Raghavendran Vijayan

CSCI55700 Project 2

Low level image processing

March 12, 2017

1 Introduction

In this project, I got a chance to learn and implement several low level image processing techiques. I have used MATLAB as the development tool to implement them. As per the project requirements, I have framed forumulas according to the formula given in the class lectures.

2 Functions Implemented

2.1 Histogram Equalization

The function aims at improving the contrast in the input image. It does so, by distributing the gray pixels to make it flat. So,It is also referred by histogram Flattening.
As result, We can infer more details on the darker regions of the given input image.

2.2 Log Transformation

Log mapping can be used to infer details on the darker regions by contrast stretching.
The implementation of the function is explained in the upcoming section of this report.

2.3 Rotation by some θ

It's simply rotationg the image with some angle,θ. The positive θ denotes, It is going to be rotated counte-clockwise. The negative θ denotes vice versa.

2.4 Gaussian Averaging Filter

It is handy in the situations to blur the image adn rove the noise. It is acheived by performing convolution. It has the following form

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The degree of maing it blur depends on the standard deviation of the Gaussian.

2.5 Median Filter

It is one of the ways to remove the noise in the image. One benefit of using this filter would be preseving the edges while removing noise.
It does so by replacing the pixels with the median of the surronding pixels.

3 Implementation

3.1 Histogram Equalisation

1. Input image is read and its color type is determined.
2. If it is found to be truecolor, We are converting it into grayscale image using inbuilt function, rgb2gray.
3. Total number of pixels are calculated by multiplying number of rows and number of columns.
4. First We are finding the number of occurance of all the possible intensity values.
5. The probability density function for all the intensities are computed and the cummulative density function of all the intensity values are also calculated.
6. Multiply the cummulative density function with the intensity value 255.
7. Adding one to the final matrix gives us the flattened image.

3.2 Log Transformation

1. Input image is read first
2. Transform the every pixel vlaue with double precision.
3. Add 1 to the matrix, so that comting erraneous log of 0 will be avoided.
4. Update matix values are multiplied with a general constant 0.1 and the log of the pixels.
5. They are converted back to gray image

3.3 Rotation by θ

1. Read the input image and find the number of rows,columns and pages present.
2. The end user's desired angle of rotation is dynamically get using console. And the corresponding radian for the angle is computed
3. The central values of the original and final images are calculated
4. The rotation matrix is computed using the genral form,

$$R = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \tag{1}$$

5. Adjusting the rotation matrix witht the found central values of the original image.
6. Checking the uncovered regions of the image and mark it with zero, to avoid holes in the image.

3.4 Gaussian Averaging Filter

1. Input image is read and its color type is determined.
2. If it is found to be truecolor, We are converting it into grayscale image using inbuilt function, rgb2gray.
3. I have implemented salt nd pepper, gaussian and poisson noise. The user's preffered noise is get and the noise will be added to the image.
4. Also, the user's desired standard deviation is also taken. and variance is computed.
5. The mask window which is thrice that of rounded σ is calculated.
6. The kernal value which is,

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

   is computed.
7. The final image padded with as many window size zeros on all the sides of the noisy image is computed.
8. Apply convolution, by multiplying the appropriate row and column values with the keranal value computed earlier.
9. The final image is shown as the gaussian filtered image.

3.5 Median Filter

1. Input image is read and its color type is determined.
2. If it is found to be truecolor, We are converting it into grayscale image using inbuilt function, rgb2gray.
3. I have implemented salt nd pepper, gaussian and poisson noise. The user's preffered noise is get and the noise will be added to the image.
4. Create a new matrix by padding the noisy image matrix with zeros on all sides.
5. Start from the second row and second column till the penultimate row and column, compute the median values of all the surrounding matrix values and store the value in the new matrix.
6. The newly framed matrix will be the final filtered image.

4 Results and comparisions

4.1 Histogram Equalisation tested on auto.pnm

Figure 1 shows the transformed histogram equalised image and comparision of histrograms of the original and flattened image using inbuilt and our own function.

We can see the contrast getting increased between the images and gray levels getting distributed to some extent.

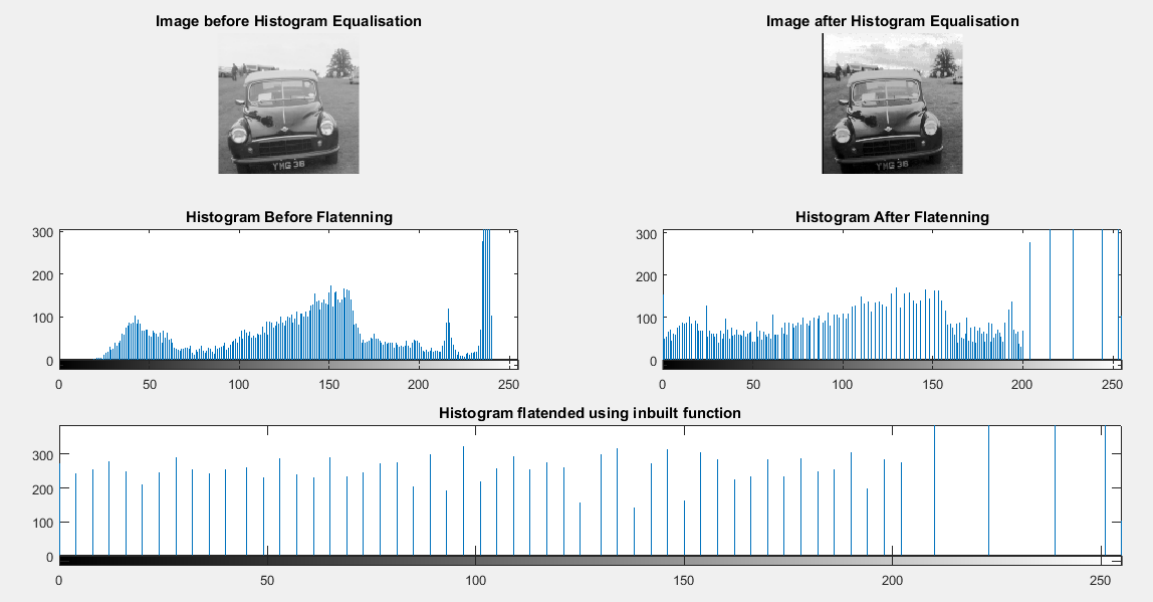

Figure 1: Histogram Comparision

Though It involves lot of operations, It yields some good performance in general, as that of inbuilt functions(histeq).
Also, It achieves better quality images in all test cases.

4.2 Log Transformation tested on cameraman.tif

Figure 2 shows the comparision between original and log transfomed image.

There are good details that can be inferred from the image in the darker regions of the original image.
More details about the background(building, tunnel) are visible.

4.3 Rotation by θ tested on child.pnm

Figure 3 and 4 shows the rotated original image by some θ and -θ.
The angle to be rotated is get from the user.
I feel, Using our code yields the same result as that of using imrotate.
They both resemeble each other. But, using our function, the edges corresponding to the original images, on rotation, were sometimes not continous.

4.4 Gaussian Averaging Filter tested on tire.pnm

Following 5 and 6 shows the filtered image which was added with salt and pepper noise and gaussian noise.
Another comparision with gaussian noise added is as follows.

I have observed, as the standard deviation gets increases, the degree of smoothness gets increased and vice versa.

It is a low pass filter.

Compared to median filter, It is effective in removing gaussian noise.

Figure 2: Log Mapping

I have used The 3 sigma rule to determine the mask size. In our case, I have kept mean as 0.

Figure 3: Rotation through 45 degree

Figure 4: Rotation through - 45 degree

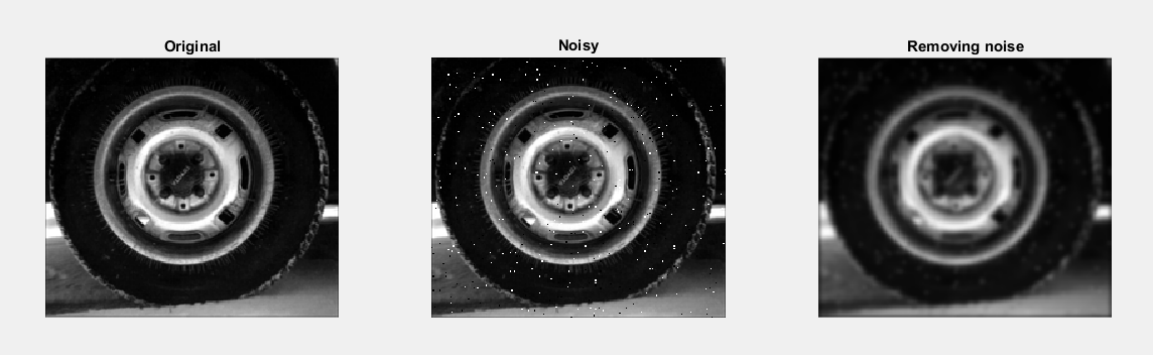

Figure 5: After filtering salt and pepper noisy image with density 0.01 and standard deviation for the filter 1.76

5 Median Filter tested on auto.pnm

Figure 7 and 8 and shows the median filtered image with salt and pepper noise and poisson noise added.

In the last figures, all the corner pixels will be black(zero). Because, I have used an array that is created by padding the input array with zeros on all the sides. For any corner image, If we pick the 9 neighbourhood points, minimum 5 points will be zero. The median in the case will be zero.

It is not linear by operation.

The great advantage of this filter is it preserves the edges.

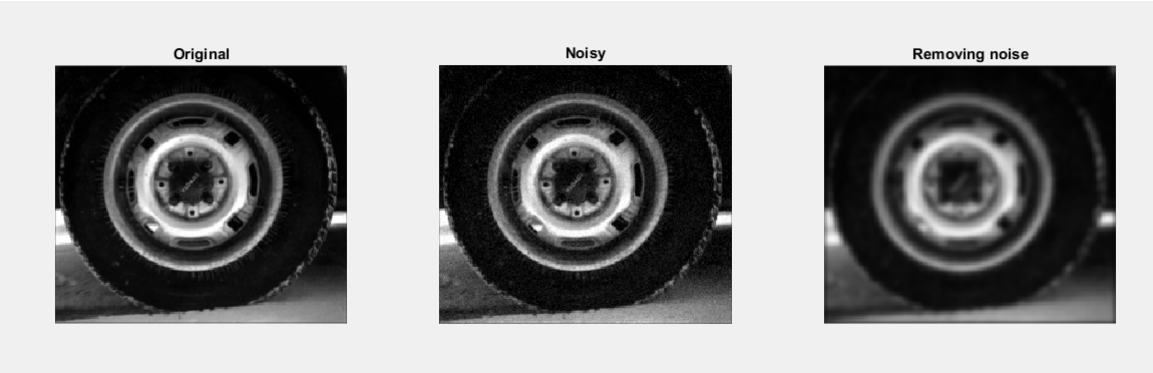

Figure 6: After filtering gaussian noisy image with mean 0 and variance 0.001 and standard deviation for the filter 2

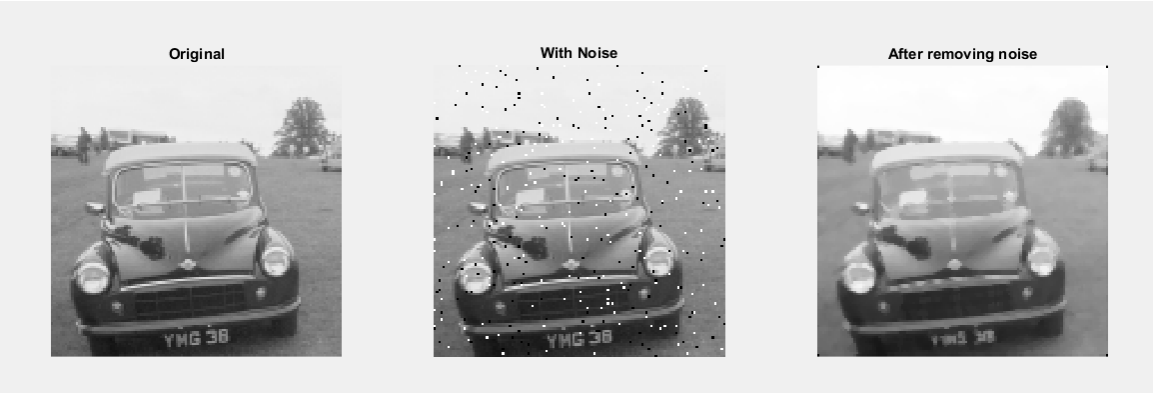

Figure 7: After filtering salt and pepper noisy image with density 0.02

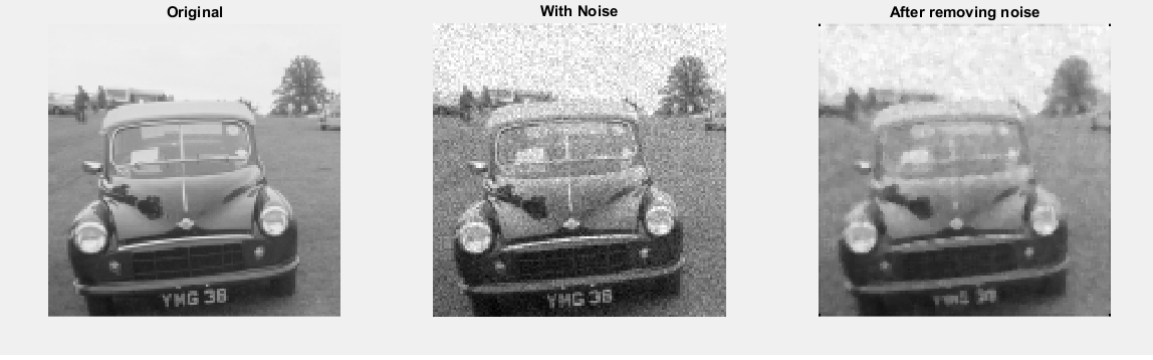

Figure 8: After filtering poisson noisy image