CSE585/EE555: Digital Image Processing I

Computer Project # 3:

# Nonlinear Filtering and Anisotropic Diffusion

*Lindsey Schwartz, Ke Liang, Xilun Liu*

*Date: 03/29/2019*

---

## A. Objectives

The objectives of this project are divided into two parts. The first is understanding Nonlinear Filtering by implementing several filters and performing analysis on the results after 5 iterations. The second is understanding Anisotropic Diffusion for Image Filtering by implementing the anisotropic diffusion algorithm.

## B. Methods

### Part 1 (Nonlinear Filtering)

The goal of this section was to implement four different filters (median, alpha-trimmed mean, sigma, and symmetric nearest-neighbor mean) and analyze the histogram, mean, and standard deviation of an ROI after one and five iterations for each filter. To generate the results shown in this report, the file nonlinear_filtering.m should be run and the file path should point to the test image "disk".

(a) 5x5 Median filter

The median filter replaces the central pixel in a neighborhood with the median value from the neighborhood. This equation was implemented using MATLAB's built in median filter, *medfilt2*. The function takes in the image and an optional parameter for the size of the window, in our case [5 5], and returns the median filtered image. Once the median filtered image was returned, a region of interest (ROI) on the interior of the large disk was manually defined for analysis. MATLAB's *mean* and *std* functions were used to get the average and standard deviation, respectively, of the ROI. MATLAB's *imhist* function was also used to view the histogram of the entire image. This process was repeated for both one and five iterations.

(b) 5x5 Alpha-trimmed mean ($\alpha = 0.25$)

The equation for the alpha-trimmed mean filter is as follows,

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

where,

$n$ is the window length

$\alpha$ ranges between 0 and 0.5

This equation was implemented using MATLAB's built in order-statistic function, *ordfilt2*. This function replaces each pixel in the image with the value at the specified index from the sorted neighborhood array. The function takes in the image, specified index, and the size of the neighborhood. A *for loop* was used in conjunction with the *ordfilt2* function to loop through indices $\lfloor \alpha n \rfloor + 1$ to $n - \lfloor \alpha n \rfloor$ and return an output image for each index. The output images are averaged to return a final alpha-trimmed mean filter that ignores values outside the specified indices $\lfloor \alpha n \rfloor + 1$ to $n - \lfloor \alpha n \rfloor$.

Once the alpha-trimmed mean filtered image was returned, the same ROI on the interior of the large disk was analyzed using MATLAB's *mean* and *std* functions. The histogram of the entire image was also produced. This process was repeated for both one and five iterations.

(c) 5x5 Sigma filter ($\sigma = 20$)

The equation for the sigma filter is as follows,

$$\hat{y}_k = \frac{1}{N_C} \sum_{i=-N}^{N} \delta_i \, x_{k-i}$$

where,

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

$$N_C \triangleq \# \text{ of points } x_{k-i} \text{ having } \delta_i = 1$$

To implement this equation, a local neighborhood was created and centered at each pixel in the image using a *for loop*. The difference between the center pixel and each pixel in

the neighborhood was computed. The central pixel value was replaced by the average of those pixels in the neighborhood that had a difference of $<= 2\sigma$.

Once the sigma filtered image was returned, the same ROI on the interior of the large disk was analyzed using MATLAB's *mean* and *std* functions. The histogram of the entire image was also produced. This process was repeated for both one and five iterations.

(d) 5x5 Symmetric nearest-neighbor mean

This method takes pixels that are symmetrically opposite one another in a given neighborhood and selects the pixel that is most similar (minimizes the difference in pixel values) and stores it in an array. Once this has been done for each pair of symmetrically opposite pixels, the mean of the stored array is computed and assigned as the new image value at the location of the center pixel in the neighborhood.

Once the symmetric nearest-neighbor mean filtered image was returned, the same ROI on the interior of the large disk was analyzed using MATLAB's *mean* and *std* functions. The histogram of the entire image was also produced. This process was repeated for both one and five iterations.

## Part 2 (Anisotropic Diffusion for Image Filtering)

The Anisotropic Diffusion Equation is

$$\frac{\partial I}{\partial t} = c(x,y,t)\Delta I + \nabla c \cdot \nabla I$$

To work image behavior into c(x,y,t), let

$$c(x,y,t) = g(\| \bigtriangledown I(x,y,t) \|)$$

$$\uparrow$$

norm (magnitude) of gradient of $I$

$$= g\left(\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}\right)$$

As for 2 choices for g(), the first one is exponential and the second one is inverse quadratic, which is shown below.

1. $g(\| \nabla I \|) = \exp\{-(\| \nabla I \| / k)^2\}$
   – high–contrast edges favored

2. $g(\| \nabla I \|) = \frac{1}{1 + (\|\nabla I\|/k)^2}$
   – wide regions favored

In this report, we choose K=50 to get the results.
A discrete numerical solution can be derived for the anisotropic case using the method as follows:

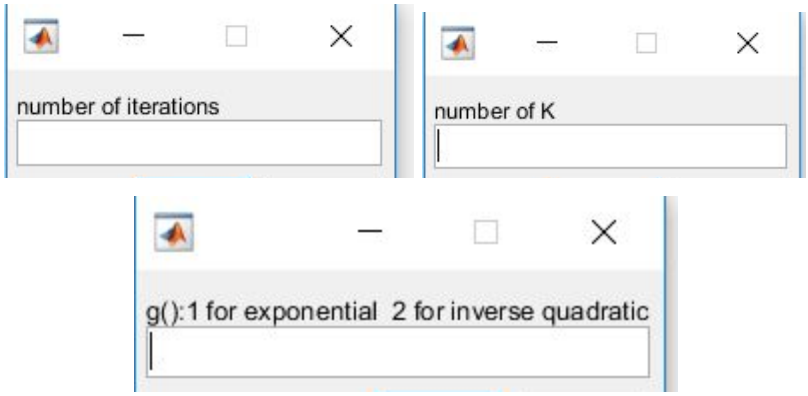
$$I_{i,j}^{t+1} = I_{i,j}^{t} + \lambda[c_N^t \nabla_N I + c_S^t \nabla_S I + c_E^t \nabla_E I + c_W^t \nabla_W I]_{i,j}^t$$

$N, S, E, W$ North, South, East, West neighbors of $(i, j)$

$0 \leq \lambda \leq 1/4$ a parameter

We choose λ=0.25.

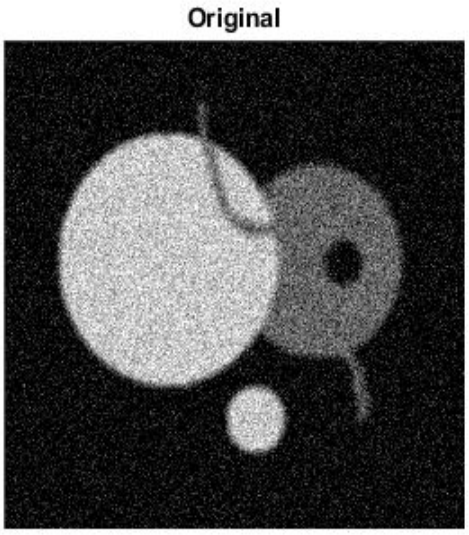
We make the codes "anistro_cwheelnoise.m", "anistro_lake.m" and function "ConCoefficient.m" for the questions. When you run the code you can print in "the number of iterations", "number of K" and "g():1 for exponential 2 for inverse quadratic" as you want, which are shown below.



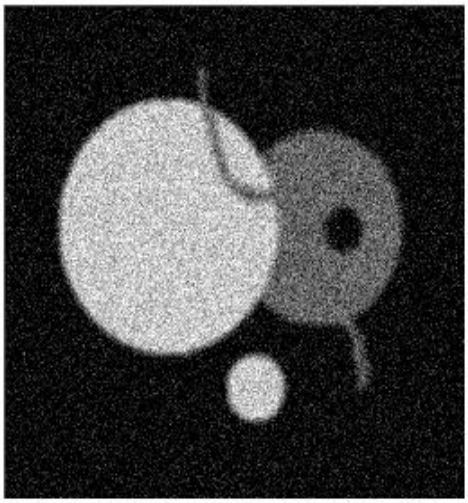
## C. Results

### Part 1 (Nonlinear Filtering)

This part of the project focused on the effects of repeated application of several filtering algorithms. The results were observed in the form of image output and by obtaining the histogram of the image, as well as mean and standard deviation statistics for a particular ROI. All of the algorithms considered are implemented for use on grayscale images. Below is the original image, “disks”, used for this section. The ROI is displayed over the original image using a red box on the right.



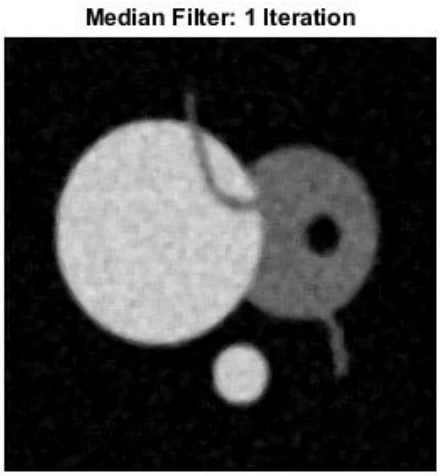

Original disk image

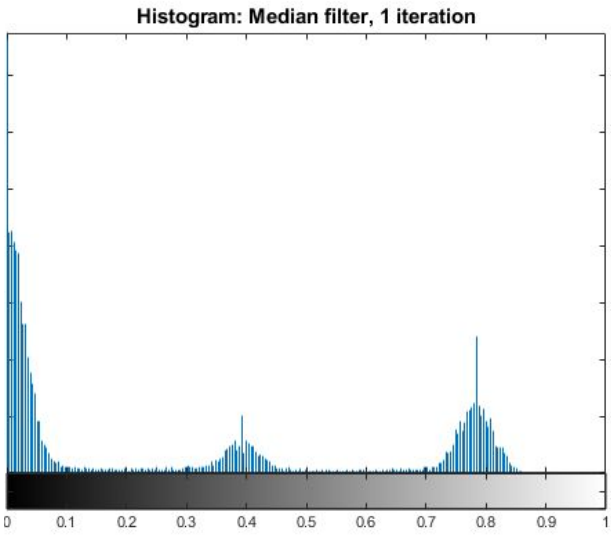


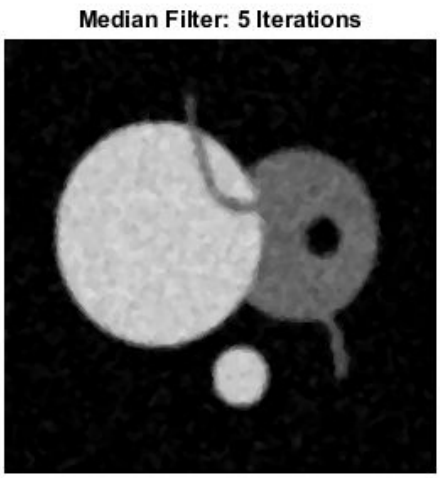
grayscale image

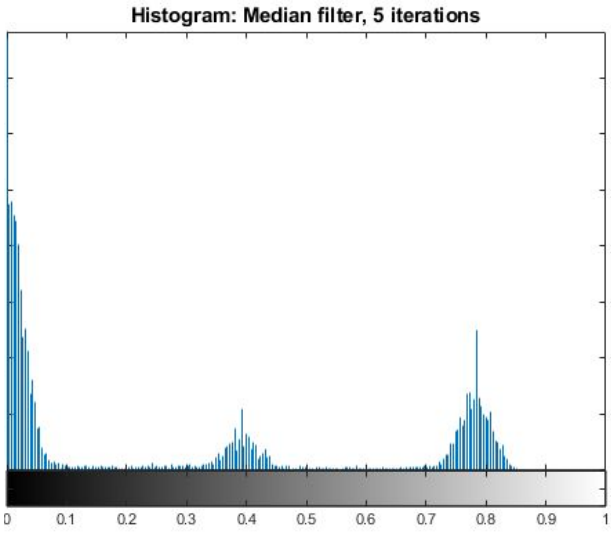
The following figures are the results for the 5x5 median filter after 1 and 5 iterations, respectively. The table contains the mean and standard deviation statistics for the ROI for both 1 and 5 iterations.

| **Number of Iterations** | **Mean** | **Standard Deviation** |
|---|---|---|
| 1 | 0.7851 | 0.0279 |
| 5 | 0.7853 | 0.0258 |



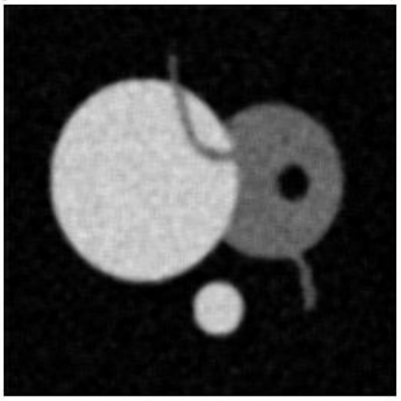



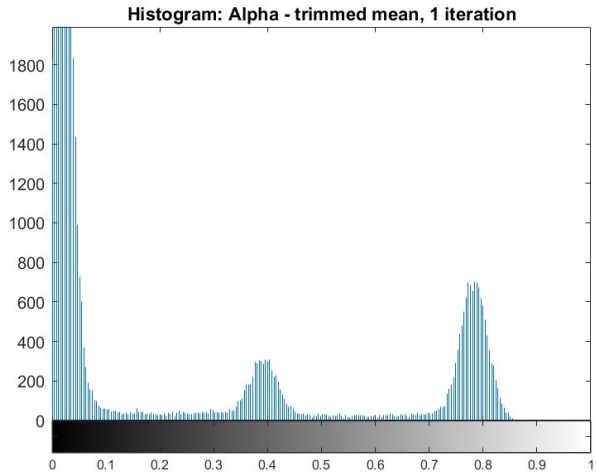




The following figures are the results for the 5x5 alpha-trimmed mean filter after 1 and 5 iterations, respectively. The table contains the mean and standard deviation statistics for the ROI for both 1 and 5 iterations.

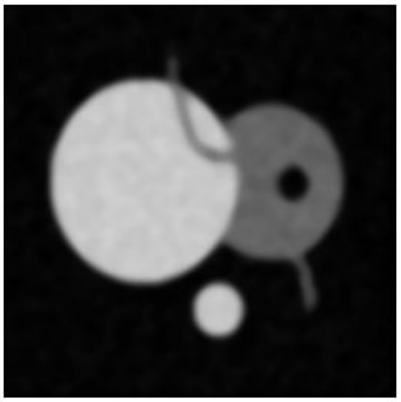
| Number of Iterations | Mean | Standard Deviation |
|---|---|---|
| 1 | 0.7851 | 0.0244 |
| 5 | 0.7854 | 0.0117 |

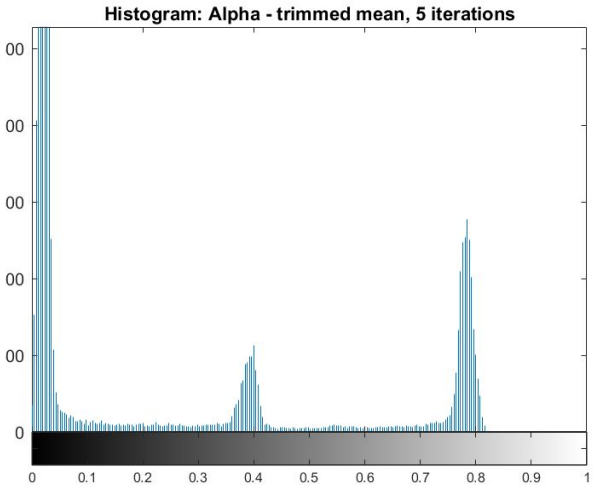
Alpha - trimmed mean: 1 Iteration





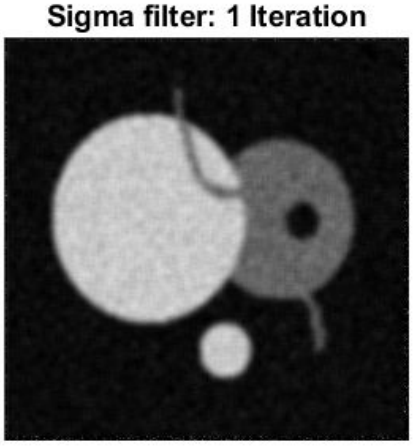

Alpha - trimmed mean: 5 Iterations

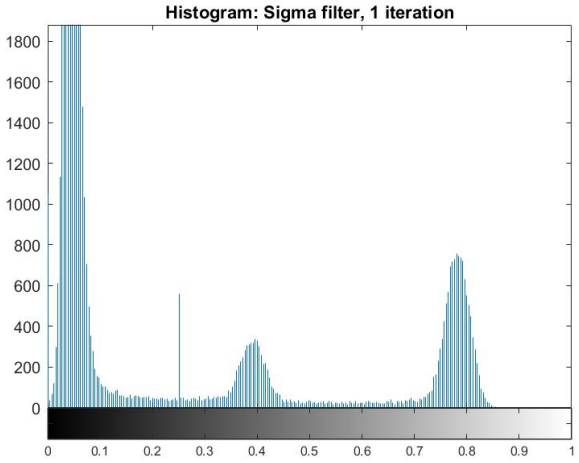


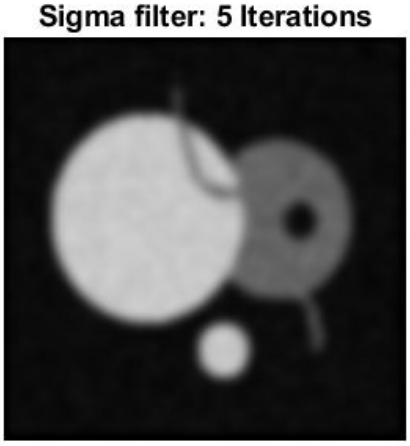


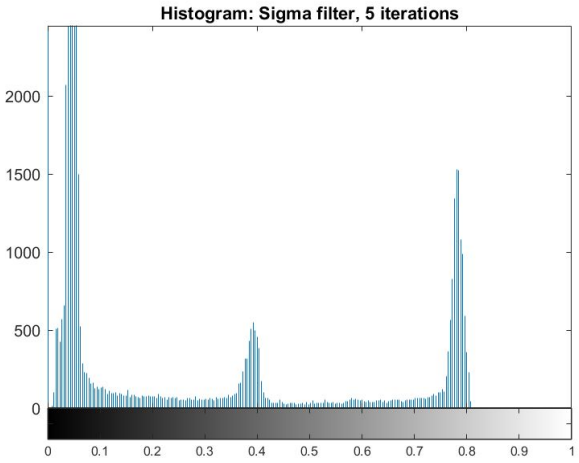

The following figures are the results for the 5x5 sigma filter (σ= 20) after 1 and 5 iterations, respectively. The table contains the mean and standard deviation statistics for the ROI for both 1 and 5 iterations.

| Number of Iterations | Mean | Standard Deviation |
|---|---|---|
| 1 | 0.7843 | 0.0221 |
| 5 | 0.7846 | 0.0096 |



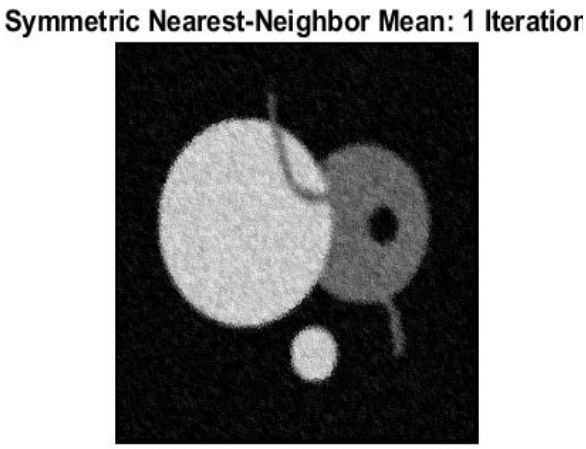



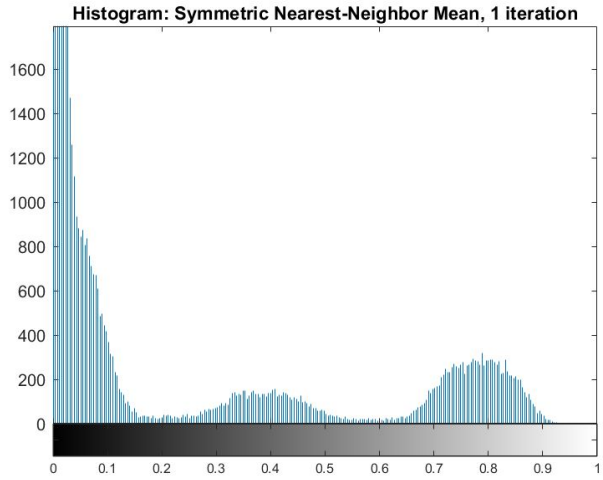



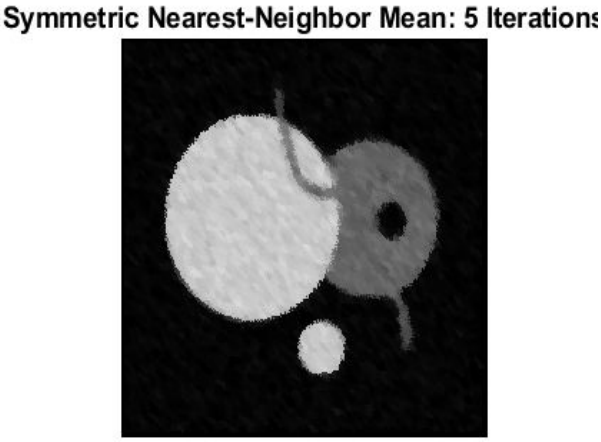



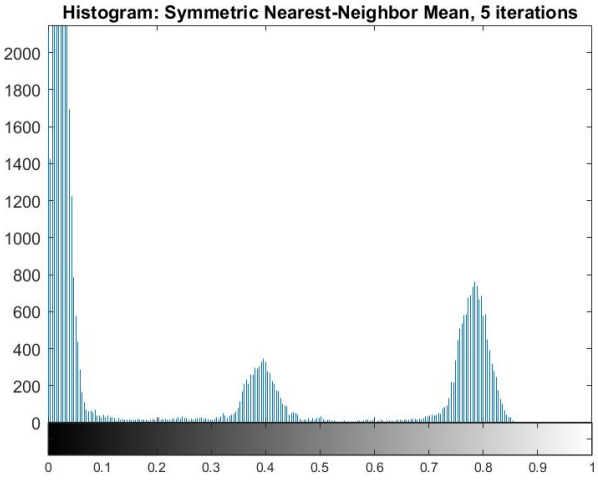

The following figures are the results for the 5x5 symmetric nearest-neighbor mean filter after 1 and 5 iterations, respectively. The table contains the mean and standard deviation statistics for the ROI for both 1 and 5 iterations.

| Number of Iterations | Mean | Standard Deviation |
|---|---|---|
| 1 | 0.7847 | 0.0563 |
| 5 | 0.7846 | 0.0243 |



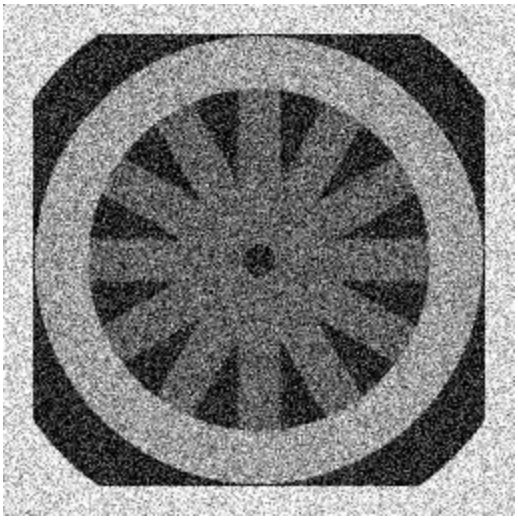




Part 2 (Anisotropic Diffusion for Image Filtering)

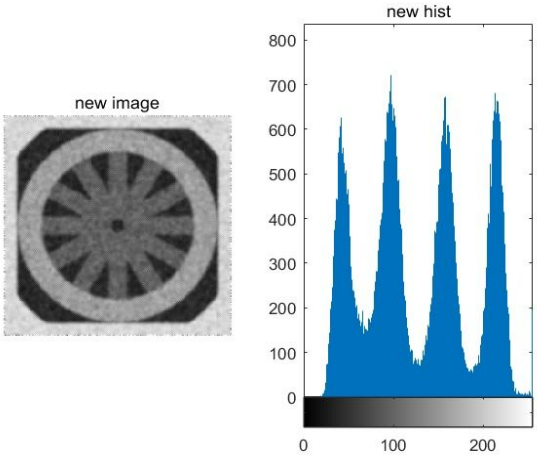
The results for each iterations are stacked vertically. In the segmented image sections, we are attempting to best extract the spokes of the wheel. Each image has a plot of its horizontal cross section at y=128.

a) The following figure is the original figure “cwheelnoise”.

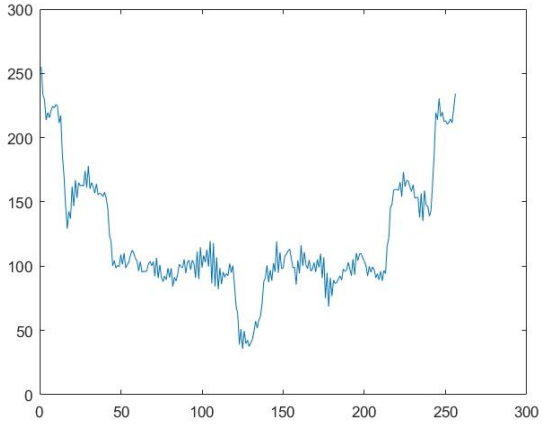


Original cwheelnoise.gif

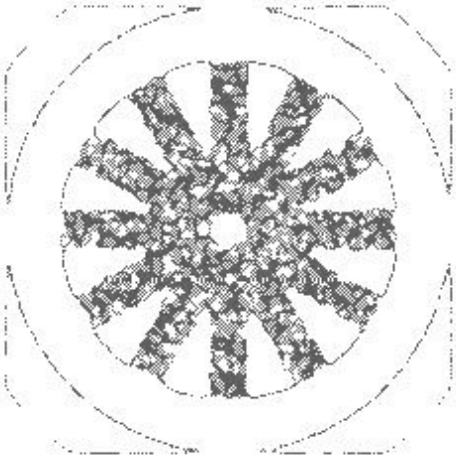
As for g() is exponential and the K=50, the following figures are the anisotropic-diffusion results after  the 5 iterations.




5 iteration cwheelnoiseand the historgram ( g() is exponential K=50)

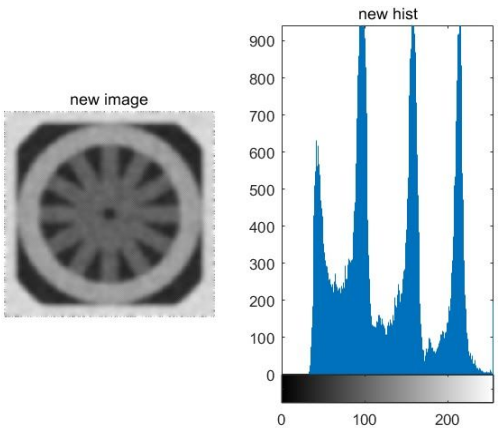



the line y = 128 through the 5 iteration cwheelnoise( g() is exponential K=50 )

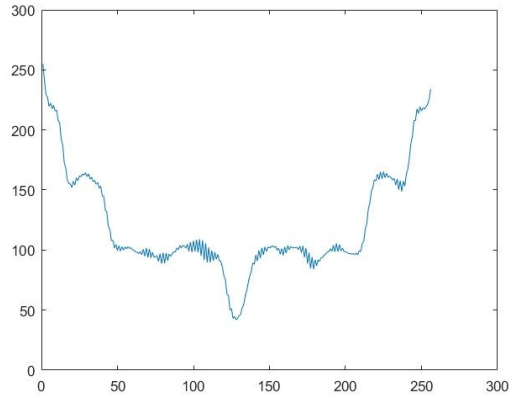


Segmented version of the 5 iteration cwheelnoise( g() is exponential K=50 )

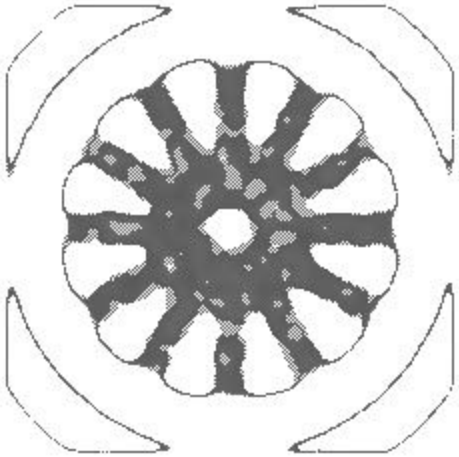
As for g() is exponential and the K=50, the following figures are the anisotropic-diffusion results after  the 20 iterations.




20 iteration image and the historgram
( g() is exponential K=50 )

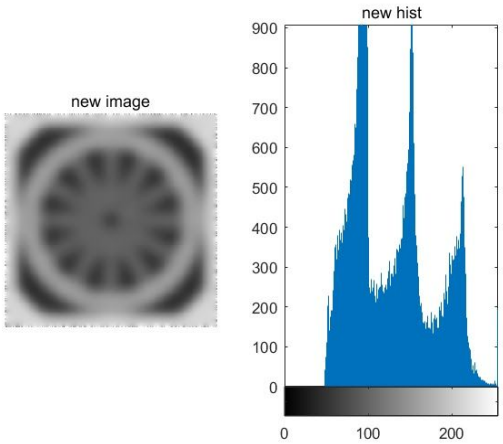


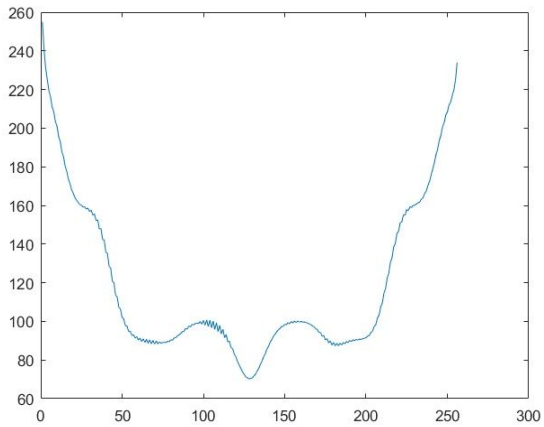

the line y = 128 through the 20 iteration
cwheelnoise( g() is exponential K=50)



Segmented version of the 20 iteration cwheelnoise( g() is exponential K=50)

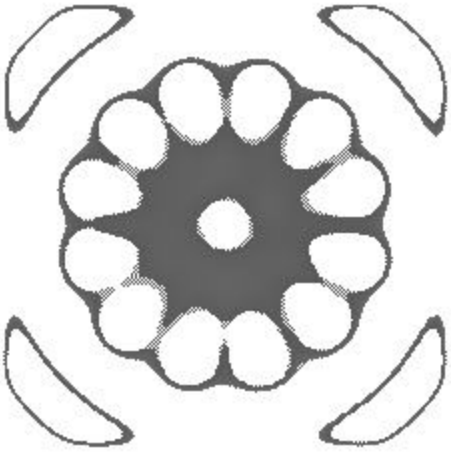
As for g() is exponential and the K=50, the following figures are the anisotropic-diffusion results after the 100 iterations.



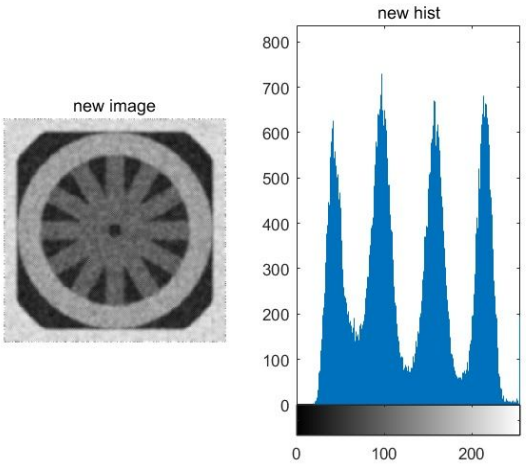

100 iteration cwheelnoise and the historgram ( g() is exponential K=50)

the line y = 128 through the 100 iteration cwheelnoise ( g() is exponential K=50)

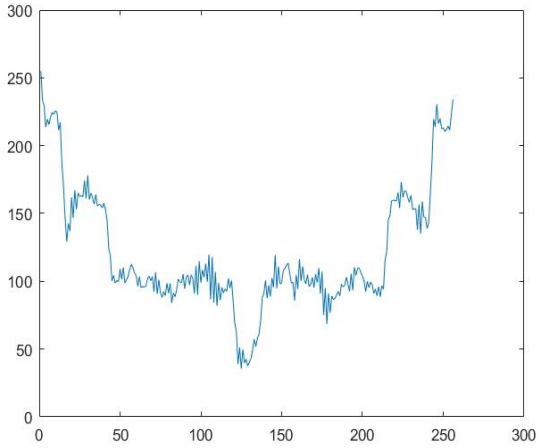


Segmented version of the 100 iteration cwheelnoise( g() is exponential K=50)

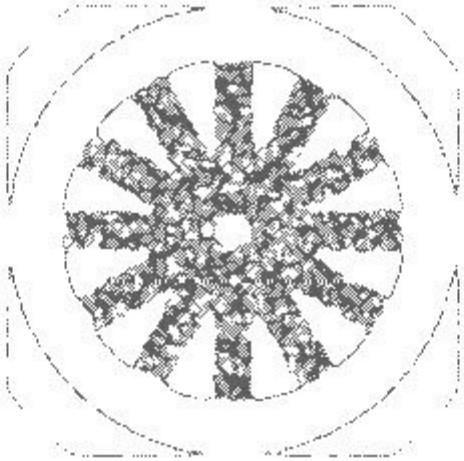
As for g() is inverse quadratic and the K=50, the following figures are the anisotropic-diffusion results after the 5 iterations.




5 iteration cwheelnoise and the historgram ( g() is inverse quadratic K=50)

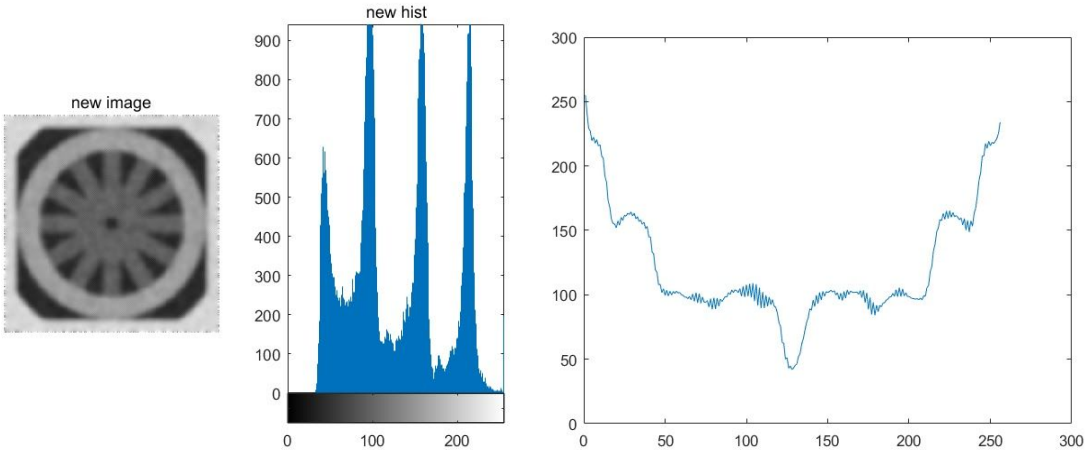



the line y = 128 through the 5 iteration cwheelnoise( g() is inverse quadratic K=50)



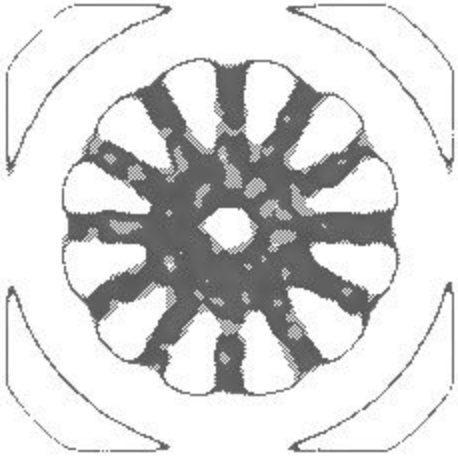
Segmented version of the 5 iteration cwheelnoise( g() is inverse quadratic K=50)

As for g() is inverse quadratic and the K=50, the following figures are the anisotropic-diffusion results after the 20 iterations.



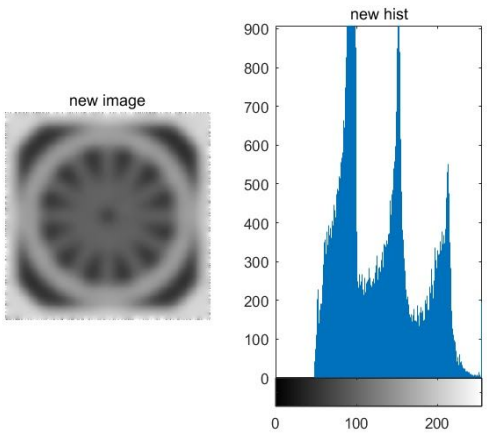

20 iteration cwheelnoise and the historgram ( g() is inverse quadratic K=50)

the line y = 128 through the 20 iteration cwheelnoise( g() is inverse quadratic K=50)

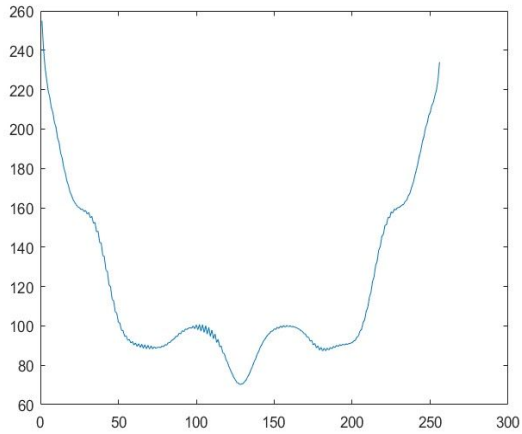


Segmented version of the 20 iteration cwheelnoise( g() is inverse quadratic K=50)

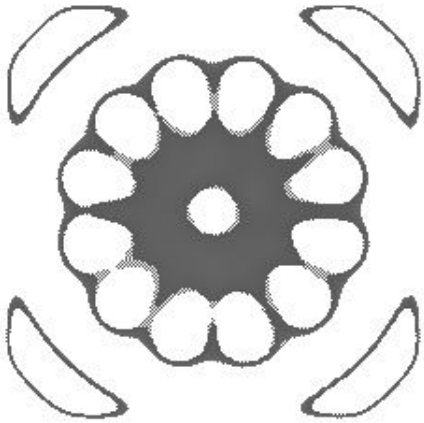
As for g() is inverse quadratic and the K=50, the following figures are the anisotropic-diffusion results after the 100 iterations.




100 iteration cwheelnoise and the historgram ( g() is inverse quadratic K=50)

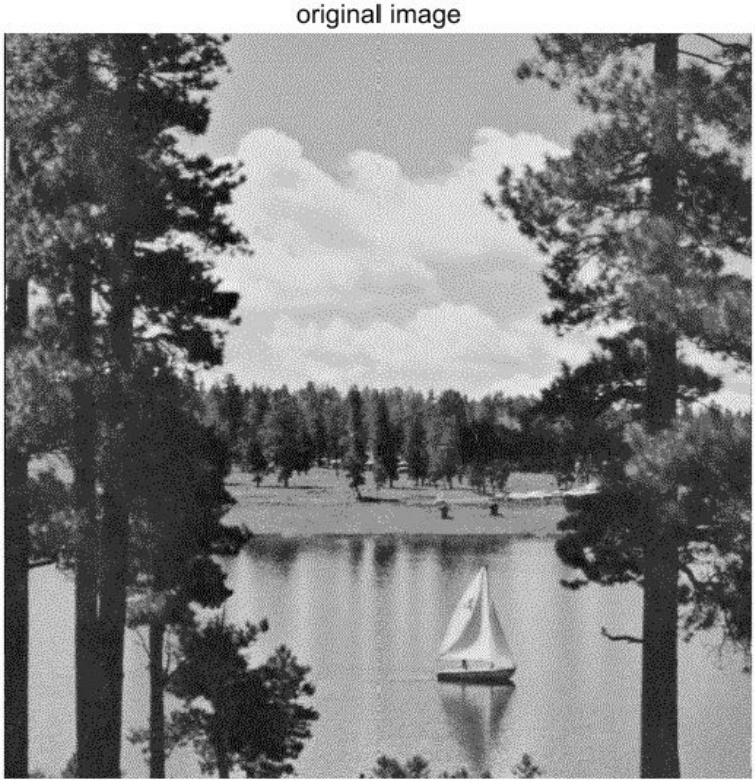



the line y = 128 through the 100 iteration cwheelnoise ( g() is inverse quadratic K=50)



Segmented version of the 100 iteration cwheelnoise ( g() is inverse quadratic K=50)

b) The following figure is the original figure “lake”.




Original Image of Lakes

As for g() is exponential and the K=50, the following figures are the anisotropic-diffusion results after  the 5, 20 and 100 iterations.


5 iteration image Lakes
( g() is exponential K=50)


20 iteration image Lakes
( g() is exponential K=50 )


100 iteration image Lakes( g() is exponential K=50)

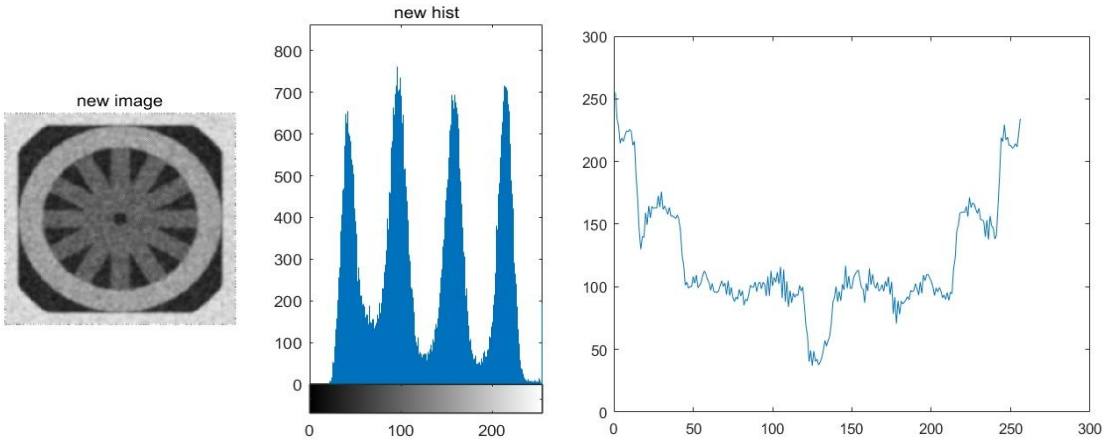
As for g() is inverse quadratic and the K=50, the following figures are the anisotropic-diffusion results after the 5, 20 and 100 iterations.


5 iteration image Lakes
( g() is inverse quadratic K=50)


20 iteration image Lakes
( g() is inverse quadratic K=50)

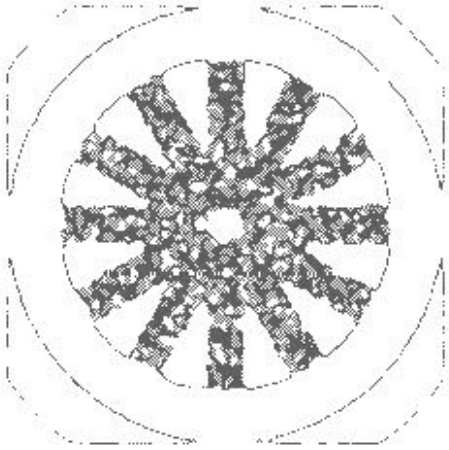

100 iteration image Lakes( g() is inverse quadratic K=50)

As for how K affects the result, we can run the the code again, this time K=30 and g() is exponential and inverse quadratic, the following figures are the anisotropic-diffusion results after the 5, 20 and 100 iterations. And compare the results to the former results.

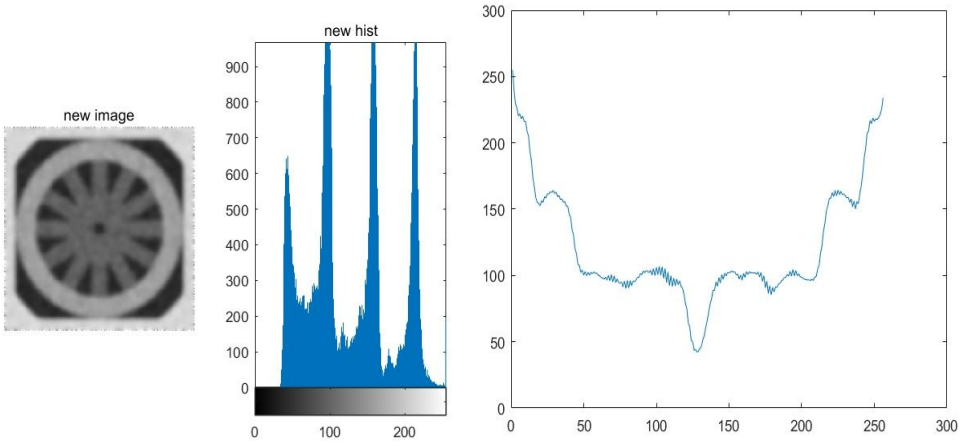



5 iteration cwheelnoiseand the historgram ( g() is exponential K=30)

the line y = 128 through the 5 iteration cwheelnoise( g() is exponential K=30)

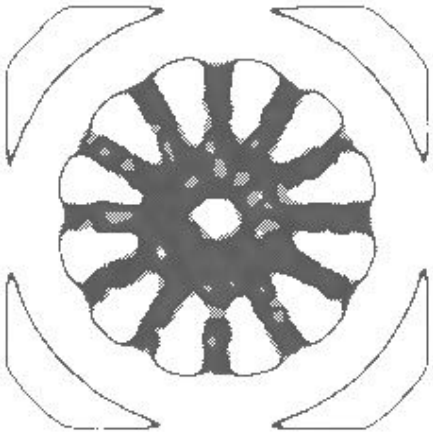


Segmented version of the 5 iteration cwheelnoise ( g() is exponential K=30)

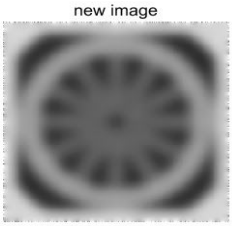


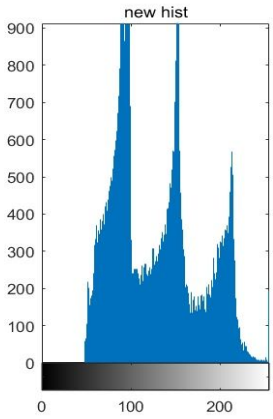

20 iteration cwheelnoiseand the historgram ( g() is exponential K=30 )

the line y = 128 through the 20 iteration cwheelnoise( g() is exponential K=30)

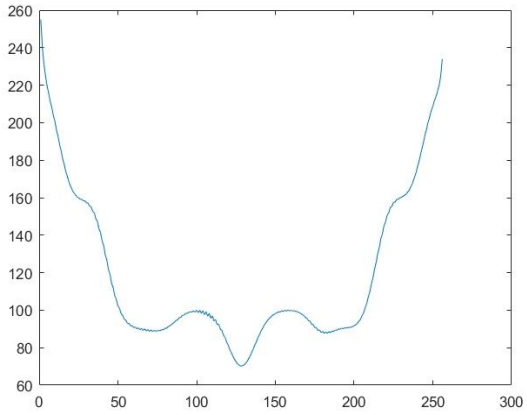


Segmented version of the 20 iteration cwheelnoise ( g() is exponential K=30)

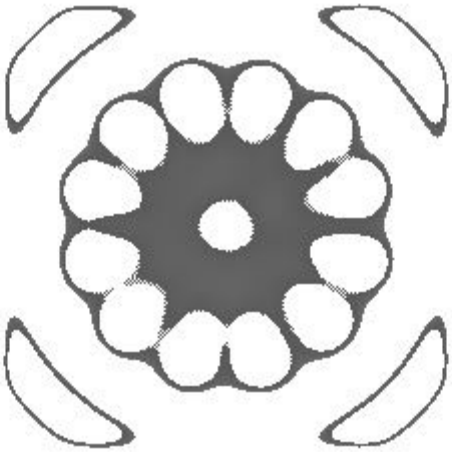


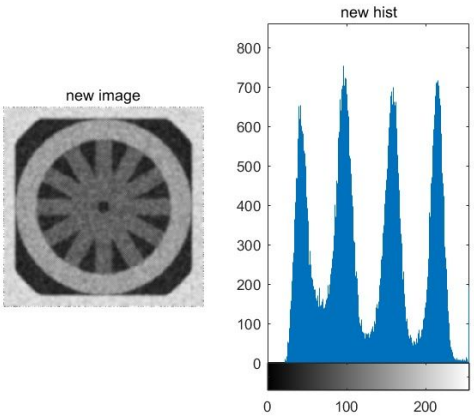

100 iteration cwheelnoiseand the historgram (g() is exponential K=30)



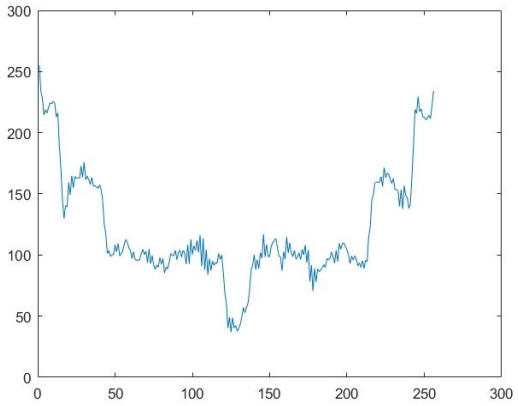

the line y = 128 through the 100 iteration cwheelnoise( g() is exponential K=30)



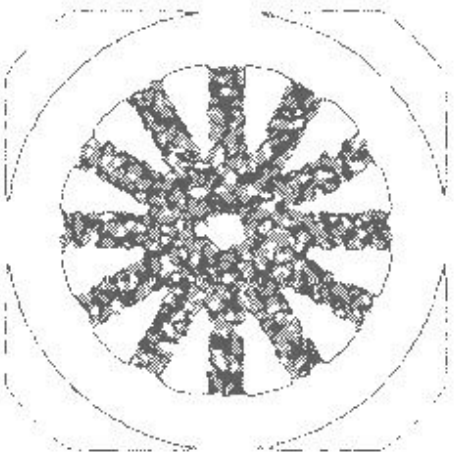
Segmented version of the 100 iteration cwheelnoise ( g() is exponential K=30)



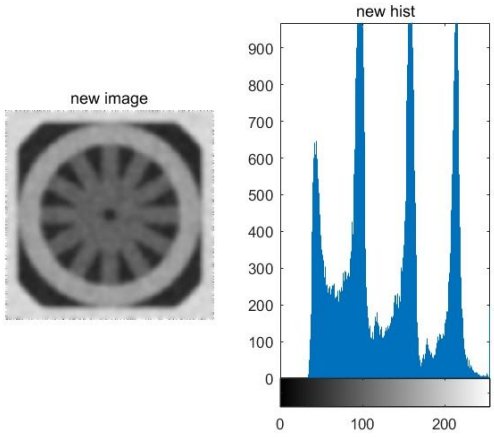

5 iteration cwheelnoiseand the historgram ( g() is inverse quadratic K=30)



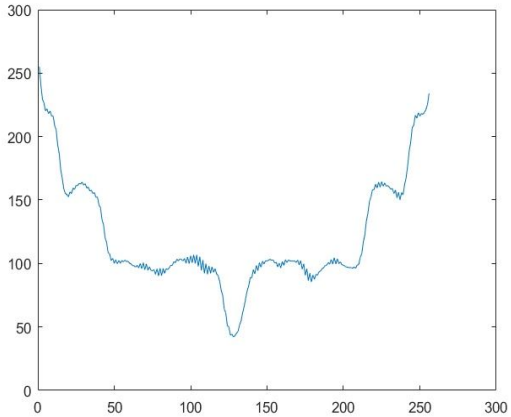

the line y = 128 through the 5 iteration cwheelnoise( g() is inverse quadratic K=30)



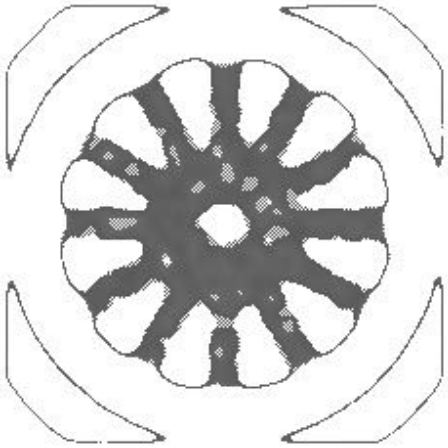
Segmented version of the 5 iteration cwheelnoise ( g() is inverse quadratic K=30)



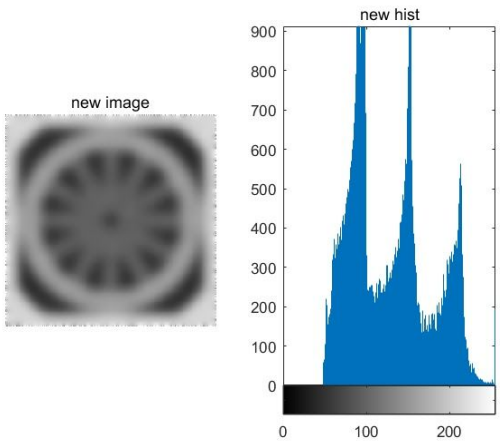

20 iteration cwheelnoiseand the historgram ( g() is inverse quadratic K=30)



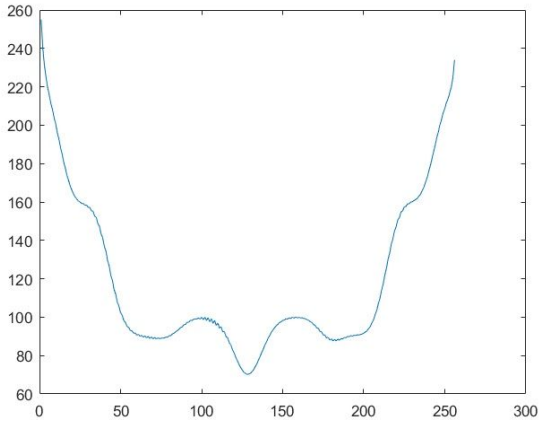

the line y = 128 through the 20 iteration cwheelnoise( g() is inverse quadratic K=30)



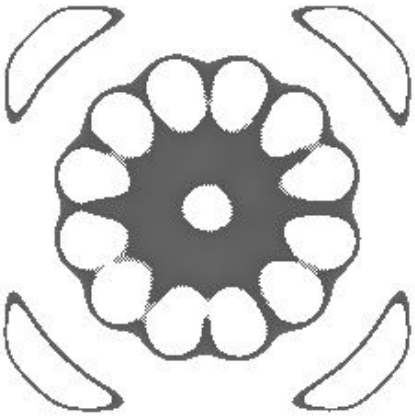
Segmented version of the 20 iteration cwheelnoise ( g() is inverse quadratic K=30)




100 iteration cwheelnoiseand the historgram ( g() is inverse quadratic K=30)




the line y = 128 through the 100 iteration cwheelnoise( g() is inverse quadratic)



Segmented version of the 100 iteration cwheelnoise ( g() is inverse quadratic K=30)

As we can see in the results above we can answer the part (c) on discussing questions on your results of parts as first question.

The anisotropic diffusion works well at low iterations in removing some of the noise. However, as the iterations increase, the image becomes more blurry. This is especially visible in the cross sections of each image at y=128. This fact makes image segmentation much harder as the gray level tend to mesh together to a constant. This is shown in the segmented images as with the higher iterations, the spokes are harder to isolate. With few iterations, the spokes still have visible noise in them which affects the image segmentation preventing a solid image of the spokes to be realized correctly.

The lower K value, K=30, used to generate these images seems to make the results much more usable as far as segmenting goes. The images tend not to blur as much as the iterations increase. In fact, K=30 is very good at preserving the edges throughout the iterations and does a fair job at removing noise and restoring the original, noiseless image. The exponential used for the images shows this preservation of details in the histograms. These histograms have very high peaks with little surrounding noise when compared to the quadratic.

It appears that K controls the amount of diffusion the noise is spread throughout the image. The edges are preserved very well when using the exponential. However, the quadratic is much better at removing noise from large blobs of gray color.

The use of g( ) as an exponential yields preservation of high contrast, large gradient edges in each iterations. There is less blurry and "leakage" at the edges of the spokes and rubber with the exponential. The use of g( ) as a quadratic really favors the large areas of one shade of gray. These areas are much better preserved over many iterations. This fact is especially apparent with the segmented images as the spokes are better resolved at the edges with the exponential rather than the quadratic. This fact is also displayed in the histograms as there appears to be more "peakiness" in the histograms using the exponential and not the quadratic.

As demonstrated with the wheel image, the quadratic tends to better preserve large blobs of color and the exponential better preserves edges and details. This difference is very apparent if you inspect the Lake image. The edges are clearing when using the exponential than with the quadratic.

## D. Conclusions

This project demonstrated the method to this project are divided into two parts. The first is understanding Nonlinear Filtering by implementing several filters and performing analysis on the results after 5 iterations. The second is understanding Anisotropic Diffusion for Image Filtering by implementing the anisotropic diffusion algorithm. In the project we learn how to solve these problems by using the methods.