TRIBHUVAN UNIVERSITY

Sagarmatha College of Science & Technology

Lab Report On: Image Processing (CSC 321)

Lab Report No.: 01 - 08

Date: 2077 – 10 - 21

SUBMITTED BY

Name: Ramesh Neupane

Roll no.: 37

SUBMITTED TO

CSIT Department

LAB – 01

OBJECTIVES

To implement some spatial transformation function and histogram equalization

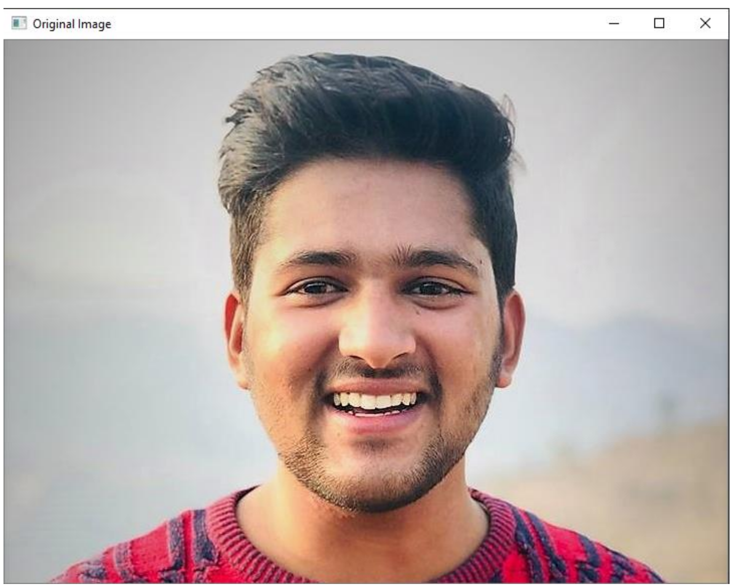
PROGRAMS

Program 1: Complement of an image

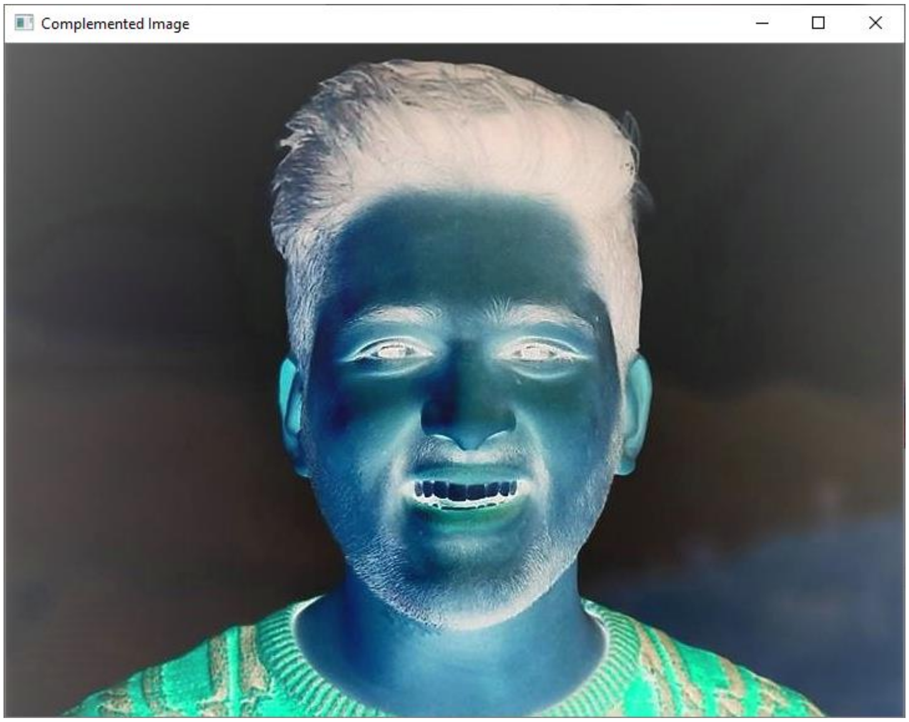
```
import cv2

img = cv2.imread('rn.jpg')
cv2.imshow('Original Image', img)
comp_img = 255 - img
cv2.imshow("Complemented Image", comp_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

i)



ii)

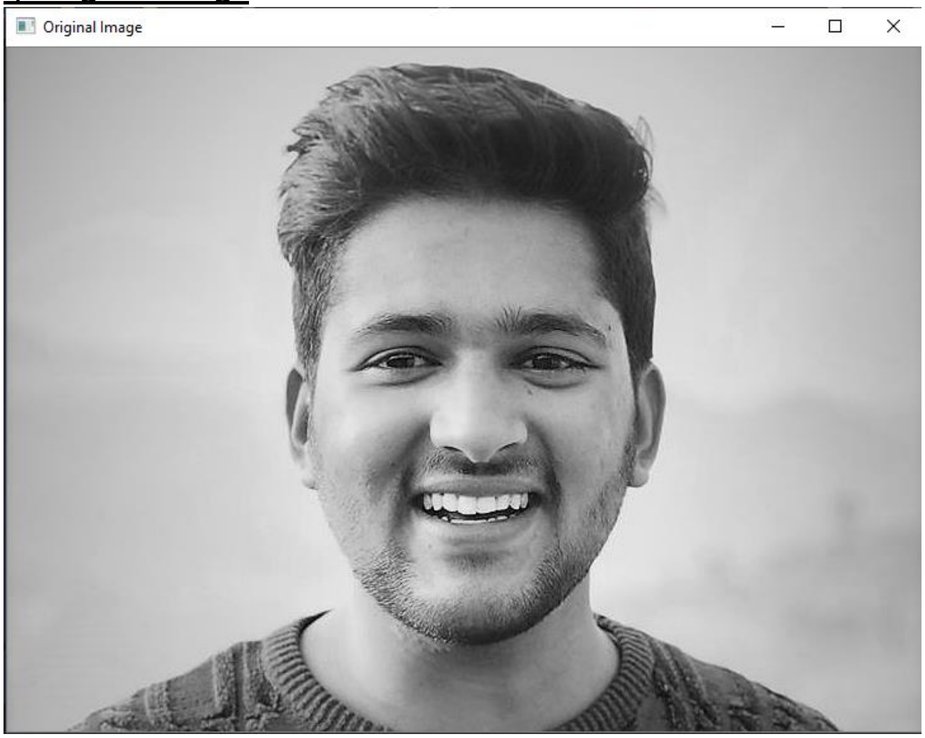


Program2: To check different values of threshold

```
import cv2 as cv
img = cv.imread('rn.jpg',0)
ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
ret,thresh2 = cv.threshold(img,127,255,cv.THRESH_BINARY_INV)
ret,thresh3 = cv.threshold(img,127,255,cv.THRESH_TRUNC)
ret,thresh4 = cv.threshold(img,127,255,cv.THRESH_TOZERO)
ret,thresh5 = cv.threshold(img,127,255,cv.THRESH_TOZERO_INV)
titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
   cv.imshow(titles[i], images[i])
cv.waitKey(0)
cv.destroyAllWindows()
```

Output:

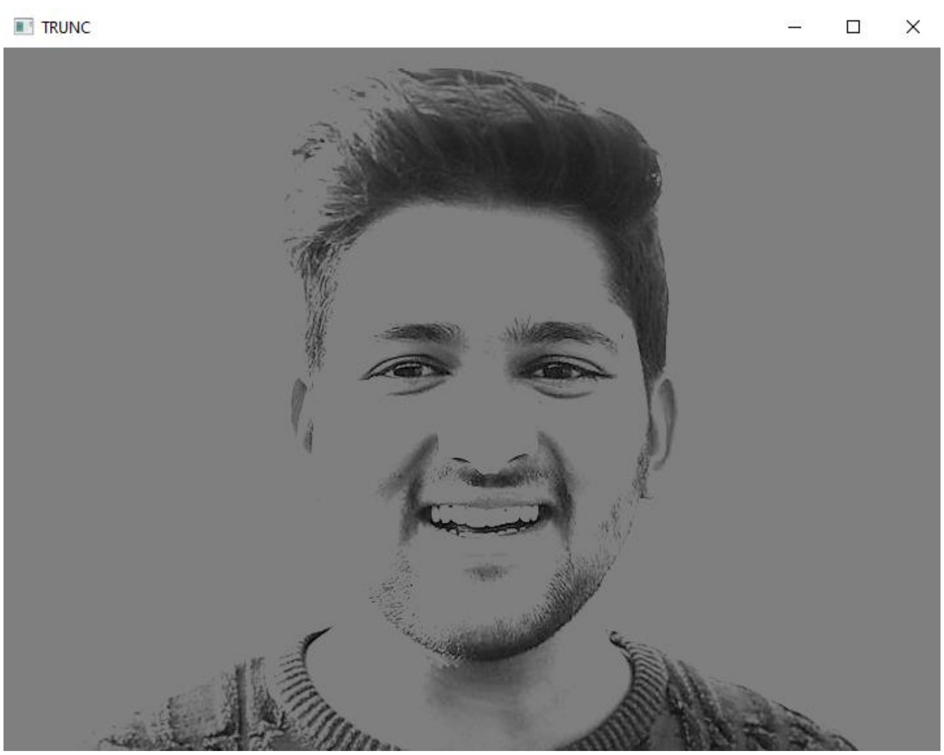
i) Original Image



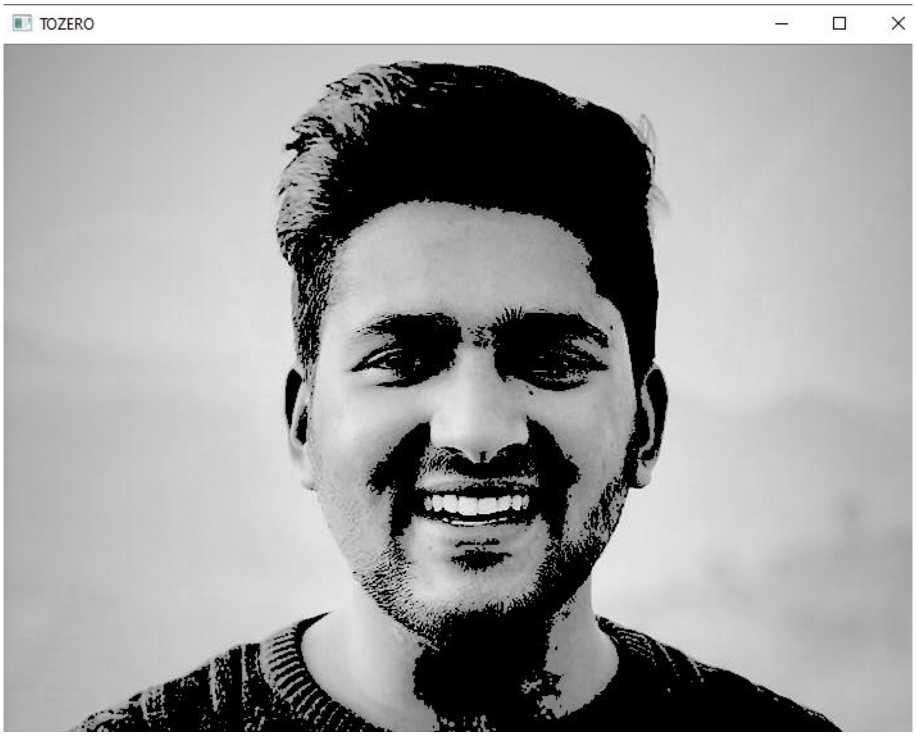
ii) Threshold binary

iii) Thresold binary inverse

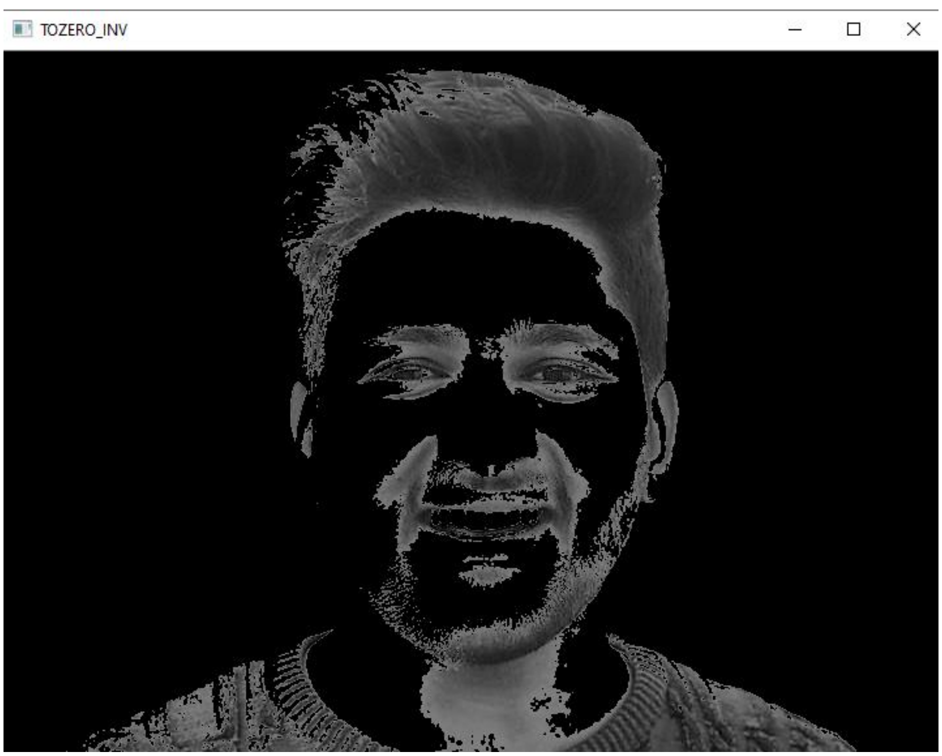
iv) Threshold truncate



v) Threshold tozero



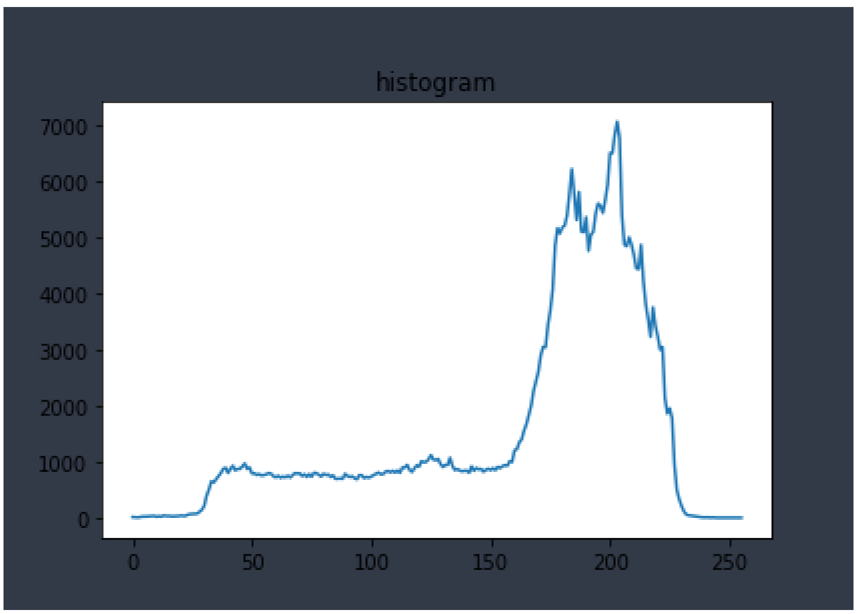
vi)Threshold tozero inverse



Program 3: Histogram calculation

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('rn.jpg', 0)
img_hist = cv2.calcHist([img], [0], None, [256], [0, 256])
plt.plot(img_hist)
plt.title('histogram')
```

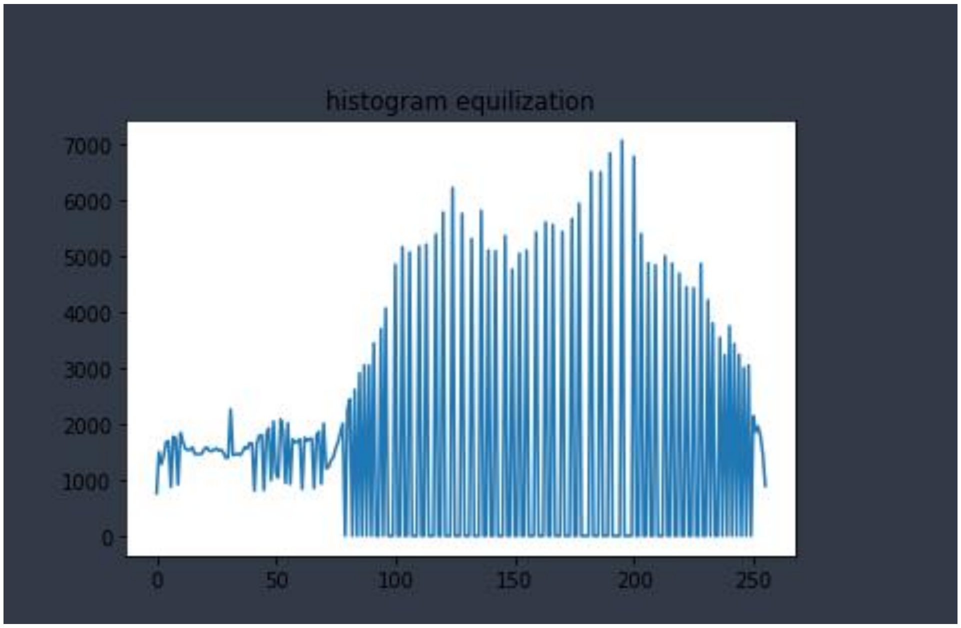
Output:

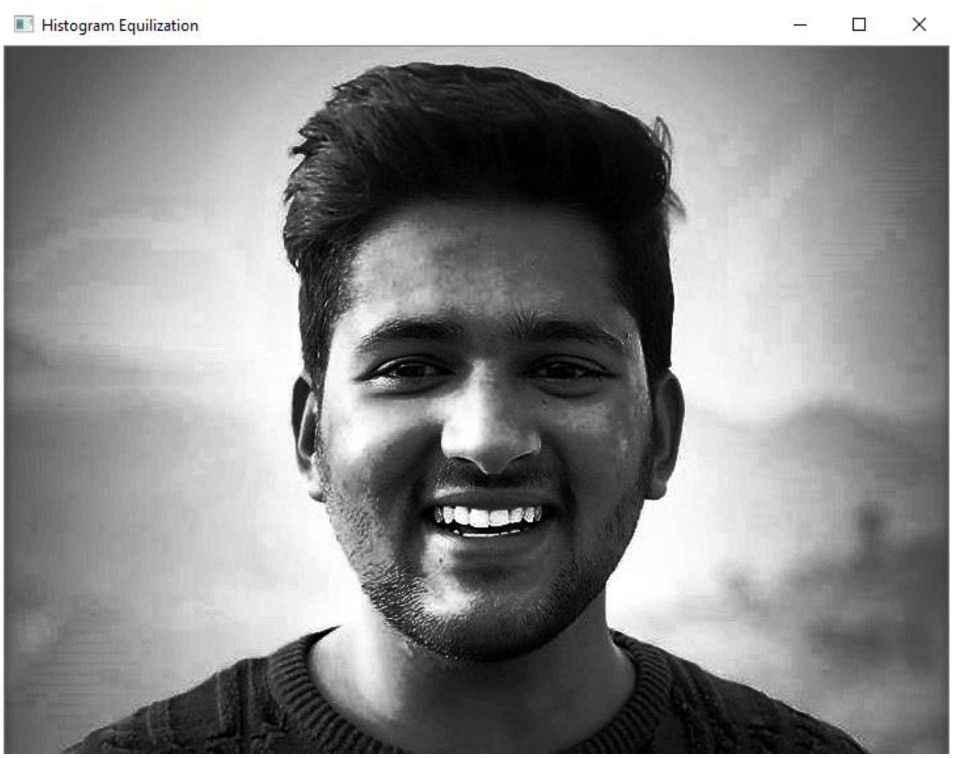


Program 4: Histogram equalization

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('rn.jpg', 0)
equi_hist = cv2.equalizeHist(img)
img_hist_eq = cv2.calcHist([equi_hist], [0], None, [256], [0, 256])
plt.plot(img_hist_eq)
plt.title('histogram equilization')
```

Output:





CONCLUSION

Hence, we're able to implement some spatial transformation (thresholding) functions and histogram equalization.

LAB - 02

OBJECTIVES

To implement some smoothing filters and to implement opening and closing of an image

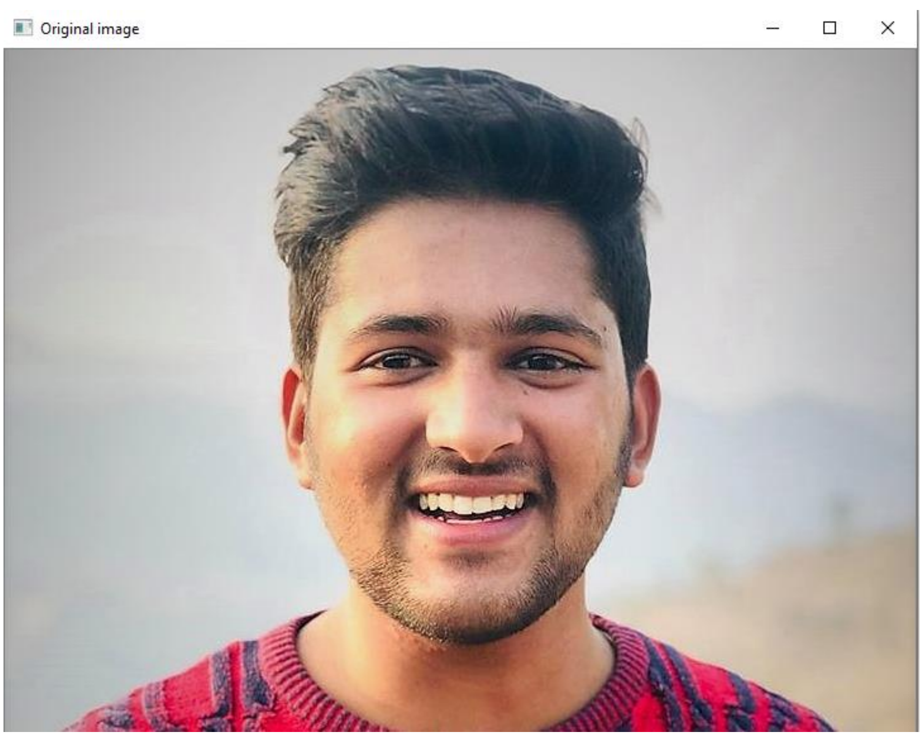
PROGRAMS

Program 1: Smoothing or average filtering

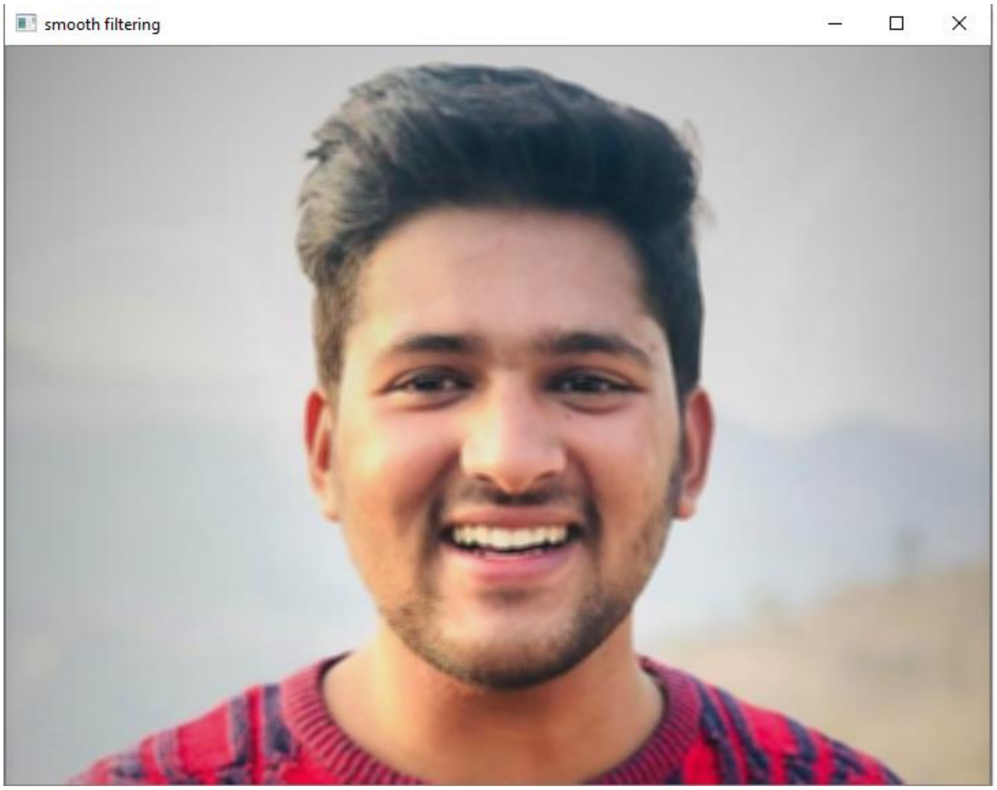
```
import cv2

img = cv2.imread('rn.jpg')
smooth_img = cv2.boxFilter(img, -1, (5, 5))
gaussian_blur = cv2.GaussianBlur(img,(5,5),0)
median_filter = cv2.medianBlur(img, 7)
cv2.imshow('Original image', img)
cv2.imshow('smooth filtering', smooth_img)
cv2.imshow('Gaussian blur', gaussian_blur)
cv2.imshow('Median blur', median_filter)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

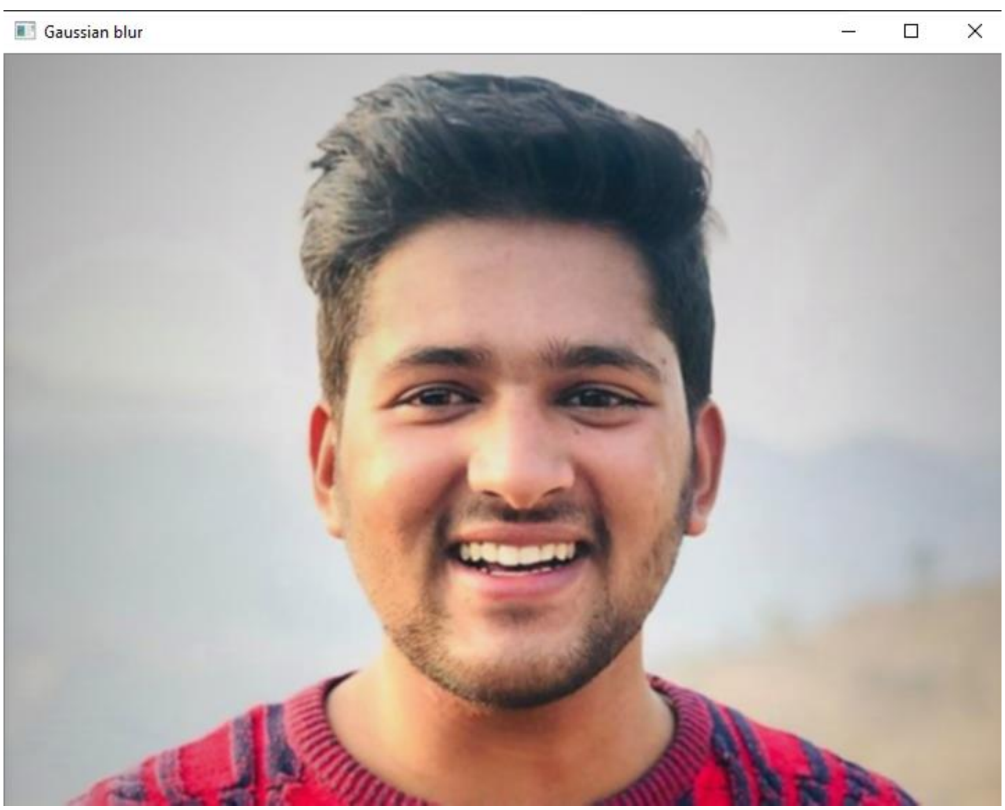
i)



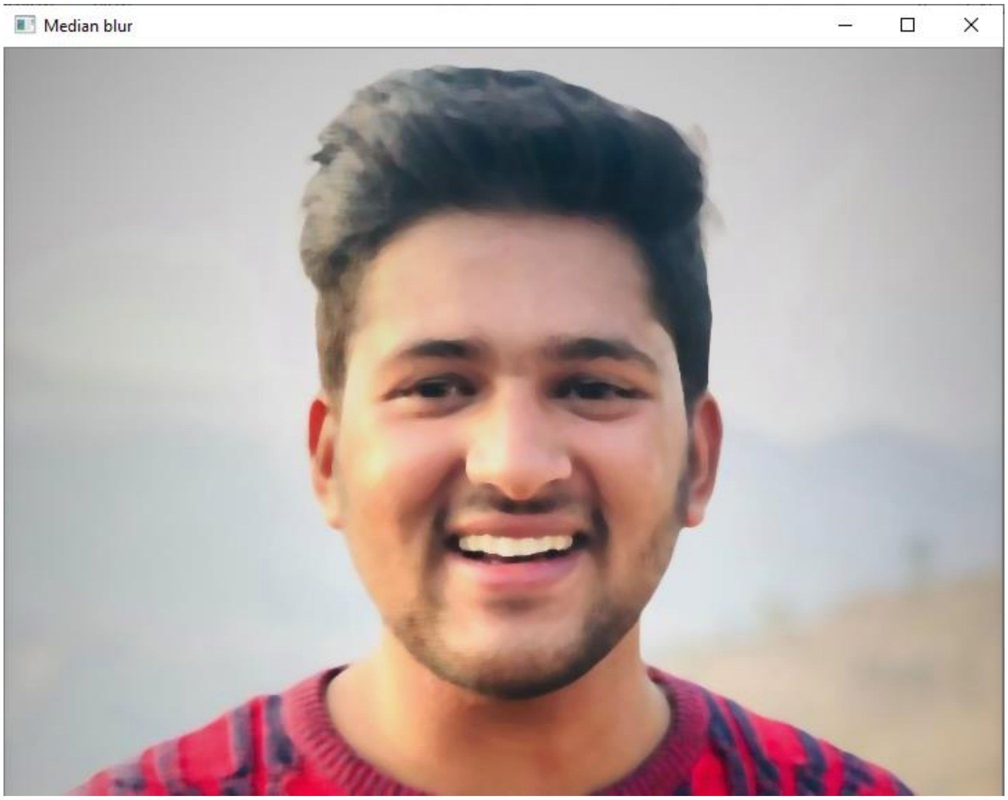
ii)



iii)



iv)

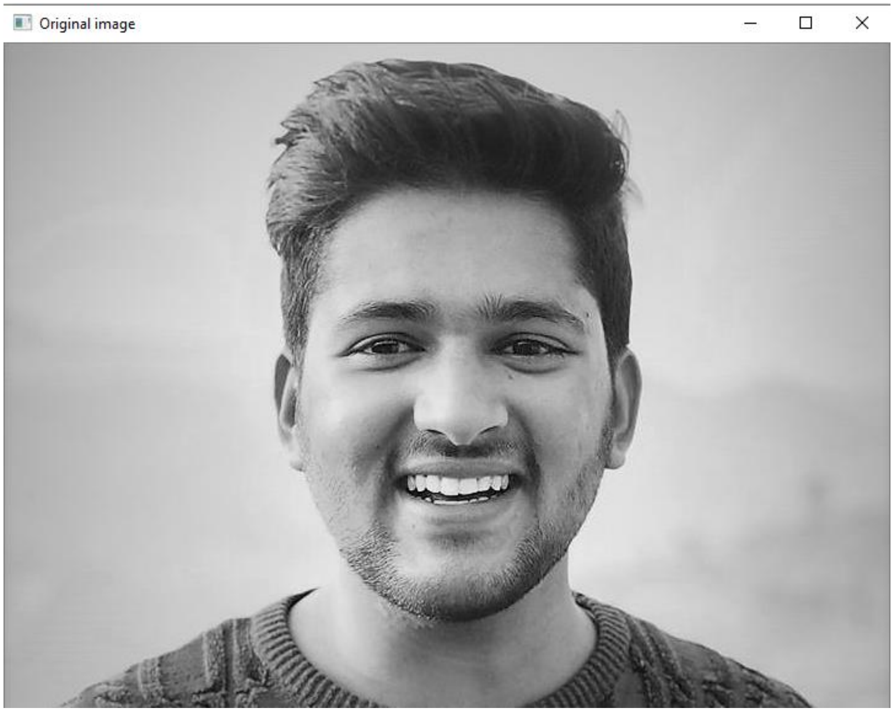


Program 2: Averaging in multidimensional matrix

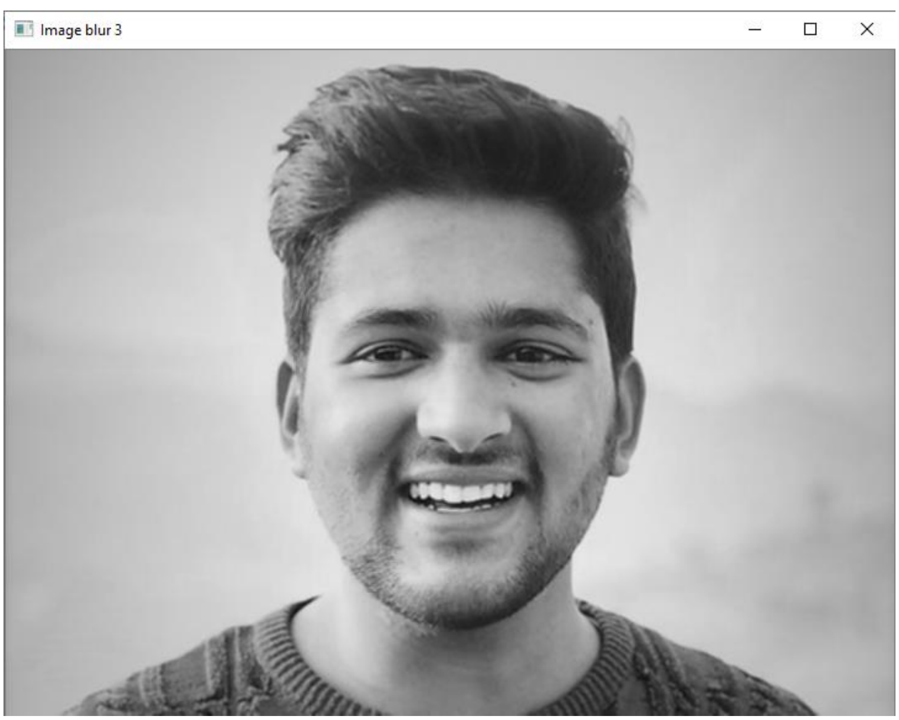
```
import cv2

img = cv2.imread('rn.jpg', 0)
img_blur3 = cv2.blur(img, (3, 3))
img_blur7 = cv2.blur(img, (7, 7))
bilateral_blur = cv2.bilateralFilter(img, 9, 50, 10)
cv2.imshow('Original image', img)
cv2.imshow('Image blur 3', img_blur3)
cv2.imshow('Image blur 7', img_blur7)
cv2.imshow('Bilateral blur', bilateral_blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

i)

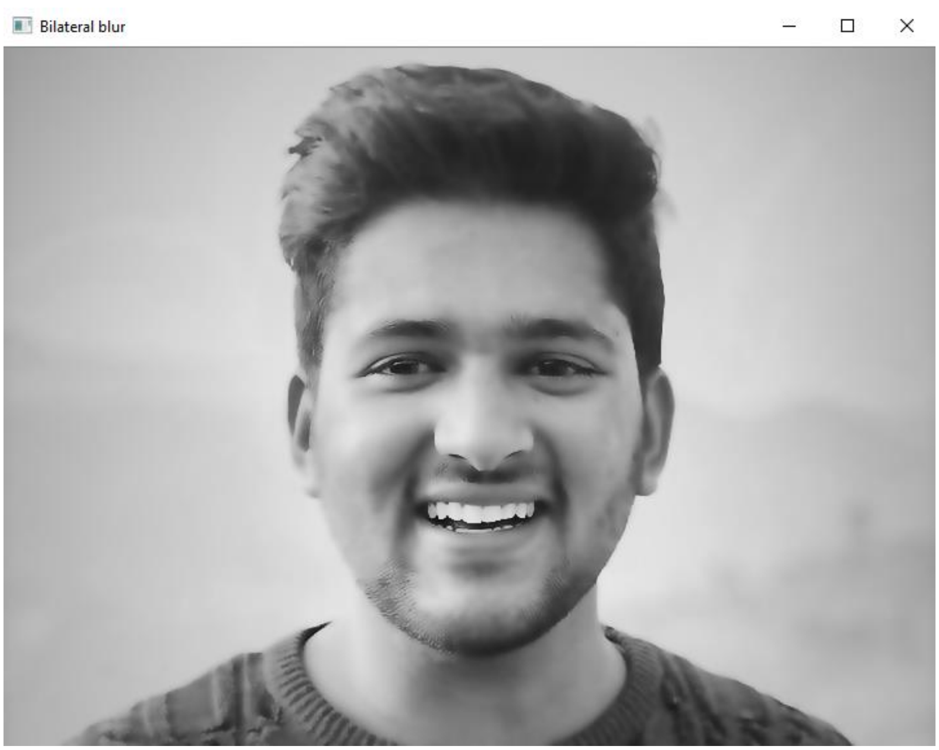


ii)



iii)

iv)

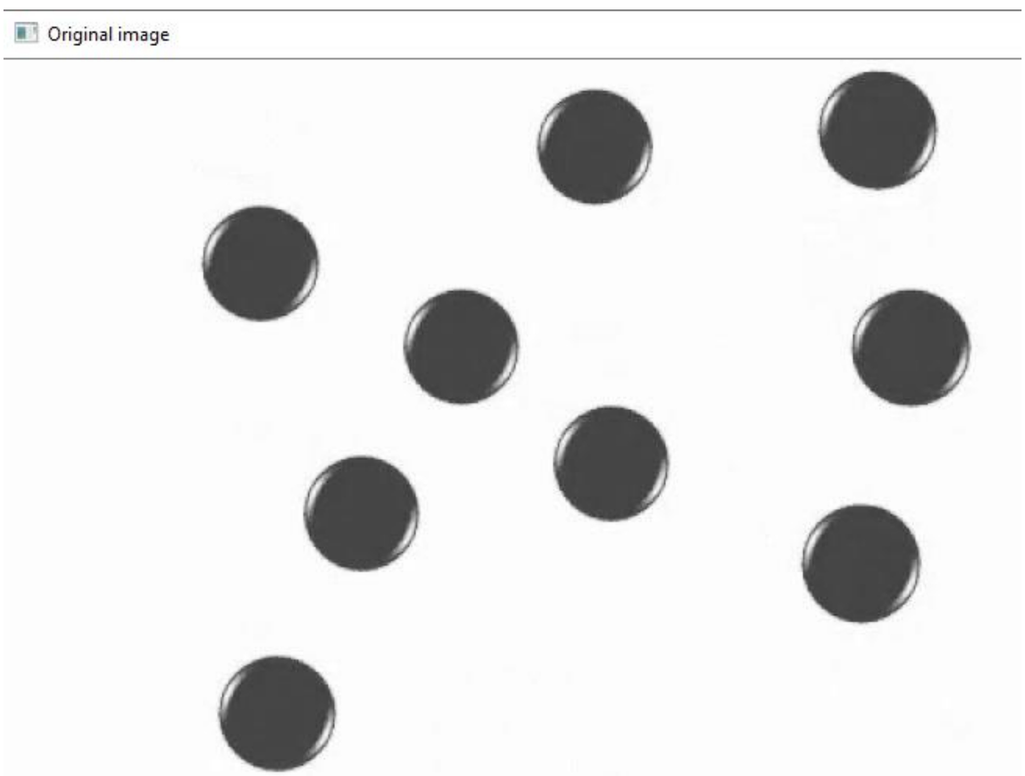


Program 3: Opening and Closing of image

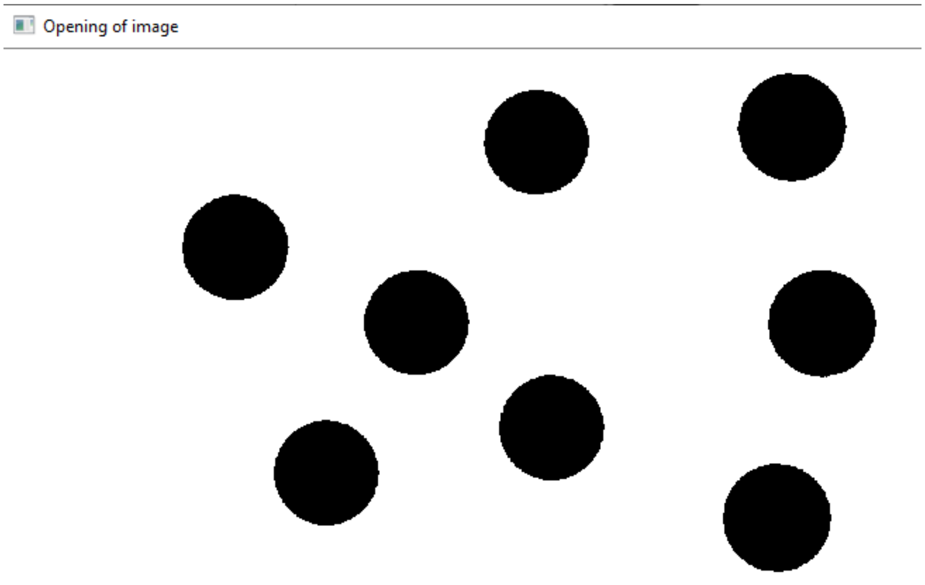
```
import cv2
img = cv2.imread('dot.jpg', 0)
# masking
_, mask = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY_INV)
_, mask1 = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY)
# opening: erosion followed by dilation
img_opening = cv2.morphologyEx(mask1, cv2.MORPH_OPEN, (3, 3), iterations = 10)
# closing: dilation followed by erosion
img_closing = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, (3, 3), iterations = 10)
cv2.imshow('Original image', img)
cv2.imshow('Opening of image', img_opening)
cv2.imshow('Closing of image', img_closing)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

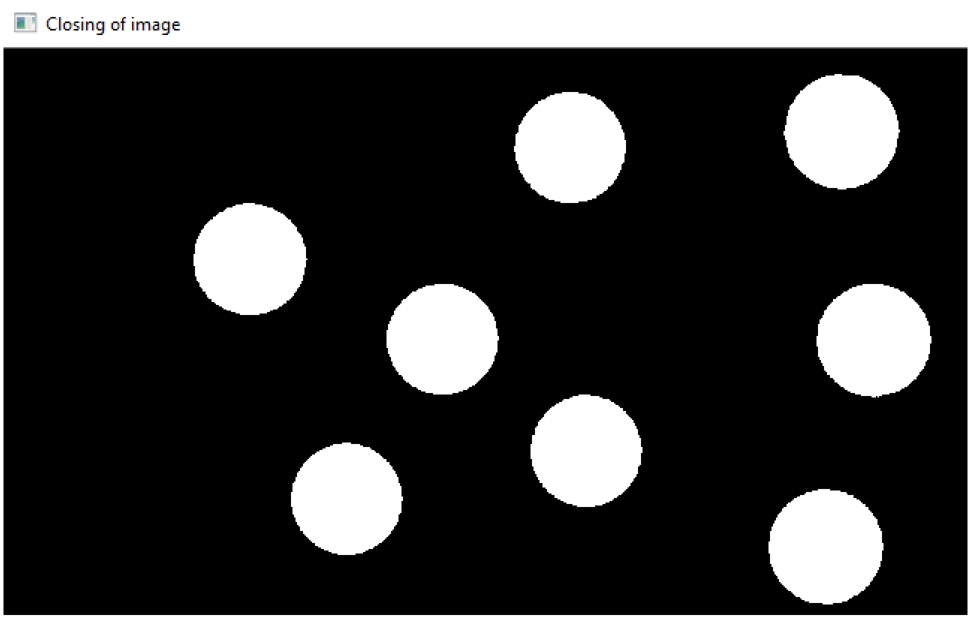
i)



ii)



iii)



CONCLUSION

Hence, we are able to implement some smoothing filters and to implement opening and closing of an image.

LAB - 03

OBJECTIVES

To implement high pass and low pass filter in spatial domain

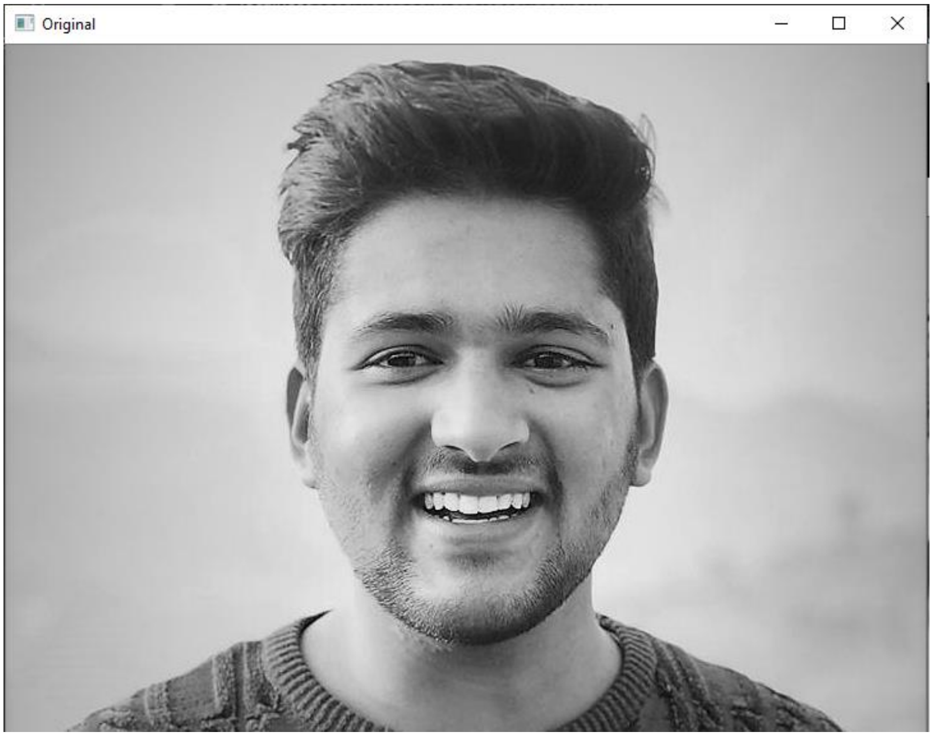
PROGRAMS

Program: High Pass and Low Pass Filter

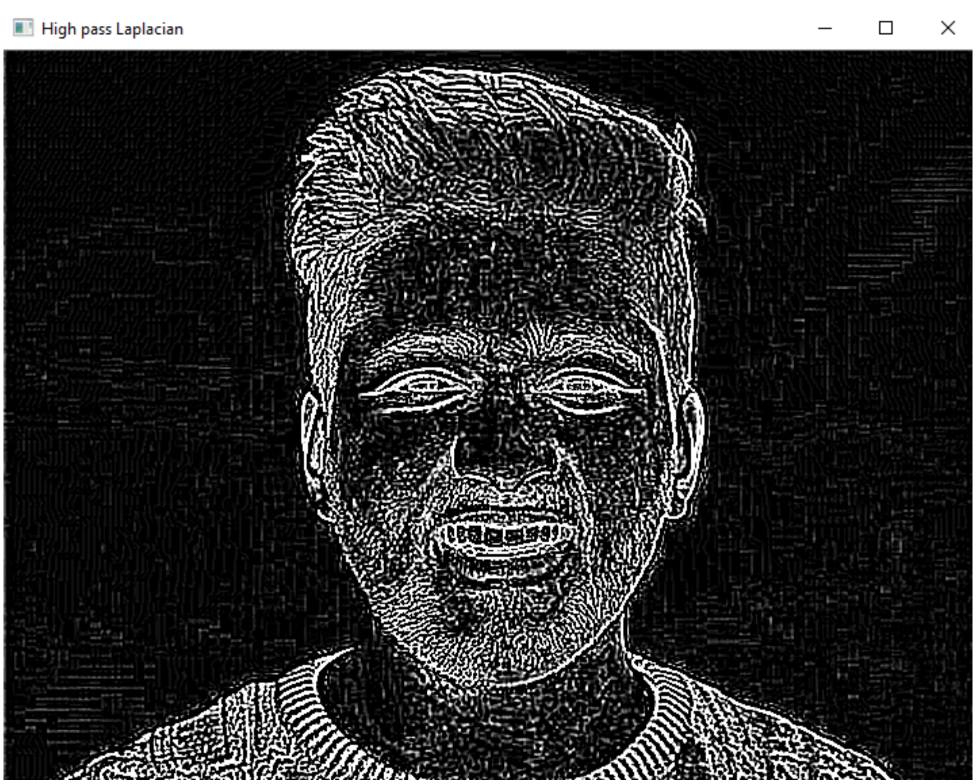
```
import cv2
import numpy as np

img = cv2.imread('rn.jpg', 0)
# High pass filter
high_pass_lap = cv2.Laplacian(img, ksize = 3, ddepth = -1)
# Add noise
mean = 5
var = 100
sigma = var ** 0.5
gaussian = np.random.normal(mean, sigma) #  np.zeros((224, 224), np.float32)

noisy_image = np.zeros(img.shape, np.float32)

if len(img.shape) == 2:
    noisy_image = img + gaussian
else:
    noisy_image[:, :, 0] = img[:, :, 0] + gaussian
    noisy_image[:, :, 1] = img[:, :, 1] + gaussian
    noisy_image[:, :, 2] = img[:, :, 2] + gaussian

cv2.normalize(noisy_image, noisy_image, 0, 255, cv2.NORM_MINMAX, dtype=-1)
noisy_image = noisy_image.astype(np.uint8)

# Low pass filter
low_pass = cv2.GaussianBlur(noisy_image, (3, 3), 7)
titles = ['Original', 'High pass Laplacian', 'Low pass filter']
images = [img, high_pass_lap, low_pass]
for i in range(len(titles)):
    cv2.imshow(titles[i], images[i])
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

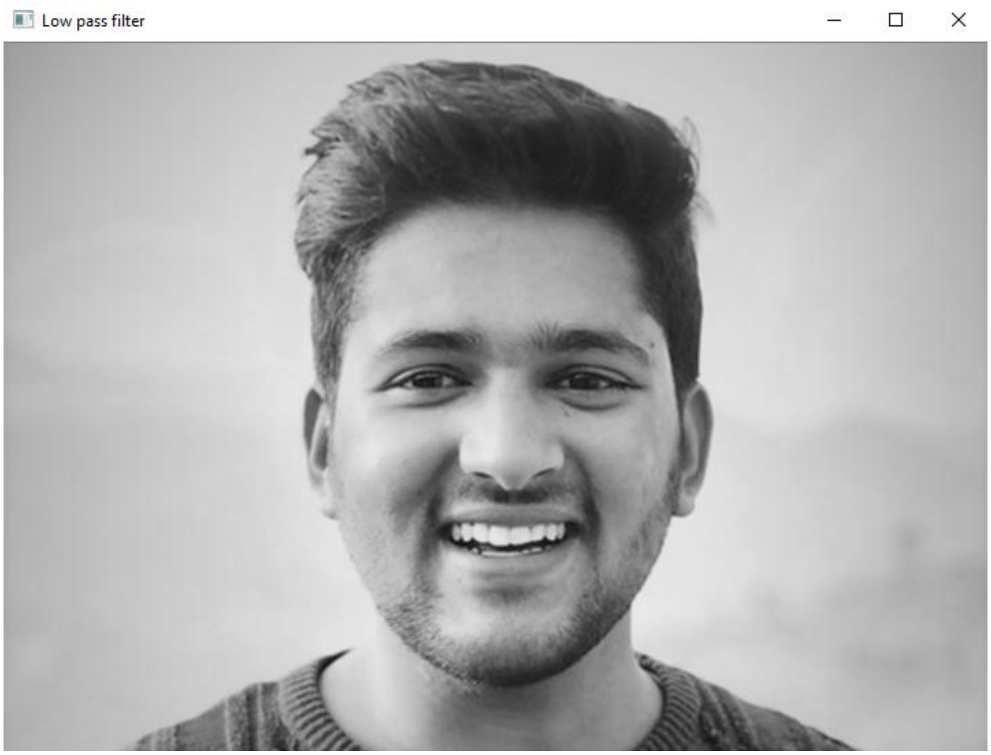
i)



ii)



iii)



CONCLUSION

Hence, we are able to implement high pass and low pass filter in spatial domain.

LAB-04

OBJECTIVES

To implement bit plane slicing and fourier transform function

PROGRAMS

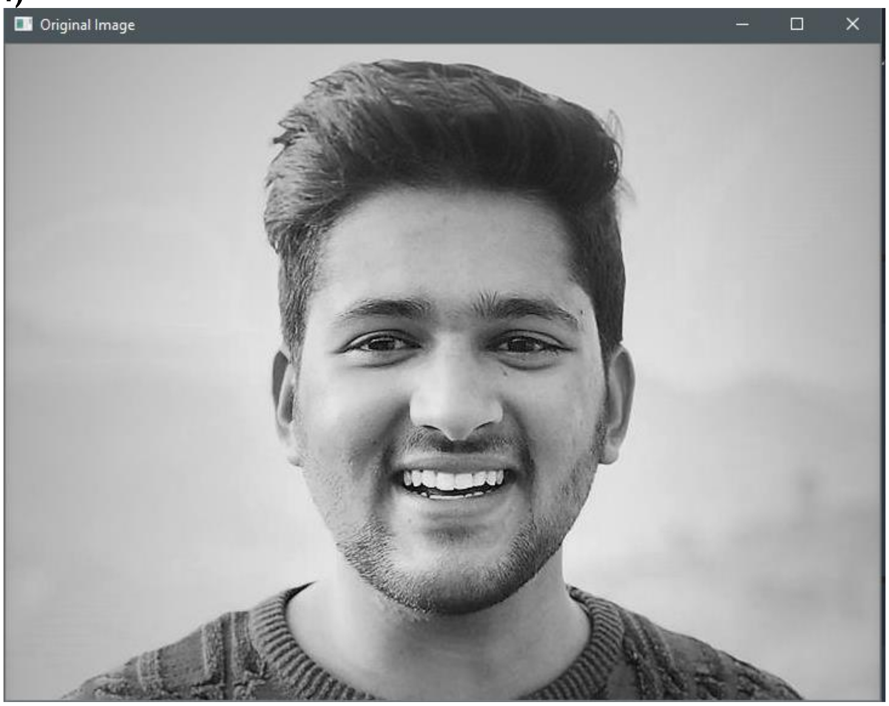
Program 01: Bit Plane Slicing

```
import numpy as np
import cv2 as cv

# Read image in grayscale
img = cv.imread('rn.jpg', 0)

# Change each pixel value to binary
img_bin  = []
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        img_bin.append(np.binary_repr(img[i][j], width = 8))

# Slice into bits
eighth = (np.array([int(i[0]) for i in img_bin], dtype = np.uint8) * 128) \
                                .reshape(img.shape[0], img.shape[1])
seventh = (np.array([int(i[1]) for i in img_bin], dtype = np.uint8) * 64) \
                                .reshape(img.shape[0], img.shape[1])
sixth = (np.array([int(i[2]) for i in img_bin], dtype = np.uint8) * 32) \
                                .reshape(img.shape[0], img.shape[1])
fifth = (np.array([int(i[3]) for i in img_bin], dtype = np.uint8) * 16) \
                                .reshape(img.shape[0], img.shape[1])
fourth = (np.array([int(i[4]) for i in img_bin], dtype = np.uint8) * 8) \
                                .reshape(img.shape[0], img.shape[1])
third = (np.array([int(i[5]) for i in img_bin], dtype = np.uint8) * 4) \
                                .reshape(img.shape[0], img.shape[1])
second = (np.array([int(i[6]) for i in img_bin], dtype = np.uint8) * 2) \
                                .reshape(img.shape[0], img.shape[1])
first = (np.array([int(i[7]) for i in img_bin], dtype = np.uint8) * 1) \
                                .reshape(img.shape[0], img.shape[1])

titles = ['Original Image', 'Eighth Plane', 'Seventh Plane', 'Sixth Plane', 'Fifth Plane',\
          'Fourth Plane', 'Third Plane', 'Second Plane', 'First Plane']

sliced_planes = [img, eighth, seventh, sixth, fifth, fourth, third, second, first]
```

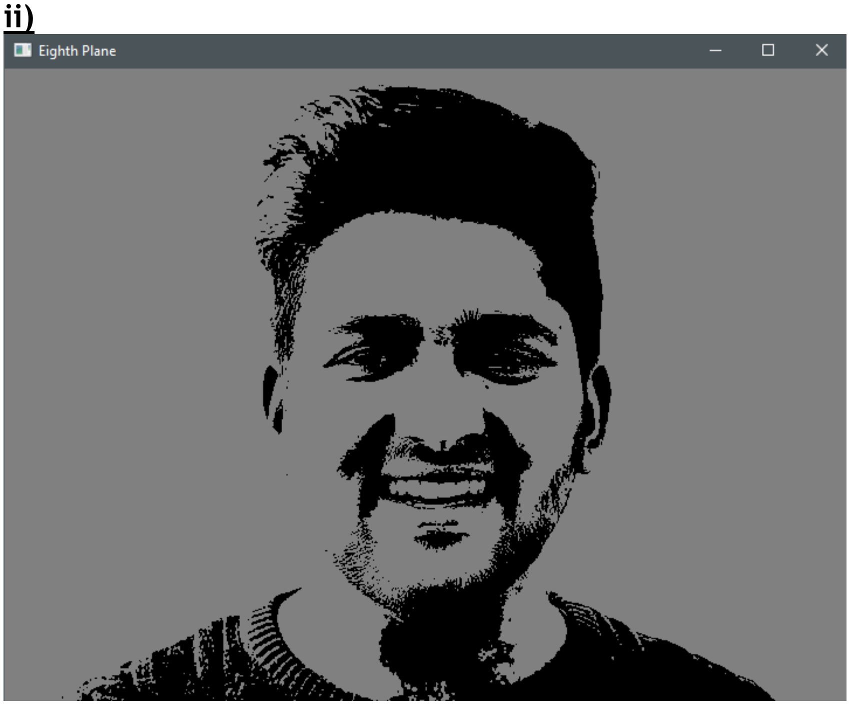
```
for i in range(8):
   cv.imshow(titles[i], sliced_planes[i])

cv.waitKey(0)
cv.destroyAllWindows()
```

Output:

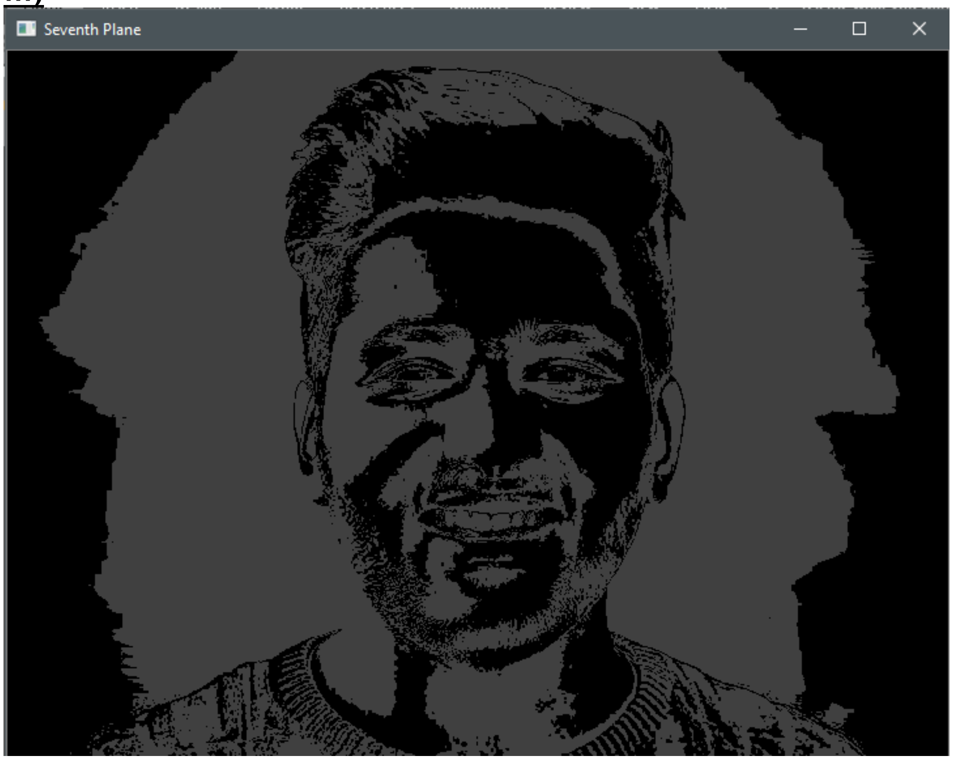
i)



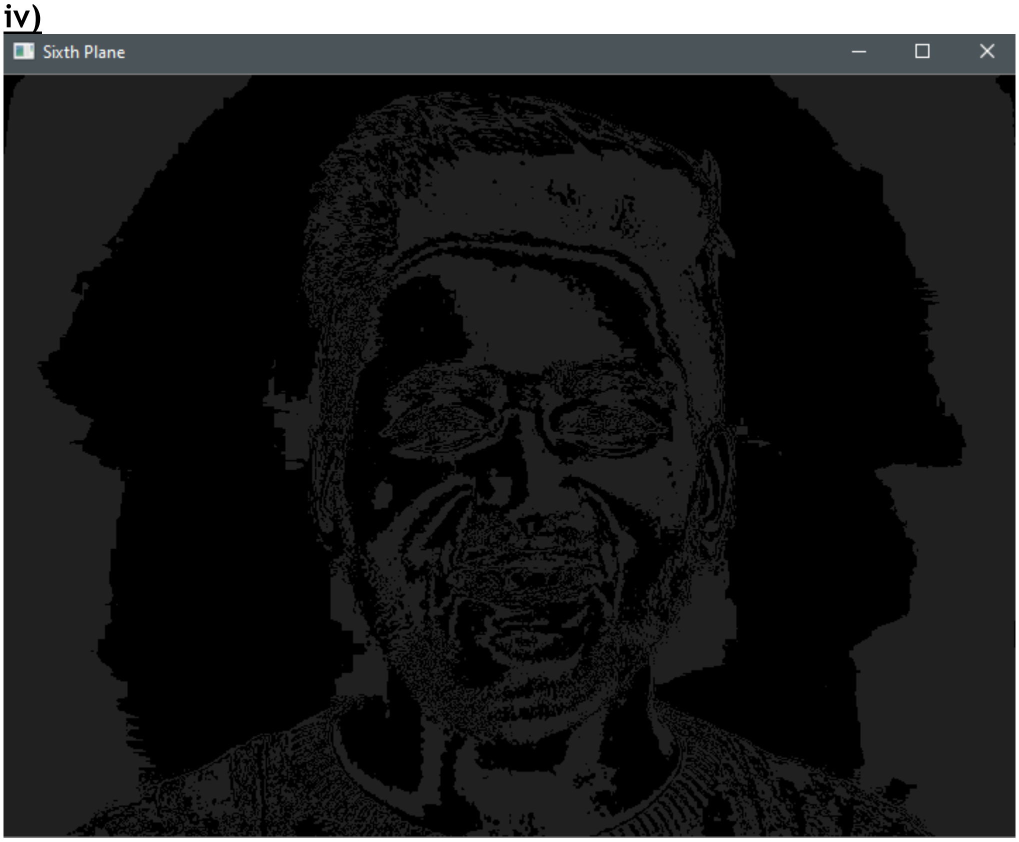
ii)



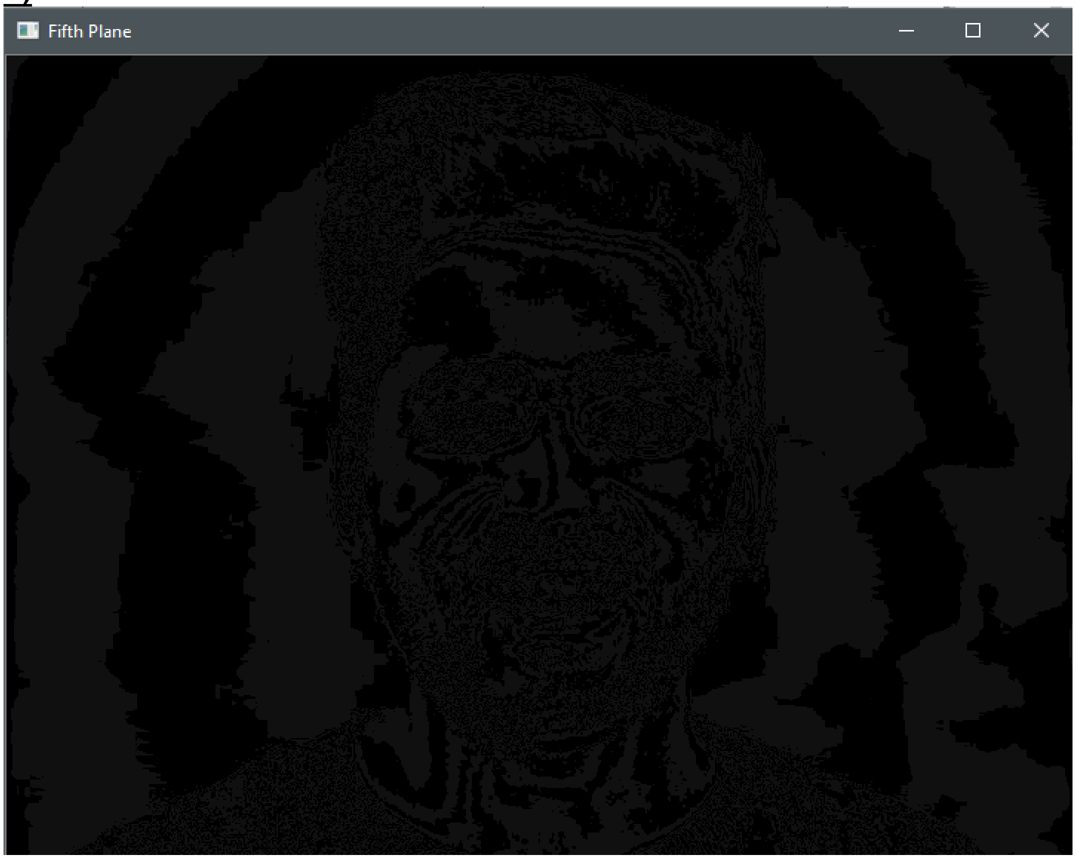
iii)



iv)



v)

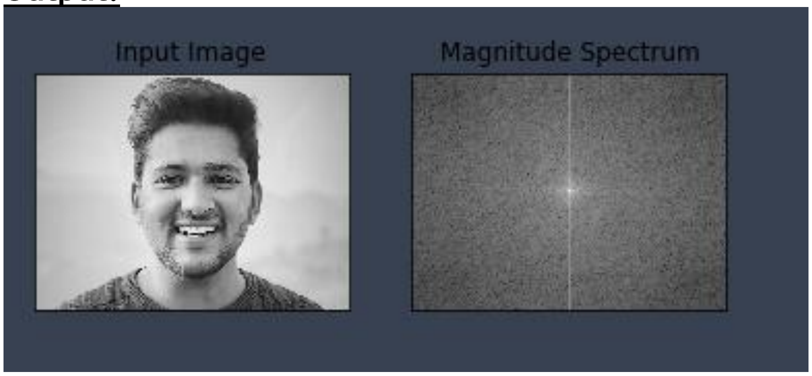


Program 02: Fourier Transform

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('rn.jpg', 0)

img_float32 = np.float32(img)

dft = cv2.dft(img_float32, flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
```

```
plt.show()
```

Output:



Program 03: Applying Low Pass Filter with Fourier Transform

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread('rn.jpg', 0)
img_dft = cv.dft(np.float32(img), flags = cv.DFT_COMPLEX_OUTPUT)
img_dft_shift = np.fft.fftshift(img_dft)
magnitude_spectrum = 20 * np.log(cv.magnitude(img_dft_shift[:,:,0],
img_dft_shift[:,:,1]))

rows, cols = img.shape
crow, ccol = rows // 2, cols // 2

# Create a mask first, center square 1, remaining all zeros
mask = np.zeros((rows, cols, 2), np.uint8)
mask[crow - 30 : crow + 30, ccol - 30 : ccol + 30] = 1

# Apply mask and inverse DFT
fshift = img_dft_shift * mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv.idft(f_ishift)
img_back = cv.magnitude(img_back[:,:,0], img_back[:,:,1])

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Applying LPF'), plt.xticks([]), plt.yticks([])

plt.show()
```

Output:

CONCLUSION

Hence, we are able to implement bit plane slicing and fourier transform function.

LAB - 05

OBJECTIVES

To implement Laplacian filter and mean filters

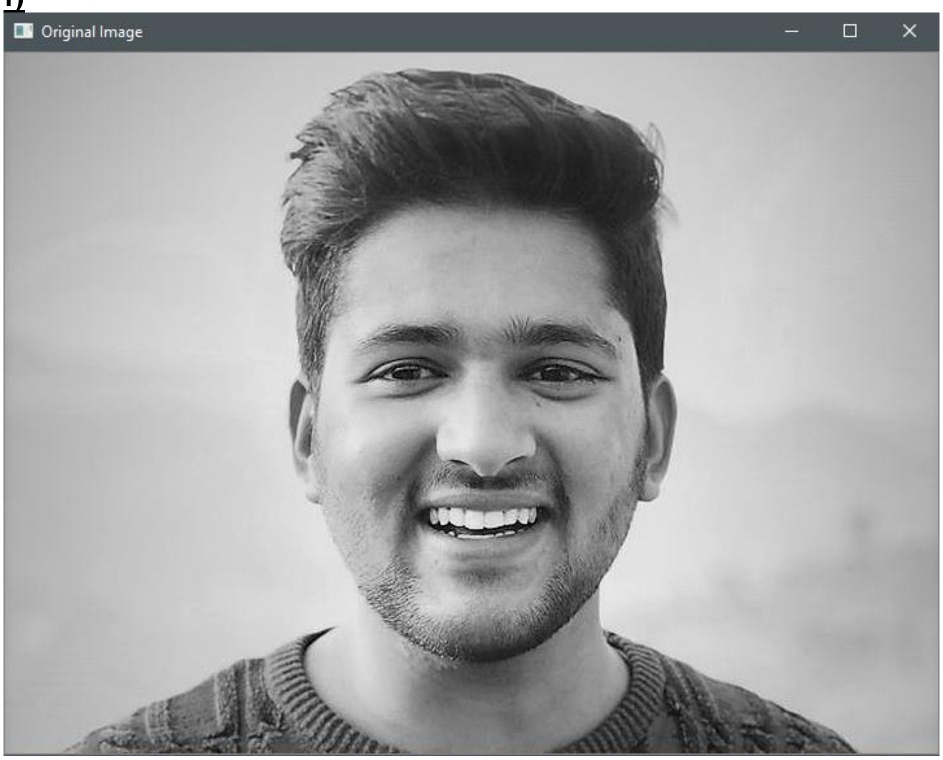
PROGRAMS

Program 01: Laplacian Filter

```
import cv2 as cv
import numpy as np

img = cv.imread('rn.jpg', 0)
blur_img = cv.GaussianBlur(img, (7, 7), 0)
laplacian_img = cv.Laplacian(blur_img, cv.CV_32F)
lap = laplacian_img / laplacian_img.max()

cv.imshow('Original Image', img)
# cv.imshow('gaussian', blur_img)
cv.imshow('Applying Laplacian', laplacian_img)
cv.imshow('Final Image', lap)
cv.waitKey(0)
cv.destroyAllWindows()
```

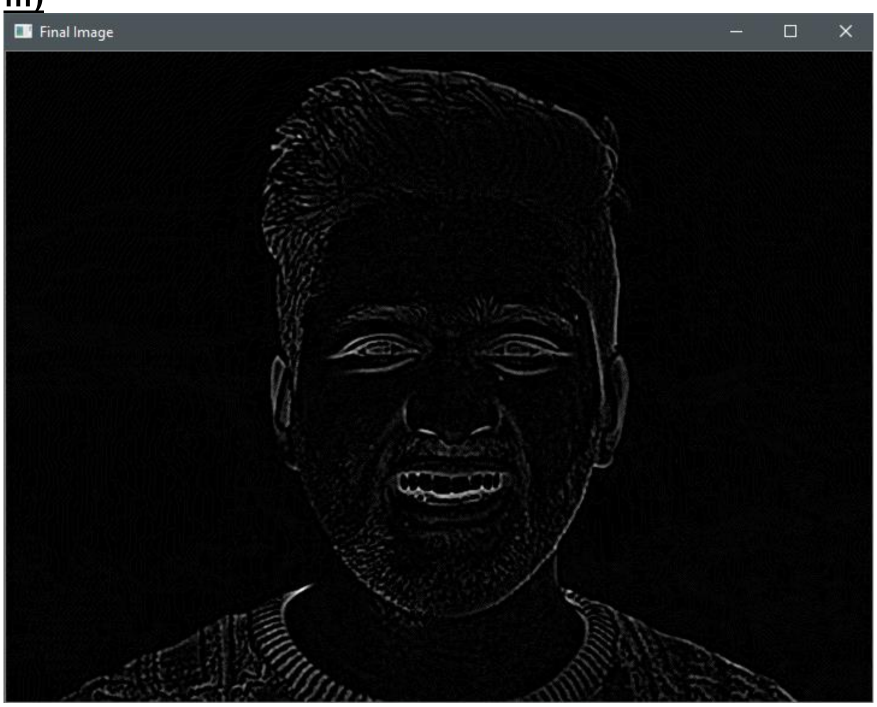
Output:

i)



ii)

iii)

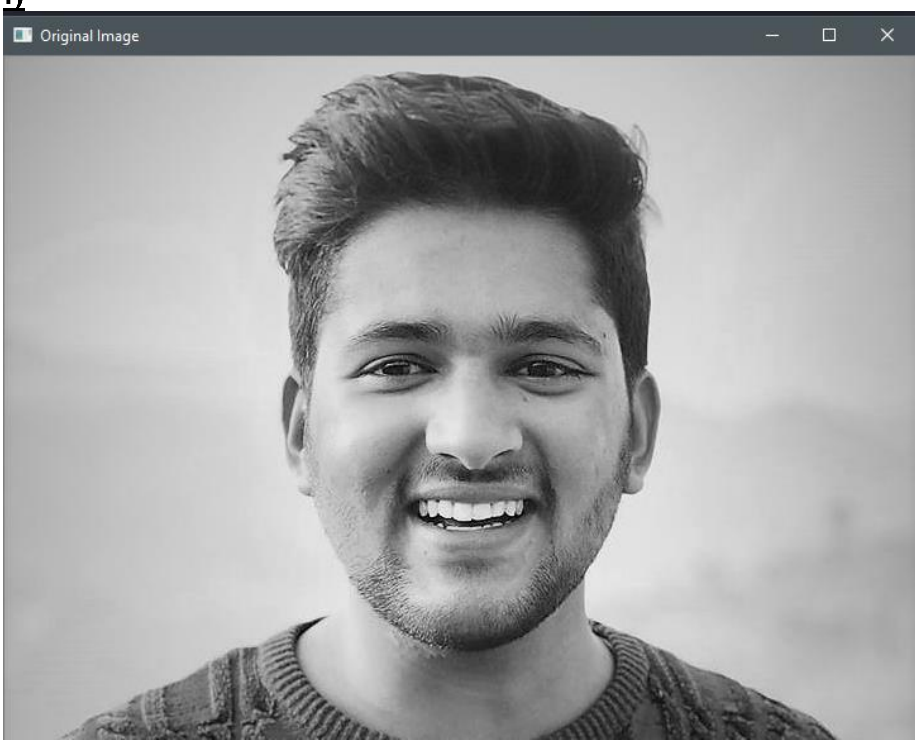


Program 02: Mean filter

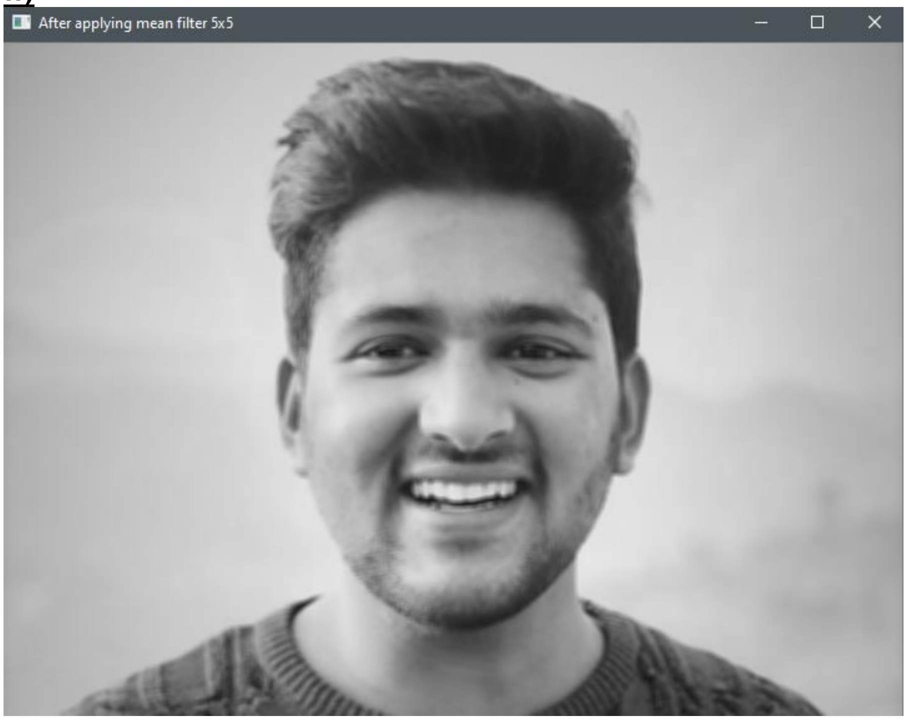
```
import cv2 as cv
import numpy as np

img = cv.imread('rn.jpg', 0)
kernel = np.ones((5, 5), np.float32) / 25
averaging_img = cv.filter2D(img, -1, kernel)

cv.imshow('Original Image', img)
cv.imshow('After applying mean filter 5x5', averaging_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Output:

i)



ii)



CONCLUSION

Hence, we are able to implement Laplacian filter and mean filters.

LAB - 06

OBJECTIVES

To implement edge detection and to implement erosion and dilation operations

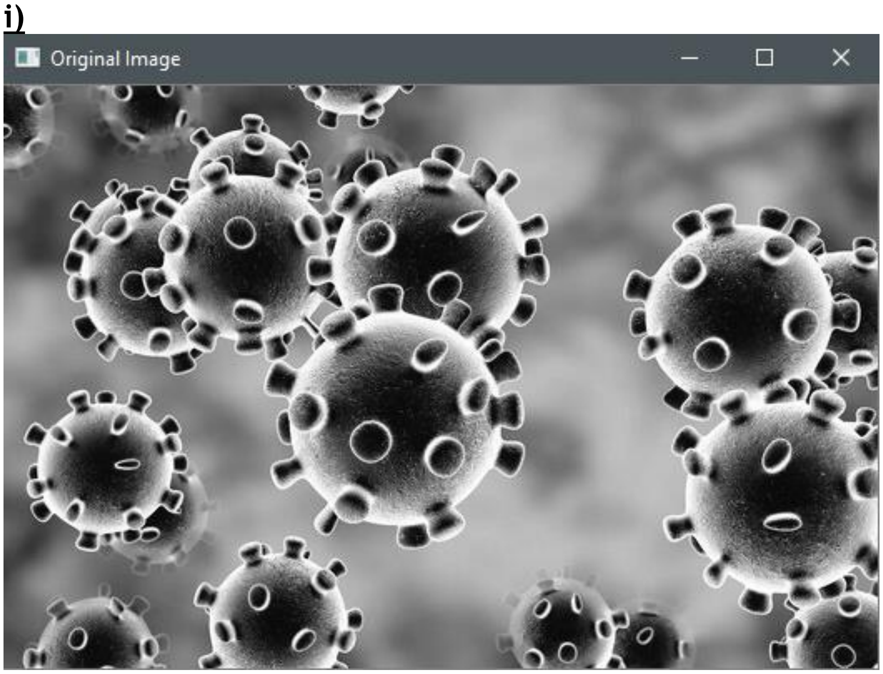
PROGRAMS

Program 01: Edge Detection

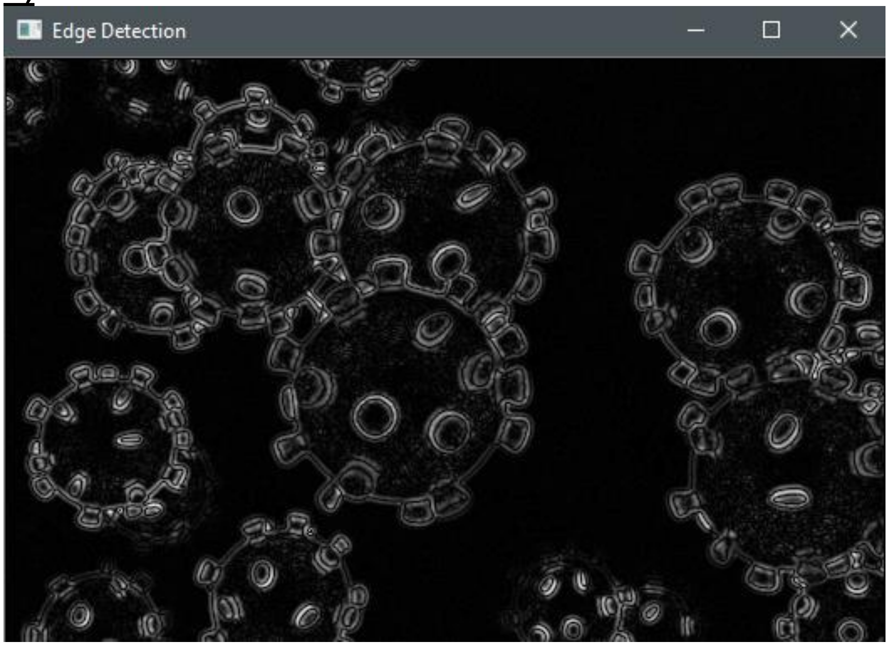
```
import cv2 as cv
import numpy as np

img = cv.imread('Coronavirus.jpg', 0)
# print(img)
blur_img = cv.GaussianBlur(img, (5, 5), 0)
img_f = np.float32(blur_img)

rows, cols = img.shape
kernel = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])
# kernel = np.array([[0, -1, 0], [-1, 4, -1], [0, -1, 0]])
final_img = np.zeros_like(img)

for i in range(1, rows - 1):
    for j in range(1, cols - 1):
        final_img[i][j] = abs(sum((img_f[i - 1 : i + 2, j - 1 : j + 2] * kernel).flatten()))
final_img = np.uint8(final_img)
cv.imshow('Original Image', img)
cv.imshow('Edge Detection', final_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Output:

i)



ii)



Program 02: Erosion and Dilation Operation

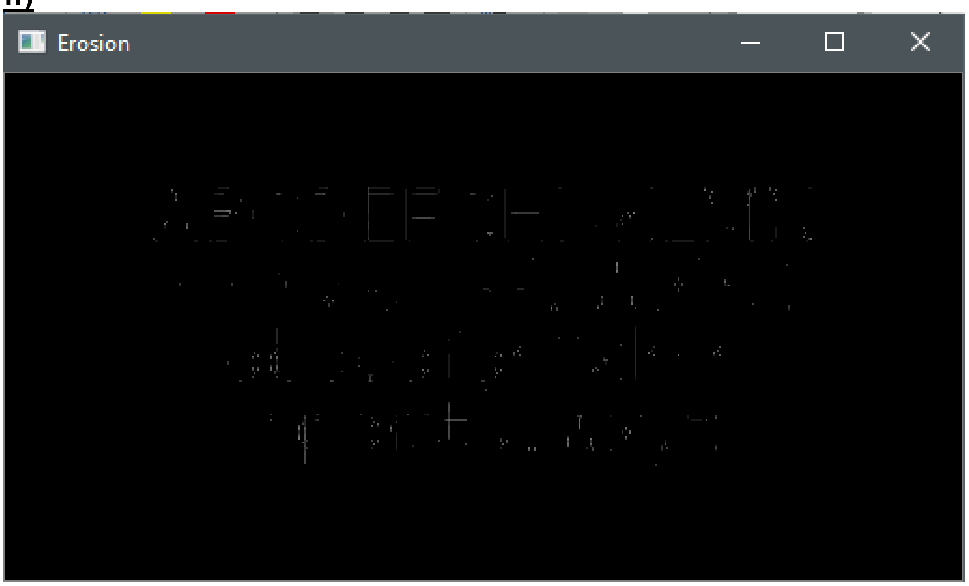
```
import cv2 as cv
import numpy as np

img = cv.imread('letter.png', 0)
# blur_img = cv.GaussianBlur(img, (3, 3), 0)
_, thres_bin = cv.threshold(img, 127, 255, cv.THRESH_BINARY)
kernel = np.ones((2, 2), np.uint8)
erosion_img = cv.erode(img, kernel, iterations = 1)
dilation_img = cv.dilate(img, kernel, iterations = 1)

cv.imshow('Original Image', img)
cv.imshow('Erosion', erosion_img)
cv.imshow('Dilation', dilation_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Output:

i)


Original Image
ABCDEFGHIJKLMN
OPQRSTUVWXYZ
abcedfghijklmn
opqrstuvwxyz

ii)


Erosion

iii)

CONCLUSION

Hence we are able to implement edge detection algorithm and to implement erosion and dilation operations.

LAB – 07

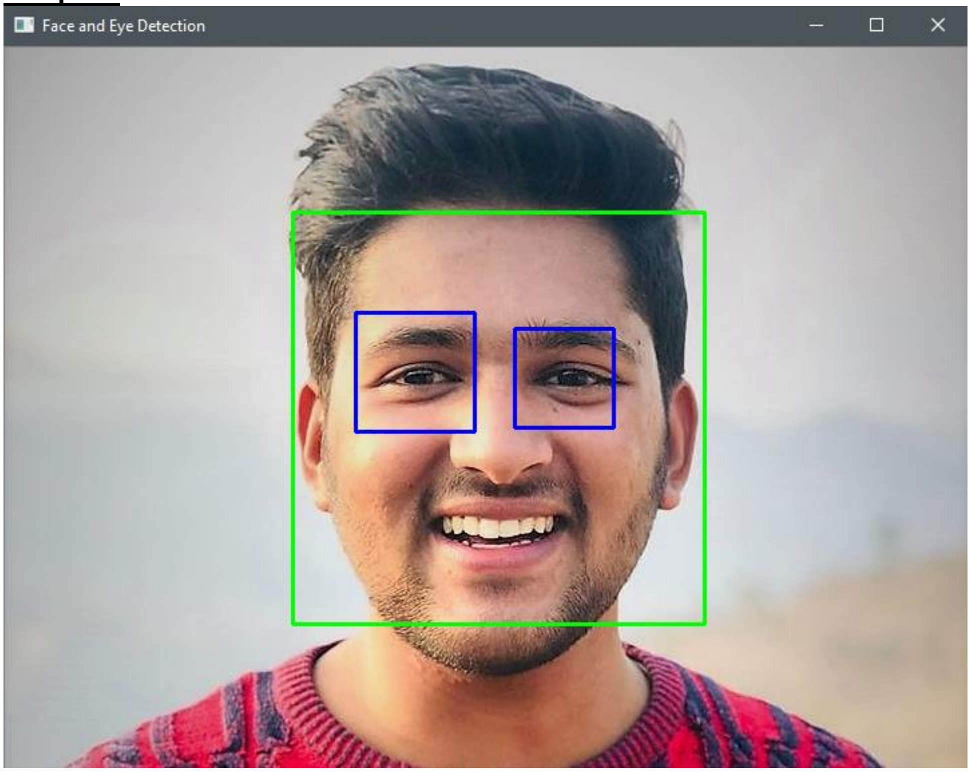
OBJECTIVES

To implement pattern recognition (face and eye detection)

PROGRAMS

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

def face_detection(img):
    faces = face_cascade.detectMultiScale(img, scaleFactor = 1.2, minNeighbors = 3,
                                minSize = (30, 30))
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    return img

def eye_detection(img):
    eyes = eye_cascade.detectMultiScale(img, scaleFactor = 1.35, minNeighbors = 3)
    for (x, y, w, h) in eyes:
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
    return img

img = cv2.imread('rn.jpg')
face = face_detection(img)
eye = eye_detection(img)

cv2.imshow('Face and Eye Detection', face)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:



CONCLUSION

Hence, we are able to implement pattern recognition (face and eye detection).

LAB - 08

OBJECTIVES

To implement image thinning

PROGRAMS

```
import cv2
import numpy as np

# Create an image with text on it
img = np.zeros((200,800),dtype='uint8')
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'RameshNeupane',(5,70), font, 2,(255),5,cv2.LINE_AA)
img1 = img.copy()

# Structuring Element
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS,(3,3))
# Create an empty output image to hold values
thin = np.zeros(img.shape,dtype='uint8')

# Loop until erosion leads to an empty set
while (cv2.countNonZero(img1)!=0):
    # Erosion
    erode = cv2.erode(img1,kernel)
    # Opening on eroded image
    opening = cv2.morphologyEx(erode,cv2.MORPH_OPEN,kernel)
    # Subtract these two
    subset = erode - opening
    # Union of all previous sets
    thin = cv2.bitwise_or(subset,thin)
    # Set the eroded image for next iteration
    img1 = erode.copy()

cv2.imshow('Original', img)
cv2.imshow('Thinned', thin)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:
i)

ii)

CONCLUSION
Hence, we are able to implement image thinning.