CLASS	**: B.E. E &TC**	**SUBJECT: DIVP**
EXPT. NO.	**: 4**	**DATE:18-09-2020**

TITLE : TO PERFORM IMAGE ENHANCEMENT IN SPATIAL DOMAIN.

CO 1:	Apply the fundamentals of digital image processing to perform various operations on an image-enhancement in spatial domain/ frequency domain, image-restoration, image compression, video filtering and video compression on a given gray image. Examine the effect of varying the mask size and density of noise in an image and comment on the obtained results.
CO4:	Carry out experiments as an individual and in a team, comprehend and write a laboratory recordand draw conclusions at a technical level.

AIM:

To implement the following filters using matlab

1. Box Filter
2. Weighted filter.
3. Median filter.
4. Laplacian filter.
5. High boost filter

SOFTWARES REQUIRED : Matlab 7.0. or above, python

THEORY:

4.1 Basics of filtering operation

Filtering refers to accepting or rejecting certain frequency components. Filtering creates a new pixel with co-ordinates equal to the co-ordinates of the centre of the neighborhood and whose value is the result of the filtering operation. The processed (filtered) image is generated as the centre of the filter mask visits each pixel in the input image.

Image enhancement approaches fall into two broad categories: spatial domain methods and frequency domain methods. The term *spatial domain* refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. *Frequency domain* processing techniques are based on modifying the Fourier transform of an image.The term *spatial domain* refers to the aggregate ofpixels composing an image. Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression

$$g(x, y) = T[f(x, y)]$$

where f(x, y) is the input image, g(x, y) is the processed image, and *T* is an operator on *f*, defined over some neighborhood of (x, y). Some neighborhood operations work with the values of the image pixels in the neighborhood *and* the corresponding values of a sub-image that has the same dimensions as the neighborhood. The sub-image is called a *filter, mask*, *kernel*, *template*, or *window*. The process of spatial filtering consists simply of moving the filter mask from point to point in an image. For *linear* spatial filtering the response is given by a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask.

When interest lies on the response, *R*, of an m*n mask at any point (x, y), and not on the mechanics of implementing mask convolution, it is common practice to simplify the notation by using the following expression:

$R = w1\ z1 + w2\ z2 + \ldots + w_{mn}z_{mn}$

$= \sum_{i=1}^{mn} w_i z_i$ where the w's are mask coefficients, the *z*'s are the values of the image gray levels corresponding to those coefficients, and m*n is the total number of coefficients in the mask. The filtering operation is based conditionally on the values of the pixels in the neighborhood under consideration, and they do not explicitly use coefficients in the sum-of-products.

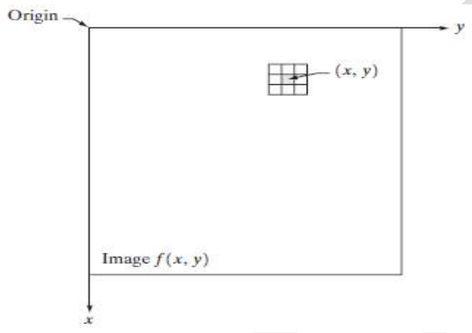


Fig: 4.1 (3*3 neighborhood about a point (x, y) in an image)

4.2 Types of spatial filtering:

4.2.1 Smoothing filters

Smoothing filters are used for blurring and for noise reduction. Blurring is used in preprocessing steps, such as removal of small details from an image prior to (large) object extraction, and bridging of small gaps in lines or curves. Noise reduction can be accomplished by blurring with a linear filter and also by nonlinear filtering.

4.2.2 Smoothing Linear Filters

The output of a smoothing, linear spatial filter is the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called *averaging filters*. It replaces the value of every pixel in an image by the average of the gray levels in the neighborhood defined by the filter mask, this process results in an image with reduced "sharp" transitions in gray levels. Averaging filters have the undesirable side effect that they

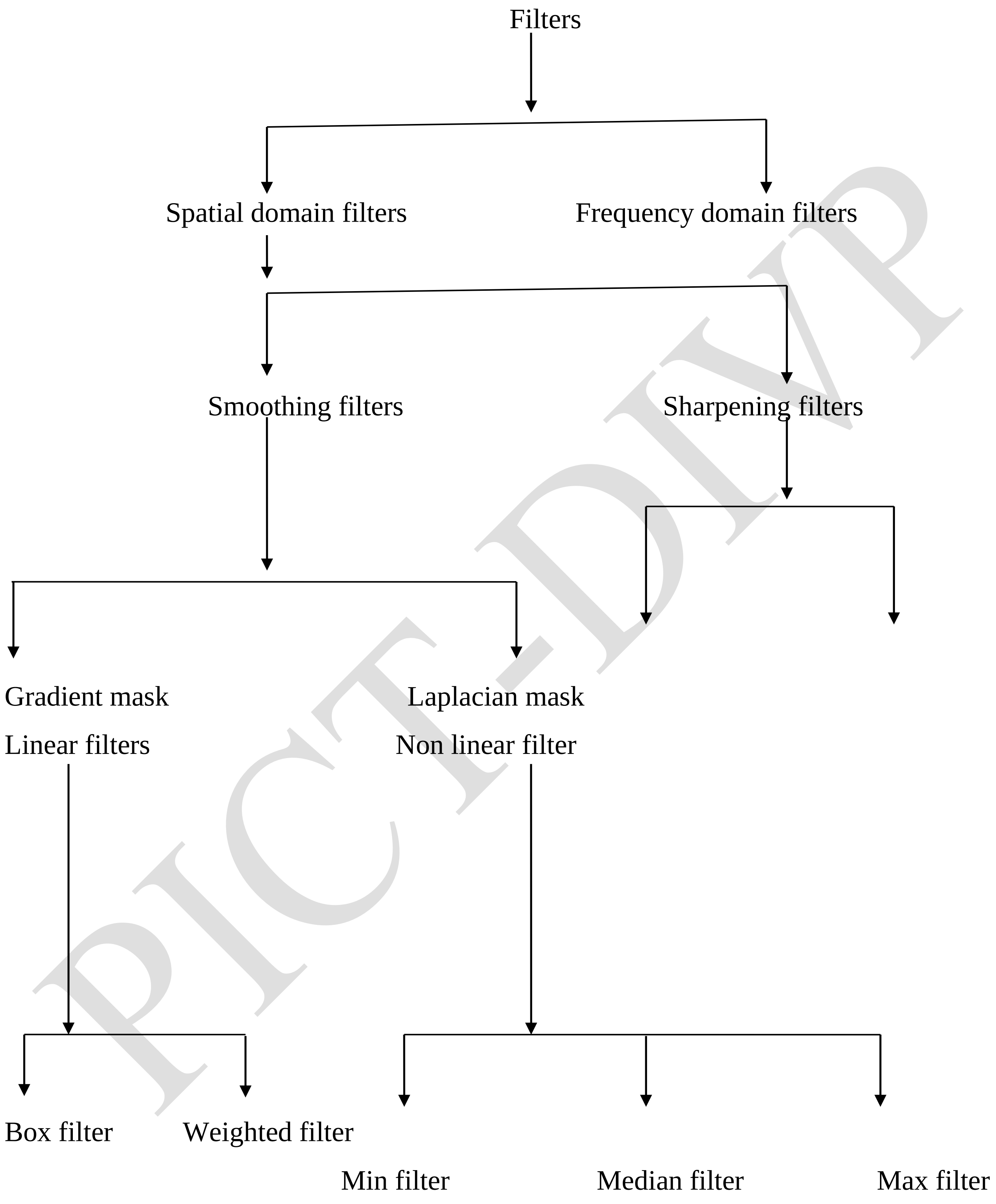
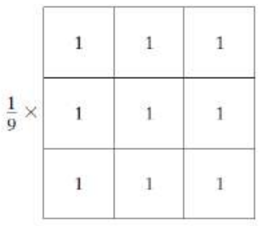

Filters
Spatial domain filters
Frequency domain filters
Smoothing filters
Sharpening filters
Gradient mask
Laplacian mask
Linear filters
Non linear filter
Box filter
Weighted filter
Min filter
Median filter
Max filter

Fig: 4.2 Filter Classifications

blur edges. A spatial averaging filter in which all coefficients are equal is sometimes called a *box filter*.

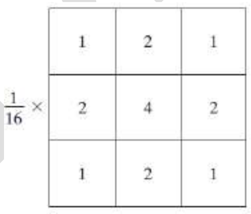
The second mask is *weighted average mask*, pixels are multiplied by different coefficients, thus giving more importance (weight) to some pixels at the expense of others. The basic strategy behind weighing the center point the highest and then reducing the value of the coefficients as a function of increasing distance from the origin is simply an attempt to reduce blurring in the smoothing process.

Box filter



$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Weighted average filter



$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Fig: 4.3Smoothing Filter Mask

4.2.3 Smoothing non-linear filters (Order-Statistics Filters)

Order-statistics filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result. There are median filters, Max and Min filters.

Median filter

1	1	1
1	1	1
1	1	1

4.2.4 Sharpening Filters

The principal objective of sharpening is to highlight fine detail in an image or to enhance detail that has been blurred. Since averaging is analogous to integration, it is logical to conclude that sharpening could be accomplished by spatial differentiation. Image differentiation enhances edges and other discontinuities (such as noise) and deemphasizes areas with slowly varying gray-level values.

Comparing the response between first- and second-order derivatives:

First-order derivatives	Second-order derivatives
(1) First-order derivatives generally produce thicker edges in an image.	(1)Second-order derivatives have a stronger response to fine detail, such as thin lines and isolated points.
(2) First order derivatives generally have a stronger response to a gray-level step.	(2) Second-order derivatives produce a double response at step changes in gray level.

A basic definition of the first-order derivative of a one-dimensional function f(x) is the difference

$$\partial f/\partial x = f(x + 1) - f(x).$$

Similarly, we define a second-order derivative as the difference

$$\partial^2 f/\partial x^2 = f(x + 1) + f(x - 1) - 2f(x)$$

4.3 Use of Second Derivatives for Enhancement–The Laplacian

The approach basically consists of defining a discrete formulation of the second-order derivative and then constructing a filter mask based on that formulation.We are interested in *isotropic* filters, isotropic filters are *rotation invariant* whose response is independent of the direction of the discontinuities in the image to which the filter is applied.

simplest isotropic derivativeoperator is the *Laplacian*, which, for a function (image) f(x, y) of two variables, is defined as

$$\partial^2 f = \partial^2 f/\partial x^2 + \partial^2 f/\partial y^2$$

Because derivatives of any order are linear operations, the Laplacian is a linear operator. Because the Laplacian is a derivative operator, its use highlights gray-level discontinuities in an image and deemphasizes regions with slowly varying gray levels. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be “recovered” while still preserving the sharpening effect of the Laplacian operation simply by adding the original and Laplacian images.

Mask :

-1	-1	-1
-1	8	-1
-1	-1	-1

$$g(x,y) = f(x,y) + \nabla^2 f(x,y) \qquad 1$$

$$g(x,y) = f(x,y) - \nabla^2 f(x,y) \qquad 2$$

Equation 1 is used when center coefficient is positive and Equation 2 is used when center coefficient is negative.

4.4 High boost filter

A process Generalization of unsharp masking is called *high boost filtering*.A high boost filtered,*f,is defined at any point (x,y)as*

$$f = Af(x,y) - \bar{f}(x,y)$$

The following masks are used

0	**-1**	**0**
-1	**A+4**	**-1**
0	**-1**	**0**

-1	**-1**	**-1**
-1	**A+8**	**-1**
-1	**-1**	**-1**

Note that,when A=1,high boost filtering becomes standard Laplacian sharpening.

4.5 Algorithm :

1. Start
2. Load required python libraries like OpenCV , matplotlib , NumPy
3. Read a grayscale image
4. Define kernels for different smoothing filters like box , median , weighted average filter.
5. Define kernels for different sharpening filters like Laplacian and high boost filter.
6. Convolve each kernel one by one with grayscale image and observe the result.
7. Plot histogram of every image after applying the filter to compare the results obtained.
8. End

4.6 Conclusion:

In the experiment we applied different smoothing and sharpening filters and compared the results as follows:

A. Smoothing filters:

1. **Box filter**: It is linear filter and generally used for blurring of images
2. **Median filter**: It is nonlinear filter, often used to remove noise from image. It is used in preprocessing of image before applying edge or line detection algorithms.
3. **Weighted filter**: It is linear filter, and it is used for reducing the blurring in smoothing operation. This is obtained by multiplying every pixel value of box value by some coefficients giving higher weightage to some pixels in expense of other.

B. Sharpening filters:

1. **Laplacian filter**: Laplacian filters are derivative filters used to find areas of rapid change (edges) in images. It is very sensitive to noise hence commonly a smoothing filter is used before using Laplacian filter. There are other variations of Laplacian filter i.e., LoG and DoG filters.
2. **High boost filter**: These are single derivative filters hence its accuracy is less than that of Laplacian filters.

4.7 References:

i. GonzalezR, Woods R, “Digital image processing”, Pearson Prentice Hall, 2008.

ii. GonzalezR, Woods R, Steven E, “Digital Image Processing Using MATLAB®”, McGraw Hill Education, 2010.

iii. Jayaraman S, Esakkirajan S and Veerakumar T, “Digital Image Processing” Tata McGraw Hill, 2010

iv. Joshi, Madhuri A. “Digital Image Processing: an algorithm approach”, PHI Learning Pvt. Ltd., 2006.

v. Pictures taken from: http://www.imageprocessingplace.com/root_files_V3/image_databases.html

(Course Teacher)

CLASS	:	B.E (E &TC)	COURSE	:	DIVP
AY	:	2020-21 (SEM- I)	DATE	:	18-09-2020
EXPT. NO.	:	4	CLASS & ROLL NO	:	BE VIII 42428
TITLE	:	TO PERFORM IMAGE ENHANCEMENT IN SPATIAL DOMAIN			

I. CODE:

Box and Weighted Average Filter

```
import cv2

# import Numpy
import numpy as np
from matplotlib import pyplot as plt

def convo(img,k,f):
    height,width=img.shape
    img_pad = np.zeros((height+2,width+2),dtype = 'uint8')
    img_pad_out = np.zeros((height+2,width+2),dtype = 'uint8')
    img2 = np.zeros((height,width),dtype = 'uint8')
    for i in range(0,width):
        for j in range(0,height):
            img_pad[j+1,i+1]=img[j,i]
    height1,width1=img_pad.shape
    for i in range(0,width1-2):
        for j in range(0,height1-2):
            s=(img_pad[j,i]*k[0,0])+(img_pad[j,i+1]*k[0,1])+(img_pad[j,i+2]*k[0,2])+(img_pad[j+1,
i]*k[1,0])+(img_pad[j+1,i+1]*k[1,1])+(img_pad[j+1,i+2]*k[1,2])+(img_pad[j+2,i]*k[2,0])+(img
_pad[j+2,i+1]*k[2,1])+(img_pad[j+2,i+2]*k[2,2]);
            img_pad_out[j+1,i+1]=s/f;
    #img_pad = np.array(img_pad, dtype = np.uint8)
    for i in range(0,width):
        for j in range(0,height):
            img2[j,i]=img_pad_out[j+1,i+1]
    return img2

# read a image using imread
img1 = cv2.imread('Images/rose.jpg',0)

#Box Filtering
k1 = np.array((
    [1, 1, 1],
    [1, 1, 1],
    [1, 1, 1]), dtype="int")
img2=convo(img1,k1,9)

#Weighted Averaging Filtering
k2 = np.array((
    [1, 2, 1],
    [2, 4, 2],
    [1, 2, 1]), dtype="int")
img3=convo(img1,k2,16)

plt.subplot(2, 3, 1),plt.imshow(img1, 'gray'),plt.title('Original Image')
plt.subplot(2, 3, 2),plt.imshow(img2, 'gray'),plt.title('Box Filtered Image')
plt.subplot(2, 3, 3),plt.imshow(img3, 'gray'),plt.title('Weighted Filtered Image')
plt.subplot(2, 3, 4), plt.hist(img1.ravel(),256,[0,256]),plt.title('Histogram of Original Image')
plt.subplot(2, 3, 5), plt.hist(img2.ravel(),256,[0,256]),plt.title('Histogram of Box Filtered Image')
plt.subplot(2, 3, 6), plt.hist(img3.ravel(),256,[0,256]),plt.title('Histogram of Weighted Filtered Im
age')
plt.show()
```

Min, Max and Median Filter

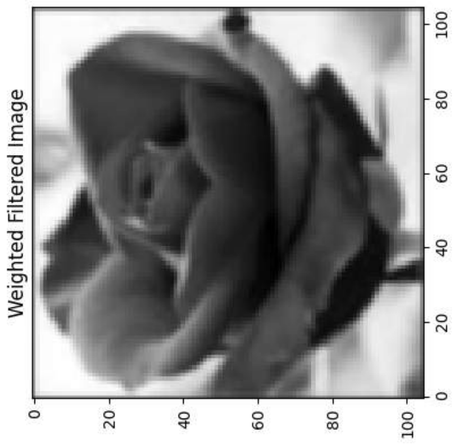
```
import cv2

# import Numpy
import numpy as np
from matplotlib import pyplot as plt

def min_filter(img):
    height,width=img.shape
    img_pad = np.zeros((height+2,width+2),dtype = 'uint8')
    img_pad_out = np.zeros((height+2,width+2),dtype = 'uint8')
    img2 = np.zeros((height,width),dtype = 'uint8')
    for i in range(0,width):
        for j in range(0,height):
            img_pad[j+1,i+1]=img[j,i]
    height1,width1=img_pad.shape
    s=258;
    for i in range(0,width1-2):
        for j in range(0,height1-2):
            for l in range(i,i+3):
                for m in range(j,j+3):
                    if(img_pad[m,l]<=s):
                        s=img_pad[m,l]
            img_pad_out[j+1,i+1]=s;
            s=258;
    for i in range(0,width):
        for j in range(0,height):
            img2[j,i]=img_pad_out[j+1,i+1]
    return img2

def max_filter(img):
    height,width=img.shape
    img_pad = np.zeros((height+2,width+2),dtype = 'uint8')
    img_pad_out = np.zeros((height+2,width+2),dtype = 'uint8')
    img2 = np.zeros((height,width),dtype = 'uint8')
    for i in range(0,width):
        for j in range(0,height):
            img_pad[j+1,i+1]=img[j,i]
    height1,width1=img_pad.shape
    s=0;
    for i in range(0,width1-2):
        for j in range(0,height1-2):
            for l in range(i,i+3):
                for m in range(j,j+3):
                    if(img_pad[m,l]>=s):
                        s=img_pad[m,l]
            img_pad_out[j+1,i+1]=s;
            s=0;
    for i in range(0,width):
        for j in range(0,height):
            img2[j,i]=img_pad_out[j+1,i+1]
    return img2

def median_filter(img):
    height,width=img.shape
    img_pad = np.zeros((height+2,width+2),dtype = 'uint8')
    img_pad_out = np.zeros((height+2,width+2),dtype = 'uint8')
    img2 = np.zeros((height,width),dtype = 'uint8')
    for i in range(0,width):
        for j in range(0,height):
            img_pad[j+1,i+1]=img[j,i]
    height1,width1=img_pad.shape
    for i in range(0,width1-2):
        for j in range(0,height1-2):
            num = [ img_pad[j,i],img_pad[j+1,i],img_pad[j+2,i],img_pad[j,i+1],img_pad[j+1,i+1],im
g_pad[j+2,i+1],img_pad[j,i+2],img_pad[j+1,i+2],img_pad[j+2,i+2] ];
            num.sort();
            img_pad_out[j+1,i+1]=num[4];
    for i in range(0,width):
        for j in range(0,height):
            img2[j,i]=img_pad_out[j+1,i+1]
    return img2

# read a image using imread
img1 = cv2.imread('Images/min_max_filter/img1.jpeg',0)

img2=min_filter(img1)
img3=max_filter(img1)
img4=median_filter(img1)

plt.subplot(2, 4, 1),plt.imshow(img1, 'gray'),plt.title('Original Image')
plt.subplot(2, 4, 2),plt.imshow(img2, 'gray'),plt.title('Min Filtered Image')
plt.subplot(2, 4, 3), plt.imshow(img3, 'gray'),plt.title('Max Filtered Image')
plt.subplot(2, 4, 4), plt.imshow(img4, 'gray'),plt.title('Median Filtered Image')
plt.subplot(2, 4, 5), plt.hist(img1.ravel(),256,[0,256]),plt.title('Histogram of Original Image')
plt.subplot(2, 4, 6), plt.hist(img2.ravel(),256,[0,256]),plt.title('Histogram of Min Filtered Image')
plt.subplot(2, 4, 7), plt.hist(img3.ravel(),256,[0,256]),plt.title('Histogram of Max Filtered Image')
plt.subplot(2, 4, 8), plt.hist(img4.ravel(),256,[0,256]),plt.title('Histogram of Median Filtered Imag
e')
plt.show()
```

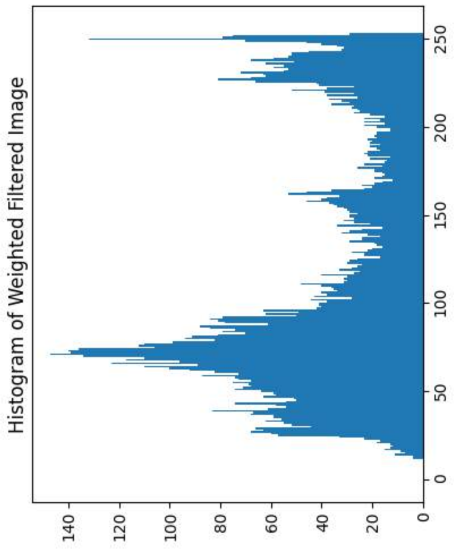
Laplacian and High Boost Filter

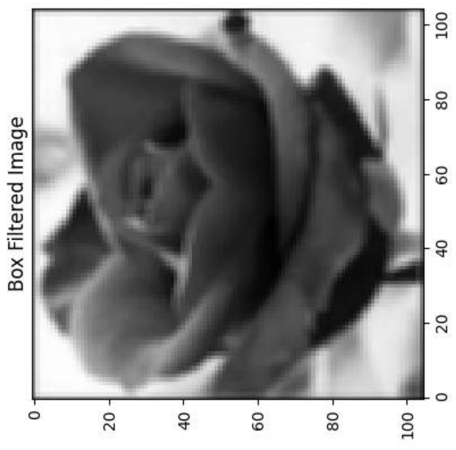
```
import cv2

# import Numpy
import numpy as np
from matplotlib import pyplot as plt

def convo(img,k):
    height,width=img.shape
    img_pad = np.zeros((height+2,width+2),dtype = 'uint8')
    img_pad_out = np.zeros((height+2,width+2),dtype = 'uint8')
    img2 = np.zeros((height,width),dtype = 'uint8')
    for i in range(0,width):
        for j in range(0,height):
            img_pad[j+1,i+1]=img[j,i]
    height1,width1=img_pad.shape
    for i in range(0,width1-2):
        for j in range(0,height1-2):
            s=(img_pad[j,i]*k[0,0])+(img_pad[j,i+1]*k[0,1])+(img_pad[j,i+2]*k[0,2])+(img_pad[j+1,
i]*k[1,0])+(img_pad[j+1,i+1]*k[1,1])+(img_pad[j+1,i+2]*k[1,2])+(img_pad[j+2,i]*k[2,0])+(img
_pad[j+2,i+1]*k[2,1])+(img_pad[j+2,i+2]*k[2,2]);
            img_pad_out[j+1,i+1]=s/9;
    img_pad = np.array(img_pad, dtype = np.uint8)
    for i in range(0,width):
        for j in range(0,height):
            img2[j,i]=img_pad_out[j+1,i+1]
    return img2+img

# read a image using imread
img1 = cv2.imread('Images/laplacian/coin.jpg',0)
#Laplacian Filtering
k1 = np.array((
    [-1, -1, -1],
    [-1, 8, -1],
    [-1, -1, -1]), dtype="int")
img2=convo(img1,k1)

#High Boost Filtering
k2 = np.array((
    [-1, -1, -1],
    [-1, 9, -1],
    [-1, -1, -1]), dtype="int")
img3=convo(img1,k2)

plt.subplot(2, 3, 1),plt.imshow(img1, 'gray'),plt.title('Original Image')
plt.subplot(2, 3, 2),plt.imshow(img2, 'gray'),plt.title('Laplacian Filtered Image')
plt.subplot(2, 3, 3),plt.imshow(img3, 'gray'),plt.title('High Boost Filtered Image')
plt.subplot(2, 3, 4), plt.hist(img1.ravel(),256,[0,256]),plt.title('Histogram of Original Image')
plt.subplot(2, 3, 5), plt.hist(img2.ravel(),256,[0,256]),plt.title('Histogram of Laplacian Filtered Im
age')
plt.subplot(2, 3, 6), plt.hist(img3.ravel(),256,[0,256]),plt.title('Histogram of High Boost Filtered I
mage')
plt.show()
```

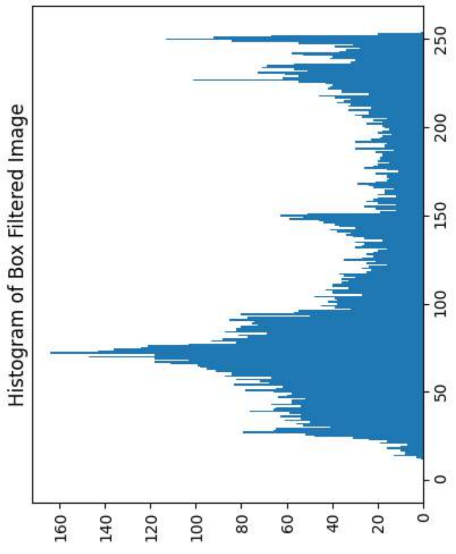
II. RESULTS:

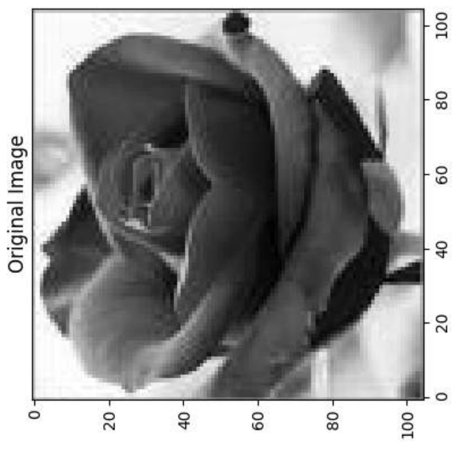
Box and Weighted Average Filter

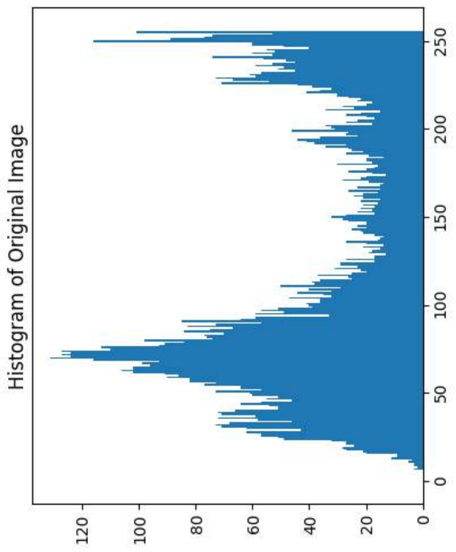




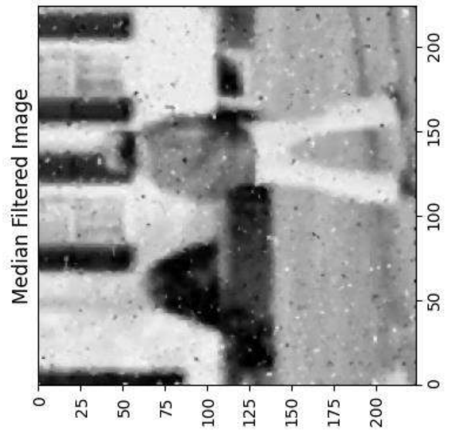


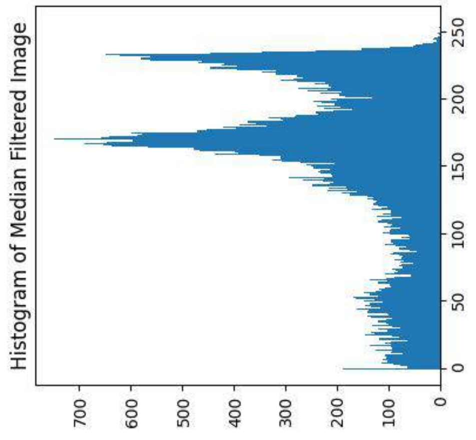


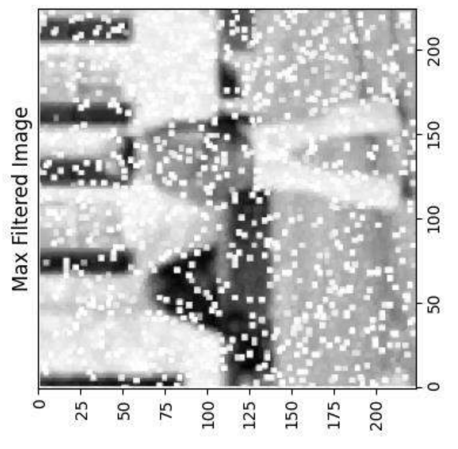


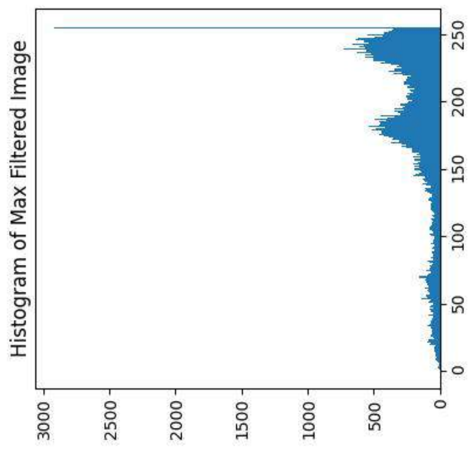


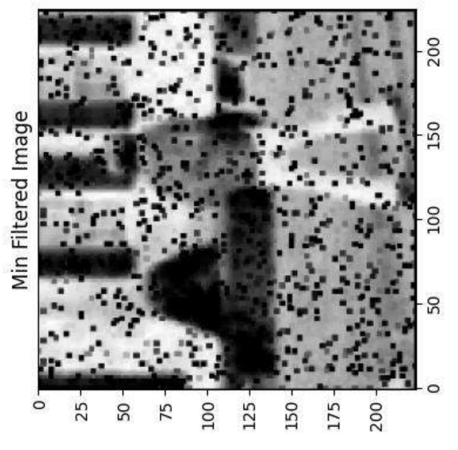
Min, Max and Median Filter

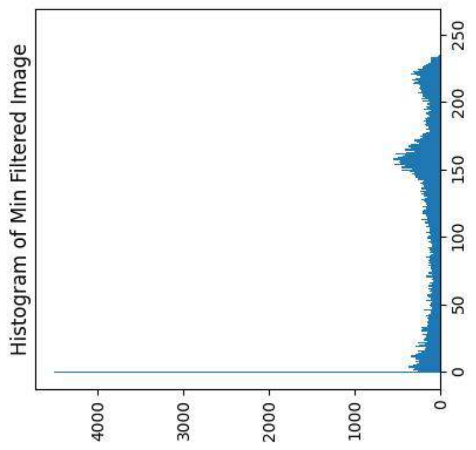


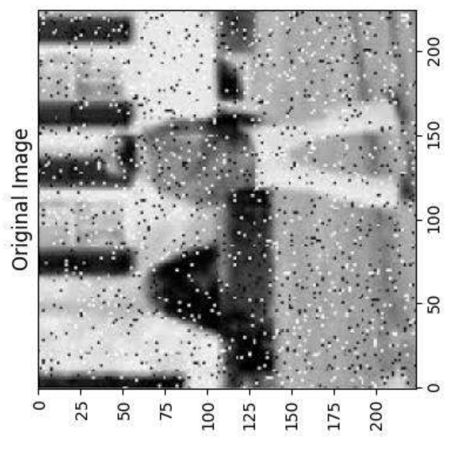


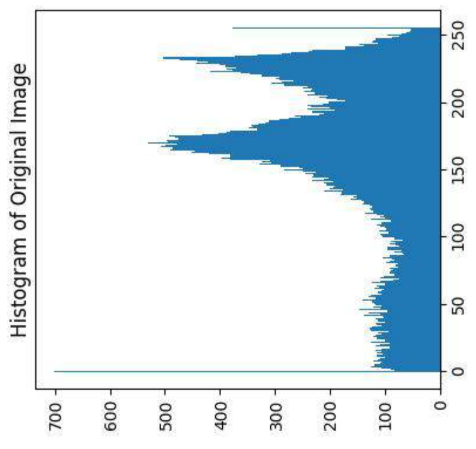




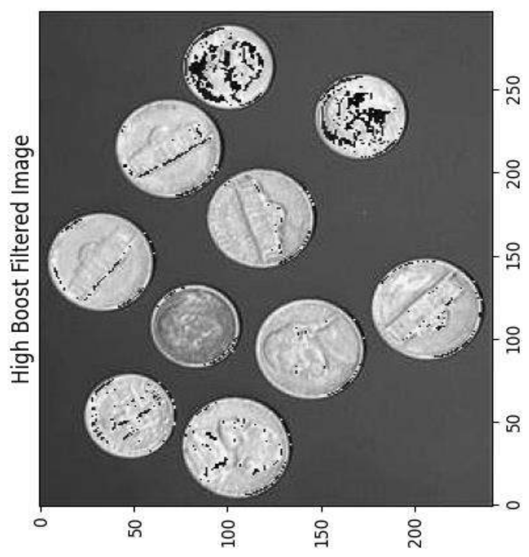


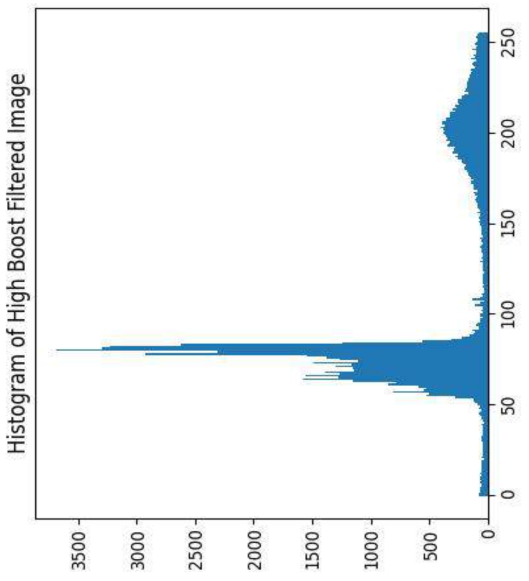


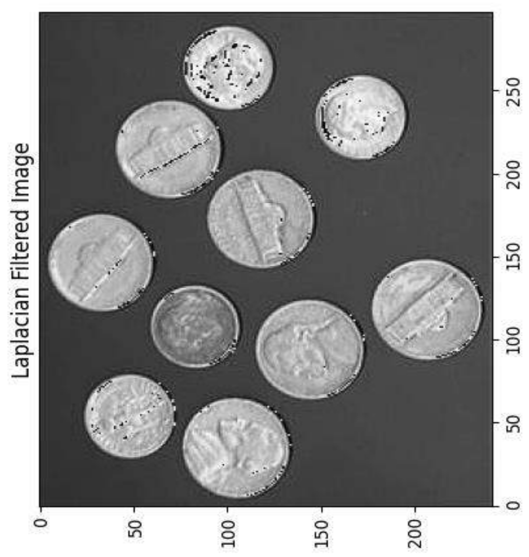


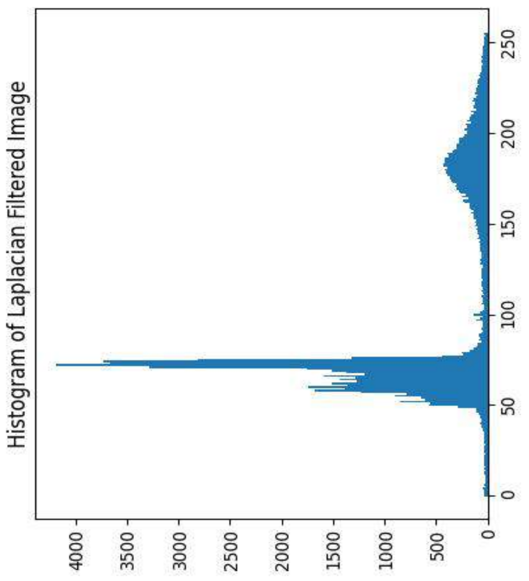


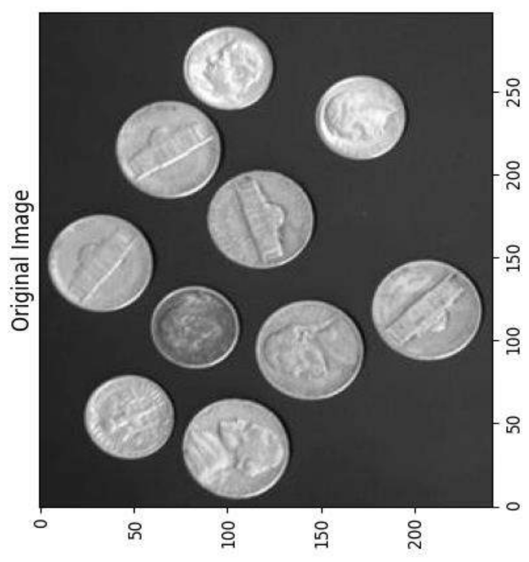
Laplacian and High Boost Filter

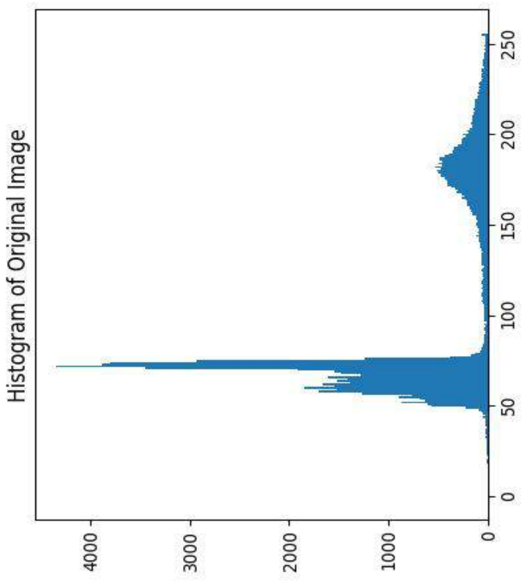



Histogram of High Boost Filtered Image




Histogram of Laplacian Filtered Image




Histogram of Original Image