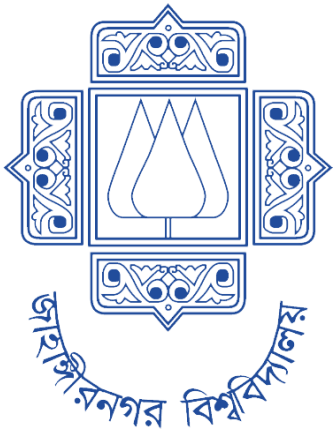
Institute of Information Technology (IIT)

Jahangirnagar University



ICT 4202– Digital Image Processing Lab

LAB REPORT

SUBMITTED BY

Name: Md. Shakil Hossain

Roll No:2023

Exam Roll:192340

SUBMITTED TO

Mehrin Anannya

Assistant Professor, IIT

Contents

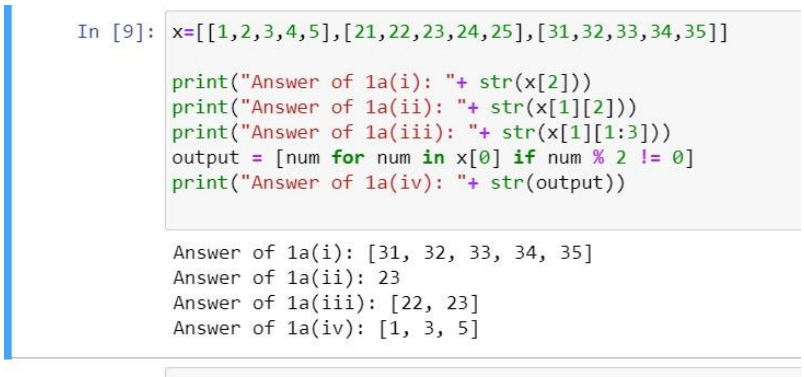
Lab Report - 01

Task 1(a):

Code:

```
print("Answer of 1a(i): "+ str(x[2]))
print("Answer of 1a(ii): "+ str(x[1][2]))
print("Answer of 1a(iii): "+ str(x[1][1:3]))
output = [num for num in x[0] if num % 2 != 0]
print("Answer of 1a(iv): "+ str(output))
```

Output:



Task 1(b):

Declare y = [0, 0, 0], now using for loop write average of first list in list 'x' on first index of listy and so on. The print(y) should give the output: [3.0, 23.0, 33.0]

Code:

```
y = [0, 0, 0]

for i in range(3):
        for j in range(5):
                y[i] += x[i][j]
print(y)
for i in range(3):
        y[i] /= 5

print(y)
```

Output:

```
In [29]: y = [0, 0, 0]

for i in range(3):
    for j in range(5):
        y[i] += x[i][j]
print(y)
for i in range(3):
    y[i] /= 5

print(y)
```

```
[15, 115, 165]
[3.0, 23.0, 33.0]
```

Task 2(a):

x = [1, 3, 5, 6, 7, 8, 6, 1, 2, 3] y = [0, 0, 0, 0, 0, 0, 0, 0]

a. Write python code using a while loop that writes the average of the first three items on the first index of y and so on. The print(y) should give the following output
Output : [3.0, 4.666666666666667, 6.0, 7.0, 7.0, 5.0, 3.0, 2.0]

Code:

```
X = [1,3,5,6,7,8,6,1,2,3]
Y = [0,0,0,0,0,0,0,0]
for i in range(8):
for j in range(3):
 y[i] += x[i+j]
 print(y)
for i in range(8):
 y[i] /= 3
print(y)
```

```
10]: x=[1,3,5,6,7,8,6,1,2,3]
y=[0,0,0,0,0,0,0,0]
for i in range(8):
    for j in range(3):
        y[i] += x[i+j]
print(y)
for i in range(8):
    y[i] /= 3

print(y)
```

```
[9, 14, 18, 21, 21, 15, 9, 6]
[3.0, 4.666666666666667, 6.0, 7.0, 7.0, 5.0, 3.0, 2.0]
```

Output:

Task 2(b):

Define a function that takes list length as argument and returns the average. Then calculate theaverage of x and y.

Code:

```
def avg1(ar,len):t=0
    for i in range(len):
    t+=ar[i]
    t/=len return t
xavg=avg1(x,len(x))
print(xavg)
yavg=avg1(y,len(y))
print(yavg)
```

Output:

In [11]:
```
def avg1(ar,len):
    t=0
    for i in range(len):
        t+=ar[i]
    t/=len
    return t
xavg=avg1(x,len(x))
print(xavg)
yavg=avg1(y,len(y))
print(yavg)
```

```
4.2
4.708333333333334
```

Lab Report - 02

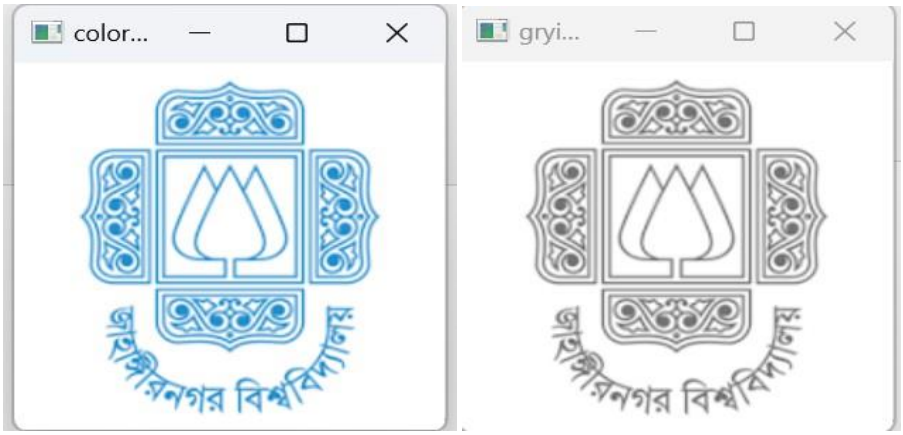
Task 1:

Save our university logo from the website and read it in gray scale and as a colour image. Showit in GUI. Then save it in a different directory.

Code:

```
import cv2
import os
import numpy as np
from matplotlib import pyplot as plt
c_img=cv2.imread('JU_logo.png',cv2.IMREAD_COLOR)
g_img=cv2.imread('JU_logo.png',0)
pr=r' F:\Github\Digital-Image-Processing-Lab'
os.chdir(pr)

print("before image")

print(os.listdir(pr))
cv2.imshow('gray image',g_img)
cv2.imshow('color image',c_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite('g_img.png',g_img)
cv2.imwrite('c_img.png',c_img)
print("after image")
print(os.listdir(pr))
```

Output:




```
before image
['.ipynb_checkpoints', '3_2 semester', '4-1 semeter', '4_2 semester', 'bushra.lnk', 'CodeBlocks.lnk', 'cp', 'Desktop', 'deskto
p.ini', 'Eclipse IDE for Java Developers - 2023-12.lnk', 'hhhd.m4a', 'khaleda', 'Microsoft Edge.lnk', 'QA Objective.xlsx', 'the
sis', 'Visual Studio Code.lnk', 'Zoom.lnk']
after image
['.ipynb_checkpoints', '3_2 semester', '4-1 semeter', '4_2 semester', 'bushra.lnk', 'CodeBlocks.lnk', 'cp', 'c_img.png', 'Deskt
op', 'desktop.ini', 'Eclipse IDE for Java Developers - 2023-12.lnk', 'g_img.png', 'hhhd.m4a', 'khaleda', 'Microsoft Edge.lnk',
'QA Objective.xlsx', 'thesis', 'Visual Studio Code.lnk', 'Zoom.lnk']
```

Lab Report - 04

Task 1:

Using the for loop and glob library read all the image files and rename all of them according toloop.

Code:

```
import os
import glob
co=0
path='*.*'
os.chdir(r'F:\Github\Digital-Image-Processing-Lab\Lab 4\Code\Images')
print("before rename")
print(os.listdir())

for i in glob.glob(path):
    va='new'+str(co)+'.jpg'
    co+=1
    os.rename(i,va)

print("after rename")
print(os.listdir())
```

Output:

```
before rename
['new0.jpg', 'new4.jpg', 'new5.jpg', 'new6.jpg', 'new7.jpg', 'new8.jpg']
after rename
['new0.jpg', 'new1.jpg', 'new2.jpg', 'new3.jpg', 'new4.jpg', 'new5.jpg']
```

```
In [ ]: 1
```

Lab Report - 05

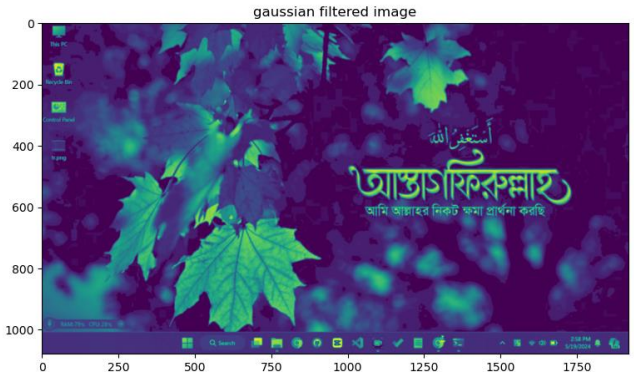
Task 1:

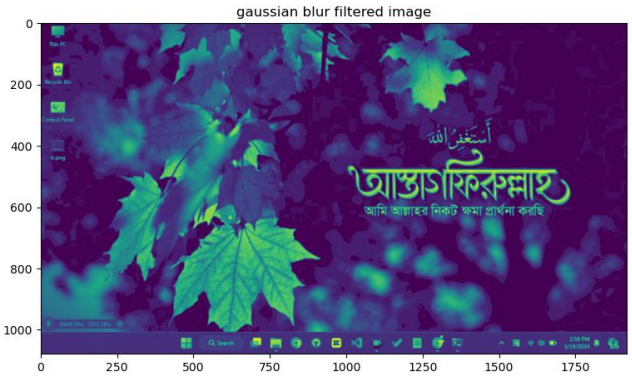
Take a screenshot of an image and apply gaussian filtering on the background of theimage and find edged from it.

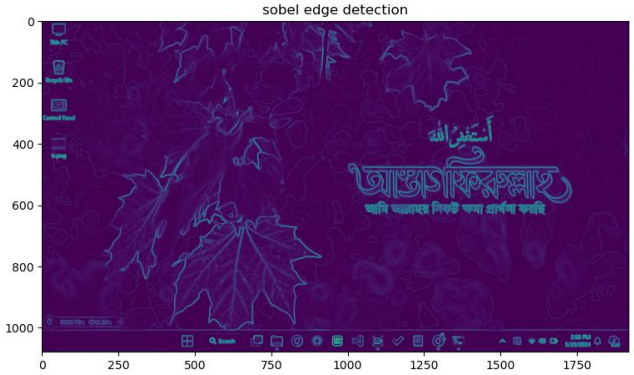
Code:

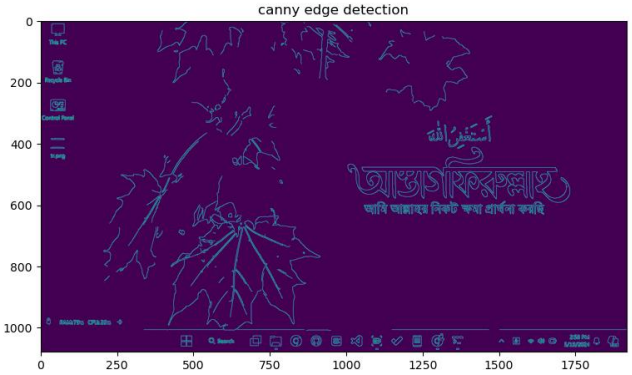
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from skimage import io
from skimage.filters import gaussian, sobel

img=(cv2.imread("ss.png",0))
fig=plt.figure(figsize=(20,20))
g1=gaussian(img, sigma=1, mode='constant', cval=0.0)
g2=cv2.GaussianBlur(img, (3,3), 0, borderType=cv2.BORDER_CONSTANT)
i1=fig.add_subplot(2,2,1)
i1.set_title("gaussian filtered image")
i1.imshow(g1)
i2=fig.add_subplot(2,2,2)
i2.set_title("gaussian blur filtered image")
i2.imshow(g2)

#sobel
so=sobel(g2)
fig=plt.figure(figsize=(20,20))
f1=fig.add_subplot(2,2,3)
f1.set_title("sobel edge detection")
f1.imshow(so)

#canny
ca=cv2.Canny(g2,100,200)
f2=fig.add_subplot(2,2,4)
f2.set_title("canny edge detection")
f2.imshow(ca)
```

Output:


gaussian filtered image
আস্তাগফিরুল্লাহ
আমি আল্লাহর নিকট ক্ষমা প্রার্থনা করছি


gaussian blur filtered image
আস্তাগফিরুল্লাহ
আমি আল্লাহর নিকট ক্ষমা প্রার্থনা করছি


sobel edge detection
আস্তাগফিরুল্লাহ


canny edge detection
আস্তাগফিরুল্লাহ

Lab Report - 06

Task 1:

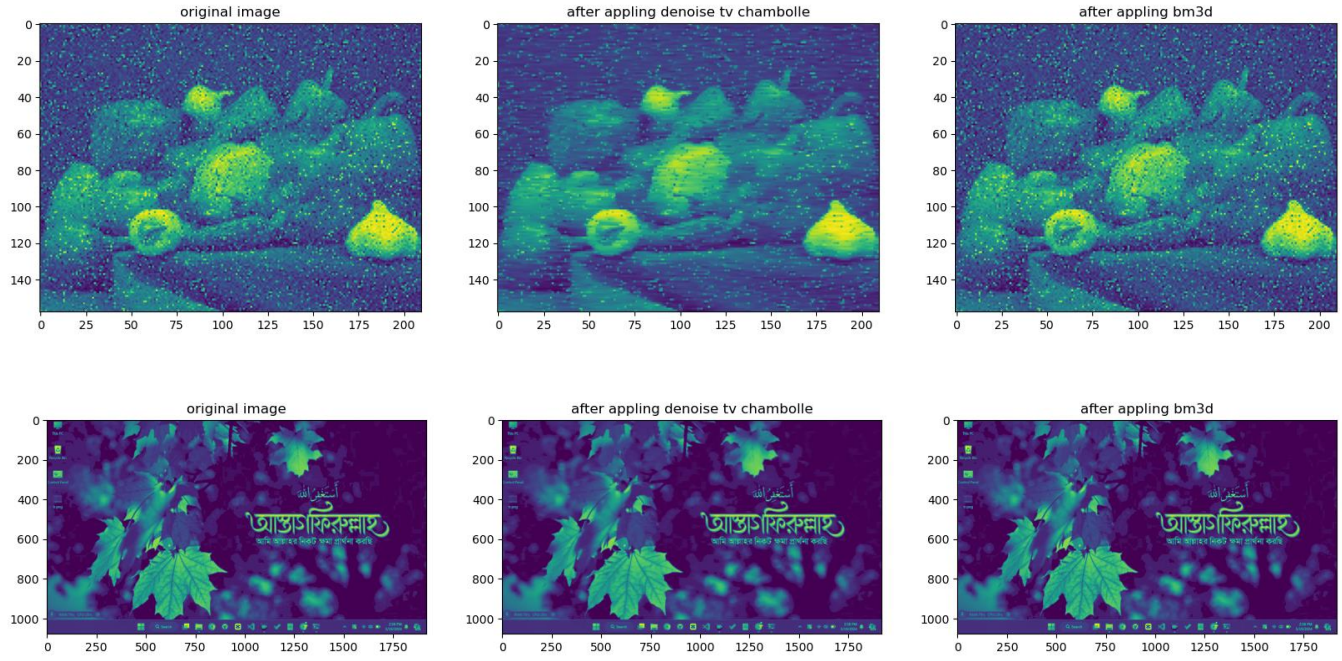
Take two images and compare between any two filtering techniques and write necessary comments on it mentioning which filtering techniques works well on the images.

Code:

```
import cv2
import bm3d
import matplotlib.pyplot as plt
from skimage.filters import median
from skimage.restoration import denoise_tv_chambolle

img=(cv2.imread("noise.jpg",0))
fig=plt.figure(figsize=(20,20))

v1=denoise_tv_chambolle(img, weight=0.1, eps=0.0002,n_iter_max=200, channel_axis=False)
b1=bm3d.bm3d(img, sigma_psd=0.2,stage_arg=bm3d.BM3DStages.HARD_THRESHOLDING)

i1=fig.add_subplot(2,3,1)
i1.set_title('original image')
i1.imshow(img)

i2=fig.add_subplot(2,3,2)
i2.set_title('after appling denoise tv chambolle')
i2.imshow(v1)

i3=fig.add_subplot(2,3,3)
i3.set_title('after appling bm3d')
i3.imshow(b1)

img1=(cv2.imread("ss.png",0))
fig=plt.figure(figsize=(20,20))

v1=denoise_tv_chambolle(img1, weight=0.1, eps=0.0002,n_iter_max=200, channel_axis=False)
b1=bm3d.bm3d(img1,sigma_psd=0.2,stage_arg=bm3d.BM3DStages.HARD_THRESHOLDING)

i1=fig.add_subplot(2,3,4)
i1.set_title('original image')
i1.imshow(img1)

i2=fig.add_subplot(2,3,5)
i2.set_title('after appling denoise tv chambolle')
i2.imshow(v1)
```

```
i3=fig.add_subplot(2,3,6)
i3.set_title('after appling bm3d')
i3.imshow(b1)
```

Output:



Lab Report - 07

Task 1:

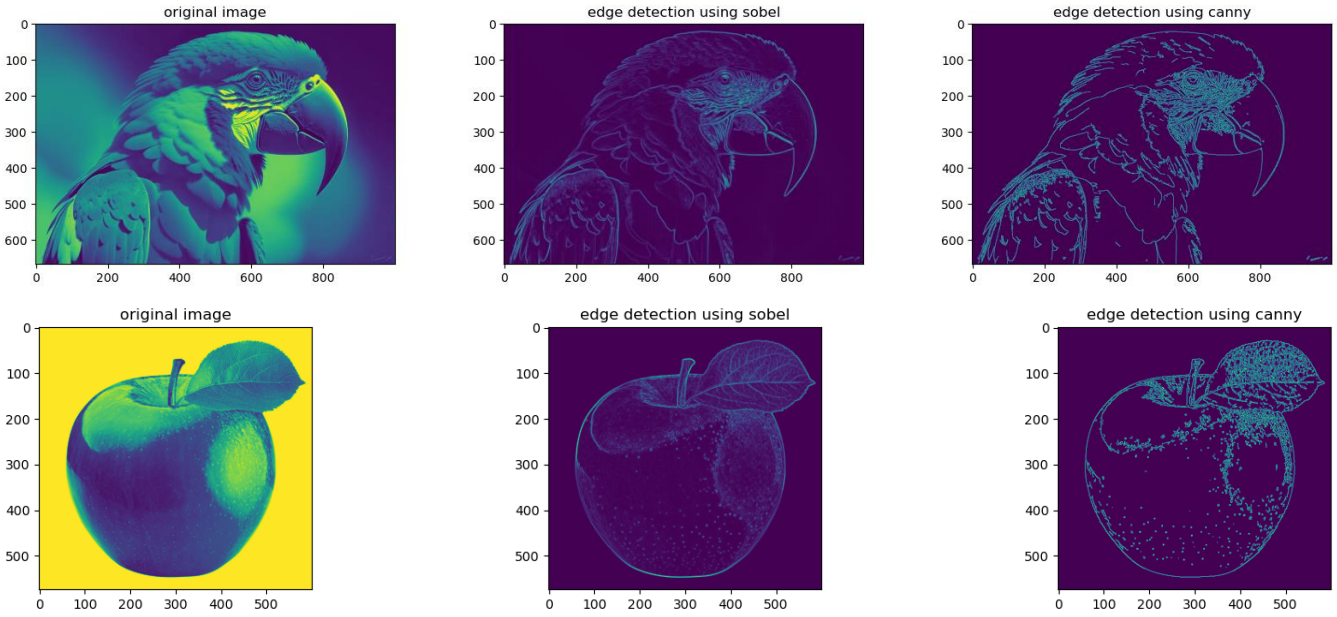
Take two images and compare between any two edge detection techniques and write necessary

Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import io
from skimage.filters import gaussian, sobel

img=(cv2.imread("new4.jpg",0))
fig=plt.figure(figsize=(20,8))

so=sobel(img)
ca=cv2.Canny(img,100,200)

i1=fig.add_subplot(2,3,1)
i1.set_title('original image')
i1.imshow(img)

i2=fig.add_subplot(2,3,2)
i2.set_title('edge detection using sobel')
i2.imshow(so)

i3=fig.add_subplot(2,3,3)
i3.set_title('edge detection using canny')
i3.imshow(ca)

img1=(cv2.imread("apple.jpg",0))
fig=plt.figure(figsize=(20,8))
so=sobel(img1)

ca=cv2.Canny(img1,100,200)
i1=fig.add_subplot(2,3,4)
i1.set_title('original image')
i1.imshow(img1)
```

```
i2=fig.add_subplot(2,3,5)
i2.set_title('edge detection using sobel')
i2.imshow(so)

i3=fig.add_subplot(2,3,6)
i3.set_title('edge detection using canny')
i3.imshow(ca)
```

Output:



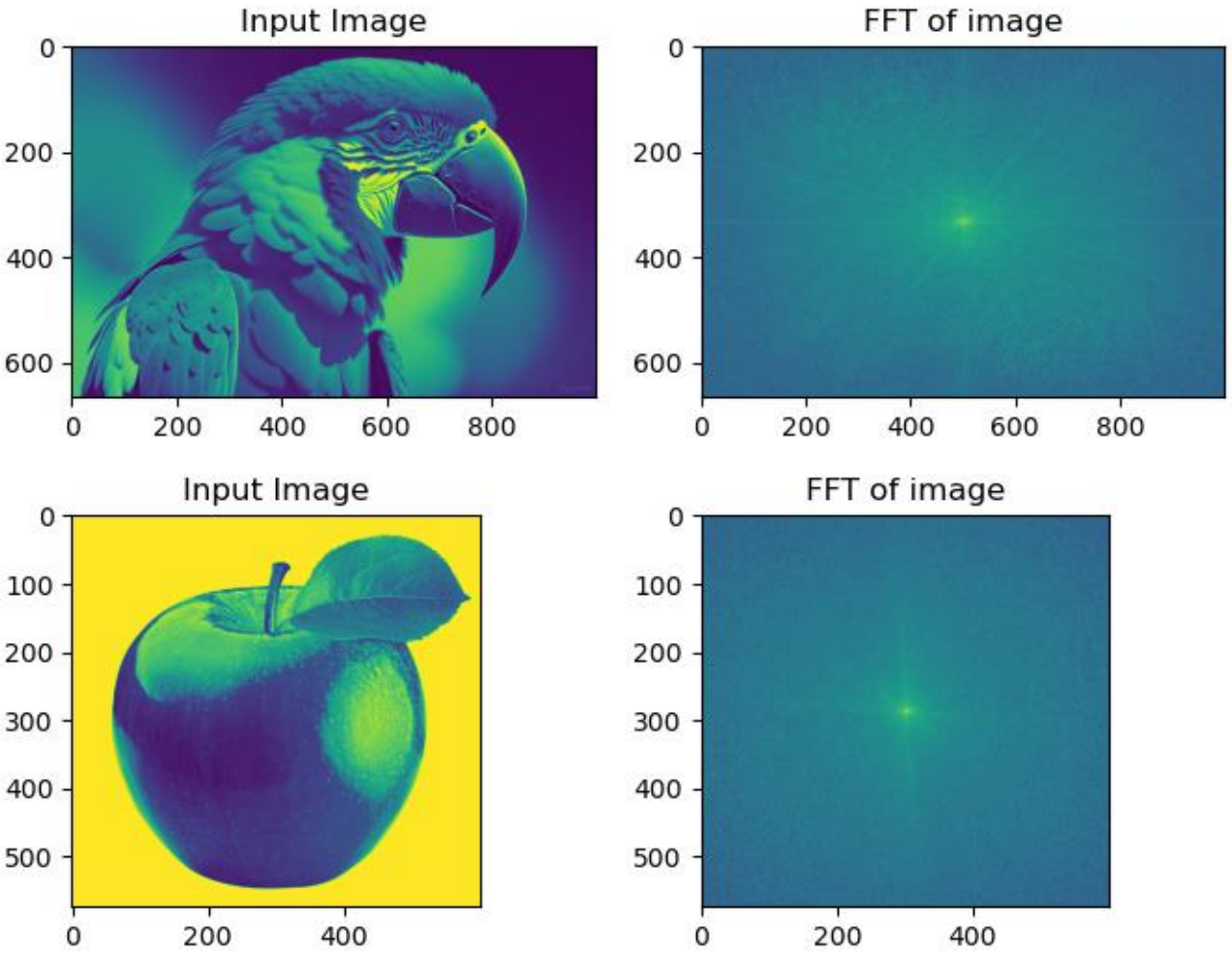
Task 2:

Apply Fourier transform on those images.

Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import io
from skimage.filters import gaussian, sobel

img = cv2.imread('new4.jpg', 0)
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20 * np.log((cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :,1]))+1)
fig = plt.figure(figsize=(8,6))
ax1 = fig.add_subplot(2,2,1)
ax1.imshow(img)

ax1.title.set_text('Input Image')
ax2 = fig.add_subplot(2,2,2)
ax2.imshow(magnitude_spectrum)
ax2.title.set_text('FFT of image')
plt.show()

img = cv2.imread('apple.jpg', 0)
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20 * np.log((cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :,1]))+1)
fig = plt.figure(figsize=(8,6))
ax1 = fig.add_subplot(2,2,3)
ax1.imshow(img)

ax1.title.set_text('Input Image')
ax2 = fig.add_subplot(2,2,4)
ax2.imshow(magnitude_spectrum)
ax2.title.set_text('FFT of image')
plt.show()
```

Output:



Task 3:

Apply three pass filters on them and write you comment mentioning which filtering process is more feasible to extract the actual image.

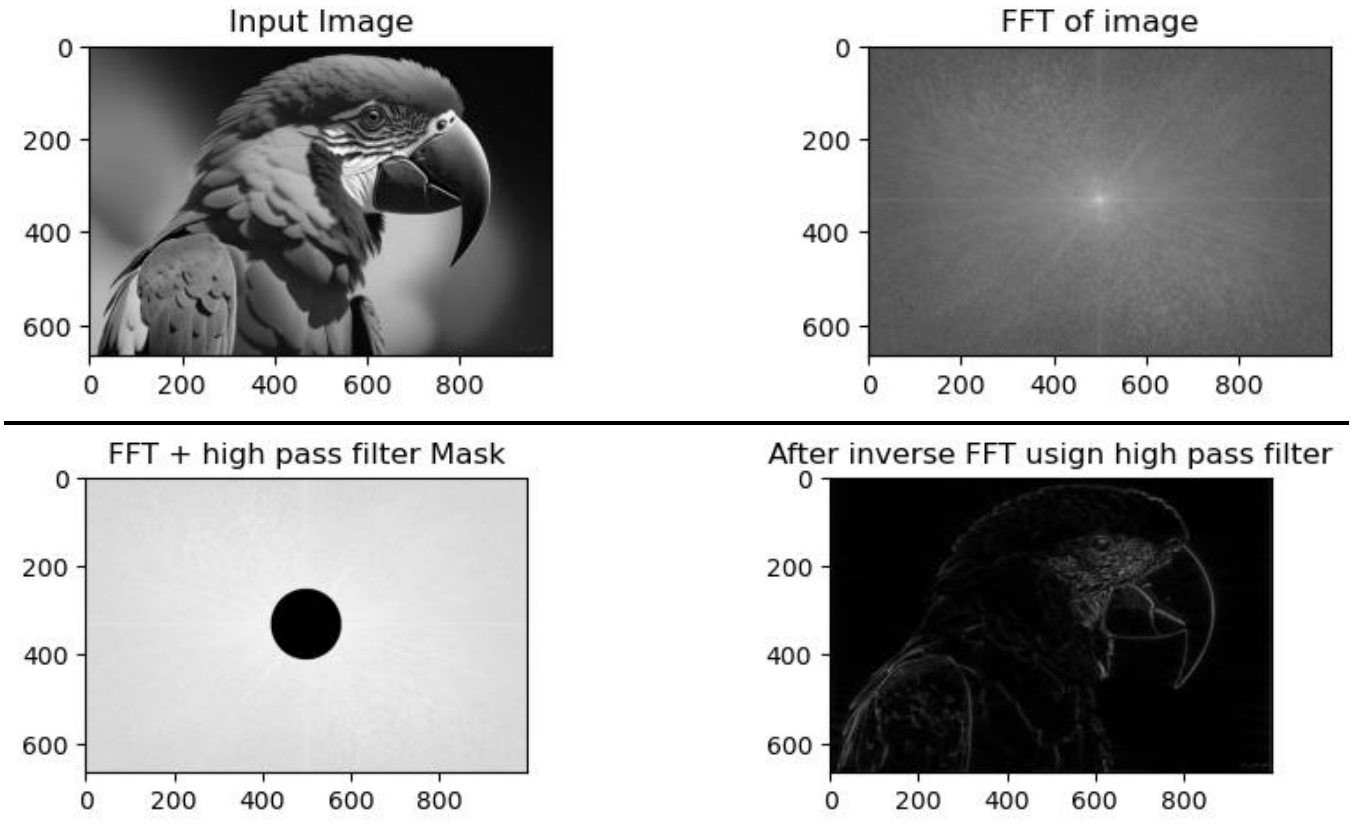
Code:

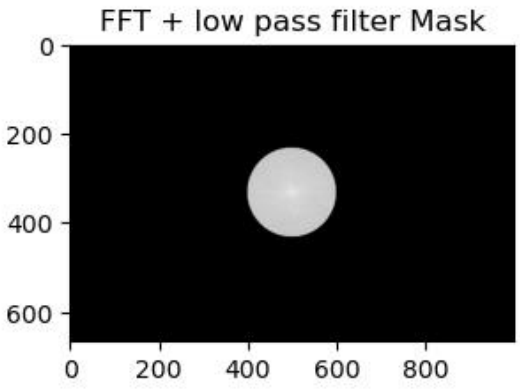
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import io
from skimage.filters import gaussian, sobel

img = cv2.imread('new4.jpg', 0)
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20 * np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))
fig = plt.figure(figsize=(10,10))

ax1 = fig.add_subplot(4,2,1)
ax1.imshow(img, cmap='gray')
ax1.title.set_text('Input Image')

ax2 = fig.add_subplot(4,2,2)
ax2.imshow(magnitude_spectrum, cmap='gray')
ax2.title.set_text('FFT of image')

# Circular HPF mask, center circle is 0, remaining all ones
rows, cols = img.shape
crow, ccol = int(rows / 2), int(cols / 2)
mask = np.ones((rows, cols, 2), np.uint8)
r = 80
center = [crow, ccol]
x, y = np.ogrid[:rows, :cols]
mask_area = (x - center[0]) ** 2 + (y - center[1]) ** 2 <= r*r
mask[mask_area] = 0
fshift = dft_shift * mask
epsilon = 1e-8
fshift_mask_mag = 20 * np.log(cv2.magnitude(fshift[:, :, 0], fshift[:, :, 1]) + epsilon)
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])
fig = plt.figure(figsize=(10,10))
ax3 = fig.add_subplot(4,2,3)
ax3.imshow(fshift_mask_mag, cmap='gray')
```

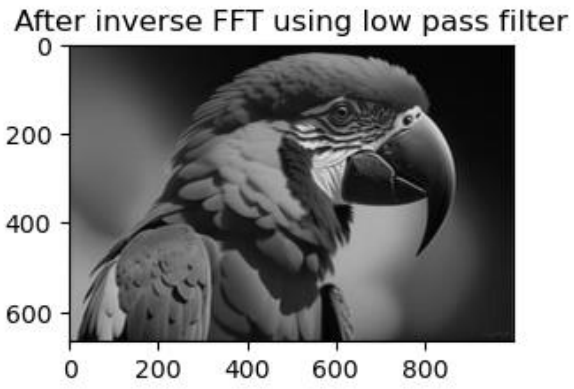
```
ax3.title.set_text('FFT + high pass filter Mask')
ax4 = fig.add_subplot(4,2,4)
ax4.imshow(img_back, cmap='gray')
ax4.title.set_text('After inverse FFT usign high pass filter')
plt.show()

img = cv2.imread('new4.jpg', 0)
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum = 20 * np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))

# Circular LPF mask, center circle is 1, remaining all zeros
rows, cols = img.shape
crow, ccol = int(rows / 2), int(cols / 2)
mask = np.zeros((rows, cols, 2), np.uint8)
r = 100
center = [crow, ccol]
x, y = np.ogrid[:rows, :cols]
mask_area = (x - center[0]) ** 2 + (y - center[1]) ** 2 <= r*r
mask[mask_area] = 1
fshift = dft_shift * mask
epsilon = 1e-8
fshift_mask_mag = 20 * np.log(cv2.magnitude(fshift[:, :, 0], fshift[:, :, 1]) + epsilon)
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])
fig = plt.figure(figsize=(10,10))
ax3 = fig.add_subplot(4,2,5)
ax3.imshow(fshift_mask_mag, cmap='gray')
ax3.title.set_text('FFT + low pass filter Mask')
ax4 = fig.add_subplot(4,2,6)
ax4.imshow(img_back, cmap='gray')
ax4.title.set_text('After inverse FFT using low pass filter')
plt.show()

img = cv2.imread('new4.jpg', 0)
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum = 20 * np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))

# Band Pass Filter - Concentric circle mask, only the points living in concentric circle are ones
rows, cols = img.shape
crow, ccol = int(rows / 2), int(cols / 2)
mask = np.zeros((rows, cols, 2), np.uint8)
r_out = 80
r_in = 10
```

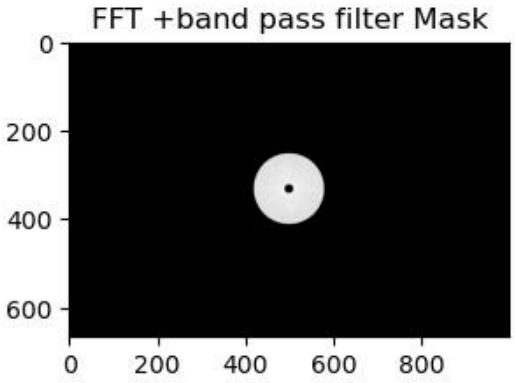
```
center = [crow, ccol]
x, y = np.ogrid[:rows, :cols]
mask_area = np.logical_and(((x - center[0]) ** 2 + (y - center[1]) ** 2 >= r_in ** 2),
((x - center[0]) ** 2 + (y - center[1]) ** 2 <= r_out ** 2))
mask[mask_area] = 1
fshift = dft_shift * mask
epsilon = 1e-8
fshift_mask_mag = 20 * np.log(cv2.magnitude(fshift[:, :, 0], fshift[:, :, 1]) + epsilon)
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])
fig = plt.figure(figsize=(10,10))
ax3 = fig.add_subplot(4,2,7)
ax3.imshow(fshift_mask_mag, cmap='gray')
ax3.title.set_text('FFT +band pass filter Mask')
ax4 = fig.add_subplot(4,2,8)
ax4.imshow(img_back, cmap='gray')
ax4.title.set_text('After inverse FFT using band pass filter')
plt.show()
```

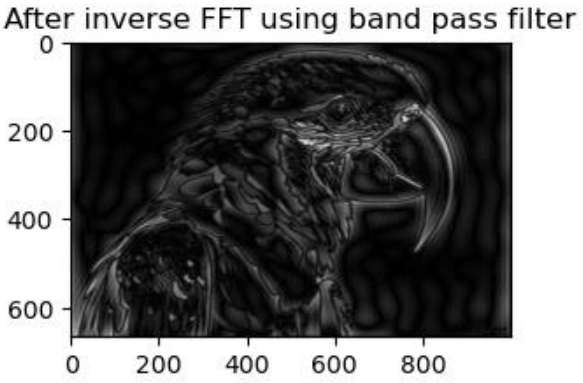
Output:




FFT + low pass filter Mask
0
200
400
600
0
200
400
600
800


After inverse FFT using low pass filter
0
200
400
600
0
200
400
600
800


FFT +band pass filter Mask
0
200
400
600
0
200
400
600
800


After inverse FFT using band pass filter
0
200
400
600
0
200
400
600
800

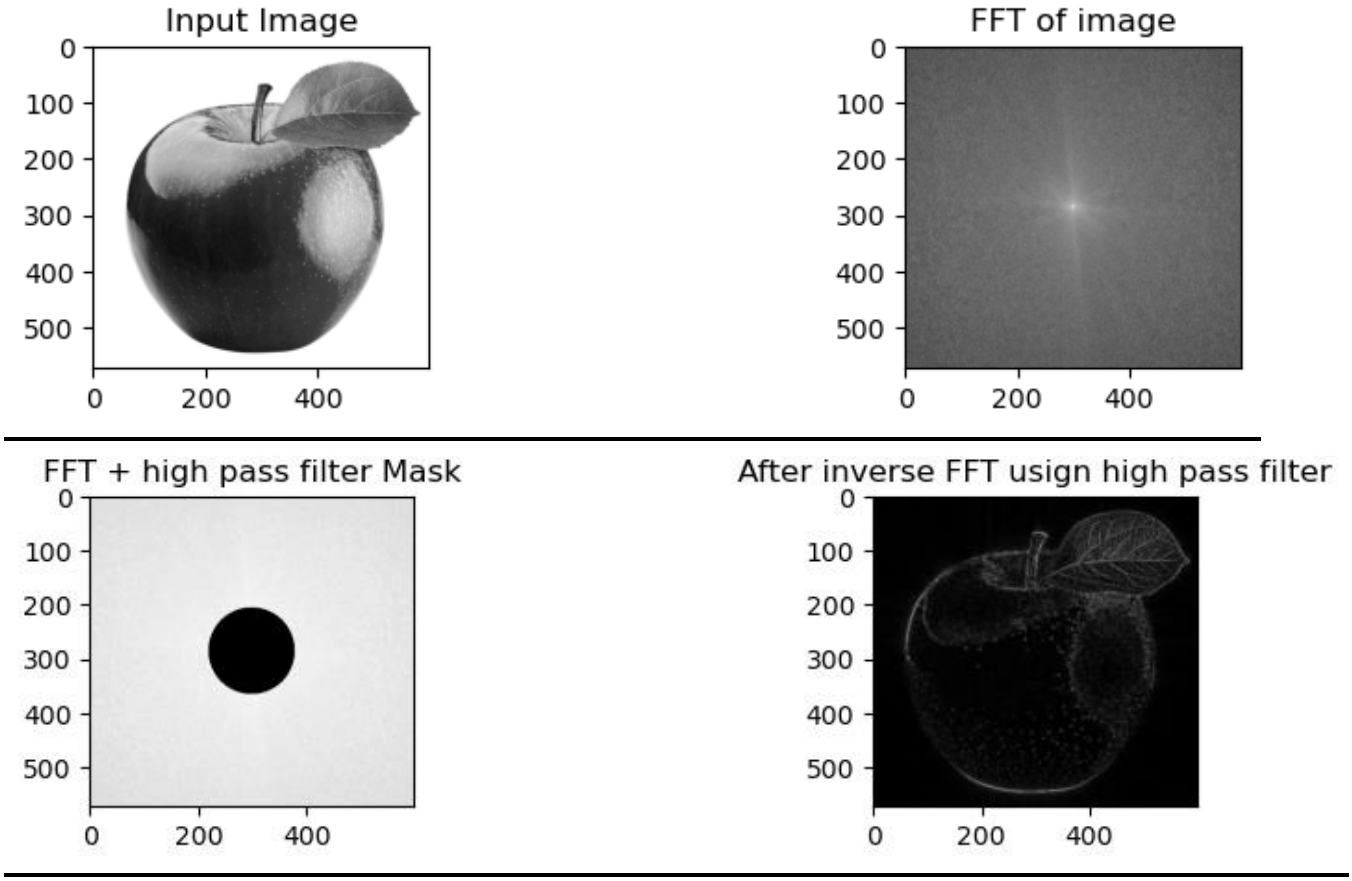
Code for another image:

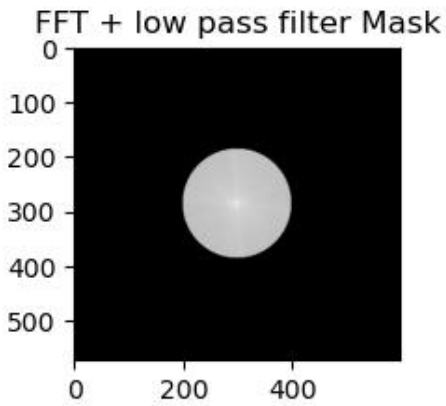
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import io
from skimage.filters import gaussian, sobel

img = cv2.imread('apple.jpg', 0)
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20 * np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))
fig = plt.figure(figsize=(10,10))

ax1 = fig.add_subplot(4,2,1)
ax1.imshow(img, cmap='gray')
ax1.title.set_text('Input Image')

ax2 = fig.add_subplot(4,2,2)
ax2.imshow(magnitude_spectrum, cmap='gray')
ax2.title.set_text('FFT of image')

# Circular HPF mask, center circle is 0, remaining all ones
rows, cols = img.shape
crow, ccol = int(rows / 2), int(cols / 2)
mask = np.ones((rows, cols, 2), np.uint8)
r = 80
center = [crow, ccol]
x, y = np.ogrid[:rows, :cols]
mask_area = (x - center[0]) ** 2 + (y - center[1]) ** 2 <= r*r
mask[mask_area] = 0
fshift = dft_shift * mask
epsilon = 1e-8
fshift_mask_mag = 20 * np.log(cv2.magnitude(fshift[:, :, 0], fshift[:, :, 1]) + epsilon)
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])
fig = plt.figure(figsize=(10,10))
ax3 = fig.add_subplot(4,2,3)
ax3.imshow(fshift_mask_mag, cmap='gray')
ax3.title.set_text('FFT + high pass filter Mask')
ax4 = fig.add_subplot(4,2,4)
ax4.imshow(img_back, cmap='gray')
ax4.title.set_text('After inverse FFT usign high pass filter')
```

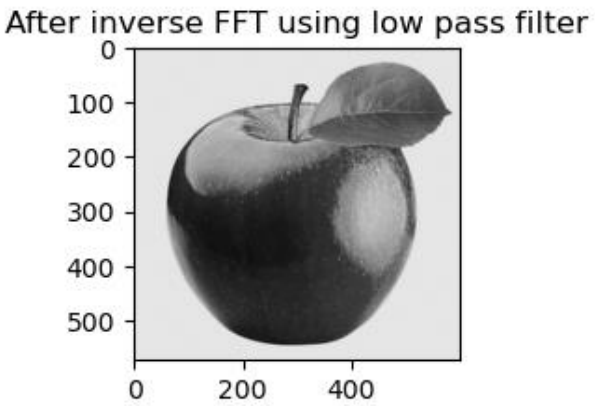
```
plt.show()

# Circular LPF mask, center circle is 1, remaining all zeros
rows, cols = img.shape
crow, ccol = int(rows / 2), int(cols / 2)
mask = np.zeros((rows, cols, 2), np.uint8)
r = 100
center = [crow, ccol]
x, y = np.ogrid[:rows, :cols]
mask_area = (x - center[0]) ** 2 + (y - center[1]) ** 2 <= r*r
mask[mask_area] = 1
fshift = dft_shift * mask
epsilon = 1e-8
fshift_mask_mag = 20 * np.log(cv2.magnitude(fshift[:, :, 0], fshift[:, :, 1]) + epsilon)
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])
fig = plt.figure(figsize=(10,10))
ax3 = fig.add_subplot(4,2,5)
ax3.imshow(fshift_mask_mag, cmap='gray')
ax3.title.set_text('FFT + low pass filter Mask')
ax4 = fig.add_subplot(4,2,6)
ax4.imshow(img_back, cmap='gray')
ax4.title.set_text('After inverse FFT using low pass filter')
plt.show()

# Band Pass Filter - Concentric circle mask, only the points living in concentric circle are ones
rows, cols = img.shape
crow, ccol = int(rows / 2), int(cols / 2)
mask = np.zeros((rows, cols, 2), np.uint8)
r_out = 80
r_in = 10
center = [crow, ccol]
x, y = np.ogrid[:rows, :cols]
mask_area = np.logical_and(((x - center[0]) ** 2 + (y - center[1]) ** 2 >= r_in ** 2),
((x - center[0]) ** 2 + (y - center[1]) ** 2 <= r_out ** 2))
mask[mask_area] = 1
fshift = dft_shift * mask
epsilon = 1e-8
fshift_mask_mag = 20 * np.log(cv2.magnitude(fshift[:, :, 0], fshift[:, :, 1]) + epsilon)
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])
fig = plt.figure(figsize=(10,10))
ax3 = fig.add_subplot(4,2,7)
ax3.imshow(fshift_mask_mag, cmap='gray')
```

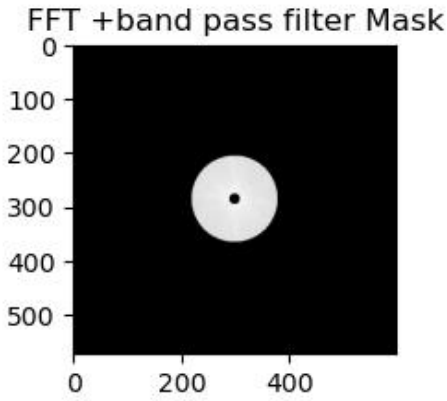
```
ax3.title.set_text('FFT +band pass filter Mask')
ax4 = fig.add_subplot(4,2,8)
ax4.imshow(img_back, cmap='gray')
ax4.title.set_text('After inverse FFT using band pass filter')
plt.show()
```

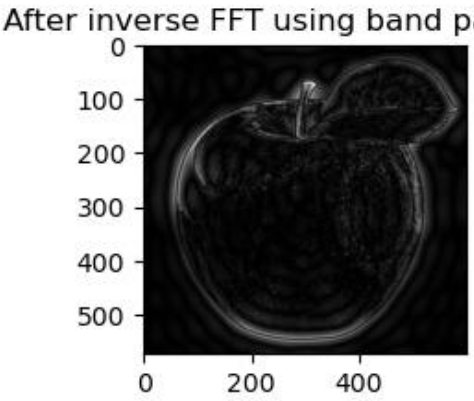
Output:




FFT + low pass filter Mask
0
100
200
300
400
500
0
200
400


After inverse FFT using low pass filter
0
100
200
300
400
500
0
200
400


FFT +band pass filter Mask
0
100
200
300
400
500
0
200
400


After inverse FFT using band pass filter
0
100
200
300
400
500
0
200
400

The End