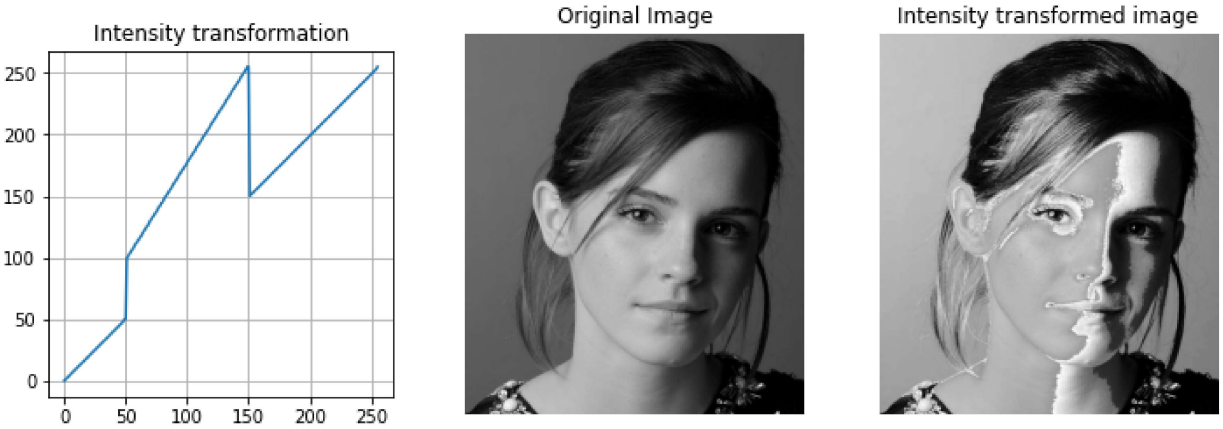
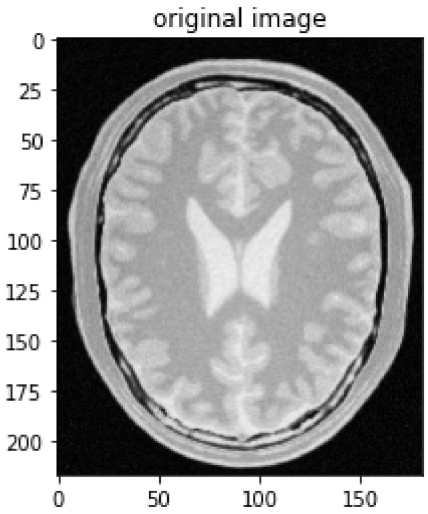
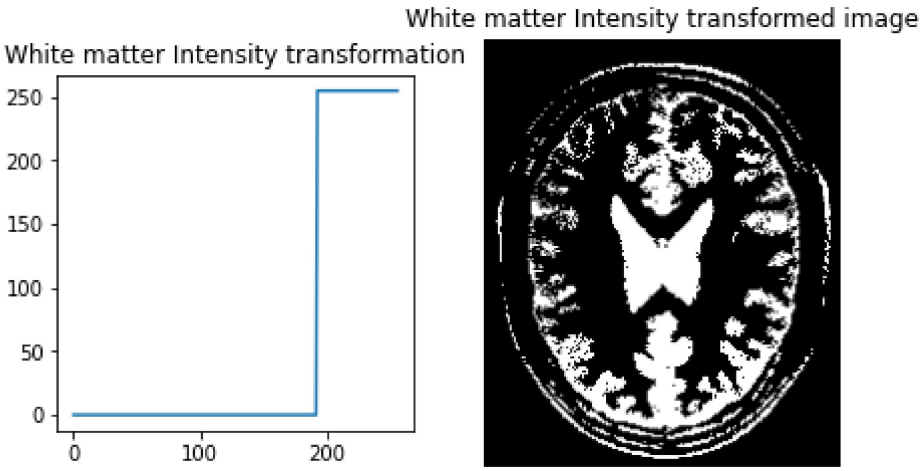
In [ ]:
```
# Question 1

import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
img= cv.imread(r'Assignment Images/emma_gray.jpg',cv.IMREAD_UNCHANGED)
assert img is not None
t1=np.linspace(0,50,50)
t2=np.linspace(50,100,1)
t3=np.linspace(100,255,99)
t4=np.linspace(255,150,1)
t5=np.linspace(150,255,105)
t=np.concatenate((t1,t2,t3,t4,t5),axis=0).astype(np.uint8)
assert len(t)==256
g=cv.LUT(img,t)

fig,ax=plt.subplots(1,3,figsize=(12,4))
ax[0].plot(t)
ax[0].set_title("Intensity transformation")
ax[0].set_aspect('equal')
ax[0].grid(1)
ax[1].imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))
ax[1].set_title("Original Image")
ax[2].imshow(cv.cvtColor(g, cv.COLOR_BGR2RGB))
ax[2].set_title("Intensity transformed image")
ax[1].axis('off')
ax[2].axis('off')
plt.show()
```

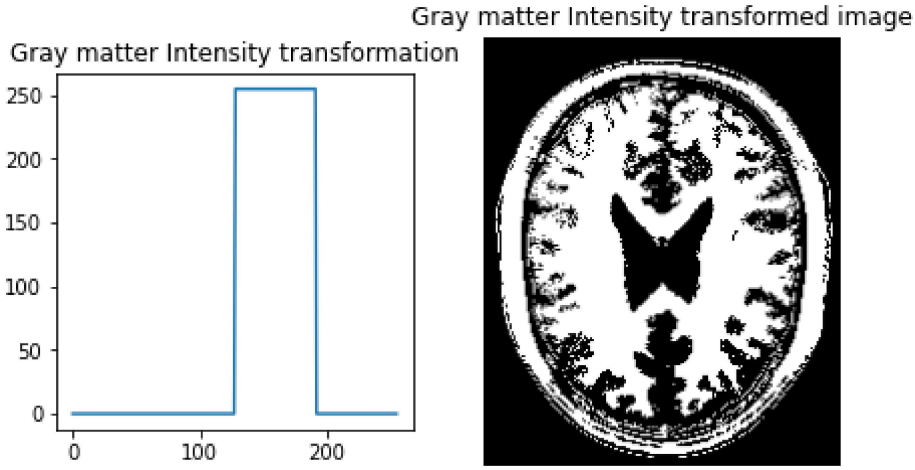


In [ ]:
```
# Question 2

img= cv.imread(r'Assignment Images/brain_proton_density_slice.png',cv.IMREAD_UNCHANGED
assert img is not None
plt.figure()
plt.imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))
plt.title("original image")
plt.show()
# White Matter
t1=np.linspace(0,0,192)
t2=np.linspace(0,255,0)
t3=np.linspace(255,255,64)
t=np.concatenate((t1,t2,t3),axis=0).astype(np.uint8)
```

```
fig,ax=plt.subplots(1,2,figsize=(7,4))
ax[0].plot(t)
ax[0].title.set_text("White matter Intensity transformation")
ax[0].set_aspect('equal')
assert len(t)==256
g=cv.LUT(img,t)
ax[1].imshow(cv.cvtColor(g, cv.COLOR_BGR2RGB))
ax[1].set_title("White matter Intensity transformed image")
ax[1].axis('off')
plt.show()

#Gray Matter
t1= np.linspace(0,0,128)
t2=np.linspace(0,255,0)
t3=np.linspace(255,255,64)
t4=np.linspace(255,0,0)
t5=np.linspace(0,0,64)
t=np.concatenate((t1,t2,t3,t4,t5),axis=0).astype(np.uint8)
assert len(t)==256
g=cv.LUT(img,t)
fig,ax=plt.subplots(1,2,figsize=(7,6))
ax[0].plot(t)
ax[0].title.set_text("Gray matter Intensity transformation")
ax[0].set_aspect('equal')
ax[1].imshow(cv.cvtColor(g, cv.COLOR_BGR2RGB))
ax[1].set_title("Gray matter Intensity transformed image")
ax[1].axis('off')
plt.show()
```



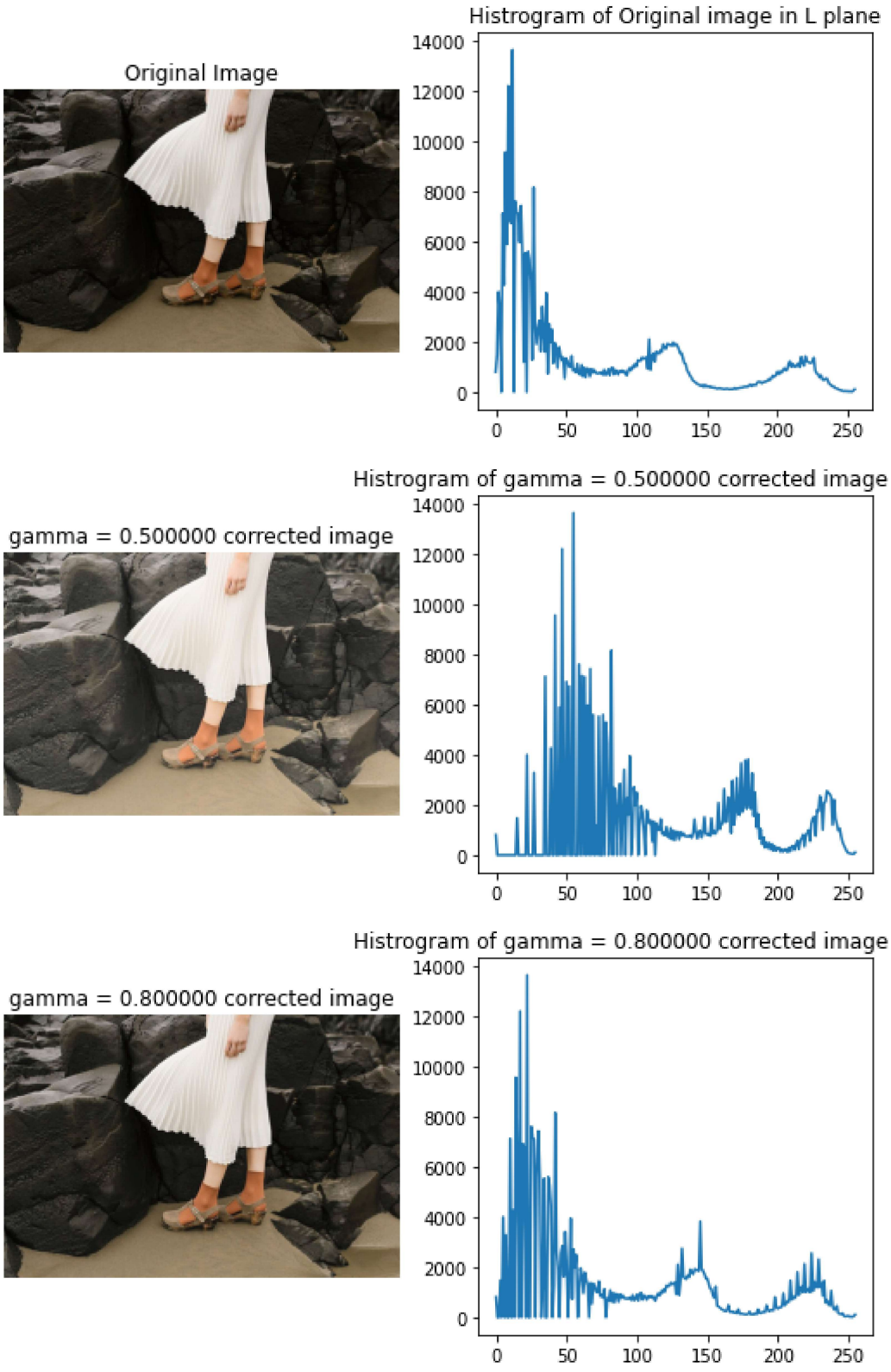


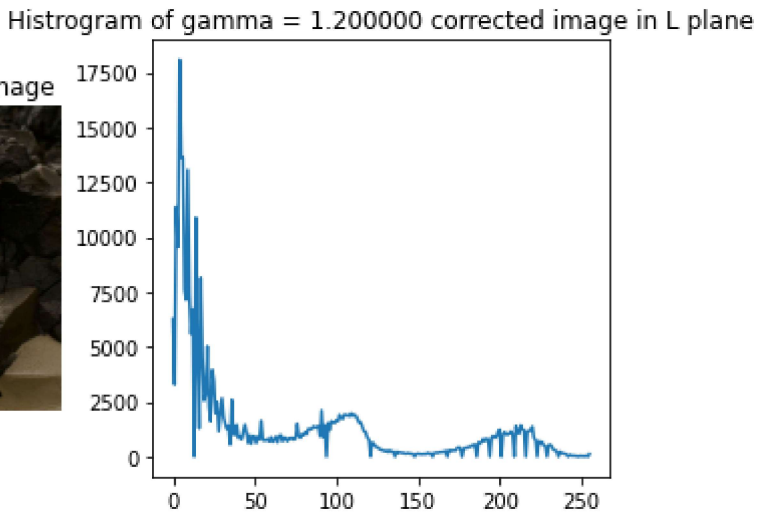


In [ ]:

```
# Question 3

img= cv.imread(r'Assignment Images/highlights_and_shadows.jpg',cv.IMREAD_UNCHANGED)
assert img is not None

lab = cv.cvtColor(img, cv.COLOR_BGR2LAB)
l,a,b = cv.split(lab)
hist_img=cv.calcHist([lab],[0],None,[256],[0,256])
gamma = [0.5,0.8,1.2]
fig,ax=plt.subplots(1,2,figsize=(9,4))
ax[0].imshow(cv.cvtColor(img,cv.COLOR_BGR2RGB))
ax[0].set_title("Original Image")
ax[1].plot(hist_img)
ax[1].set_title("Histrogram of Original image in L plane")
ax[0].axis('off')
plt.show()

for i in range(len(gamma)):
    t= np.array([(p/255)**gamma[i]*255 for p in range(0,256)]).astype(np.uint8)
    g_l=cv.LUT(l,t)
    g_lab=cv.merge([g_l,a,b])
    new_img=cv.cvtColor(g_lab,cv.COLOR_LAB2BGR)
    hist_new=cv.calcHist([g_lab],[0],None,[256],[0,256])
    # Plotting
    fig,ax=plt.subplots(1,2,figsize=(9,4))
    ax[0].imshow(cv.cvtColor(new_img,cv.COLOR_BGR2RGB))
    ax[0].set_title(f"gamma = %f corrected image" % gamma[i])
    ax[1].plot(hist_new)
```

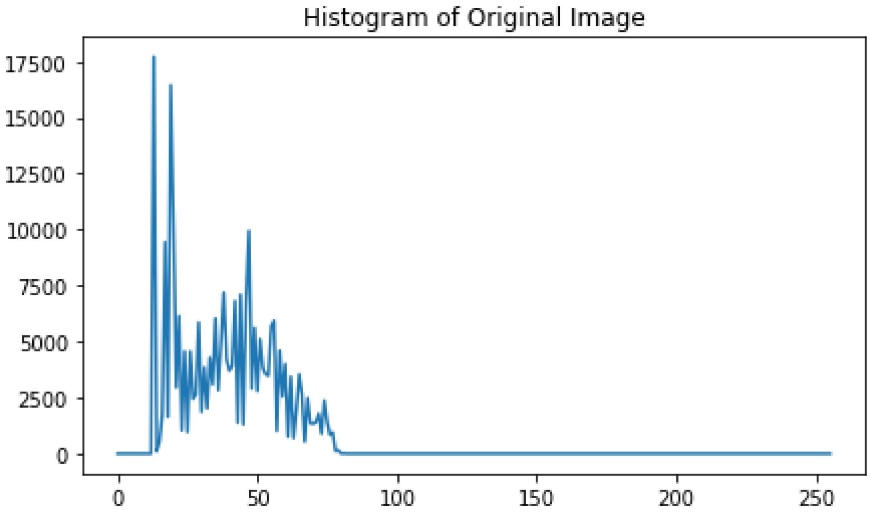
```
    ax[1].set_title(f"Histrogram of gamma = %f corrected image in L plane" % gamma[i])
    ax[0].axis("off")
    plt.show()
```

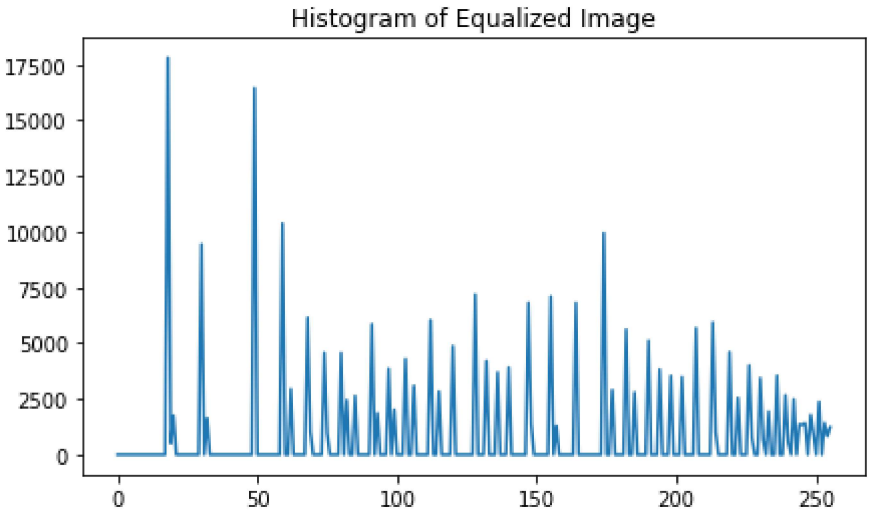


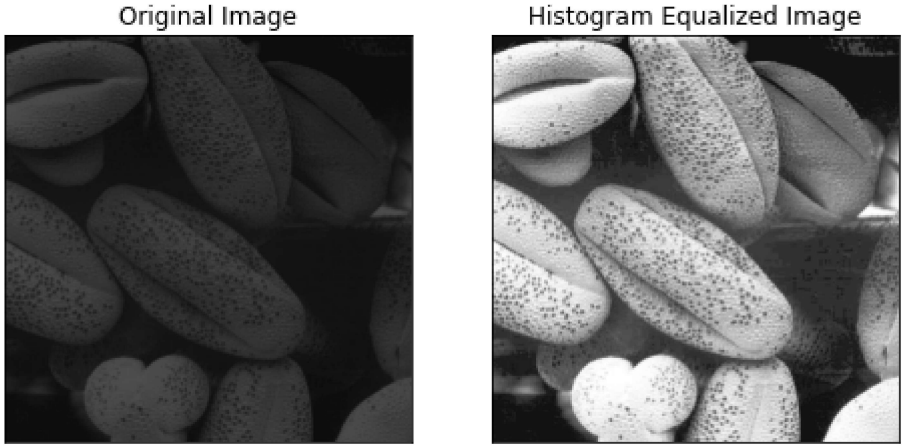


In [ ]:
```python
# Question 4

img = cv.imread(r"Assignment Images/shells.png",cv.IMREAD_GRAYSCALE)
img_flat=img.flatten()

def histogram_equalize(img,size):
    img_pixel_ =np.zeros((256,1)).astype(int)
    index_=np.linspace(0,255,256).reshape(256,1).astype(int)
    img_pixel=np.append(index_,img_pixel_,axis=1)
    for j in range(size):
        img_pixel[img[j],1]+=1 #increse the intensity count

    #Cumulative sum
    sum=0
    for i in range(256):
        sum+=img_pixel[i,1]
        img_pixel[i,1]=sum
        img_pixel[i,1]= round(img_pixel[i,1]*255/size)
    hist_img=np.zeros((1,size)).astype(np.uint8)
    for i in range(size):
        hist_img[0,i] = img_pixel[img[i],1]
    new_img=hist_img[0,:]
    return new_img

def histogram(img,size,title):
    img_pixel =np.zeros((256,1))
    index_=np.linspace(0,255,256).reshape(256,1).astype(int)
    img_pixel=np.append(index_,img_pixel,axis=1).astype(int)
    for j in range(size):
        img_pixel[int(img[j]),1]+=1
    plt.figure(figsize=(7,4))
    plt.plot(img_pixel[:,0],img_pixel[:,1])
    plt.title(title)
    plt.show()
size=np.shape(img_flat)[0]
histogram(img_flat,size,"Histogram of Original Image") #plot histogram of original ima
hist_image=histogram_equalize(img_flat,size)
histogram(hist_image,size,"Histogram of Equalized Image") #plot histogram of equalized

#plotting
fig,ax=plt.subplots(1,2,sharex='all',sharey='all',figsize=(8,5))
ax[0].imshow(cv.cvtColor(img,cv.COLOR_BGR2RGB))
```

```
ax[0].set_title("Original Image")
ax[0].set_xticks([]) , ax[0].set_yticks([])
ax[1].imshow(cv.cvtColor(hist_image.reshape(500,500),cv.COLOR_BGR2RGB))
ax[1].set_title("Histogram Equalized Image")
ax[1].set_xticks([]) , ax[1].set_yticks([])
plt.show()
for i in range(2):
    ax[i].axis("off")
```







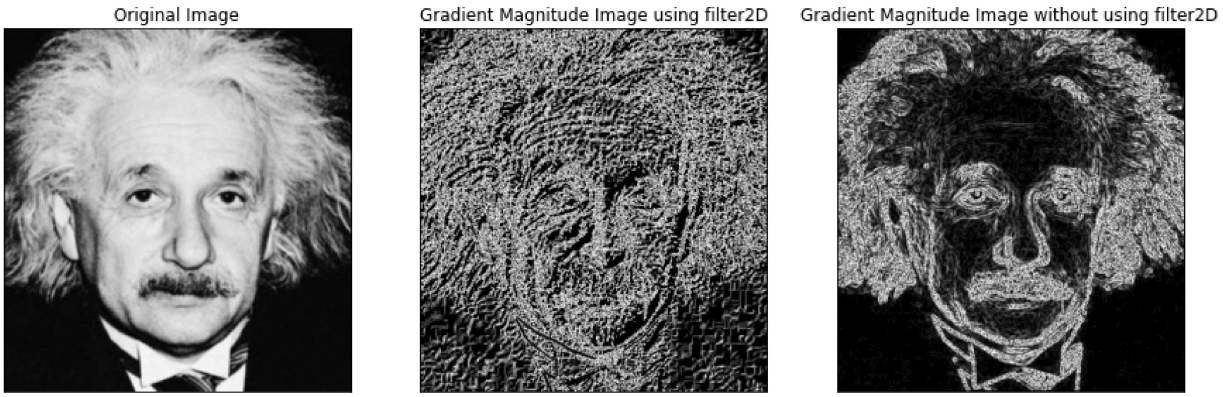
In [ ]:
```
# Question 5
```

```
img_1_small=cv.imread(r'Assignment Images/im01small.png',cv.IMREAD_GRAYSCALE)
img_2_small=cv.imread(r'Assignment Images/im02small.png',cv.IMREAD_GRAYSCALE)
img_3_small=cv.imread(r'Assignment Images/im03small.png',cv.IMREAD_GRAYSCALE)

img_1=cv.imread(r'Assignment Images/im01.png',cv.IMREAD_GRAYSCALE)
img_2=cv.imread(r'Assignment Images/im02.png',cv.IMREAD_GRAYSCALE)
img_3=cv.imread(r'Assignment Images/im03.png',cv.IMREAD_GRAYSCALE)

#Zoom image by Nearest Neighbour
def zoomimage_NN(img,zoomfactor):
    zoom_img = cv.resize(img,None, fx = zoomfactor, fy = zoomfactor, interpolation = c
    return zoom_img

#Zoom image by Bilinear Interpolation
def zoomimage_BI(img,zoomfactor):
    zoom_img = cv.resize(img,None, fx = zoomfactor, fy = zoomfactor, interpolation = c
    return zoom_img

img_small=[img_1_small,img_2_small,img_3_small]
img=[img_1,img_2,img_3]
for i in range(2):
    zoom_img_NN=zoomimage_NN(img_small[i],4)
    zoom_img_BI= zoomimage_BI(img_small[i],4)

    #Calculating SSD
    difference_NN=np.subtract(zoom_img_NN,img[i])
    squ_NN=np.square(difference_NN)
    SSD_NN=np.sum(squ_NN)

    difference_BI=np.subtract(zoom_img_BI,img[i])
    squ_BI=np.square(difference_BI)
    SSD_BI=np.sum(squ_BI)
    print('SSD Of Nearest Neighbour = ',SSD_NN)
    print('SSD of Bilinear Interpolation = ',SSD_BI)
    #Plotting
    fig,ax=plt.subplots(1,3,sharex='all',sharey='all',figsize=(18,6))
    ax[0].imshow(zoom_img_NN,cmap="gray")
    ax[0].set_title('zoomed image by Nearest Neighbour')
    ax[1].imshow(zoom_img_BI,cmap="gray")
    ax[1].set_title('zoomed image by Bilinear Interpolation')
    ax[2].imshow(img[i],cmap="gray")
    ax[2].set_title('Original large image')
    for i in range(3):
       ax[i].axis("off")
    plt.show()
```

```
SSD Of Nearest Neighbour =  64809605
SSD of Bilinear Interpolation =  64245250
```


```
SSD Of Nearest Neighbour =  27496006
SSD of Bilinear Interpolation =  24595568
```

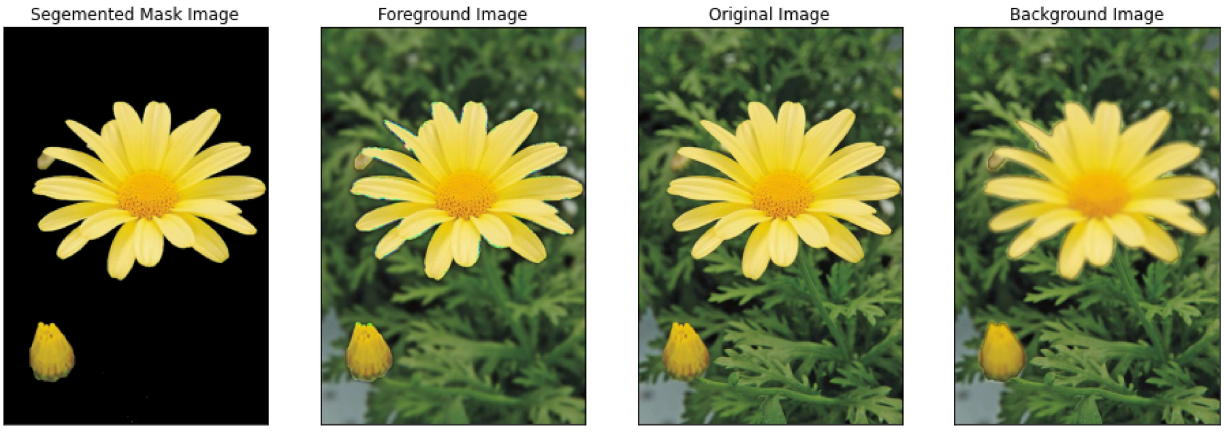
In [ ]:
```
# Question 6(a)
import math

img=cv.imread(r"Assignment Images/einstein.png",cv.IMREAD_GRAYSCALE)
assert img is not None
kernel_v,kernel_h=np.array([[-1,-2,-1],[0,0,0],[1,2,1]]),np.array([[-1,0,1],[-2,0,2],[
img_x,img_y=cv.filter2D(img,-1,kernel_v),cv.filter2D(img,-1,kernel_h)
grad_mag_filter2D=np.sqrt(img_x**2+img_y**2)

# Filter Function
def filter(image, kernel):# Filter Function
    assert kernel.shape[0]%2 == 1 and kernel.shape[1]%2==1
    k_hh,k_hw = math.floor(kernel.shape[0]/2),math.floor(kernel.shape[1]/2)
    h,w = image.shape
    image_float= cv.normalize(image.astype("float"),None,0.2,0.3,cv.NORM_MINMAX)
    result= np.zeros( image.shape,"float")
    for m in range(k_hh,h-k_hh):
        for n in range(k_hw,w-k_hw):
            result[m,n]=np.dot(image_float[m-k_hh: m+k_hh+1,n-k_hw: n+k_hw+1].flatten(
    return result
def filter_sep(image,kernel_col,kernel_row):
    k_hh,k_hw = math.floor(kernel_row.shape[1]/2),math.floor(kernel_col.shape[0]/2)
    h,w = image.shape
    image_float= cv.normalize(image.astype("float"),None,0.2,0.3,cv.NORM_MINMAX)
    result= np.zeros( image.shape,"float")
    for m in range(k_hh,h-k_hh):
        for n in range(k_hw,w-k_hw):
            result[m,n]=np.dot(np.dot(image_float[m-k_hh: m+k_hh+1,n-k_hw: n+k_hw+1],k
    return result
imgb_v,imgb_h=filter(img,kernel_v),filter(img,kernel_h)
imgb_v=(imgb_v*255.0).astype(np.uint8)
imgb_h=(imgb_h*255.0).astype(np.uint8)
grad_mag=np.sqrt(imgb_v**2+imgb_h**2)

fig,ax=plt.subplots(1,3,sharex='all',sharey='all',figsize=(15,10))
ax[0].imshow(img,cmap='gray',vmin=0,vmax=255)
ax[0].set_title('Original Image')
ax[0].set_xticks([]) , ax[0].set_yticks([])
ax[1].imshow(grad_mag_filter2D,cmap='gray')
ax[1].set_title('Gradient Magnitude Image using filter2D')
ax[2].imshow(grad_mag,cmap='gray')
ax[2].set_title("Gradient Magnitude Image without using filter2D")
plt.show()
for i in range(2):
    ax[i].axis('off')
```



In [ ]:

```
# Question 7

img = cv.imread("Assignment Images/daisy.jpg",cv.IMREAD_COLOR) # Read the original ima
assert img is not None

# Define boundary rectangle containing the foreground object
height, width,_ = img.shape
left_margin_proportion = 0.1
right_margin_proportion = 0.1
up_margin_proportion = 0.1
down_margin_proportion = 0.1

boundary_rectangle = (int(width * left_margin_proportion),int(height * up_margin_propo
    int(width * (1 - right_margin_proportion)),int(height * (1 - down_margin_proportio
# Set the seed for reproducibility purposes
cv.setRNGSeed(0)
# Initialize GrabCut mask image, that will store the segmentation result
mask = np.zeros((height, width), np.uint8)
# Arrays used by the algorithm internally
background_model = np.zeros((1, 65), np.float64)
foreground_model = np.zeros((1, 65), np.float64)
number_of_iterations = 5

grab_img = cv.grabCut(img=img,mask=mask,rect=boundary_rectangle,bgdModel=background_mo
    iterCount=number_of_iterations,mode=cv.GC_INIT_WITH_RECT,)
grabcut_mask = np.where((mask == cv.GC_PR_BGD) | (mask == cv.GC_BGD), 0, 1).astype("ui
segmented_mask_image = img.copy() * grabcut_mask[:, :, np.newaxis]
img_2=img-segmented_mask_image
blur_1=cv.GaussianBlur(segmented_mask_image,(19,19),cv.BORDER_CONSTANT)
background_img=img_2+blur_1
blur_2=cv.GaussianBlur(img_2,(19,19),cv.BORDER_CONSTANT)
foreground_img=blur_2+segmented_mask_image

fig,ax=plt.subplots(1,4,sharex='all',sharey='all',figsize=(16,16))
ax[0].imshow(cv.cvtColor(segmented_mask_image,cv.COLOR_BGR2RGB))
ax[0].set_title('Segemented Mask Image')
ax[0].set_xticks([]) , ax[0].set_yticks([])
ax[1].imshow(cv.cvtColor(foreground_img,cv.COLOR_BGR2RGB))
ax[1].set_title("Foreground Image")
ax[1].set_xticks([]) , ax[1].set_yticks([])
ax[2].imshow(cv.cvtColor(img,cv.COLOR_BGR2RGB))
ax[2].set_title('Original Image')
ax[2].set_xticks([]) , ax[2].set_yticks([])
ax[3].imshow(cv.cvtColor(background_img,cv.COLOR_BGR2RGB))
ax[3].set_title("Background Image")
```

```
ax[3].set_xticks([]) , ax[3].set_yticks([])
plt.show()
```



In [ ]:
```
blur_img = cv.GaussianBlur(segmented_mask_image, (0, 0), 100)
sharpen = cv.addWeighted(segmented_mask_image, 1.5, blur_img, -0.5, 0)
enhanced_img=sharpen+img_2
fig,ax=plt.subplots(1,2,sharex='all',sharey='all',figsize=(7,7))
ax[0].imshow(cv.cvtColor(img,cv.COLOR_BGR2RGB))
ax[0].set_title('Original Image')
ax[0].set_xticks([]) , ax[0].set_yticks([])
ax[1].imshow(cv.cvtColor(enhanced_img,cv.COLOR_BGR2RGB))
ax[1].set_title("Enhanced Image")
ax[1].set_xticks([]) , ax[1].set_yticks([])
```

Out[ ]:
```
([], [])
```