CSE473/573 Computer Vision and Image Processing

SUMMER 2016 PROGRAMMING ASSIGNMENT #1

Harsha Sudarshan
harshasu@buffalo.edu

1. 1D and 2D Convolution on Images

The objective is to implement edge filtering using Sobel filter using 1D and 2D convolution.

The image used for this is given below:

a. 2D Convolution on Images

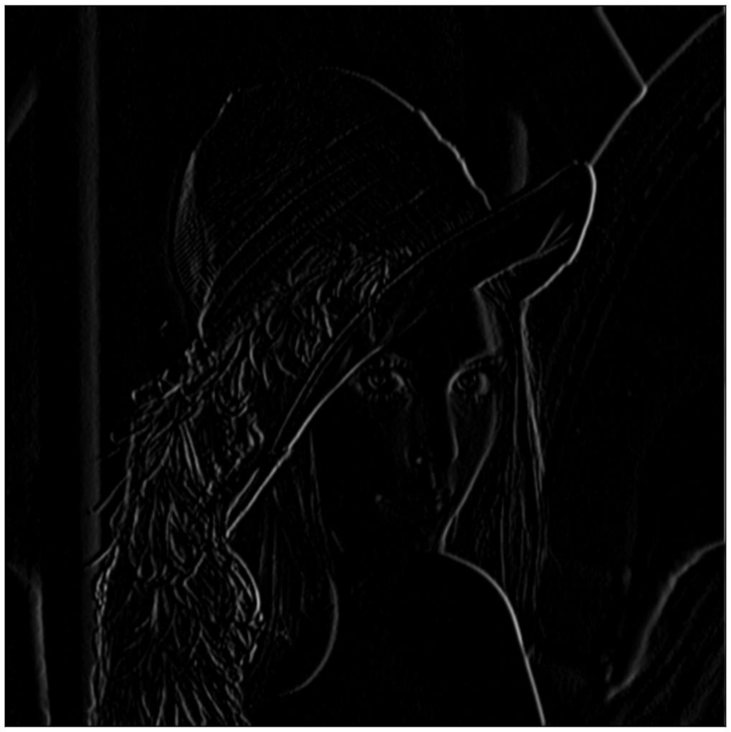
Sobel filter was implemented by using the below given x and y gradient filters on given image. The convolution was a 2D convolution and it was implemented by using the sliding window technique and by flipping the filter kernels to achieve the desired convolution effect.

The filters implemented were:

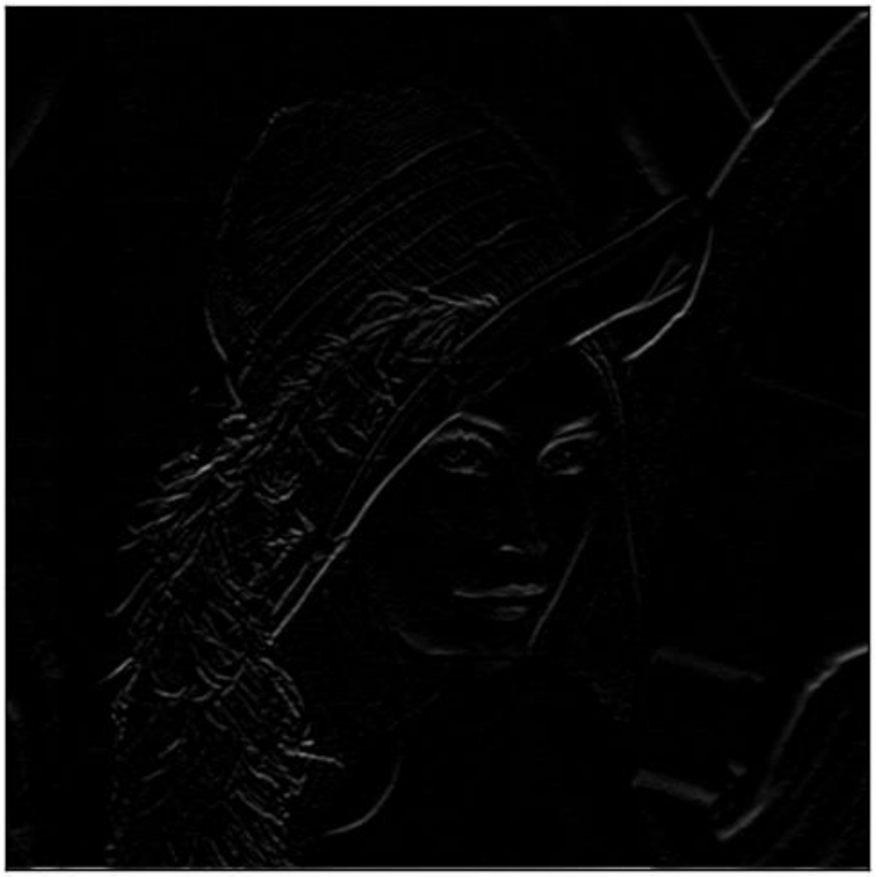
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I \qquad G_{mag} = \sqrt{G_x^2 + G_y^2}$$

The images obtained were:

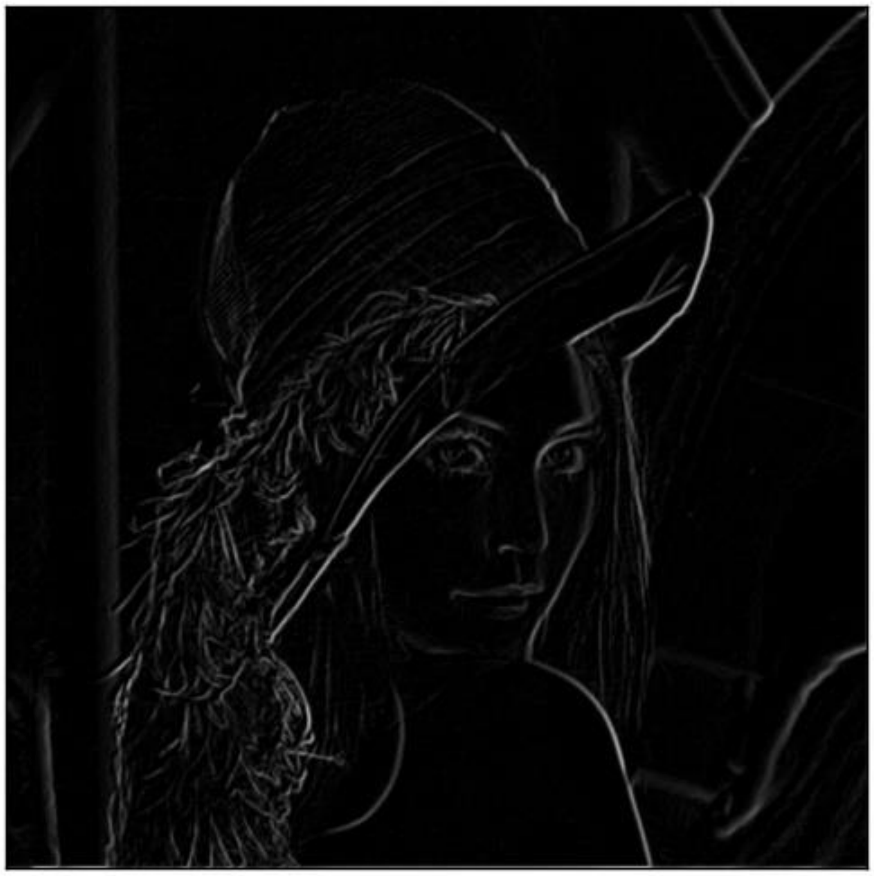
<u>Applying the x-gradient filter:</u>



Applying the y-gradient filter:



Finally, taking the magnitude of the above two images to form a composite image:



As can be seen, the composite image has the edges (that are inclined towards x and/or y axes), prominently displayed.

b. 1D Convolution on Image

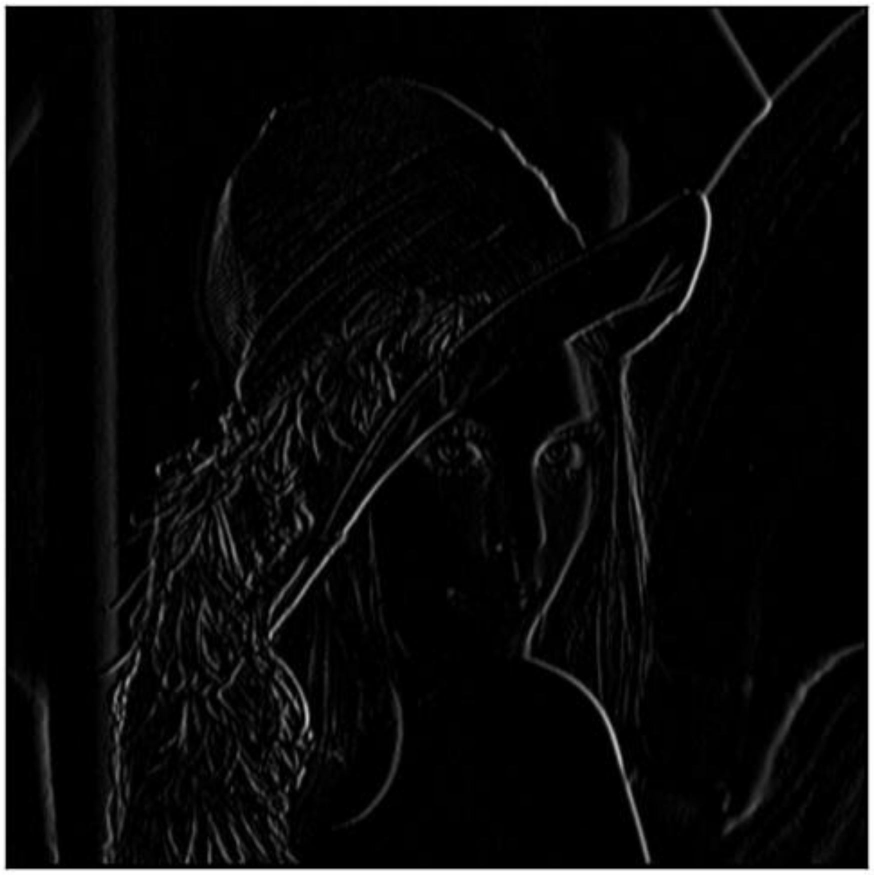
The effect of 2D convolution can be obtained by using linearly separable filter, where a single filter is split into two and the filters can be applied successively to the image giving us the same effect.

The linearly separable filters used were:

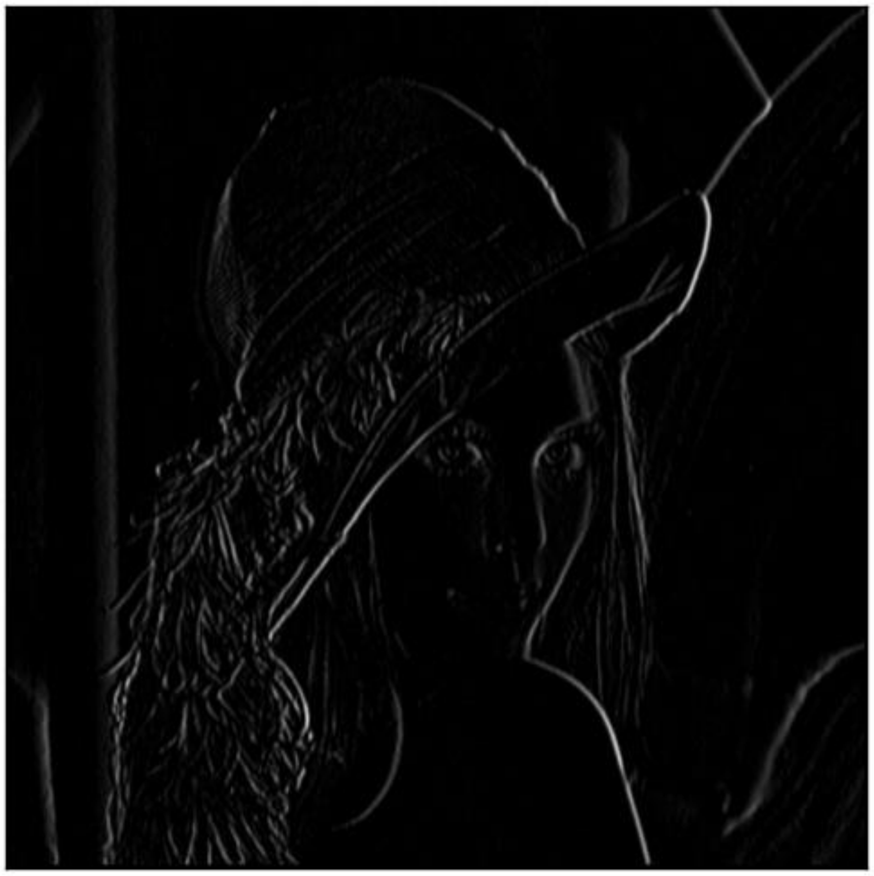
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

As can be observed, these are the same x and y gradient filters used before.

<u>The effect of applying the x gradient separated filters on the image is:</u>



<u>The effect of the applying the y gradient separated filters is:</u>



As can be seen by visual inspection, the 2D-convolution filter has the same effect as two successive 1D filters applied.

The average difference in data arrays for the images resulting from 1D and 2D convolution is less than 0.5, hence practically they produce the same effect.

c. Computational complexity of 2D filters vs linearly separable 1D filters

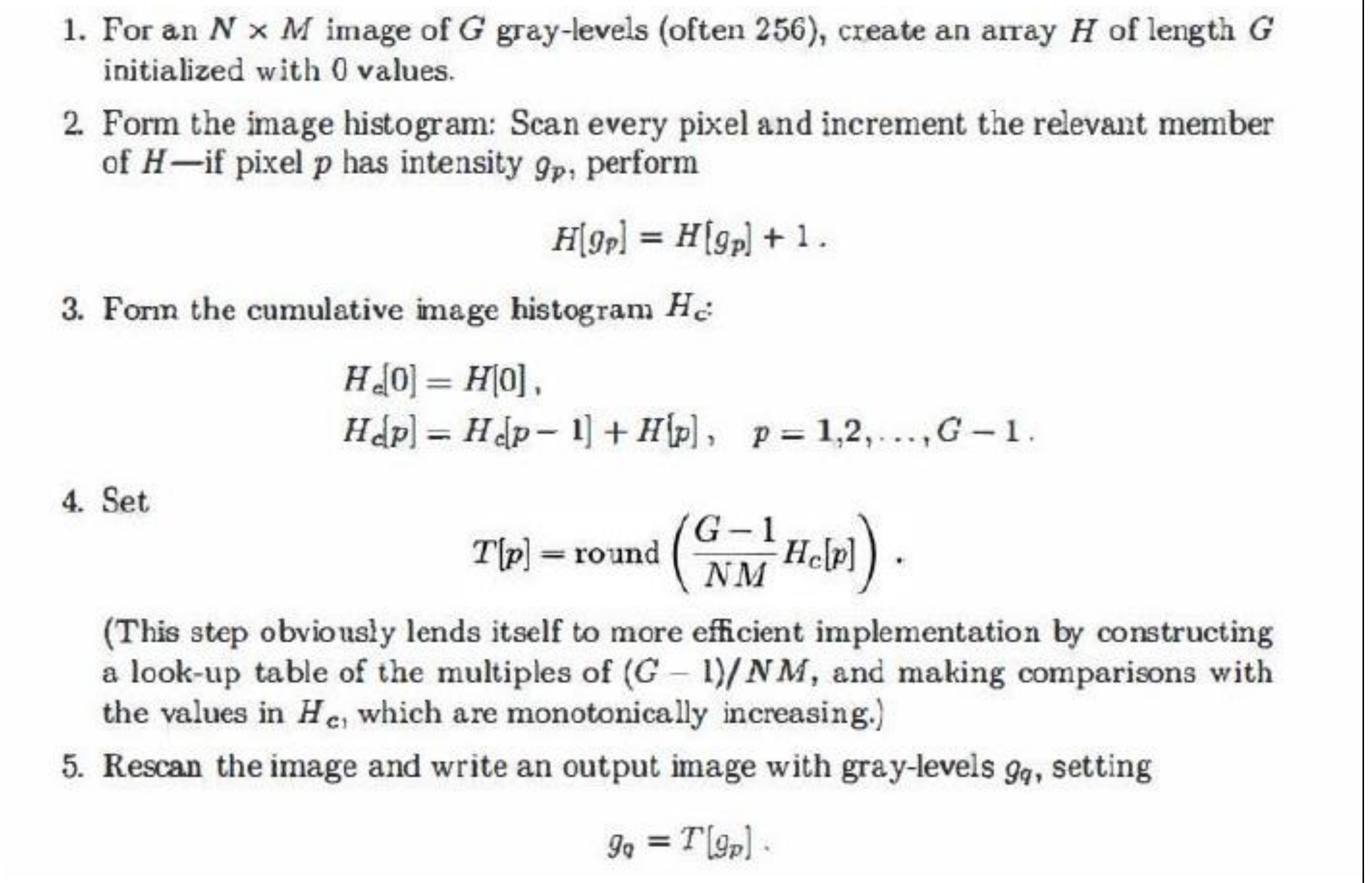
The advantage of separating a separated filter is that when we perform correlation with the matrices on the right (referring to the given filters), we don't need to actually perform any multiplications where there are zeros. In fact, it is more convenient to simply write these filters as 3x1 and 1x3, ignoring the zeros. This means that for each pixel in the image, we only need to perform 6 multiplications instead of 9.

In general, if we can separate a (M)x(N) filter into two filters that are (M)x1 and 1x(N), the total work we must do in correlation is reduced from (MxN)(PQ) to 2(N)(PQ).

2. Histogram equalization

Histogram equalization is a technique for adjusting image intensities to enhance contrast.

The algorithm provided for the image processing was:

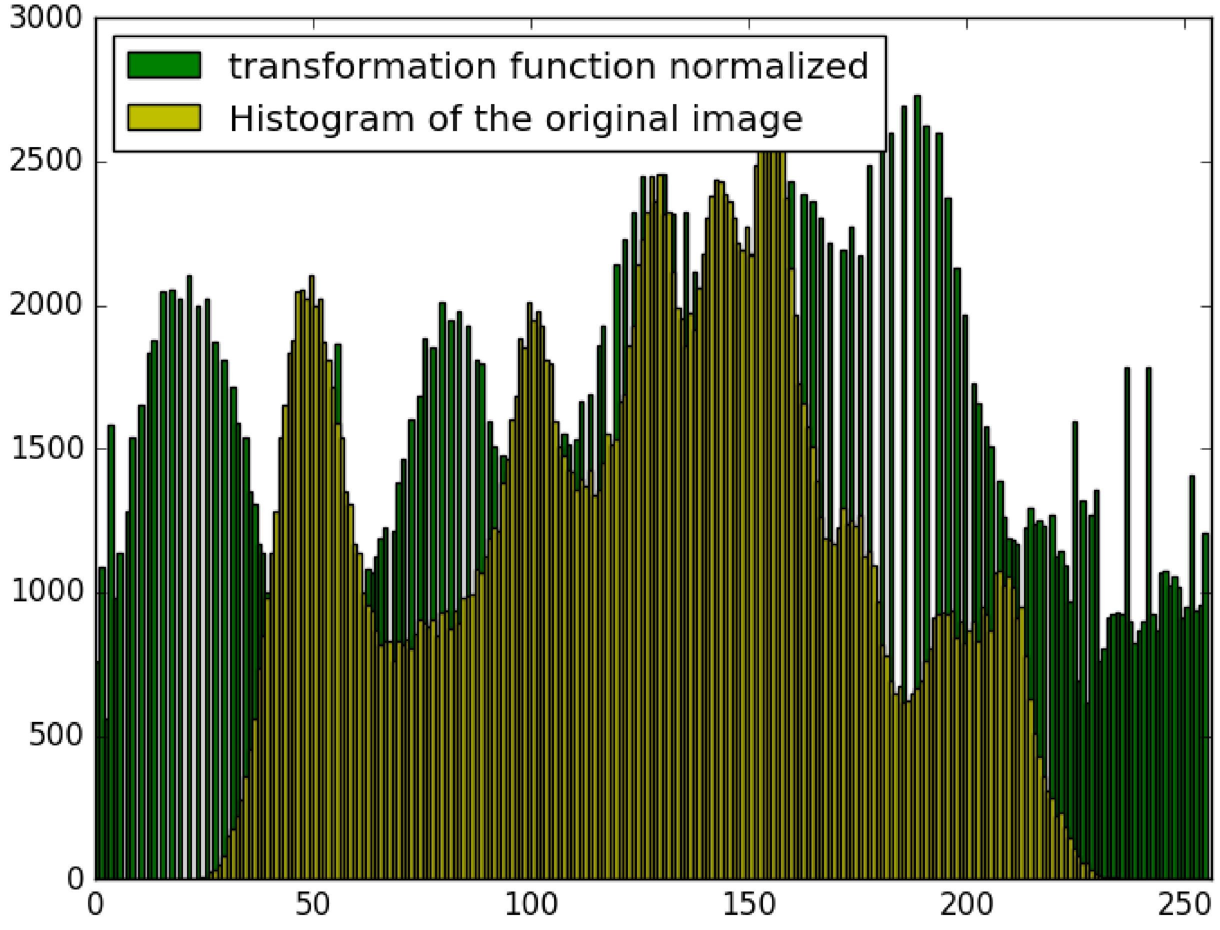


1. For an $N \times M$ image of G gray-levels (often 256), create an array H of length G initialized with 0 values.
2. Form the image histogram: Scan every pixel and increment the relevant member of H—if pixel p has intensity g_p, perform
$$H[g_p] = H[g_p] + 1 .$$
3. Form the cumulative image histogram H_c:
$$H_c[0] = H[0] ,$$
$$H_c[p] = H_c[p-1] + H[p] , \quad p = 1,2,\ldots,G-1 .$$
4. Set
$$T[p] = \text{round}\left(\frac{G-1}{NM} H_c[p]\right) .$$
(This step obviously lends itself to more efficient implementation by constructing a look-up table of the multiples of $(G-1)/NM$, and making comparisons with the values in H_c, which are monotonically increasing.)
5. Rescan the image and write an output image with gray-levels g_q, setting
$$g_q = T[g_p] .$$

This algorithm was followed closely, apart from the step 4 where the normalizing factor was changed.

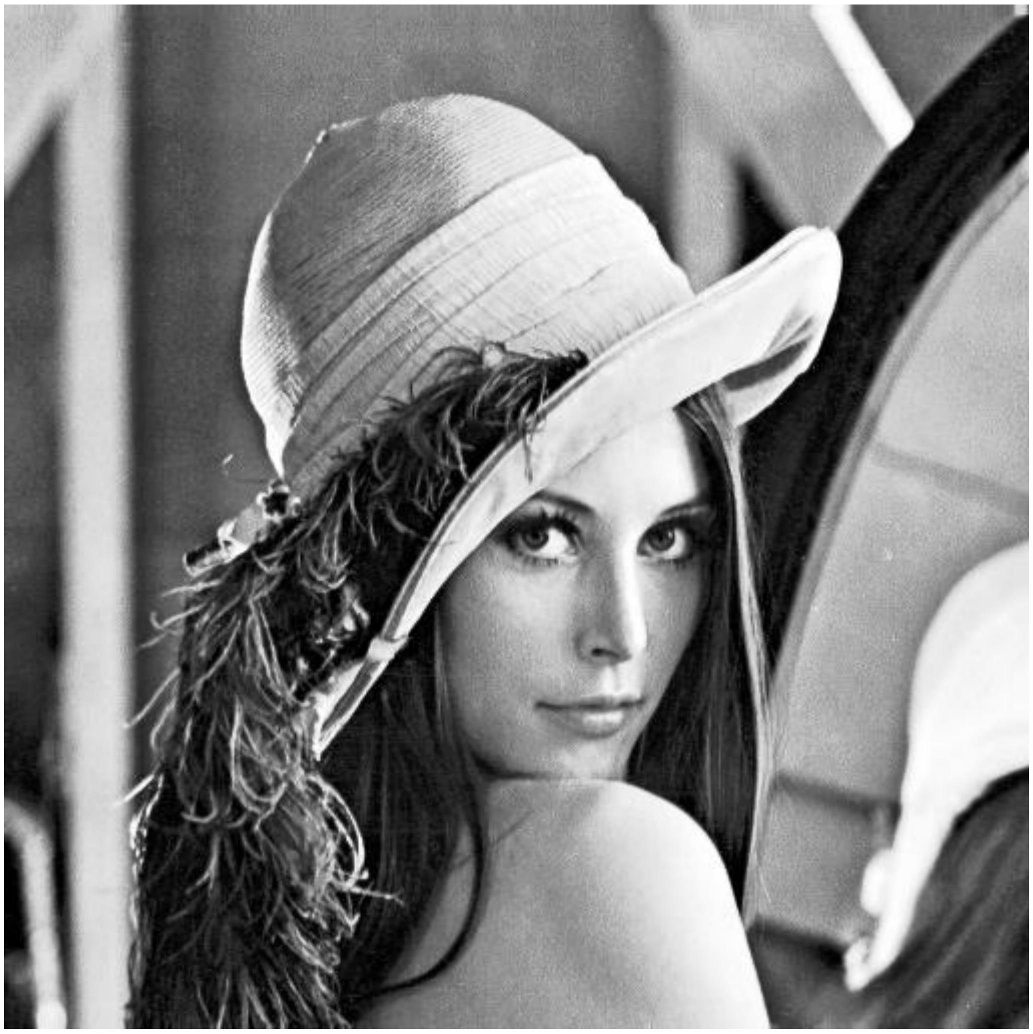
As expected, the histogram equalization enhanced the contrast of the images.

The plot of the histogram of the original image and the processed image is:



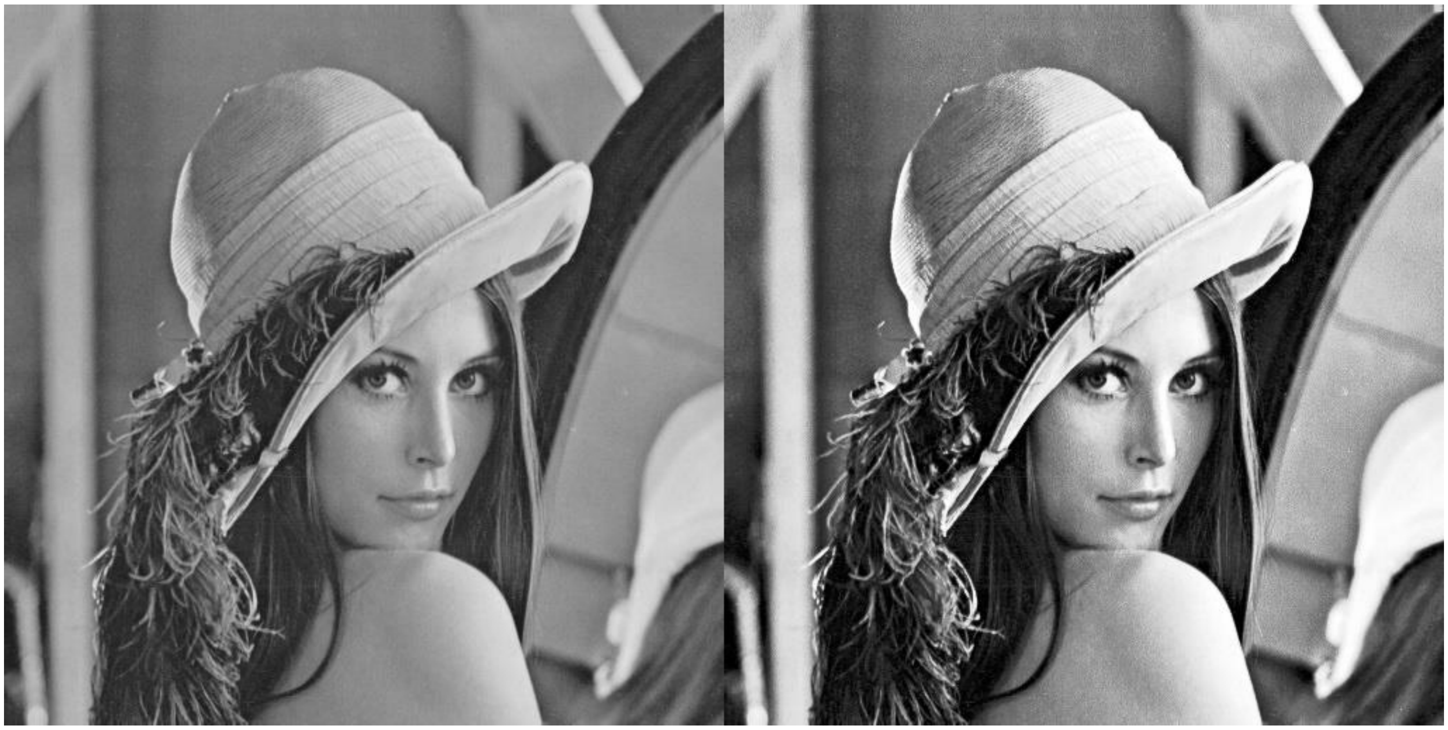
As can be observed, the transformed image's histogram is more evenly spaced out than that of the original image.

<u>The image after processing using the histogram equalization technique:</u>



For ease of comparison, the images are placed side-by-side:

The image on the left is the original image, the one on the right is the processed image. The increase in contrast is clearly evident from the comparison.



3. References:

a. https://en.wikipedia.org/wiki/Sobel_operator
b. http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm
c. http://juanreyero.com/article/python/python-convolution.html
d. http://www.songho.ca/dsp/convolution/convolution2d_example.html
e. https://en.wikipedia.org/wiki/Histogram_equalization

4. Code:

The filters were implemented in python using functions available in numpy and opencv.

The code has been submitted in the form on ipython notebook for readability.