FILTERING AND EDGE DETECTION

Assignment 1

TEAM 1

- HABIBA FATAHALLA
- RAWAN MOHAMMED FEKRY
- SARA AYMN EL-WATANY

- Eng. Peter Emad
- Eng. Laila Abbaas

Introduction:

To implement the algorithms in the assignment from scratch we used C++ and Qt for gui, to be able to obtain the same results as the algorithms in Open cv built in library.

To read and show the images we used open cv methods, and we also used it to test our output to see if it’s accurate or not compared to the built in algorithms.

Add Noise to the Image:

Gaussian Noise:

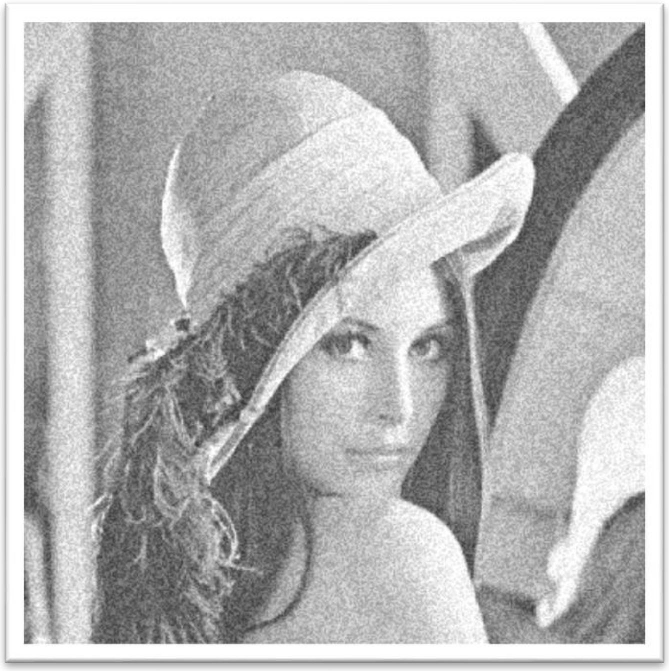
Mean val.=10   stand.val.=10

These are input parameters given by the user which defines the gaussian/normal distribution desired for the noise matrix which will then be added to the original image.

Salt & Pepper Noise:

PA Val.=10   PB Val.=10

The amount of salt and pepper noise will be identified by using some probability for the salted along with peppered pixels, where using these probabilities you can get the noise matrix having the total number of pixels present in the original image and then add it to it.

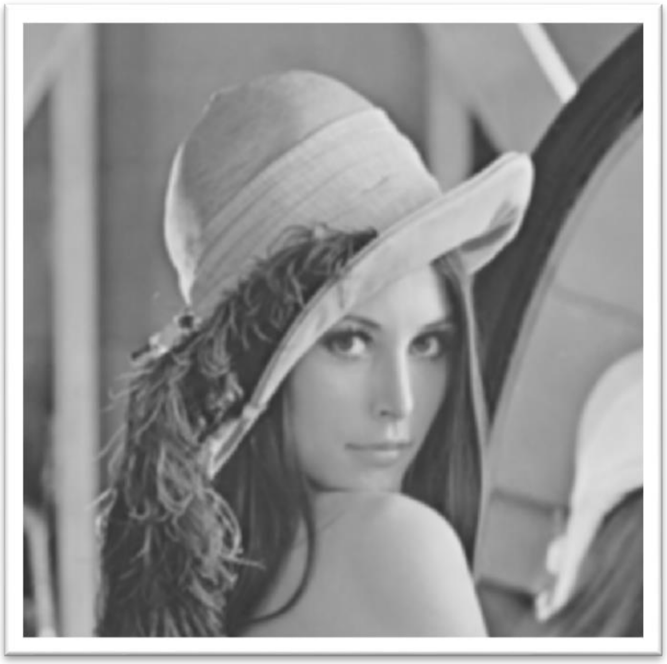


Uniform Noise:

A Val.=0.02   B Val.=0.1

These parameters define your uniform distribution for your noise matrix and they can be set by the user

Filter the Noisy Image:



Average Filter:

Kernel size= 5

Kernel size should be odd number

Algorithm:

Average filter is a linear filter that we can implement it by:

1-Building n*n mask of ones according to user choice, then multiplying the kernel by (1/ (kernel Size * kernel Size))

2- Apply convolution between original image matrix and kernel matrix.

In this algorithm smoothing happens by summate the neighbor pixels for each pixel and take the average, so all pixels become in a close range.

Results comment: output image become blurred than input **image**, **and when increasing** kernel size the image become more blurred.

Gaussian Filter:

Sigma= 3   Kernel size= 7

Algorithms:

1- Build a kernel according to Gaussian equation, which make different effect from average kernel (we **don't assume** that all neighbor pixels has the same effect. Close neighbor take big value and this value become smaller when going away from pixel)

2- Convolve the kernel with the original image

- The role of sigma in the Gaussian filter is to control the variation around its mean value. So less sigma means low less variance allowed around mean which means less smoothing.

Results comment: output image become blurred than input **image**, **and when increasing** kernel size and sigma the image become more blurred.

Median Filter:

Kernel size= 5

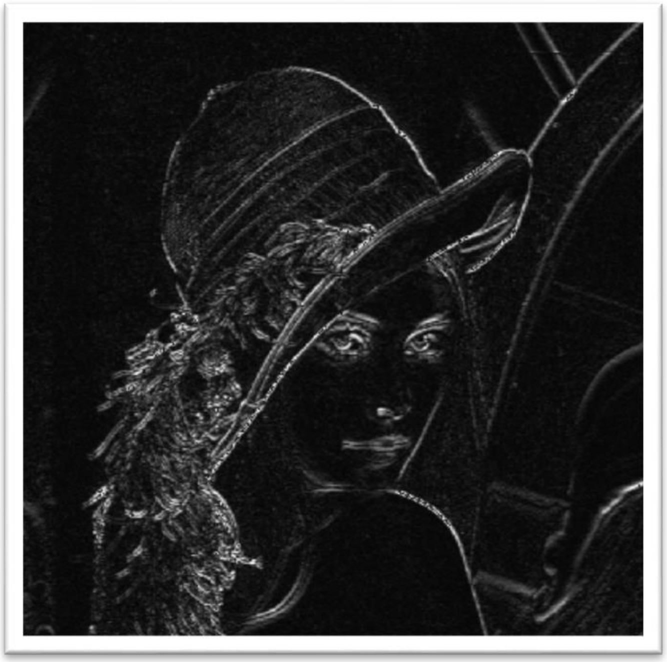
This filter is **used to remove** salt and pepper noise.

Algorithm:

1-Chosse a kernel size.
2- Sort values of kernel.
3-output pixel will be the median of sorted values.

Results comment: output image become pure than input image with no noise, and removing of noise depends on kernel size.
Bigger kernel size, better noise removal.

Edge Detection:

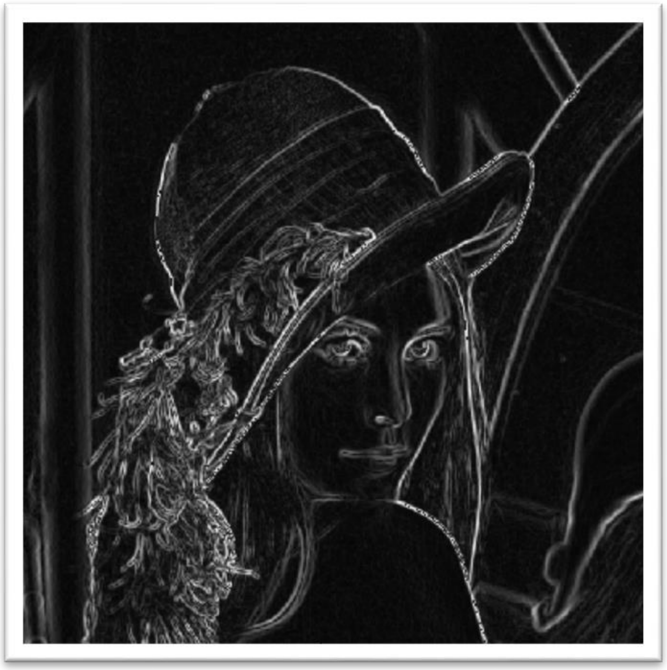


Sobel Edge Detector:

Kernal size= 3x3

Sobel edge detection works by calculating the gradient of image intensity at each pixel within the image, it finds the direction of the largest increase from light to dark and rate of change in that direction. Sobel doesn't produce image with high accuracy for edge detection, but its quality is adequate enough to be used in applications

Results comment: The output image nearly contains all the edges in the input image to Sobel and it is very sensitive to the noise

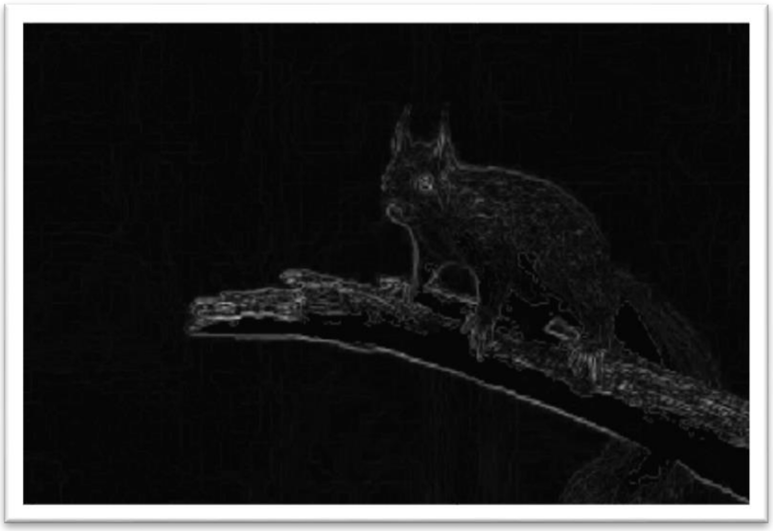


Prewitt Edge Detector:

Kernal size= 3x3

It has the same concept as Sobel edge detection, it calculate the difference between corresponding pixel intensities, The main difference between them that in Sobel we apply more weight to the pixel values around the edge region ,thus increase the edge intensity and become enhanced comparatively to the original image

Results comment: The edges is less clear tha edges in Sobel

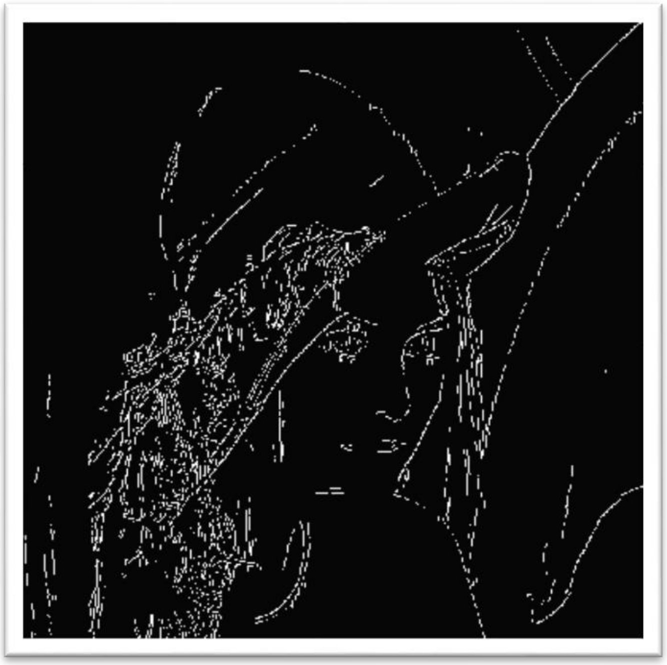


Roberts Edge Detector:

Kernal size= 2x2

By summing the squares of the differences between diagonally adjacent pixels I can be able to detect no only horizontal and vertical direction but also diagonal direction points are preserved ,best used for binary images

Results comment: The diagonal direction points are preserved



Canny Edge Detector:

Lower Threshold=50

Higher Threshold=100

Kernal size= 3

Known as **Optimal Detector,** To apply canny edge detection technique we need to :

- Apply gaussian Filters to smooth the image and remove noise from it
- Compute the gradient mag.using Sobel operator
- Apply Non-Max.Suppresion to remove the pixels that are not part of the edge
- Apply hysteresis by using to threshold values:

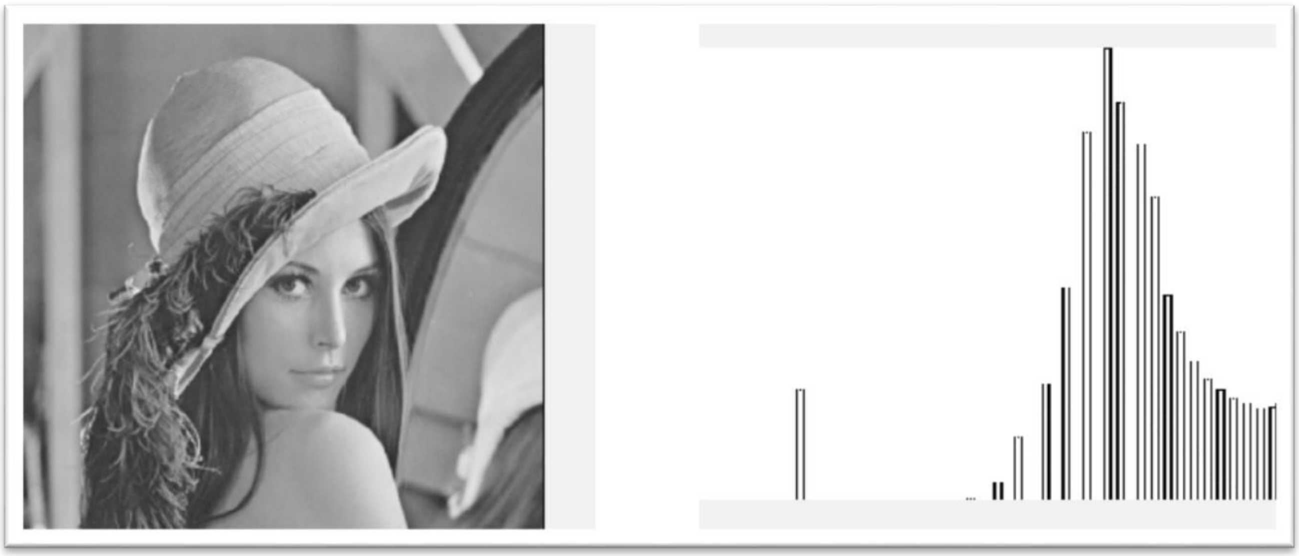
➢ if the gradient is higher than the upper threshold it will be marked as an edge]

➢ if the gradient is lower than the lower threshold it won't be marked as threshold

➢ if it lies between the two values ,the only pixel which is connected above the threshold will be marked as an edge

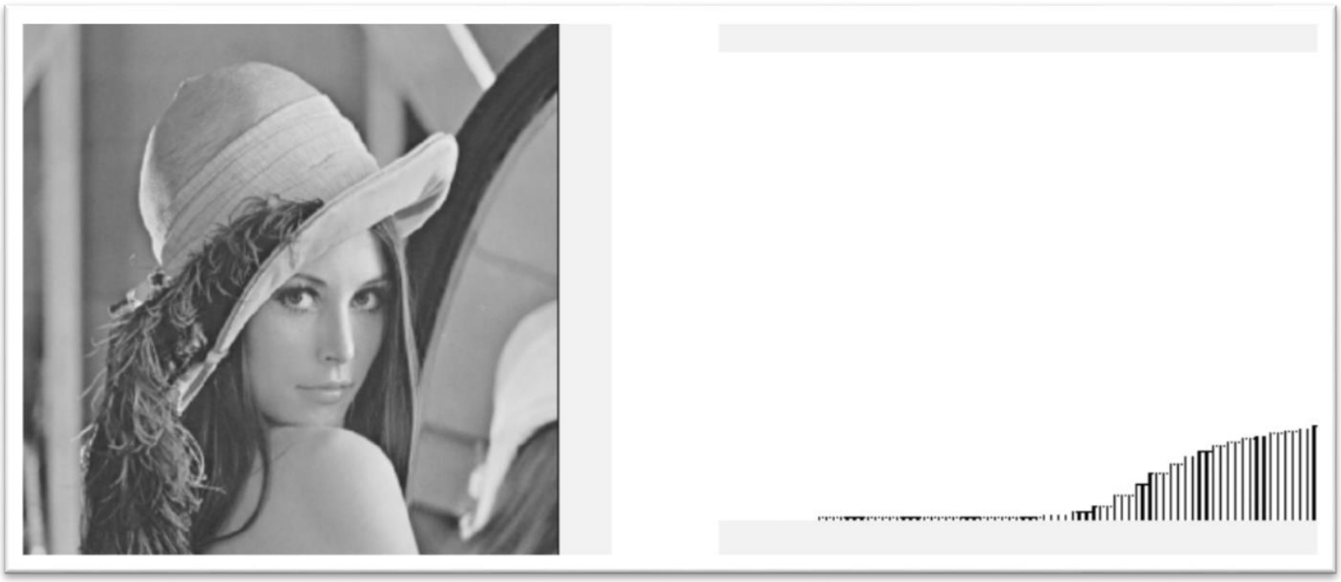
Histogram and curve Distribution:

Gray Scale Histogram:



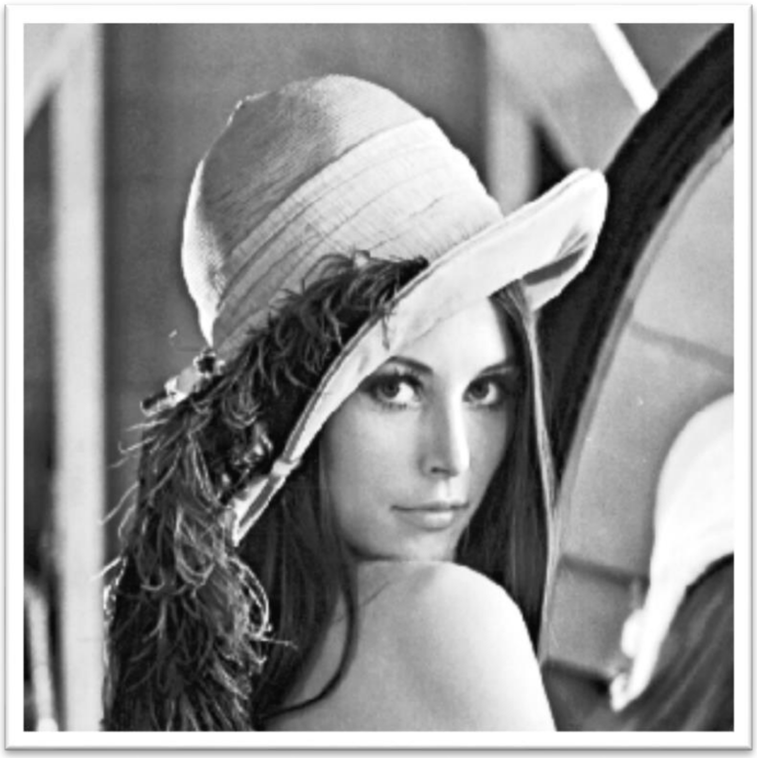
The histogram shows that there is some range where pixels don’t even exist
(the intensity of pixels isn’t well distributed).

Gray Scale Cumulative:



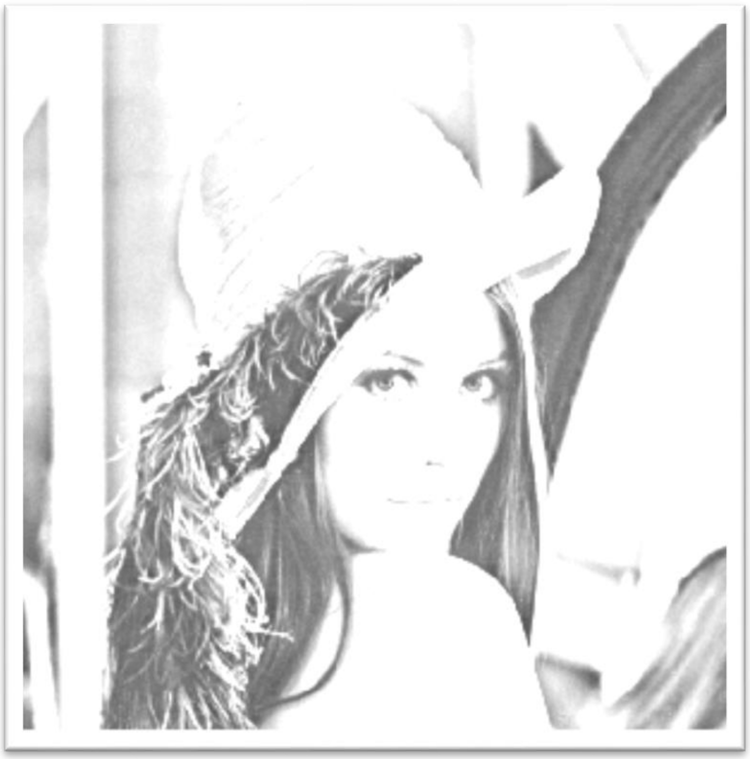
Using the cumulative distribution curve, you can clearly see that the pixels values are close to each other, and all the ranges contain certain values. This curve can then be used in the equalization of the image.

Equalize The Image:



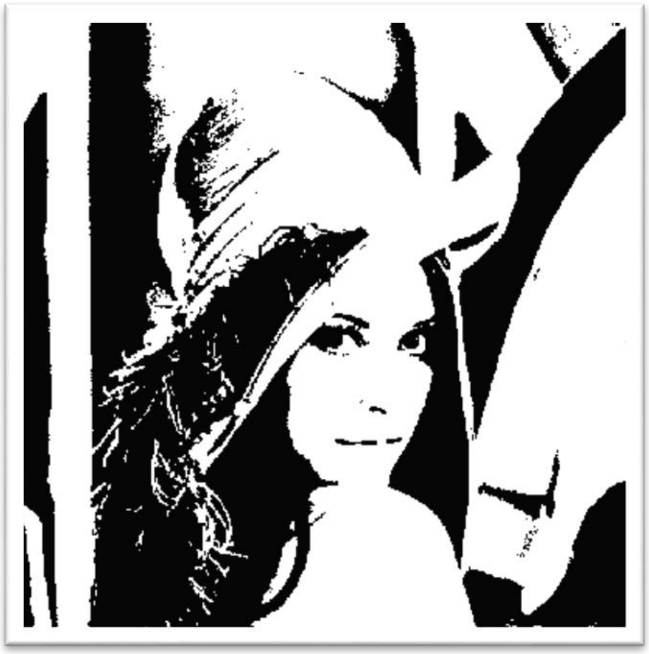
You can see that the image has higher contrast also there is no vast variation among the intensity of the pixels

Normalize The Image:



By getting the pixels with maximum and minimum intensity you can use them along with the original image to obtain the normalized image.

Local & Global Thresholding:



Global Threshold:

Threshold**=128** Max_val.=255

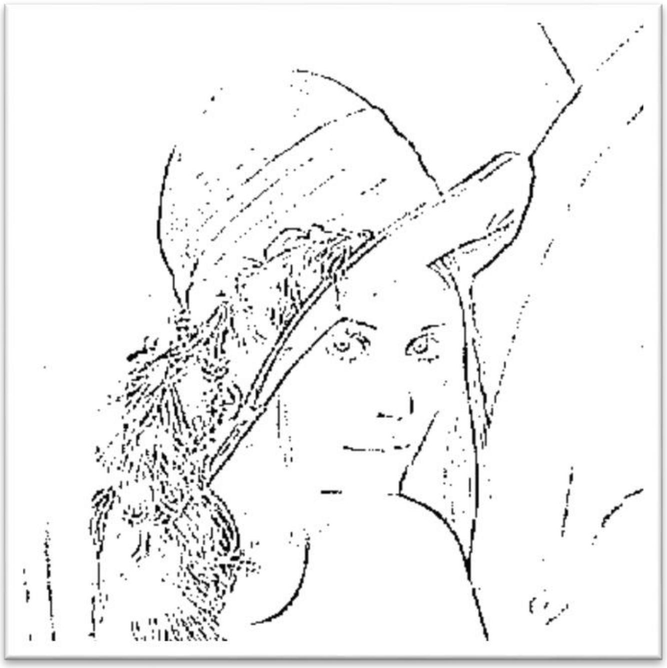
Global threshold can be implemented by many algorithms, we choose binary threshold algorithm.

Algorithm:

Choose a threshold value and check if:

- Pixel value > threshold then it's new value become max.val..
- Pixel value < threshold then it's new value become 0.

Results comment: output image is dependent on the threshold value and maximum value



Local Thresholding:

Box Size=7 C=11

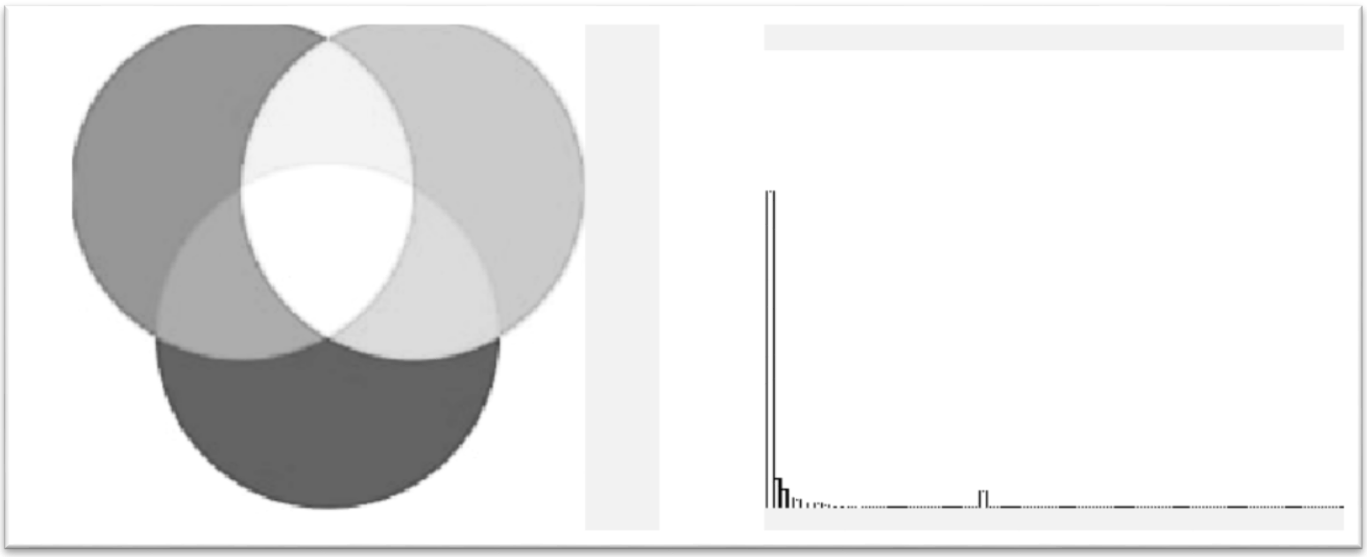
Algorithm:

- This algorithm is similar to global threshold, but threshold is updated according to mean of kernel.
- representing the constant subtracted from the mean or weighted mean

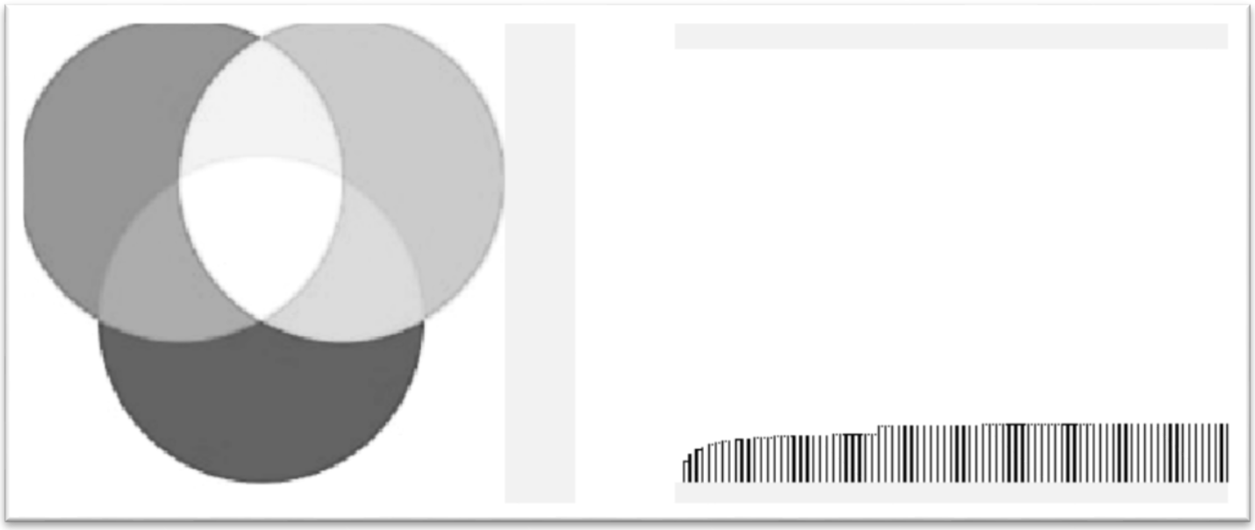
Results comment: output image is dependent on box size and C value.

Image color Transformation:

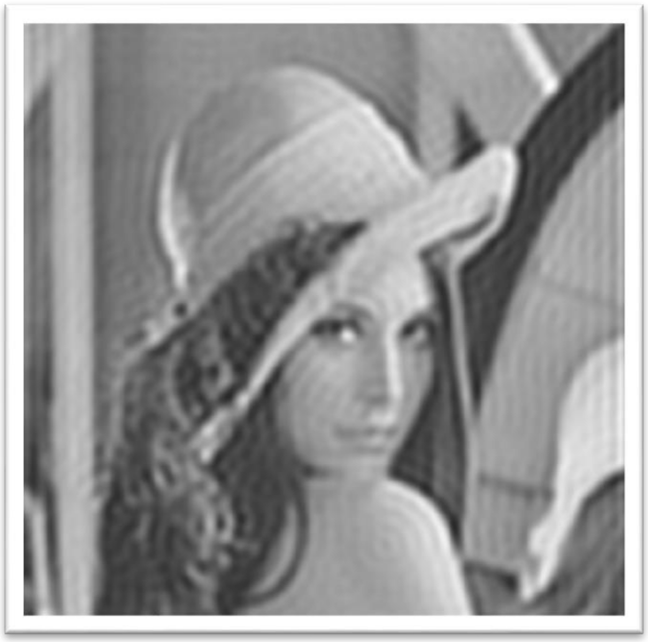
R-Channel Histogram:



G-hannel Cumulative:



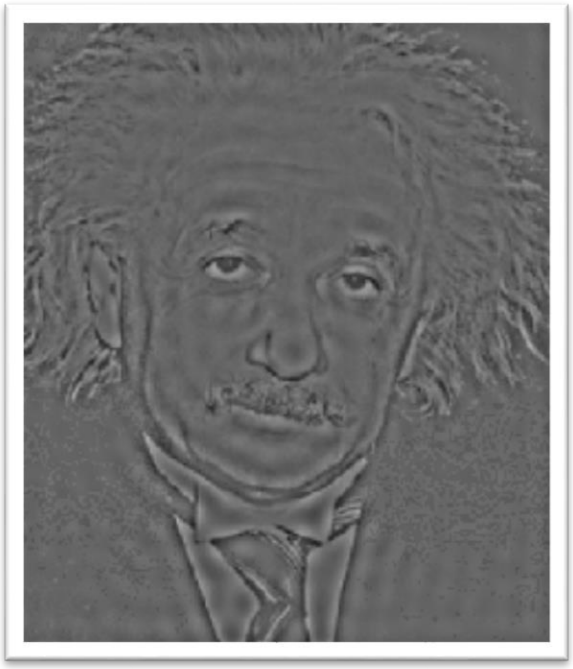
Frequency Domain Filters:



Low Pass Filter:

Radius=40

By applying low pass filter in frequency domain, Mainly to cut the frequency component from the frequency domain and the image will be blurred as seen in the output

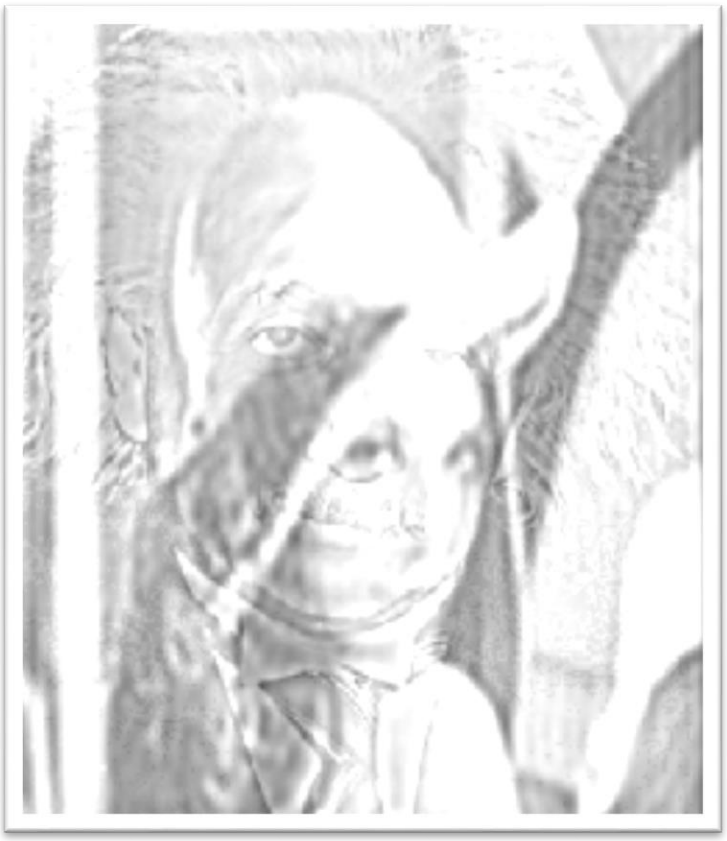


High Pass Filter:

Radius=20

By cutting the frequency component in the frequency domain which leads to brightness the center pixel which have large value (high freq.)

Hybrid Image:



When applying low pass filter and high pass filters on two images and then multiply them we get the combination of the two images with blurred version from the image which was applied to low pass filter and brightness from the image which was applied to high pass filter.