Department Of Computer Science

Gujarat University

Certificate

Roll No: 15 Seat No: ____

This is to certify that Mr. / Ms. _LUHAR PRINCE KUMAR GHISULAL_ student of MCA Semester – III, has duly completed his/her term work for the semester ending in December 2021, in the subject of _COMPUTER VISION _ towards partial fulfillment of his / her Degree of Masters in Computer Science & Application.

Date of Submission
14-DEC-2021

Internal Faculty
DR.SUCHIT PUROHIT

Head of Department
DR.JYOTI PAREKH

School Of Computer Science
Gujarat University

MCA – III

Subject: - __COMPUTER VISION_____

Name: - __LUHAR PRINCE GHISULAL_____

Roll No.: - _15___ **Exam Seat No.: -_____**

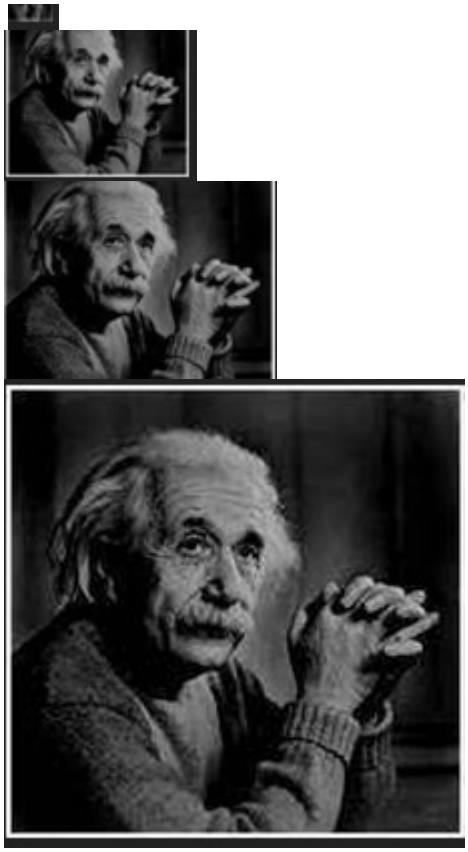
S C H O O L O F C O M P U T E R S C I E N C E
G U J A R A T U N I V E R S I T Y
M.C.A. – III

R O L L NO : 15
N A M E : LUHAR PRINCE KUMAR GHISULAL
S U B J E C T : COMPUTER VISION

Assigment - 1

1. write a program to read image from PiL and open cv library and perform the effects of changing spatial and gray level resolution in it

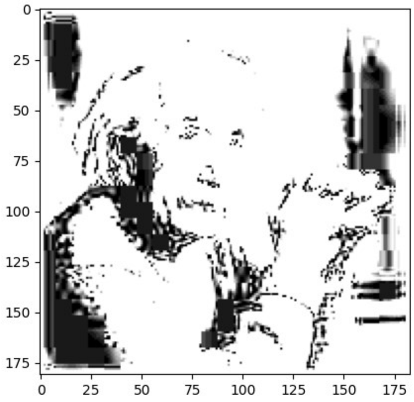
```
from PIL import Image

img = Image.open("grey.jpg")

width, height = img.size

print(width, height)

a=img.resize((int(0.1*width),int(0.1*height)));

b=img.resize((int(0.6*width),int(0.6*height)));

c=img.resize((int(0.4*width),int(0.4*height)));

img.show(img)

a.show()

b.show()

c.show()
```

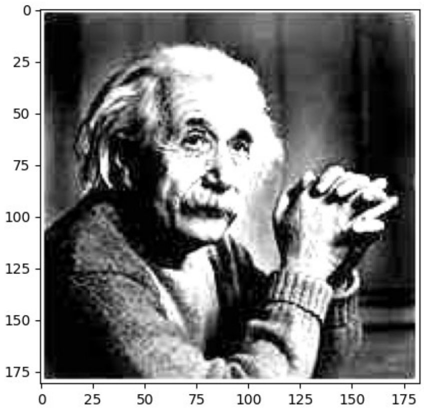


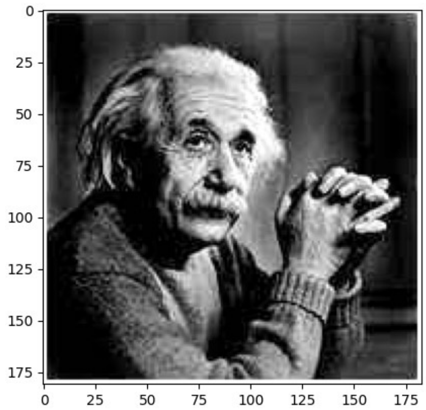
Grey Level Change

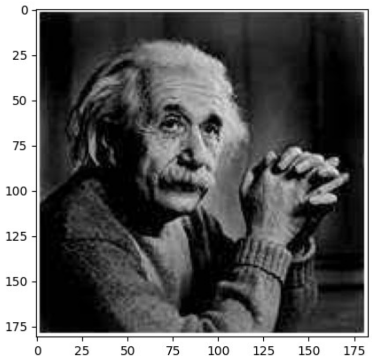
```
import matplotlib.pyplot as plt
img = plt.imread("grey.jpg")
for i in range (10,255,50):
    img2 = img / i
    plt.imshow(img2)
    plt.show()
```

Output:









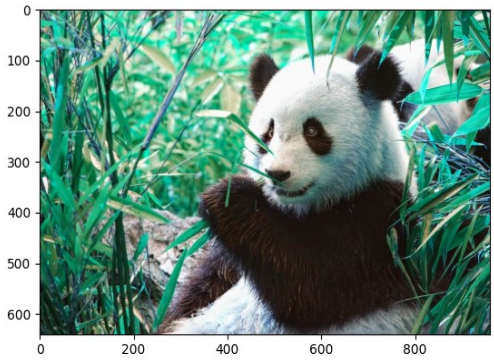
Assignment - 2

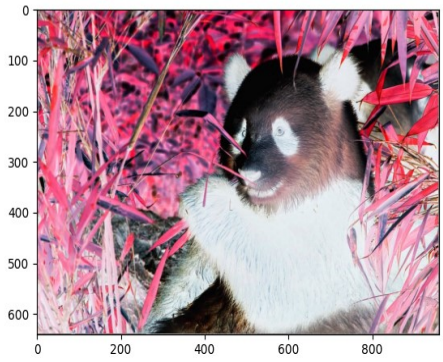
1.Write python programs (without using any methods) for Image Negation, power law and log transformation on image . Show output with different parameter settings

Negative transformation

```
import cv2

import matplotlib.pyplot as plt


img = cv2.imread('panda.jpg')

color = ('b','g','r')


height, width, _ = img.shape

plt.imshow(img)

plt.show()


for i in range(0, height - 1):

    for j in range(0, width - 1):

        pixel = img[i, j]

        pixel[0] = 255 - pixel[0]

        pixel[1] = 255 - pixel[1]

        pixel[2] = 255 - pixel[2]

        img[i, j] = pixel

plt.imshow(img)

plt.show()
```

OUTPUT



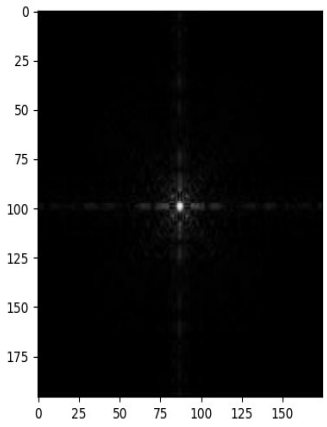


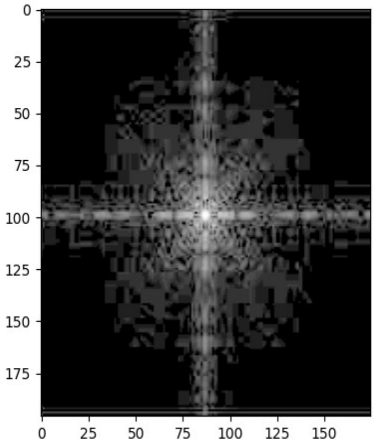
2.Log Transformation

```
import cv2

import numpy as np

import matplotlib.pyplot as plt


# Read an image

image = cv2.imread('log_image.png')


# Apply log transformation method

c = 255 / np.log(1 + np.max(image))

log_image = c * (np.log(image + 1))


# Specify the data type so that

# float value will be converted to int

log_image = np.array(log_image, dtype = np.uint8)


# Display both images
```

```
plt.imshow(image)
plt.show()
plt.imshow(log_image)
plt.show()
```

OUTPUT



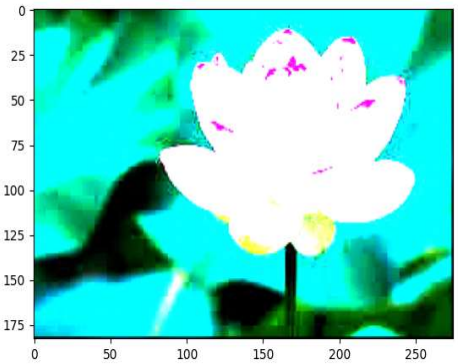


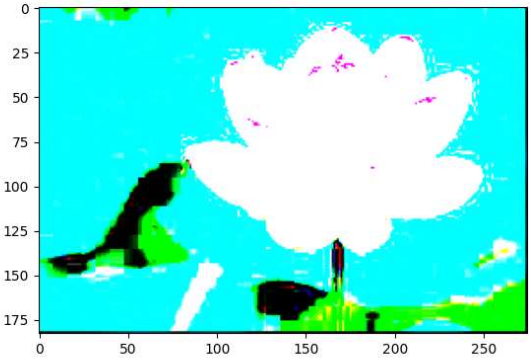
3.Power Low Transformation

```
import math
from PIL import Image
from PIL import ImageFilter
import matplotlib.pyplot as plt
# Load the image
img = Image.open("lotus.jpg");
plt.imshow(img)
plt.show()
# Read pixels and apply negative transformation
```

```
gamma=1;
while(gamma<=2):
    for i in range(0, img.size[0] - 1):
        for j in range(0, img.size[1] - 1):
            pixelColorVals = img.getpixel((i, j));
            redPixel = math.floor(pixelColorVals[0] ** gamma);  # Negate red pixel
            greenPixel = math.floor(pixelColorVals[1] ** gamma);  # Negate green
pixel
            bluePixel = math.floor(pixelColorVals[2] ** gamma);  # Negate blue
pixel
            img.putpixel((i, j), (redPixel, greenPixel, bluePixel));
    plt.imshow(img)
    plt.show()
    gamma=gamma+0.5
    print(gamma)
```

OUTPUT


0
25
50
75
100
125
150
175
0
50
100
150
200
250


0
25
50
75
100
125
150
175
0
50
100
150
200
250

Assignment - 3

1.Write program for min-max contrast stretching

```
import cv2

import numpy as np

# Read the image

img1 = cv2.imread('cs.PNG',0)

# Create zeros array to store the stretched image

minmax_img = np.zeros((img1.shape[0],img1.shape[1]),dtype = 'uint8')

# Loop over the image and apply Min-Max formulae

for i in range(img1.shape[0]):

    for j in range(img1.shape[1]):

        minmax_img[i,j] = 255*(img1[i,j]-np.min(img1))/(np.max(img1)-
np.min(img1))

# Displat the stretched image

cv2.imshow('Minmax',minmax_img)

cv2.waitKey(0)
```

OUTPUT

Class Work

1.Bit Plane Slicing

```
import cv2

import numpy as np

# Read the image in greyscale

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


# Iterate over each pixel and change pixel value to binary using np.binary_repr() and store it in a list.

lst = []

for i in range(img.shape[0]):

    for j in range(img.shape[1]):

        lst.append(np.binary_repr(img[i][j], width=8))  # width = no. of bits


# We have a list of strings where each string represents binary pixel value. To extract bit planes we need to iterate over the strings and store the characters corresponding to bit planes into lists.

# Multiply with 2^(n-1) and reshape to reconstruct the bit image.

eight_bit_img = (np.array([int(i[0]) for i in lst], dtype=np.uint8) * 128).reshape(img.shape[0], img.shape[1])

seven_bit_img = (np.array([int(i[1]) for i in lst], dtype=np.uint8) * 64).reshape(img.shape[0], img.shape[1])

six_bit_img = (np.array([int(i[2]) for i in lst], dtype=np.uint8) * 32).reshape(img.shape[0], img.shape[1])

five_bit_img = (np.array([int(i[3]) for i in lst], dtype=np.uint8) * 16).reshape(img.shape[0], img.shape[1])
```

```
four_bit_img = (np.array([int(i[4]) for i in lst], dtype=np.uint8) *
8).reshape(img.shape[0], img.shape[1])

three_bit_img = (np.array([int(i[5]) for i in lst], dtype=np.uint8) *
4).reshape(img.shape[0], img.shape[1])

two_bit_img = (np.array([int(i[6]) for i in lst], dtype=np.uint8) *
2).reshape(img.shape[0], img.shape[1])

one_bit_img = (np.array([int(i[7]) for i in lst], dtype=np.uint8) *
1).reshape(img.shape[0], img.shape[1])


# Concatenate these images for ease of display using cv2.hconcat()

finalr = cv2.hconcat([eight_bit_img, seven_bit_img, six_bit_img,
five_bit_img])

finalv = cv2.hconcat([four_bit_img, three_bit_img, two_bit_img, one_bit_img])


# Vertically concatenate

final = cv2.vconcat([finalr, finalv])


# Display the images

cv2.imshow('a', final)

cv2.waitKey(0)
```

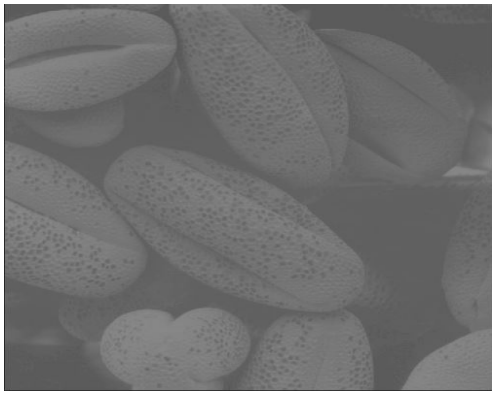
OUTPUT

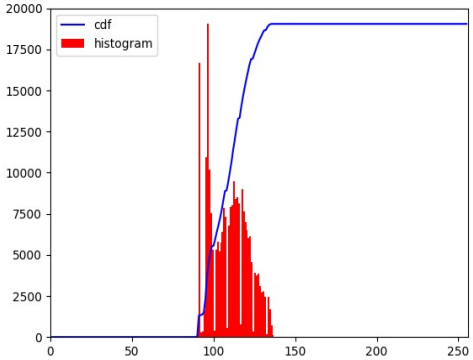
2.Histogram Equalization

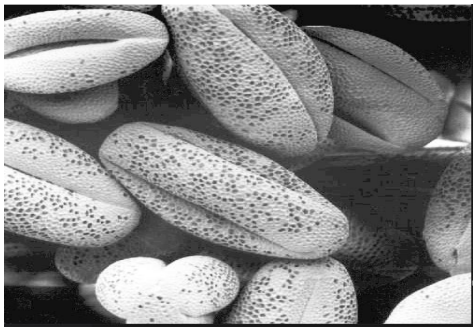
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread("image.jpg",0)
cv.imshow('image',img)
cv.waitKey(0)
cv.destroyAllWindows()

hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
```

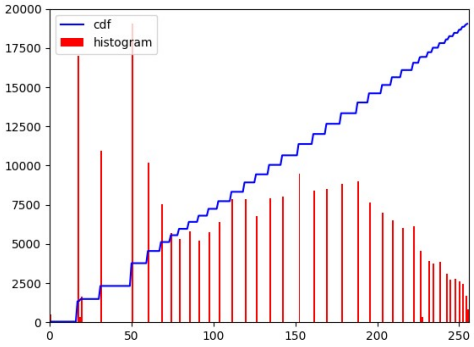
```
plt.show()



equ= cv.equalizeHist(img)



cv.imshow('equ.png',equ)
cv.waitKey(0)
cv.destroyAllWindows()



hist,bins = np.histogram(equ.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(equ.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```

OUTPUT









3.SUM MASKING (Spatial Filtering)

#order static filter (median filtering -> 0th percentile)

non linear filter

import numpy as np

from matplotlib import pyplot as plt

import cv2

plt.rcParams["figure.figsize"] = [7.50, 3.50]

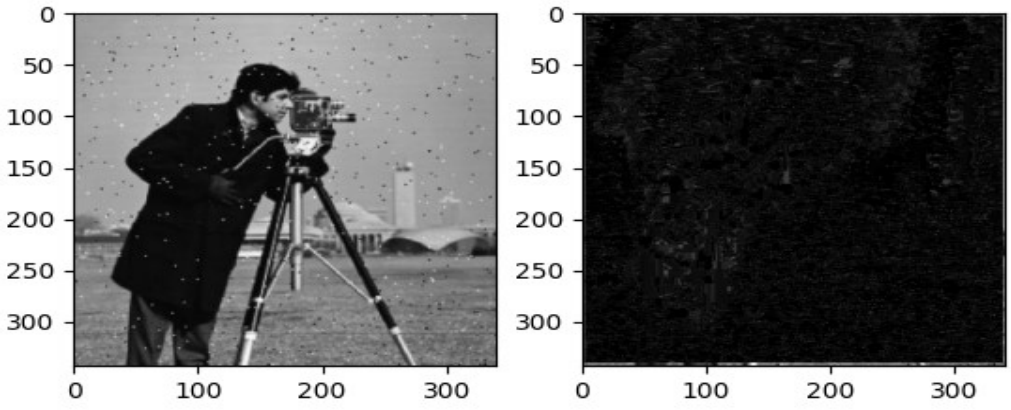
plt.rcParams["figure.autolayout"] = True

```
img = cv2.imread("saltpepper.jpg",1)



height,width,dim = img.shape

# height,width = img.shape
img_new = cv2.imread("saltpepper.jpg",1)



# mask = np.ones((7,7))

mask =
np.array([[35,25,45,25,66],[70,80,255,65,39],[49,45,48,49,49],[72,72,72,72,72],
[68,68,68,68,68]])
height = height - 1
width = width - 1

kernal_size = mask.shape[0]
print(kernal_size)
kernal_size_range = kernal_size // 2

# print(mask)

for i in range(kernal_size_range,height-kernal_size_range):
    for j in range(kernal_size_range,width - kernal_size_range):
```

```
        lst = []
        for k in range(-kernal_size_range ,kernal_size_range + 1):


            for l in range(-kernal_size_range,kernal_size_range + 1):
                if k==0 and l==0 :
                    continue
                else:
                    lst.append(img[i+k][j+l] *
mask[k+kernal_size_range][l+kernal_size_range])
        img_new[i][j] = np.min(lst)


        # print("Min is ",np.min(lst))
        # print("-----------------------------------------------------")
        # print("Using  numpy Median is  = {}".format(np.median(lst)))
        # print("-----------------------------------------------------")
fig = plt.figure()


ax1 = fig.add_subplot(121)
ax1.imshow(img)
ax2 = fig.add_subplot(122)
ax2.imshow(img_new)



# plt.hist(img,256)
```

```
# plt.hist(img_new,256)
plt.show()


# cv2.imshow("Actual Image",image)
# cv2.imshow("Filtered Image",image1)
# cv2.waitKey(0)

# plt.show()
```

OUTPUT



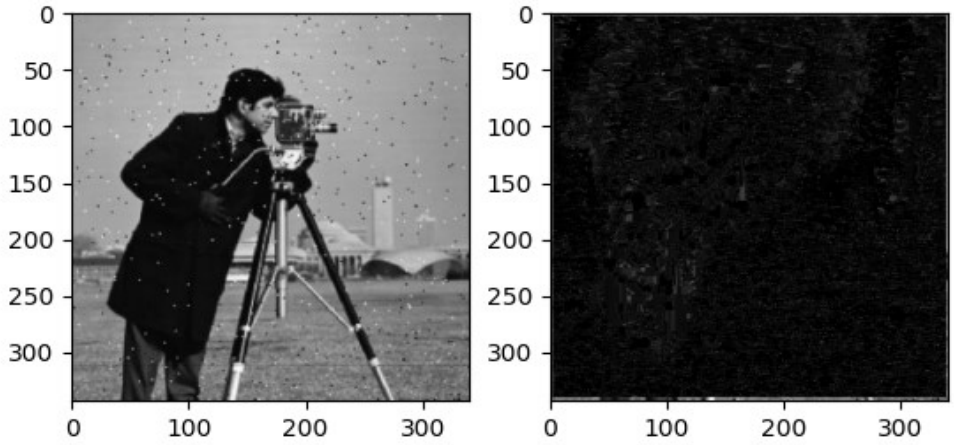
4.Median Masking

```
#order static filter (median filtering -> 50th percentile)
# non linear filter
import numpy as np
from matplotlib import pyplot as plt
import cv2



# plt.rcParams["figure.figsize"] = [7.50, 3.50]
# plt.rcParams["figure.autolayout"] = True



img = cv2.imread("saltpepper.jpg",1)



height,width,dim = img.shape

# height,width = img.shape
img_new = cv2.imread("saltpepper.jpg",1)



# mask = np.ones((7,7))
```

```
mask = np.array([[1,1,1,1,1],[1,1,1,1,1],[1,1,1,1,1],[1,1,1,1,1],[1,1,1,1,1]])
height = height - 1
width = width - 1

kernal_size = mask.shape[0]
print(kernal_size)
kernal_size_range = kernal_size // 2

print(mask)

for i in range(kernal_size_range,height-kernal_size_range):
    for j in range(kernal_size_range,width - kernal_size_range):

        lst = []
        for k in range(-kernal_size_range ,kernal_size_range + 1):

            for l in range(-kernal_size_range,kernal_size_range + 1):
                if k==0 and l==0 :
                    continue
                else:
                    lst.append(img[i+k][j+l] *
mask[k+kernal_size_range][l+kernal_size_range])
        img_new[i][j] = np.median(lst)
```

```
        # print("-----------------------------------------------------")
        # print("Using  numpy Median is  = {}".format(np.median(lst)))
        # print("-----------------------------------------------------")
fig = plt.figure()

ax1 = fig.add_subplot(121)
ax1.imshow(img)
ax2 = fig.add_subplot(122)
ax2.imshow(img_new)



# plt.hist(img,256)
# plt.hist(img_new,256)
plt.show()



# cv2.imshow("Actual Image",image)
# cv2.imshow("Filtered Image",image1)
# cv2.waitKey(0)

# plt.show()
```

OUTPUT



5.Ramp.py

```
import cv2
import numpy as np
blank_image = np.ones((256,256,1), np.uint8)


val = 0
for x in range(0,256):
    val = 0
    for y in range(0,256):
        val = val + 1
        blank_image[x][y] = val
```

```
cv2.imshow("ARR" ,blank_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

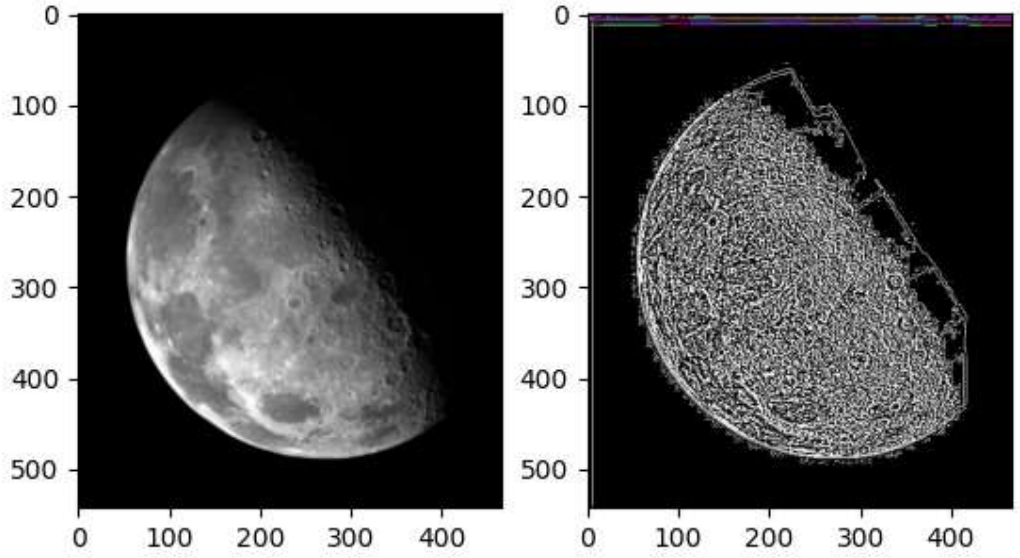
OUTPUT

6.Laplacian Filter

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



img = cv2.imread("moon.jpg",1)
cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
height,width,dim = img.shape
print(img.shape)


mask = np.array([[-1,-1,-1],[-1,8,-1],[-1,-1,-1]])
```

```
height = height - 1

width = width - 1


kernal_size = mask.shape[0]

print(kernal_size)

kernal_size_range = kernal_size // 2


print(mask)


for i in range(kernal_size_range,height):

    for j in range(kernal_size_range,width):

        ans = (img[i + 1][j] + img[i - 1][j] + img[i][j + 1]  + img[i][j - 1]) -
(img[i][j] * 4.0)

        img_new[i][j] = ans

fig = plt.figure()


ax1 = fig.add_subplot(121)

ax1.imshow(img)

ax2 = fig.add_subplot(122)

ax2.imshow(img_new)

plt.show()
```

OUTPUT



7.Edge and Border Detection

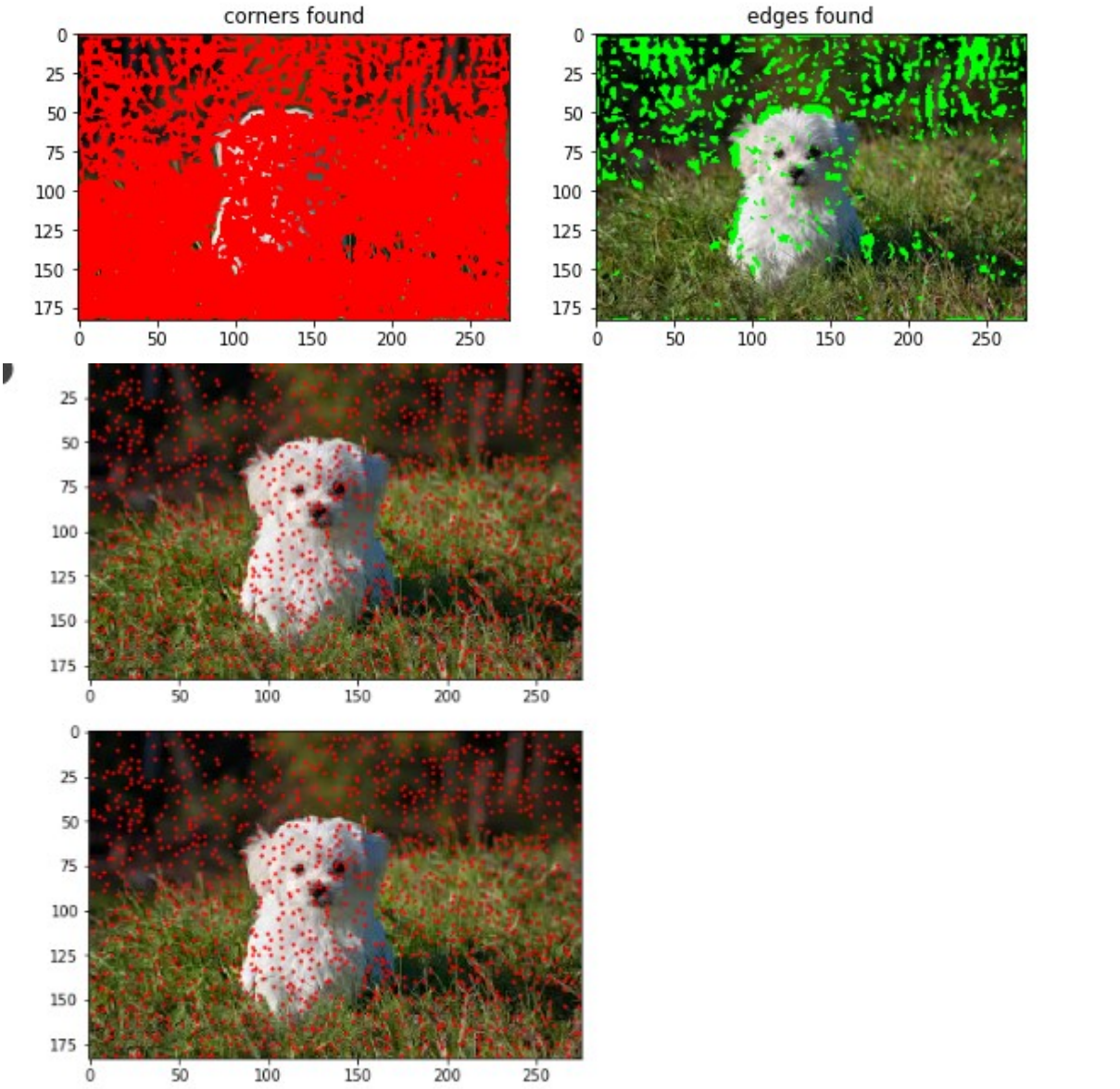
```
from skimage.io import imread
from skimage.color import rgb2gray
import matplotlib.pyplot as plt
from scipy import ndimage as ndi
from skimage.feature import peak_local_max
from skimage.feature import corner_harris, corner_peaks
img = imread('/content/dog,jpg.jpeg')
imggray = rgb2gray(img)
```

```
from scipy import signal as sig
import numpy as np
def gradient_x(imggray):
  kernel_x = np.array([[-1, 0, 1],[-2, 0, 2],[-1, 0, 1]])
  return sig.convolve2d(imggray, kernel_x, mode='same')
def gradient_y(imggray):
  kernel_y = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
  return sig.convolve2d(imggray, kernel_y, mode='same')
I_x = gradient_x(imggray)
I_y = gradient_y(imggray)
Ixx = ndi.gaussian_filter(I_x**2, sigma=1)
Ixy = ndi.gaussian_filter(I_y*I_x, sigma=1)
Iyy = ndi.gaussian_filter(I_y**2, sigma=1)
k = 0.05 # determinant
detA = Ixx * Iyy - Ixy ** 2
# trace traceA = Ixx + Iyy harris_response = detA - k * traceA ** 2
k = 0.05
# determinant
detA = Ixx * Iyy - Ixy ** 2
# trace
traceA = Ixx + Iyy

harris_response = detA - k * traceA ** 2
```

```
img_copy_for_corners = np.copy(img)
img_copy_for_edges = np.copy(img)

for rowindex, response in enumerate(harris_response):
    for colindex, r in enumerate(response):
        if r > 0:
            # this is a corner
            img_copy_for_corners[rowindex, colindex] = [255,0,0]
        elif r < 0:
            # this is an edge
            img_copy_for_edges[rowindex, colindex] = [0,255,0]

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10,10))
ax[0].set_title("corners found")
ax[0].imshow(img_copy_for_corners)
ax[1].set_title("edges found")
ax[1].imshow(img_copy_for_edges)
plt.show()
corners = corner_peaks(harris_response)
fig, ax = plt.subplots()
ax.imshow(img, interpolation='nearest', cmap=plt.cm.gray)
ax.plot(corners[:, 1], corners[:, 0], '.r', markersize=3)
#Harris corner detection using skimage library
from skimage.feature import corner_harris, corner_peaks
```

```
coords = corner_peaks(harris_response)
fig, ax = plt.subplots()
ax.imshow(img, interpolation='nearest', cmap=plt.cm.gray)
ax.plot(coords[:, 1], coords[:, 0], '.r', markersize=3)
```

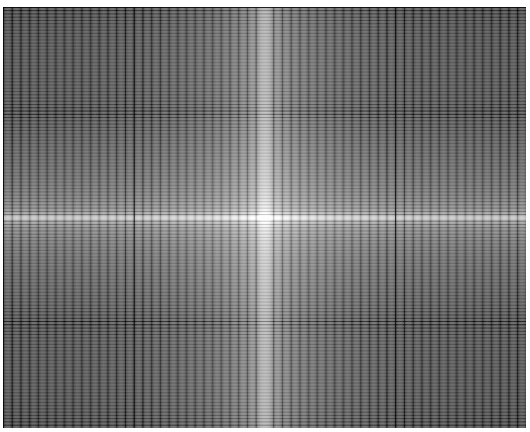
OUTPUT

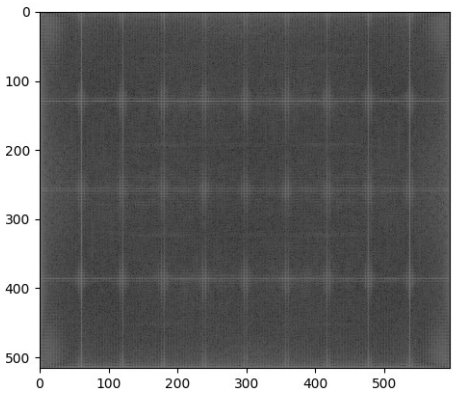


8.Fourier Transform

```
from  scipy.fft import fft,fftfreq
import numpy as np
import cv2
import math
import matplotlib.pyplot as plt

img1 = np.zeros((250,250))
img2 = np.full((250,250),255)
img3 = np.hstack((img1,img2))
cv2.imwrite("blackwhite.jpg",img3)
img = cv2.imread("idft.jpg",0)

img1 = np.fft.fft2(img)
# cv2.imshow("No Filtrer",img)
plt.imshow(np.log(1 + np.abs(img1)),cmap="gray")
plt.show()
```

OUTPUT





9.Synthetic Image

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import math
# Read the image in greyscale
rows, cols = (100, 100)
arr = [[0 for i in range(cols)] for j in range(rows)]
arr=np.array(arr)
for i in range(100):
    n=0
    for j in range(20):
        arr[i][j]=n
        n=n+10
    for k in range(41,61):
        arr[i][k]=255
    arr[50][80]=255
    arr[50][85]=100
    arr[50][90]=255
    arr[50][95]=100

print(type(arr))
print(arr)
```

```
plt.imshow(arr,cmap="gray")
plt.show()
cv2.waitKey(0)
```

OUTPUT