CSE585/EE555: Digital Image Processing II

Computer Project # 3:

Nonlinear Filtering and Anisotropic Diffusion

Yanxi Yang, Jiuchao Yin, Hongjie Liu

Date: 03/26/2020

A. Objectives

- Study the algorithms and design different filters with given neighborhood and implement expecting applications: mean, median, alpha-trim, sigma, symmetric nearest-neighbor mean, and anisotropic diffusion.
- Get familiar with gray-scale image histograms and extract useful information from the histograms.
- Learn how to segment images from applying filters and read histograms by manually setting the threshold.
- Give observation of single iteration vs repeated application of filters on real-world images.
- Evaluate the performance of different algorithms with different parameters of anisotropic diffusion.
- Observe and examine the behaviors of anisotropic diffusion filter with different conduction coefficient functions.

B. Algorithms

1. Non-linear filtering algorithms

i. 5x5 mean filter

A 5x5 mean filter is to simply replace each pixel value in an image with the mean (average) of its neighboring pixels (window size is 5x5). Since the output pixel is linear combinations of the neighboring input pixels, thus, 5x5 mean filter is a linear filter.

$$y_k = \text{mean}(x_{k-N}, x_{k-N+1}, x_{k-N+2}, \ldots\ldots x_{k+N})$$

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

where, middle element is the kth element, total number of pixels are 2N+1.

ii. 5x5 median filter

A 5x5 median filter is a non-linear filter. At each position of window, the sample values inside are ranked according to their magnitude and the middle element of this ranking is defined to be the output.

$$y_k = \text{median}\,(x_{k-N,}x_{k-N+1,}x_{k-N+2,}\ldots\ldots x_{k+N})$$

where, middle element is the kth element, total number of pixels are 2N+1. In our case, k is 13 and N is 12.

iii. alpha-trimmed mean filter

Alpha-trimmed mean filter is a subclass of order-statistical filter. The filter averages a subset of samples in the filter window: the points excluded are those of very high or very low rank.

$$y_k = \frac{1}{n-2\lfloor \alpha n \rfloor}\sum_{i=\lfloor \alpha n \rfloor+1}^{n-\lfloor \alpha n \rfloor} x_{(i)}$$

where n= 2N+1=25, is the window length, α =0.25.

iv. Sigma filter

The 2-D 5 x 5 sigma filter takes 25 pixels as input. If we consider $x_{i,j}$ as the pixel of interest, the input pixels for the 5 x 5 sigma filter are

$$\begin{bmatrix} x_{i-2,j-2} & \cdots & x_{i-2,j+2} \\ \vdots & x_{i,j} & \vdots \\ x_{i+2,j-2} & \cdots & x_{i+2,j+2} \end{bmatrix}.$$

After sigma filter, the output

$$y_{i,j} = \frac{1}{N_c}\sum_{k=-2,l=-2}^{k=2,l=2} \delta_{k,l} * x_{i+k,j+l}.$$

Here,

$$\delta_{k,l} = \begin{cases} 1, & |x_{i+k,j+l} - x_{i,j}| \le 2\sigma\ (\sigma = 20) \\ 0, & otherwise \end{cases},$$

and

$$N_c \triangleq \#\ of\ points\ x_{i+k,j+l}\ having\ \delta_{k,l} = 1.$$

Overall, sigma filter averages values belonging to the same distribution as the interest input $x_{i,j}$, while excluding noisy outliers in the input.

v. Symmetric NNMF:

The symmetric nearest neighbor mean filter is a 2-D filter. In this project, the size of the window is 5*5. The function of this filter is to select the point which is the most similar to the central pixel in the window from each symmetrically opposite pair. So, there are 13 pixels after the selection, and then get the average value of these 13 pixels to represent the value of the central pixel in the new image. Here is an example of symmetric nearest neighbor mean filter.

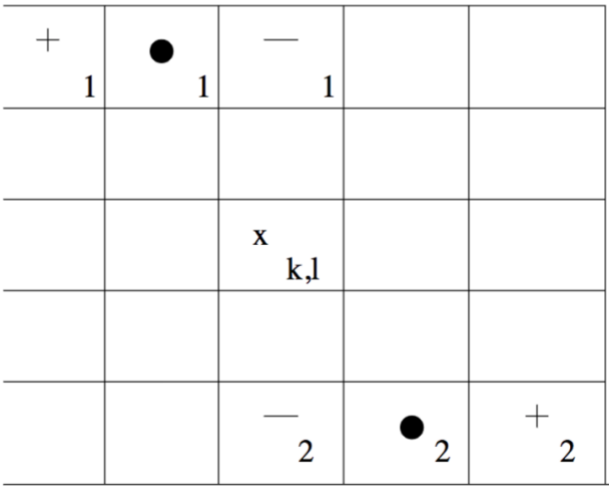


Figure 1 An example of symmetric nearest neighbor mean filter.

The formula of the output with respect to Figure 1 is

$$y_{k,l} = mean\{[+_1, +_2\ ], [\cdot_1, \cdot_2], [-_1, -_2], \dots, x_{k,l}\}$$

The symmetric nearest neighbor mean filter can reduce the noise, sharpen edges, and reduce thin lines.

vi. Mean and standard deviation of the interior of the large disk region:

The matrix containing the large disk is isolated manually by taking the row 50 to row 181 and column 32 to column 150 out from the original image. The left panel of Figure 2 demonstrates the area cropped by this method from the image after mean filter for 5 iterations. The right panel shows it histogram. We can see the background peaks at around 20, while the large disk peaks at around 200. I also carefully checked the brightness of the large disk as well as the line inside the disk. They are all above 100 after mean filter for five iterations, since all the dark noises are gone after mean filter application. Therefore, our method to extract the large disk is to extract the coordinates of pixels whose brightness is higher than 100 based on the image after mean filer. Then the mean and standard deviation of these selected pixels are calculated. For consistency, we use the same coordinates which we extract from the result after mean filter for the results of all the five filters.

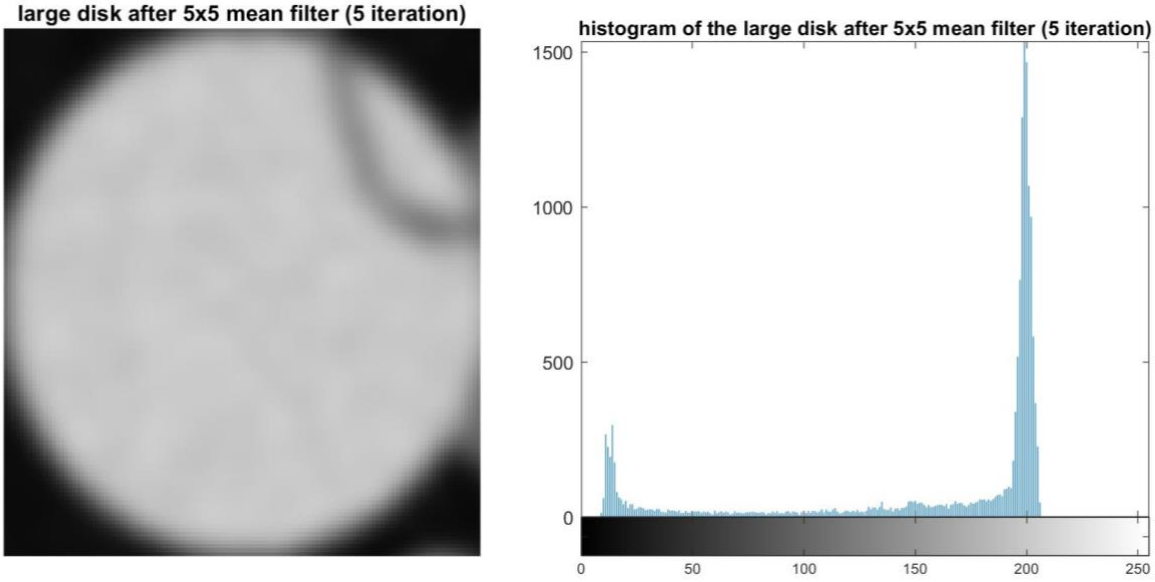


Figure 2 The matrix containing the large disk and its histogram after applying 5 x 5 mean filter to the original disk image.

2. Anisotropic Diffusion for Image Filtering:

The implementation of anisotropic diffusion for image filtering is based on the equation

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

In this equation, N, S, E, W represent north, south, east, west neighbors of pixel (i, j). λ is a parameter and is set to be 0.25 in our project. $\nabla_* I$ refers to difference between pixel (i, j) and one neighbor of it, for example:

$$\nabla_N I_{i,j}^t = I_{i,j+1}^t - I_{i,j}^t$$
$$\nabla_S I_{i,j}^t = I_{i,j-1}^t - I_{i,j}^t$$
...

$\{c_N^t, c_S^t, c_E^t, c_W^t\}$ are conduction coefficients and updated after every iteration. In this project, we have tried two computation methods of it. How to compute them is shown below:

$$c_*^t(i,j) = g(|\nabla_* I_{i,j}^t|)$$

In this project, we have tried two forms of the function g. The first one is the exponential formula:

$$g(\| \nabla \mathrm{I} \|) = \exp\{-(\| \nabla \mathrm{I} \| / k)^2\}$$

The second one is the inverse quadratic function:

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

The k in both functions is a parameter and the value of it is selected by us. We have tried several values of k, and the different results are shown in results section.

For cwheelnoise.gif figure, the images after 0, 20, 50, 100 iterations, gray-scale histogram, plot of the line y=128 through the image, and segmented version are also shown. We select pixels which values are between 80 and 110 to be the segmented image which shows the shape of the

wheel, and the range of the value is according to the gray-scale histograms, we just pick the second peak in the histogram.
The same process is also conducted on cameraman.tif image.

C. Structure of codes (Readme)

main_1.m
The main function.

5x5mean.m
Function 5x5mean.m takes an image as input and applies a 5x5 neighborhood mean filter to it. The output image will be the image after 5x5mean filter. This function does not call any other function.

alpha_trim.m
Function alpha_trim.m takes an image and parameter alpha as input and applies an 5x5 alpha-trimmed mean filter to it. The output image will be the image after the alpha-trimmed filter. This function does not call any other function.

sigma_filter.m
Function sigma_filter.m takes an image as input and applies sigma filter to it. The output image will be the image after sigma_filter. This function does not call any other function.

symmetry_NNMF.m
Function symmetry_NNMF.m takes an image as input, and applies symmetric nearest neighbor mean filter to it. The output image will be the image after symmetric nearest neighbor mean filter. This function does not call any other function.

Anisotropic_Diffusion_exp.m
Function Anisotropic_Diffusion_exp.m takes an image, parameter t, and parameter k as the inputs. Parameter t refers to the number of iterations, and k refers to the parameter in $g(\cdot)$. The function $g(\cdot)$ in this function is the exponential form. The outputs are the images after the 0_{th}, 20_{th}, 50_{th}, and 100_{th} iteration. This function does not call any other function.

Anisotropic_Diffusion_rev.m
Function Anisotropic_Diffusion_rev.m is the same as function Anisotropic_Diffusion_exp.m except $g(\cdot)$. The $g(\cdot)$ in this function is the inverse quadratic form.

Flow chart:

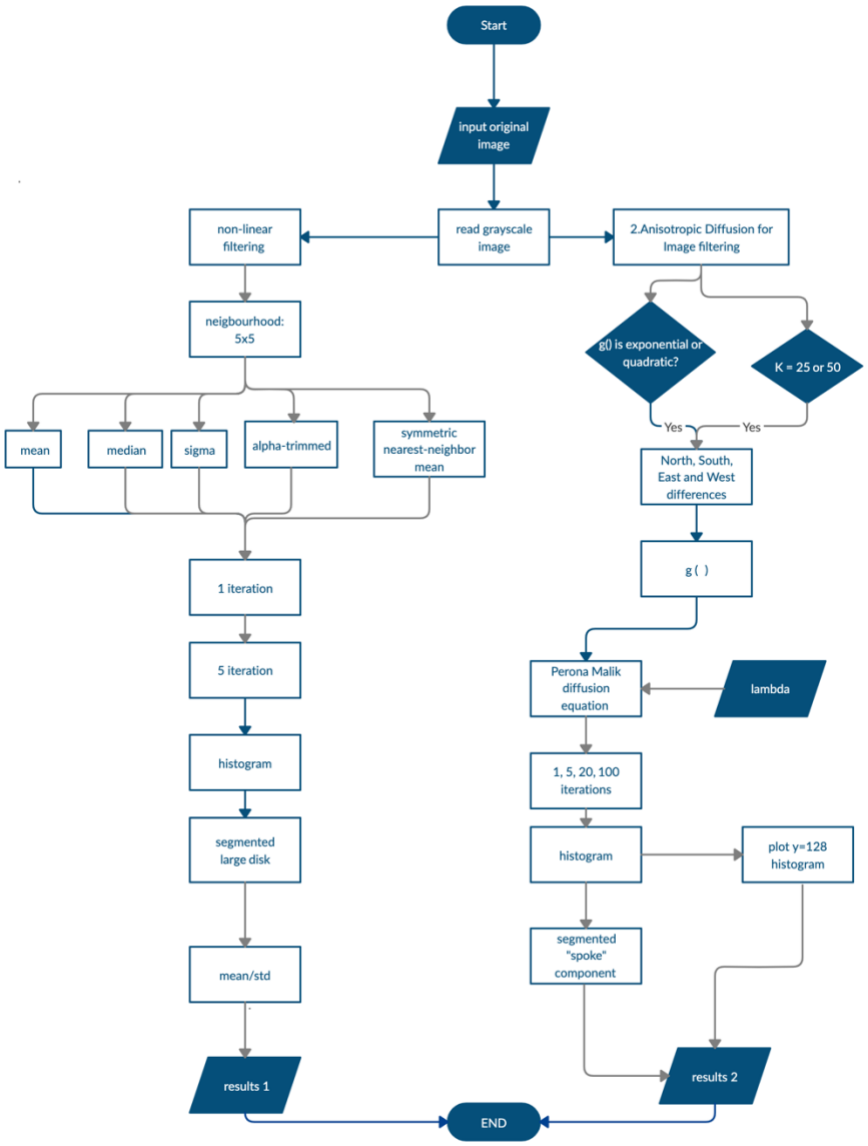


Figure 3 flowchart of project 3. Left: procedure of nonlinear filtering, right: procedure of anisotropic diffusion.

D. Results

1. Results of Nonlinear Filtering

The first part of this project focuses on applying different types of nonlinear filters and understanding their effects. 5 types of nonlinear filters have been tried to apply on the disk.gif image for up to 5 iterations. The 5 filters are: 5×5 mean filter, 5×5 median filter, 5×5 alpha-trimmed mean (alpha= 0.25), 5×5 sigma filter (σ = 20), and 5×5 symmetric nearest-neighbor mean filter. The image output of the first iteration and fifth iteration. After 5 iteration, the gray-scale histograms were shown for each filter. Meanwhile, the mean and standard deviation of the interior of the large disk were also calculated.

Figure 4A is the original image with the gray-scale histogram. We can see the distribution is very disperse due to the presence of noise. We first applied mean filter to it. Figure 4 B showed the result of mean filter after one iteration and 5 iterations respectively. We could see mean filter sharpened the histogram greatly comparing with the original image, but the edge tended to become blurred. Mean filter takes the mean of all the 25 inputs, so it is easy to be affected by outliers. As a result, it will be easy to be affected by the background at the edge. According to the method described in 2.1.6, we extracted the interior of the large disk, and calculated the mean and standard deviation, which are 188.70 and 22.60 (Table 1).

In contrast, median filter preserved the edge better than mean filter (Figure 4C left two panels). Mean filter also greatly sharpened the gray-scale histogram (Figure 4C right panel). Its mean (191.80) of the large disk is slightly higher than the result from mean filter, and its standard deviation (21.01) is slightly lower, indicating median filter is relatively less affected by the dark outlier inside the large disk.

Alpha-trimmed mean filter is also one type of mean filter, the difference is that it will eliminate the outliers. In this project, the 6 brightest and the 6 darkest pixels are not considered when we calculate the output for the interest pixel. We can see from Figure 4D, it has similar gray-scale histogram, but it preserves the edge relatively better than the mean filter. The mean (190.13) and standard deviation (22.16) of the interior of the big disk also confirmed that the alpha-trimmed mean filter is less affected by outliers which are dark points in the large disk.

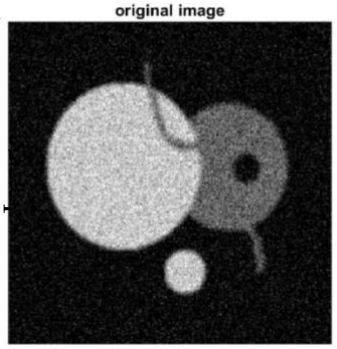
Figure 4E showed the result of sigma filter, which yields sharp edges after 5 iteration. However, there are still some noises inside the disks. Sigma also take mean of the inputs, but it only calculates the mean of pixels of the same statistical distribution with the pixel of interest. Because of this operation, the background will not affect the edge, while some noises would also be kept, they are too different from their surrounding pixels (larger than 40). Similar to the other three filters, sigma filter also sharpened the histogram significantly (Figure 4E). The mean brightness (192.50) of the large disk is the highest among all the five filters, the standard deviation (21.50) is also relatively low (Table 1).

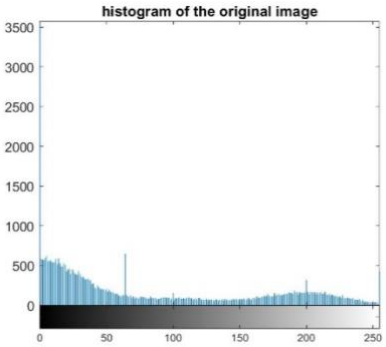
Last but not least, we applied symmetric nearest-neighbor mean filter (SNNMF). Figure 4F demonstrated that SNNMF reduces noises and preserve edges. This method first selects the most similar point to the point of interest from the two symmetrically opposite ones, then averages these selected values. This method considers similar statistical distribution, thus preserving edges. Moreover, for outliers (noises) inside disks, it still selects 13 values from the 25 inputs to average. This avoids the problem of sigma filter which could keep some noises. It has a highest mean (193.63) and standard deviation (23.67) of the large disk (Table 1).

Overall, all the 5 filters sharpened the histogram greatly, meaning all of them could reduce noises. Among them, mean filter is easy to be affected by outliers and has blurring edges. Median filter had good performance in both reducing noises and preserving edges. Alpha-trimmed mean filter, sigma filter, and symmetric nearest-neighbor mean filter all try to modify the mean filter to reduce the effect of outliers. Except that sigma filter might keep some noises, they all improved the performance of mean filter in terms of edge preserving without affecting the performance of noise reduction much.

A

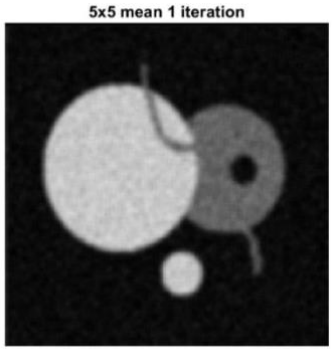
Original

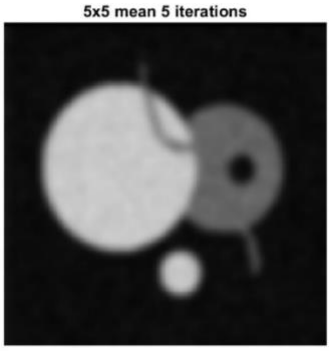


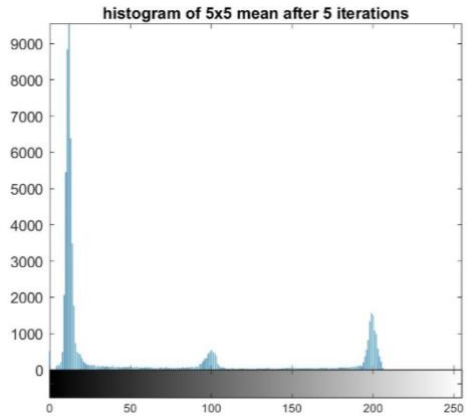


B

Mean Filter

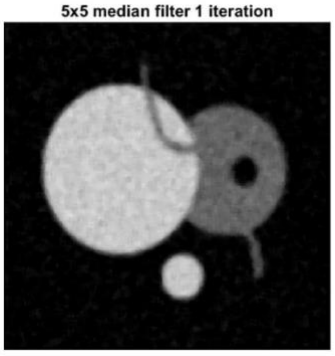


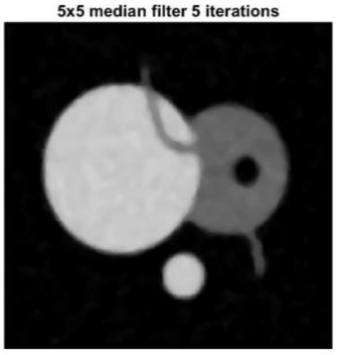


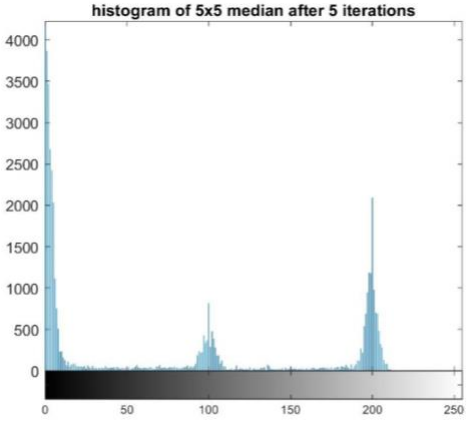


C

Median Filter

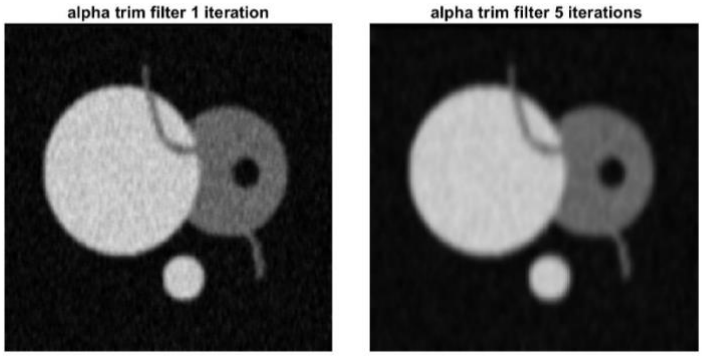


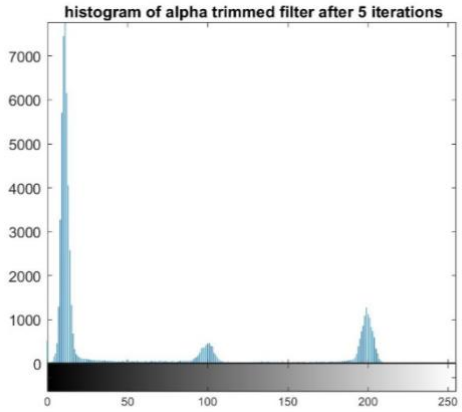




D

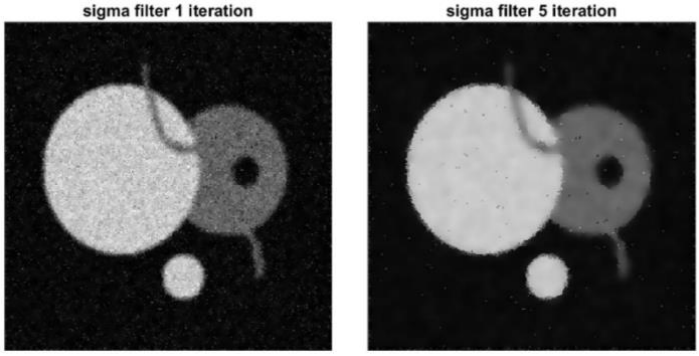
Alpha-Trimmed Mean

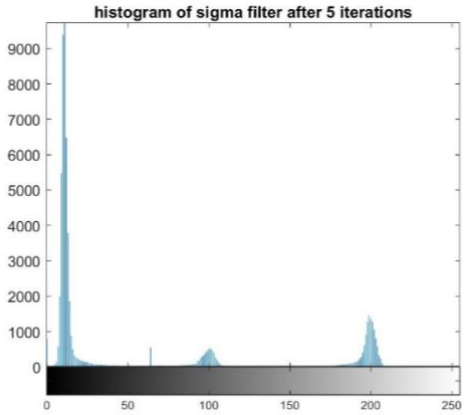




E

Sigma Filter





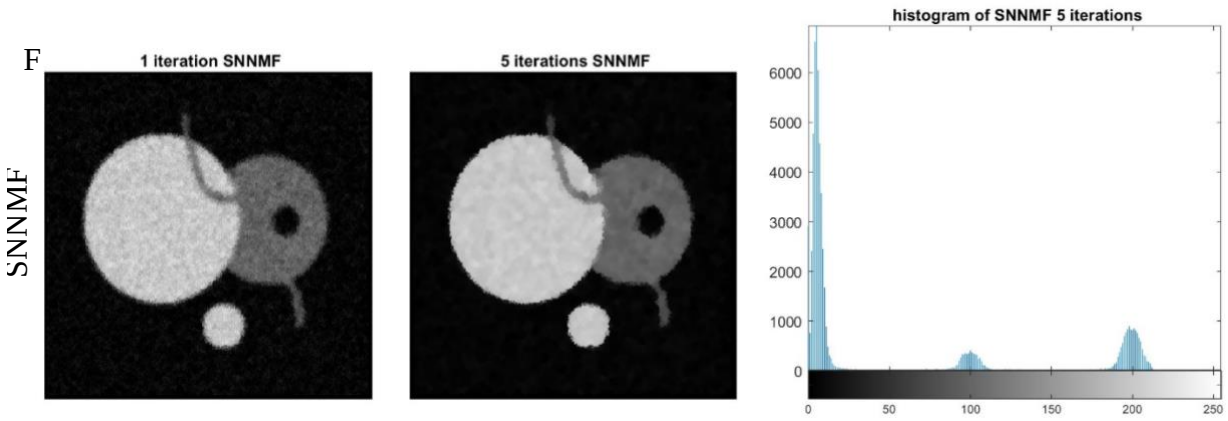


Figure 4 The image output and histogram of the original image (A), and images after applying mean filter (B), median filter (C), alpha-trimmed mean filter (D), sigma filter (E), and symmetric nearest-neighbor mean filter (F) for 1 time and 5 times.

	Mean	Std
5x5 mean filter	188.70	22.60
5x5 median filter	191.80	21.02
5x5 alpha-trimmed mean filter	190.13	22.16
5x5 sigma filter	192.50	21.50
5x5 symmetric nearest-neighbor mean filter	193.63	23.67

Table 1 mean and standard deviation of the interior of the large disk region for each filter after 5 iterations.

2. Results of Anisotropic Diffusion for Image Filtering:

2.1 Anisotropic Diffusion Filter on "cwheelnoise" Image

After the image processed by anisotropic diffusion for image filter, we tried to find the spokes of the wheel and the result images, the gray-scale histogram, the plot of the line y=128 through the image, and the segmented images are given in the result. The results under different parameters are shown in the following figures.

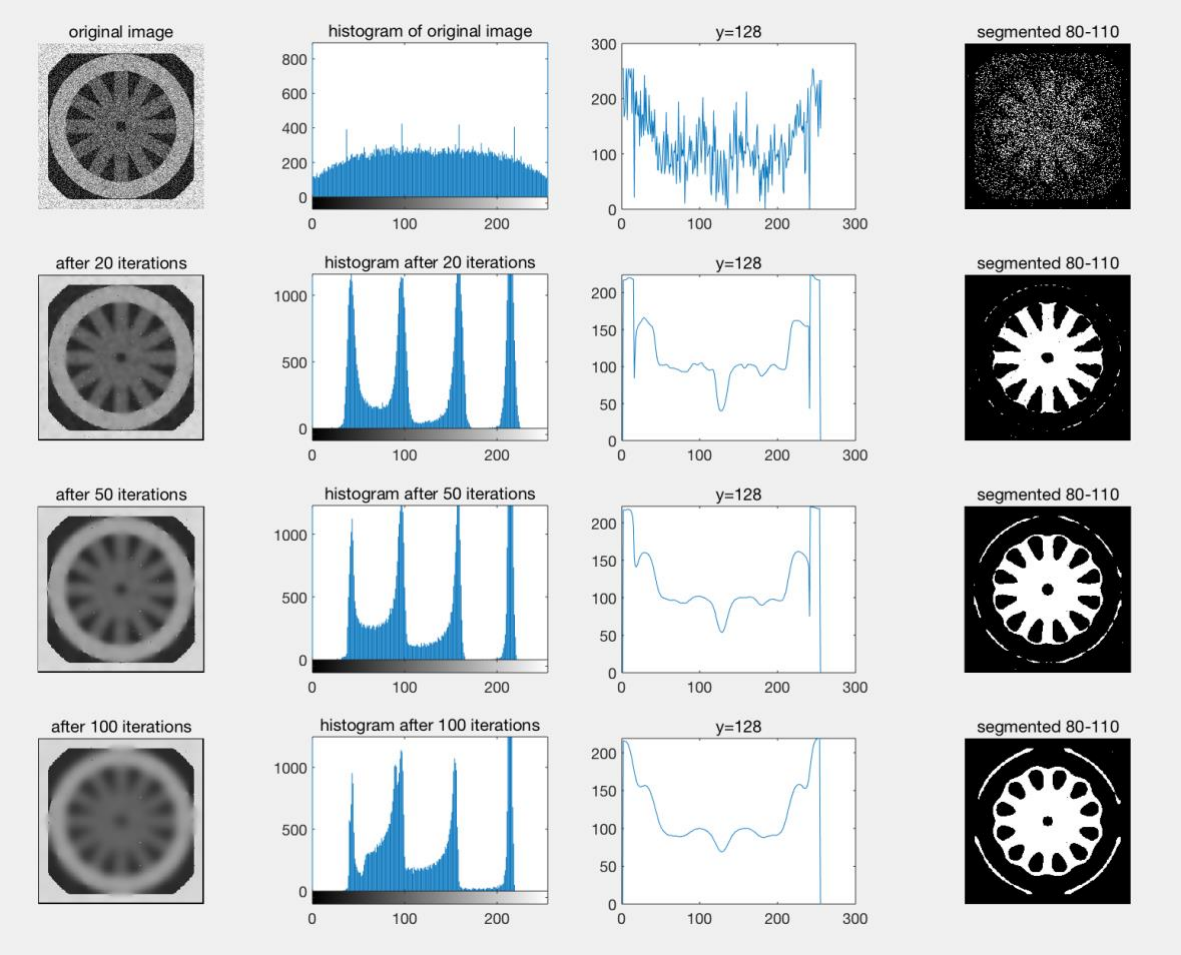


Figure 5 The image outputs, histograms of images, the plot of the line y=128 through the images, and the segmented images after 0th, 20th, 50th, 100th iteration in Anisotropic Diffusion for Image Filter with the parameter $k = 50, g(\cdot) = exp.$

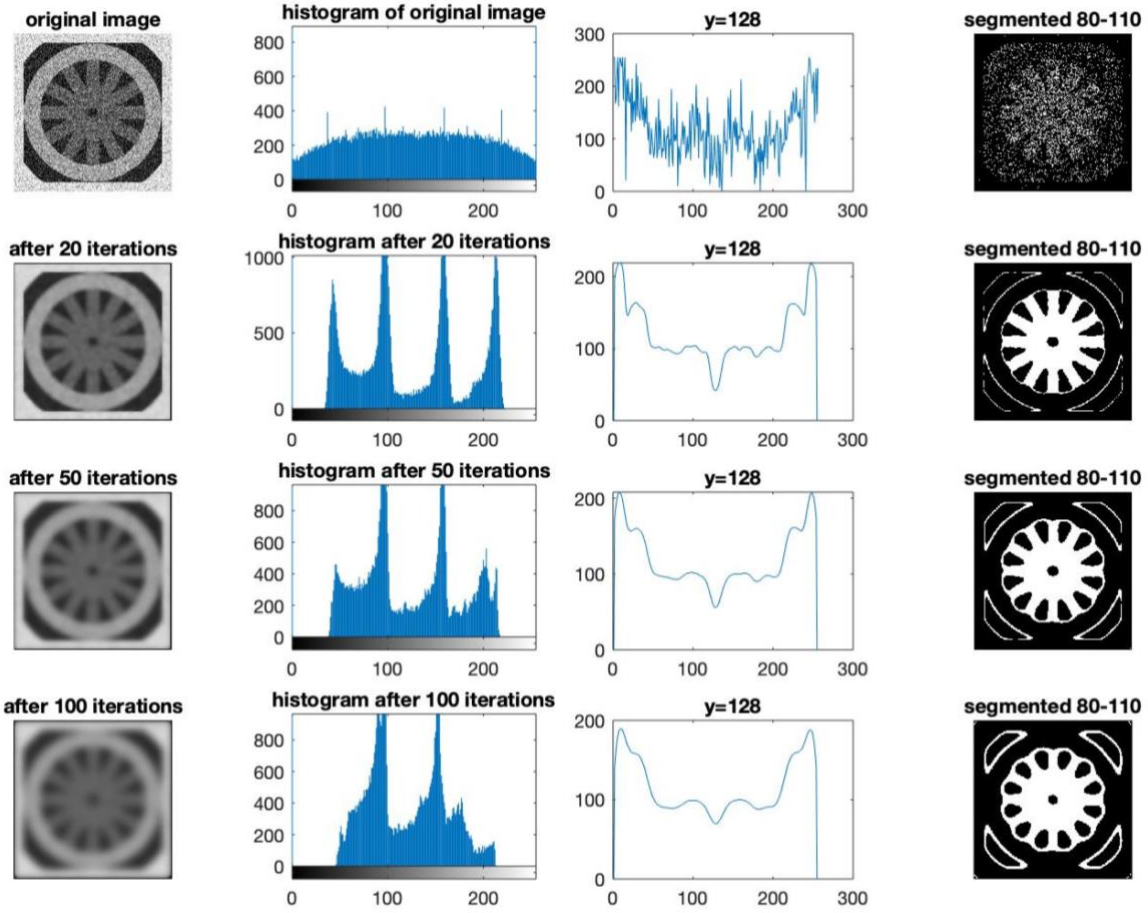


Figure 6 The image outputs, histograms of images, the plot of the line y=128 through the images, and the segmented images after 0th, 20th, 50th, 100th iteration in Anisotropic Diffusion for Image Filter with the parameter $k = 50, g(\cdot) = inverse\ quadratic.$

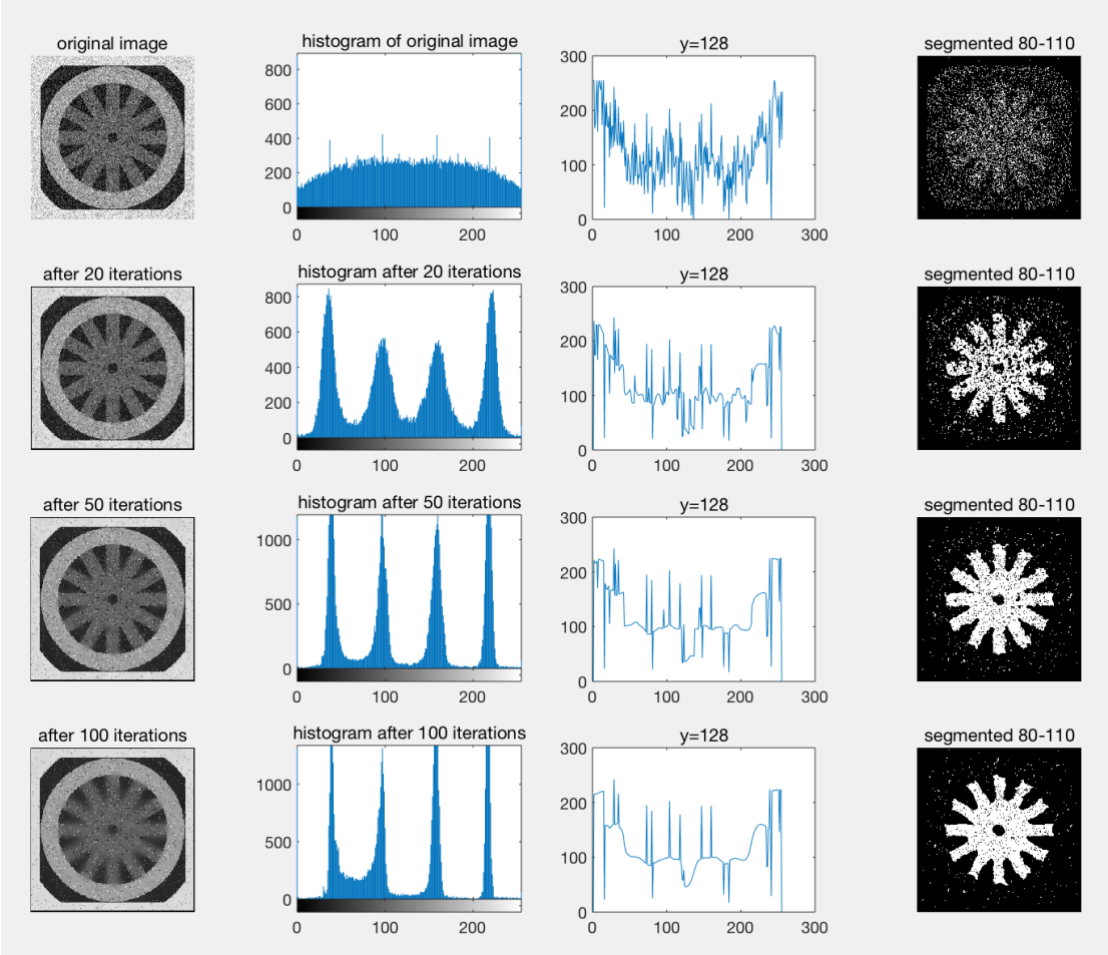


Figure 7 The image outputs, histograms of images, the plot of the line y=128 through the images, and the segmented images after 0th, 20th, 50th, 100th iteration in Anisotropic Diffusion for Image Filter with the parameter $k = 25, g(\cdot) = exp.$

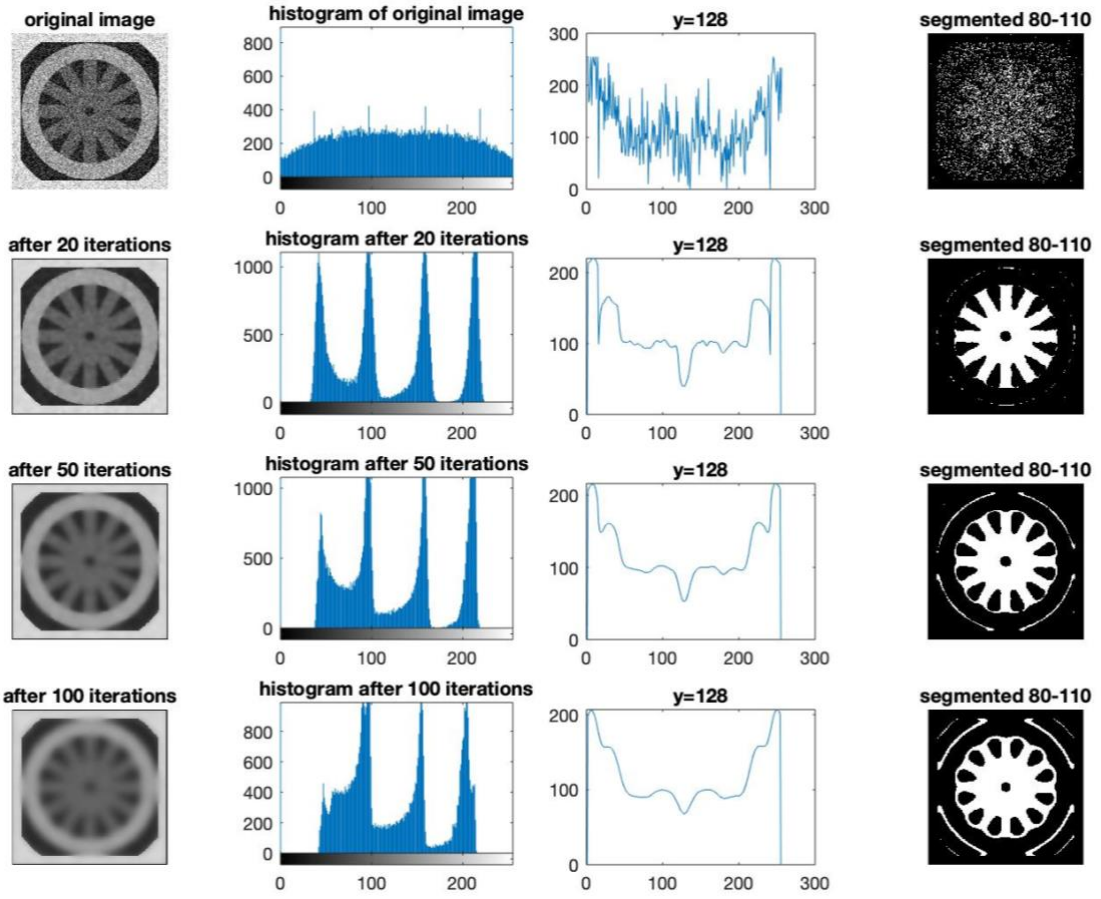


Figure 8 The image outputs, histograms of images, the plot of the line y=128 through the images, and the segmented images after 0th, 20th, 50th, 100th iteration in Anisotropic Diffusion for Image Filter with the parameter $k = 25, g(\cdot) = inverse\ quadratic.$

As we can see from figure 3 to 6, the anisotropic diffusion can remove noises in the original figures after some iterations but with the increment of the iteration, the image becomes more blurred.

From the plot of the line y=128 through the image we can find that the image is smoothed after iterations, the gray level gets similar to the pixels around it, so we can extract the gray-scale component like the spokes in the wheel.
After iterations, there are several peaks in the histogram, and each peak refers to one gray-scale component in the image, we selected the range 80-110, which is the second peak in the histogram, to represent the spokes component of the wheel since the color of the wheel spokes is the second darkest. And the segmented results also verify that our selection is correct.
Use the exponential function as $g(\cdot)$ can preserve high-contrast edges, and the inverse quadratic function can preserve wide regions. As we can see from the segmented images in figure 3 to 6, the exponential function can segment the spokes of the wheel better than the inverse quadratic function. And there are more peaks in the histogram when using exponential function. This observation can also verify the property of these two functions.

We also tried different value of k in this part. In figure 3, with $k = 50$ the image will get blurred after iterations, and a lot of details are lost finally, so the segmented result is not very ideal. But it can remove the noise more quickly. In figure 6, with a lower value of k ($k = 25$), the image is not blurred even after 100 iterations and it can help to extract the spokes of the wheel much better with the exponential function.

2.2 Anisotropic Diffusion Filter on "cameraman" Image

We processed the "cameraman" image with the same filters used above. Figure 9 showed that when k = 25, g(.) is the exponential function, each area tends to be smoothed by the anisotropic diffusion filter, but the edge or the segmentation was preserved very well. However, when g(.) is inverse quadratic function, the whole image was smoothed and the edge became very blurring (Figure 10).

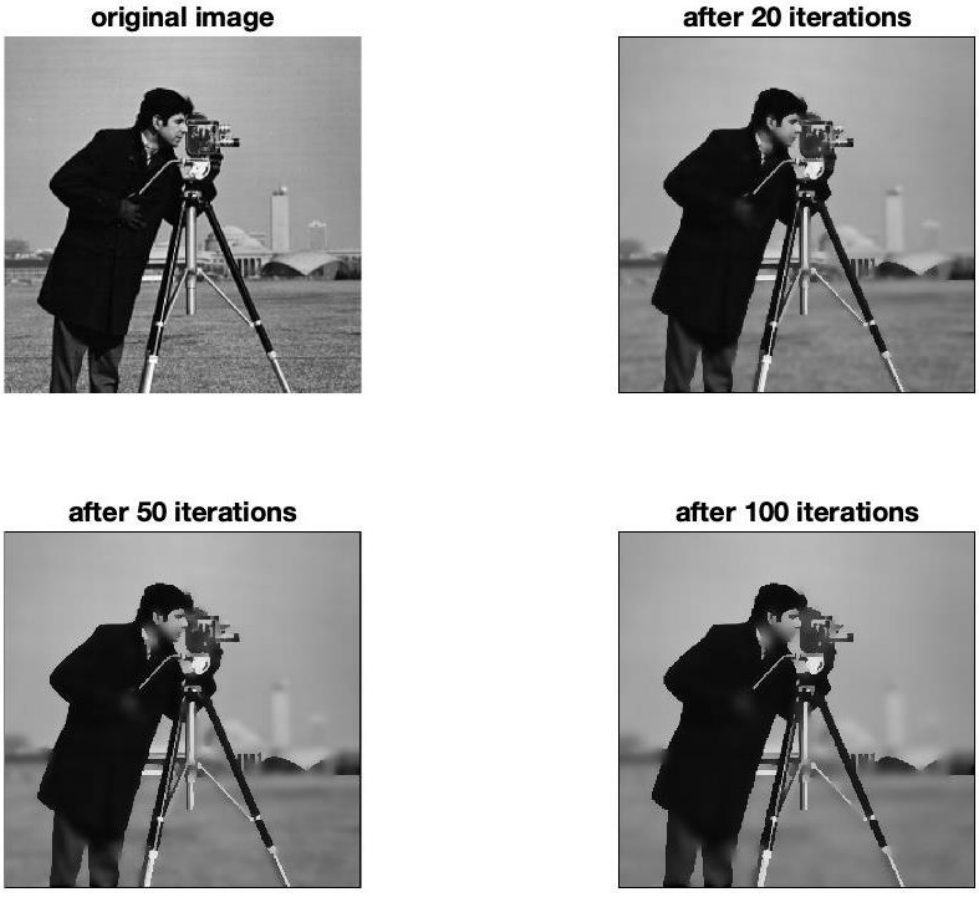


Figure 9 Applying anisotropic diffusion for up to 100 iteration. k = 25, g(.) = exponential.

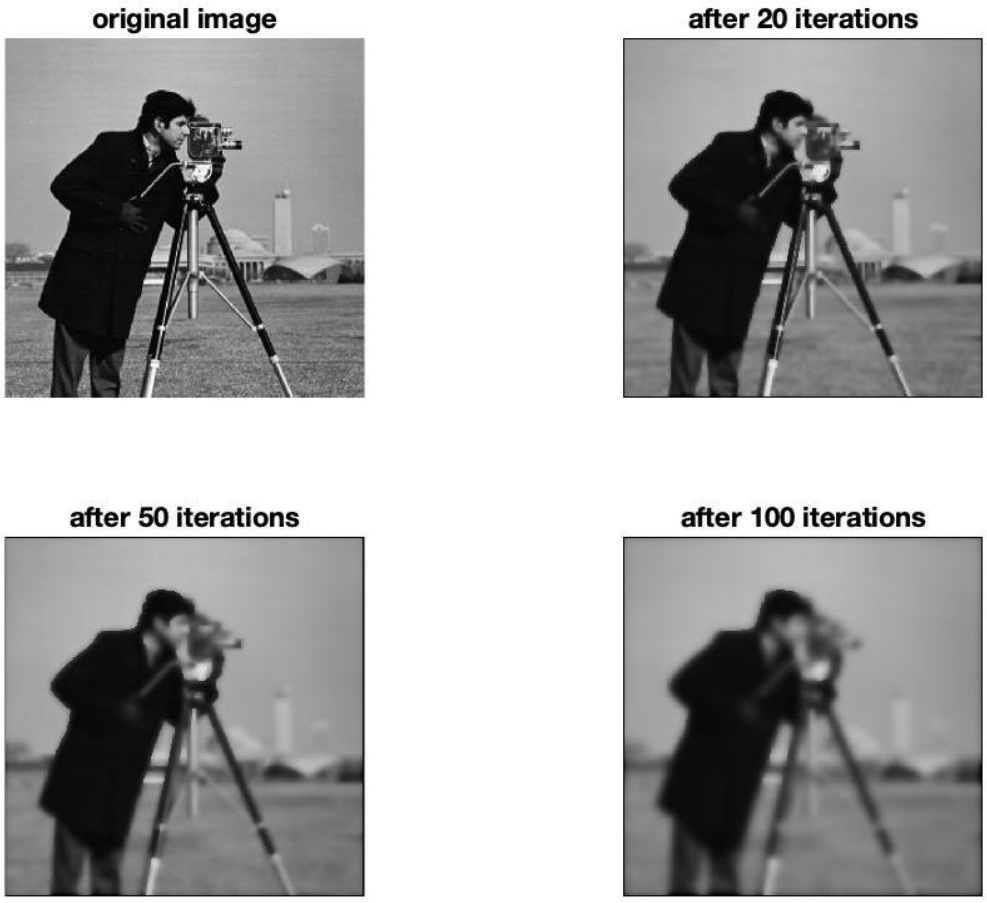


Figure 10 Applying anisotropic diffusion for up to 100 iteration. k = 25, g(.) = inverse quadratic.

When k is 50, the result using exponential function as the conduction coefficient function smoothened a bigger area comparing with that when k = 25. Only the edge that separates regions having higher contrast was preserved after more than 50 iterations. When the pixel value difference between two adjacent regions was relatively small, the edge was blurred as well (Figure 11). The process that uses g(.) = inverse quadratic function (Figure 12) blurred the edge even faster than k = 25 (Figure 10).

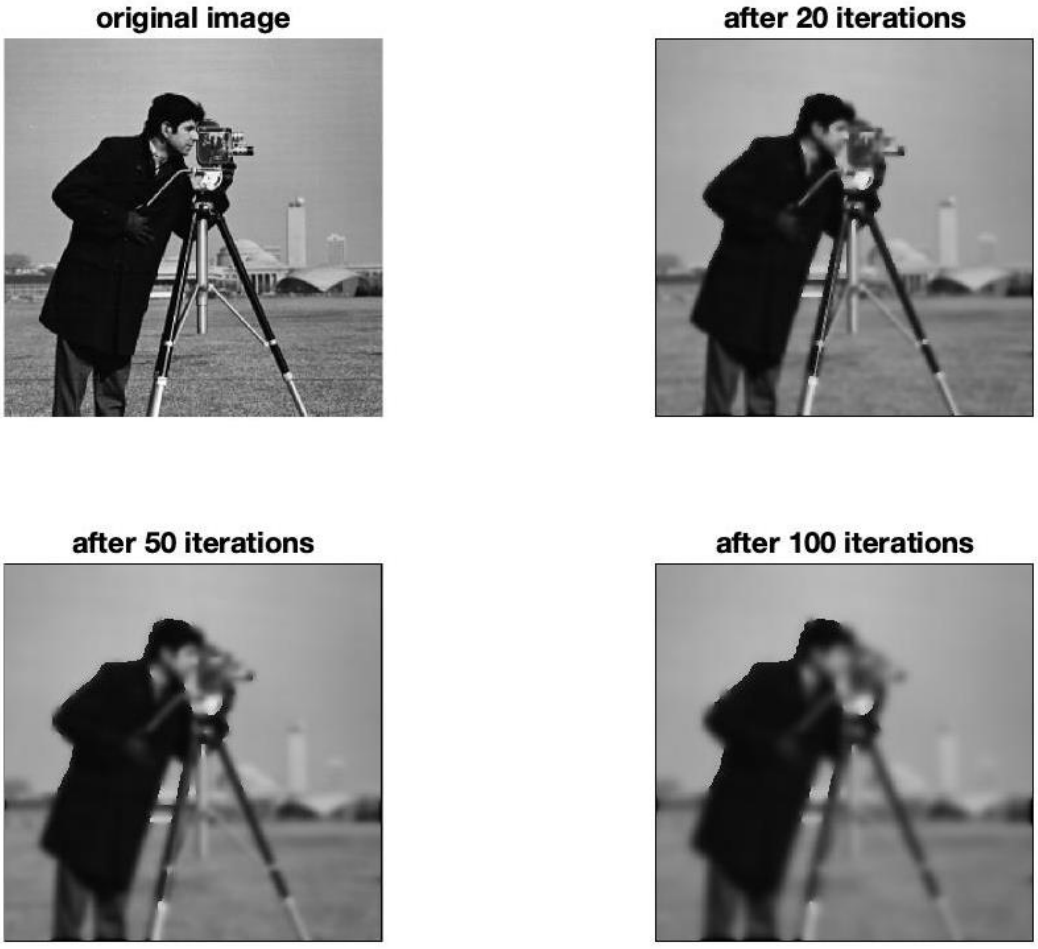


Figure 11 Applying anisotropic diffusion for up to 100 iteration. k = 50, g(.) = exponential.

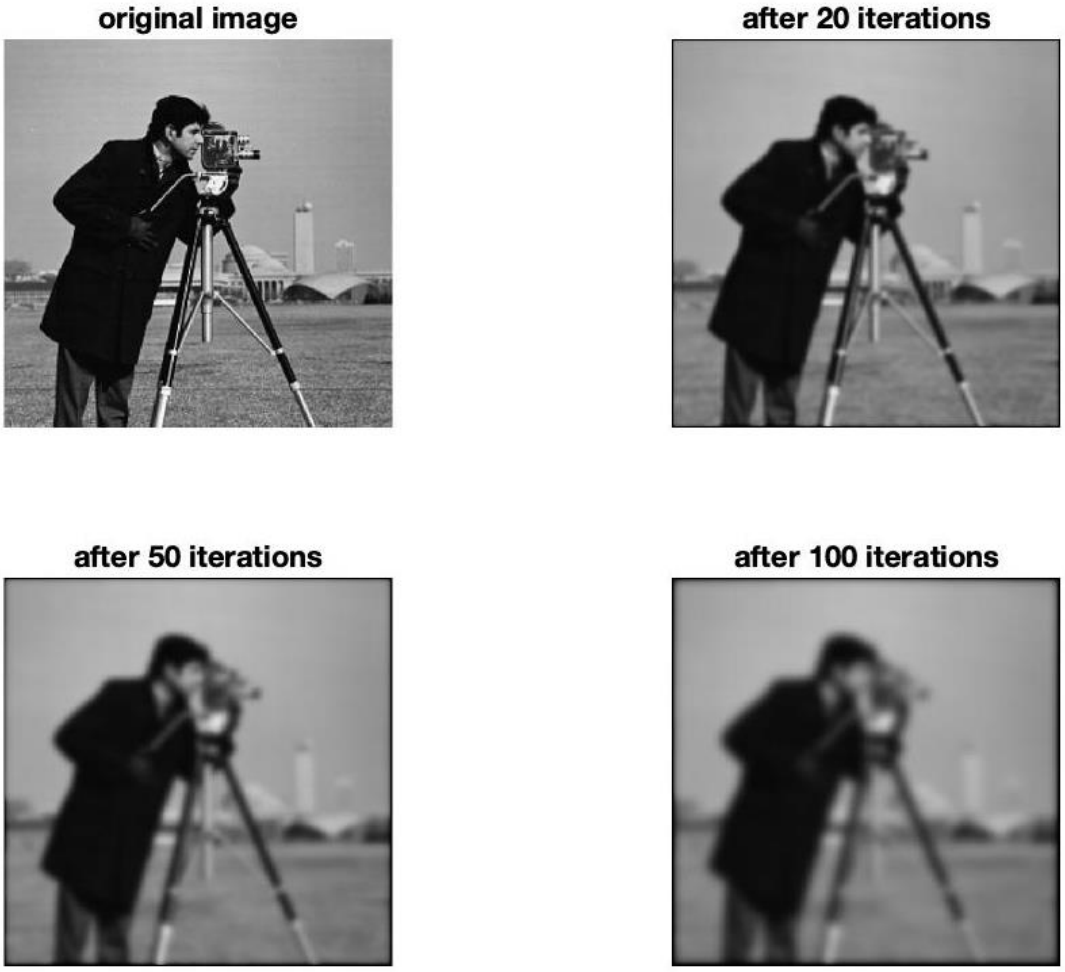


Figure 12 Applying anisotropic diffusion for up to 100 iteration. k = 50, g(.) = inverse quadratic.

2.3 Discussion

In general, the conduction coefficient function $g(\cdot)$ distinguishes the interior region and the border, and the parameter k determines a threshold to distinguish the border. When k is the same, the exponential function converges faster than the inverse quadratic function, so it is more sensitive to the border, thus preserving the edges better. The inverse quadratic function, in contrast, can remove the noise in the wide region better.

The "cwheelnoise" image is relatively simple. Each region has regular and well-defined shape, and the contrast between each region is relatively high. The content in "cameraman" is relatively complex. For example, the tripod has thin and long shape, and the lower part between the grass and the lower part of the tripod is very small. As a result, when k is 25, the edges are well-preserved in both images, though the noise are not totally gone in the "cwheelnoise" image. When k becomes 50, the edges in "cwheelnoise" image are still roughly kept, but most edges are blurred in the "cameraman" image.

E. Conclusion

In conclusion, the project firstly evaluated the behaviors of different 5x5 filters based on various algorithms with repeated iterations. Properties were examined through both the output images and histograms. Those filters all show great ability to remove noise from image but differ in specific applications. In our experiment, median filter preserves the edge better than mean filter, and other ono-linear filters also show improvement compared with mean filter for outliers' details, except that the sigma filter left some noises.

For the anisotropic diffusion, at low iterations, it shows great performance at low iterations to reduce noise but tends to lose details at high iterations. Finding a proper parameter K helps preserve the segmented image details. When parameter k is the same, the exponential function is more sensitive to the border compared with inverse quadratic function, thus preserving the edges better. The inverse quadratic function, in contrast, can remove the noise in the wide region better. It is concluded that the behavior differs on images that whether it is dominated by a lot of small details or not.

Thus, for different filters, their success in a specific application depends heavily on the characteristics of the problem, including finding an optimal compromise of image structures, number of iterations, parameters and algorithms.