EN2550 Assignment 1 on Intensity Transformations and Neighborhood Filtering

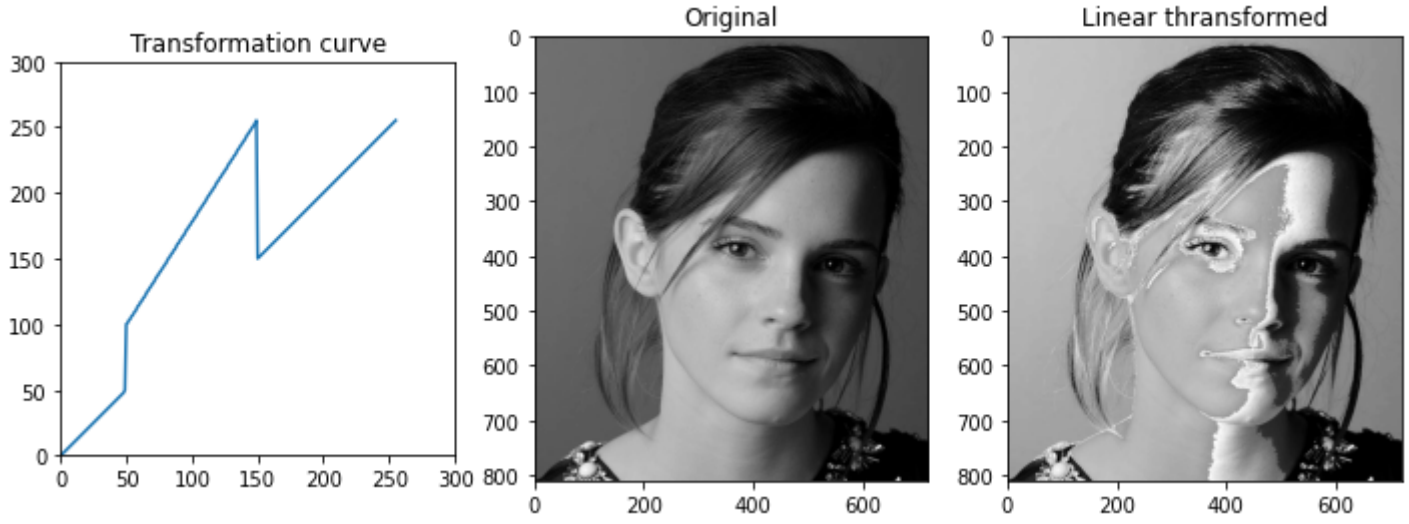
Name : Vakeesan.K

Index N.O: 190643G

Github Link: https://github.com/vakeesanvk/image_processing_assignment_01

1)

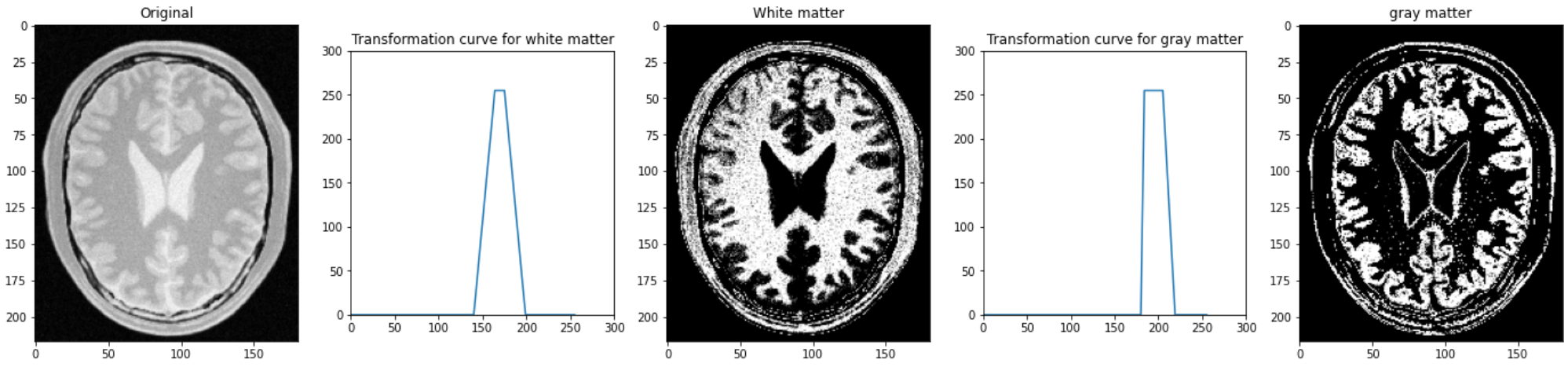
In [ ]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv

img=cv.imread(r'C:\Python39\cv\assignment_01\emma_gray.jpg',cv.IMREAD_GRAYSCALE)
assert img is not None

t1=np.linspace(0,50,50) ;t2=np.linspace(50,100,0);t3=np.linspace(100,255,100);t4=np.linspace(150,255,106)
t=np.concatenate((t1,t2,t3,t4),axis=0).astype(np.uint8)
fig,ax=plt.subplots(1,3,figsize=(12,6))

ax[0].plot(t);ax[0].set_aspect('equal');ax[0].set_xlim(0,300);ax[0].set_ylim(0,300);ax[0].set_title('Transformation curve')

assert len(t)== 256
g = cv.LUT(img,t)
ax[1].imshow(img,cmap='gray');ax[1].set_title('Original');ax[2].imshow(g,cmap='gray');ax[2].set_title('Linear thransformed')
plt.show()
```



Number of elements we have given in the x axis should be equal to 256. that's why i chose 106 istead of 105 when defining t4 .

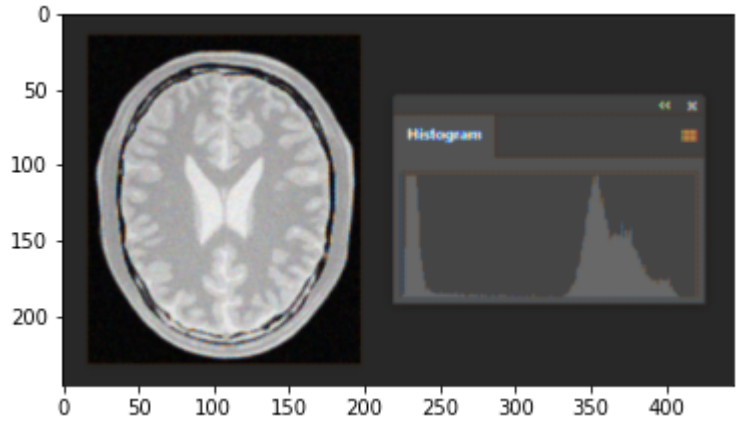
2)

In [ ]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv

img=cv.imread(r'C:\Python39\cv\assignment_01\brain_proton_density_slice.png',cv.IMREAD_GRAYSCALE)
assert img is not None

t1=np.linspace(0,0,140); t2=np.linspace(0,255,25); t3=np.linspace(255,255,10); t4=np.linspace(255,0,25); t5=np.linspace(0,0,56)

t=np.concatenate((t1,t2,t3,t4,t5),axis=0).astype(np.uint8)
assert len(t)== 256
g = cv.LUT(img,t)

t1_=np.linspace(0,0,180); t2_=np.linspace(0,255,5); t3_=np.linspace(255,255,20); t4_=np.linspace(255,0,15); t5_=np.linspace(0,0,36)

t_=np.concatenate((t1_,t2_,t3_,t4_,t5_),axis=0).astype(np.uint8)

assert len(t_)== 256
h = cv.LUT(img,t_)

fig,ax=plt.subplots(1,5,figsize=(24,6))
ax[0].imshow(img,cmap='gray');ax[0].set_title('Original')

ax[1].plot(t); ax[1].set_aspect('equal'); ax[1].set_xlim(0,300); ax[1].set_ylim(0,300)
ax[1].set_title('Transformation curve for white matter')

ax[2].imshow(g,cmap='gray');ax[2].set_title('White matter')

ax[3].plot(t_); ax[3].set_aspect('equal'); ax[3].set_xlim(0,300); ax[3].set_ylim(0,300)
ax[3].set_title('Transformation curve for gray matter')

ax[4].imshow(h,cmap='gray'); ax[4].set_title('gray matter')
plt.show()
```



i got the linear transformation grpahs by setting approximate graph using Adobe Photoshop software. output images looks noisy because of the width of the curves that i used. if i change that value large enough then i get an image which is highlighting other parts too. I've included that reference below.

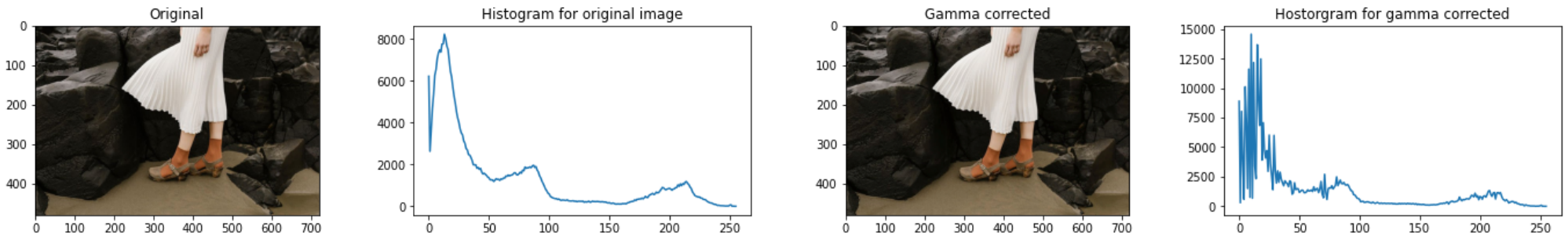
In [ ]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv

img=cv.imread(r'C:\Python39\cv\assignment_01\photoshop.png',cv.IMREAD_GRAYSCALE)
assert img is not None
plt.imshow(cv.cvtColor(img,cv.COLOR_BAYER_BG2RGB))
plt.show()
```



3)

In [ ]:
```
%matplotlib inline
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img=cv.imread(r'C:\Python39\cv\assignment_01\highlights_and_shadows.jpg')
assert img is not None

gamma = 1.5

lab_img=cv.cvtColor(img,cv.COLOR_BGR2Lab).astype("float32")
lab_img[1,:,:] = lab_img[1,:,:] * gamma
gamma_img = cv.cvtColor(lab_img.astype("uint8"),cv.COLOR_Lab2BGR)

hist_i = cv.calcHist([img],[0],None,[256],[0,256])

hist_g = cv.calcHist([gamma_img],[0],None, [256],[0,256])

print("Gamma is %f"%gamma)

fig,ax=plt.subplots(1,4,figsize=(24,3))

ax[0].imshow(cv.cvtColor(img,cv.COLOR_BGR2RGB)); ax[0].set_title('Original')
ax[1].plot(hist_i); ax[1].set_title('Histogram for original image')

ax[2].imshow(cv.cvtColor(gamma_img,cv.COLOR_BGR2RGB)); ax[2].set_title('Gamma corrected')
ax[3].plot(hist_g); ax[3].set_title('Hostorgram for gamma corrected')

plt.show()
```

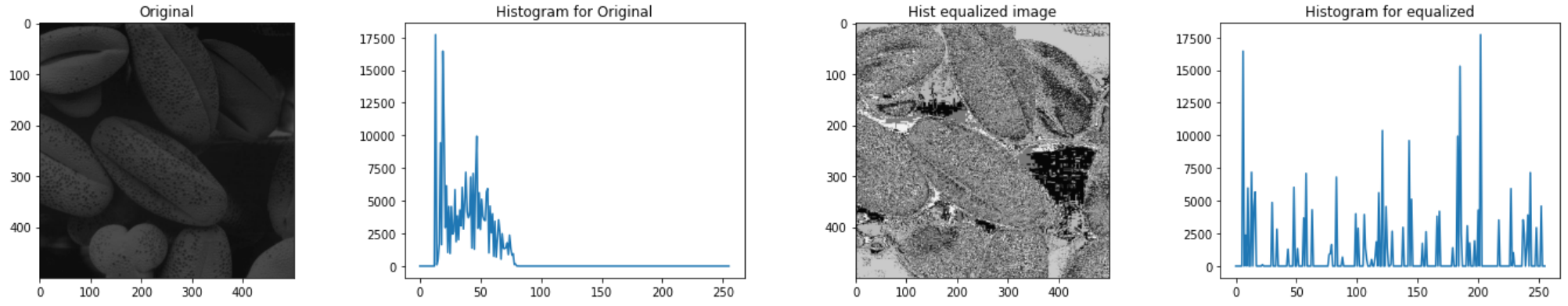
```
Gamma is 1.500000
```



4)

In [ ]:
```
%matplotlib inline
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

f=cv.imread(r'C:\Python39\cv\exercices\lec 2\shells.tif', cv.IMREAD_GRAYSCALE)
assert f is not None
```

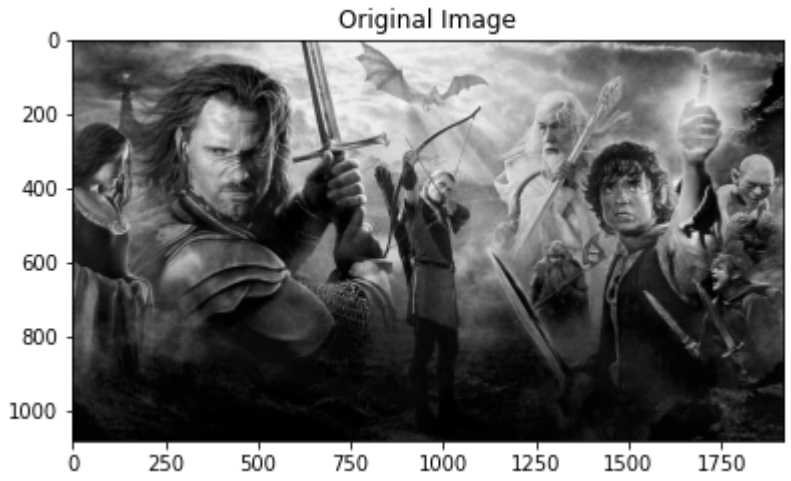
```
histogram_array = np.bincount(f.flatten(), minlength=256)

num_pixels = np.sum(histogram_array)
hist = histogram_array/num_pixels

hist_c = np.cumsum(histogram_array)

transform_map = np.floor(255 * hist_c).astype(np.uint8)

img_list = list(f.flatten())

eq_img_list = [transform_map[i] for i in img_list]

hist_eq = np.reshape(np.asarray(eq_img_list), f.shape)

hist_f = cv.calcHist([f],[0],None,[256],[0,256])

hist_g = cv.calcHist([hist_eq],[0],None, [256],[0,256])

fig, ax = plt.subplots(1,4, figsize=(24,4))
ax[0].imshow(cv.cvtColor(f,cv.COLOR_BGR2RGB)); ax[0].set_title('Original')
ax[1].plot(hist_f); ax[1].set_title('Histogram for Original')

ax[2].imshow(cv.cvtColor(hist_eq,cv.COLOR_BGR2RGB)); ax[2].set_title('Hist equalized image')
ax[3].plot(hist_g); ax[3].set_title('Histogram for equalized')

plt.show()
```

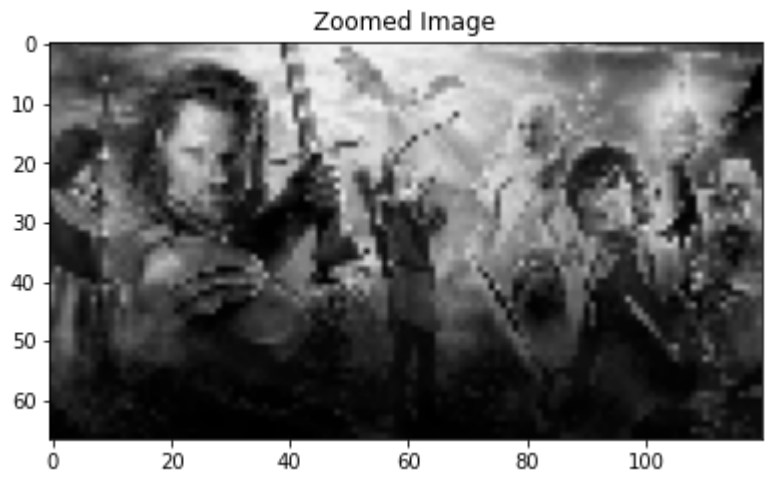


5)

In [ ]:
```
%matplotlib inline
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img_large= cv.imread(r'C:\Python39\cv\assignment_01\a1q5images\im01.png',cv.IMREAD_GRAYSCALE)
img_small=cv.imread(r'C:\Python39\cv\assignment_01\a1q5images\im01small.png',cv.IMREAD_GRAYSCALE)
assert img_large is not None
assert img_small is not None

s=4
scale=1/s
rows = int(scale*img_small.shape[0])
cols = int(scale*img_small.shape[1])
zoomed = np.zeros((rows,cols),dtype=img_small.dtype)

for i in range(0,rows):
    for j in range (0,cols):
        zoomed[i,j]= img_small[int(i/scale),int(j/scale)]
#calculating SSD
ssd = 0
for i in range(0,rows):
    for j in range (0,cols):
        diff = img_large[i][j] - zoomed[i][j]
        ssd += diff * diff
print(ssd)
fig, ax = plt.subplots(1,2,figsize=(14,6))
ax[0].imshow(cv.cvtColor(img_large,cv.COLOR_BGR2RGB)); ax[0].set_title("Original Image")
ax[1].imshow(cv.cvtColor(zoomed,cv.COLOR_BGR2RGB)); ax[1].set_title("Zoomed Image")
plt.show()
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_11768\1717817046.py:24: RuntimeWarning: overflow encountered in ubyte_scalars
  diff = img_large[i][j] - zoomed[i][j]
C:\Users\USER\AppData\Local\Temp\ipykernel_11768\1717817046.py:25: RuntimeWarning: overflow encountered in ubyte_scalars
  ssd += diff * diff
868663
```



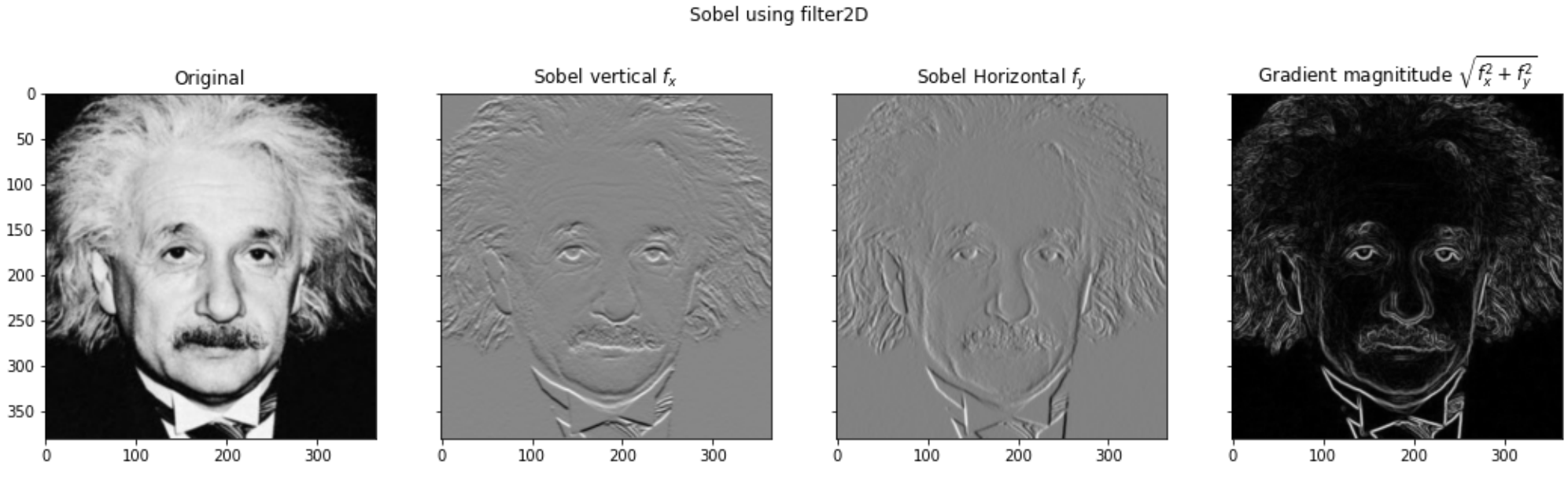


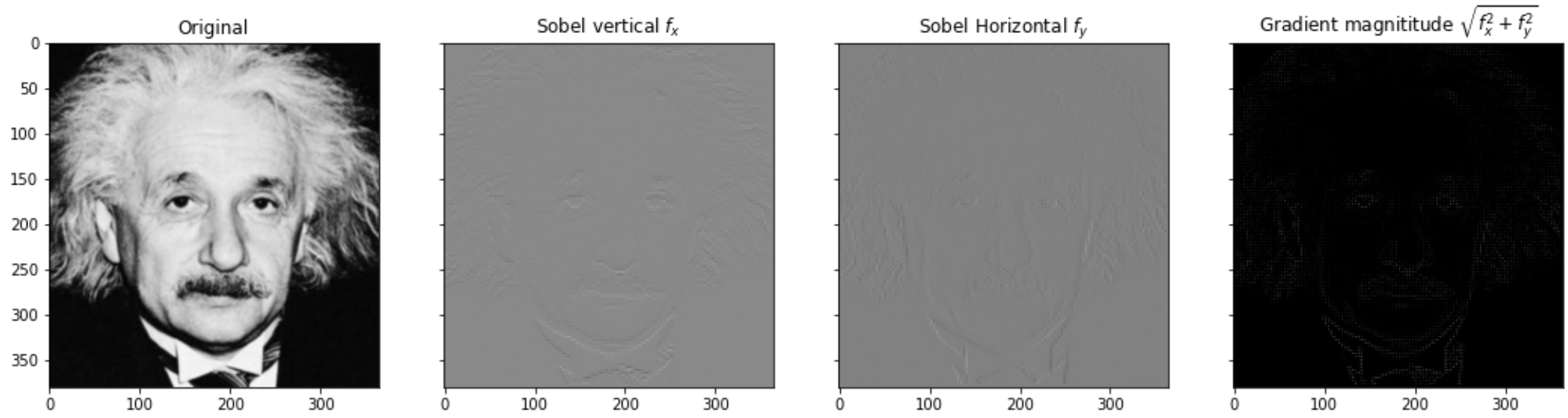
i get a SSD value which is very big. That's why zoomed image looks more mosaic. When scaling factor is increasing, output image can't be in an expected image quality.

6)

In [ ]:

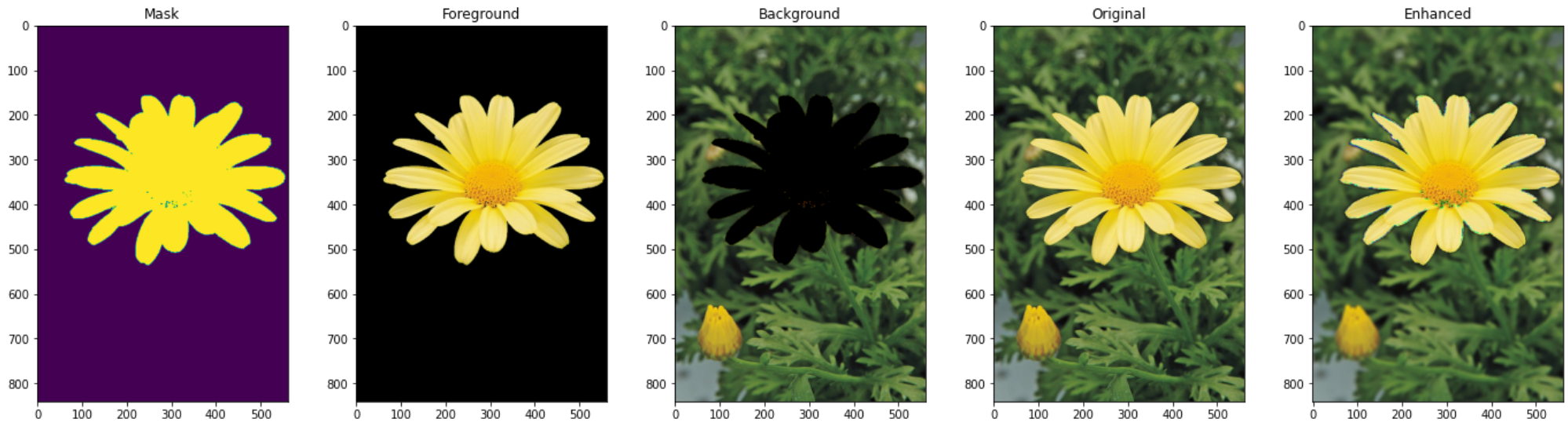
```
%matplotlib inline
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread(r'C:\Python39\cv\assignment_01\einstein.png',cv.IMREAD_GRAYSCALE).astype(np.float32)
assert img is not None

#sobel vertical
kernel_v=np.array([(-1,-2,-1),(0,0,0),(1,2,1)], dtype=np.float32)
imgv = cv.filter2D(img,-1,kernel_v)

#sobel horizontal
kernel_h=np.array([(-1,0,1),(-2,0,2),(-1,0,1)], dtype=np.float32)
imgh = cv.filter2D(img,-1,kernel_h)

grad_mag = np.sqrt(imgv**2+imgh**2)

fig1,ax = plt.subplots(1,4,sharex='all', sharey='all',figsize=(18,5))
fig1.suptitle("Sobel using filter2D")
ax[0].imshow(img,cmap='gray'); ax[0].set_title('Original')
ax[1].imshow(imgv,cmap='gray'); ax[1].set_title('Sobel vertical $f_x$')
ax[2].imshow(imgh,cmap='gray'); ax[2].set_title('Sobel Horizontal $f_y$')
ax[3].imshow(grad_mag,cmap='gray'); ax[3].set_title('Gradient magnititude $\sqrt{f_x^2 + f_y^2}$')

#sobel verical & horizontal manual method
height, width = img.shape
man_imgv=np.ones(( height,width))
man_imgh=np.ones((height,width))
for i in range(0,width-2,3):
    for j in range(0,height-2,3):
        man_imgv[j][i] = np.sum(np.multiply(img[j:j+3,i:i+3],kernel_v))
        man_imgh[j][i] = np.sum(np.multiply(img[j:j+3,i:i+3],kernel_h))
man_grad_mag = np.sqrt(man_imgv**2+man_imgh**2)

fig2,ax_ = plt.subplots(1,4,sharex='all', sharey='all',figsize=(18,5))
fig2.suptitle("Sobel using own code")
ax_[0].imshow(img,cmap='gray'); ax_[0].set_title('Original')
ax_[1].imshow(man_imgv,cmap='gray'); ax_[1].set_title('Sobel vertical $f_x$')
ax_[2].imshow(man_imgh,cmap='gray'); ax_[2].set_title('Sobel Horizontal $f_y$')
ax_[3].imshow(man_grad_mag,cmap='gray'); ax_[3].set_title('Gradient magnititude $\sqrt{f_x^2 + f_y^2}$')

plt.show()
```





Here i used simple vector multiplication as first step for convolution and then i get the sum of that into a new image vectors(e.g. man_imgv). i don't sharpen the final image at all. that's why it looks faded.

In [ ]:
```
%matplotlib inline
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread(r'C:\Python39\cv\assignment_01\daisy.jpg')

mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
rect = (0,0,800,550)
cv.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv.GC_INIT_WITH_RECT)

mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')

fg_img = img*mask2[:,:,np.newaxis]
bg_img=img-fg_img
blur_bg=cv.GaussianBlur(bg_img,(9,9),cv.BORDER_DEFAULT)
fin_img=blur_bg+fg_img

fig, ax=plt.subplots(1,5,figsize=(24,6))
ax[0].imshow(mask2); ax[0].set_title("Mask")
ax[1].imshow(cv.cvtColor(fg_img,cv.COLOR_BGR2RGB)); ax[1].set_title("Foreground")
ax[2].imshow(cv.cvtColor(bg_img,cv.COLOR_BGR2RGB)); ax[2].set_title("Background")

ax[3].imshow(cv.cvtColor(img,cv.COLOR_BGR2RGB)); ax[3].set_title("Original")
ax[4].imshow(cv.cvtColor(fin_img,cv.COLOR_BGR2RGB)); ax[4].set_title("Enhanced")

plt.show()
```



Mask2 is used to filter out all the zero values between the foreground pixels and background pixels. such that it will leave the one values at the sepeartion line between foreground and background. Obviously, it will be looking like a dark edege there.