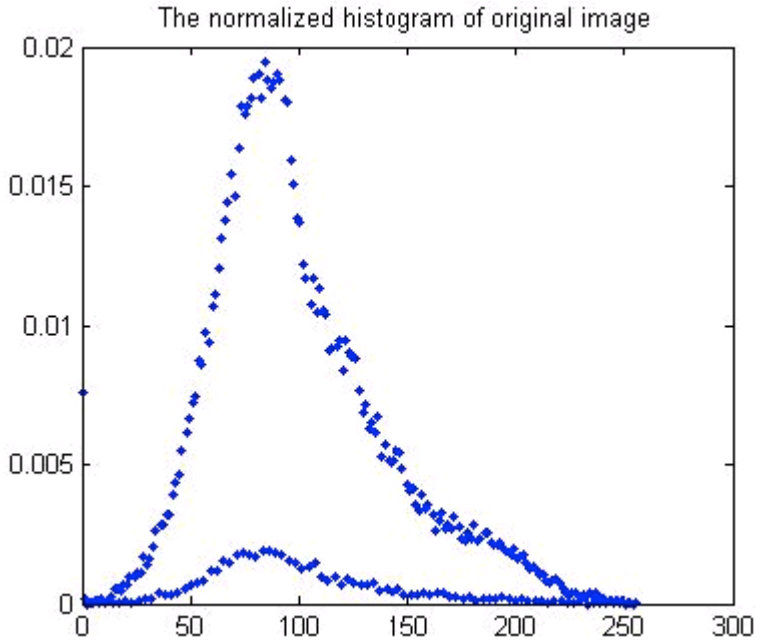
1. Write code (to be submitted) to perform global histogram equalization. Use it to equalize the image “white_house256.jpg”. Plot the histograms of the image before and after equalization and also print the images before and after equalization.

Answer:
The original image is a 256*256 size gray image. Use the Matlab command “imshow( )” to show this image as follow:

In the lecture, we know the probability of each gray level is histogram. The following equation can be used to calculate it:

$$p(r_k) = \frac{r_k}{N} \qquad (1)$$

where $p(r_k)$ denotes the probability of the intensity of k, and r_k is the number of pixels the intensity k has. N is the total number of pixels. The histogram of original image is shown as follow:

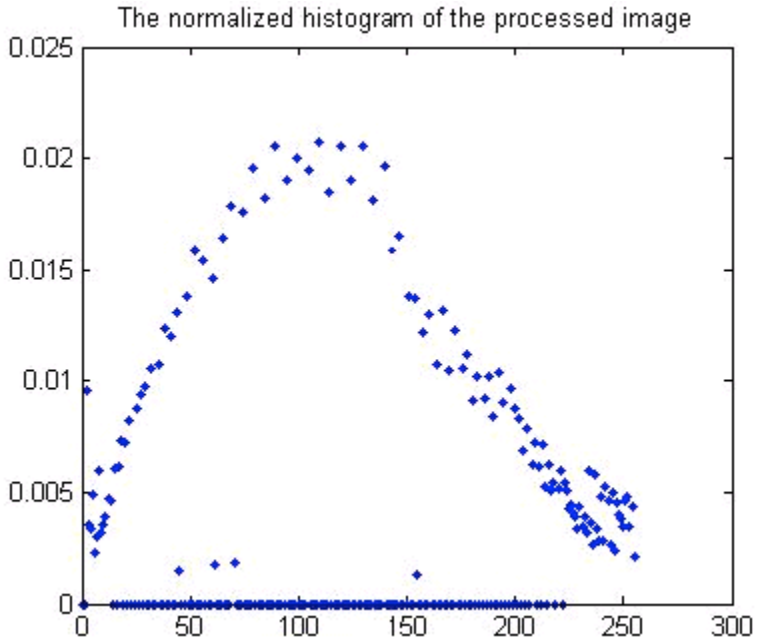


The problem requests to process the image with global equalization. The mapping function from the original image to the new image is as follow:

$$s(k) = 255 \sum_{i=0}^{k} p_i \qquad (2)$$

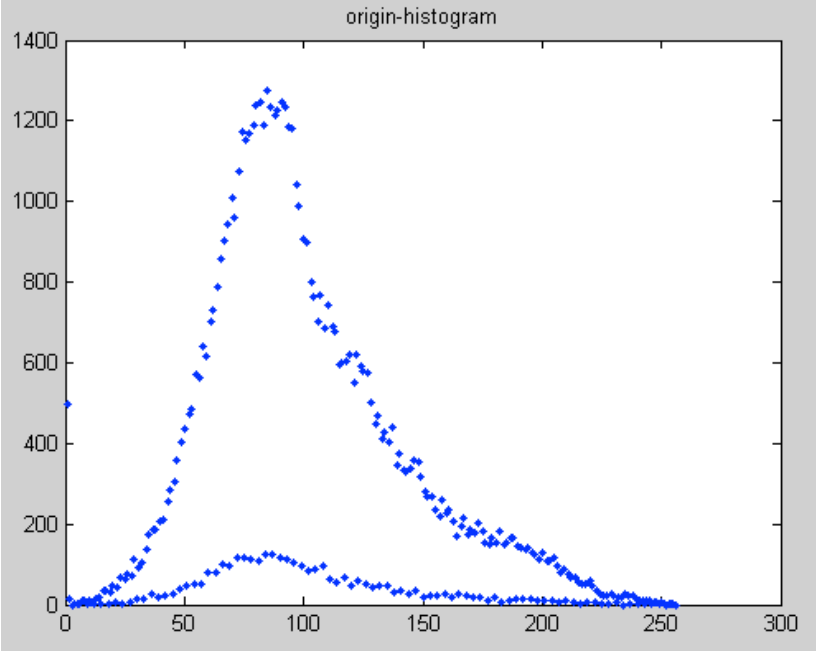
Then we can figure out the 256 intensities of the new image. Mapping the intensities in the old image into theirs corresponding intensity from (2), and the new image are generated. The following image is the new image after global equalization.

Use the same method described in (1) to get the histogram of the new image.



2. Write a code that performs local histogram equalization with a window of 15x15 pixels on the image "white_house256.jpg". Plot the histograms of the images before and after equalization and also print the images before and after equalization. To avoid tiling effects perform the equalizations with overlapped windows.

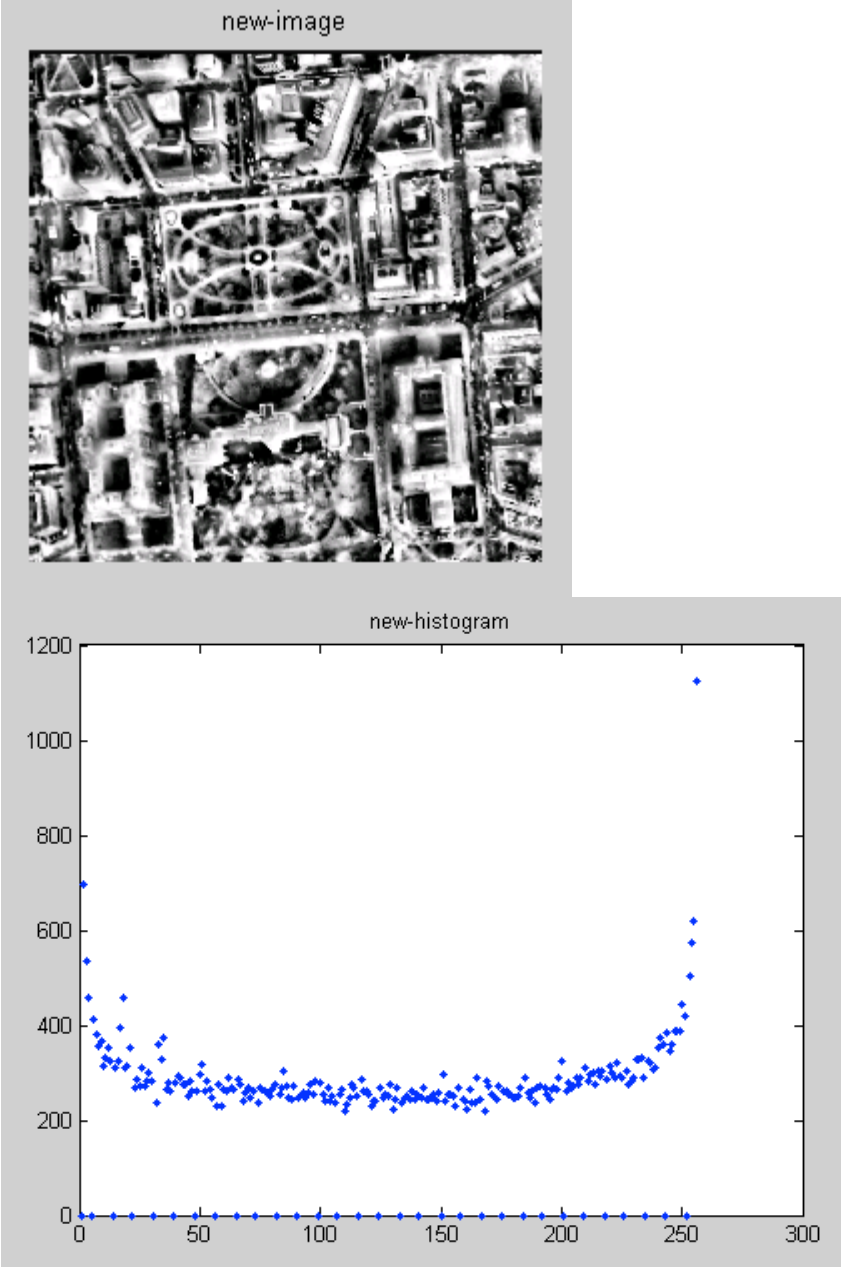
Answer:
Use the same method in Problem 1, we can have the image of original image and the histogram of it.



In the Problem 2, it requests to process the image with local equalization of the window 15*15.

Local equalization is a technique compared to global equalization, which is shown in Problem 1. In local equalization, calculate the histogram of the pixels in the region of the window, then do the histogram equalization as Equation (1) in Problem 1. The center of the window will change to the value of histogram equalization in this area.

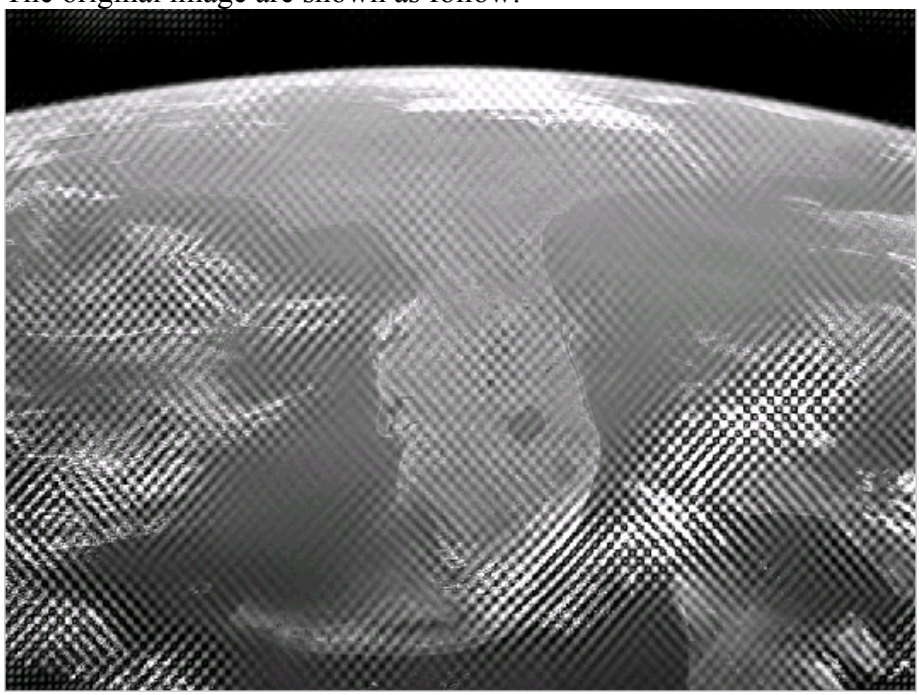
In this problem, I build a 15*15 window and start from the first pixel in the left corner and change it to the value of histogram in the area of 15*15. Right move one pixel do the same operation each time. Then move to the pixel in the next row until to the end of the image. In this way, we can find out the intensities of the new image. Switch the intensities in original image into new intensities and we get the new image. The new image and corresponding histogram are as follow:



3. Analyze the noisy image “nasaNoise.jpg” and find the properties of the interference noise. Design a filter to remove the interference. Plot the Fourier transform of the filter in 3D and print the image before and after filtering. Explain in detail your analysis and design.

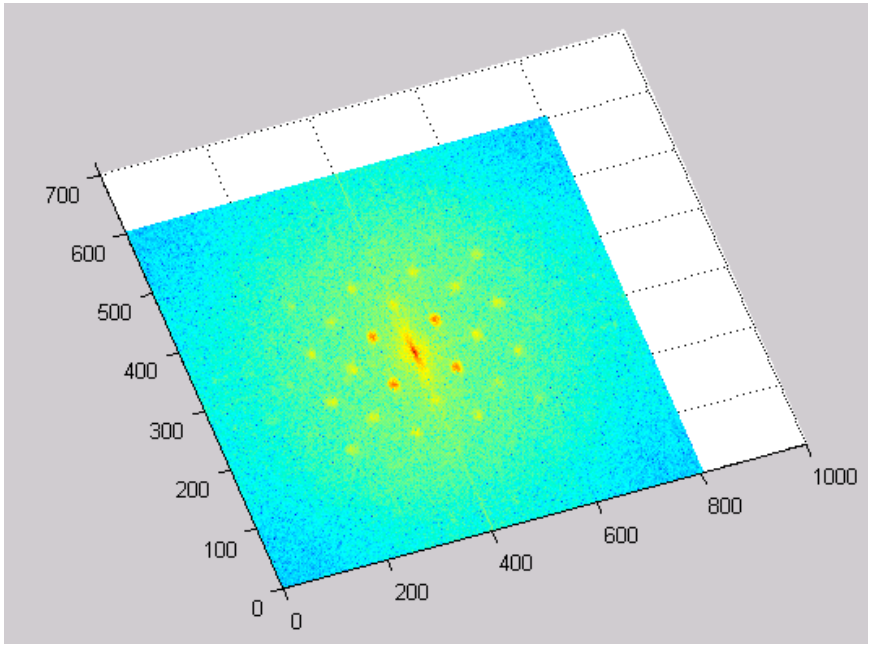
Answer:

The original image are shown as follow:



In this image, we can see it has noises of parallel lines in two diagonal directions which makes the image cannot be seen clearly. The purpose of the process is to remove the parallel noises. I remove the noise with the method in frequency domain. Do the Fourier Transform of the image and shift it to the center of the plot. The Fourier Transform plot can be shown as follow

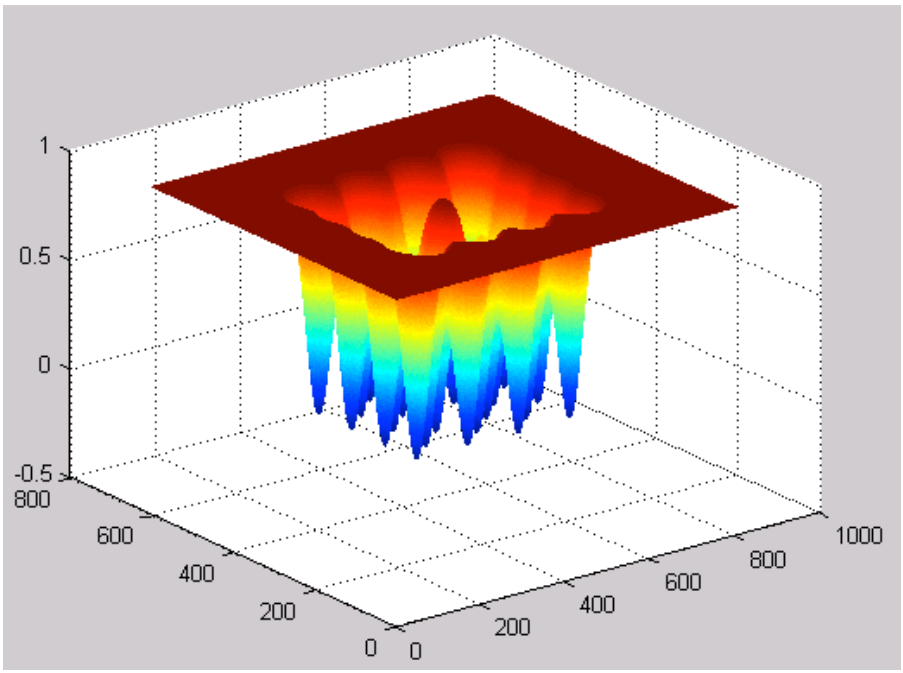
Fourier Transform



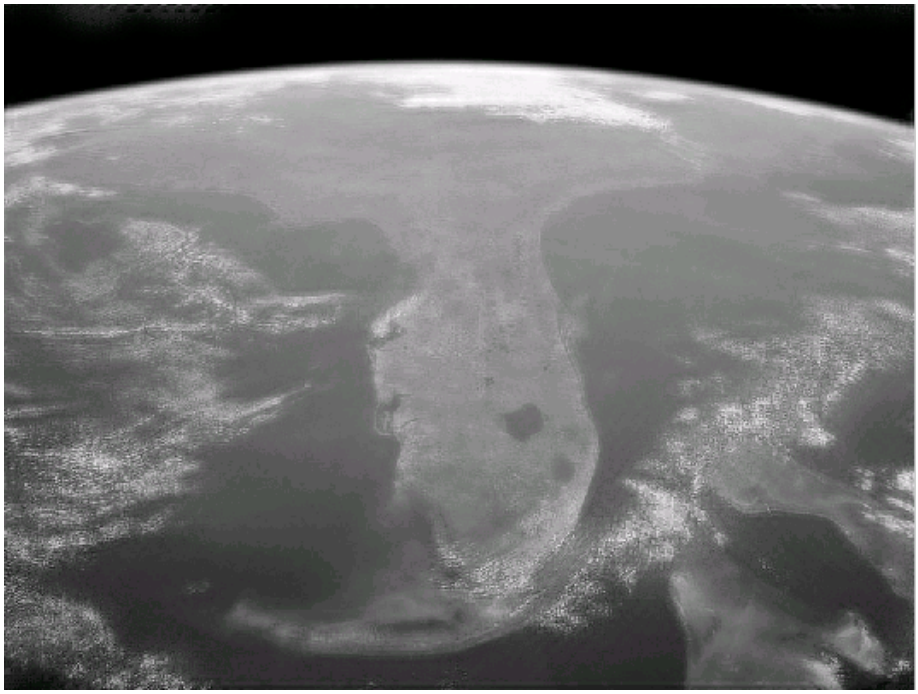
We can see from the Fourier Transform plot that the bright points are the noise (except the biggest one in the center). The noises are symmetric about the x-axis and y axis and also parallel to the two diagonal lines. What I'm trying to do is to eliminate the noises at those positions.

I choose notch filter to remove the noises. The filter is implemented in in frequency domain with high pass Gaussian filters. The center of these filters should be at the noise points. The equation used to design notch filter is as follow:

$$H_{NR} = \prod_{k=1}^{Q} H_k(u,v) H_{-k}(u,v)$$

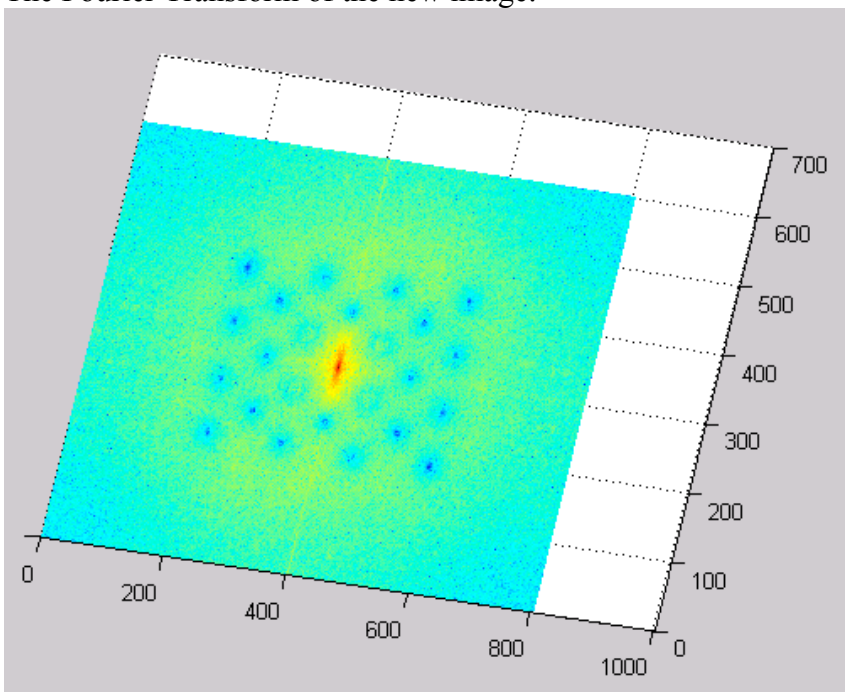


The frequency response of the image goes through the filter I designed, and take the inverse Fourier transform of it. The new image is as follow:



We can see from the image above that the noises are removed.

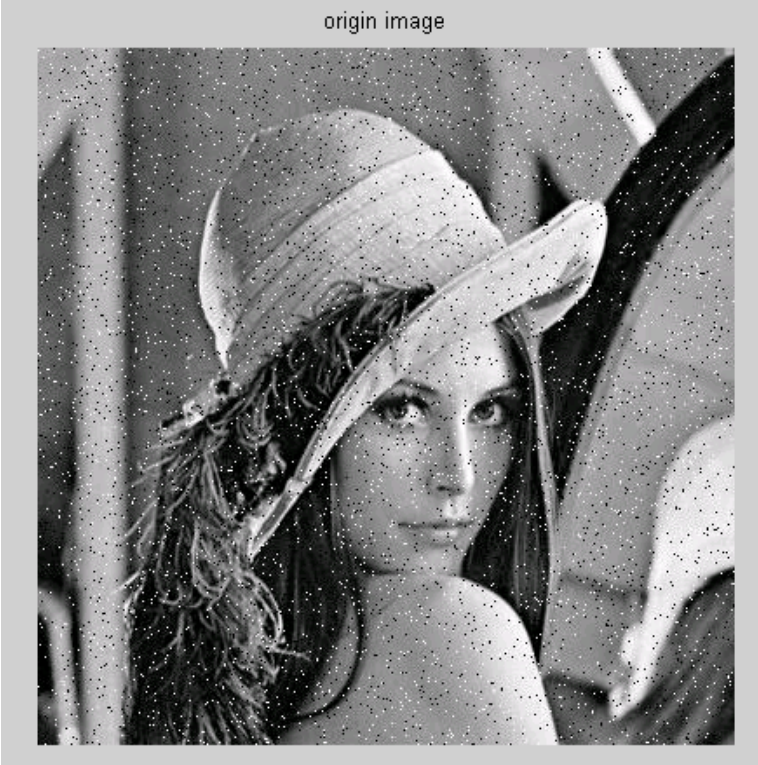
The Fourier Transform of the new image:



We can see from the Fourier Transform of new image that the noises are removed and only leave the center bright point which is the energy of the image. However, except the center four noise points, the others are overreacted and left the pits at the position where it had noises.

4. Design the best filter to remove the shot noise from the image "LenaDeltaNoise.jpg". Select various sizes of the filter and try to find the optimal size. Explain your design and print the results with several such filters.

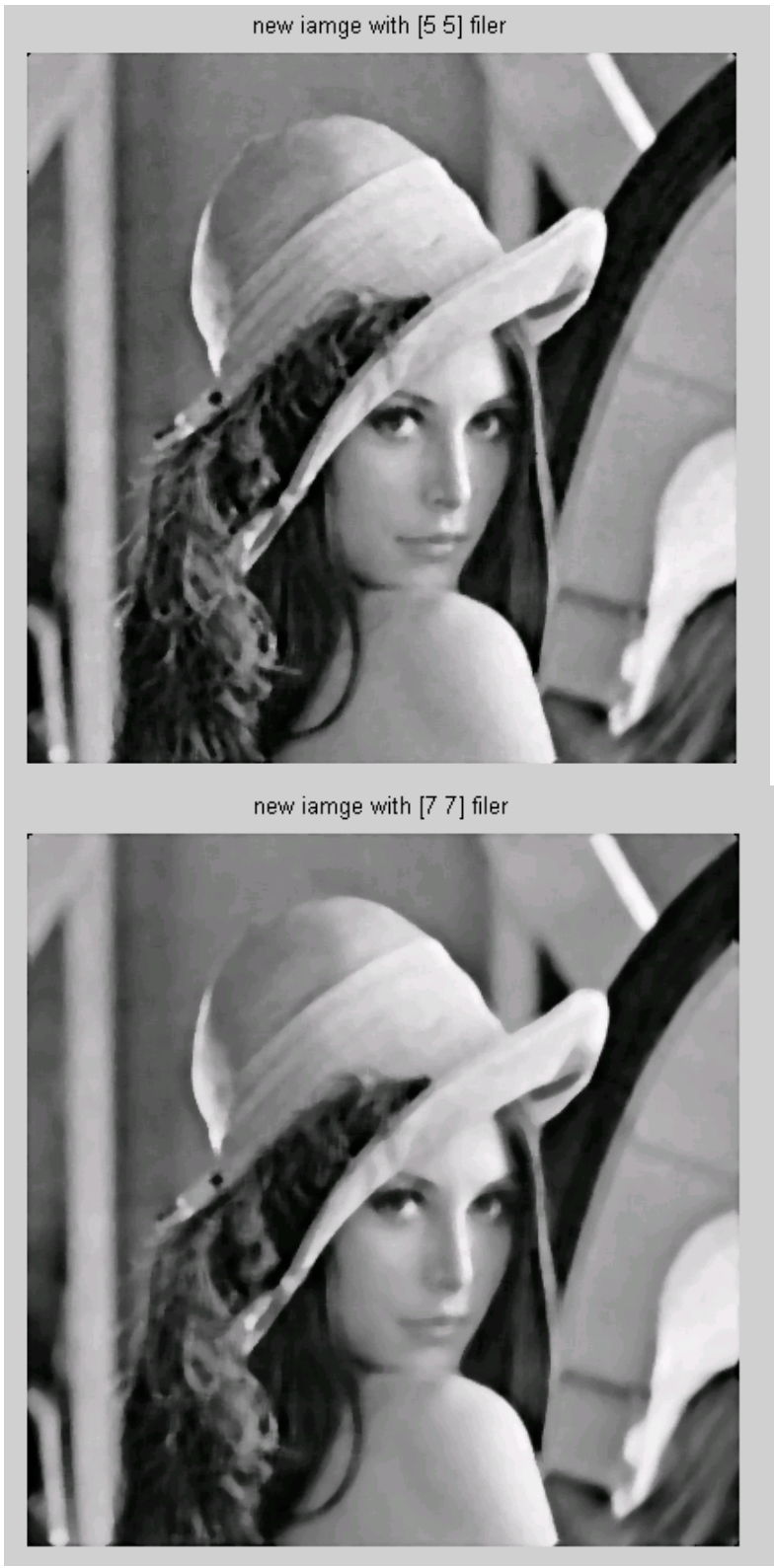
Answer:
First we show the original image. We can see from the image that it has salt and pepper noise which is also called impulse noise. We learnt from the lecture that the best filter used to process this kind of noise is median filter. So the following process are finished by using median filters with different size.



Median filter can be implemented in the following steps:
Step 1. Choose a size of filter. For example, we use 3*3 median filter.
Step 2. For each pixel, process it with the pixels in the neighborhood. Sort the pixels in this area and find the median value.
Step 3. Change this pixel into the median value.
Step 4. Move to next pixel to do the same operation.

new iamge with [2 2] filer

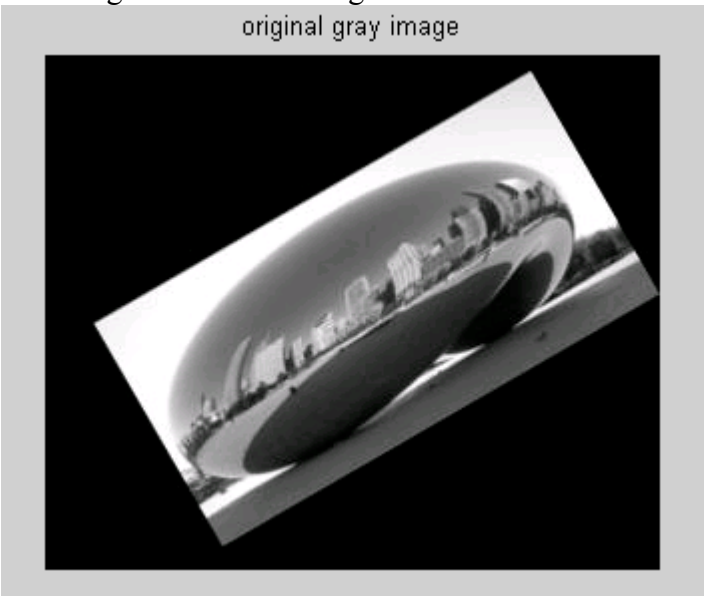
new iamge with [3 3] filer



I tried the median filter with the size of 2*2, 3*3, and 5*5 and 7*7. In my opinion, 3*3 median filter has the best performance. It removes the noise at the same time the edge is clear. However, the 5*5 and 7*7 filters blur the image.

5. The image “PeanutRotated.jpg” is distorted by rotation, translation and scaling. Write a program to restore the given image using bilinear interpolation. Display the restored image and the given image. Detail the design parameters you used for the restoration.

Answer:
The image is a colored image. First I transferred it into gray image with the size 257*307.



We can see from the image that there is another image inside it and rotated and scaled. Our purpose it to rotate back the image and make it into its size it should be (I define it as 250*307).

We use the following equation to find the corresponding coordinates of the pixels in the image:

$$\begin{cases} u = c_1 x + c_2 y + c_3 xy + c_4 \\ v = c_5 x + c_6 y + c_7 xy + c_8 \end{cases}$$

Four corners in the original image are found as (134, 25), (9, 243), (246, 89) and (120, 307). We want to rotate these points into (1,1), (1, 307), (257, 0) and (257, 307). There are 8 unknown variables and 8 equations and the coefficients c1, c2….c8 can be solved.
c1=1.69 c2=0.9687 c3=0 c4=-250.0119
c5=-0.6126 c6=1.0477 c7=0 c8=57.8553.

With the coefficients solved, the coordinates of the pixels are found. But the image has some noise. In the original image, there are as many pixels as the one in the new image. So we have to use some method to fill the black dots in the new image. Here we use bilinear interpolation.

The following are the new image with the size of 250*307: