Computer Vision

Assignment 2

Name - Ritik Gautam

Roll No - B180630CS

Q. Apply the following filtering operation on an input image(Choose your own photo as input) and display both the input and out images. Enhance the image by,

```
i) Low pass filtering
ii) Sobel operator, (3x3 & 5x5)
iii) Laplacian operator. (3x3 & 5x5)
iv) LOG
v) Canny Edge Detection
vi) High-boost filtering
```

Solution

```
import cv2
import numpy as np
from PIL import Image,ImageDraw, ImageFilter
import matplotlib.pyplot as plt


img = cv2.imread('/content/My Photo.png')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

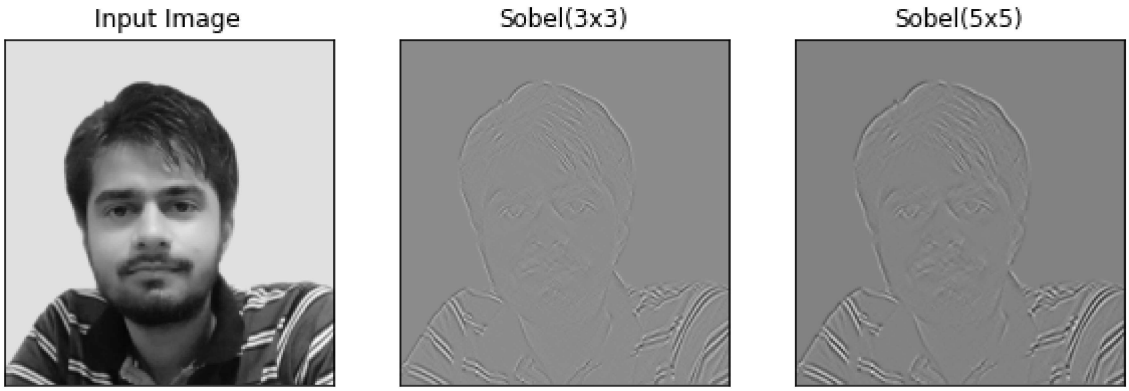
i) Low Pass Filtering

```
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
dft = cv2.dft(np.float32(img_gray),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)   # shift the zero-frequncy component to the center of th
rows, cols = img_gray.shape
crow,ccol = rows//2 , cols//2
mask = np.zeros((rows,cols,2),np.uint8) # create a mask first, center square is 1, remaini
mask[crow-30:crow+30, ccol-30:ccol+30] = 1
fshift = dft_shift*mask # apply the mask the inverse DFT
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])
```

```
plt.figure(figsize=(8,6))
plt.subplot(121),plt.imshow(img_gray, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Low Pass Filter'), plt.xticks([]), plt.yticks([])
plt.show()
```

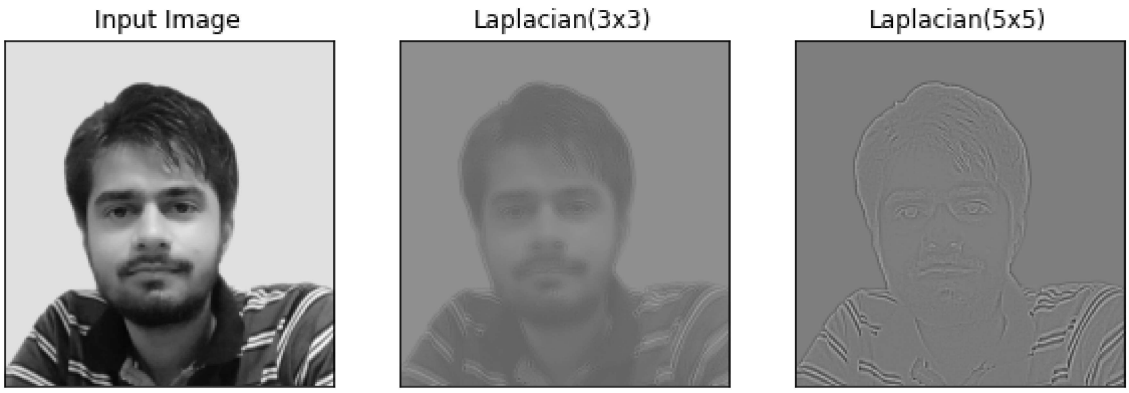
ii) Sobel Operator

```
# sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Edge De
# sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Edge De
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
new_image1 = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=3) # Combined X
img_blur = cv2.GaussianBlur(img_gray, (5,5), 0)
new_image2 = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) # Combined X
plt.figure(figsize=(10,6))
plt.subplot(131), plt.imshow(img_gray, cmap='gray'),plt.title('Input Image')
plt.xticks([]), plt.yticks([])
plt.subplot(132), plt.imshow(new_image1, cmap='gray'),plt.title('Sobel(3x3)')
plt.xticks([]), plt.yticks([])
plt.subplot(133), plt.imshow(new_image2, cmap='gray'),plt.title('Sobel(5x5)')
plt.xticks([]), plt.yticks([])
plt.show()
```



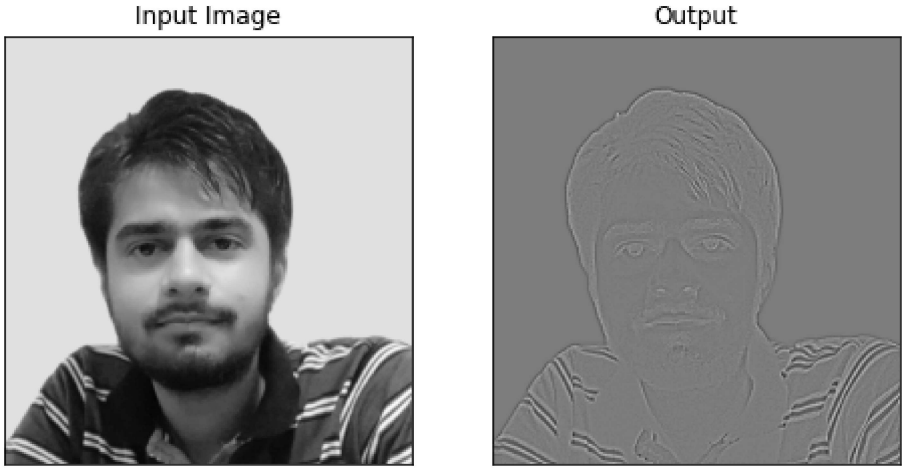
iii) Laplacian Operator

```
new_image1 = cv2.Laplacian(img_gray,cv2.CV_64F,ksize=3)
new_image2 = cv2.Laplacian(img_gray,cv2.CV_64F,ksize=5)
plt.figure(figsize=(10,6))
plt.subplot(131), plt.imshow(img_gray, cmap='gray'),plt.title('Input Image')
plt.xticks([]), plt.yticks([])
plt.subplot(132), plt.imshow(img_gray + new_image1, cmap='gray'),plt.title('Laplacian(3x3)
plt.xticks([]), plt.yticks([])
plt.subplot(133), plt.imshow(img_gray + new_image2, cmap='gray'),plt.title('Laplacian(5x5)
plt.xticks([]), plt.yticks([])
plt.show()
```



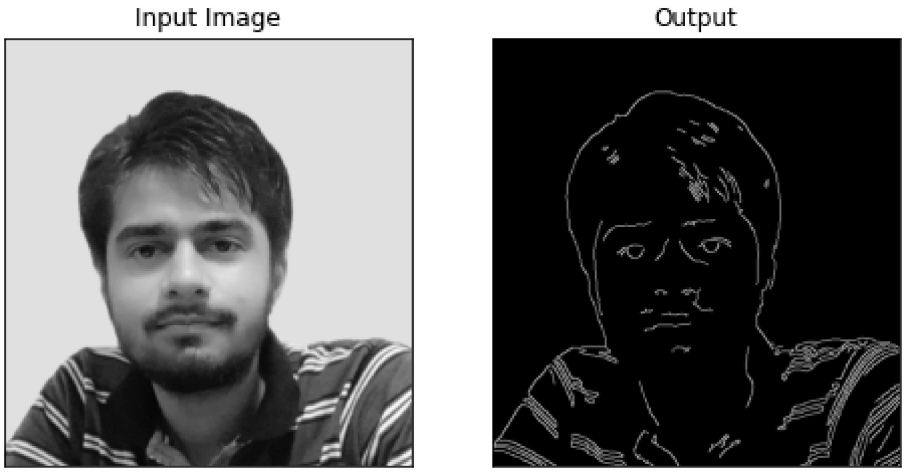
iv) Laplacian of Gaussian

```
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
new_image = cv2.Laplacian(img_blur,cv2.CV_64F)
new_image = new_image/new_image.max()
plt.figure(figsize=(8,6))
plt.subplot(121),plt.imshow(img_gray, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(new_image, cmap = 'gray')
plt.title('Output'), plt.xticks([]), plt.yticks([])
plt.show()
```



v) Canny Edge Detection

```
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200)
plt.figure(figsize=(8,6))
plt.subplot(121),plt.imshow(img_gray, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges, cmap = 'gray')
plt.title('Output'), plt.xticks([]), plt.yticks([])
plt.show()
```



vi) High-boost filtering

```
img_blur = cv2.GaussianBlur(img, (7,7), 0)
hboost_image = cv2.addWeighted(img, 3, img_blur, -2, 0)
plt.figure(figsize=(8,6))
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(hboost_image, cmap = 'gray')
plt.title('Output'), plt.xticks([]), plt.yticks([])
plt.show()
```