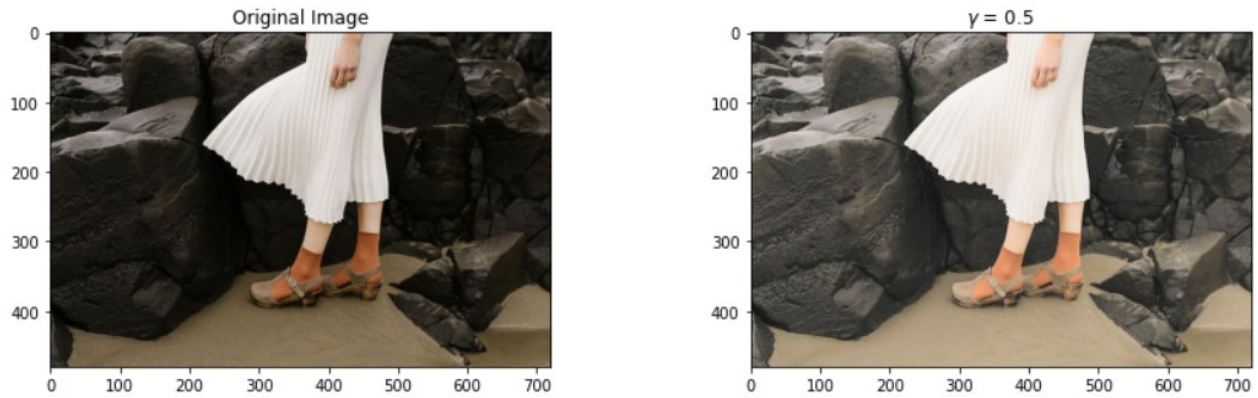
```
# accenuate Grey
t1=np.linspace(0,0,100)
t2=np.linspace(155,255,80)
t3=np.linspace(0,0,76)
accenuate_grey = np.concatenate((t1,t2,t3), axis = 0).astype(np.uint8)  #connect all transformations
assert len(accenuate_grey) == 256
grey_accenuated_brain = cv.LUT(brain,accenuate_grey)
```



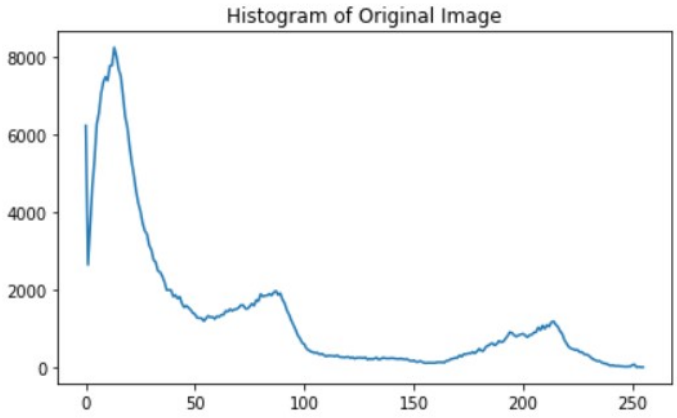
To accentuate gray matter, the intensity of middle range color values are increased while keeping the other color values at 0. Because of the gradient, we can compare the intensity variation as well.

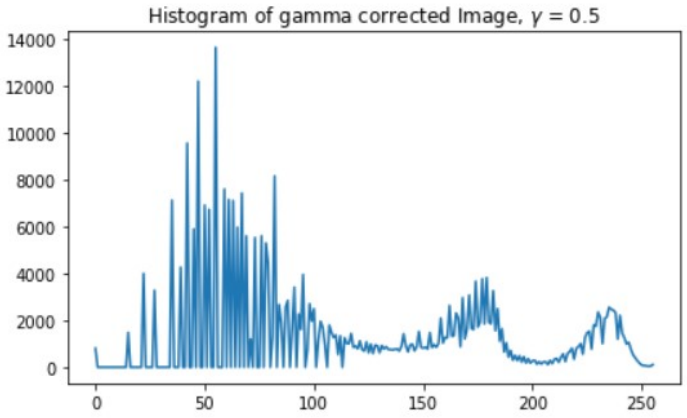
Question 3

```
im_lab = cv.cvtColor(im, cv.COLOR_BGR2Lab)
(l,a,b)= cv.split(im_lab) # split the l,a,b components
gamma = 0.5
g_transform = np.array([(p/255)**gamma*255 for p in range(0,256)]).astype(np.uint8)
g_transformed = cv.LUT(l, g_transform) # gamma correction for lightness
im_lab[:,:,0]=g_transformed

im_hist = cv.calcHist([im],[0],None,[256],[0,256])
im_lab_hist = cv.calcHist([im_lab],[0],None,[256],[0,256])
```



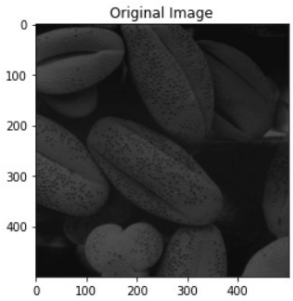
The gamma transform is applied to L plane of CIELAB color space. Therefore, the details in darker areas are more visible in the gamma transformed image.

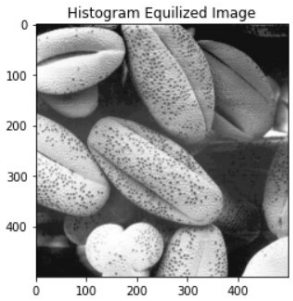


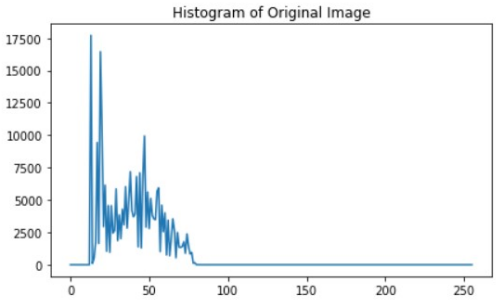


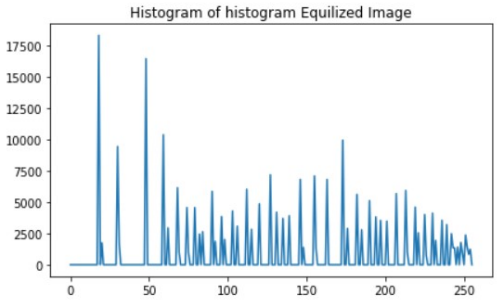
Question 4

```
shells =  cv.imread("shells.png",cv.IMREAD_GRAYSCALE)
assert shells is not None
shells_hist = cv.calcHist([shells],[0],None,[256],[0,256])

#normalize
n_pixels = np.sum(shells_hist)
shells_histo = shells_hist/n_pixels

#normalized cumulative histogram
cdf_hist = np.cumsum(shells_histo)

transform = np.floor(255 * cdf_hist).astype(np.uint8)
eq_shells = cv.LUT(shells,transform)
equi_shells_histo = cv.calcHist([eq_shells ],[0],None,[256],[0,256])
```









According to the histogram of original image, we can see that the color values are concentrated in the lower region of values. After the histogram equalization, the intensity values are distributed all over the region,therefore we can see the details of the image clearly.In here, we are using discrete intensity values instead of continuous values.

Question 5

```python
def zoom_im(image, method, scaling_fact):
    rows, cols =scaling_fact* image.shape[0], scaling_fact* image.shape[1]
    zoomed_img = np.zeros((rows,cols,3),dtype=image.dtype)
    if (method == 0):
        #nearest-neighbor method
        for r in range(rows):
            for c in range(cols):
                im_r, im_c= int(np.round(r/scaling_fact)), int(np.round(c/scaling_fact))
                if im_r >= image.shape[0]:
                    im_r -=1
                if im_c >= image.shape[1]:
                    im_c -=1
                zoomed_img[r][c]= image[im_r][im_c]
        return zoomed_img
    else:
        #bilinear interpolation method
        for r in range(rows-1):
            for c in range(cols-1):
                im_r, im_c= r/scaling_fact, c/scaling_fact
                left_bottom = [math.floor(im_r), math.floor(im_c)]
                right_bottom = [math.floor(im_r), math.ceil(im_c)]
                left_top = [math.ceil(im_r), math.floor(im_c)]
                right_top = [math.ceil(im_r),math.ceil(im_c)]

                if left_top[0] >= image.shape[0]:
                    left_top[0] =image.shape[0] - 1
                    right_top[0] =image.shape[0] - 1
                if right_bottom[1] >= image.shape[1]:
                    right_bottom[1]=image.shape[1] - 1
                    right_top[1] =image.shape[1] - 1


                ratio_vertical = im_r-left_bottom[0]
                ratio_horizontal = im_c-left_bottom[1]

                horizontal_1 =ratio_vertical*image[left_top[0]][left_top[1]] +(1-ratio_vertical)*image[left_bottom[0]][left_bottom[1]]
                horizontal_2 =ratio_vertical*image[right_top[0]][right_top[1]] +(1-ratio_vertical)*image[right_bottom[0]][right_bottom[1]]
                zoomed_img[r][c]= np.rint(horizontal_1*(1-ratio_horizontal) + horizontal_2*ratio_horizontal)
        return zoomed_img
```

```
nn_ssd=np.sum((org_img_2[:,:,0:3]-nn_2[:,:,0:3])**2)/org_img_2.size
bi_ssd=np.sum((org_img_2[:,:,0:3]-bi_2[:,:,0:3])**2)/org_img_2.size
print("SSD for nearest-neighbor", nn_ssd)
print("SSD for bilinear interpolation", bi_ssd)
```

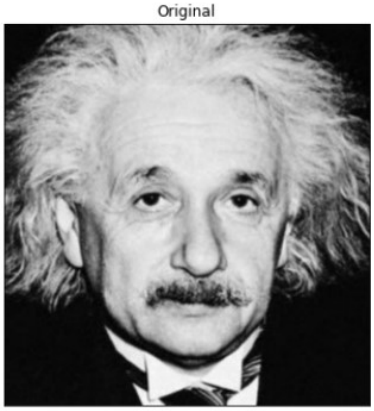
✓ 0.2s

SSD for nearest-neighbor 16.792970920138888
SSD for bilinear interpolation 16.40395240162037

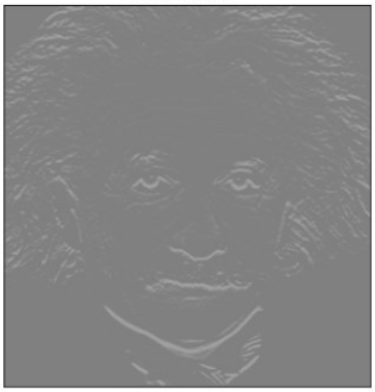
Question 6 (a)

```
einstein = cv.imread("einstein.png",cv.IMREAD_GRAYSCALE)
assert einstein is not None

sobel_v = np.array([(-1,-2,-1),(0,0,0),(1,2,1)],dtype=np.float32)
einstein_x = cv.filter2D(einstein,-1,sobel_v)
sobel_h = np.array([(-1,0,1),(-2,0,2),(-1,0,1)],dtype=np.float32)
einstein_y = cv.filter2D(einstein,-1,sobel_h)
```

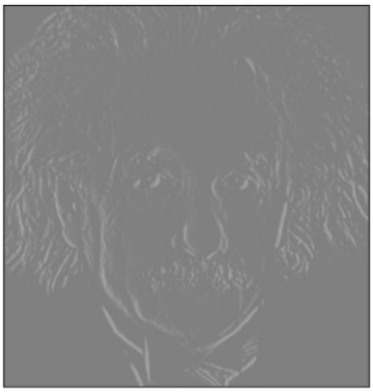
Original



sobel vertical

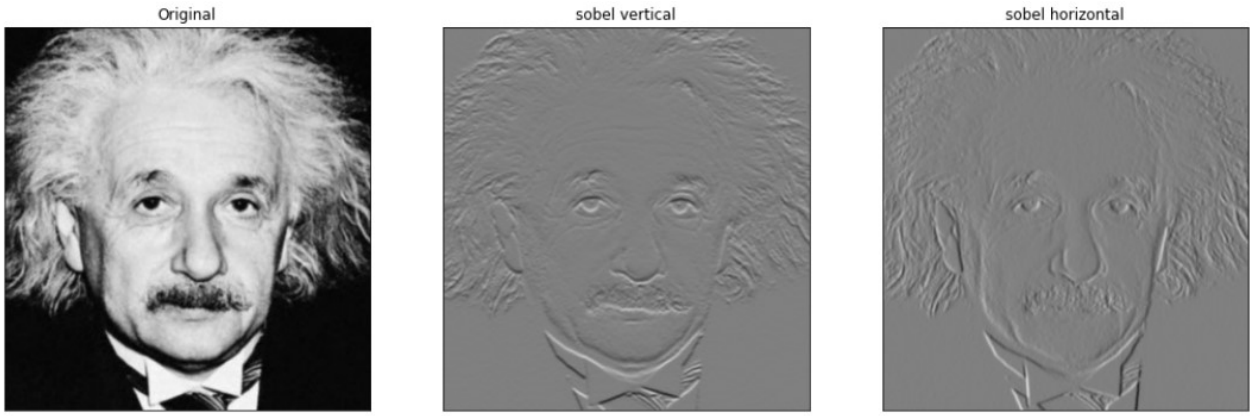


sobel horizontal



Question 6 (b)

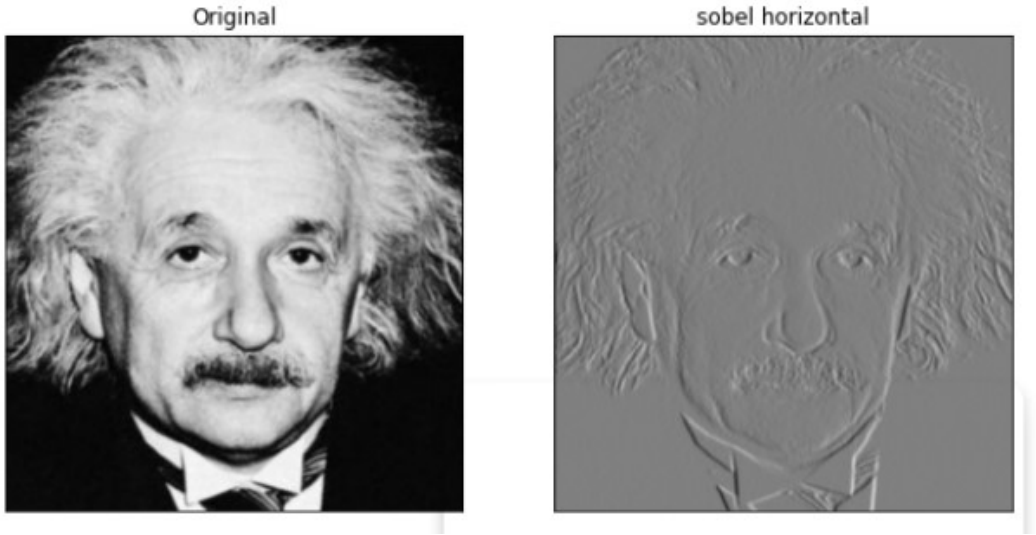
```
def sobel_filter(image,kernal):

    [r,c]= image.shape
    filtered_image = np.zeros((r,c),dtype=np.float32)
    padded_array = np.zeros((r+2,c+2),dtype=np.float32)
    padded_array[1:r+1,1:c+1] = image
    n=1
    m=1
    height_kernal=math.floor(kernal.shape[0]/2)
    if len(kernal.shape)==1:
        width_kernal=0
        m=2
    else:
        width_kernal=math.floor(kernal.shape[1]/2)
        if kernal.shape[1]==1:
            m=2
    if kernal.shape[0]==1:
        n=2


    for row in range(height_kernal,r+2-height_kernal-n):
        for col in range(width_kernal,c+2-width_kernal-m):
            filtered_image[row,col]=np.dot(padded_array[row-height_kernal:row+height_kernal+1,col-width_kernal:col+width_kernal+1].flatten(),kernal.flatten())

    return filtered_image

image_v =sobel_filter(einstein,sobel_v )
image_h =sobel_filter(einstein,sobel_h )
```



The outputs of filter2D filter and the user defined code are similar. In the user defined code, some depth details can be seen.

Question 6 (c)

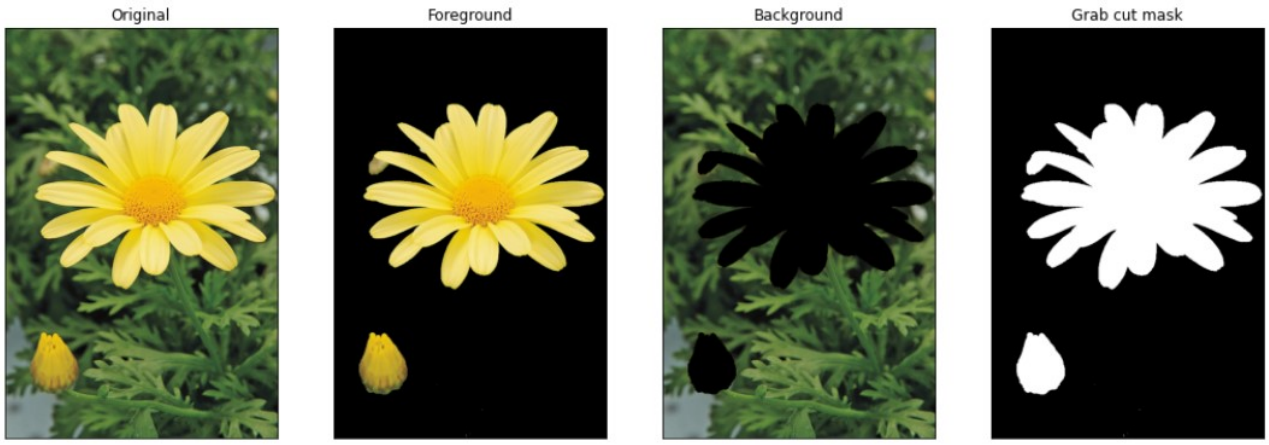
```
sobel_h1=np.array([(1),(2),(1)] ,dtype=np.float32)
sobel_h2=np.array([(1,0,-1)] ,dtype=np.float32)

filltered_1=sobel_filter(einstein,sobel_h1)
Filtered_2=sobel_filter(filltered_1,sobel_h2)
```



The output is same, but computational complexity decreases in this method.

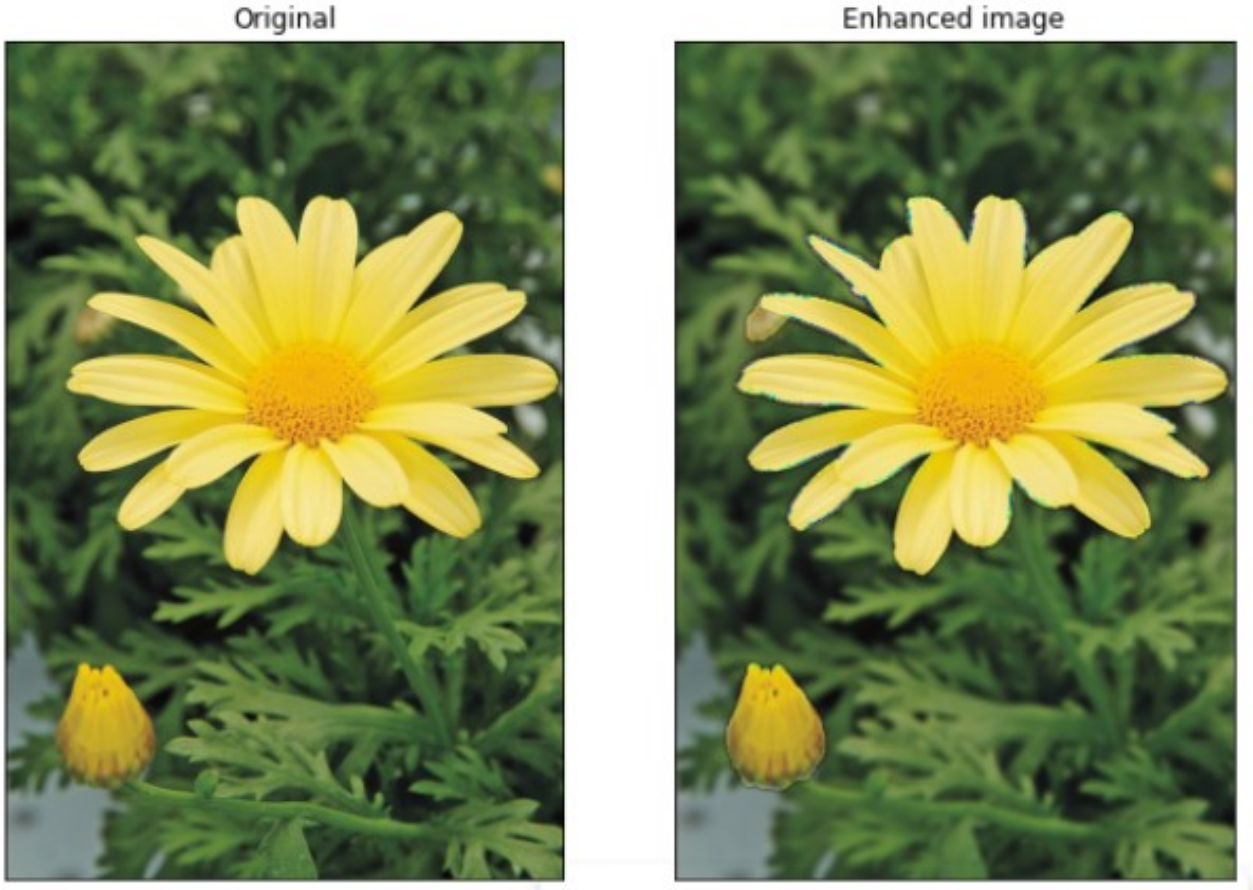
Question 7 (a)

```
mask = np.zeros(daisy.shape[:2],dtype=daisy.dtype)
rectangle= (40,40,540,800)
background_mask = np.zeros((1, 65), np.float64)
foreground_mask = np.zeros((1, 65), np.float64)

cv.grabCut(daisy, mask, rectangle, background_mask, foreground_mask, 15, cv.GC_INIT_WITH_RECT)

grab_cut_mask = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')
daisy_fg = daisy*grab_cut_mask[:,:,np.newaxis]
daisy_bg=np.subtract(daisy,daisy_fg)
```



Question 7 (b)

```
k_size = 9
sigma = 4
blurred_bg =cv.GaussianBlur(daisy_bg,(k_size,k_size),sigma)

enhanced_daisy=np.add(daisy_fg,blurred_bg)
```



Question 7 (c)

When Gaussian blur is applied to the background image, the green and black pixels near the edge of the flower get mixed.