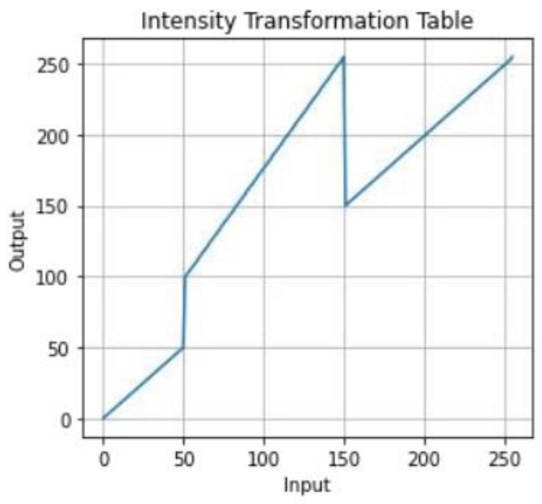
EN2550 – Assignment 01

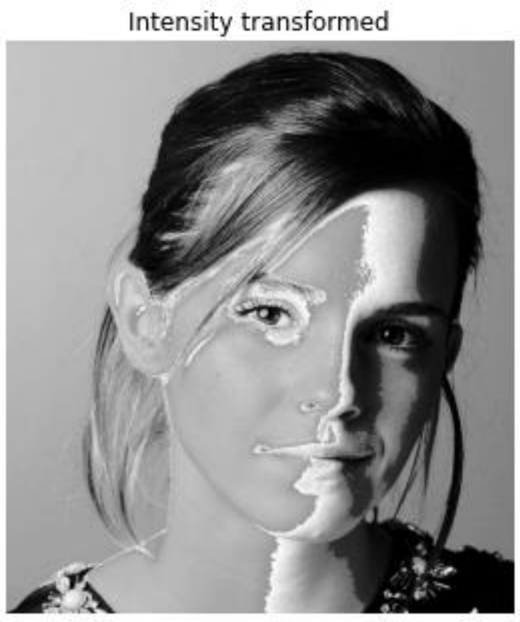
Name: Abeywansa C.N.

Index: 190005E

Question-1

In this question, ask for intensity transformation of the given image. As shown in the intensity transformation table, the output intensity values are changed to higher intensity values between 50 and 150 input intensity range. Therefore, we can see a clear intensity change of the face (in the intensity transformed image) when compare with the original image.

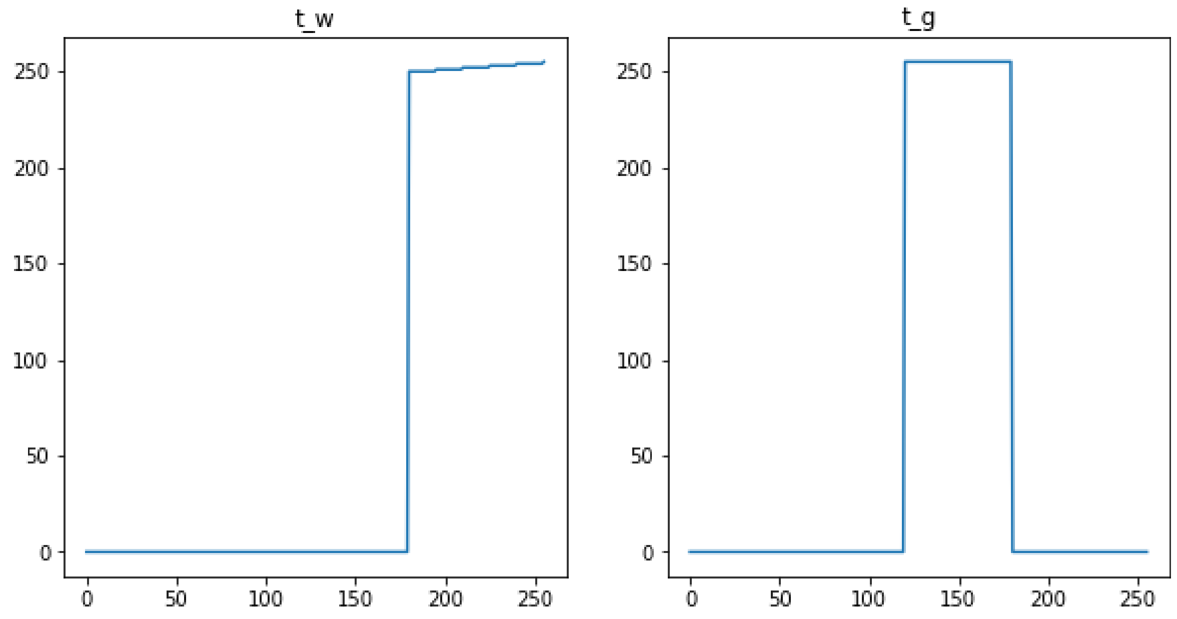


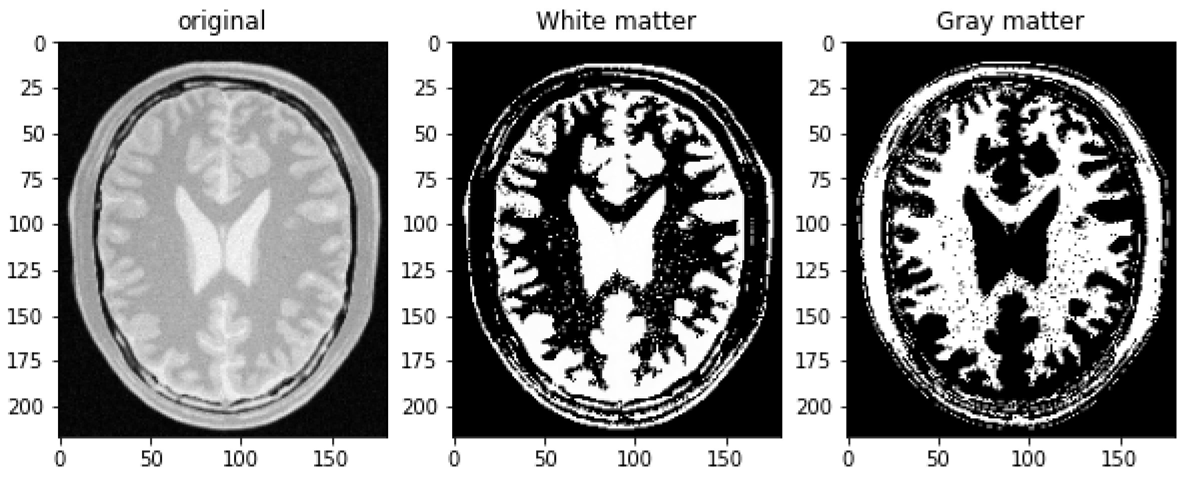


Question-2

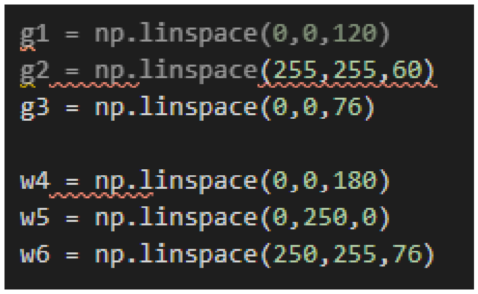
Same as the question 1, here also use intensity transformations to separate white matter and the gray matter of the brain.

Here ,by guessing several values and checking the output ,the more appropriate values have been chosen.





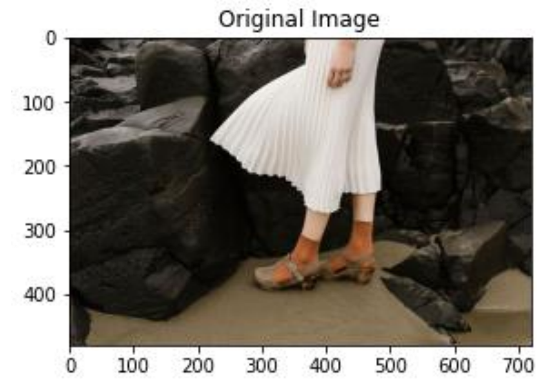
To accentuate the white matter and to accentuate the gray matter, the ranges are mapped as follows.

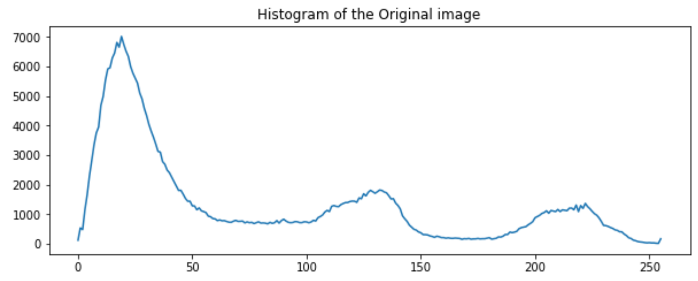


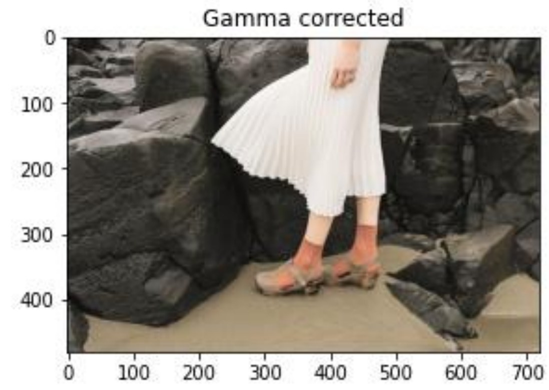
```
g1 = np.linspace(0,0,120)
g2 = np.linspace(255,255,60)
g3 = np.linspace(0,0,76)

w4 = np.linspace(0,0,180)
w5 = np.linspace(0,250,0)
w6 = np.linspace(250,255,76)
```

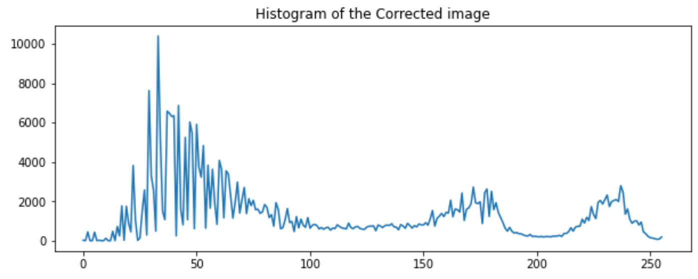
<u>Question-3</u>

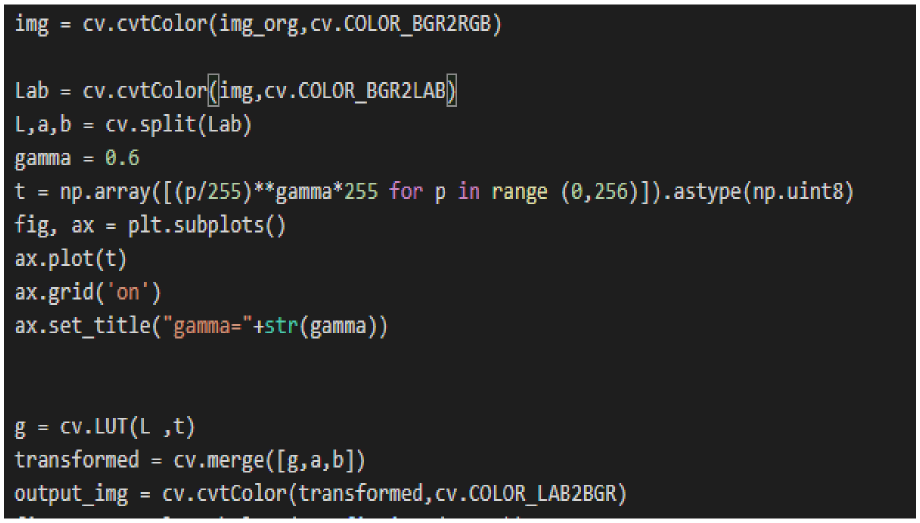
This question is about the gamma correction of an image. An image has a LAB color space which represents lightness(L), color range of green-magenta (A), color range of blue-yellow (B).Here gamma correction is on the L plane.









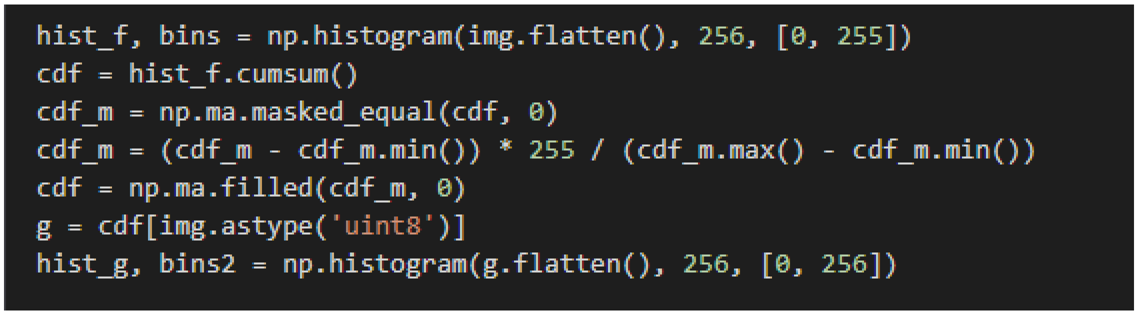


```
img = cv.cvtColor(img_org,cv.COLOR_BGR2RGB)

Lab = cv.cvtColor(img,cv.COLOR_BGR2LAB)
L,a,b = cv.split(Lab)
gamma = 0.6
t = np.array([(p/255)**gamma*255 for p in range (0,256)]).astype(np.uint8)
fig, ax = plt.subplots()
ax.plot(t)
ax.grid('on')
ax.set_title("gamma="+str(gamma))


g = cv.LUT(L ,t)
transformed = cv.merge([g,a,b])
output_img = cv.cvtColor(transformed,cv.COLOR_LAB2BGR)
```

First, convert the image to LAB color space from BGR. Then split the 3 planes L,a,b by using cv.split(image) function. I choose gamma value as 0.6 and ,assign the corrected values by lookup table cv.LUT() and again merged to unchanged a,b planes cv.merge() and create 3D array.

Question-4

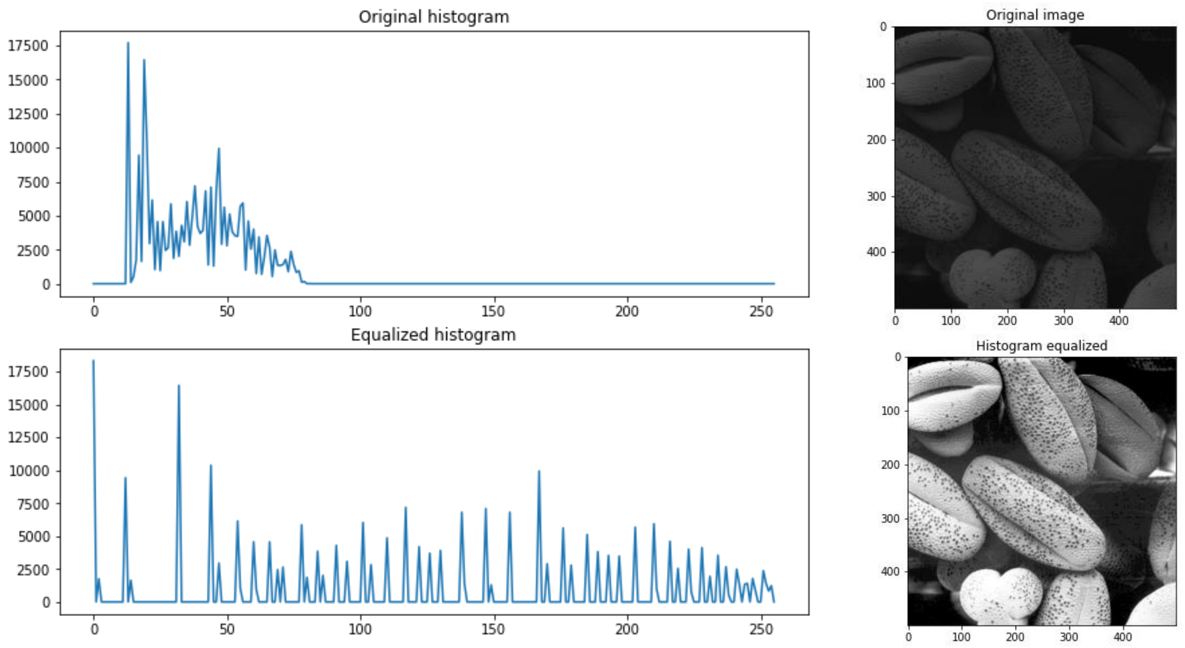
Histogram equalization is used for getting more vibrant images. OpenCV has inbuilt function cv.equalizeHist() to equalizing the histograms. But here, histogram equalization function is developed manually, and it is as follows.



```
hist_f, bins = np.histogram(img.flatten(), 256, [0, 255])
cdf = hist_f.cumsum()
cdf_m = np.ma.masked_equal(cdf, 0)
cdf_m = (cdf_m - cdf_m.min()) * 255 / (cdf_m.max() - cdf_m.min())
cdf = np.ma.filled(cdf_m, 0)
g = cdf[img.astype('uint8')]
hist_g, bins2 = np.histogram(g.flatten(), 256, [0, 256])
```

The histogram of the image is calculated as a cumulative sum (cdf) and normalized it by dividing from the difference of the highest and lowest values and multiplying by 255.

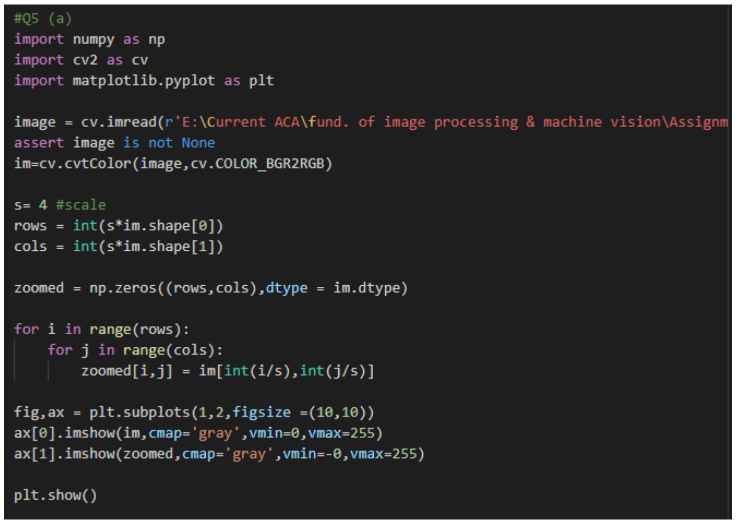
The following figures shows the histograms of original and equalized images histograms and relevant images.



Question-5

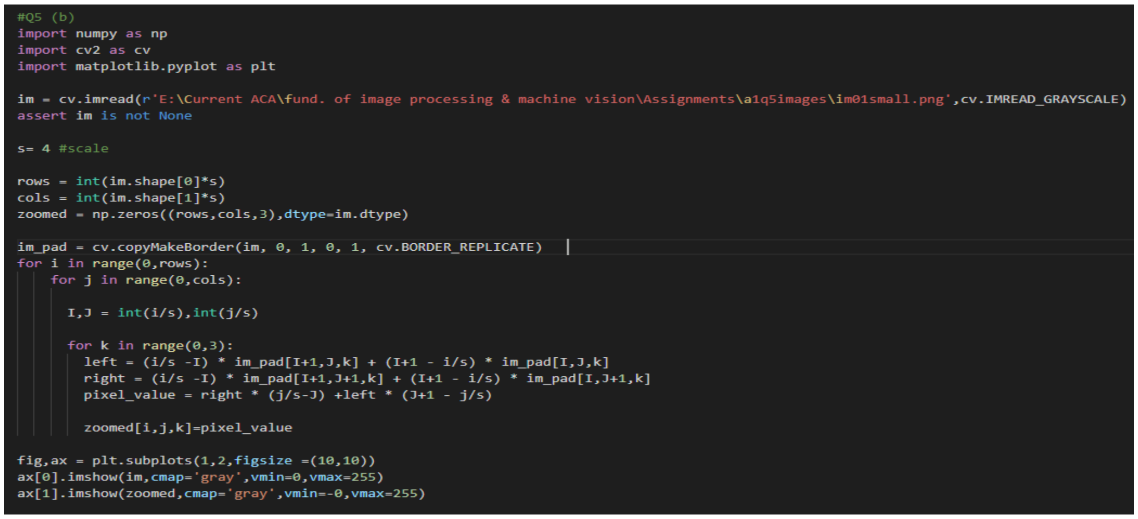
Nearest neighbor and bilinear interpolation are used for zooming an image.

(a)nearest neighbor - the pixel value of the zoomed image is matched to the value of the pixel, which is more closer to it.



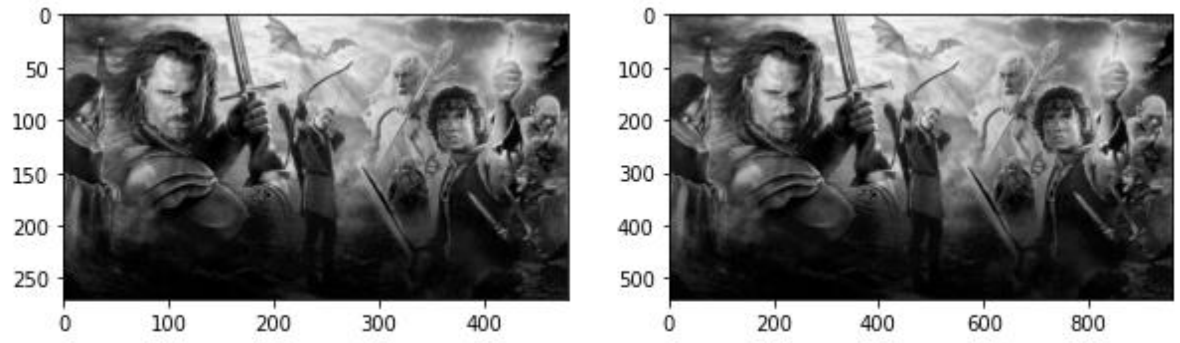
```
#Q5 (a)
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

image = cv.imread(r'E:\Current ACA\fund. of image processing & machine vision\Assignm
assert image is not None
im=cv.cvtColor(image,cv.COLOR_BGR2RGB)

s= 4 #scale
rows = int(s*im.shape[0])
cols = int(s*im.shape[1])

zoomed = np.zeros((rows,cols),dtype = im.dtype)

for i in range(rows):
    for j in range(cols):
        zoomed[i,j] = im[int(i/s),int(j/s)]

fig,ax = plt.subplots(1,2,figsize =(10,10))
ax[0].imshow(im,cmap='gray',vmin=0,vmax=255)
ax[1].imshow(zoomed,cmap='gray',vmin=-0,vmax=255)

plt.show()
```

(b)bilinear interpolation - a linear interpolation is done to get the values of the pixels. So, a pixel value in zoomed image, which will be a value in the original image corresponding pixel values around it.



```
#Q5 (b)
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

im = cv.imread(r'E:\Current ACA\fund. of image processing & machine vision\Assignments\a1q5images\im01small.png',cv.IMREAD_GRAYSCALE)
assert im is not None

s= 4 #scale

rows = int(im.shape[0]*s)
cols = int(im.shape[1]*s)
zoomed = np.zeros((rows,cols,3),dtype=im.dtype)

im_pad = cv.copyMakeBorder(im, 0, 1, 0, 1, cv.BORDER_REPLICATE)
for i in range(0,rows):
    for j in range(0,cols):

      I,J = int(i/s),int(j/s)

      for k in range(0,3):
        left = (i/s -I) * im_pad[I+1,J,k] + (I+1 - i/s) * im_pad[I,J,k]
        right = (i/s -I) * im_pad[I+1,J+1,k] + (I+1 - i/s) * im_pad[I,J+1,k]
        pixel_value = right * (j/s-J) +left * (J+1 - j/s)

        zoomed[i,j,k]=pixel_value

fig,ax = plt.subplots(1,2,figsize =(10,10))
ax[0].imshow(im,cmap='gray',vmin=0,vmax=255)
ax[1].imshow(zoomed,cmap='gray',vmin=-0,vmax=255)
```

By comparing the two SSD values in the two methods, it can be seen that the bilinear interpolation method gives bit lower error than the nearest neighbor method.

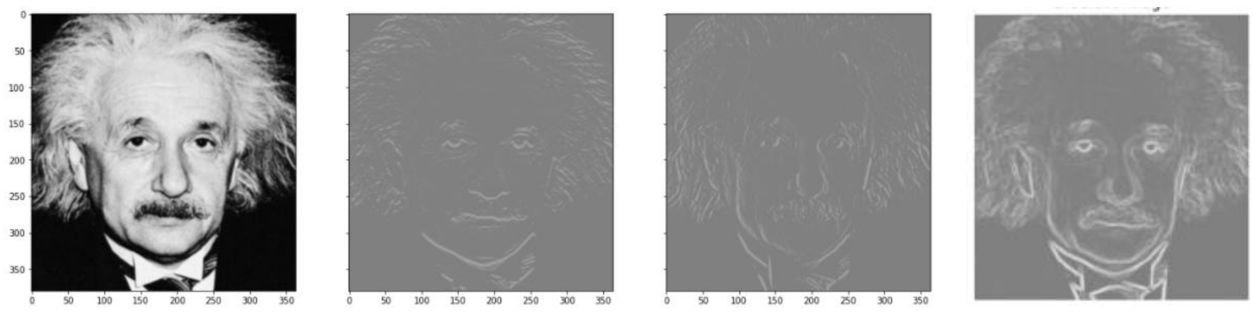


Question-6

The Sobel operator is used for edge detection in image processing.

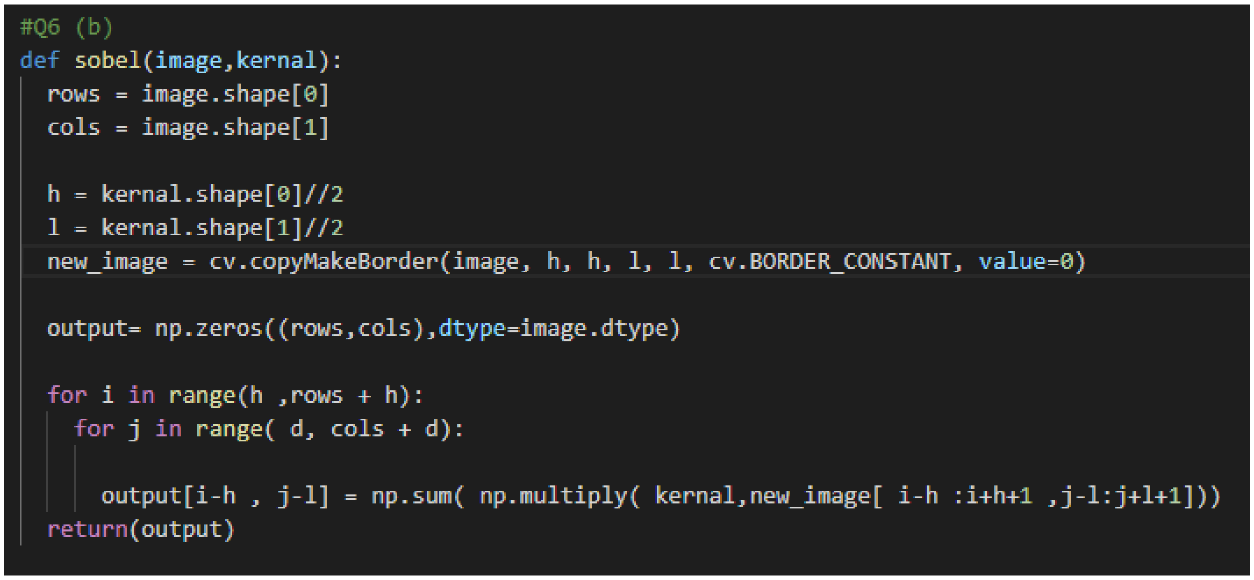
The Sobel vertical filter is used to detect horizontal edges, and the Sobel horizontal filter is used to detect vertical edges. The gradient of the image can be taken from the magnitude matrix of the outputs of these 2 filters.

OpenCV inbuilt function cv.filter2D() was used for getting the outputs of the filters.



In the above figures, original image, sobel vertical, sobel horizontal and gradient images are shown in the order.

Next, A code is designed for Sobel filters manually. The image, and the 3x3 kernel will be given to this function. Inside the nested for loops, the convolution is taken manually. The previously defined Sobel vertical and Sobel horizontal filters will be given.



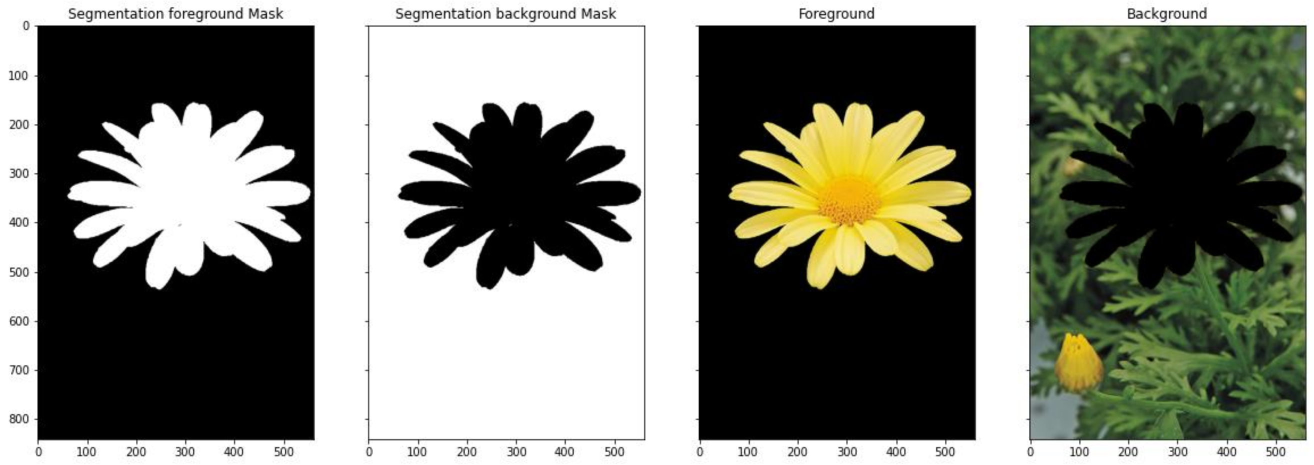
```
#Q6 (b)
def sobel(image,kernal):
  rows = image.shape[0]
  cols = image.shape[1]

  h = kernal.shape[0]//2
  l = kernal.shape[1]//2
  new_image = cv.copyMakeBorder(image, h, h, l, l, cv.BORDER_CONSTANT, value=0)

  output= np.zeros((rows,cols),dtype=image.dtype)

  for i in range(h ,rows + h):
    for j in range( d, cols + d):

      output[i-h , j-l] = np.sum( np.multiply( kernal,new_image[ i-h :i+h+1 ,j-l:j+l+1]))
  return(output)
```

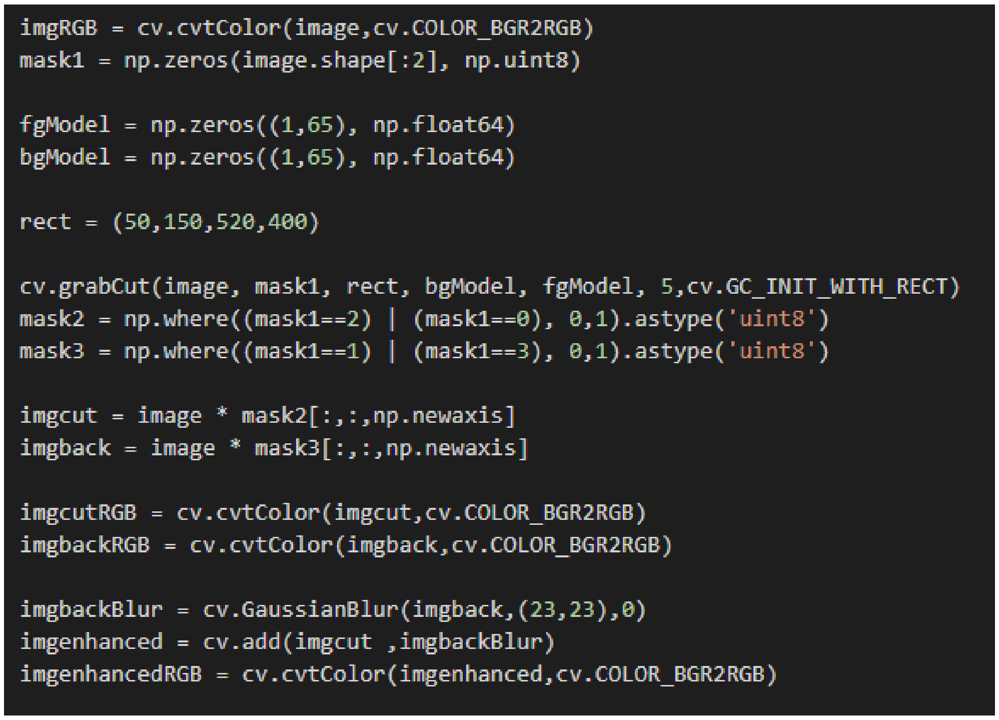
Finally, without defining sobel vertical or sobel horizontal filters we tried to do the sobel filtering. Although in the previous part, a 3x3 matrix is used to convolute the image, here it is convolved with 1x3 matrix and 3x1 matrix separately. But this filtering is also closer to the sobel vertical filtering.

<u>Question-7</u>

Grabcut is an algorithm which was developed for foreground extraction by judging the color distribution, with the use of a Gaussian Mixture Model (GMM).

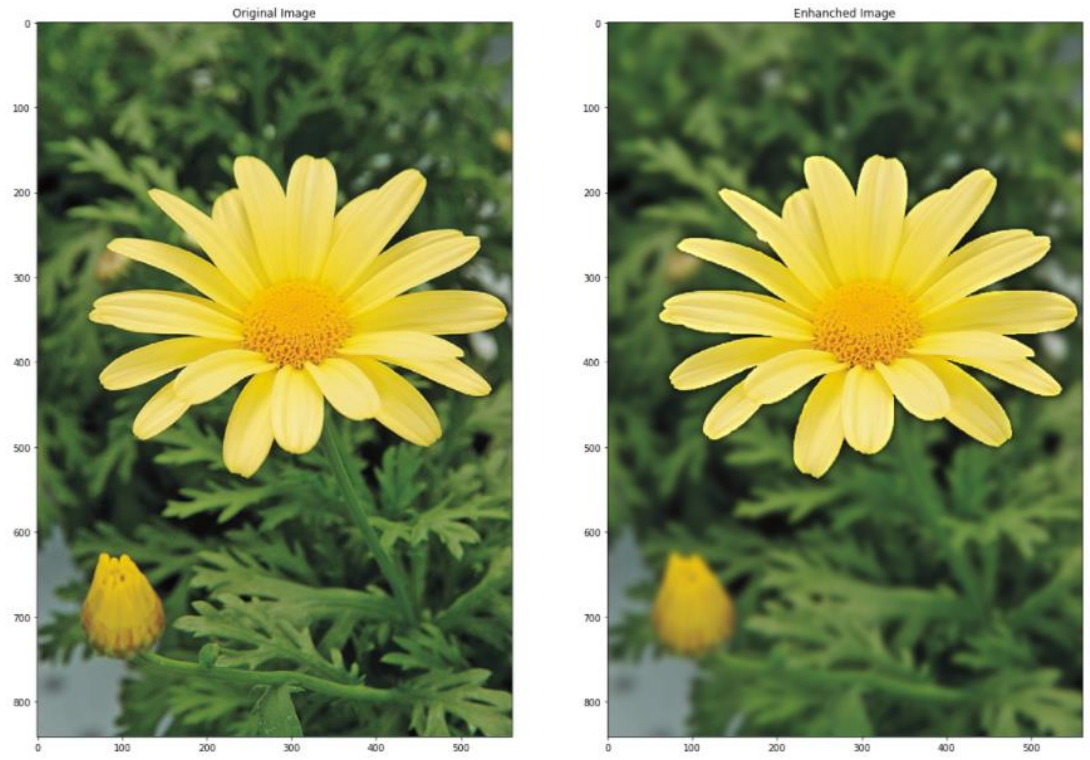
In the first part, using cv.Grabcut() function following models are carried out.





```
imgRGB = cv.cvtColor(image,cv.COLOR_BGR2RGB)
mask1 = np.zeros(image.shape[:2], np.uint8)

fgModel = np.zeros((1,65), np.float64)
bgModel = np.zeros((1,65), np.float64)

rect = (50,150,520,400)

cv.grabCut(image, mask1, rect, bgModel, fgModel, 5,cv.GC_INIT_WITH_RECT)
mask2 = np.where((mask1==2) | (mask1==0), 0,1).astype('uint8')
mask3 = np.where((mask1==1) | (mask1==3), 0,1).astype('uint8')

imgcut = image * mask2[:,:,np.newaxis]
imgback = image * mask3[:,:,np.newaxis]

imgcutRGB = cv.cvtColor(imgcut,cv.COLOR_BGR2RGB)
imgbackRGB = cv.cvtColor(imgback,cv.COLOR_BGR2RGB)

imgbackBlur = cv.GaussianBlur(imgback,(23,23),0)
imgenhanced = cv.add(imgcut ,imgbackBlur)
imgenhancedRGB = cv.cvtColor(imgenhanced,cv.COLOR_BGR2RGB)
```

Here, convert color plane to RGB at first and The background model and the foreground model are given as arrays of zeros. After getting the mask, foreground, and the background, the background image is blurred using the cv.GaussianBlur() function. At last, to get the enhanced image, the blurred background image is added with the foreground image. By checking the following images, it can be clearly seen the background is blurred on the enhanced image and the borders of the flower become bit darker.



<u>GitHub link</u>: https://github.com/chalindunis