CMPEN 555: Digital Image Processing II

Computer Project # 3:

Nonlinear Filtering and Anisotropic Diffusion

Naichao Yin, Wenrui Wang, Zekai Liu

Date:3/27/2020

A. Objectives

1. Learn how to use MATLAB for general image processing.
2. Do nonlinear filtering for the target images
3. Compare the results of different nonlinear filters
4. Do anisotropic diffusion filter for the target images
5. Change the parameters of anisotropic diffusion filter to get the influence of each parameter

B. Methods

1) Question one (Nonlinear Filtering)

Top-level Process Flow:

1. Input image: binary image "disk.gif"
2. Do five different nonlinear filtering to this binary image respectively, included:
   i. 5x5 mean filter
   ii. 5x5 median filter
   iii. 5x5 alpha-trimmed mean filter(α=0.25)
   iv. 5x5 sigma filter(σ=20)
   v. 5x5 symmetric nearest-neighbor mean filter
3. Repeat step 2 five times to get the 5 iterations results
4. For each result in step 3, give its gray-scale histogram
5. For each result in step 3, give its mean and standard deviation of the interior of the left largest disk region

The filter is one of the most important research objects in image processing, can highlight the useful information about the original signal through various combinations. Almost all applications involving features need to consider filter, even if it is developed to the high level of deep learning, etc., all kinds of filters, if well designed, can greatly improve the accuracy and efficiency of the entire algorithm.

There are no more than two kinds of filtering in the spatial domain, linear filtering and nonlinear filtering. The meaning of filtering is to operate on the pixels in a certain range around each pixel of the original image. The operation range is called “mask”. If the operation is only simple processing of the gray value of each pixel (such as multiplied by a weight) and finally get the summation, it is called linear filtering; If the calculation of pixel gray value is more complex than the simple calculation of the summation, it is called nonlinear filtering.

F1	F2	F3	F4	F5
F6	F7	F8	F9	F10
F1	F12	F13	F14	F15
F16	F17	F18	F19	F20
F21	F22	F23	F24	F25

Figure 1.1. 5x5 mask

5*5 Mean Filter:

$$F_{13} = \frac{1}{25} \times \sum_{i=1}^{25} F_i \tag{1.1}$$

Mean filtering is a commonly used method in image processing. From the perspective of the frequency domain, mean filtering is a low-pass filter(LPF), and the high-frequency signal will be truncated. The mean filter can help to eliminate the Gaussian additive white noise of the image and realize the functions of image smoothing and blur. The ideal method of mean filtering is to replace each pixel in the image with the average of its surrounding pixels(included itself).

Taking 5*5 mean filter as an example, we build our mean filter in file “meanFilter.m”, create a mask by setting the second input parameter of 5*5 all one’s matrix of type uint8. In function “meanFilter”, we first periodic padding the input image, then using this mask to slide over the input image, the principle of mean filter is shown in Figure 1 and Formula 1.1.

5*5 Median Filter:

$$\left(F_1, F_2, F_3, \ldots, F_{13}, \ldots, F_{25}\right) \tag{1.2}$$

$$\Big\Downarrow \quad \text{rank order}$$

$$\left(F_{(1)}, F_{(2)}, \ldots, F_{(13)}, \ldots, F_{(25)}\right) \tag{1.3}$$

$$\Big\Downarrow$$

$$F_{13} = \text{median}\left[F_1, F_2, F_3, \ldots, F_{13}, \ldots, F_{25}\right] = F_{(13)} \tag{1.4}$$

Median filtering is a kind of nonlinear smoothing filtering based on sorting statistics theory. The operation of applying median filtering is to first determine a mask containing multiple pixels and centered on one pixel, then sort the value of each pixel in the mask, and take the intermediate value as the new value of the center pixel. This algorithm is simple and the time complexity is low, but the median filter is not suitable for the image with many points, lines, and spikes. The median filter is good for removing the salt and pepper noise and edge, but not so good for the gaussian noise.

Taking 5*5 median filter as an example, we build our median filter in the file “medianFilter.m”, set the second input parameter that is the same as the mean filter. In function “meanFilter”, we first pad the input image periodically, then use the mask to slide on the input image and replace each pixel in the image with the median value of its surrounding pixels, the principle of mean filter is shown in Figure 1.1 and Formula 1.1.

5*5 Alpha-Trimmed Mean Filter:

$$F_{\text{mid}} = \left(\frac{1}{n - 2 * f\,\text{loor}(\alpha n)} \right) \sum_{i = f\,\text{loor}(\alpha n)+1}^{n - f\,\text{loor}(\alpha n)} F_{(i)} \tag{1.5}$$

$$F_{13} = \left(\frac{1}{25 - 2 * \mathit{floor}(25/4)} \right) \sum_{i = \mathit{floor}(25/4)+1}^{25 - \mathit{floor}(25/4)} F_{(i)} \tag{1.6}$$

$$\Downarrow$$

$$F_{13} = \left(\frac{1}{13} \right) \sum_{i=7}^{19} F_{(i)} \tag{1.7}$$

Alpha-trimmed mean filter is an order-statistics filter summarized by this Formula (1.5), where α is the trim parameter and the size of the window is n. After sorting the value of each pixel in the mask by Formula (1.2 and 1.3), we set n equal to 25 and α equal to 0.25 which dictates the 1/4 lowest and 1/4 highest values to be deleted before averaging the remaining pixel values $F_{(i)}$.

It can be seen from the Formula 1.5 that the performance of the algorithm affected largely by the parameter α. When the type of image noise is Gaussian noise, the parameter α should not be too large; while when the noise is salt and pepper noise, the parameter α should not be too small. The situation is even more complicated when the image contains some different noises. Thus, although the principle of the algorithm is relatively good, the satisfied filtering result cannot be got when the parameter α is not suitable.

Taking 5*5 alpha-trimmed mean filter as an example, we use MATLAB’s order-statistic function “ordfilt2” to construct our alpha-trimmed mean filter: F_K = ordfilt2(Mask, Order K), for a given n values $\{F_1, F_2, \ldots, F_N\}$, arrange them in order of value and take the element in the Kth position as the output of image filtering. In our project, we replace each element in input the image by the 7th to 19th element in the sorted set of neighbors to get 13 different images, and then average them to get the output image.

5*5 Sigma Filter:

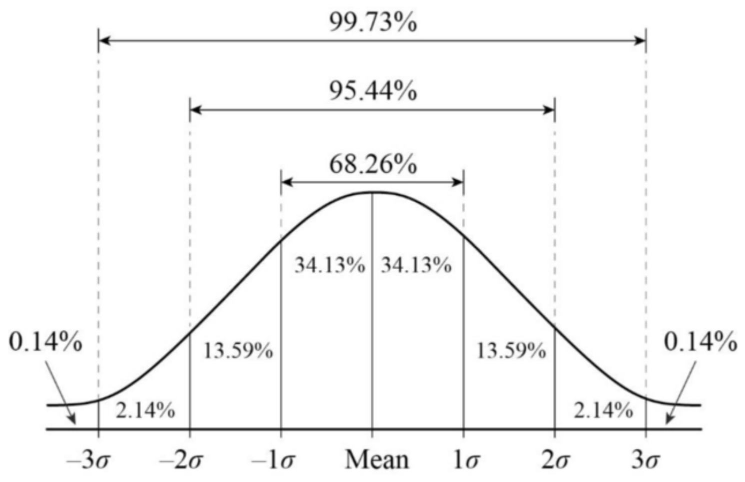


Figure 1.2. Gaussian curve

$$\left(F_{13-12},\ldots,F_{13},\ldots,F_{13+12}\right) \tag{1.8}$$

$$\Downarrow$$

$$F_{13}=\left(\frac{1}{N_c}\right)\sum_{i=-12}^{12}\delta_i F_{(13-i)} \tag{1.9}$$

$$\delta_i=\begin{cases}1 & \left|F_{13-i}-F_{13}\right|\le 40\\ 0 & \text{otherwise}\end{cases} \tag{1.10}$$

$$N_c \text{ is \# of points } F_{13-i} \text{ having } \delta_i=1 \tag{1.11}$$

The idea of the sigma filter is based on the assumption that the noise in the image is distributed Gaussian shaped. When using a binary image with lots of noise, the frequency of the intensity value to the left and right of the mean value also drops away in a Gaussian curve (Figure 1.2).

The Sigma filter replaces the intensity value of the current pixel by the average of all of the intensity values whose distance is smaller than Sigma to the current intensity value. The Sigma value defines the intensity interval within which the pixels in the neighborhood are supposed to be.

Taking 5*5 sigma filter as an example, we define the size of 25 and σ equal to 20 to apply the filter in our project. We build our sigma filter in the file “sigmaFilter.m”, compare each pixel in the mask with the middle pixel, check whether its difference is less than the sigma value, if so, bin that value into a new matrix, then count the number of these pixels, and take the average to replace the target pixel. The detail of this filter is shown in the Formula 1.8-1.11.

5*5 Symmetric Nearest-Neighbor Mean Filter:

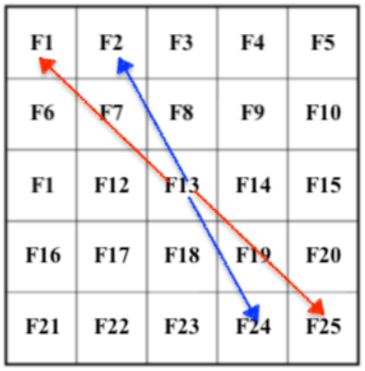


Figure 1.3. Symmetric pairs of pixels in SNN filter

$$[F_1, F_{25}] = \text{ point most similar to } F_{13} \tag{1.12}$$

$$F_{13} = \text{mean}\{[F_1, F_{25}], [F_2, F_{24}], [F_3, F_{23}], \ldots, F_{13}\} \tag{1.13}$$

The symmetric nearest-neighbor filter is an edge-preserving smoothing filter that compares the neighboring pixels in a symmetric fashion. By example of a 5×5 filter, the symmetric pairs of neighbor pixels around the center pixel, i.e. F_1-F_{25}, F_2-F_{24}, … (Figure 1.3), are inspected by selecting the gray values of those closest to the center pixel from each pair(Formula 1.12). Finally, the value of the center pixel is computed by the mean value of these selected neighborhood pixels (Formula 1.13).

We build our symmetric nearest-neighbor mean filter in the file “snn.m”, do a trick to operate the mask in Figure 3 easily achieve our goal. We create an 2*12 matrix, input F_1 to F_{12} in the first row and F_{14} to F_{25} in the second row, then subtract F_{13} from all these elements. Compare these results up and down to pick the value closest to 0 to get the 1*12 matrix. Finally, we add the center pixel F_{13} to this matrix to get the average.

Geometrically, this concept is edge-preserving as the filter votes for the neighboring pixels that most closely resemble the center pixel before taking the average value of the selected neighbors. This compensates the smoothing effect with retention of intensity gradients.

Iteration Process Flow:

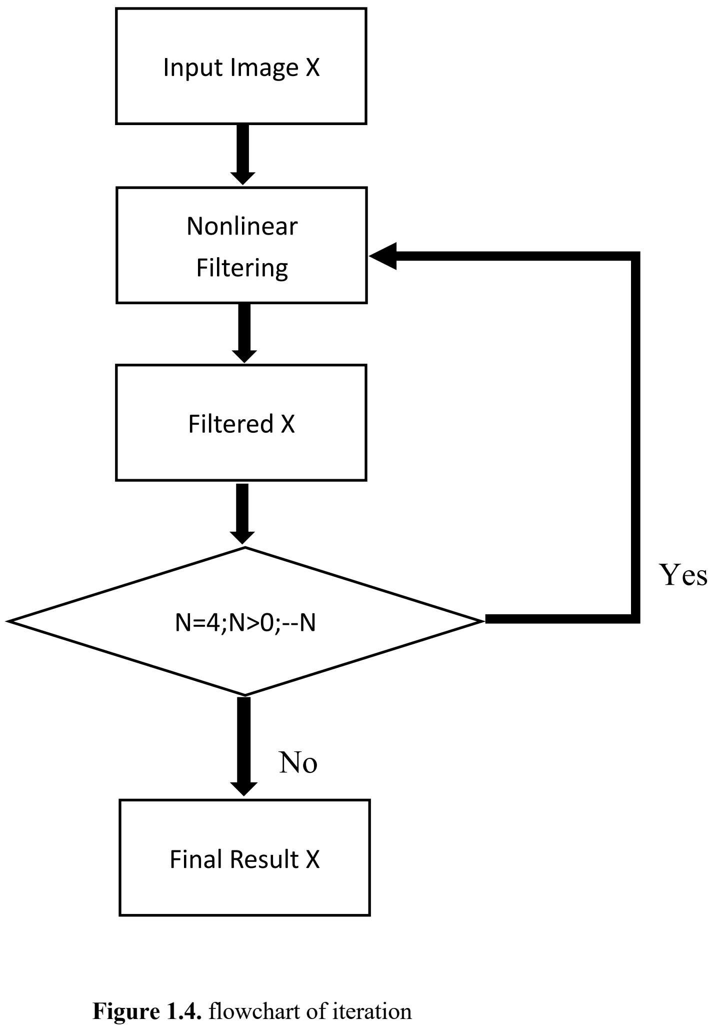


Figure 1.4. flowchart of iteration

We set N to 4 in order to do filtering iteration 5 times. For the nonlinear filter, only one of the above five filters is selected at a time to iterate over with the original image "disk". After the end of the loop, we get the gray-scale histogram of the final result image X by using our own histogram function "ph". Also, calculate the mean and standard deviation of the interior of the large disk region by manually define the region of the left bright disk in our function "m_and_d".

Mean and Standard Deviation Process Flow:

1. Select the target region manually by setting x from 1 to 180 to divide the white large disk from the small white disk, and then choose the large disk as our target region
2. According to the observation of the gray-scale histogram of the input image "disk.gif" after using the function "ph", we set threshold value of 194 to divide the foreground(large disk) and background and select the foreground pixels x_1-x_n
3. Calculate the mean μ of these foreground pixels in step 2

$$\mu = \frac{x_1 + x_2 + \ldots + x_n}{n} \tag{1.14}$$

4. Calculate the standard deviation σ of these foreground pixels in step 2

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu\right)^2} \tag{1.15}$$

MATLAB

There are 8 code files in question 1 : "proj3q1_1.m", "proj3q1_5.m", "meanFilter.m", "medianFilter.m", "sigmaFilter.m", "snnFilter.m", "ph.m", "m_and_d.m". The main code is in file "proj3q1_1.m", which is used to iterate one time over the input image of all these five nonlinear filters, and the file "proj3q1_5.m", which is used for five iterations. In these main files of question 1, we create our target 5*5 mean, median, sigma and symmetric nearest-neighbor mean filter using the function "meanFilter", "medianFilter", "sigmaFilter", "snnFilter". After 5 iterations, we using function 'ph' to get the gray-scale histogram of the output images, and 'm_and_d.m' to calculate the mean and standard deviation of the same foreground region we manually defined in these output images.

2) Question Two (Anisotropic Diffusion for Image Filtering)

1. Theory and Algorithms

A. Theory

(a) Anisotropic Diffusion

Anisotropic diffusion can be used for scale-space and edge detection. It can reduce image noise without removing significant parts of the image content, typically edges, lines or other details that are meaningful. This algorithm is an iterative nonlinear filtering method. Edges is preserved while regions are being smoothed. The smoothing process is motivated by scale space interpretation of image features. The family of images generated by Gaussian blurring plus isotropic diffusion can be represented by following formula.

$$I(x,y,t) \;=\; I_o(x,y) \;*\; G(x,y,t) \tag{1.16}$$

Where the left is original image at resolution t, I_0 is the original image and G is Gaussian blurring function with variance is t. The bigger t is, the coarser details will be preserved. Think of the image as a thermal field. Each pixel is regarded as a heat flow, and it is determined whether to diffuse to the surrounding according to the relationship between the current pixel and surrounding pixels. The point (x, y)'s isotropic diffusion can be represented as following:

$$\begin{aligned} I_t &= I_{xx} + I_{yy} \equiv \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \\ I_t &= \frac{\partial I}{\partial t} = \nabla \cdot \nabla I = div[\nabla I] \end{aligned} \tag{1.17}$$

The initial condition is $I(x, y, 0) = I_0(x, y)$.

Then we can introduce a conduction coefficient c(x, y, t) to force the flow in a certain way in order to permit tracking. The new anisotropic diffusion equation can be defined as following:

$$\begin{aligned} I_t &= \operatorname{div}[c(x,y,t)\nabla I] = \nabla \cdot [c(x,y,t)\nabla I] \\ I_t &= c(x,y,t)\nabla^2 I + \nabla c \cdot \nabla I \end{aligned} \tag{1.18}$$

Our goal is to smooth pixels inside regions and keep pixels in edges. So the conduction coefficient can be set as following:

$$c(x,y,t) = \begin{cases} 1, \text{(a.) inside regions ["conduct"]} \\ 0, \text{(b.) at region boundaries (edges). ["don't conduct"]} \end{cases} \tag{1.19}$$

To satisfy this property, we can design c(x, y, t) in this way.

$$\begin{aligned} & c(x,y,t) = g(\| \nabla I(x,y,t) \|) = g(\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}) \\ & g(\| \nabla I \|) = \exp\left\{-(\| \nabla I \| / k)^2\right\} \text{ or } g(\| \nabla I \|) = \frac{1}{1+(\frac{\| \nabla I \|}{k})^2} \end{aligned} \tag{1.20}$$

Where ∇I(x, y, t) is norm of gradient of *I* and two functions of g can satisfy that output of g is big with small input and small with big input. The first g is high-contrast edges favored and the last one is wide regions favored. These properties can be shown as figure 2.1.

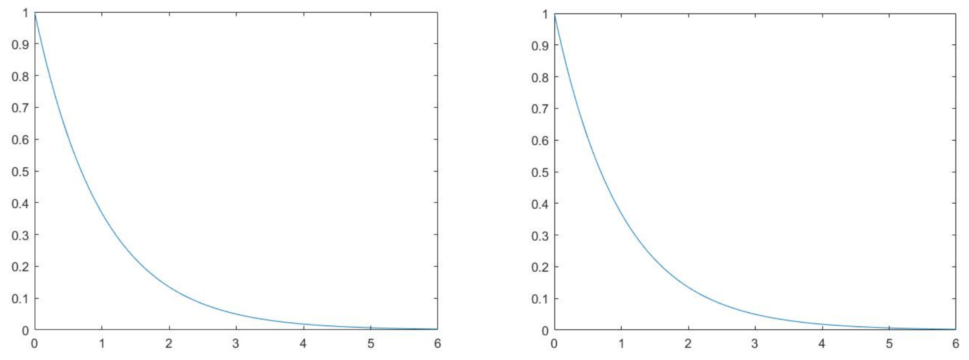


Figure 2.1 plots of g(x)=exp(-x) and g(x)=1/(1+x^2)

All maxima appear only in *I*(x, y, 0) and no new edges created with t increasing.

In digital image, the anisotropic diffusion can be represented as following:

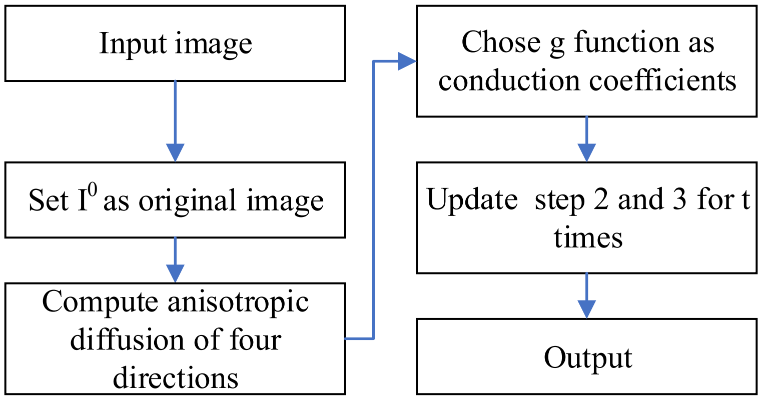
$$\begin{aligned} I_{i,j}^{t+1} &= I_{i,j}^{t} + \lambda\left[c_N^t \nabla_N I + c_S^t \nabla s I + c_E^t \nabla_E I + c_W^t \nabla_W I\right]_{i,j}^{t} \\ \nabla_N I_{i,j}^t &= I_{i,j+1}^t - I_{i,j}^t \\ \nabla_S I_{i,j}^t &= I_{i,j-1}^t - I_{i,j}^t \end{aligned} \tag{1.21}$$

The conduct coefficients are updated every iteration t:

$$\begin{aligned} c_N^t(i,j) &= g\left(\left|\nabla_N I_{i,j}^t\right|\right) \\ c_S^t(i,j) &= g\left(\left|\nabla_S I_{i,j}^t\right|\right) \end{aligned} \tag{1.22}$$

B. Algorithms

Our goal is to complete anisotropic-diffusion filter for image "cwheelnoise" and "cameraman" image. The process of anisotropic-diffusion filter is shown as figure 2.2.



Input image
Set I^0 as original image
Compute anisotropic diffusion of four directions
Chose g function as conduction coefficients
Update step 2 and 3 for t times
Output

Figure 2.2 Flow chart of algorithm

By changing the value of k in function g, we can focus on different parts of images. The smaller the value of k, the smoother the result, the less difficult it is to retain edges. In question a, if we want to segment out of the gray "spokes" component of the wheel by manual thresholding, we need to observe the plot of the line y = 128. The reason is that the most points in this line are belong to "spokes" components and "spokes" is in the middle. Therefore, we can design filter based on it. In this question, [80, 100] is a proper choice.

MATLAB

There are 6 code files in our project: Anisodiff.m, Anisodiff2.m, findspokes.m, main.m, main2.m, and ph.m. Function “Anisodiff.m” is used for doing Anisotropic Diffusion operation whose g(.) is exponential. Function “Anisodiff2.m” is used for doing Anisotropic Diffusion operation whose g(.) is inverse quadratic. Function “findspokes” is used for segmenting out spokes by applying different threshold. Function “ph.m” is used for plotting gray-scale histogram. “main.m” and “main2.m” are two main functions of Question(2)(a) and Question(2)(b). Directly running these two functions can get the final results. Function ‘main.m’ read image “cwheelnoise.gif” and call “Anisodiff.m” ,”Anisodiff2.m”, “ph.m” and “findspokes.m” to get the result(Figure 2.3 to Figure 2.14) after Anisotropic Diffusion operation, segment out “spokes” and also show grey scale histogram. Function “main2.m” read image “cameraman.tif” and call“Anisodiff.m”,”Anisodiff2.m” to get final result(Figure 2.15).

C. Result

1) Question one

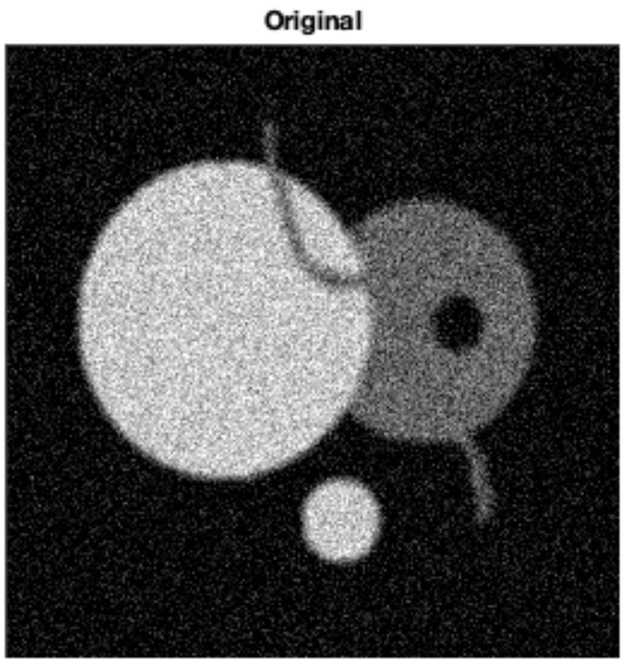


Figure 1.5. Original Image of disk

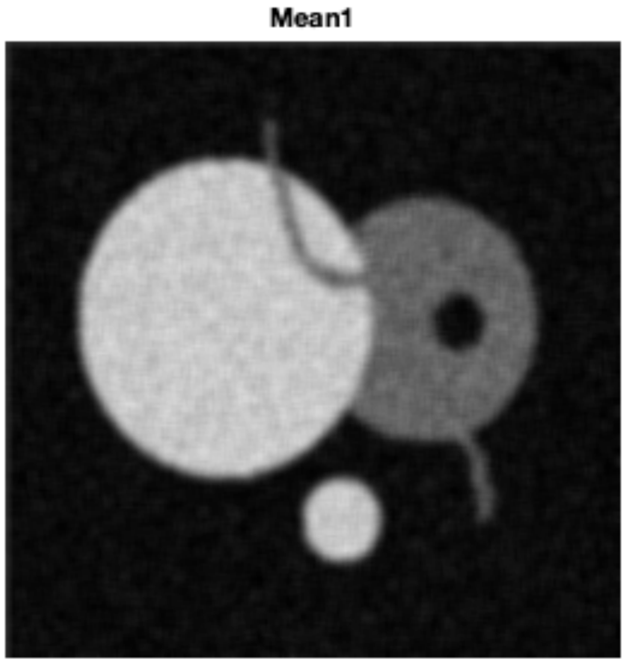


Figure 1.6. Image after 5*5 Mean filter

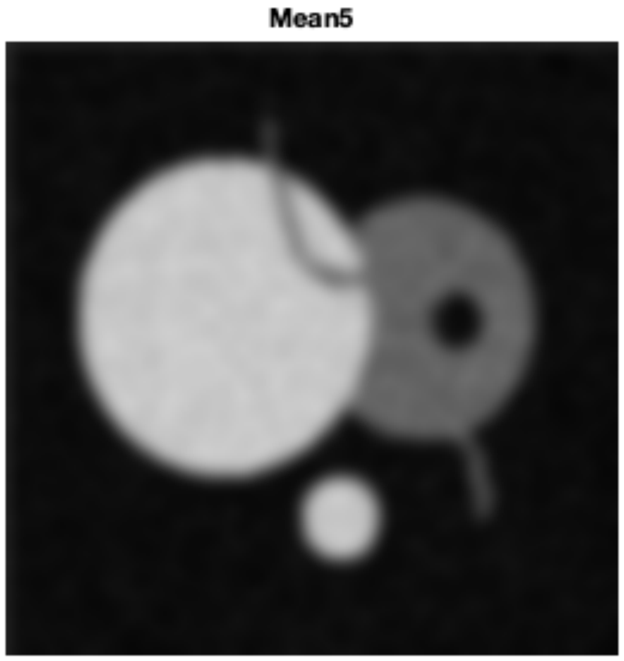


Figure 1.7. Image after 5 iterations of 5*5 Mean filter

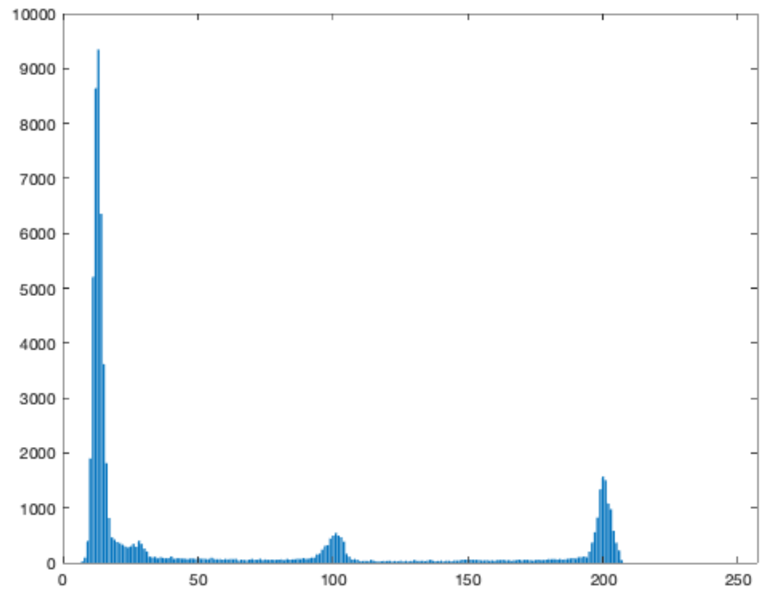


Figure 1.8. Histogram of image after 5 iterations of 5*5 Mean filter

As can be seen from Figure 1.6, the image becomes blurred and the noise is not effectively removed after one iteration. The mean filtering itself has inherent defects, that is, it cannot protect the image details well, especially the result after 5 iterations in Figure 1.7. Meanwhile, it also destroys the details of the image, so that the image becomes fuzzy, but the noise seems to have been improving.

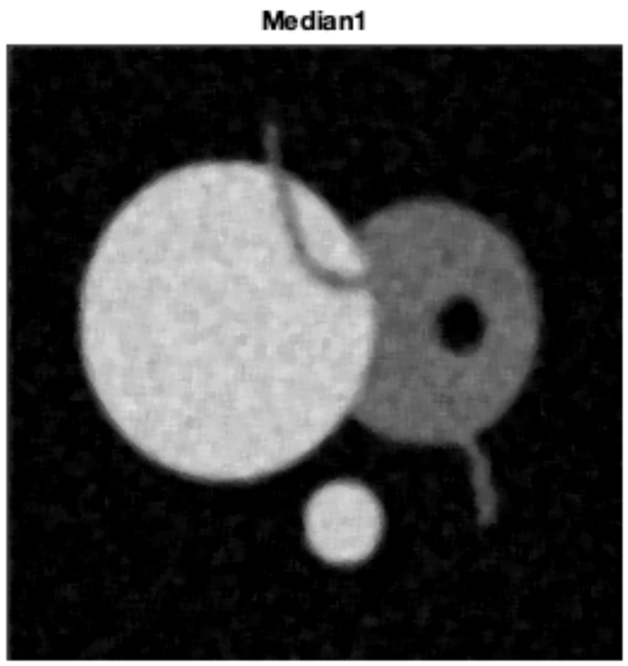


Figure 1.9. Image after 5*5 Median filter

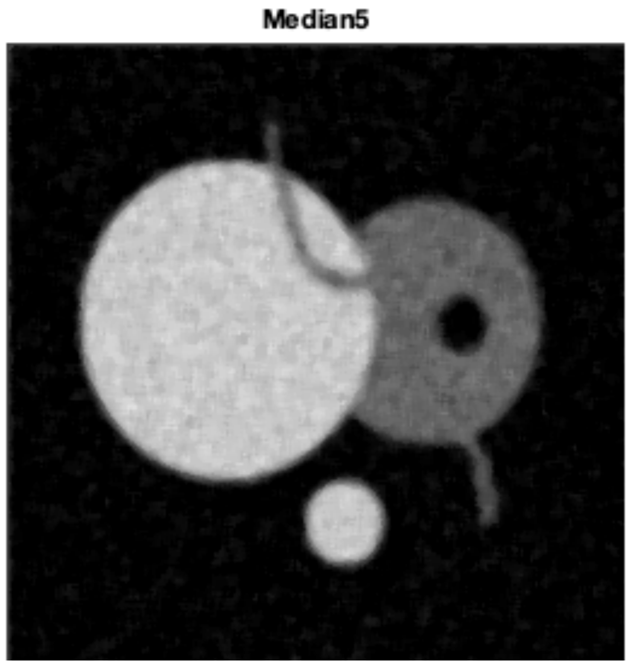


Figure 1.10. Image after 5 iterations of 5*5 Median filter

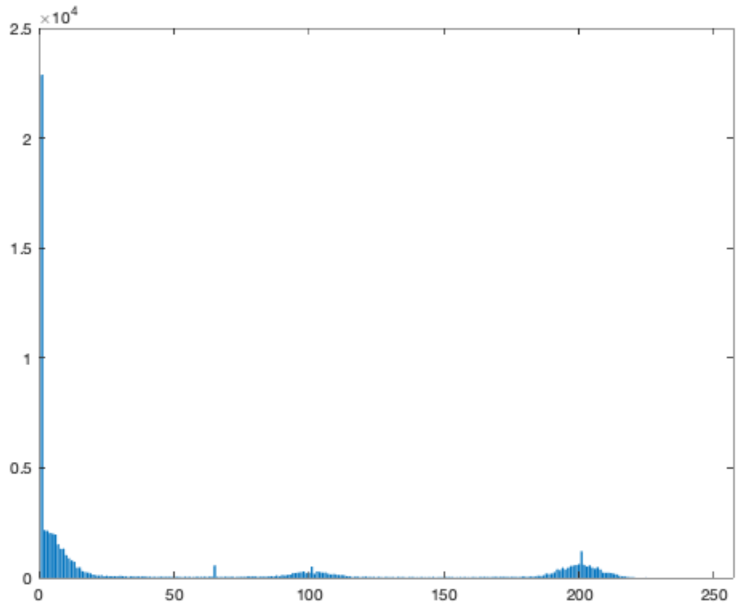


Figure 1.11. Histogram of image after 5 iterations of 5*5 Median filter

Unlike mean filtering, median filtering is good for preserving the edge sharpness, but it washes away the texture in the homogeneous medium region(Figure 1.9). This nonlinear filtering mainly takes advantage of the fact that the median is not affected by the maxima and minima of the distribution sequence, but it cannot remove the noise in the original image, even after five iterations, the noise still exist prominently(Figure 1.10).

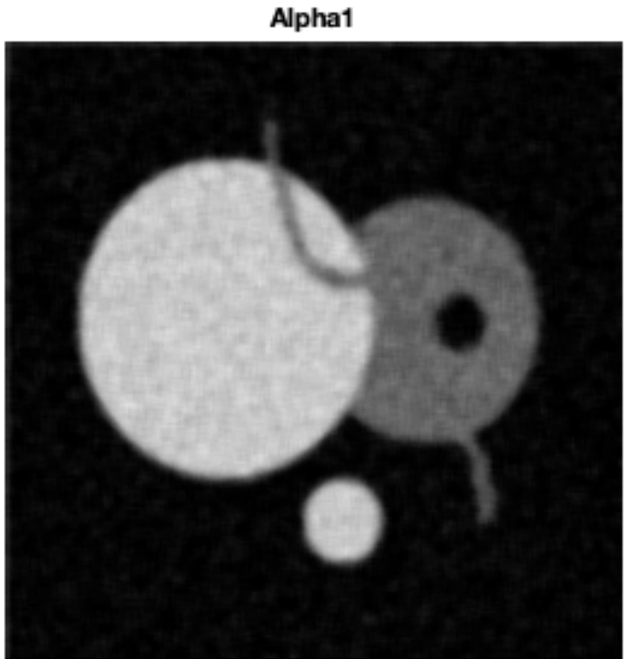


Figure 1.12. Image after 5*5 Alpha-Trimmed Mean filter

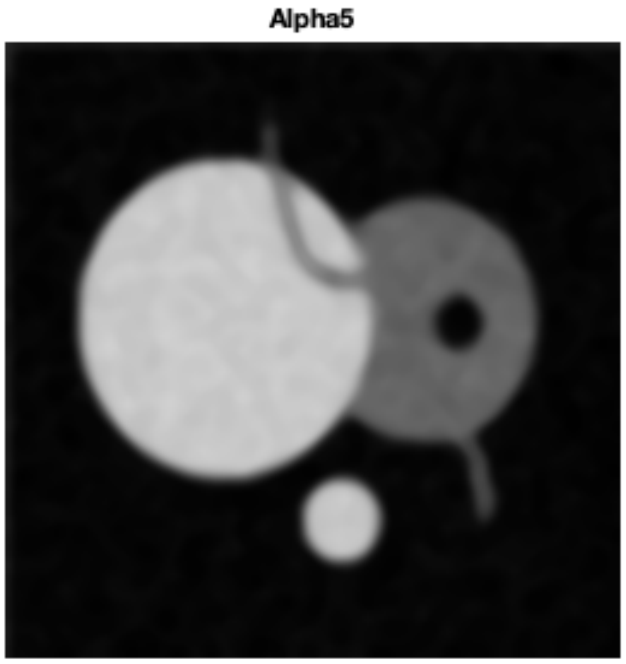


Figure 1.13. Image after 5 iterations of 5*5 Alpha-Trimmed Mean filter

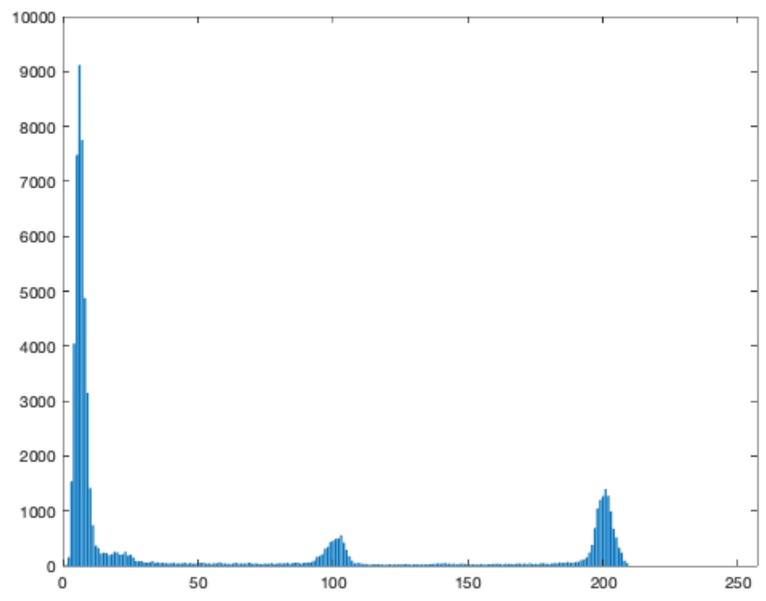


Figure 1.14. Histogram of image after 5 iterations of 5*5 Alpha-Trimmed Mean filter

As seen from the visual effects in Figure 1.13, the alpha-trimmed mean filter shows better filtering performance, enabling a sharper edge of the original image. The bimodal peak in the histogram is full shows that the information of the image is better protected by this nonlinear filter, the ability to protect the edge of the image also to be improved and can filter the noise effectively. In short, this kind of nonlinear filtering is the most suitable method to remove the target noise.

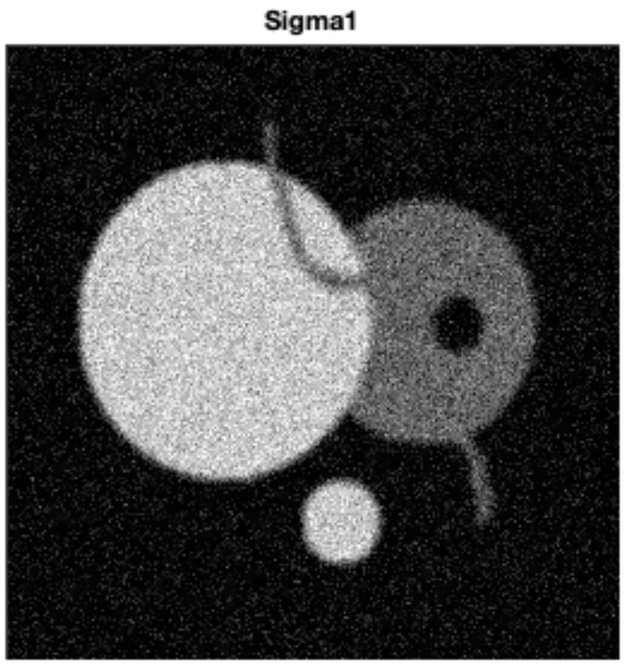


Figure 1.15. Image after 5*5 Sigma filter

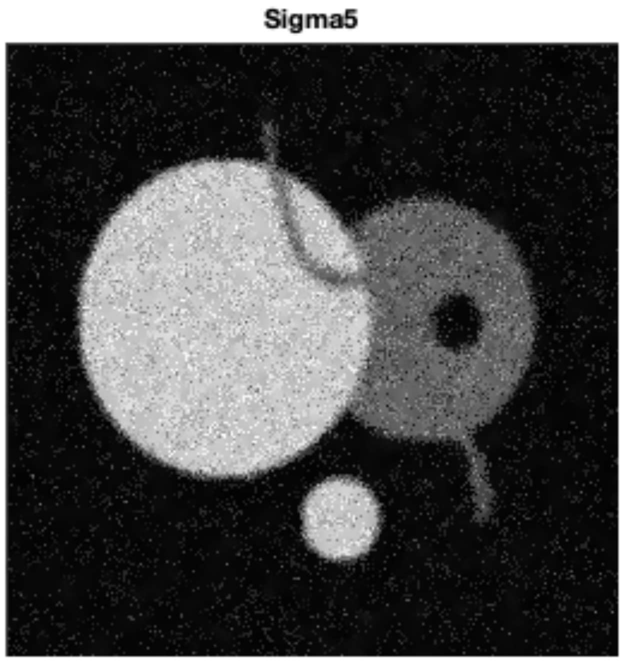


Figure 1.16. Image after 5 iterations of 5*5 Sigma filter

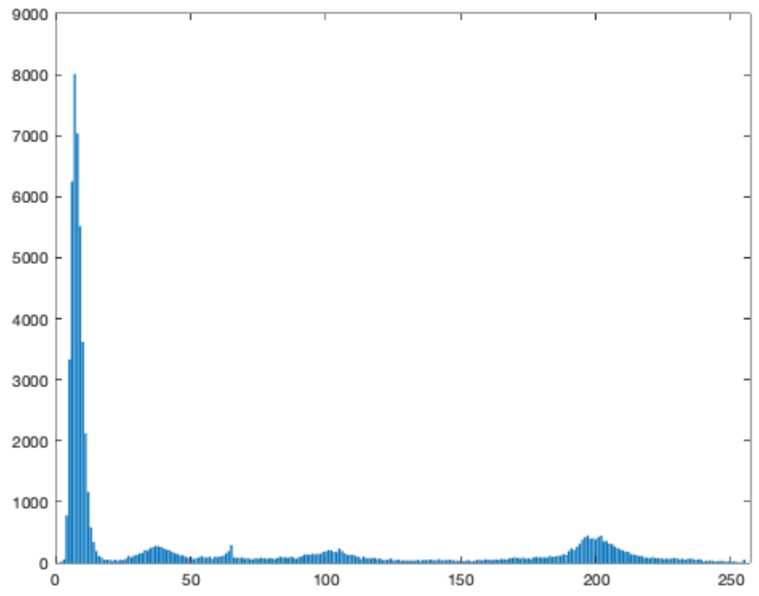


Figure 1.17. Histogram of image after 5 iterations of 5*5 Sigma filter

The result of this filtered image after several iterations is similar to that of the median filter(Figure 1.16). Although it keeps edges and details, the trade-off is that it retains a lot of noise in the image, filter strength is weaker along the borders and it is time-consuming. Also, we see two weird peaks in the histogram with the grayscale approaching 50, all those above shows that the sigma filter cannot deal with this noise well(Figure 1.17).

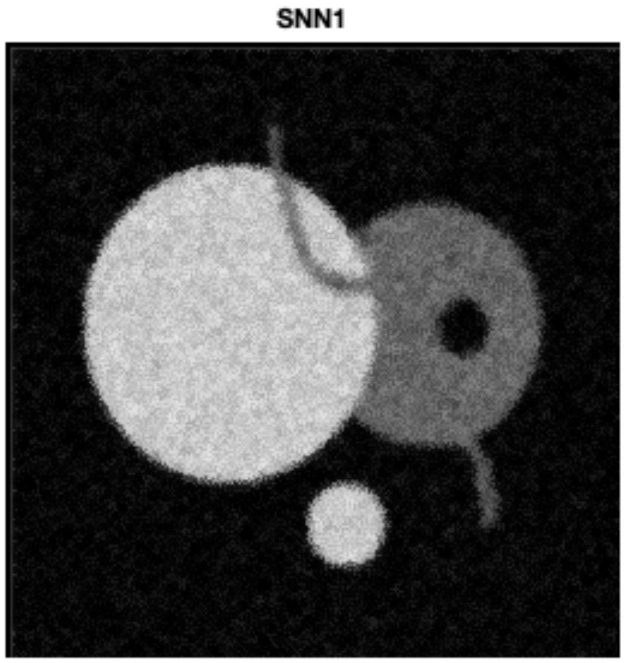


Figure 1.18. Image after 5*5 Symmetrix Nearest-Neighbor Mean filter

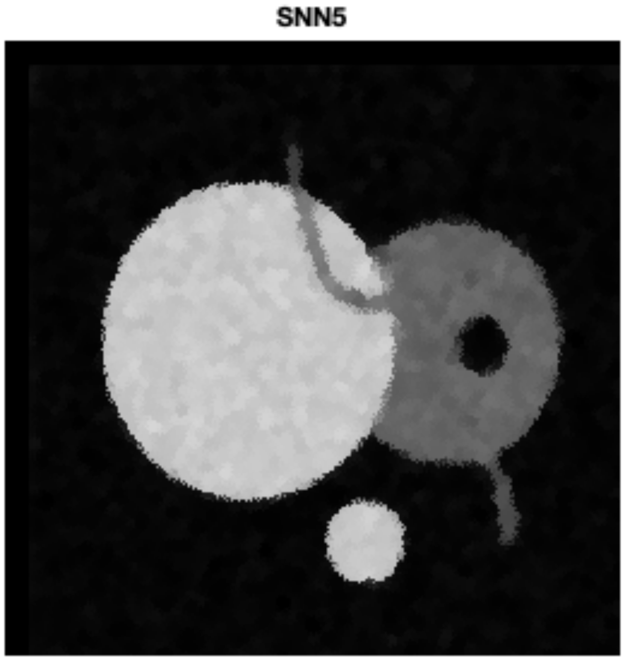


Figure 1.19. Image after 5 iterations of 5*5 Symmetrix Nearest-Neighbor Mean filter

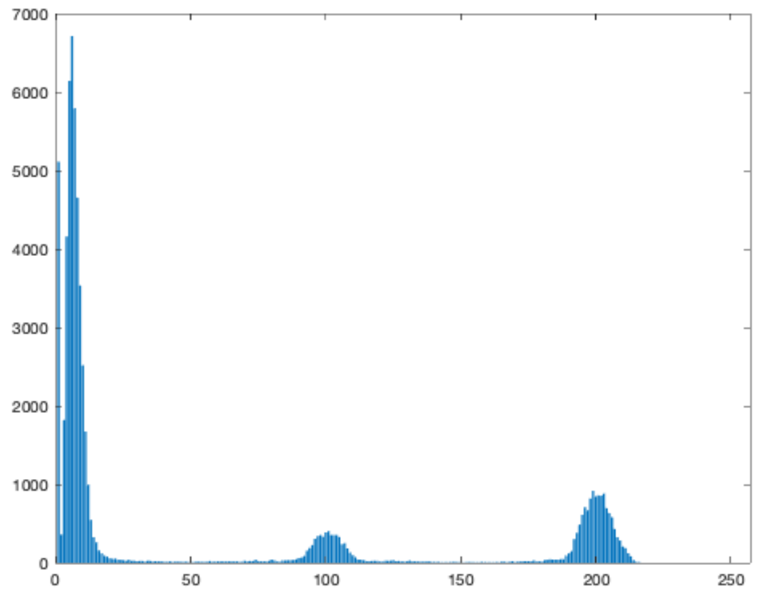
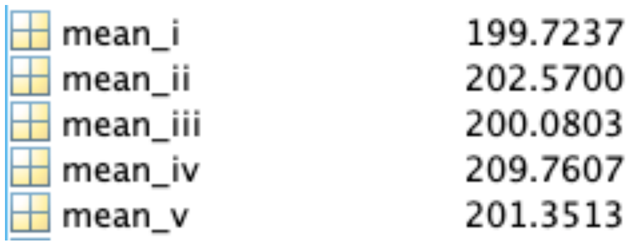


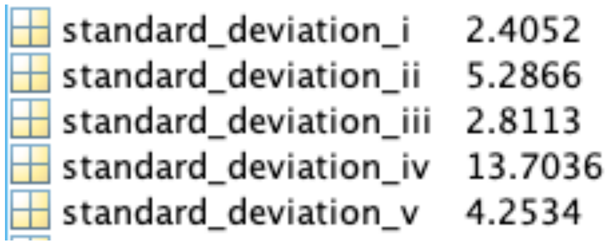
Figure 1.20. Histogram of image after 5 iterations of 5*5 Symmetrix Nearest-Neighbor Mean filter

As seen from the visual effects in Figure 1.18, the symmetric nearest neighbor filter has relations to the mean and median filters with desirable edge-preserving properties, and also it has a fast calculation speed than previous filters. Unfortunately, the SNN is susceptible to local patch-like artifacts after five iterations and these edges are partially distorted and become irregularly jagged(Figure 1.19).



mean_i	199.7237
mean_ii	202.5700
mean_iii	200.0803
mean_iv	209.7607
mean_v	201.3513

Figure 1.21. Mean of target region of all these five nonlinear filters



standard_deviation_i	2.4052
standard_deviation_ii	5.2866
standard_deviation_iii	2.8113
standard_deviation_iv	13.7036
standard_deviation_v	4.2534

Figure 1.22. Standard deviation of target region of all these five nonlinear filters

According to all the means and standard deviations of the region that we manually defined, we can result that the image filtered by sigma filter is brighter than filtered by any other, and the image passed by mean filter becomes the darkest one. Also, we can evaluate the performance of denoising by judging the value of the standard deviation, theoretically, the smaller the standard deviation, the better the information keeping. Therefore, the 5*5 alpha-trimmed mean filter is the best option for noise suppression in “disk” image.

2) Question two

The original image, histogram and line y=128 are shown as figure 2.3, 2.4, 2.5.

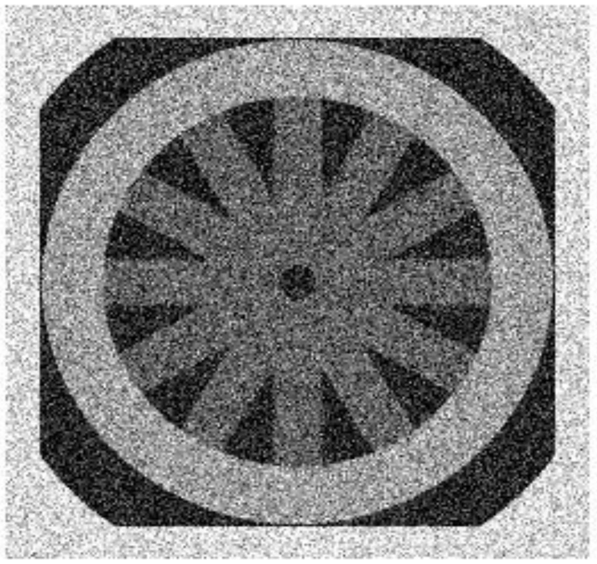


Figure 2.3 Original image

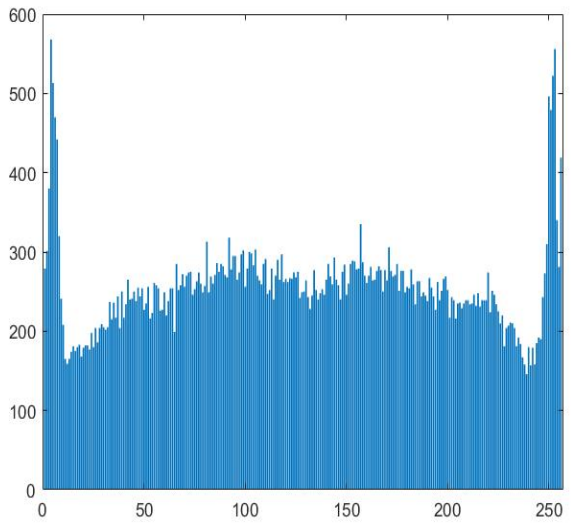


Figure 2.4 Original image histogram

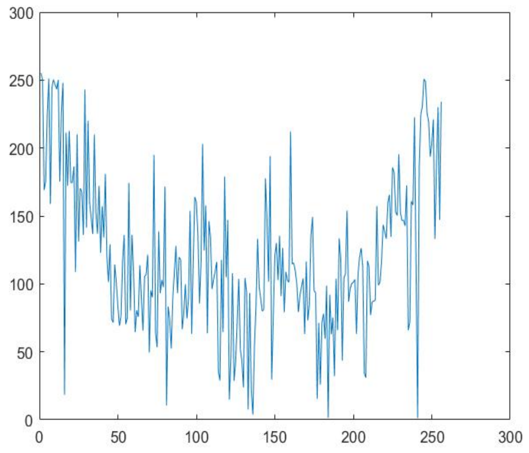


Figure 2.5 Original image's line y=128

The image of 5 iterations, histogram and line y=128 are shown as figure 2.6, 2.7, 2.8. With g is exponential function and inverse function.

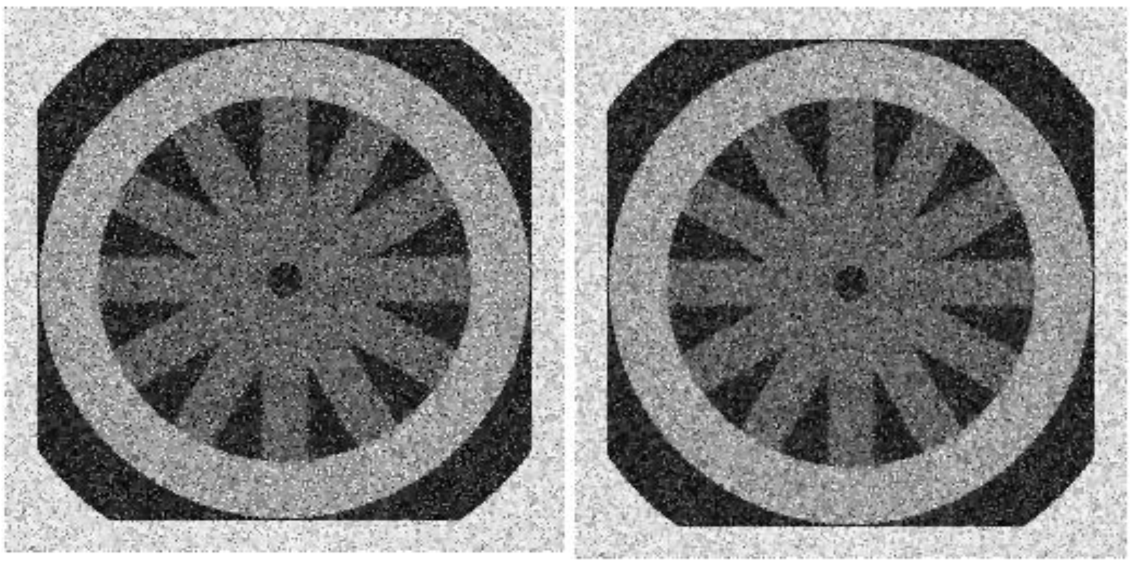


Figure 2.6 5 iterations image

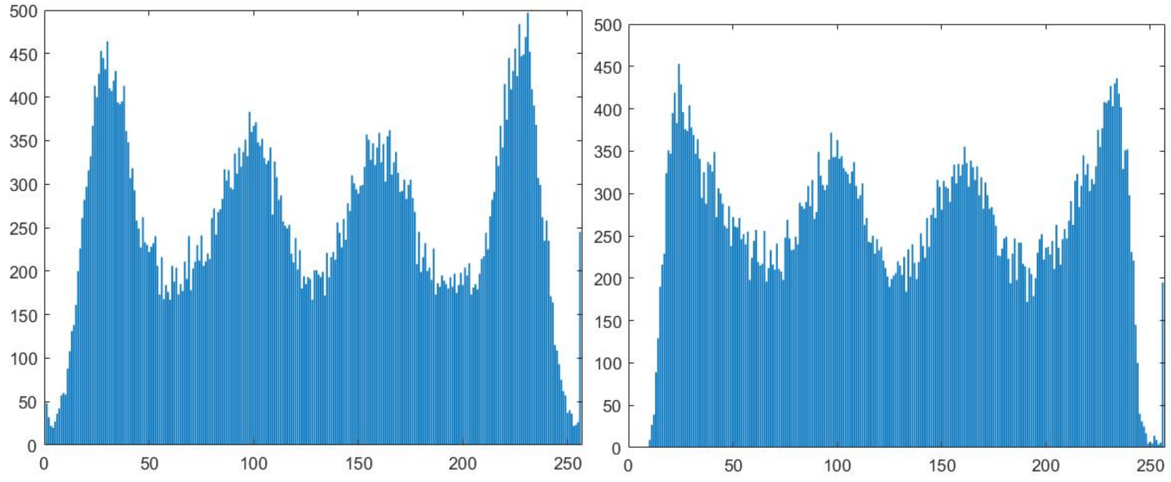


Figure 2.7 5 iterations image histogram

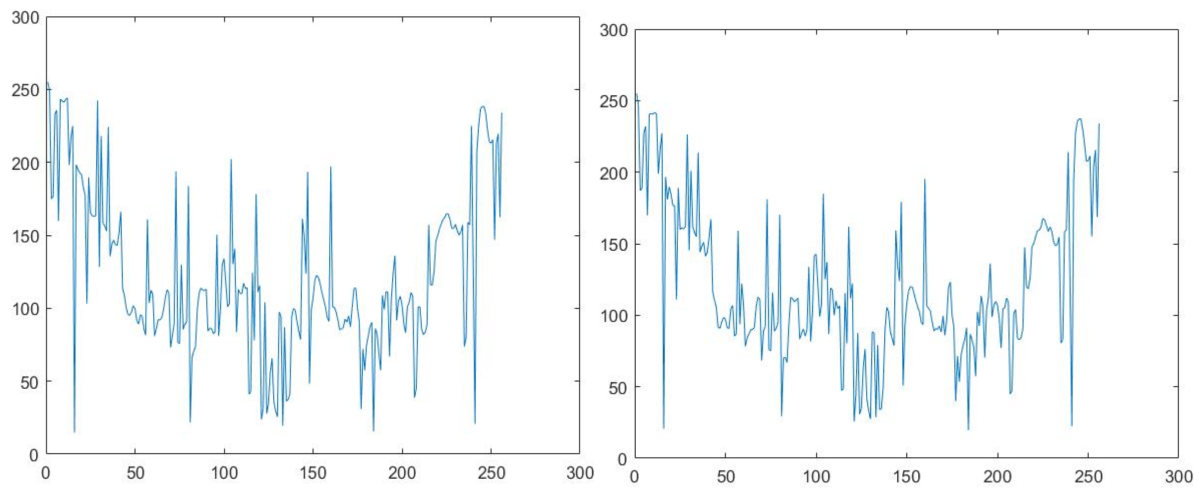


Figure 2.8 5 iterations image's line y=128

The image of 20 iterations, histogram and line y=128 are shown as figure 2.9, 2.10, 2.11. With g is exponential function and inverse function.

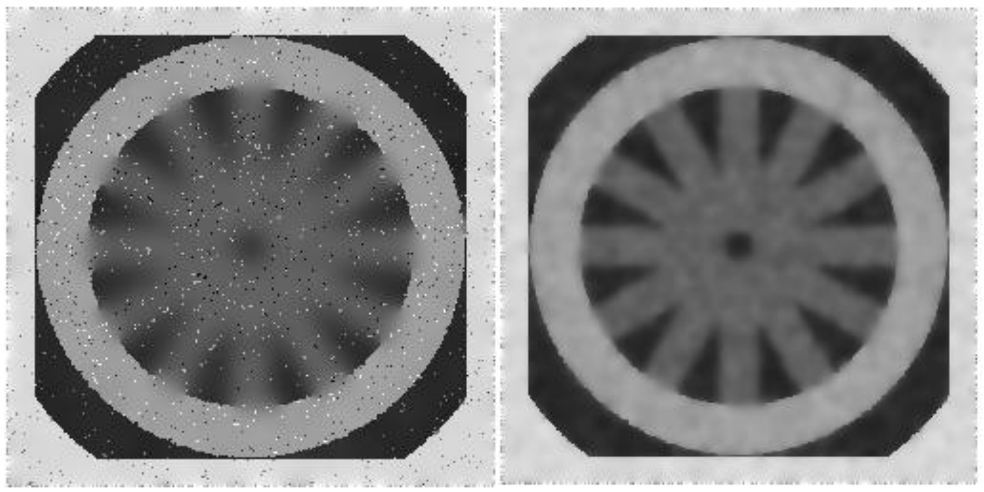


Figure 2.9 20 iterations image

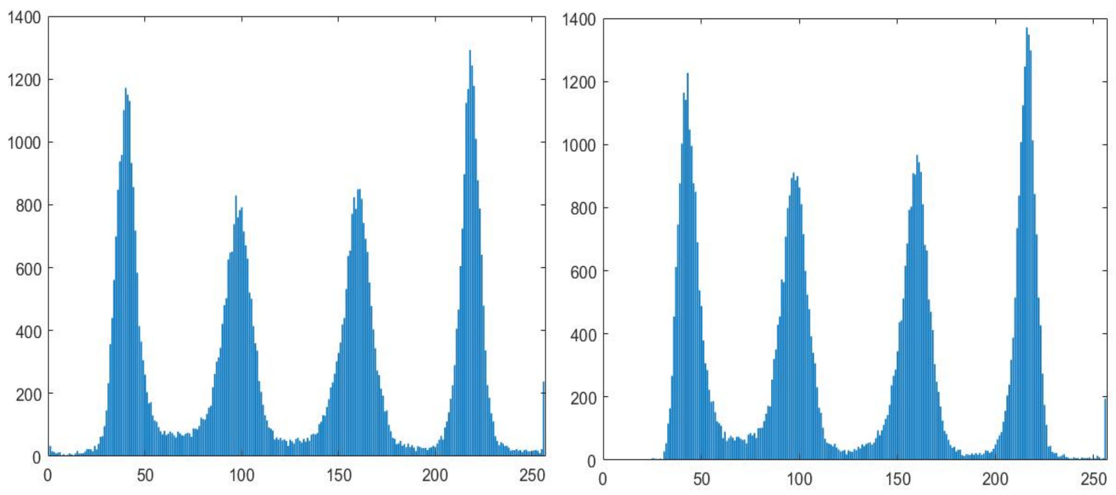


Figure 2.10 20 iterations image histogram

The image of 100 iterations, histogram and line y=128 are shown as figure 2.11, 2.12, 2.13. With g is exponential function and inverse function.

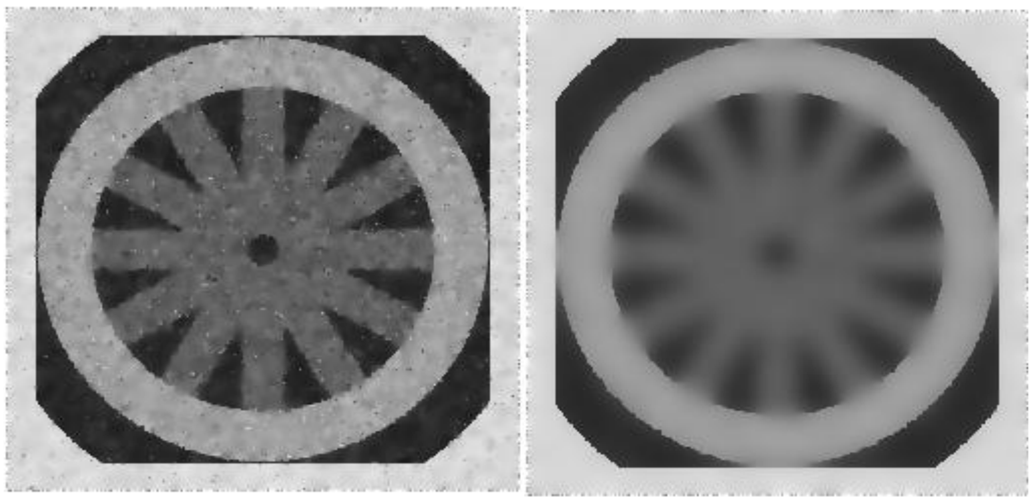


Figure 2.11 100 iterations image

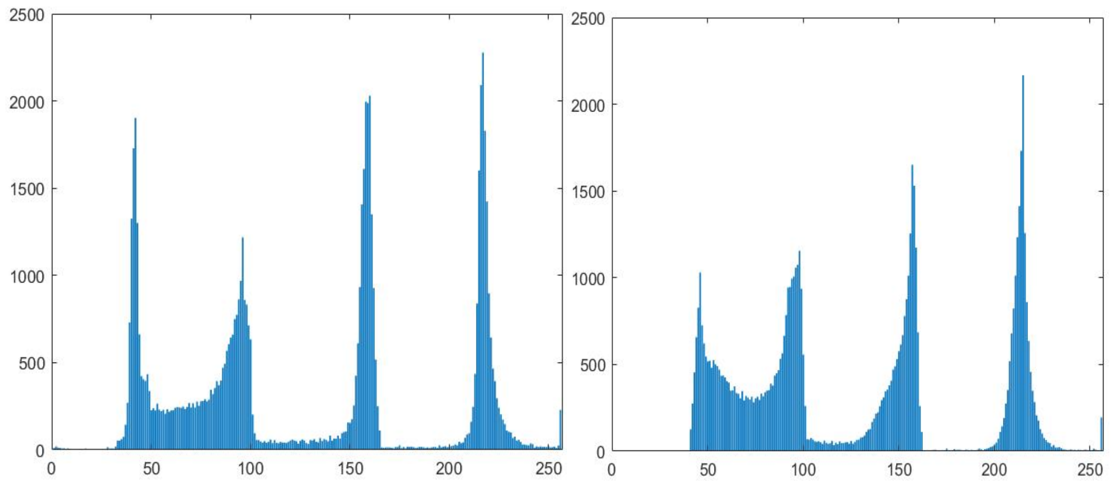


Figure 2.12 100 iterations image histogram

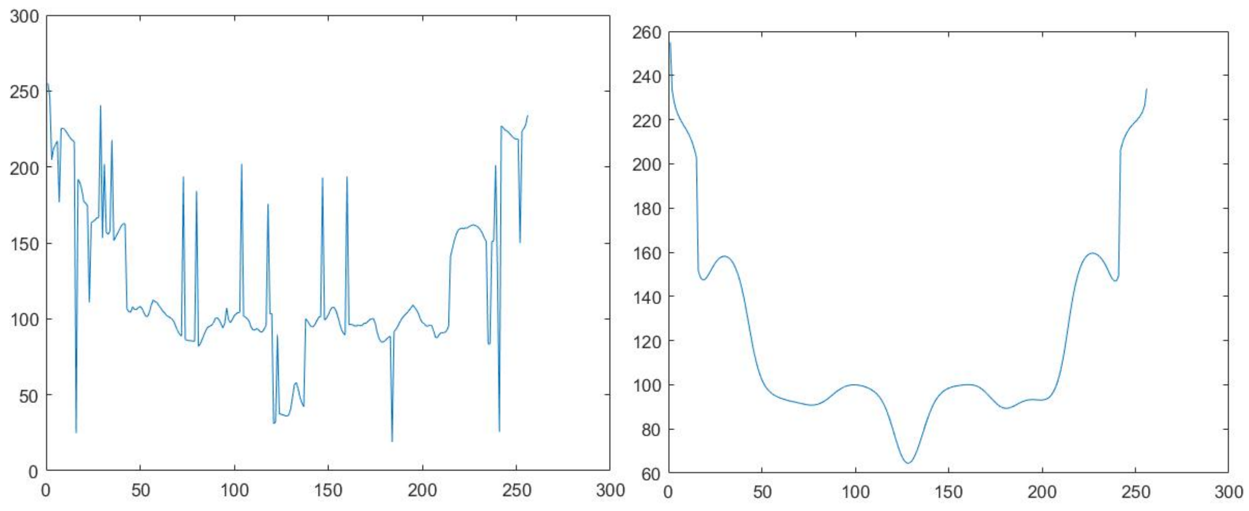


Figure 2.13 100 iterations image's line y=128

The "spokes" components segmented with different g are shown as figure 2.14.

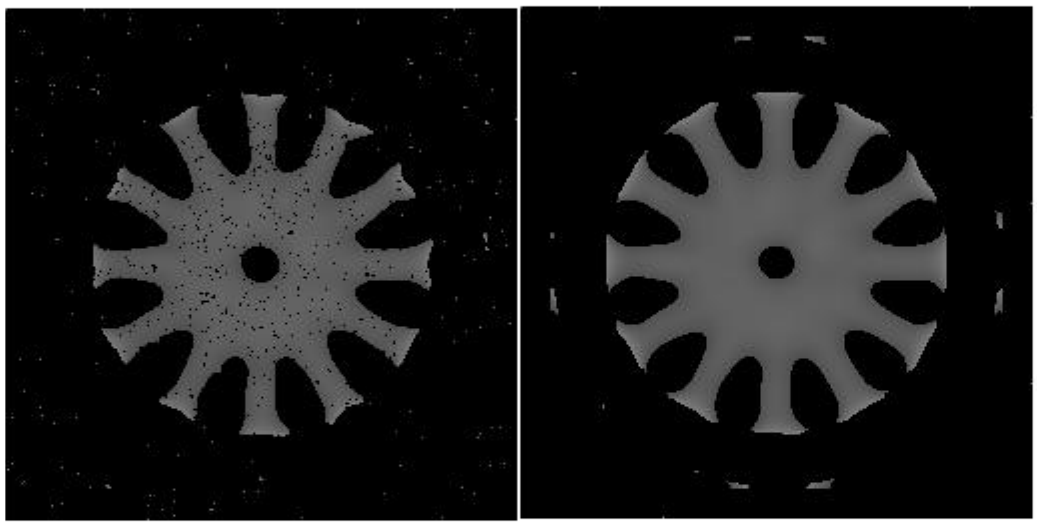


Figure 2.14 "spokes" components

From the experimental results, we can see that the noise in images are decreased with the iterate time increasing. The histograms of results show that the values of pixels in edges are preserved. Besides, we can also conclude that g using exponential function is high-contrast edges favored and g using inverse function is wide regions favored.

The result of question b can be shown from figure 2.15 to 2.16.

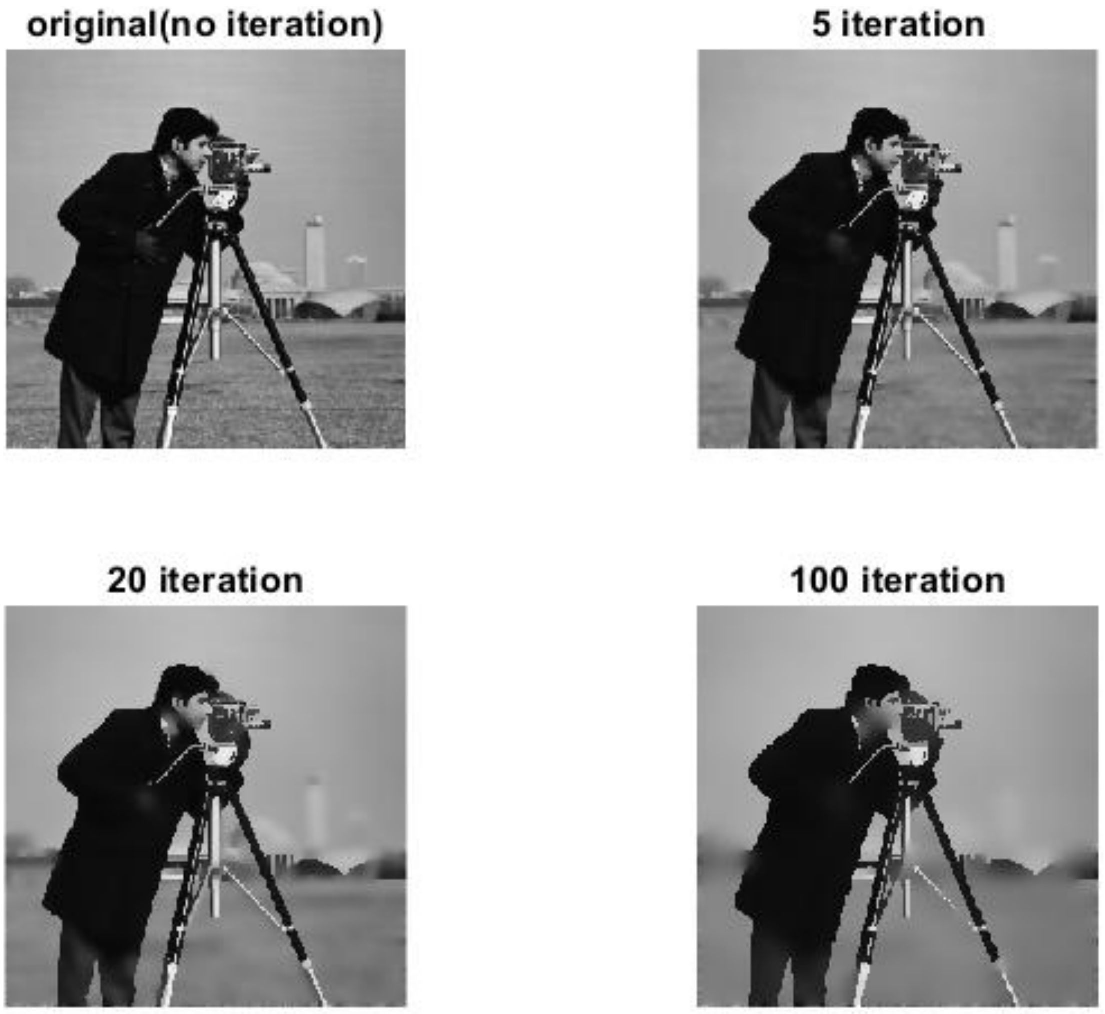


Figure 2.15 result of question b with exponential g

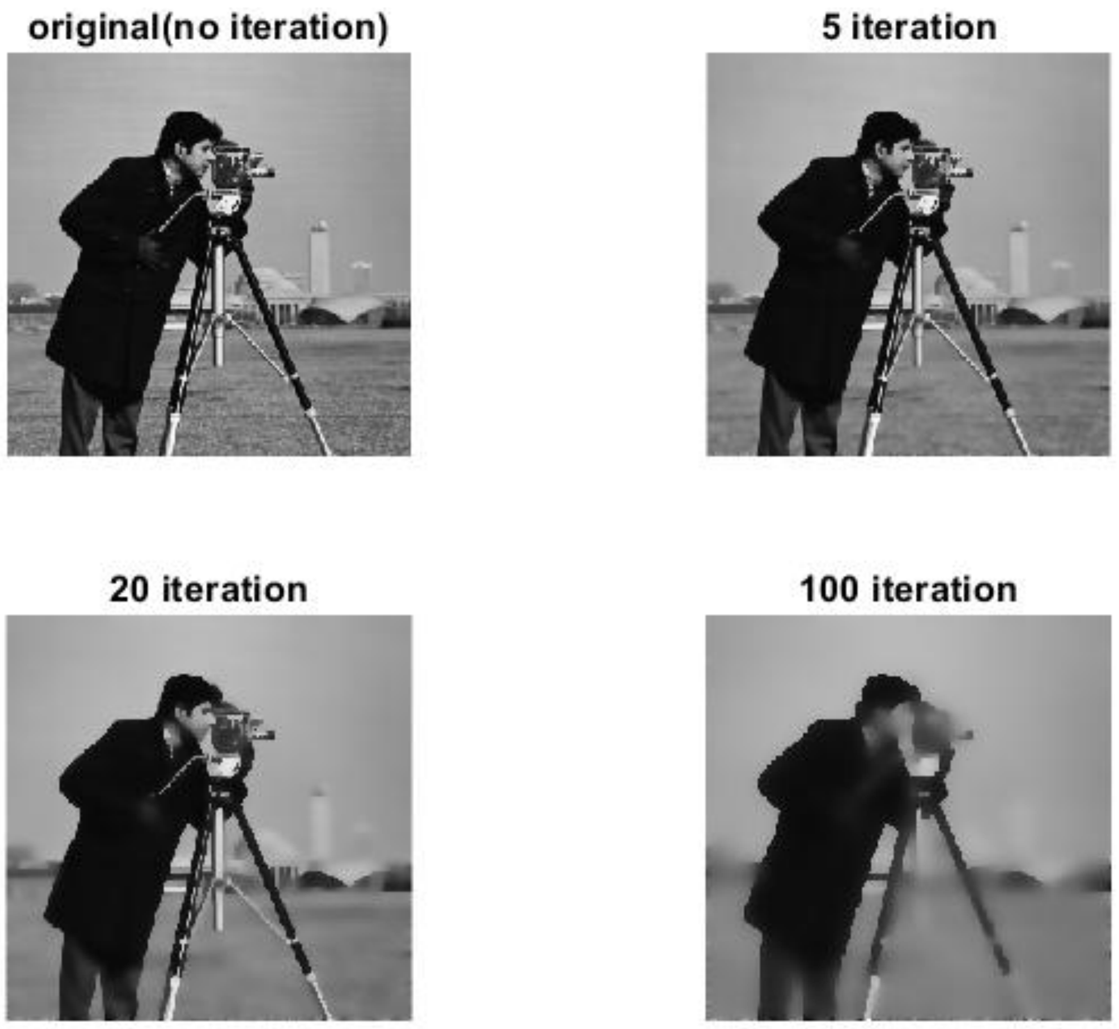


Figure 2.15 result of question b with inverse g

The k values in the above questions are 30 for exponential g and 15 for inverse g. Different k will lead to different smoothing effect. If we increase the value of k in question a (k=40 for exponential g and 25 for inverse g). The new results are shown in figure 2.16 and 2.17.

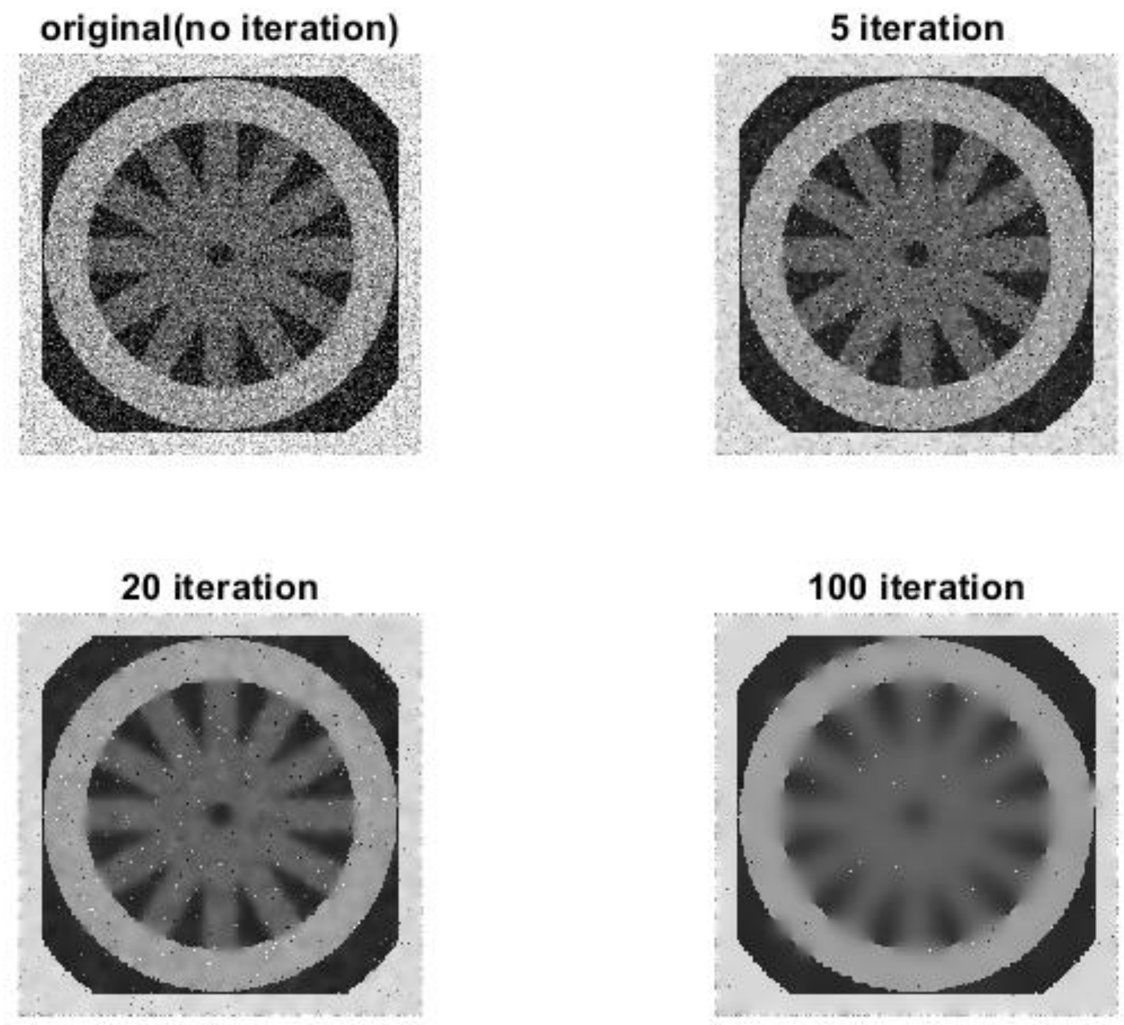


Figure 2.16 result of k = 40 for exponential g

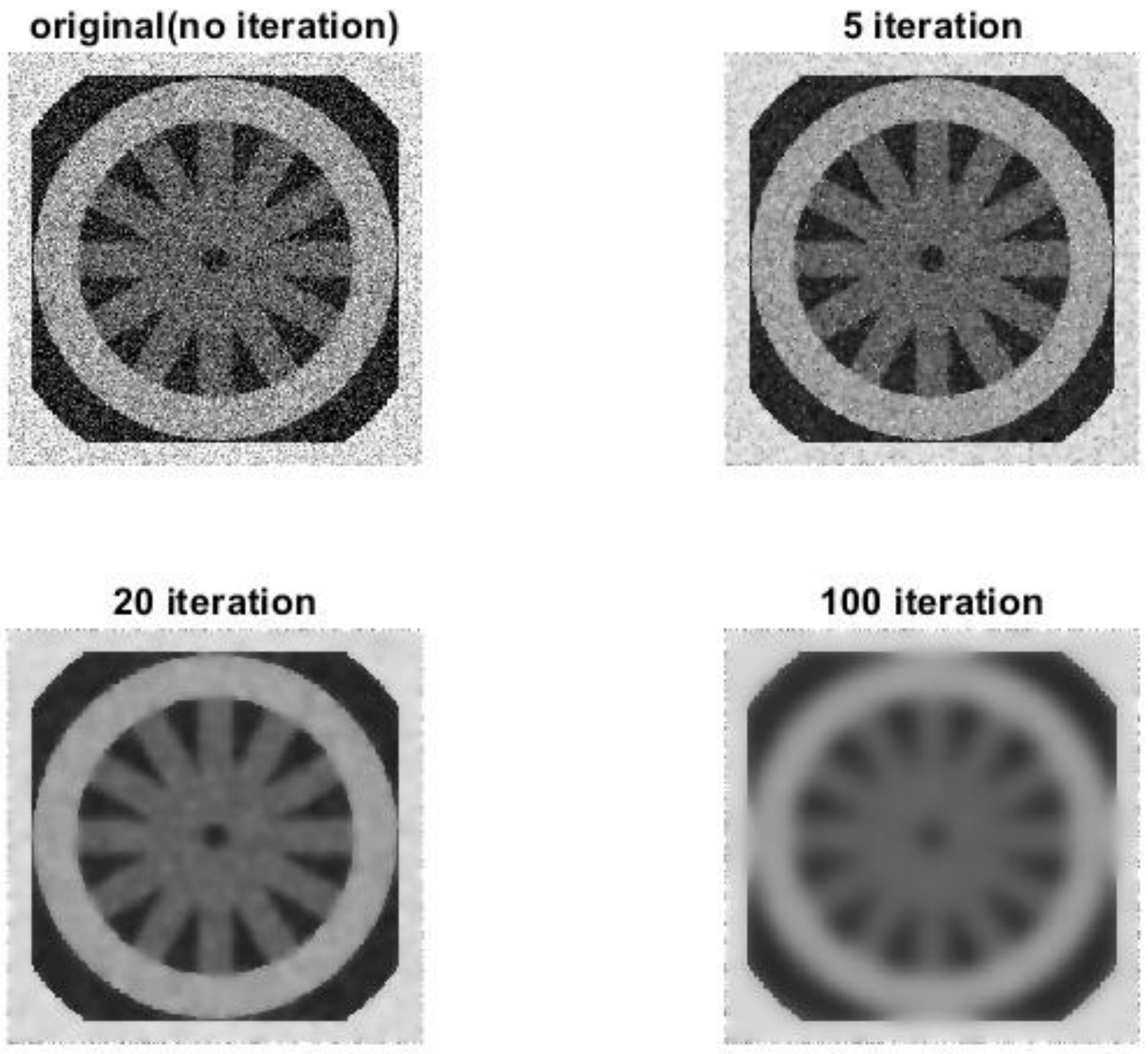


Figure 2.17 result of k = 25 for inverse g

We can see that the images are smoothed more with k increasing. We will loose more details but make edges more distinguished. It can be also proved in “cameraman” in figure 2.18 and 2.19.

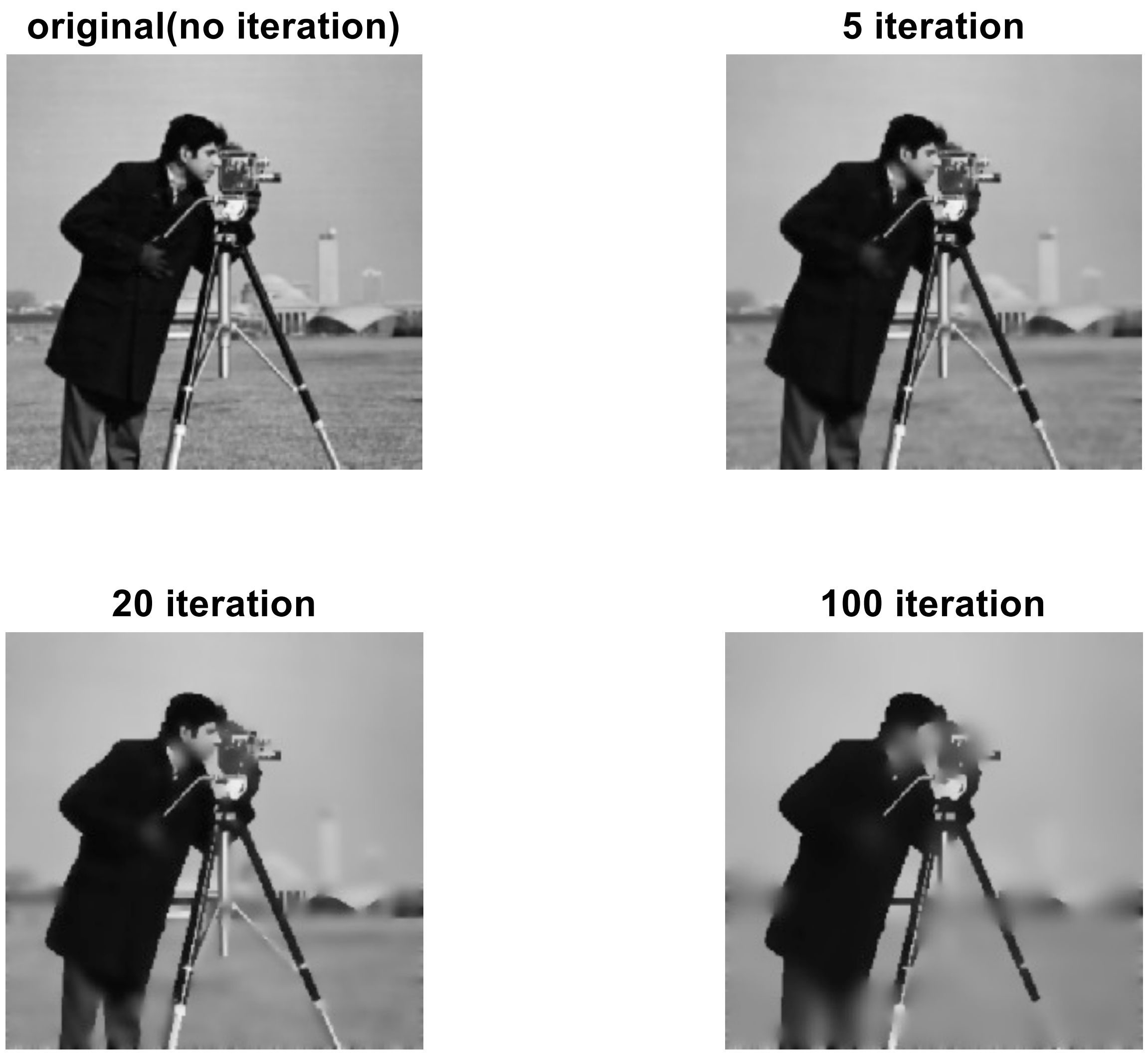


Figure 2.18 result of k = 40 for exponential g

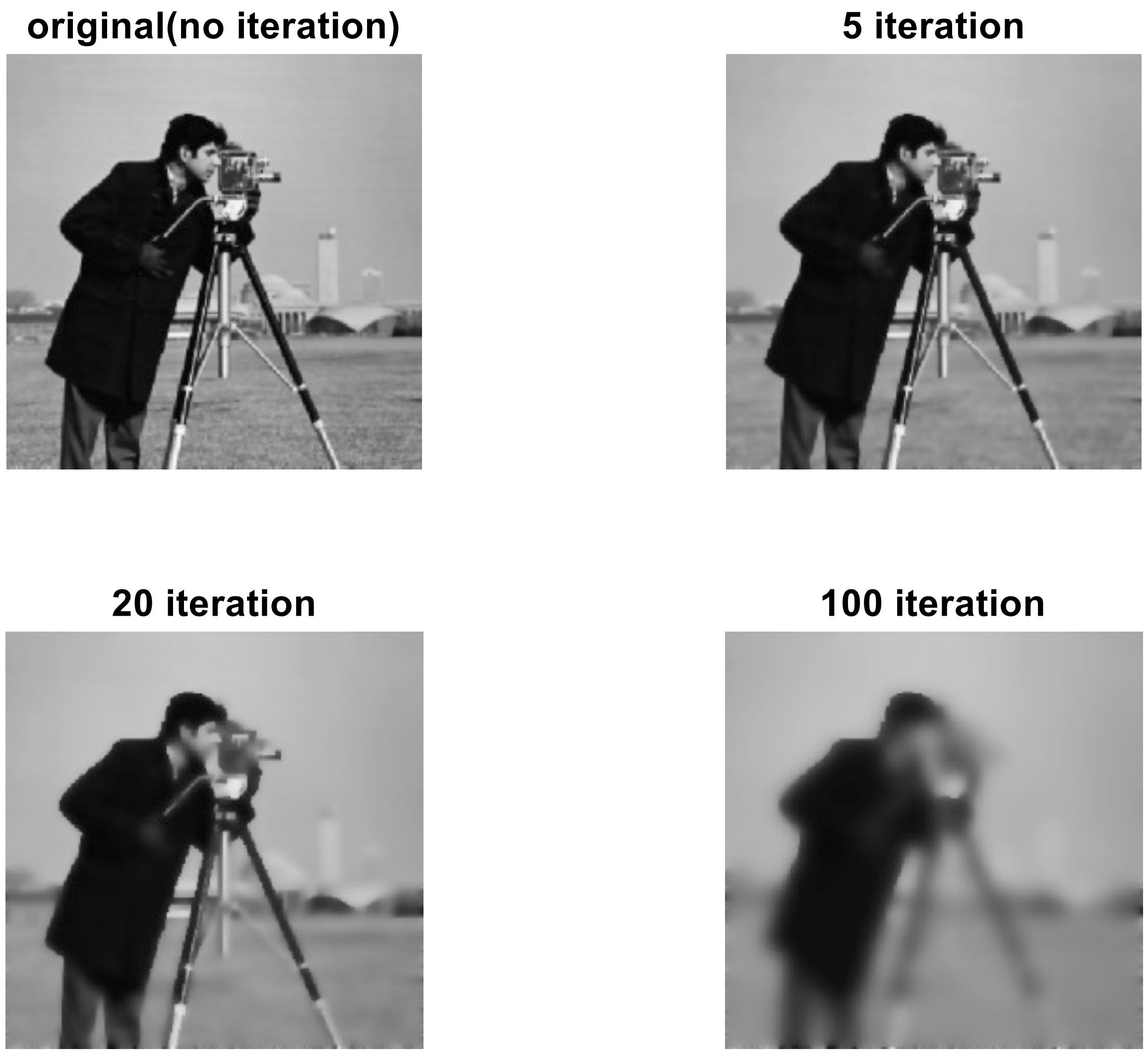


Figure 2.19 result of k = 25 for inverse g

Secondly, if we decrease k to 20 and 5. The results is shown from figure 2.20 to 2.23.

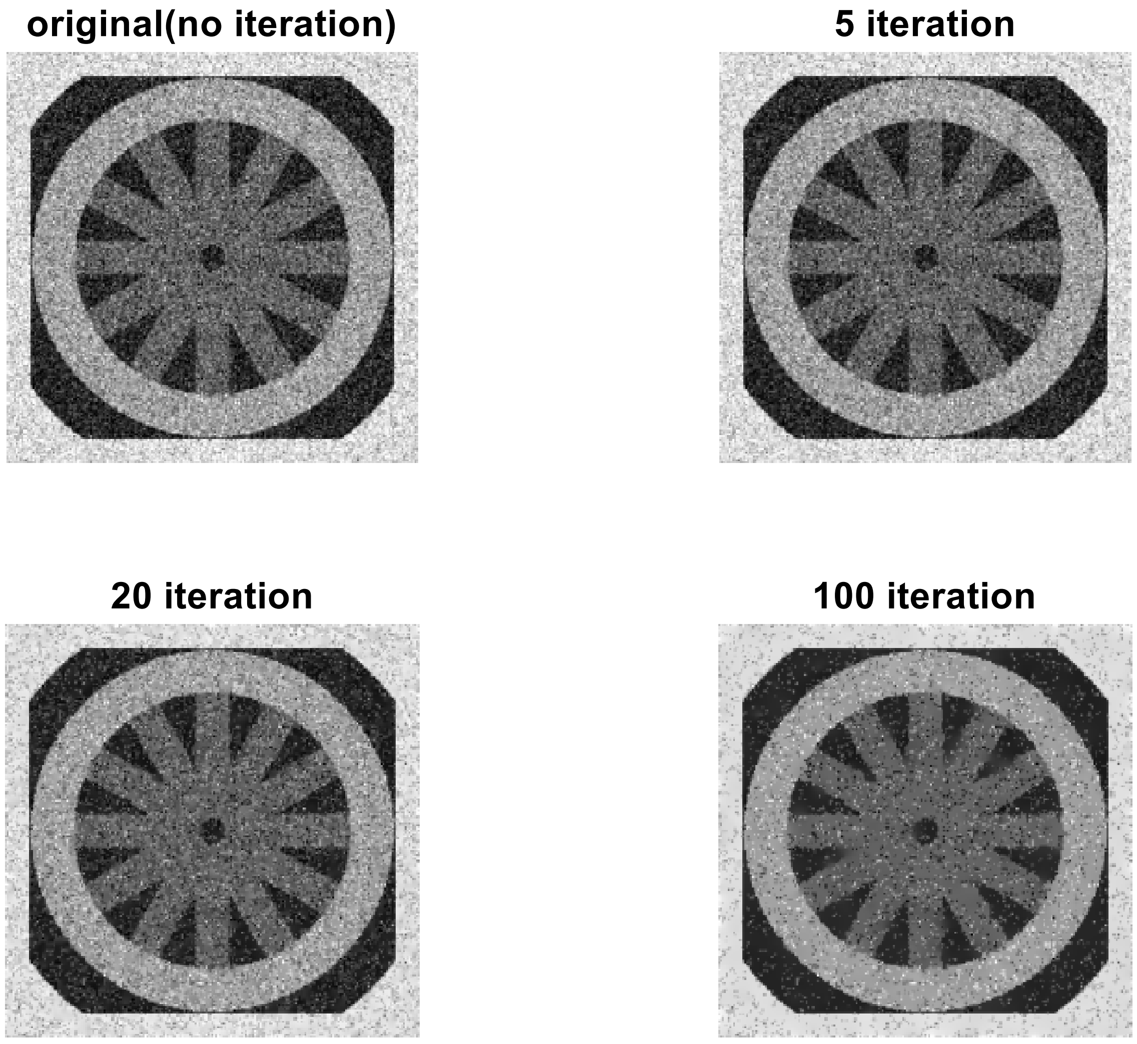


Figure 2.20 result of k = 20 for exponential g

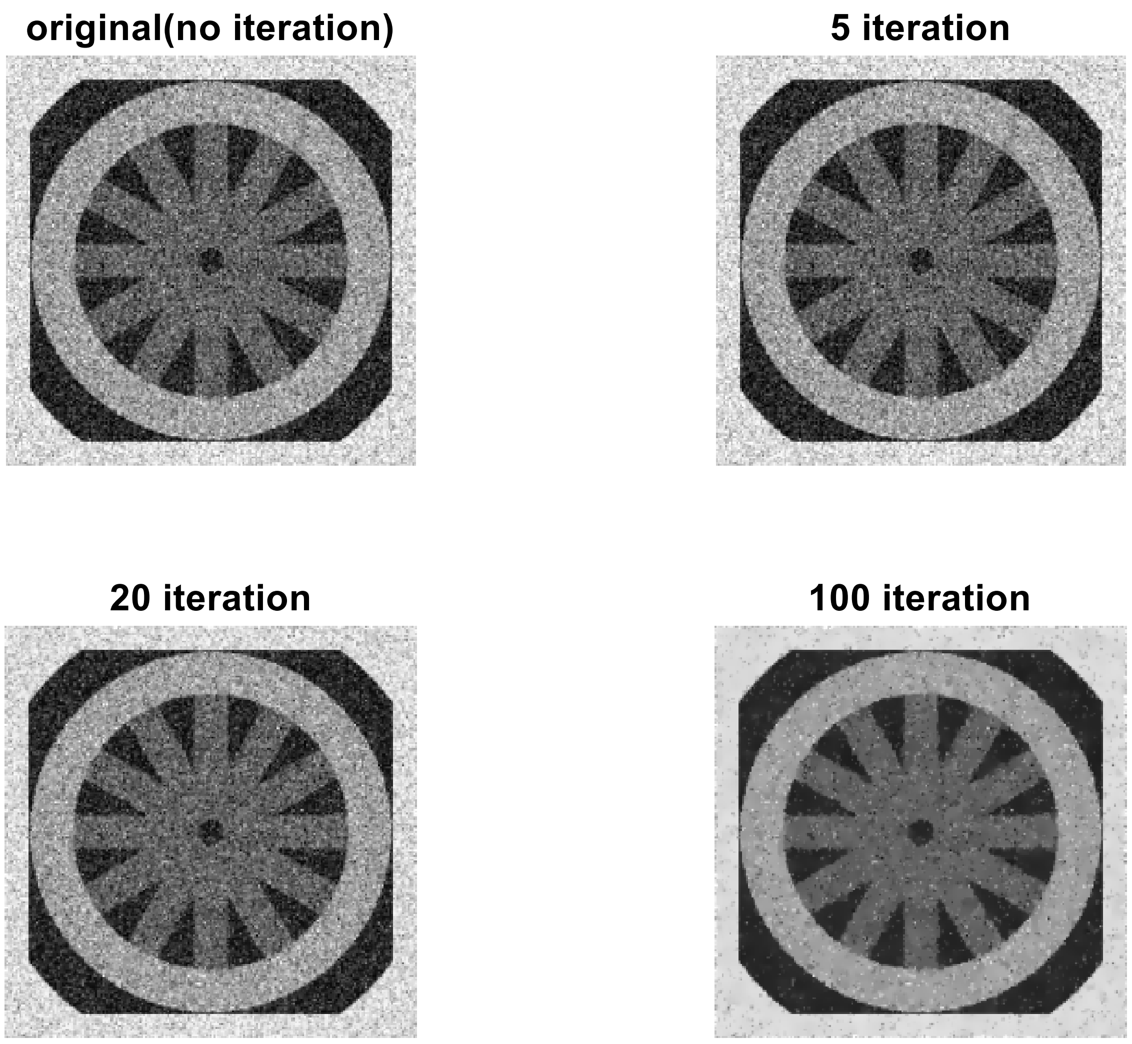


Figure 2.21 result of k = 5 for inverse g

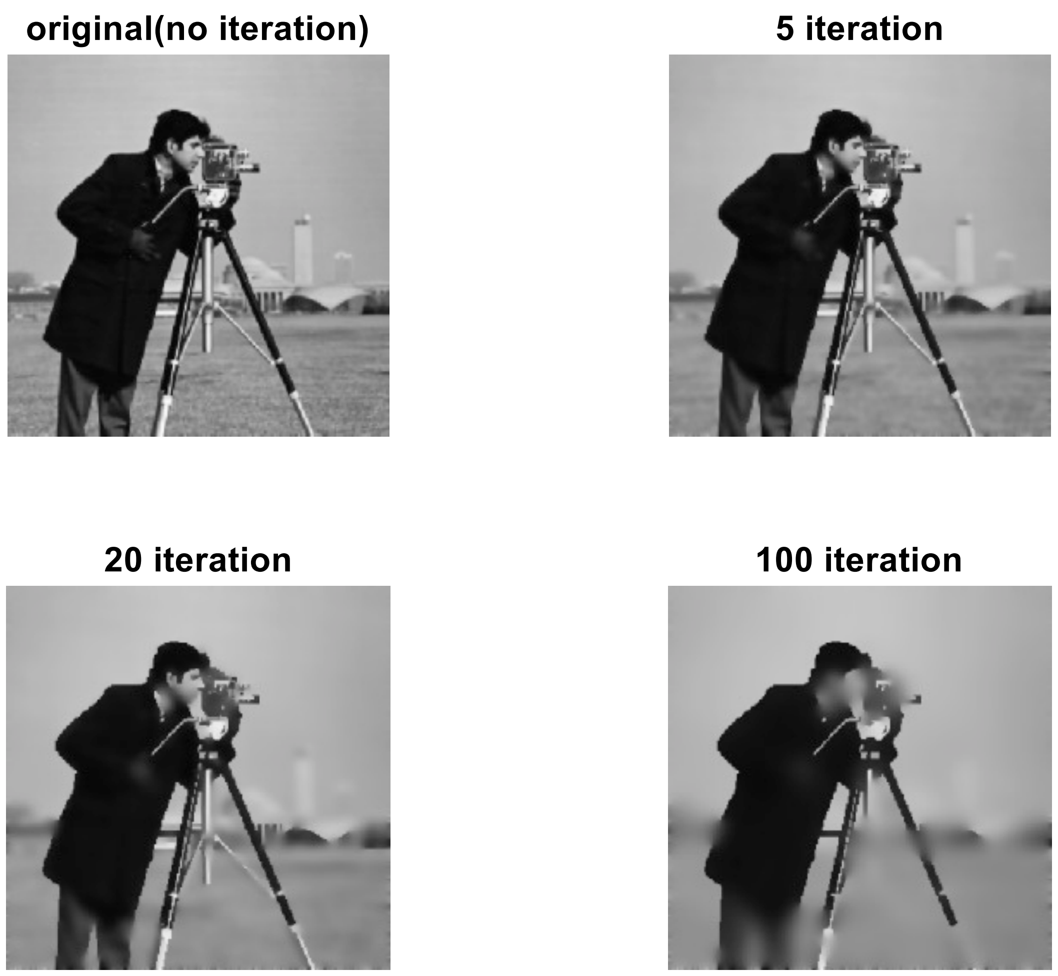


Figure 2.22 result of k = 20 for exponential g

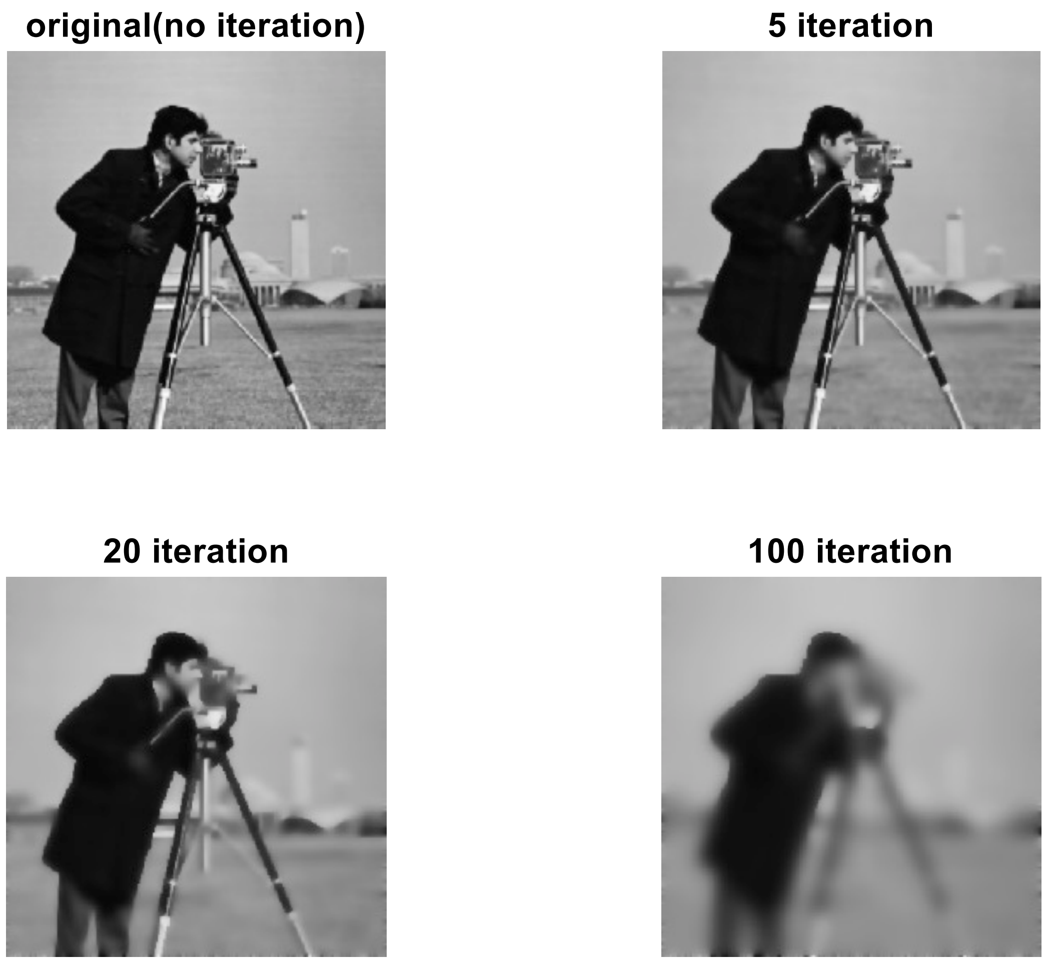


Figure 2.23 result of k = 5 for inverse g

It can be concluded that the images will be smoothed less if we decrease values of k. The more details will be preserved but noises will also be preserved. From figure 2.14("spokes" segmented with different g), it can be observed that g using exponential function can get a result with more contrast and g using inverse function can get a result with less noise. The image "cwheelnoise" has more edges than the "cameraman". Therefore, when we apply anisotropic diffusion to these two images, it will get a better result in "cameraman" with the same iteration time. The image "cwheelnoise" has more gaussian noises which can be removed better in algorithm.

D. Conclusion

For question 1, all tested filters have reduced noise on the surface, but the final output results have changed significantly. A useful way to lose the filtering concept. Considering that the image smoothing process is usually successful, this smoothing process results in the loss of various types of local information (such as the reduction of region boundaries) at the cost. Filters can be useful if we can understand what type of information is lost. We should design filter in response to specific noise.

For question 2, anisotropic diffusion considers images as heat conduction and force it flows depending on the relationship between current pixel and surrounding pixels. For example, if a neighborhood pixel is significantly different from the current pixel, it means that this neighborhood pixel is likely to be a boundary, so the current pixel will not diffuse in this direction, and this boundary will be retained. There are two different method to construct conduction coefficients: exponential function and inverse function. The first g is high-contrast edges favored and the last one is wide regions favored. And the values of k can make influence on smoothing effect. Big k will smooth more and preserve less details and noises. Small k will smooth less and preserve more details and noises.