Project-02

Image Processing | Instructor: Dr. Mihran Tuceryan

Submitted By: Kishan K. Ramoliya | kramoliy@umail.iu.edu

Histogram Equalization

Introduction and Description:

Histogram equalization is a method using which we can adjust image intensities that will enhance the contrast of the image. By applying the histogram, the intensities of the image can be distributed properly which will allow the area with lower contrast to gain proper higher contrast.

Given below is the algorithm to perform the histogram equalization:

- First of all find the frequency of each and every pixel value from the image.
- After that we are supposed to find the probability of each pixel value frequency.
- Once we have obtained the probability of each pixel frequency, we have to find the cumulative histogram of each of them, and further find the cumulative distribution of each pixel value.
- Now perform the multiplication of this cumulative distribution with the number of bins which is 255 here and replace them with the original values.

Results: (With image BC.jpg)

- Input and Output images (original image).

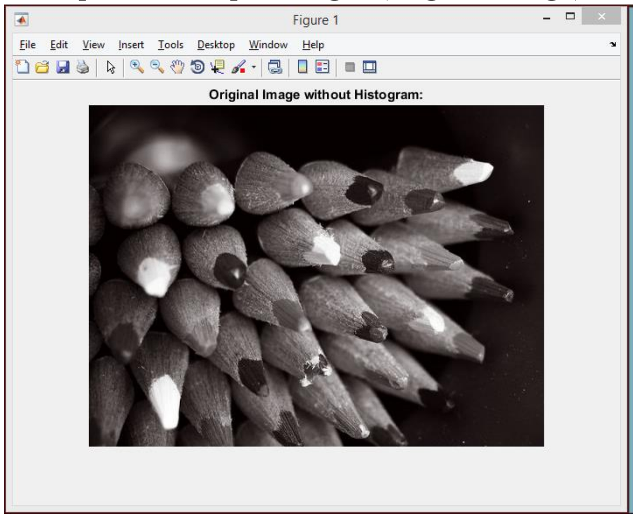


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image without Histogram:

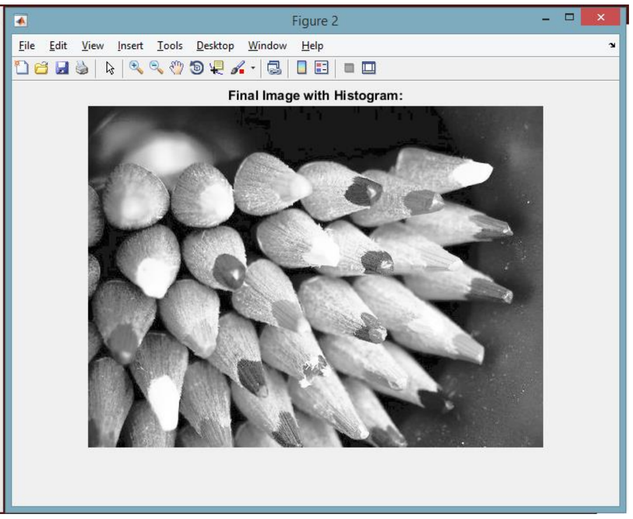



Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image with Histogram:

- Input and Output images with Salt & Pepper noise (0.04 density).

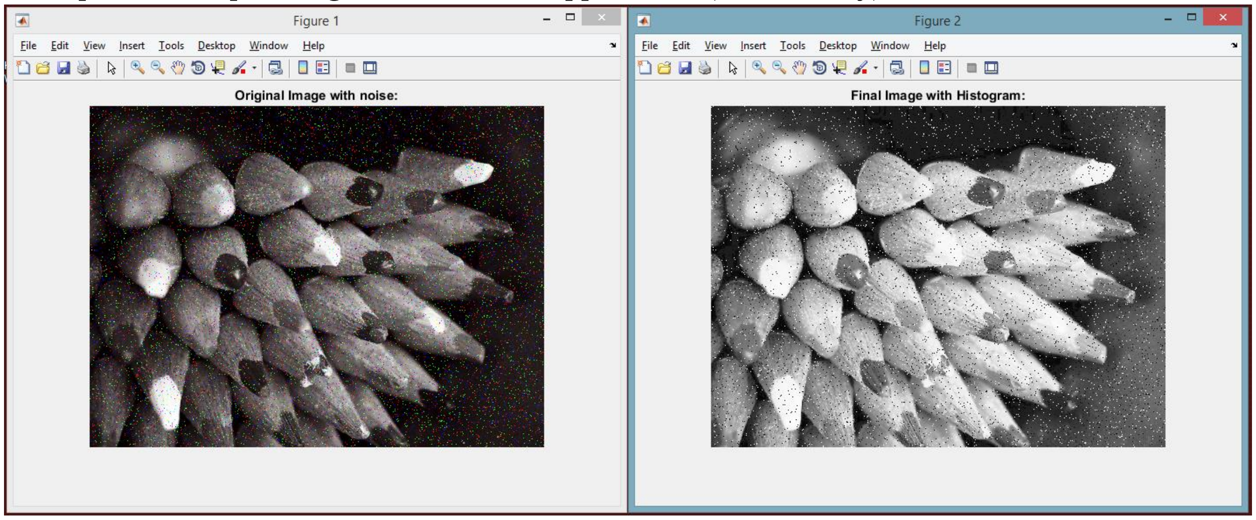


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image with noise:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image with Histogram:

- Input and Output images from inbuilt function of MATLAB with (0.04 density) and without noise.

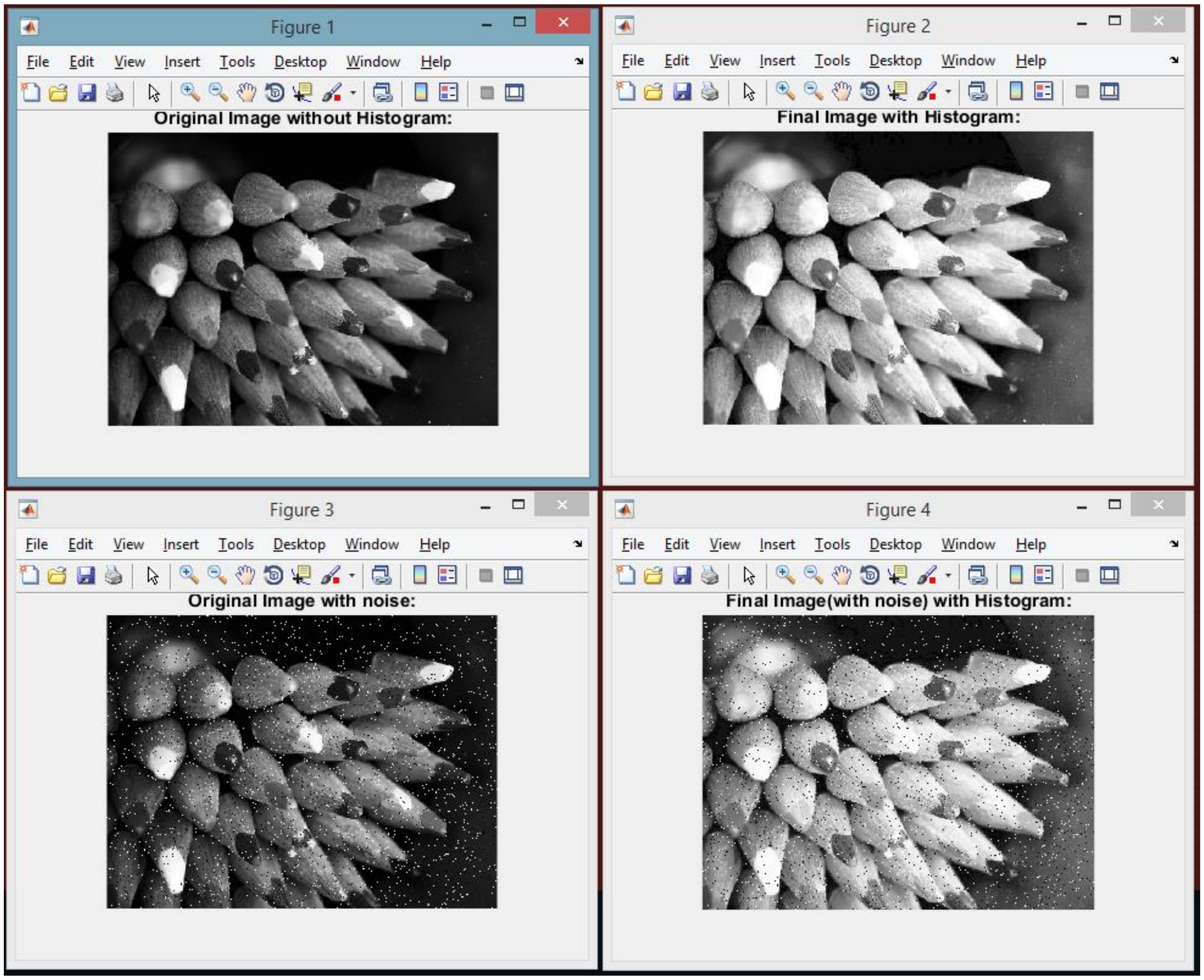


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image without Histogram:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image with Histogram:
Figure 3
File Edit View Insert Tools Desktop Window Help
Original Image with noise:
Figure 4
File Edit View Insert Tools Desktop Window Help
Final Image(with noise) with Histogram:

Comparison:

From the above outputs we can observe that the inbuilt function of the histogram and the created function works properly and the results are almost same. Also I have tested the function by inserting some salt and pepper noise to the image and it works in the same manner as well.

Analysis and Application:

From the above figure we can observe that some of the darker regions in the image are made lighter and the intensity is equally distributed in the image.

In real world some lower intensity area may create problem during the visual detection, so by applying the histogram the problem with lower intensity can be removed which intern will improve the performance of the visual detection of the system.

Logarithmic Mapping

Introduction and Description:

To compress the dynamic range of an image we can replace each and every pixel value with their log values. By implementing this, we enhance the low intensity pixels of the image.

Logarithmic mapping is a simple point process in which the mapping function is a log curve. Since the logarithm is not defined for, the equation of the log mapping function looks like given below:

$I'(r,c)=\log(I(r,c)+1)$

Results: (With image building.pnm)

- Input and Output images (original image).

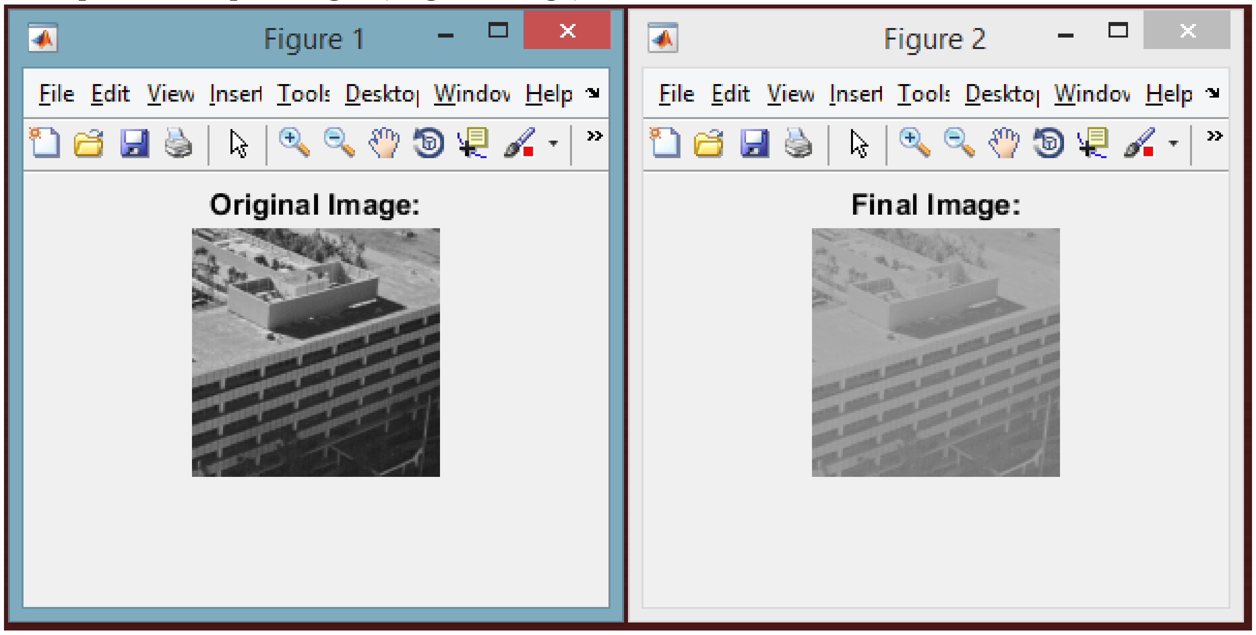


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:

- Input and Output images with Salt & Pepper noise (0.04 density).

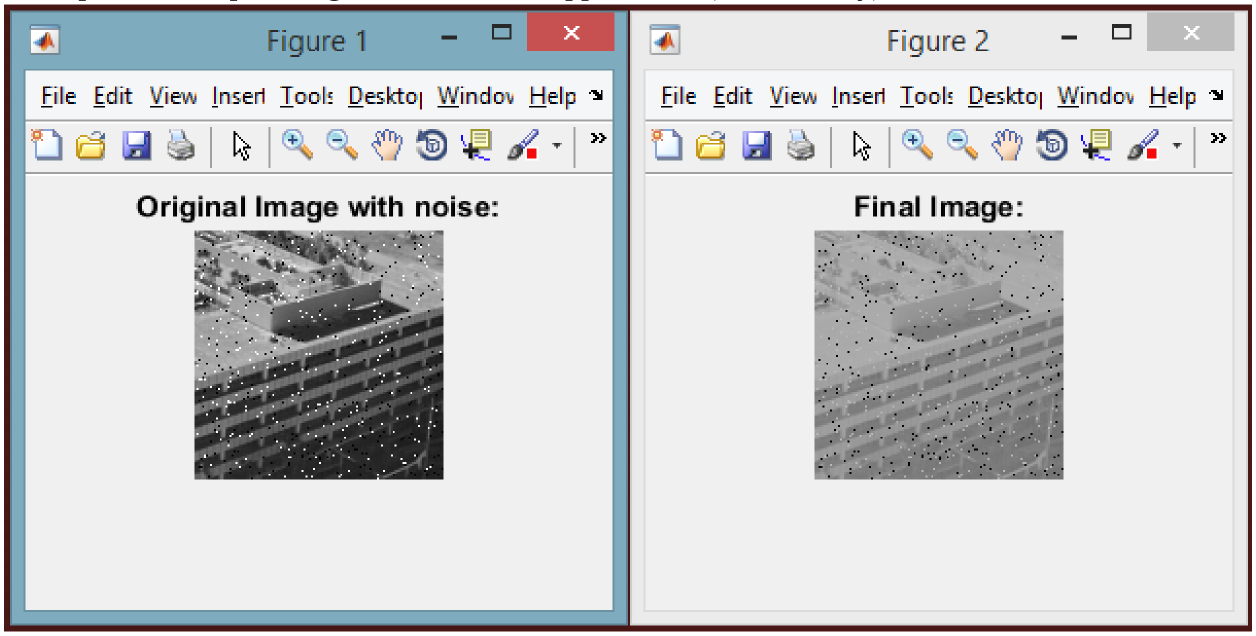


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image with noise:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:

Comparison:

From the above outputs we can observe that the function of the log mapping works properly. Also I have tested the function by inserting some salt and pepper noise to the image and it works in the same manner as well.

Analysis and Application:

From the above figure we can observe that some of the lower image intensity is enhanced by implementing the log function.

In real world we can use this function where the dynamic range of the image is too large that it cannot be displayed on the normal screen. Also this can be used to display the Fourier transform.

Image Rotation

Introduction and Description:

Rotation is the operation that will perform the geometric transformation of the image element which will be rotated at the given angle about an origin.

The rotation of the image element is performed in the given below manner:

$$x' = \cos(\theta) * (x - x_0) - \sin(\theta) * (y - y_0) + x_0$$

$$y' = \sin(\theta) * (x - x_0) - \cos(\theta) * (y - y_0) + y_0$$

(Equation taken from - http://homepages.inf.ed.ac.uk/rbf/HIPR2/rotate.htm)

Here, *(x', y')* are new coordinate of the image element, *(x, y)* are coordinate f image element to be rotated, *(x0, y0)* is the origin and θ is the angle with which the image is to be rotated.

Results: (With image ct_scan.pnm)

- Input and Output images (original image) with angle -30.

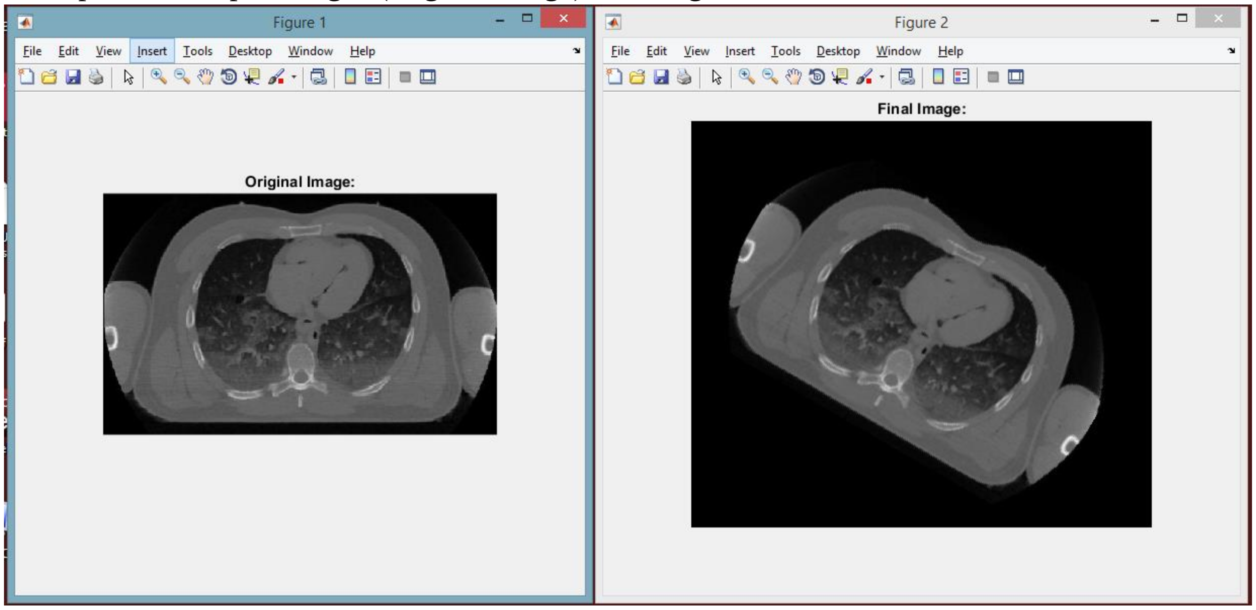


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:

- Input and Output images with Salt & Pepper noise (0.04 density), angle 45.

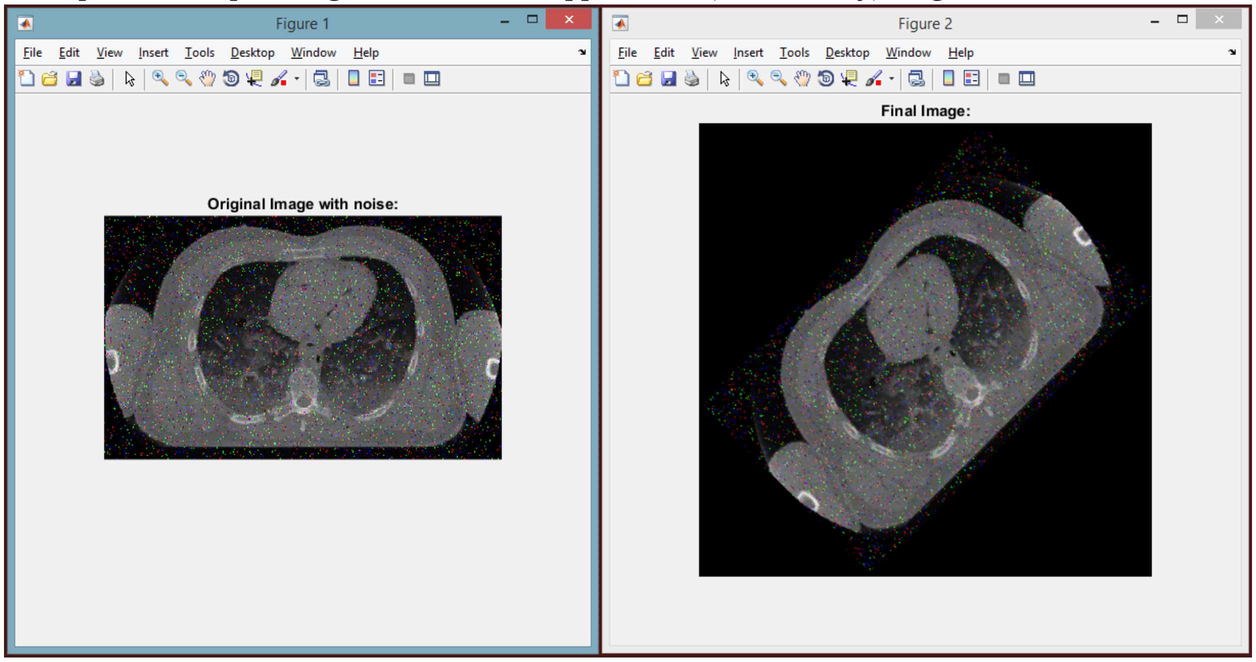


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image with noise:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:

- Input and Output images from inbuilt function of MATLAB with (0.04 density) and without noise, angle 30.

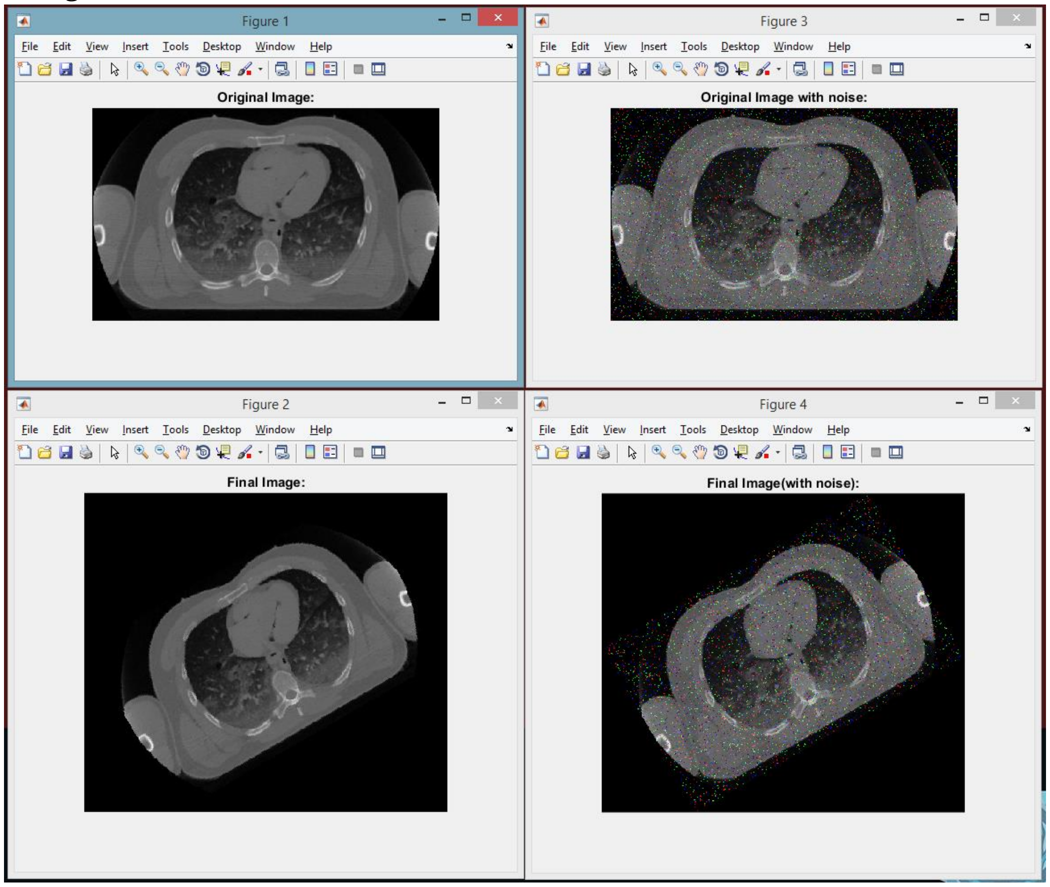


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image:
Figure 3
File Edit View Insert Tools Desktop Window Help
Original Image with noise:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:
Figure 4
File Edit View Insert Tools Desktop Window Help
Final Image(with noise):

Comparison:

From the above outputs we can observe that the inbuilt function of the image rotation and the created function works properly and the results are almost same. Also I have tested the function by inserting some salt and pepper noise to the image and it works in the same manner as well.

Analysis and Application:

From the above figure we can observe that the image is rotated at the desired given angles.

In real world rotation of the image can be used to improve the visual of the image and also as image or visual preprocessor where the direction operations are important like navigation.

Gaussian Averaging Filter

Introduction and Description:

Gaussian filter have a special type of property of not having any overshoot towards a step input function and it minimizes the fall and rise time, thus it has minimum delay in group. Due to this property the Gaussian filter is considered the ideal time domain filter.

The equation of the 2D Gaussian function is given below:

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

(Equation taken from https://en.wikipedia.org/wiki/Gaussian_filter)

Also it involves some steps for convolution too.

Results: (With image auto.pnm)

- Input and Output images (original image) with sigma 1.22.

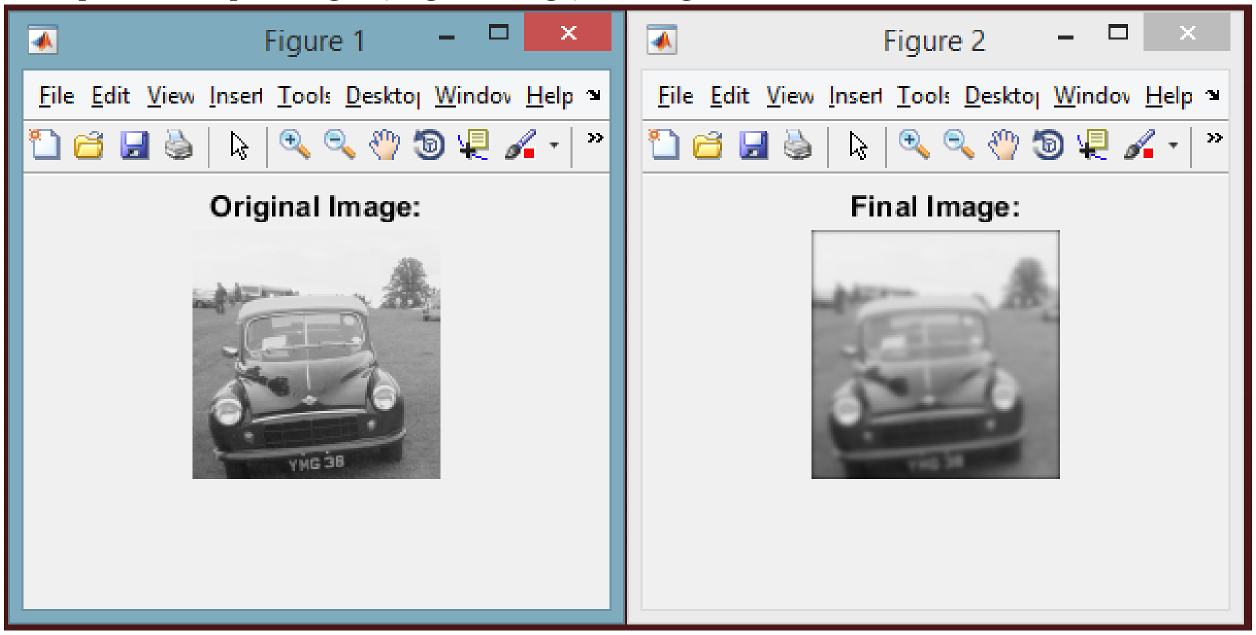


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image:
YMG 38
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:

- Input and Output images with Salt & Pepper noise (0.04 density), sigma 1.3.

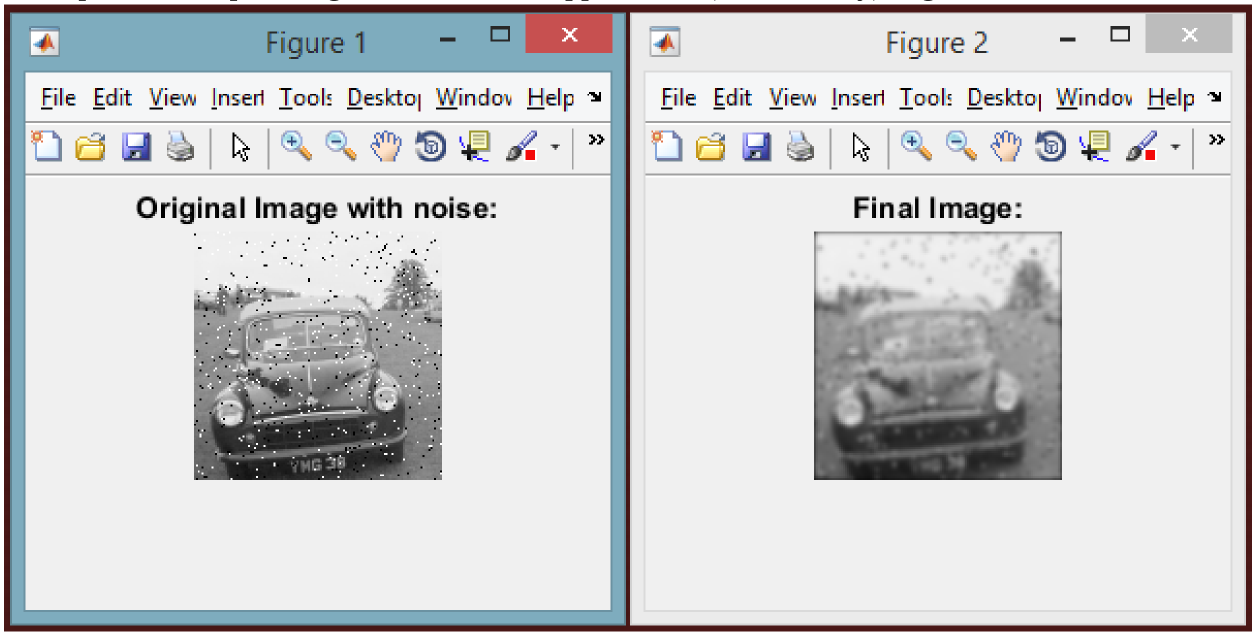
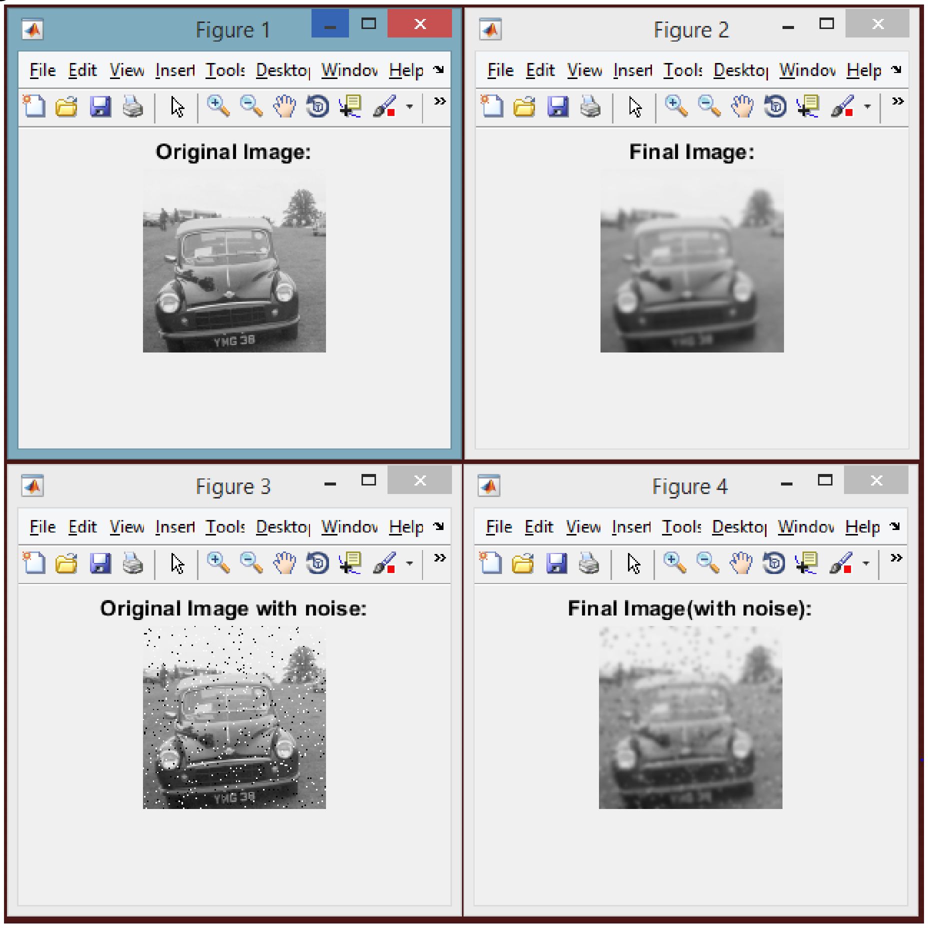


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image with noise:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:

- Input and Output images from inbuilt function of MATLAB with (0.04 density) and without noise, sigma 1.3.



Comparison:

From the above outputs we can observe that the inbuilt function of the Gaussian filter and the created function works properly and the results are almost same. Also I have tested the function by inserting some salt and pepper noise to the image and it works in the same manner as well.

Application:

Gaussian filter is mainly used in the process of Canny Edge Detection method to detect the edges in the image.

Median Filter

Introduction and Description:

Median filter is basically used to remove the nonlinear noise in the images. Generally this is used in the preprocessing step in different image processing algorithms. The most significant reason to use the median filtering is that it preserves the edge information even after removing the nonlinear noise.

The main idea in the median filter is to replace the image element by the median of their neighboring entries.

Results: (With image auto.pnm)

- Input and Output images (original image).

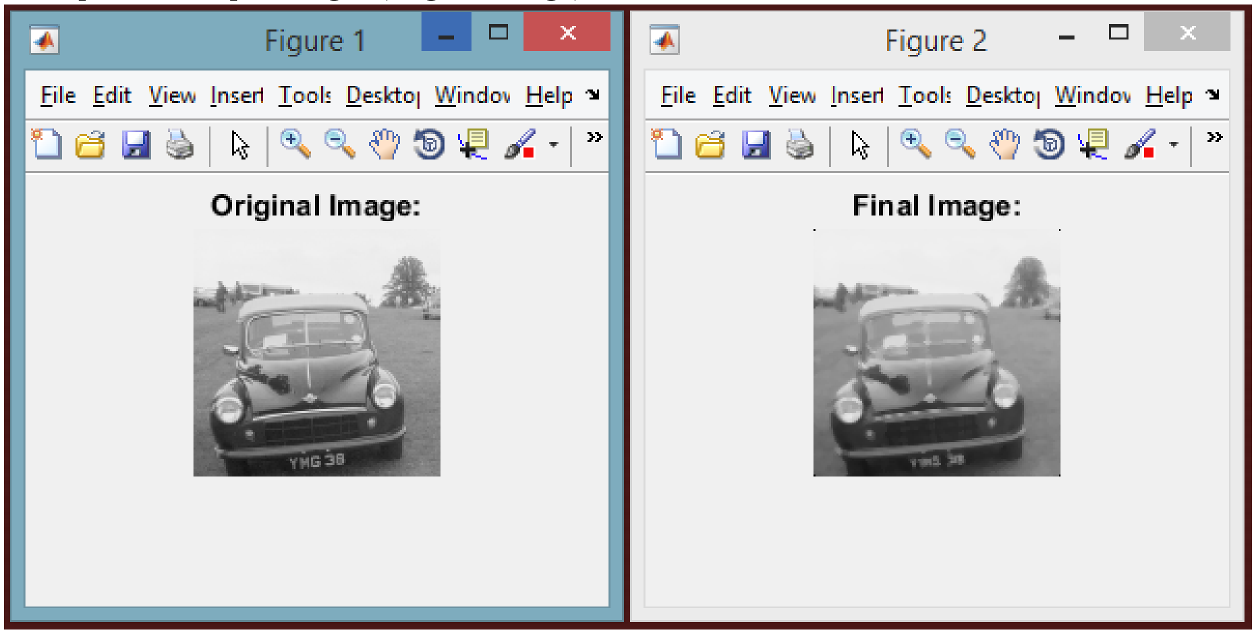


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:

- Input and Output images with Salt & Pepper noise (0.04 density).

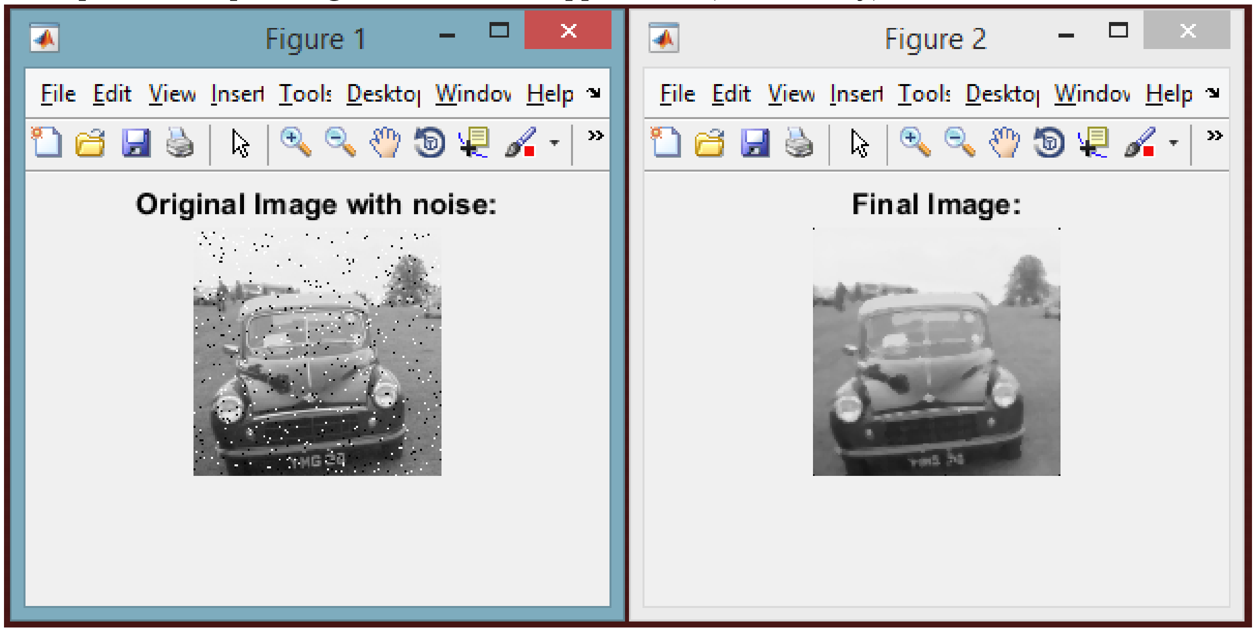
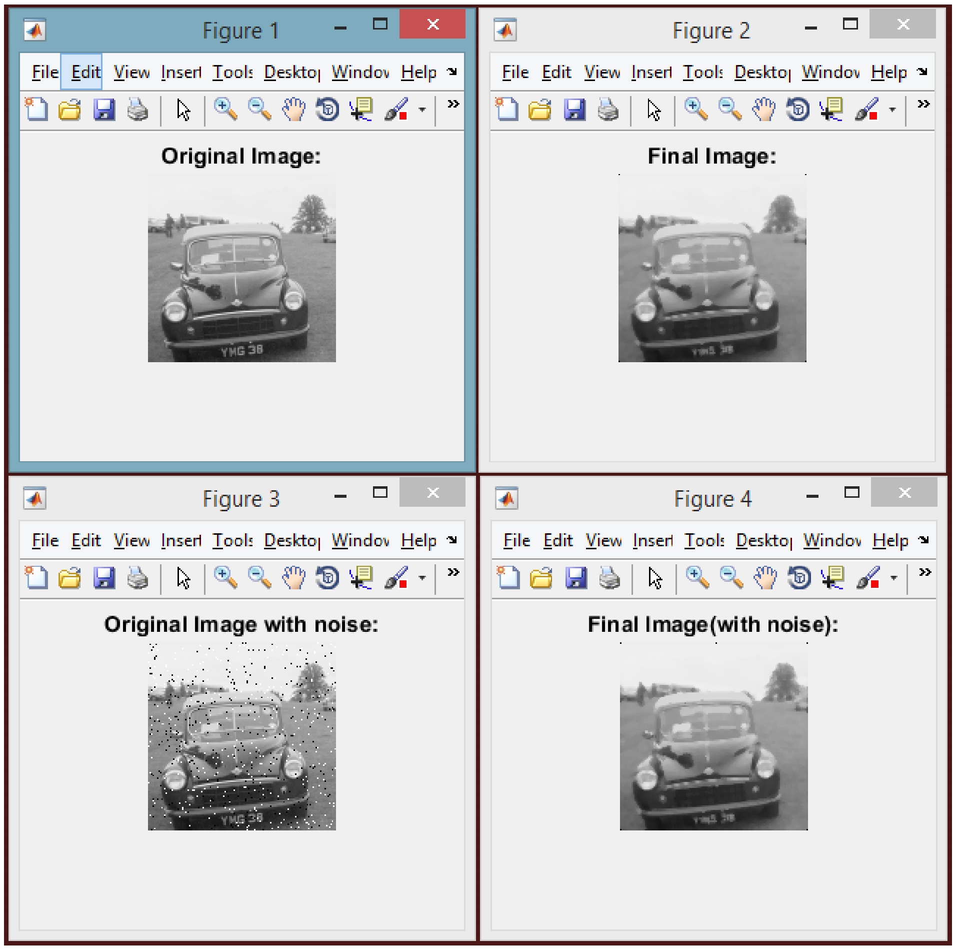


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image with noise:
Figure 2
File Edit View Insert Tools Desktop Window Help
Final Image:

- Input and Output images from inbuilt function of MATLAB with (0.04 density) and without noise.



Comparison:

From the above outputs we can observe that the inbuilt function of the median filter and the created function works properly and the results are almost same. Also I have tested the function by inserting some salt and pepper noise to the image and it works in the same manner as well.

Analysis and Application:

From the above figure we can observe that the nonlinear noise in image are removed and the edge properties of the image is also preserved.

As mentioned in the previous section the median filter is basically used as preprocessing step through which we can remove the unwanted nonlinear noises and also preserve the edge information.

References:

- https://en.wikipedia.org/wiki/Histogram_equalization
- http://www.math.uci.edu/icamp/courses/math77c/demos/hist_eq.pdf
- http://www.mathworks.com/help/images/ref/histeq.html?requestedDomain=www.mathworks.com
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixlog.htm
- http://www.tutorialspoint.com/dip/gray_level_transformations.htm
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/rotate.htm
- http://www.datagenetics.com/blog/august32013/index.html
- http://www.mathworks.com/help/images/ref/imrotate.html
- https://en.wikipedia.org/wiki/Gaussian_filter
- http://www.mathworks.com/help/images/ref/imgaussfilt.html
- https://en.wikipedia.org/wiki/Median_filter
- http://www.mathworks.com/help/images/ref/medfilt2.html
- Lecture slides by Dr. Mihran Tuceryan.

Note:

- All the functions and filters are implemented using MATLAB as it provides much easy and simpler interface for image processing.
- For each task to be performed, I have created different MATLAB functions.
- All the functions were tested on all the images mentioned in the project document like auto, building, tire, child, ct_scan, but I have showed few results only for the sake of simplicity. I have also used one image from interned named BC just for testing.
- Following is the list of files:
  - **Histogram Equalization:**
    - HistogramEqualization.m
    - HistogramEqualizationWithNoise.m with Salt & Pepper noise (0.04 density).
    - HistogramInbuilt.m
  - **Log Mapping:**
    - LogMapping.m
    - LogMappingWithNoise.m with Salt & Pepper noise (0.04 density).
  - **Image Rotation:**
    - ImageRotation.m
    - ImageRotationWithNoise.m with Salt & Pepper noise (0.04 density).
    - ImageRotationInbuilt.m
  - **Gaussian Filter:**
    - GaussianFilter.m
    - GaussianFilterWithNoise.m with Salt & Pepper noise (0.04 density).
    - GaussianFilterInbuilt.m
  - **Median Filter:**
    - MedianFilter.m
    - MedianFilterWithNoise.m with Salt & Pepper noise (0.04 density).
    - MedianFilterInbuilt.m
- Also for the Image Rotation and Gaussian Filter functions expects the input of angle to which image is to be rotated and the sigma respectively.