CSE585: Digital Image Processing II
Computer Project 3
Nonlinear Filtering and Anisotropic Diffusion

Aishwarya Mallampati, Wushuang Bai, Mrunmayi Ekbote,

Date:03/28/2020

---

1 Objectives

This project aims to familiarize with two techniques of digital image processing namely nonlinear filtering and anisotropic diffusion which have the capability to remove noise while trying to keep the important features in a given image. The main objectives of the project include the following:

- To apply mean, median, alpha trimmed mean, sigma and symmetric nearest neighbor mean nonlinear filters on a given image and observe their performance with respect to removing noise and preserving edges in an image.
- To implement anisotropic diffusion for two sets of images.

2 Methods

2.1 Nonlinear Filtering

2.1.1 Theory and Algorithms

Non-linear filters are not based on linear relationships between an input and an output via a system function. Instead they represent a much broader class of operations. Nonlinear filters find application in a range of image processing and coding applications such as: denoising, edge preserving operations and in some forms of prediction. The following are the definitions of nonlinear filters used in this project:

- Mean Filter: The mean filter is a simple sliding-window spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window. The window, or kernel, is usually square but can be any shape.

$$y_k = \frac{1}{2N+1} \sum_{i=1}^{2N+1} x_{k-N+i-1} \tag{2.1}$$

  where (2N+1) is the length of the window

- Median Filter: The median filter is also a sliding-window spatial filter, but it replaces the center value in the window with the median of all the pixel values in the window. As for the median filter, the kernel is usually square but can be any shape.

- Alpha-Trimmed Mean Filter: The alpha-trimmed mean filter is based on order statistics and varies between a median and a mean filter. It is used when an image contains both short and long tailed types of noise (e.g. both Gaussian and salt and pepper noise). To define the alpha-trimmed mean filter, 2N pixels on either side of the pixel x_k are first ordered from minimum to maximum value. Then the alpha-trimmed mean filter given in L12-5 is applied on the ordered set as mentioned in the eq2.2

$$y_k = \frac{1}{n - 2\lfloor \alpha n \rfloor} \sum_{i=\lfloor \alpha n \rfloor + 1}^{n - \lfloor \alpha n \rfloor} x_i \tag{2.2}$$

  where n=(2N+1) is the window length

- Sigma Filter: The idea of the sigma filter is based on the assumption that the noise in the image is distributed bell-shaped, i.e., Gaussian shaped. When using a monochrome image with lots of noise, the frequency of the intensity value to the left and right of the mean value also drop away in a bell-shaped curve, i.e., Gaussian curve.
  The Sigma filter is also identified as a neighborhood filter. It replaces the intensity value of the current pixel by the average of all of the intensity values whose distance is smaller than $(2 * Sigma)$ to the current intensity value(eq2.3)

$$y_k = \frac{1}{N_C} \sum_{i=-N}^{N} \delta_i x_{k-i} \tag{2.3}$$

$$\delta_i = \begin{cases} 1 & |x_{k-1} - x_k| \leq 2\sigma \\ 0 & otherwise \end{cases} \tag{2.4}$$

  where N_C is the number of points x_{k-i} having $\delta_i = 1$

- Symmetric Nearest Neighbor Mean Filter: The Symmetric Nearest Neighbor Mean Filter or Symmetric NNM filter is a non-linear edge preserving image processing filter. It is a very effective noise reduction technique that removes noise while maintaining sharp image edges.
  Similar to the other image processing filters, the Symmetric NNM filter works using a sliding window at each image pixel. At each position, the pixels under the window are split into pairs of opposite points. Each of these pairs are compared with the central pixel, the pixel closest to the central pixel is selected. The central pixel value is replaced by the average of all the selected pixels.

2.1.2 Matlab

All the code used for this problem is placed in Prob1_NonlinearFiltering folder. In order to perform nonlinear filtering in matlab, RandomDisks-P10.jpg is used as input image. Input image and filter window size = $(5 * 5)$ are passed as inputs to each filter used for this problem.
The following are the matlab files used for this problem:

- main_nonlinearfiltering.m: The following steps are performed in this file:

- Input image is read and it is padded with zeros of size(2,2).
- Mean, median, alpha-trimmed mean, sigma and symmetric nearest neighbor mean filters are applied on the input image for five times iteratively.
- Matlab inbuilt function medfilt2(A) is used to perform median filtering.
- Displays and stores the results after applying each filter.
- Histogram is displayed and stored for each filter result.
- Interior of large disk region is extracted by manually specifying the sub-region boundary values.
- Mean and standard deviation are computed for interior of large disk using matlab inbuilt functions mean2(A) and std2(A) respectively

- mean_filter.m: Implements mean filter algorithm specified by equation2.1
- alphatrim_filter.m: Implements alpha trimmed mean filter algorithm as specified by equation2.2. This filter also needs alpha value as input. ($\alpha = 0.25$ in this problem). Matlab inbuilt sort() function is used by this filter to sort the elements around x_k.
- sigma_filter.m: Implements sigma filter algorithm as specified by equation2.3. This filter needs sigma values as input($\sigma = 20$for this problem)
- symmNNM_filter.m: Implements symmetrical nearest neighbor algorithm as specified in section2.1.1

The results obtained are displayed in section3.
Note: Run main_nonlinearfiltering.m file in Prob1_NonlinearFiltering folder to get the results

2.2 Anisotropic Diffusion

2.2.1 Theory and Algorithm

Anisotropic diffusion, also called PeronaMalik diffusion, is a technique aiming at reducing image noise without removing significant parts of the image content, typically edges, lines or other details that are important for the interpretation of the image. The anisotropic diffusion equation is given as(eq:2.5b):

$$I_t = div[c(x,y,t)\nabla I] = \nabla.[c(x,y,t)\nabla I] \tag{2.5a}$$
$$= c(x,y,t)\nabla^2 I + \nabla c.\nabla I \tag{2.5b}$$

where c(x,y,t) is the conduction coefficient, to achieve piecewise smoothing c(x,y,t) is given as:

$$c(x,y,t) = \begin{cases} 1 & (a.)insideregions[conduct] \\ 0 & (b.)atregionboundaries(edges)[don'tconduct] \end{cases} \tag{2.6}$$

This results in two possibilities:
(a) smoothing occurs inside regions: $I_t = \nabla^2 I$
(b) smoothing inhibited at boundaries: $I_t = 0$ (no change)

Discrete(computer) implementation of Anisotropic Diffusion:

$$I_{i,j}^{t+1} = I_{i,j}^{t} + \lambda[c_N^t \nabla_N I + c_S^t \nabla_S I + c_E^t \nabla_E I + c_W^t \nabla_W I] \tag{2.7}$$

where $0 \leq \lambda \leq 0.25$
$\nabla_N I = I_{i,j+1}^t - I_{i,j}^t$, $\nabla_S I = I_{i,j-1}^t - I_{i,j}^t$, $\nabla_E I = I_{i+1,j}^t - I_{i,j}^t$, $\nabla_W I = I_{i-1,j}^t - I_{i,j}^t$
And the conduction coefficient for images is defined as:

$$c(x,y,t) = g(||(x,y,t)||) = g(\sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2}) \tag{2.8}$$

where $||(x,y,t)||$ is the norm(magnitude) of gradient of I
There are two choices for g(.):

- Exponential method: This method favours high contrast edges

$$g(||\nabla I||) = exp{-}(\frac{||I||}{k})^2 \tag{2.9}$$

- Inverse Quadratic Method: This method favors wide regions

$$g(||\nabla I||) = \frac{1}{1 + (\frac{||I||}{k})^2} \tag{2.10}$$

k can be any arbitrary value. But large k values are better for exponential g(.) and small values of k are preferred for inverse quadratic g(.).

2.2.2 Matlab

All the matlab code used for this problem is placed in Prob2_AnisotropicDiffusion folder. cwheelnoise.gif and cameraman.tif are used as input images. The following are the matlab files used for this problem:

- main_anisotropicdiffusion.m: The following are the steps performed in this file:
  - Asks the user to choose between the two input images available
  - As per users choice, reads the image.
  - Applies anisotropic diffusion on the input image 100 times iteratively for both exponential and inverse quadratic methods of g(.).
  - Sets the variable values required by anisotropic diffusion function such as $\lambda = 0.25$, $k = 30/255$ for exponential method of g(.) and $k = 10/255$ for inverse quadratic method of g(.).
  - Displays the results of anisotropic diffusion for 0,5,20 and 100th iterations.
  - Computes gray-scale histogram on the results using matlab inbuilt function histogram(A) and displays it.
  - Plots the line y=128 through the resultant image

  - Calls function that segments the output image and displays the results.(threshold =128 for segmentation)

- fcn_AnisoDiff.m: Performs anisotropic diffusion on the input image using the λ and k values passed to it. The function also needs to know which method of g(.) to be used. It implements anisotropic diffusion using the equation2.7.

- fcn_CondCoe.m: This function computes the conduction coefficients using the k value and method of g(.) passed to it as inputs.(implements equation2.8

- fcn_segmentImg.m: This function performs segmentation on the image using the threshold passed to it as input. The threshold value is chosen manually as 128 to segment out the gray spokes component of the wheel.

The results obtained are displayed in section3.
Note: Run main_anisotropicdiffusion.m file in Prob2_AnisotropicDiffusion folder to get the results

The flowcharts for nonlinear filtering is displayed in Figure1 and for anisotropic diffusion is displyed in Figure 2.

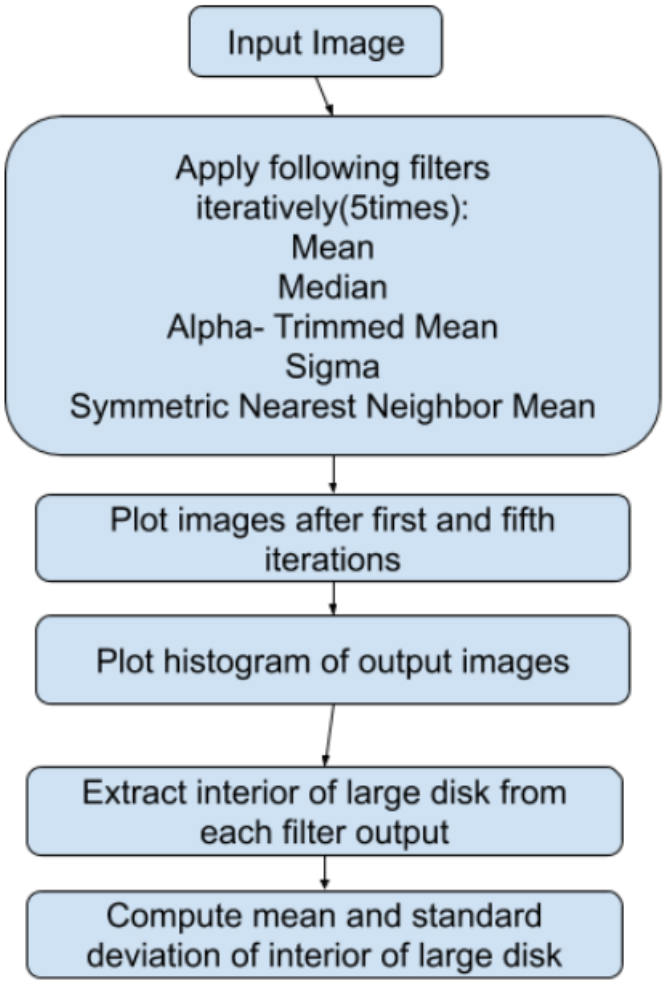

Figure 1: Flowchart :Nonlinear Filtering

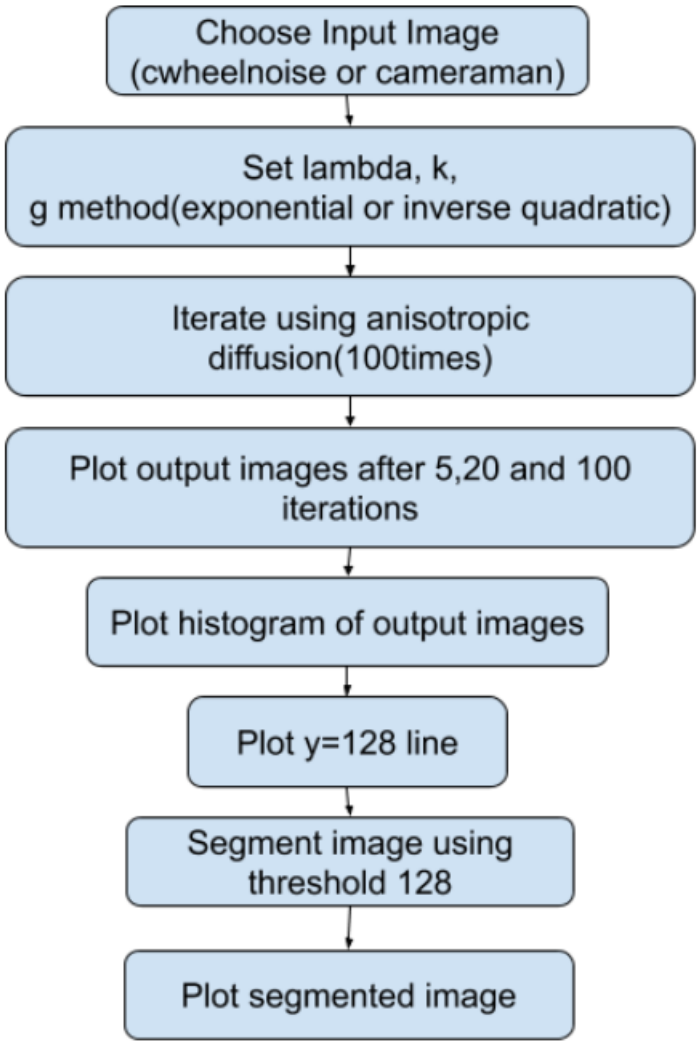

Figure 2: Flowchart : Anisotropic Diffusion

3 Results

3.1 Nonlinear Filtering

To implement nonlinear filters, RandomDisks-P10.jpg is used as inpput image which is shown in Figure3. Once user runs main_nonlinearfiltering.m, the program applies all the five non linear filter on the input image for five times iteratively.

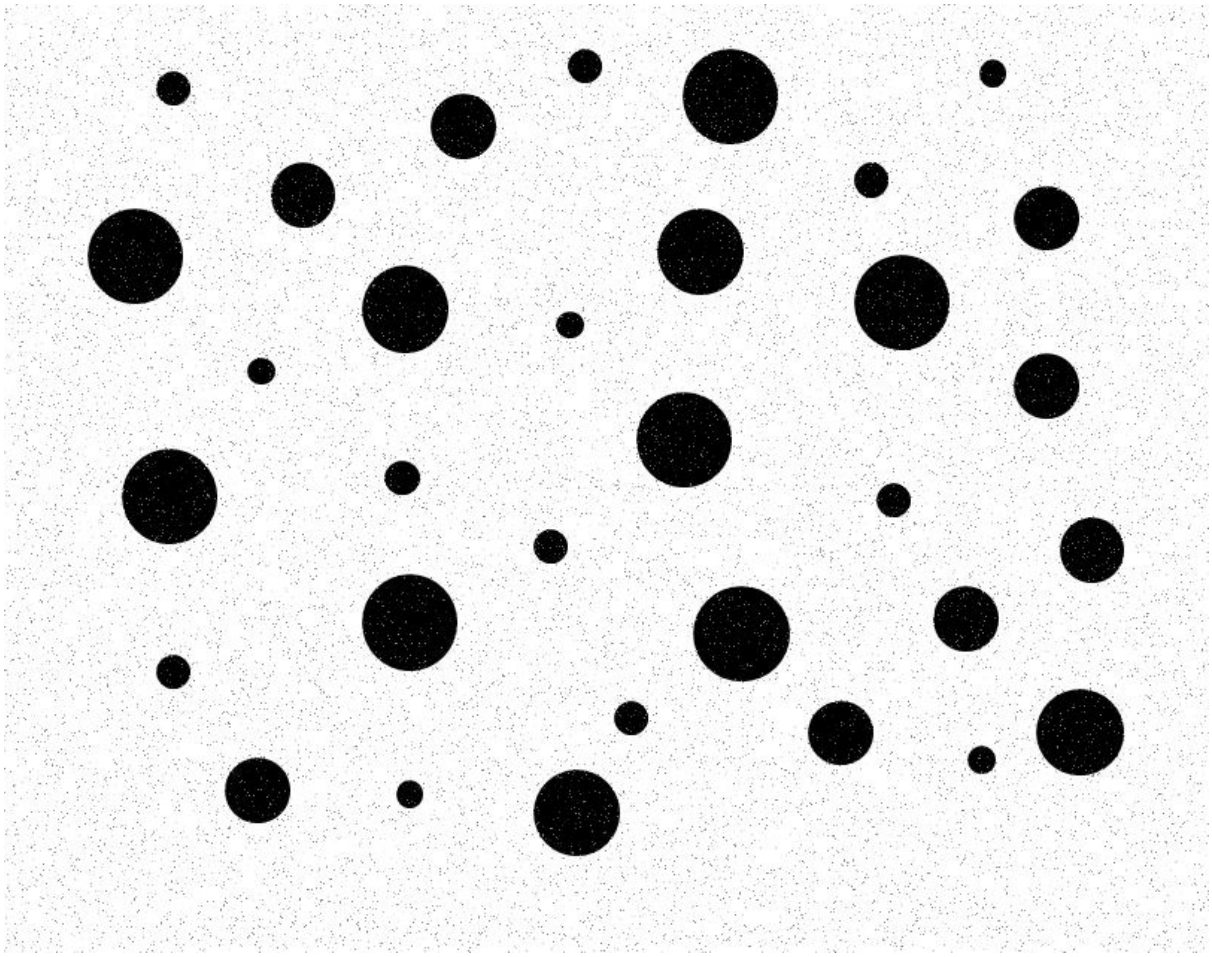

Figure 3: Input Image - RandomDisks-P10.jpg

Figures4,6,8, 10, 12 displays the results of mean, median, alpha-trimmed mean, sigma and symmetric nearest neighbor mean filters after first iteration respectively. Figures5,7,9, 11, 13 displays the gray scale histograms of results of mean, median, alpha-trimmed mean, sigma and symmetric nearest neighbor mean filters after first iteration respectively.

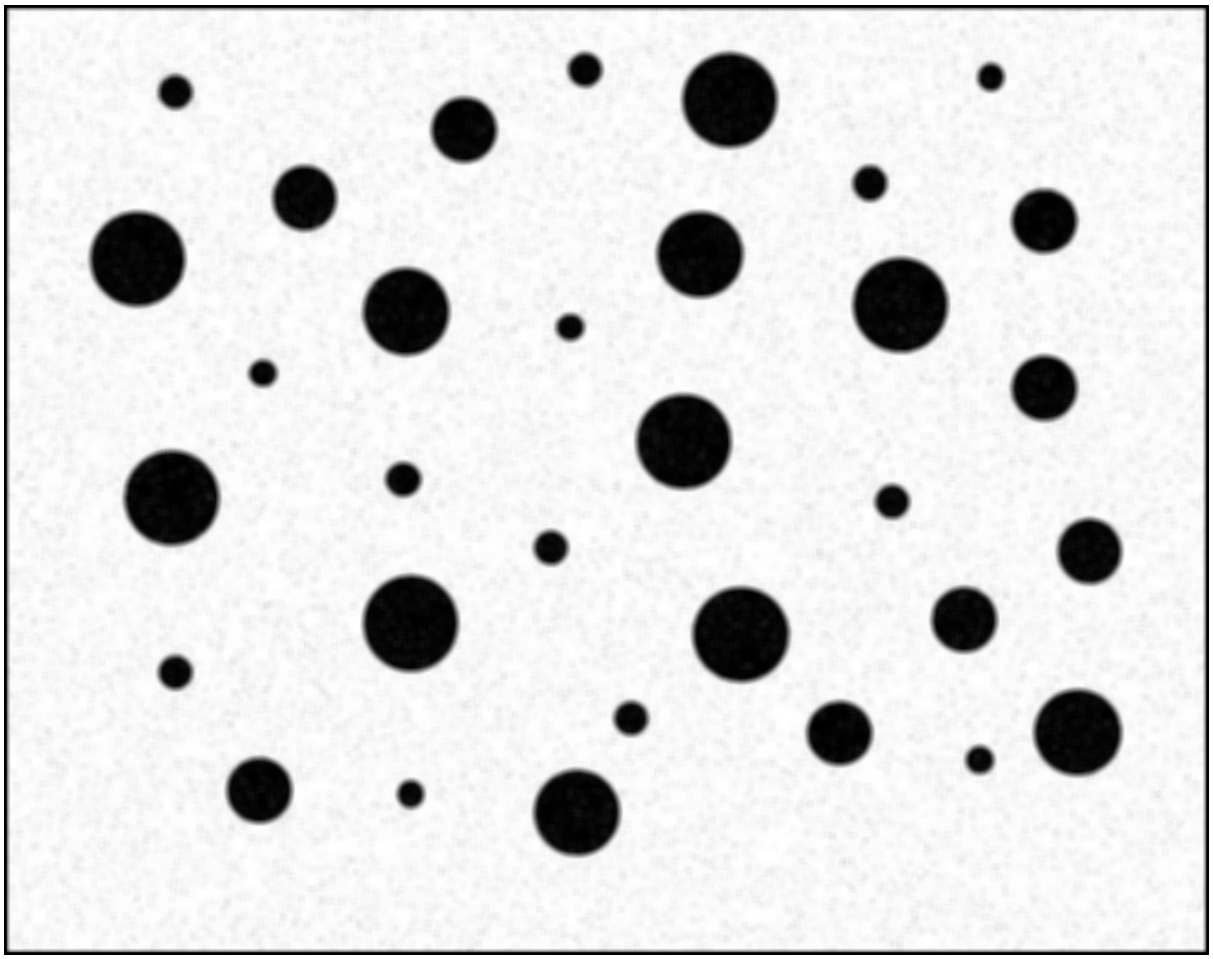

Figure 4: Mean Filter Output(iteration:1)

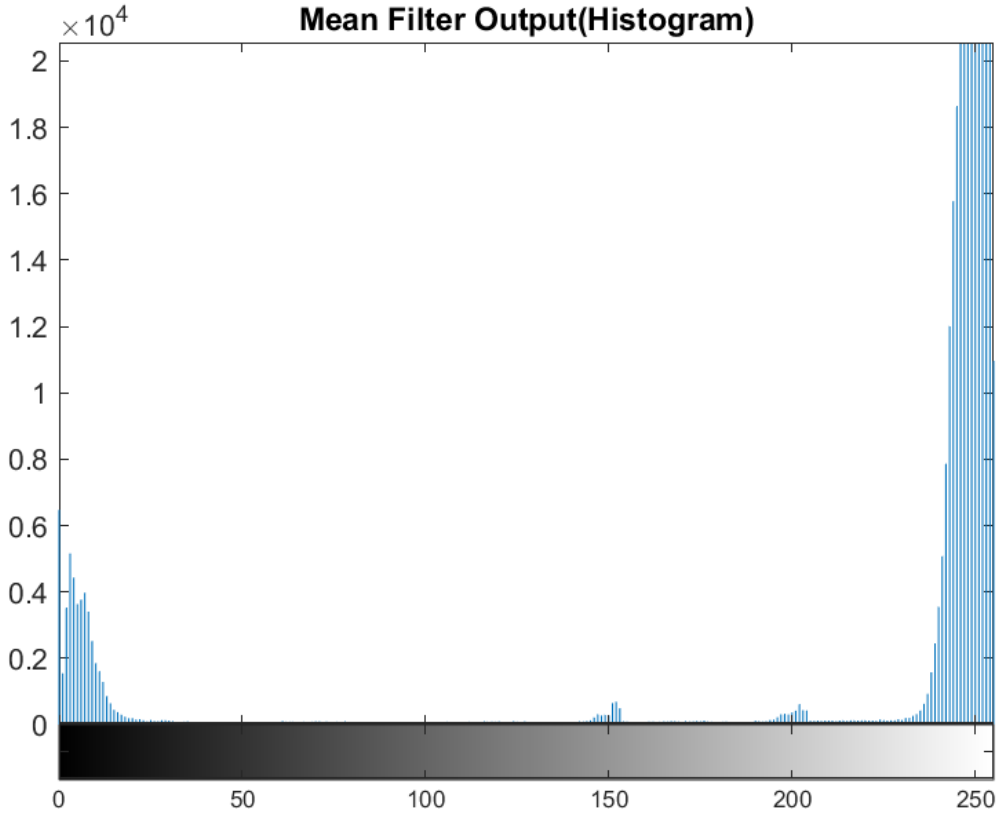

Figure 5: Histogram of Mean Filter Output(iteration:1)

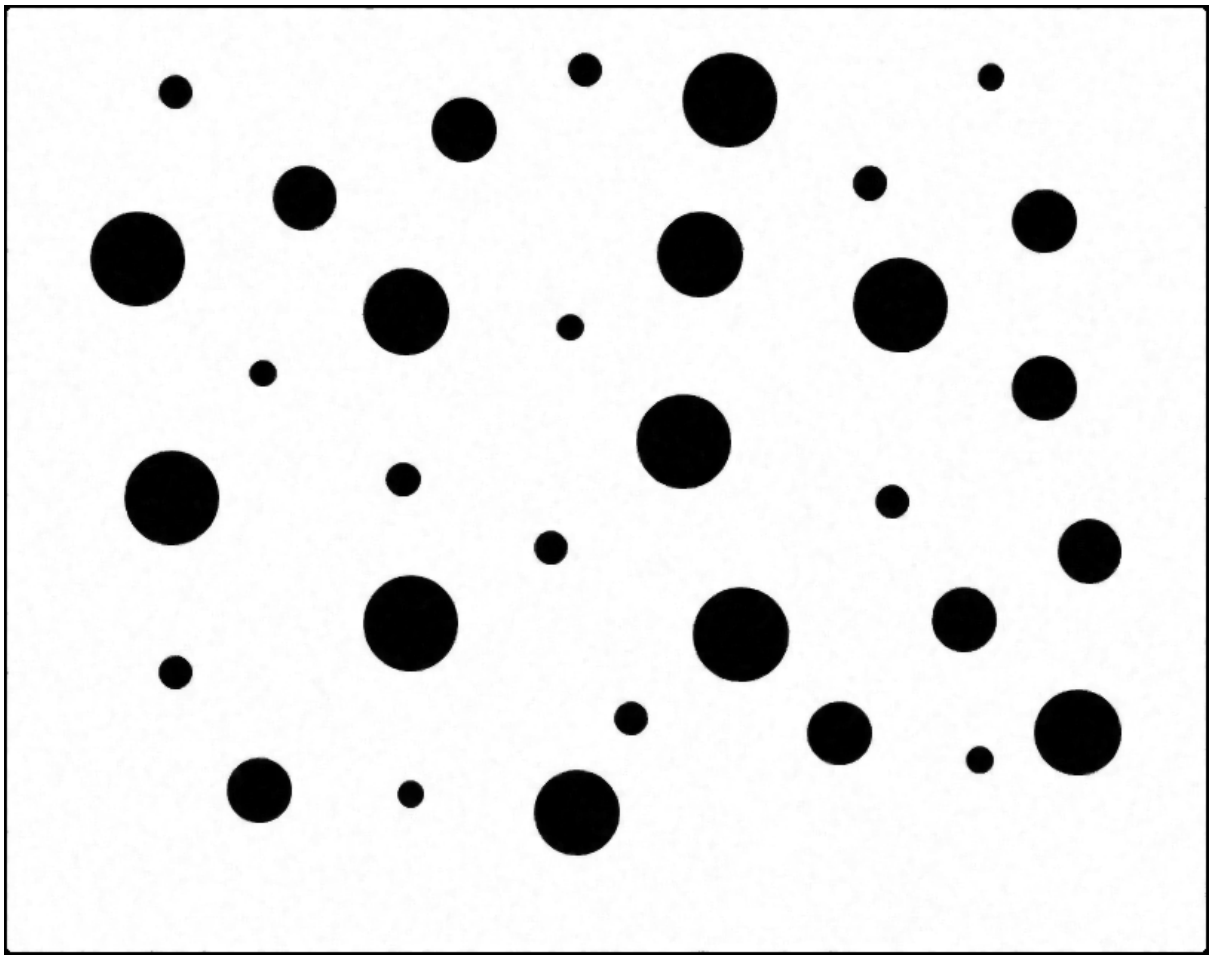

Figure 6: Median Filter Output(iteration:1)

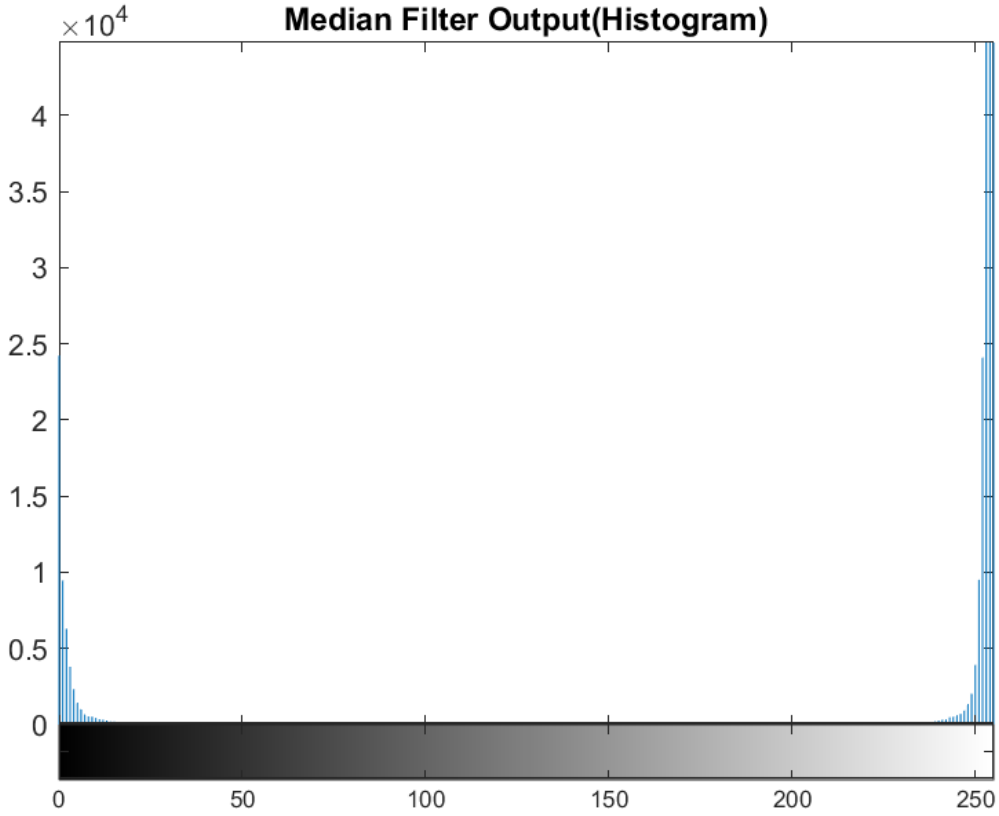

Figure 7: Histogram of Median Filter Output(iteration:1)

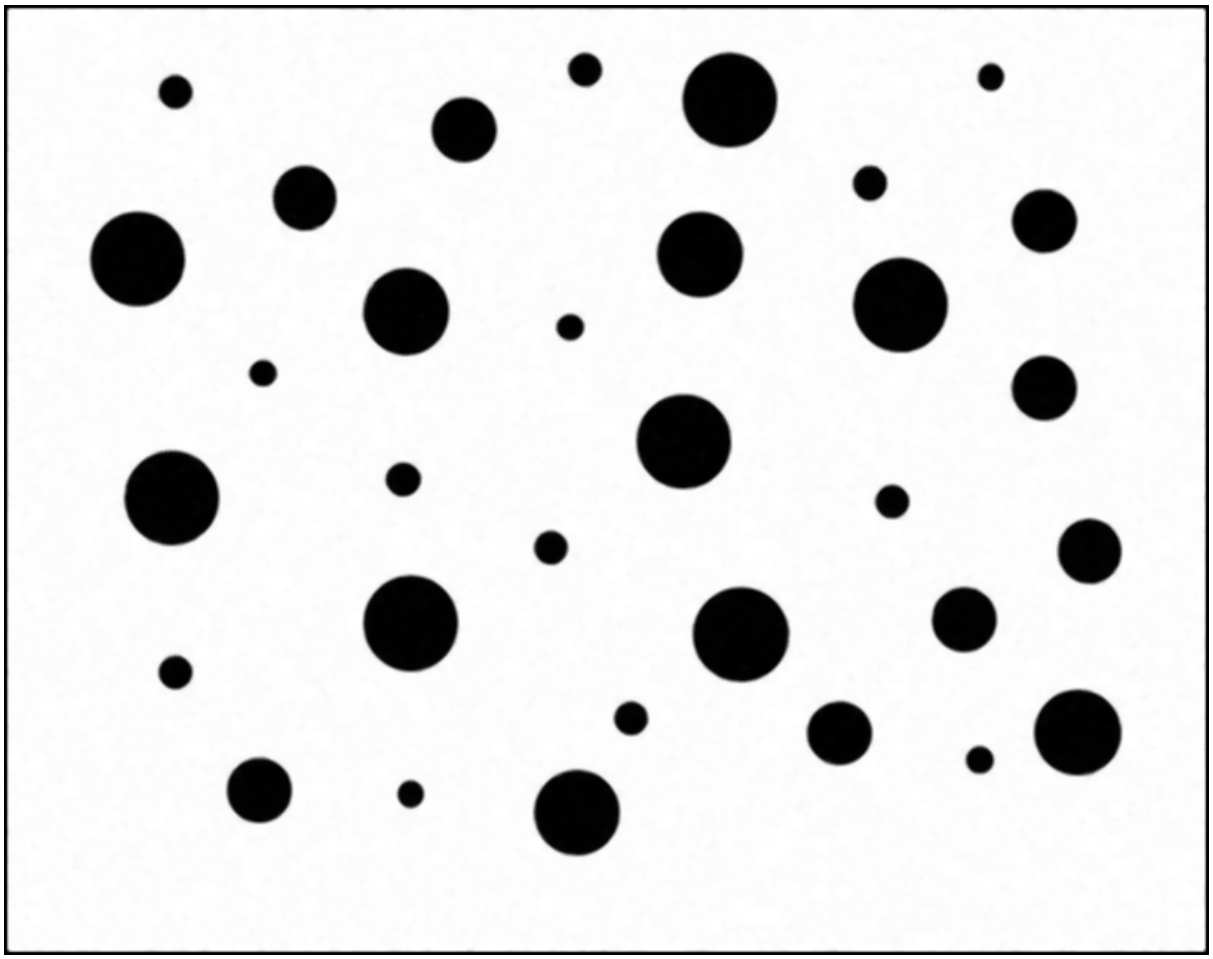

Figure 8: Alpha-Trimmed Filter Output(iteration:1)

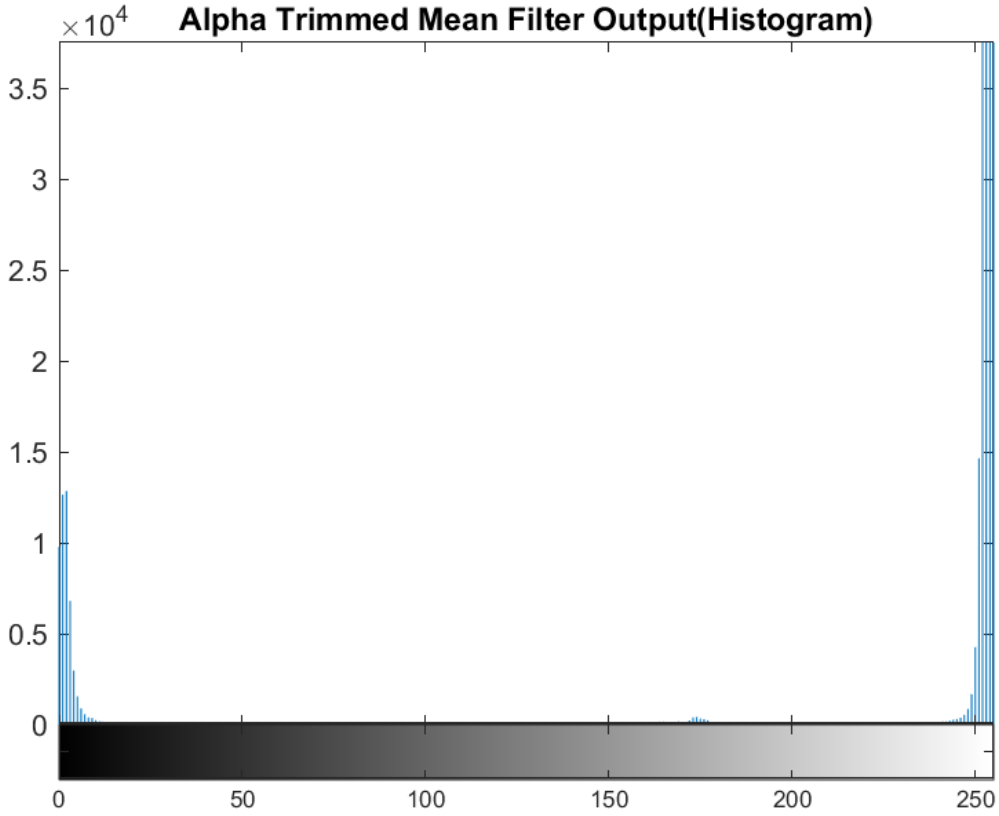

Figure 9: Histogram of Alpha-Trimmed Filter Output(iteration:1)

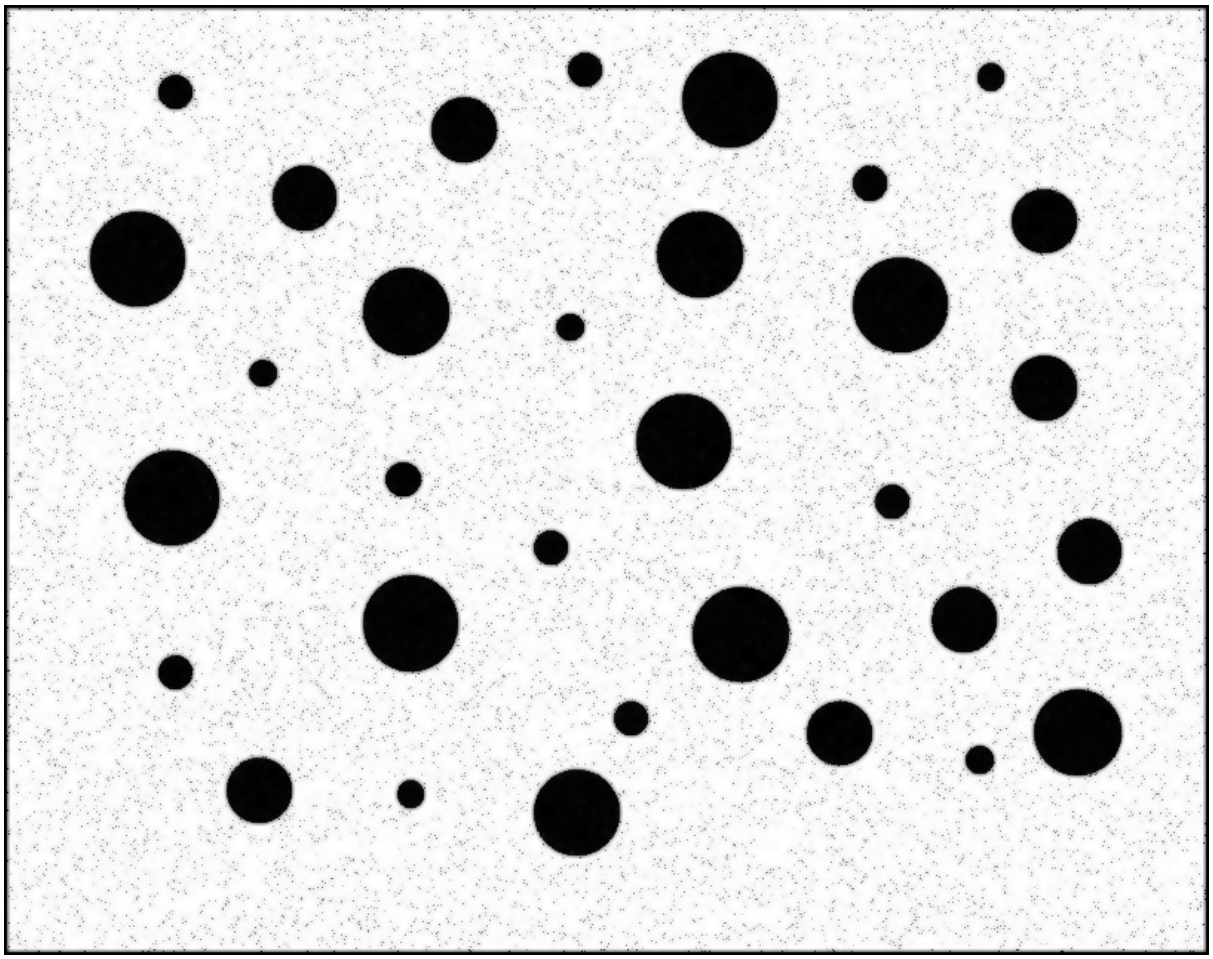

Figure 10: Sigma Filter Output(iteration:1)

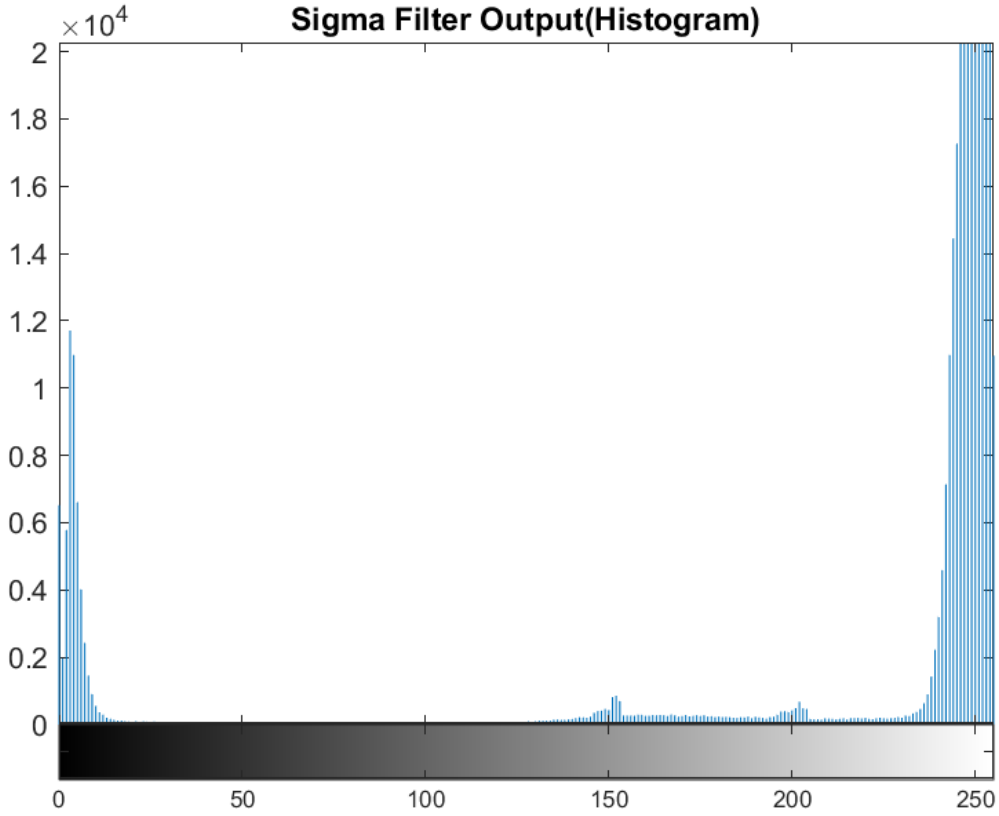

Figure 11: Histogram of Sigma Filter Output(iteration:1)

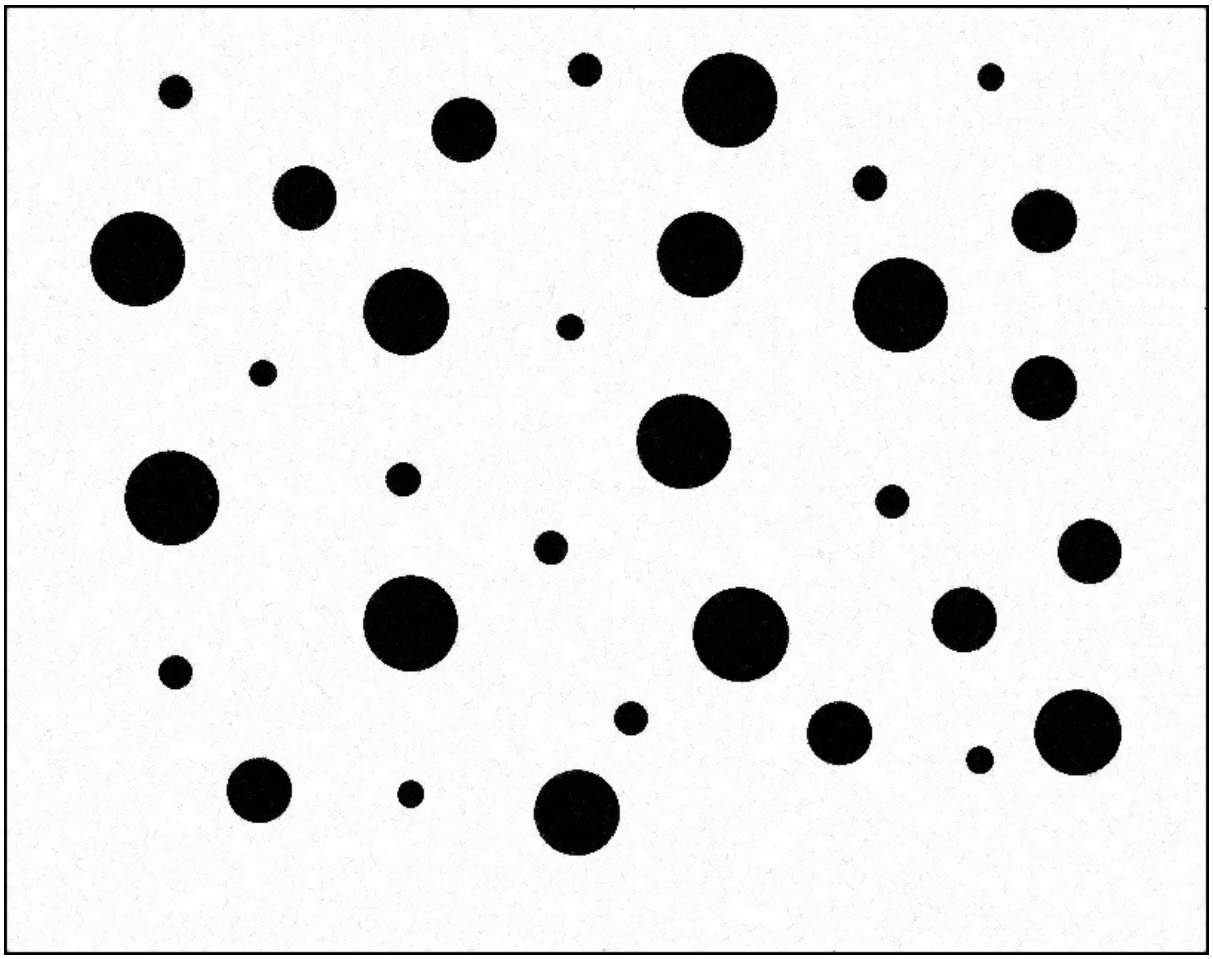

Figure 12: Symmetric NNM Filter Output(iteration:1)

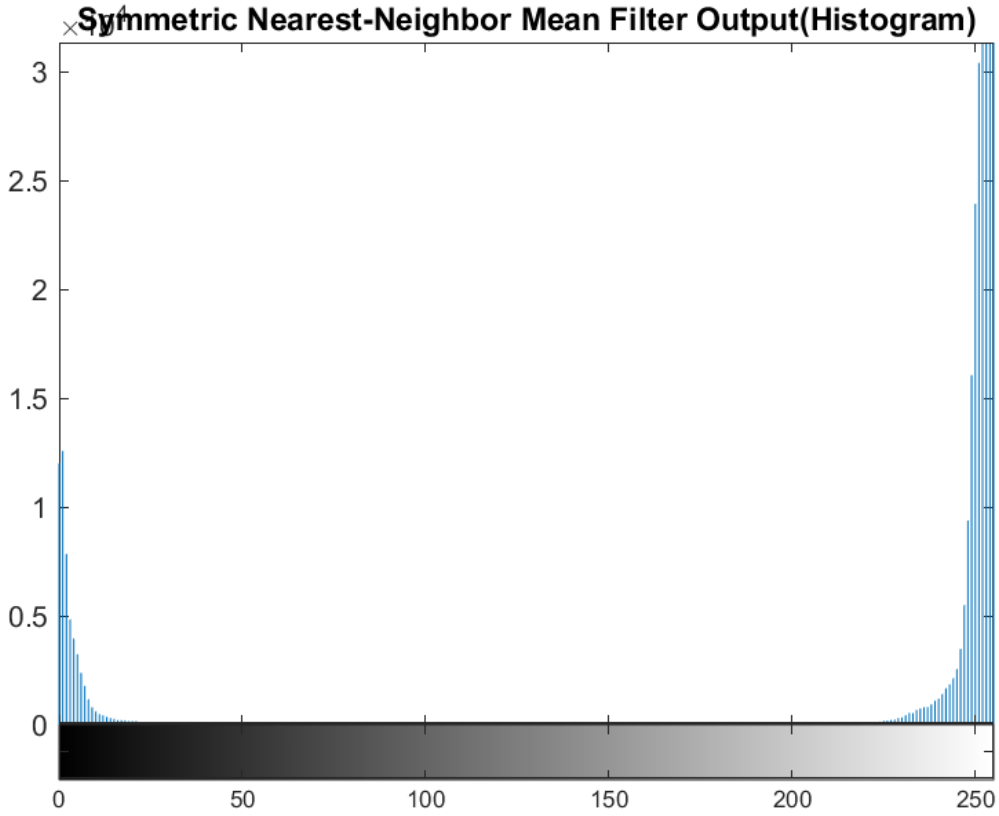

Figure 13: Histogram of Symmetric NNM Filter Output(iteration:1)

Figures14,16,18, 20, 22 shows the results of mean, median, alpha-trimmed mean, sigma and symmetric nearest neighbor mean filters after fifth iteration respectively. Figures15,17,19, 21, 23 displays the gray scale histograms of results of mean, median, alpha-trimmed mean, sigma and symmetric nearest neighbor mean filters after fifth iteration respectively.

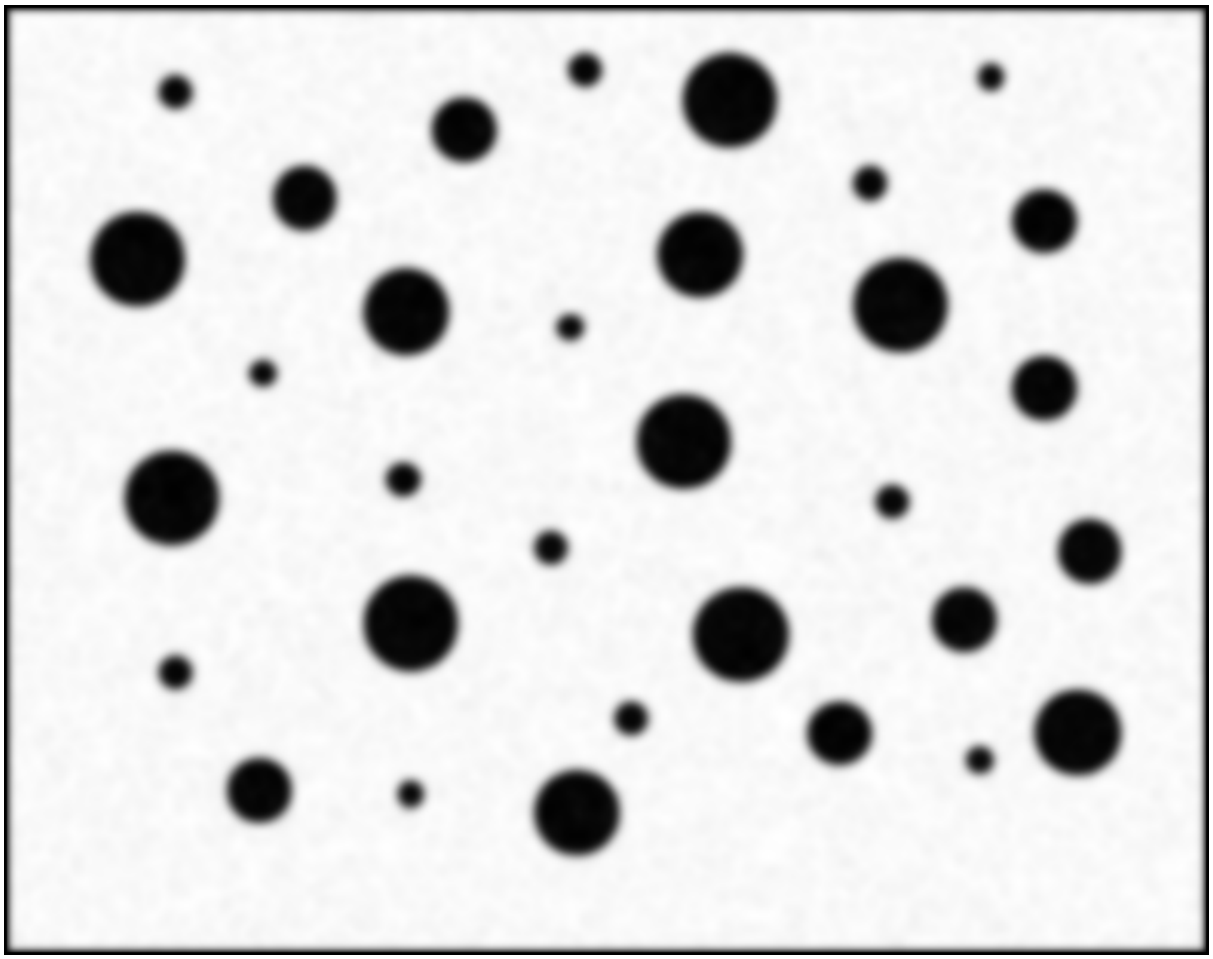

Figure 14: Mean Filter Output(iteration:5)

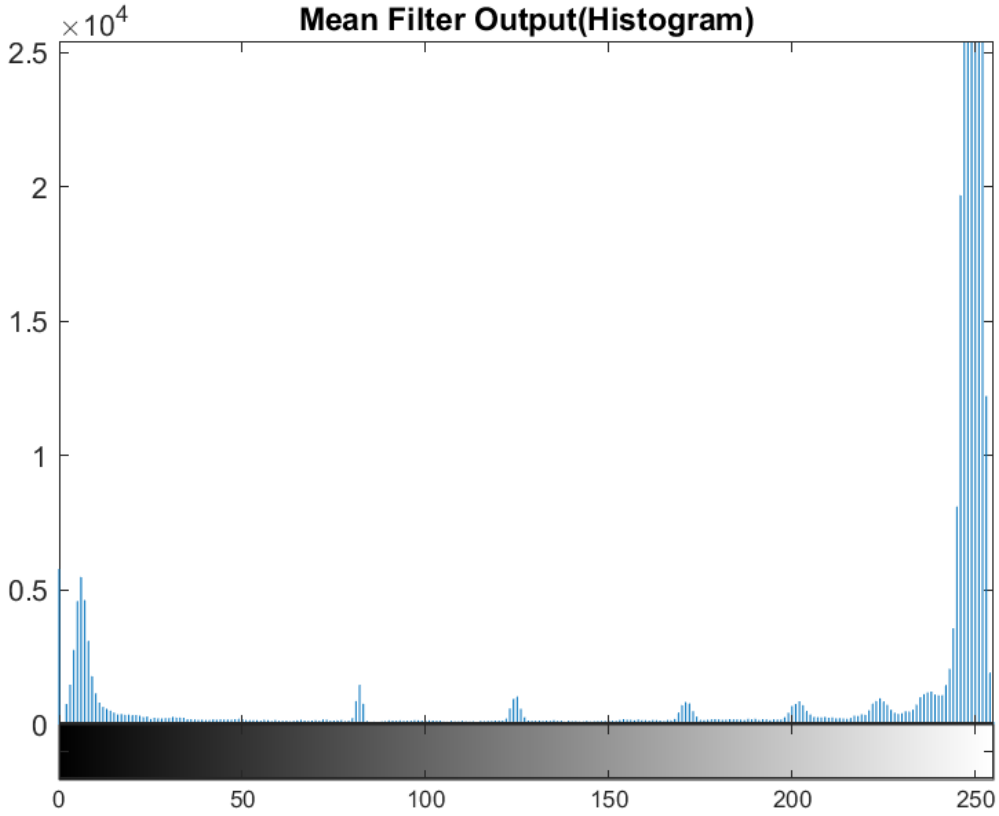

Figure 15: Histogram of Mean Filter Output(iteration:5)

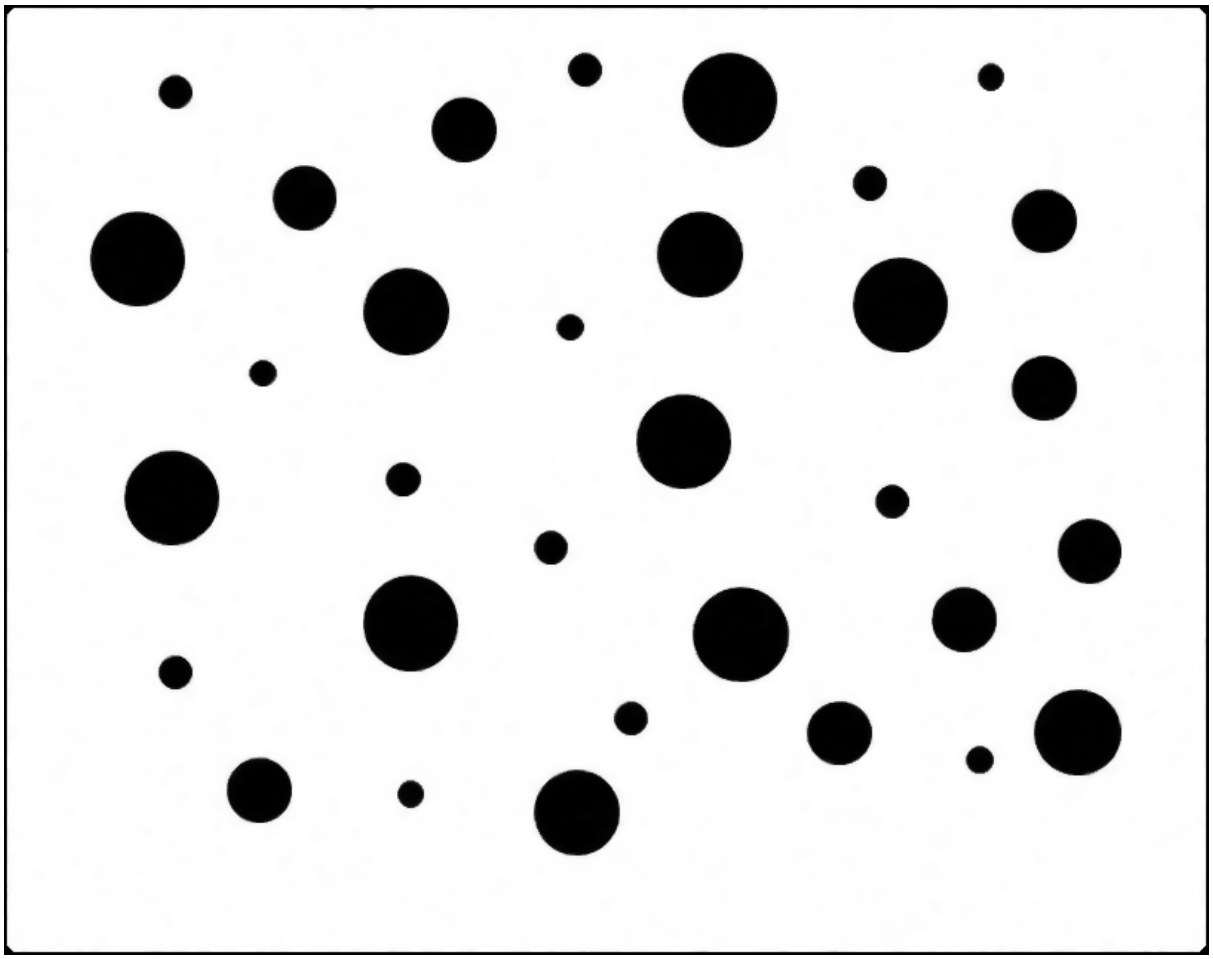

Figure 16: Median Filter Output(iteration:5)

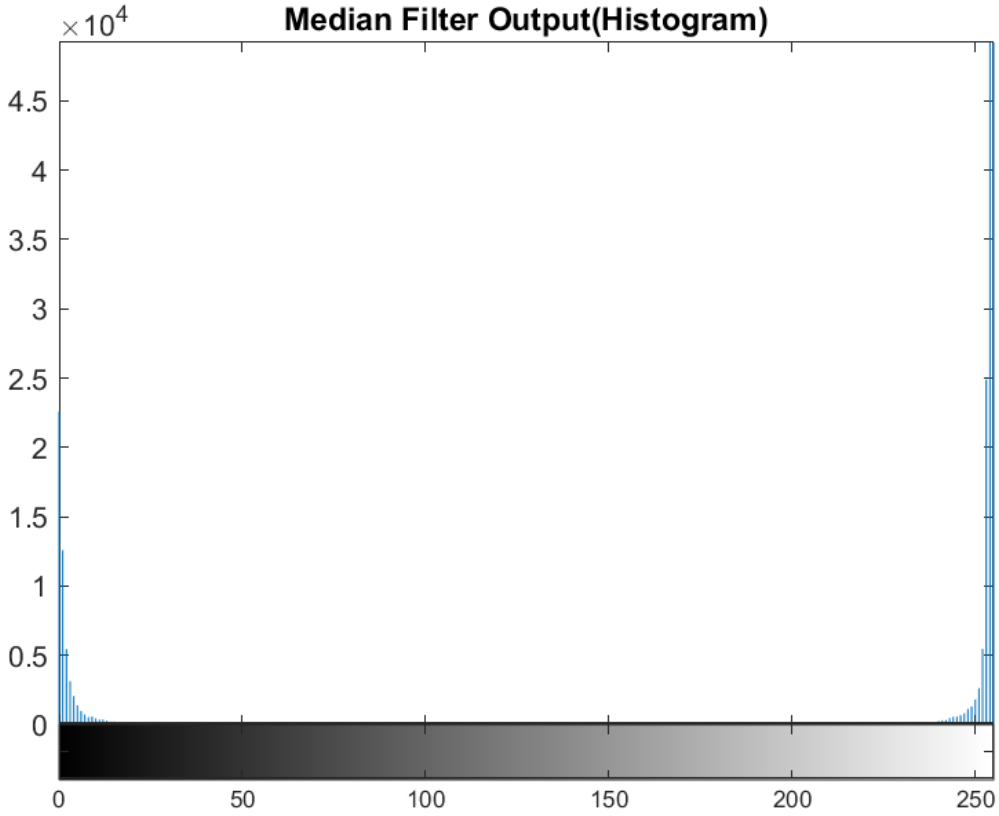

Figure 17: Histogram of Median Filter Output(iteration:5)

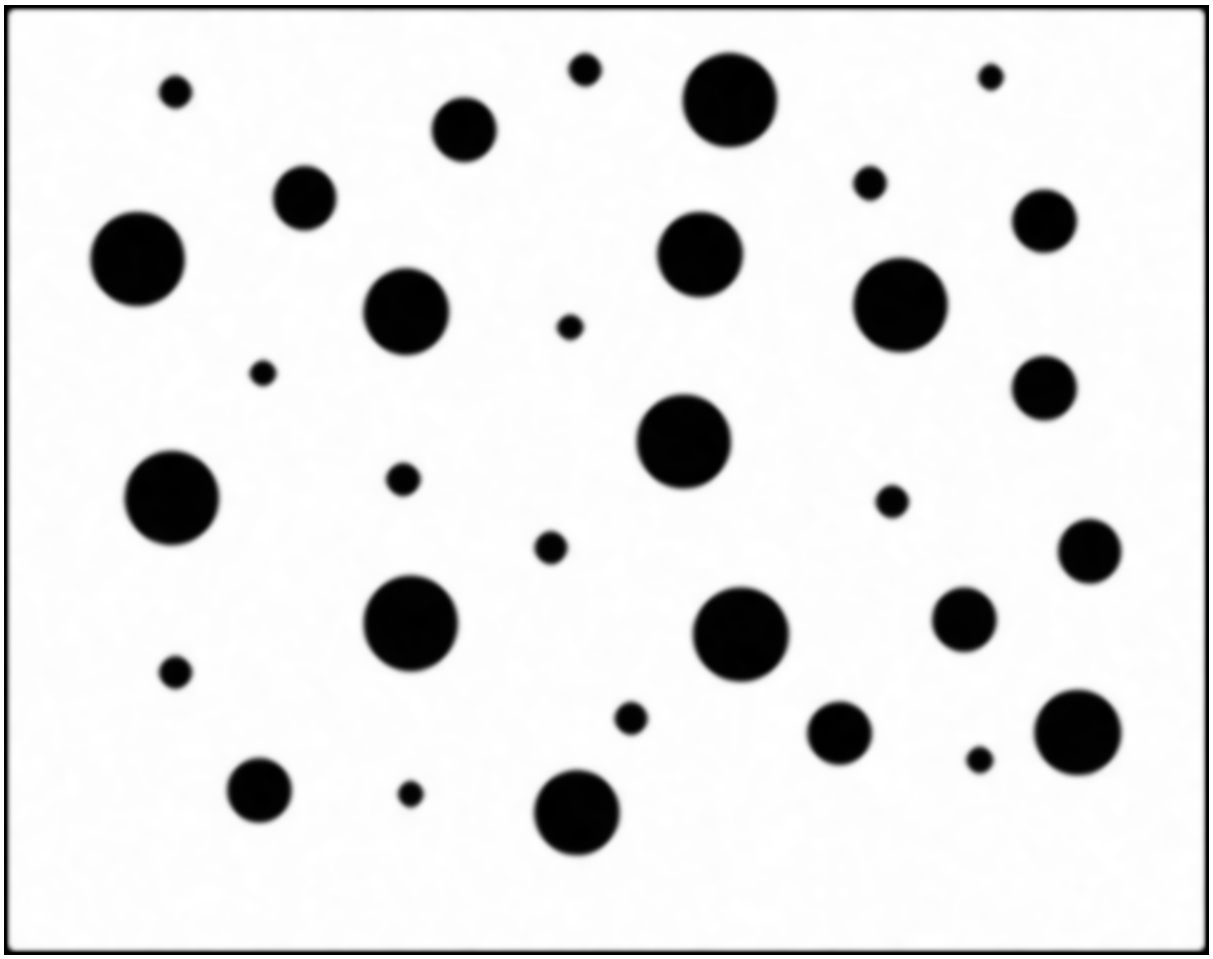

Figure 18: Alpha-Trimmed Filter Output(iteration:5)

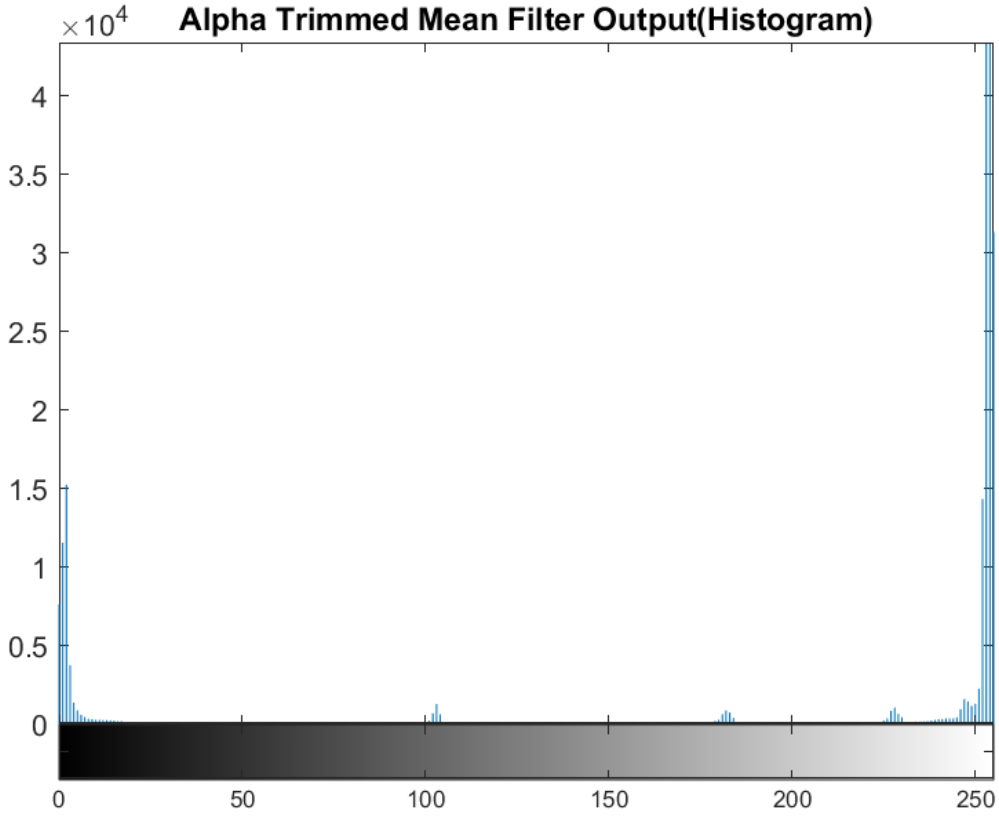

Figure 19: Histogram of Alpha-Trimmed Filter Output(iteration:5)

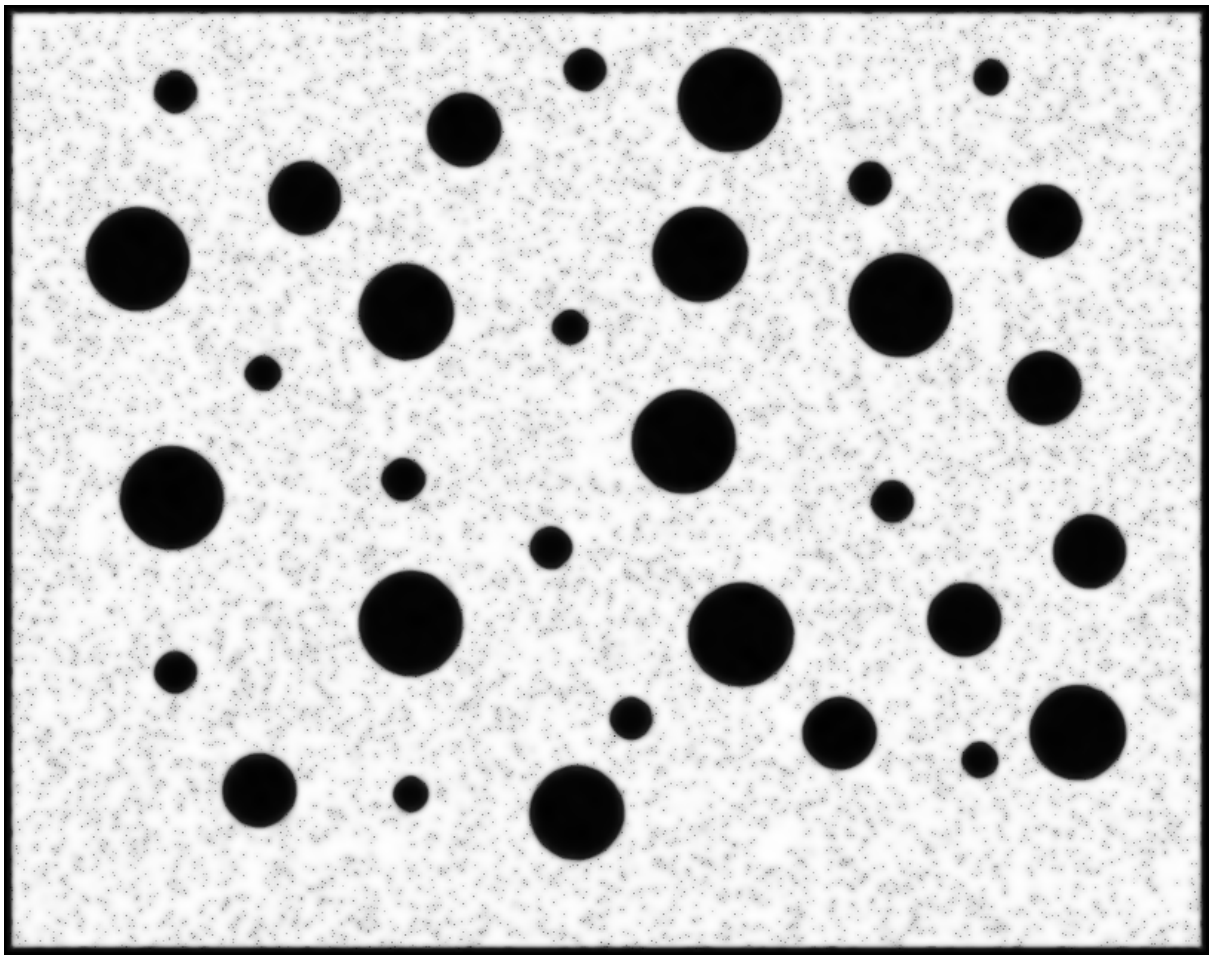

Figure 20: Sigma Filter Output(iteration:5)

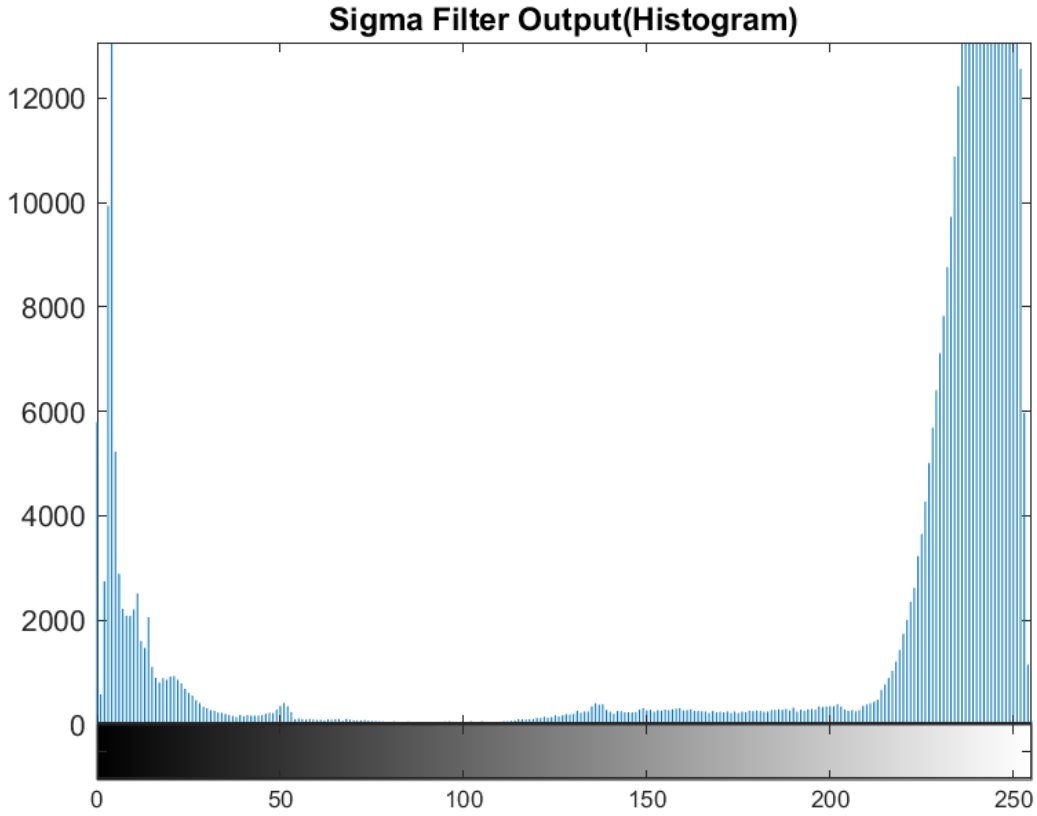

Figure 21: Histogram of Sigma Filter Output(iteration:5)

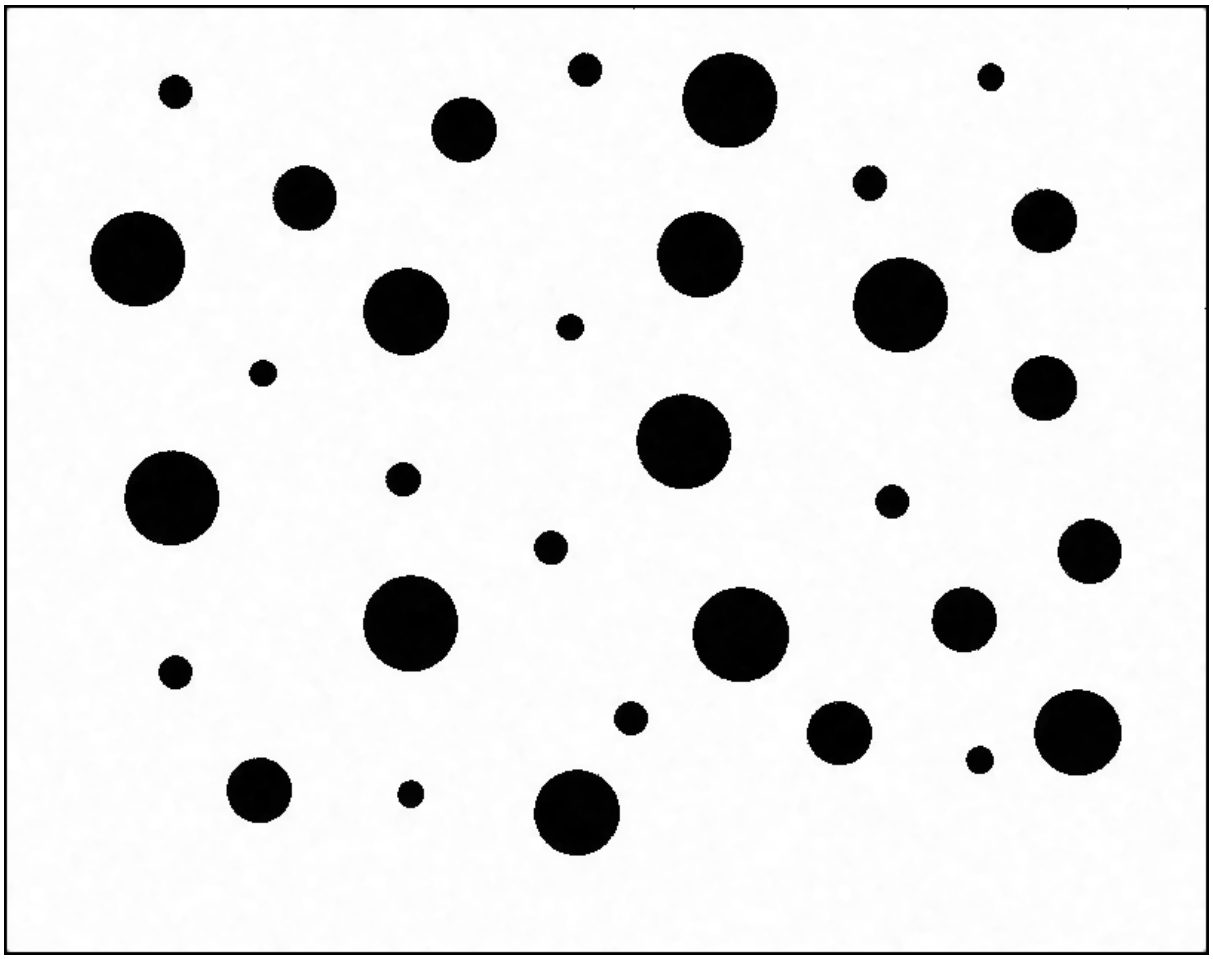

Figure 22: Symmetric NNM Filter Output(iteration:5)

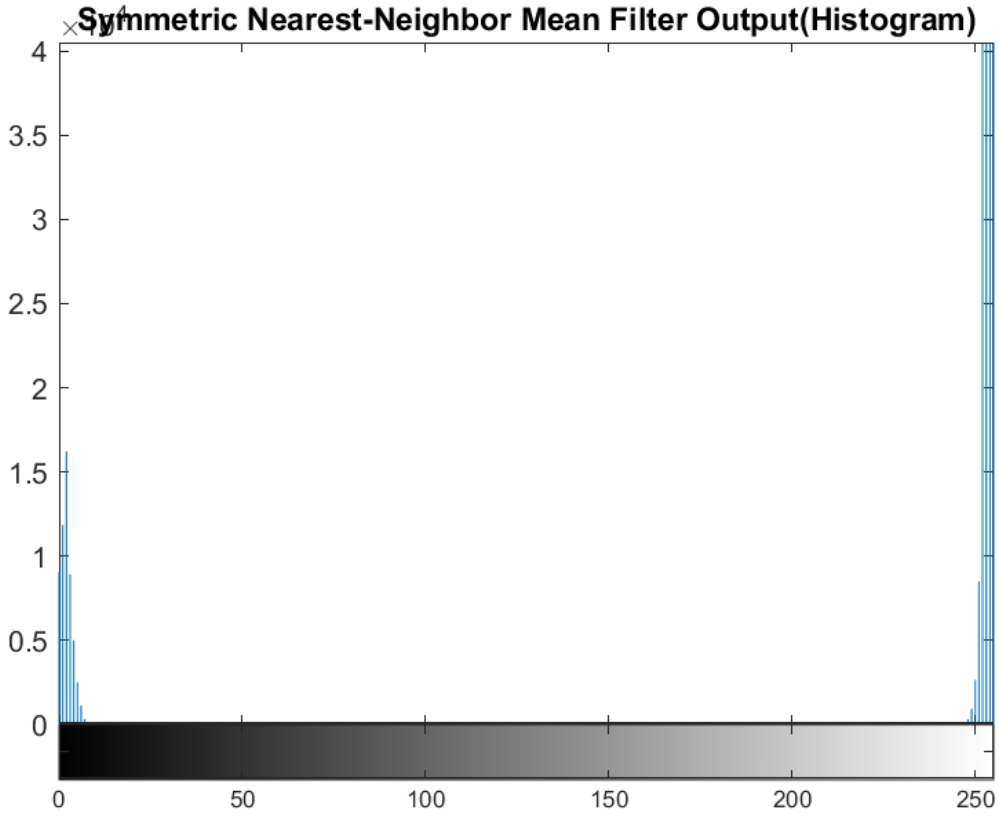

Figure 23: Histogram of Symmetric NNM Filter Output(iteration:5)

Large disk and interior of large disk region are extracted from the input image by manually specifying the boundaries which are displayed in Figures24 and Fig25. Interior of large disk is extracted from the result of each filter, then mean and standard deviation of extracted large disk interior image are computed and displayed in table1.

Figure 24: Large Disk

Figure 25: Large Disk Interior

		Mean	STD
2***Mean Filter**	**Iteration:1**	6.166667e+00	3.492226e+00
	Iteration:5	6.593951e+00	3.386537e+00
2***Median Filter**	**Iteration:1**	7.689833e-01	1.040105e+00
	Iteration:5	4.433719e-01	6.653588e-01
2***Alpha Trimmed Mean Filter**	**Iteration:1**	1.691763e+00	8.904424e-01
	Iteration:5	1.709781e+00	7.123740e-01
2***Sigma Filter**	**Iteration:1**	3.505792e+00	1.653198e+00
	Iteration:5	3.568211e+00	7.498743e-01
2***Symmetric NNM Filter**	**Iteration:1**	2.745817e+00	3.548995e+00
	Iteration:5	1.741956e+00	7.868545e-01

Table 1: Mean and standard deviation of interior of large disk image extracted from the output of each filter after first and fifth iterations

Prob1(c)Observations on results obtained after filtering: The following are observations made:

- Mean Filter: Fig4 and Fig14 show that the image(edges) got blurred and noise is also not removed after applying mean filter.
- Median Filter: This filter gave good results by removing noise completely as can be seen in fig6 and fig16.
- Alpha-Trimmed Mean Filter: The outputs of this filter shown in fig8,18 show that the filter was successful in removing noise from the input image but the output image appears to be blurred.
- Sigma Filter:This filter preserves edges while reducing the variance. So, the noise is not removed as can be seen in fig10,20.
- Symmetric Nearest Neighbor Mean Filter: From output images 12 and 20, it can be the this filter gave good results in removing noise and gave clear outputs.

3.2 Anisotropic Diffusion

To implement anisotropic diffusion, two input images cwheelnoise.gif and cameraman.tif are considered. These are shown in Figure 26 and Figure 27. Once user runs $main\_anisotropicdiffusion$ file, the program asks the user to select the input image. Press 1 to select cwheelnoise and press 2 to select cameraman. Anisotropic diffusion is performed on the selected image iterativley 100 times. The outputs obtained in iteration 0,5,20 and 100 are diplayed in this report. For anisotropic diffusion, λ is given as 0.25.

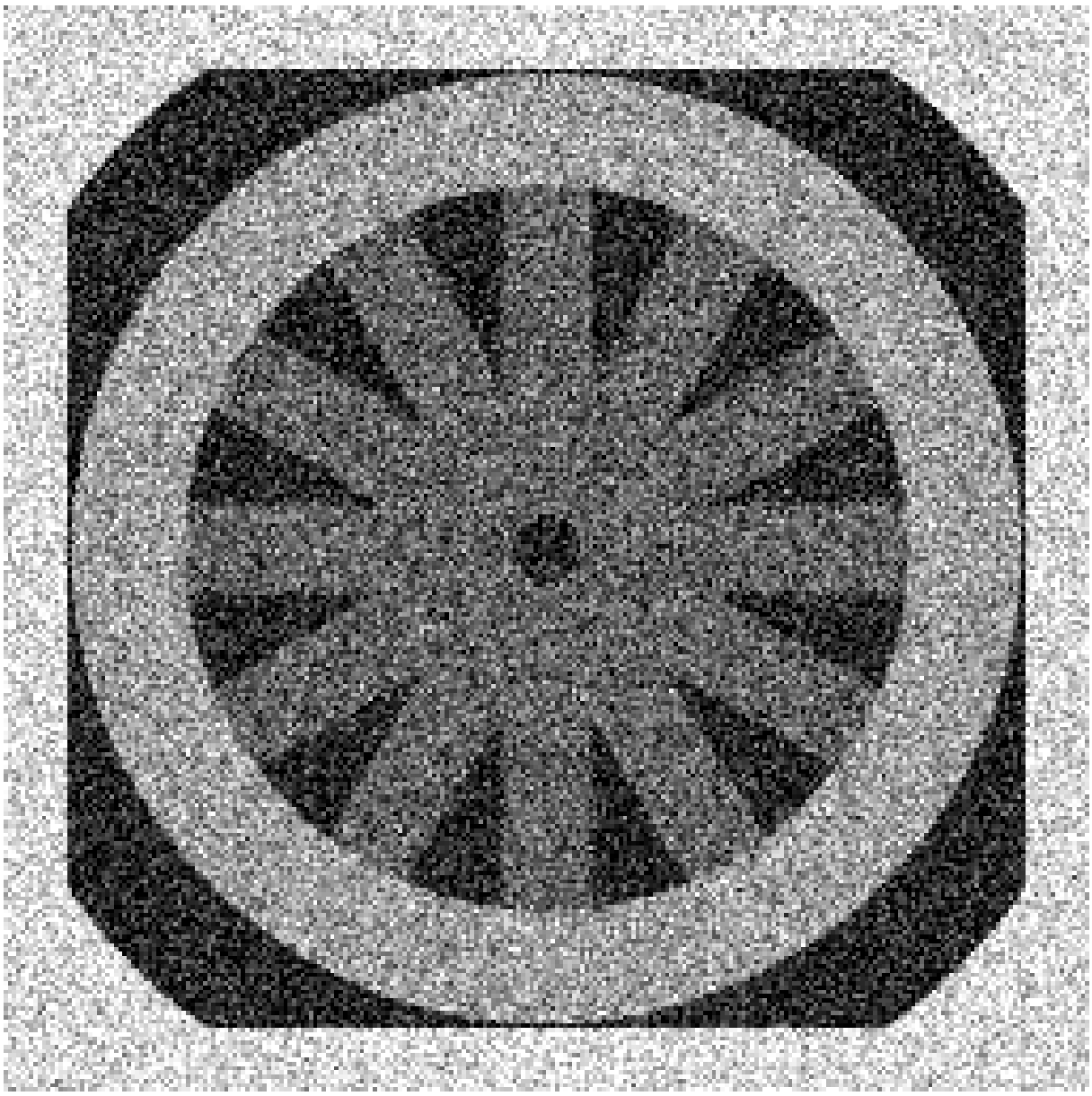

Figure 26: Input Image: cwheelnoise

Figure 27: Input Image: cameraman

For cwheelnoise image, anisotropic diffusion with exponential method of g(.)(k=30/255) results are displayed in Fig28. Histogram of results is shown in Fig29. Line plot for y=128 is displayed in fig30. The output images are segmented using 128 as threshold, the segmented images are shown in fig31.

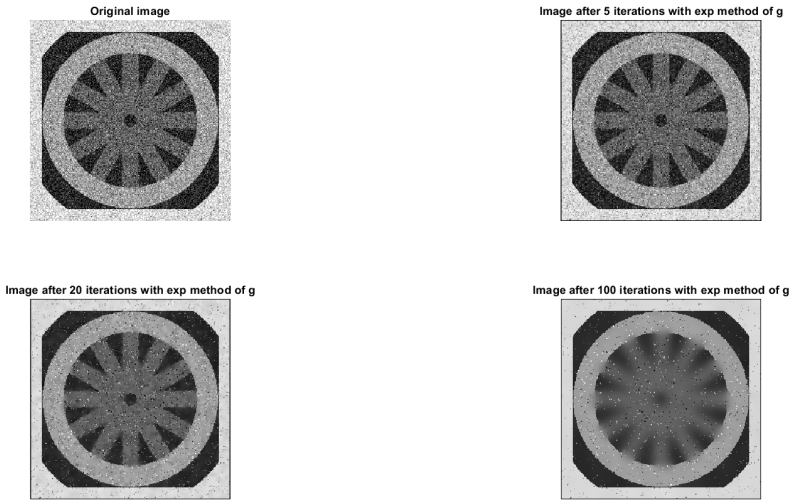

Figure 28: cwheelnoise: Images with exponential method of g

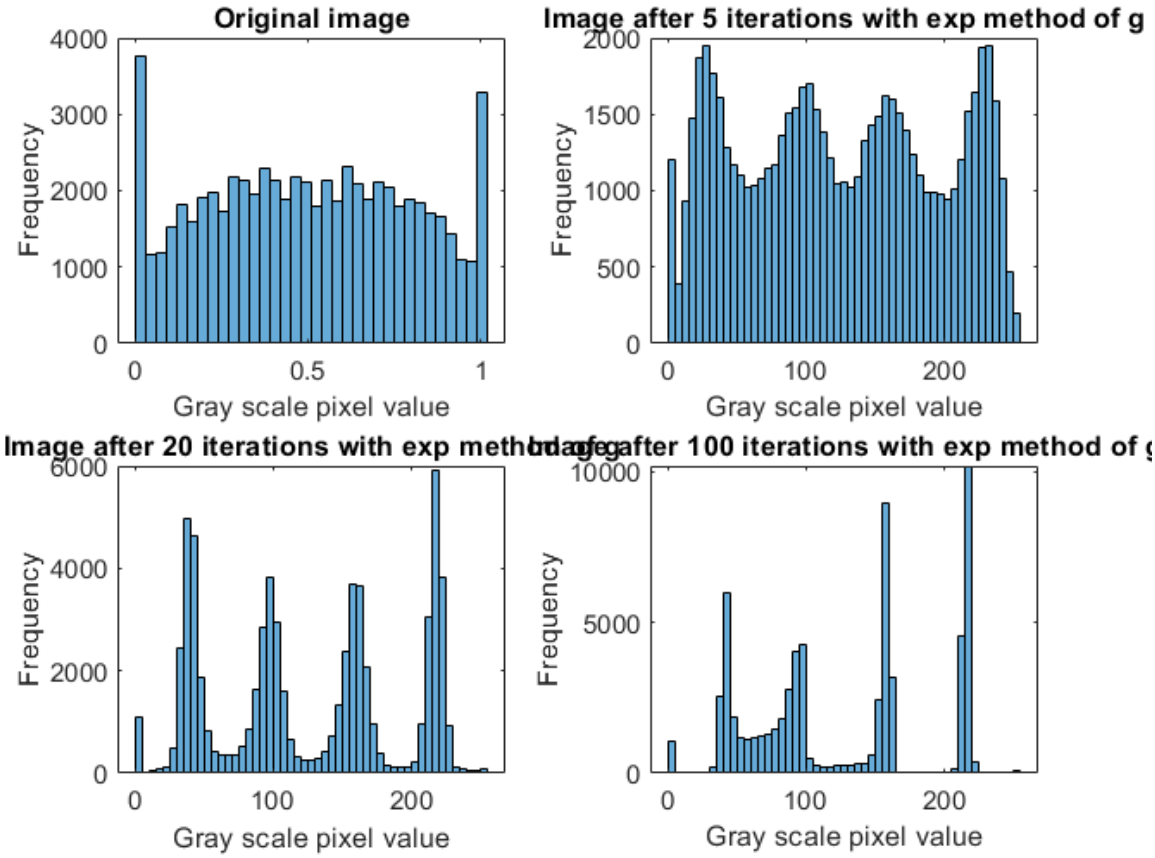

Figure 29: cwheelnoise: Histograms with exponential method of g

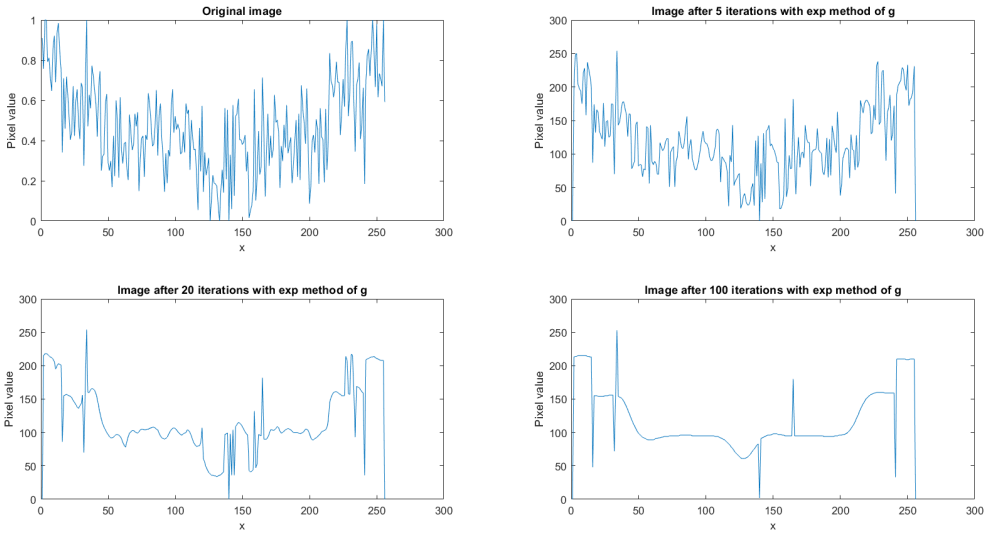

Figure 30: cwheelnoise: Line y=128 through the images with exponential method of g

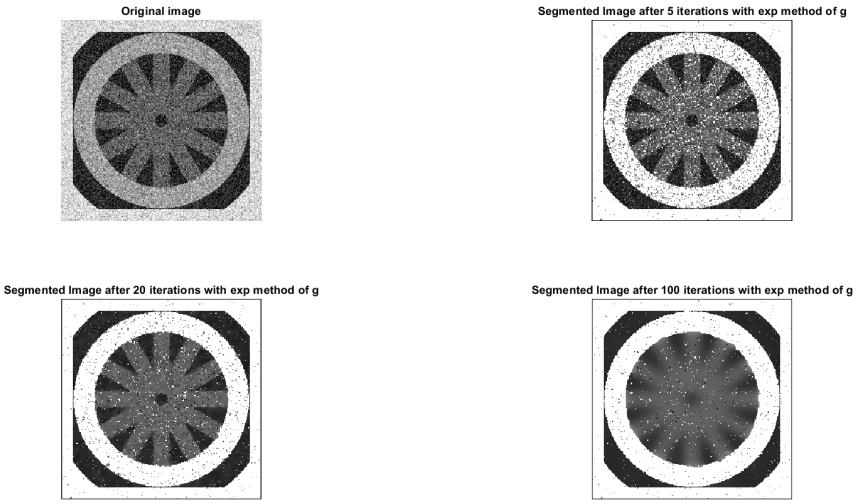

Figure 31: cwheelnoise: Segmented images with exponential method of g

For cwheelnoise image, anisotropic diffusion with inverse quadratic method of g(.)(k=10/255) results are displayed in Fig32. Histogram of results is shown in Fig33. Line plot for y=128 is displayed in fig34. The output images are segmented using 128 as threshold, the segmented images are shown in fig35.

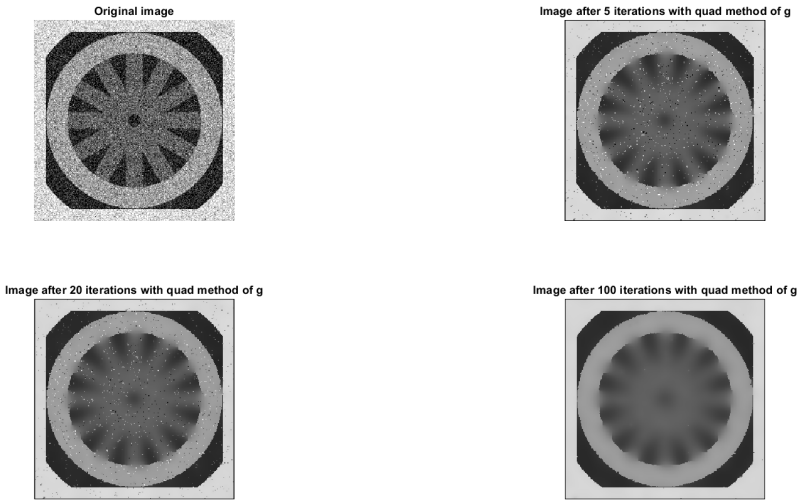

Figure 32: cwheelnoise: Images with quadratic method of g

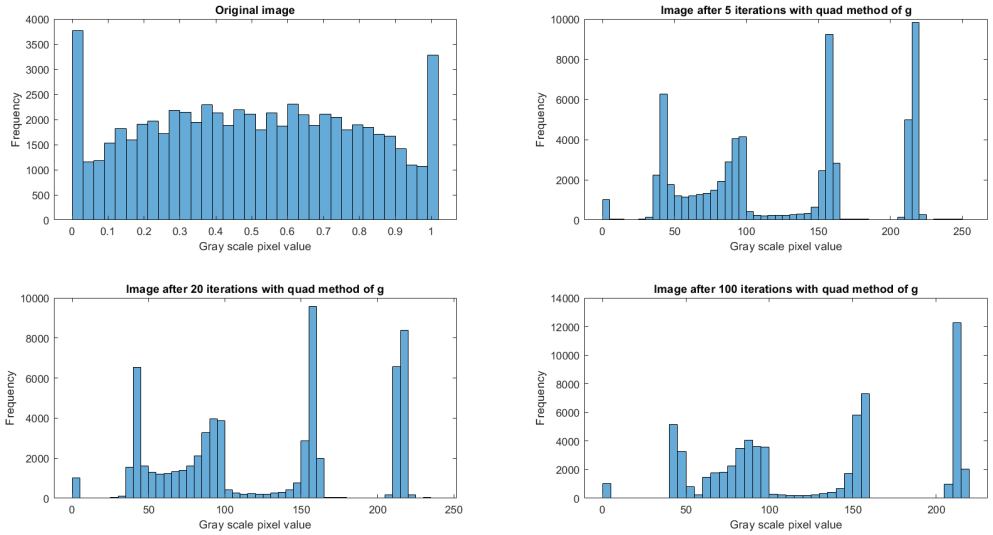

Figure 33: cwheelnoise: Histograms with quadratic method of g

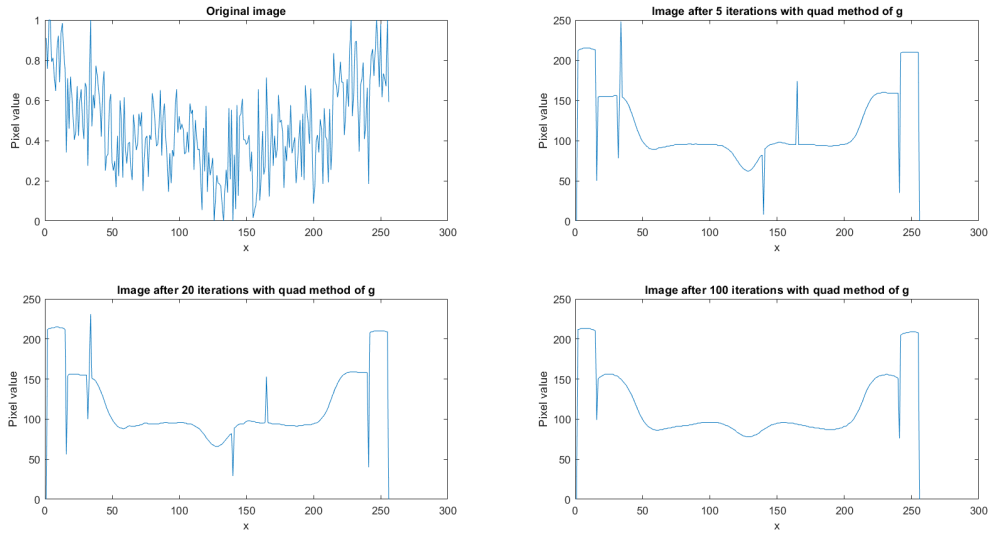

Figure 34: cwheelnoise: Line y=128 through the images with quadratic method of g

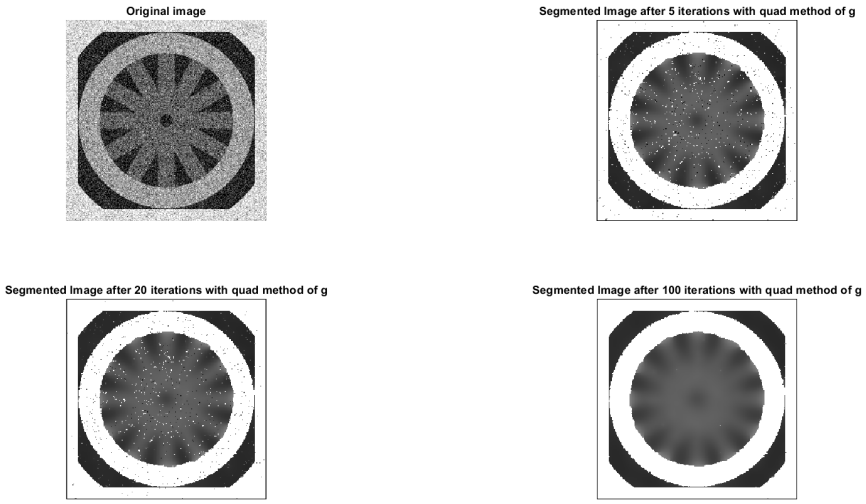

Figure 35: cwheelnoise: Segmented images with quadratic method of g

For cameraman image, anisotropic diffusion with exponential method of g(.)(k=30/255) results are displayed in Fig36. Histogram of results is shown in Fig37. Line plot for y=128 is displayed in fig38. The output images are segmented using 128 as threshold, the segmented images are shown in fig39.

Figure 36: cameraman: Images with exponential method of g

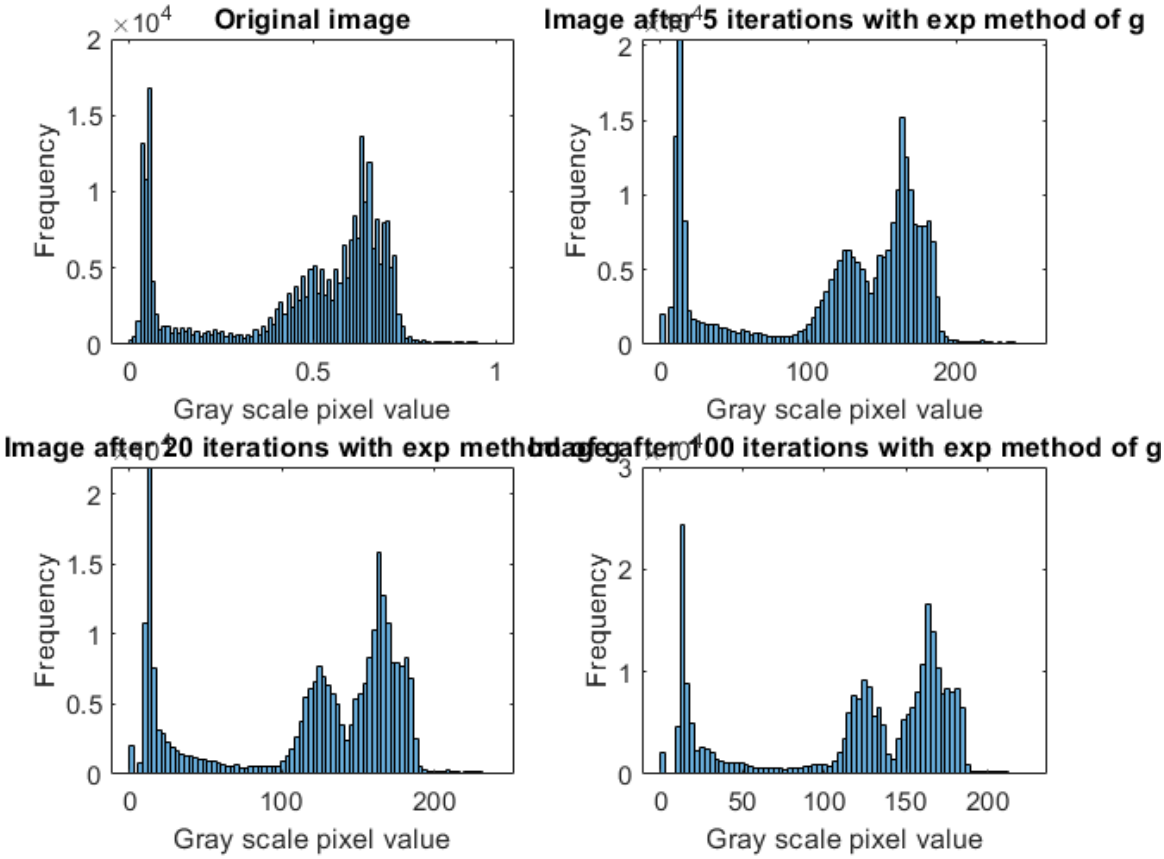

Figure 37: cameraman: Histograms with exponential method of g

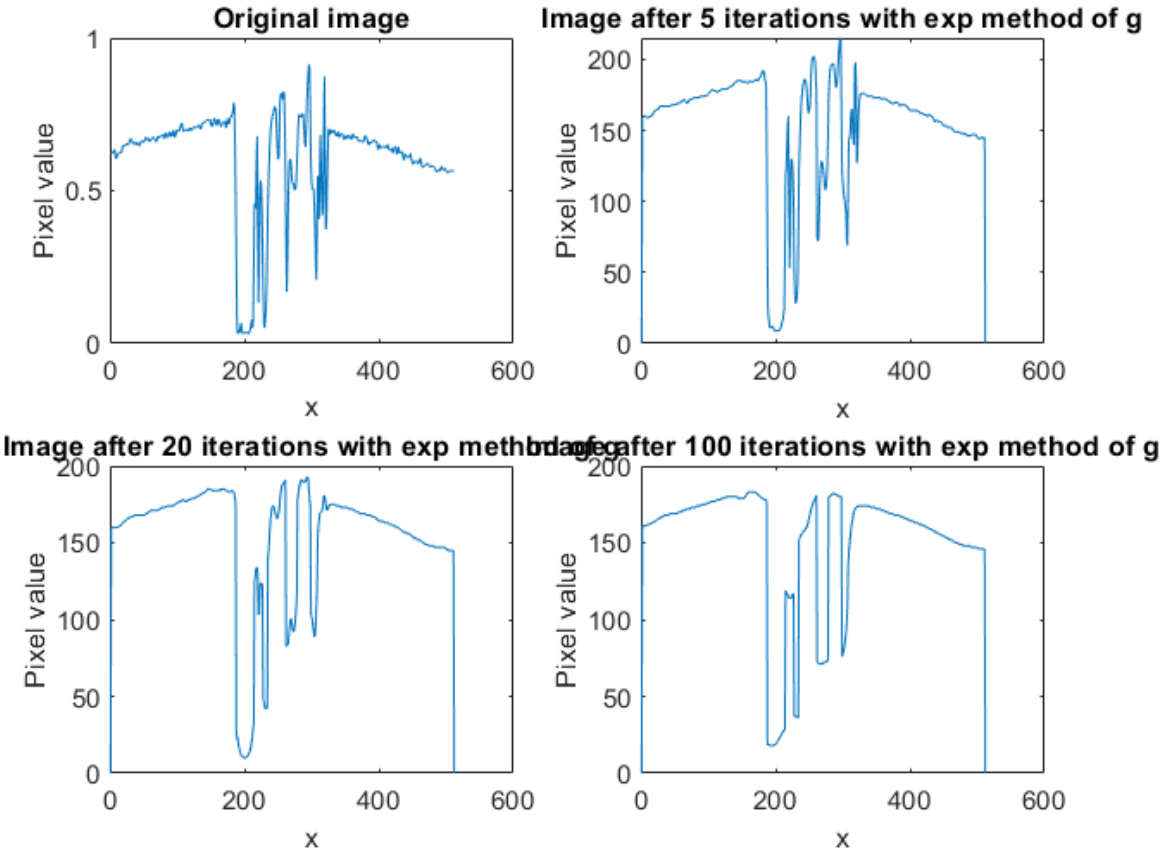

Figure 38: cameraman: Line y=128 through the images with exponential method of g

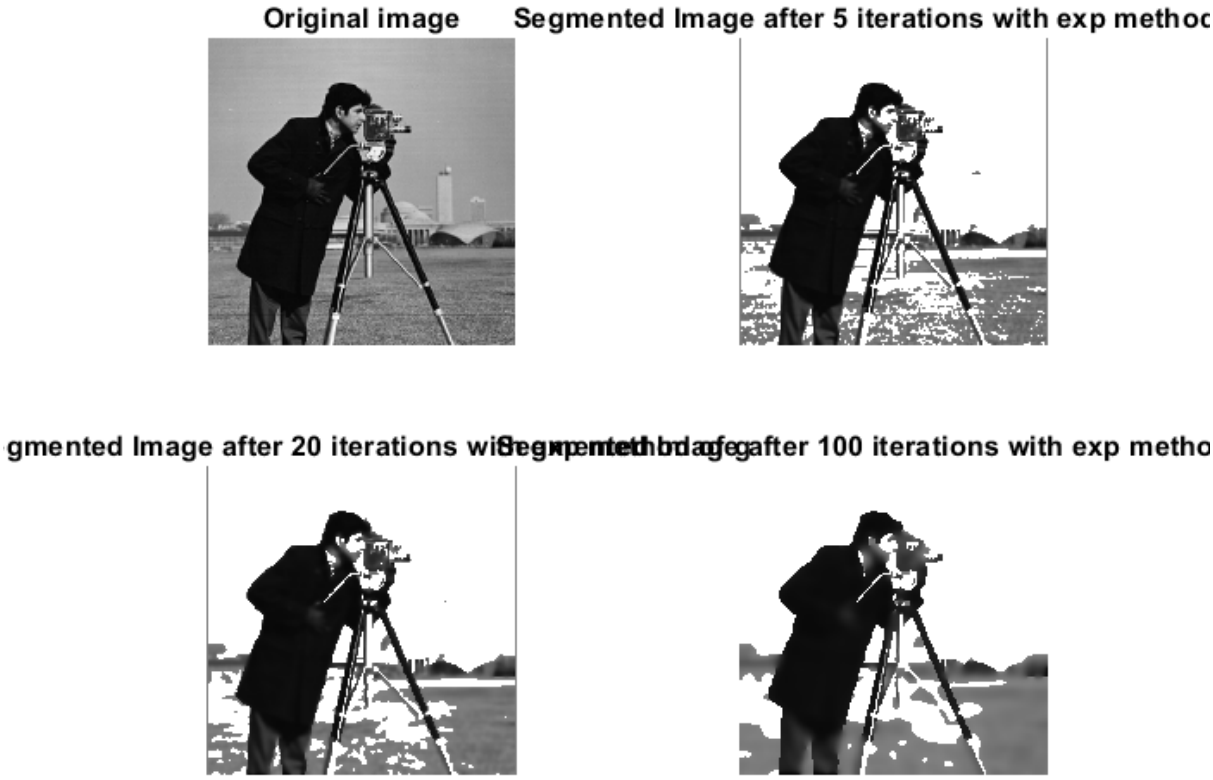

Figure 39: cameraman: Segmented images with exponential method of g

For cameraman image, anisotropic diffusion with inverse quadratic method of g(.)(k=10/255) results are displayed in Fig40. Histogram of results is shown in Fig41. Line plot for y=128 is displayed in fig42. The output images are segmented using 128 as threshold, the segmented images are shown in fig43.

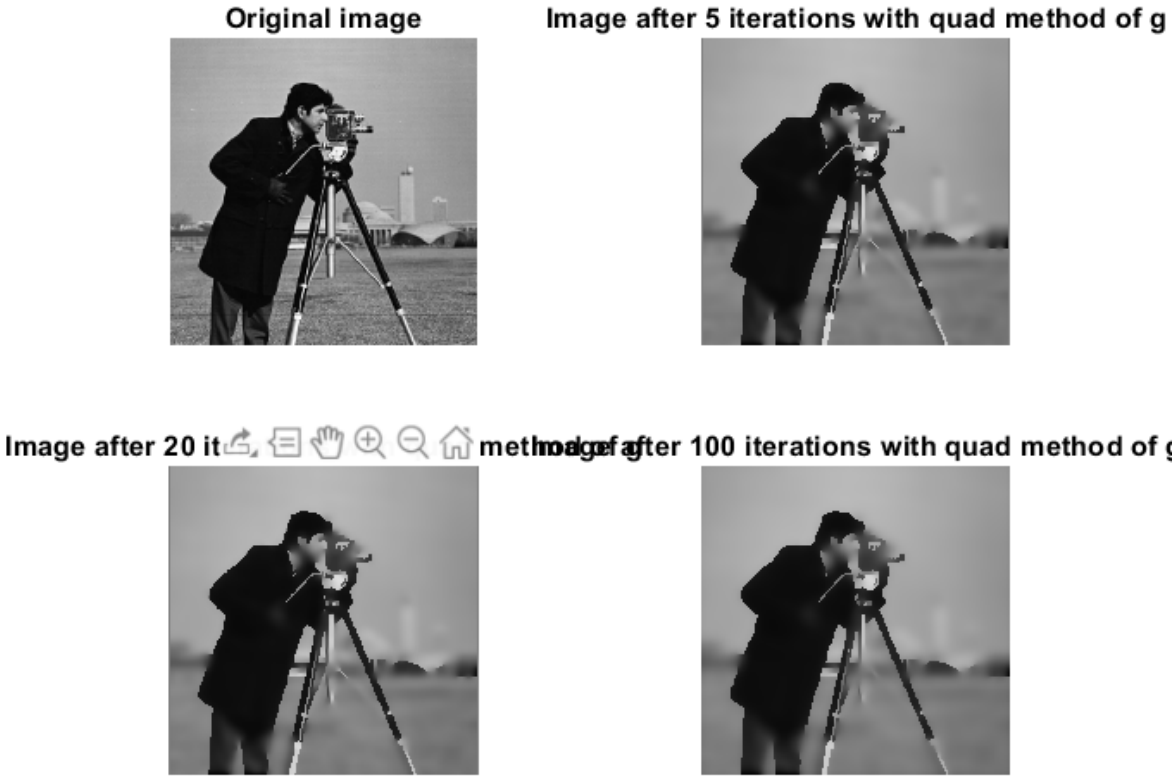

Figure 40: cameraman: Images with quadratic method of g

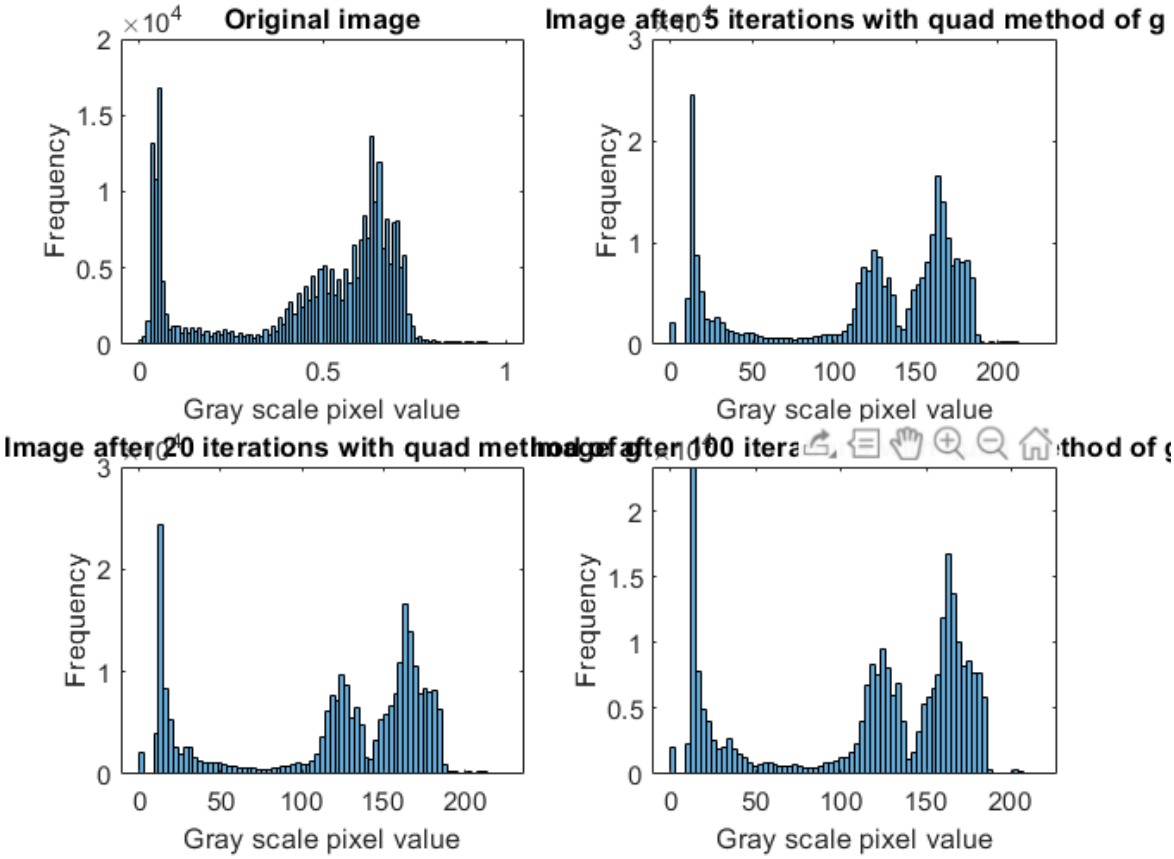

Figure 41: cameraman: Histograms with quadratic method of g

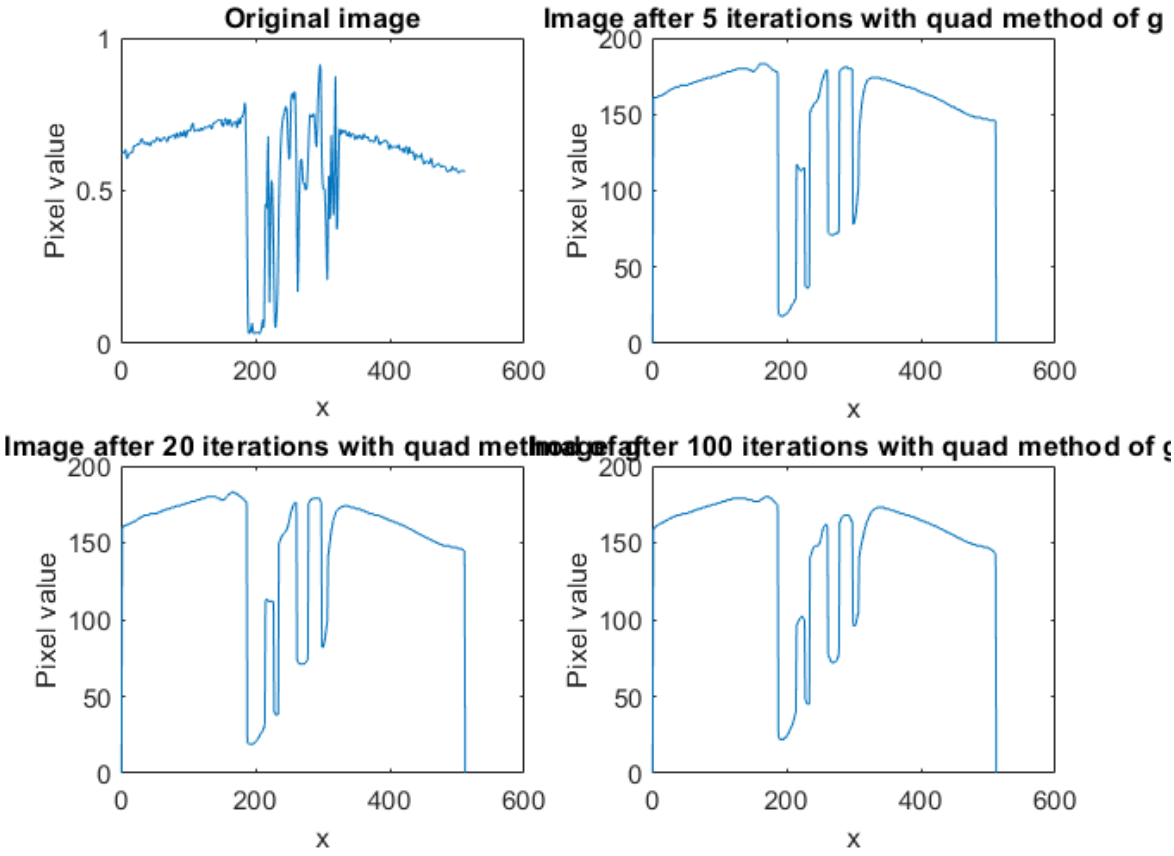

Figure 42: cameraman: Line y=128 through the images with quadratic method of g

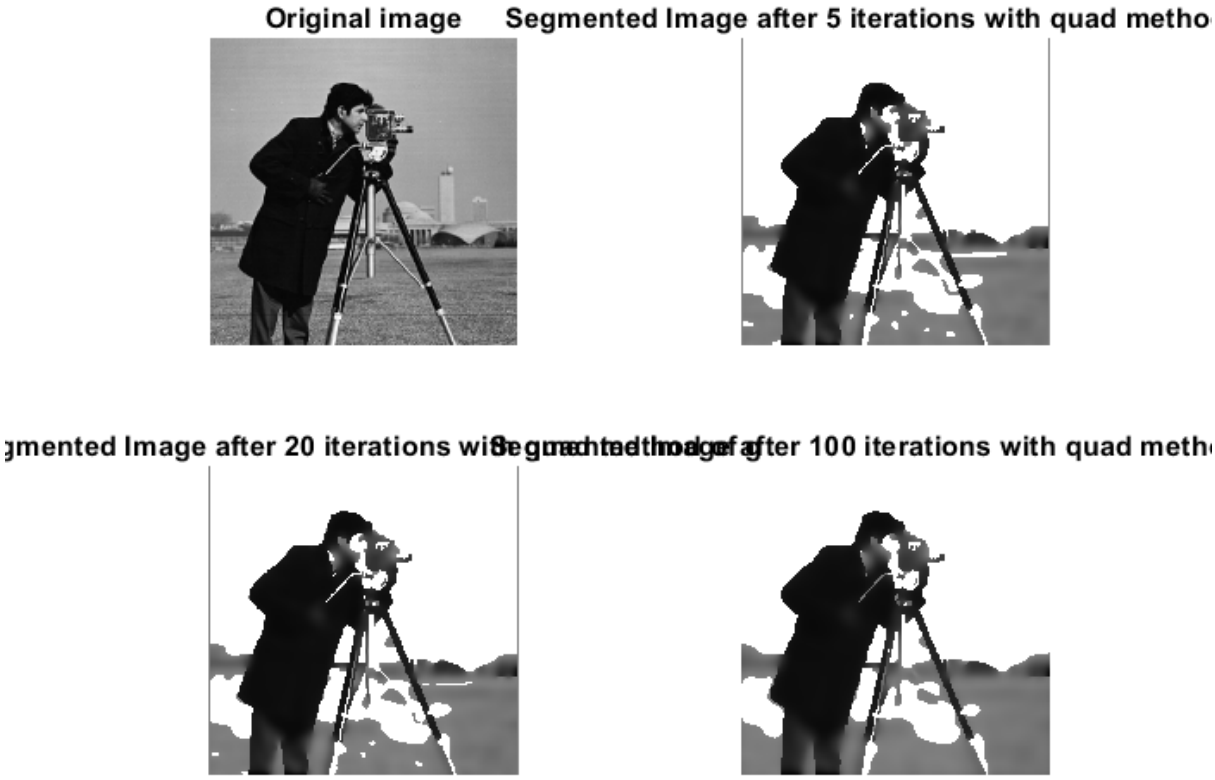

Figure 43: cameraman: Segmented images with quadratic method of g

Prob2(c)Discuss the following questions on your results of parts (a-b):

- How does the result change as you iterate? How does K affect the results?
  Observation: As we iterate more, we remove more noise on the image. The higher k, the image resolution gets coarser.

- How does g(.) affect the results (filtered and segmented)?
  Observation: With a trial of same k = 30/255, quadratic method of g() can remove more noise but get coarser resolution than exponential method.

- How does anisotropic diffusion run on "cwheelnoise" versus "cameraman"?
  Observation: cameraman is clean while cwheelnoise contains more noise, so it is more clear to see resolution get coarser with iteration goes on cameraman than on cwheelnoise.

4 Conclusion

After completing the necessary tasks to meet the objectives of this project, the following conclusions can be made:

- All nonlinear filters do not give the same performance with respect to noise removal when applied on the same image.
- In anistropic diffusion, the parameter such as k, type of method g() used and the number of times diffusion is applied iteratively have great effect on the amount of noise removed from an image.