DEPARTMENT OF ELECTRONICS AND
TELECOMMUNICATION ENGINEERING
UNIVERSITY OF MORATUWA


EN2550 FUNDAMENTALS OF IMAGE
PROCESSING & MACHINE VISION

---

# ASSIGNMENT 01 : Intensity Transformations and Neighborhood Filtering

---

MUNASINGHE M.M.R.H.
190399L
Github : Click Here

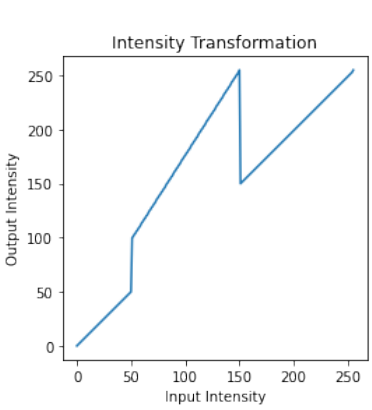
This is submitted as a partial fulfilment for the module EN2550.

---

March 5, 2022

(1) In this intensity transformation, the only intensity of input intensity range from 50 to 150 will increase other input intensities remain the same.

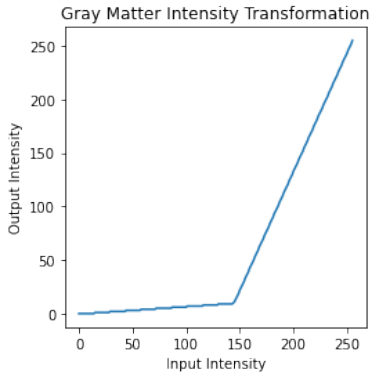
```
inp1 = np.linspace(0,50,51)
inp2 = np.linspace(100,255,100)
inp3 = np.linspace(150,255,105)
Out_int = np.concatenate((inp1,inp2,inp3),axis=0).astype(np.uint8)
assert len(Out_int) == 256
Out_Img1 = cv.LUT(Img1, Out_int)
```

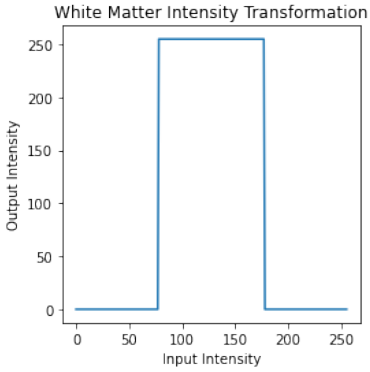


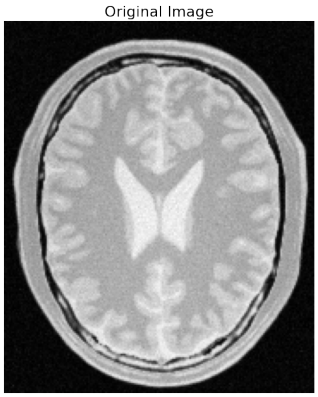


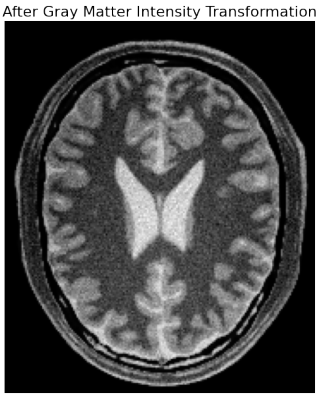

(2) To accentuate grey matter and white matter, we have to increase intensity as shown in the intensity transformation plots.

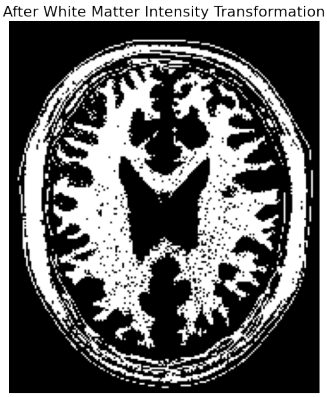
```
#Gray Matter
inpg1 = np.linspace(0, 10, 145)
inpg2 = np.linspace(11, 255, 111)
Out_int = np.concatenate((inpg1,inpg2),axis=0).astype(np.uint8)
assert len(Out_int) == 256
Out_Img1 = cv.LUT(Img2, Out_int)
#White Matter
inpw1= np.zeros(78)
inpw2 = 255*np.ones(100)
inpw3= np.zeros(78)
Out_int = np.concatenate((inpw1, inpw2, inpw3),axis=0).astype(np.uint8)
assert len(Out_int) == 256
Out_Img2 = cv.LUT(Img2, Out_int)
```



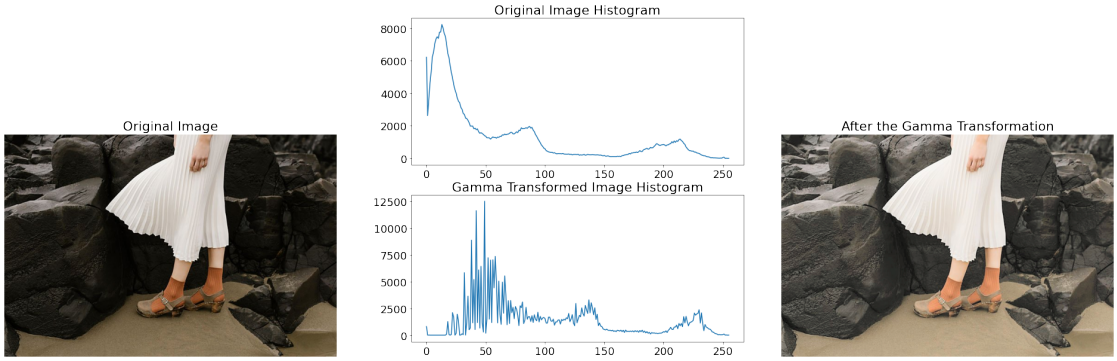



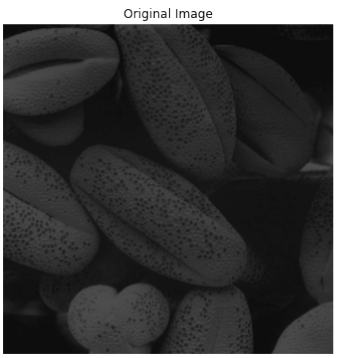

(3) Gamme = 0.5. When Comparing two images dark parts of the transformed image is more brighter than original image. Also we can spot low values of the distribution is lower in gamma transformed histogram compared to original histogram.

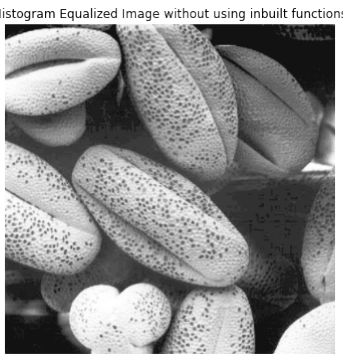
```
#Gamma Correction
gamma = 0.5
Lab_Img3 = cv.cvtColor(Img3, cv.COLOR_BGR2Lab)
L,a,b=cv.split(Lab_Img3)
Gamma_Trans = np.array([(p/255)**gamma*255 for p in range(0,256)]).astype(np.uint8)
L_trans =  cv.LUT(L,Gamma_Trans)
Lab_Img3[:,:,0] = L_trans
#Histograms of the images
hist_Input = cv.calcHist([Img3], [0], None, [256], [0,256])
hist_Output = cv.calcHist([Out_Img3], [0], None, [256], [0,256])
```

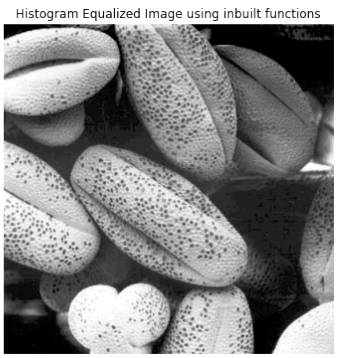


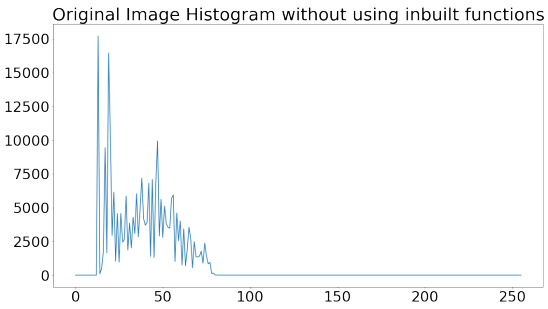

(4) We are applying histogram equalization to improve contrast of the image.

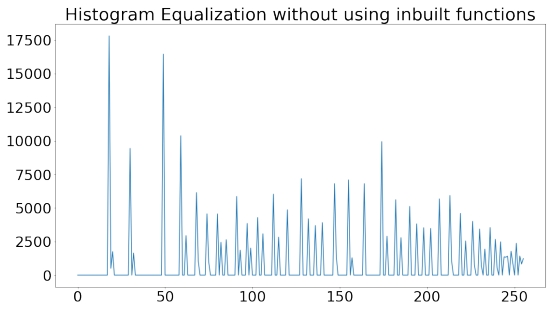
```
#Histogram Equalization without using inbuilt functions
a = np.zeros((256,),dtype=np.float32)
height,width=img.shape
for i in range(width):
    for j in range(height):
        g = img[j,i]
        a[g] = a[g]+1
MN = 1.0/(height*width)
b = np.zeros((256,))
for i in range(256):
    for j in range(i+1):
        b[i] += a[j]
    b[i] = round(b[i] * 255 * MN)
b=b.astype(np.uint8)
Equalizer = np.zeros((256,))
for i in range(width):
    for j in range(height):
        g = img[j,i]
        img[j,i]= b[g]
for i in range(width):
    for j in range(height):
        g = img[j,i]
        Equalizer[g] = Equalizer[g] + 1
#Histogram Equalization using inbuilt functions
hist_f = cv.calcHist([img2], [0], None, [256], [0,256])
g = cv.equalizeHist(img)
hist_g = cv.calcHist([g], [0], None, [256], [0,256])
```

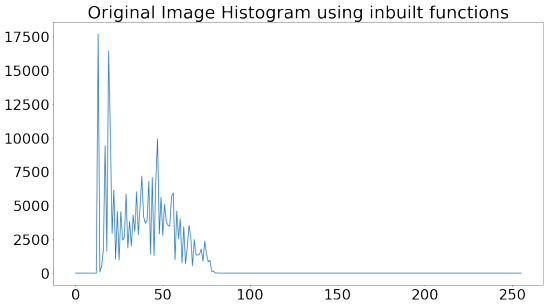

Original Image

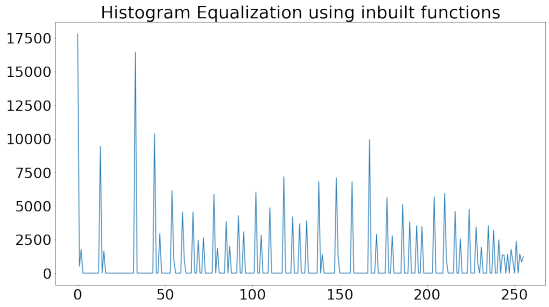

Histogram Equalized Image without using inbuilt functions


Histogram Equalized Image using inbuilt functions


Original Image Histogram without using inbuilt functions
17500
15000
12500
10000
7500
5000
2500
0
0
50
100
150
200
250


Histogram Equalization without using inbuilt functions
17500
15000
12500
10000
7500
5000
2500
0
0
50
100
150
200
250


Original Image Histogram using inbuilt functions
17500
15000
12500
10000
7500
5000
2500
0
0
50
100
150
200
250


Histogram Equalization using inbuilt functions
17500
15000
12500
10000
7500
5000
2500
0
0
50
100
150
200
250

(5) Normalized SSD comparison for Nearest Neighbor (NN) Method and Bilinear Interpolation (BI) Method as Follows. Low normalized SSD value means output zoomed image is much closer to given original image.
Using NN Normalized SSD Zoomed Image 01 and Original Image 01: 40.11174270190329
Using NN Normalized SSD Zoomed Image 02 and Original Image 02 : 16.792970920138888
Using BI Normalized SSD Zoomed Image 01 and Original Image 01 : 39.257033179012346
Using BI Normalized SSD Zoomed Image 02 and Original Image 02 : 16.21177662037037

```
def Resize(original_img, new_h, new_w, scale, Method):
 if Method == 'NN': #'NN'(Nearest Neighbor)
   zoomed_image = np.zeros((new_h,new_w,3))
   for i in range(new_h):
    for j in range(new_w):
     zoomed_image[i][j] = original_img[min(original_img.shape[0] - 1,\
     round(i/scale))][min(original_img.shape[1] - 1, round(j/scale))]
   zoomed_image = zoomed_image.astype('uint8')
   return zoomed_image
 else: #'BI'(Bilinear Interpolation)
    old_h, old_w, c = original_img.shape
    resized = np.zeros((new_h, new_w, c))
    w_scale_factor = (old_w ) / (new_w )
    h_scale_factor = (old_h ) / (new_h )
    for i in range(new_h):
            for j in range(new_w):
                    x = i * h_scale_factor
                    y = j * w_scale_factor
                    x_floor = math.floor(x)
                    x_ceil = min( old_h - 1, math.ceil(x))
                    y_floor = math.floor(y)
                    y_ceil = min(old_w - 1, math.ceil(y))
                    if (x_ceil == x_floor) and (y_ceil == y_floor):
                            q = original_img[int(x), int(y), :]
                    elif (x_ceil == x_floor):
                            q1 = original_img[int(x), int(y_floor), :]
                            q2 = original_img[int(x), int(y_ceil), :]
                            q = q1 * (y_ceil - y) + q2 * (y - y_floor)
                    elif (y_ceil == y_floor):
                            q1 = original_img[int(x_floor), int(y), :]
                            q2 = original_img[int(x_ceil), int(y), :]
                            q = (q1 * (x_ceil - x)) + (q2* (x - x_floor))
                    else:
                            v1 = original_img[x_floor, y_floor, :]
                            v2 = original_img[x_ceil, y_floor, :]
                            v3 = original_img[x_floor, y_ceil, :]
                            v4 = original_img[x_ceil, y_ceil, :]
                            q1 = v1 * (x_ceil - x) + v2 * (x - x_floor)
                            q2 = v3 * (x_ceil - x) + v4 * (x - x_floor)
                            q = q1 * (y_ceil - y) + q2 * (y - y_floor)
```

```
                resized[i,j,:] = q
    return resized.astype(np.uint8)
```

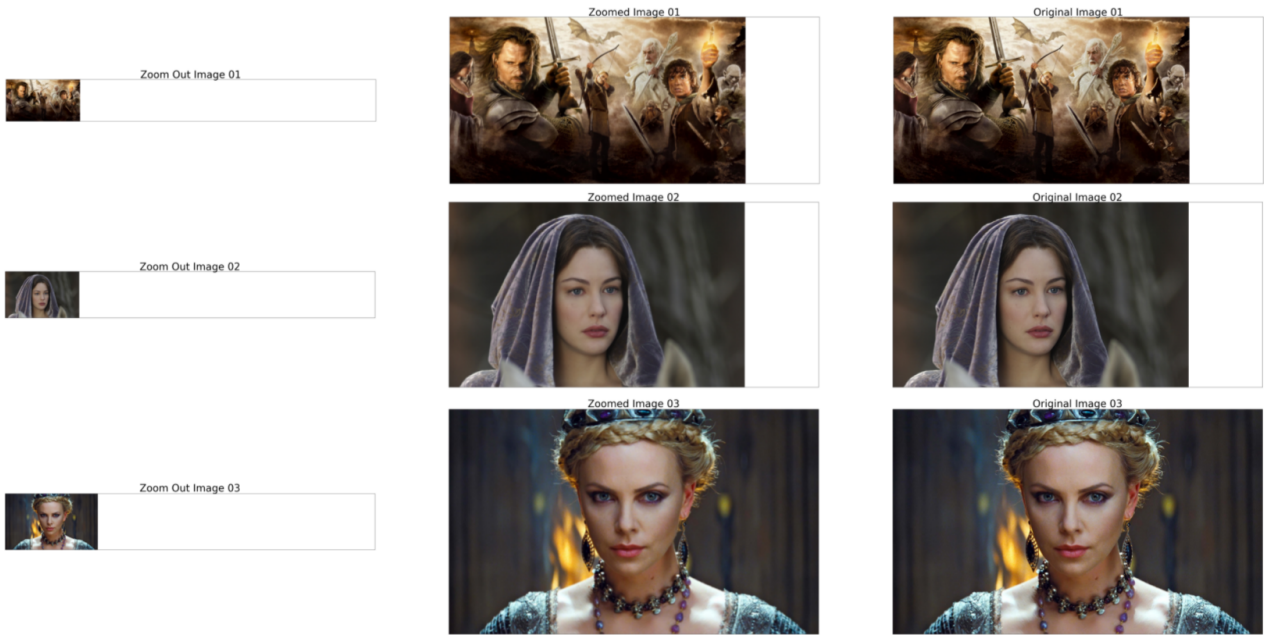



Figure 1: Using Nearest Neighbor (NN) Method

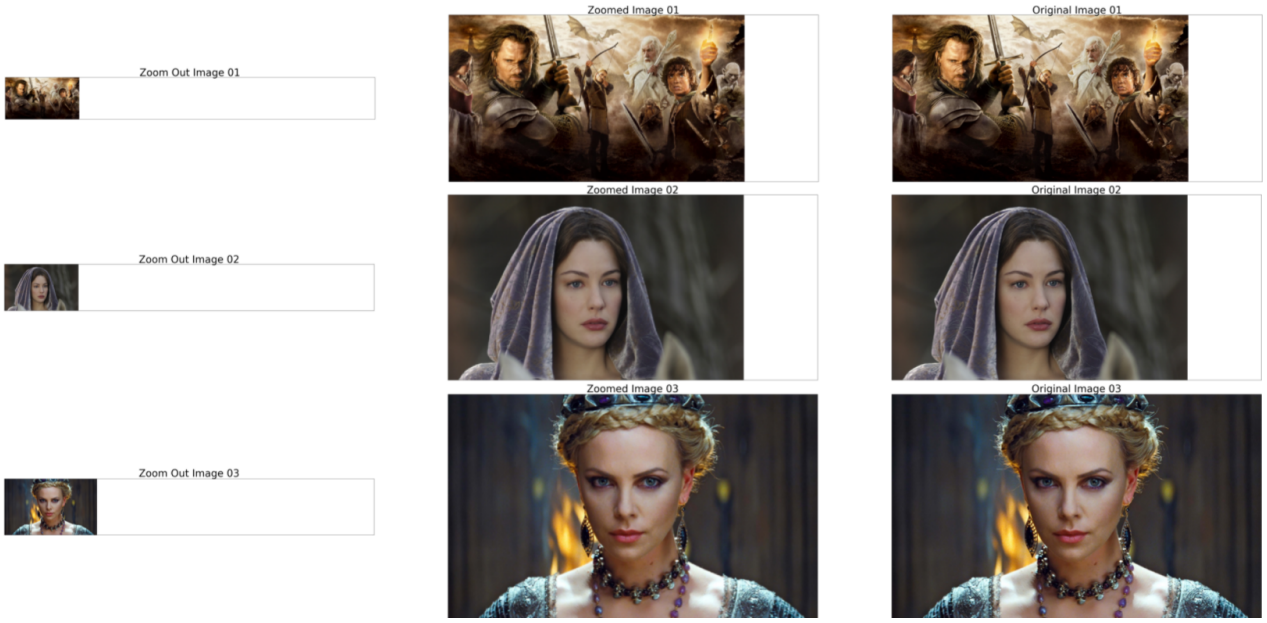
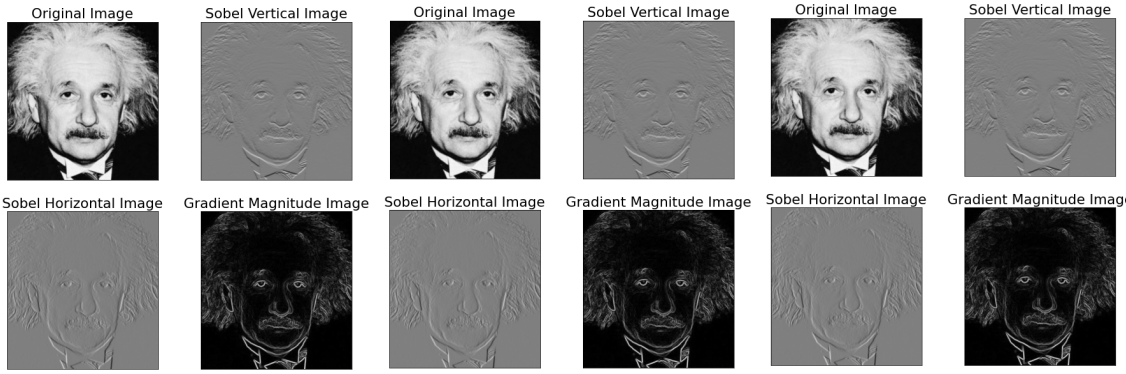



Figure 2: Using Bilinear Interpolation (BI) Method

(6) In part c using associative property, First convolve with column vector then output image again convolve with row vector.

```python
#part a
sobel_ver_kernal = np.array([(-1, -2, -1), (0, 0, 0), (1, 2, 1)], dtype='float32')
Img5_x = cv.filter2D(Img5, -1, sobel_ver_kernal)
sobel_hor_kernal = np.array([(-1, 0, 1), (-2, 0, 2), (-1, 0, 1)], dtype='float32')
Img5_y = cv.filter2D(Img5, -1, sobel_hor_kernal)
Grad_Img5 = np.sqrt(Img5_x**2 + Img5_y**2)
#part b
def convolve2d(image, kernel, bias):
    m, n = kernel.shape
    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) + bias
    return new_image
sobel_ver_kernal = np.array([(-1, -2, -1), (0, 0, 0), (1, 2, 1)], dtype='float32')
Img5_x = convolve2d(Img5, sobel_ver_kernal, 0)
sobel_hor_kernal = np.array([(-1, 0, 1), (-2, 0, 2), (-1, 0, 1)], dtype='float32')
Img5_y = convolve2d(Img5, sobel_hor_kernal,0)
Grad_Img5 = np.sqrt(Img5_x**2 + Img5_y**2)
#part c
sobel_ver_kernal_1 = np.array([[-1], [0], [1]], dtype='float32')
sobel_ver_kernal_2 = np.array([[1, 2, 1]], dtype='float32')
Img5_x = cv.filter2D(Img5, -1, sobel_ver_kernal_1)
Img5_x = cv.filter2D(Img5_x, -1, sobel_ver_kernal_2)
sobel_hor_kernal_1 = np.array([[-1], [-2], [-1]], dtype='float32')
sobel_hor_kernal_2 = np.array([[1, 0, -1]], dtype='float32')
Img5_y = cv.filter2D(Img5, -1, sobel_hor_kernal_1)
Img5_y = cv.filter2D(Img5_y, -1, sobel_hor_kernal_2)
Grad_Img5 = np.sqrt(Img5_x**2 + Img5_y**2)
```

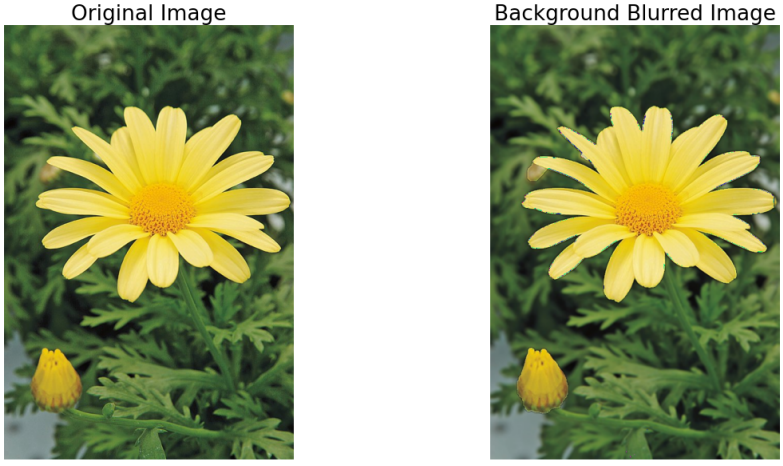



QUESTION 6 (A) QUESTION 6 (B) QUESTION 6 (C)

(7)(a) Using opencv grabCut method, We can obtain background and foreground images as follows. (b) We add gaussian blur to background image and add it to foreground image to obtain enchanced image. (c) When we apply gaussian blur to the background it also affects the foreground part. when we combine background and foreground the edges of the foreground will appear as black.

```
#part a
Mask = np.zeros(Img.shape[:2],np.uint8)
BackGModel = np.zeros((1,65),np.float64)
ForeGModel = np.zeros((1,65),np.float64)
rectangle = (50,50,500,800)
cv.grabCut(Img,Mask,rectangle,BackGModel,ForeGModel,5,cv.GC_INIT_WITH_RECT)
Mask1 = np.where((Mask==2)|(Mask==0),0,1).astype('uint8')
ImgFG = Img*Mask1[:,:,np.newaxis]
ImgBG=np.subtract(Img,ImgFG)
#part b
sigma = 2
Gauss_Kernal_1D = cv.getGaussianKernel(5, sigma)
Blurred = cv.sepFilter2D(ImgBG, -1, Gauss_Kernal_1D, Gauss_Kernal_1D)
Back_Blured_Image = np.add(ImgFG, Blurred)
```


*QUESTION 7(A)*




*QUESTION 7(B)*

**********