EE569 Digital Image Processing

HOMEWORK #1

# All problems where padding is needed apply odd mirror padding.

Problem 1:

(a), (b) Image Demosaicing

I. Abstract and Motivation

Because the property of COMS sensors, only a Bayer-Pattern image is obtained when we take a photo, which contain only one-channel pixels. To reconstruct the RGB image from the Bayer-Pattern image, three channels of pixels representing red channel, green channel and blue channel of the color photo must be calculated by combination of pixels in the Bayer-Pattern image.

Bilinear demosaicing is a linear method which suppose that the whole image is linear, which means the pixel somewhere could be calculated by the linear combination of its surround points. On a Bayer-Pattern image, points represent red, green and blue are distributed separately, leaving a number of missing points on each channel distributed regularly. We could reconstruct the color image if we get the pixel of missing points.

Malvar-He-Cutler (MHC) demosaicing is another method to demosaic, it is a refinement of Bilinear demosaicing. In MHC demosaicing, the image is no longer regarded as a flat one, on the contrary, there are concave and convex on image, on the basic of bilinear demosaicing method, the center pixel is revised by surrounded pixels in the same color. This algorithm assumes that the distribution of different channels are relative, so, for example, the green pixels are revised by the red or blue pixels around it. In this way, the image could be sharper than the bilinear approach because of the revise term.

II. Approach and Procedures

In the beginning, we have to padding the image for we want to reconstruct the color in the corners or edges because there lack some pixels around them. The normal way to fill an image is reflection padding. Theoretically, according to the radius of filter (if the largest length of a filter is LMax, the radius of it is (LMax – 1) / 2), the image should be enlarged by four edges, the value of the additional position is assigned according to the symmetric position reflected by edges. In my approach, the image is not enlarged actually, the function be_m(index, boundary) is used to get the value of pixels by whether the position is out of range. If a point is in the image, be_m will return the value right there, otherwise, it will return the value reflected by the relative edge, which could achieve the same outcome as enlarging the image directly. Then, with the bilinear formula mentioned in class, calculate the missing value of each channel by its surrounding point, get the three channels of the RGB image separately and arrange them in right order, the color image could be reconstructed. As for the MHC method, just code in the same way as bilinear approach and add some revised term shown in lecture.

To compare the two algorithm in more detail, the PSNR of the outcome of the algorithms are calculated and compared.

III. Experimental Results

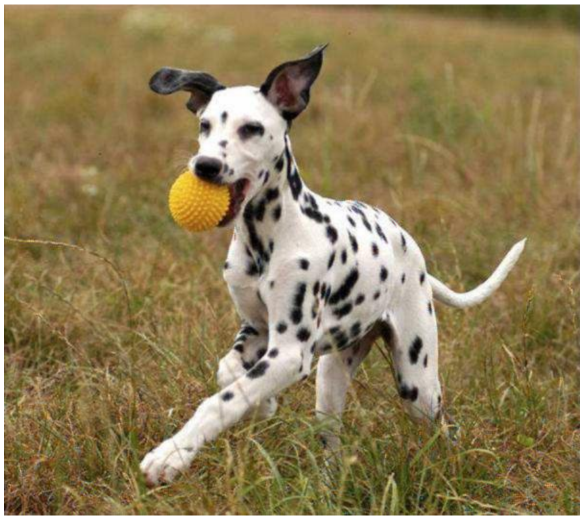


Fig 1.1 RGB image of dog reconstructed by bilinear demosaicing

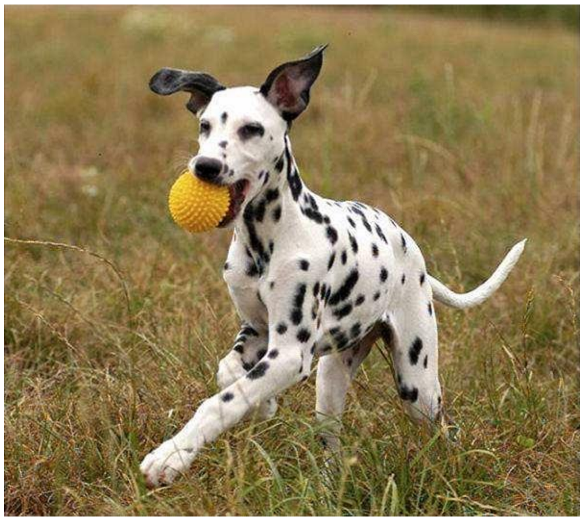


Fig 1.2 RGB image of dog reconstructed by MHC demosaicing

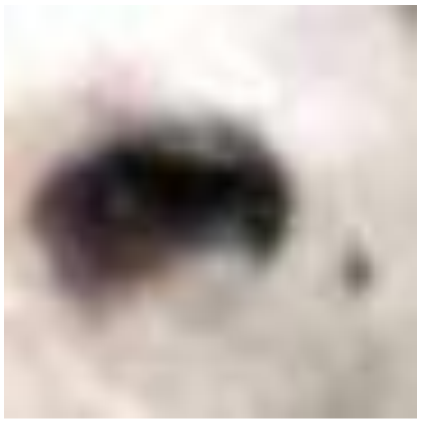


Fig 1.3 A finer look at Fig 1.1

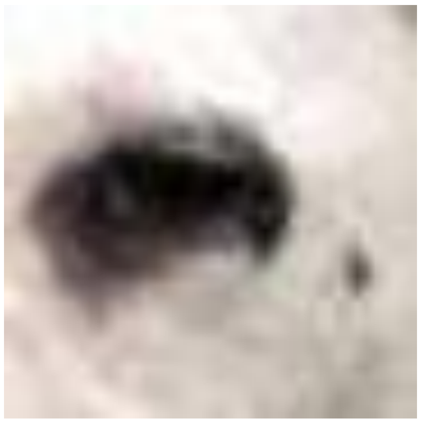


Fig 1.4 A finer look at Fig 1.2

Figure	Red	Green	Blue	Average
Bilinear	30.0929	33.8309	30.2267	31.3835
MHC	36.2109	39.9823	35.6247	37.2726

Table 1.1 PSNR (dB) of Bilinear and MHC approach by different channel

IV. Discussion

(a)-(2): Figure 1.1 shows the result obtained by bilinear demosaicing. In general, the reconstructed image could represent the feature of original image. However, there are a lot of blurry edges where the color changed abruptly. The reason is that images are not flat distributed in the most of time, there are a lot of convex and concave between pixels, especially where the pixels change abruptly, so, the revise term in MHC approach could make up this difference somehow, leading to a better outcome.

(b)-(2): In figure 1.2, the effect of MHC demosaicing on these edges is much sharper than that of bilinear approach, the detail of MHC approach is more similar to the original image. According to table 1.1, MHC approach has higher dB in every single channel and thus a higher dB in average than bilinear approach, which means the image reconstructed by MHC approach is more similar to the original image than bilinear approach.

(c) Histogram Manipulation

I. Abstract and Motivation

There are some images with little contrast between bright color and dark color, which means objects on images could not be recognized clearly. The reason of this phenomenon is that the distribution of most of pixels in these images concentrate in a narrow range. For example, the range of pixels is between 0 and 255, if 70% of these pixels distributed between 0 and 50, leaving the other pixels lies in the remaining part, then the image is going to be dark, just like the picture Toy.raw. If the pixels could be projected by some transformation, so that the distribution of pixels is more like a uniform distribution, then the contrast of the image could be improved and the image could be clearer. There are two ways to achieve this, one is transfer-function-based method, and the other one is cumulative-probability-based method. In the first method, a transfer function is calculated by the sum of the former pixel distribution of histogram, then the histogram is projected by the transfer function, obtaining a distribution more uniformly. In the second approach, all pixels are arranged in a 1D array according to the value of pixels. Then, according to the total number of pixels in the image, these pixels are divided by 256, and the pixel array is grouped sequentially with same number of pixels in these 256 groups. To get a finer image, the pixels with the same value should be re-arranged randomly, so the pixels being changed could distributed throughout the whole image with the same probability.

II. Approach and Procedures

The first step is to reconstruct the image form the raw file, and separate the three channels. For every channel, because the range of pixels is between 0 and 255, so allocate an array with length 256, then traverse the whole channel to count the number of each value of pixels, then the histogram is obtained. Then, in the transfer-function-based method, there is another array with length 256 is created, which stores the pre-item sum of histogram, that is, the i^{th} item of the sum array is the summation from 0 to i in the pixel array. So the sum array stands for the cumulative property of pixels, then the sum array is normalized, so that the first item in the sum array is equal or larger than 0, the last item is equal to 256, then the transfer function is obtained. The target image could be calculated by mapping the pixel in the original image by the transfer function. In the cumulative-probability-based method, a structure stores the coordinate of each point is created, then the coordinates are arranged according to the value of pixels. Each group of points with the same pixel is re-arranged by a random function, which could make points with the same pixel distribute more uniformly. Then, the random-arranged array is then divided into 256 groups with the same length, pixels in the same group is reassigned with the same value according to the sequence of appearance. Then image is reconstructed according to the new pixel value and coordinates stored in the structure.

III. Experimental Results

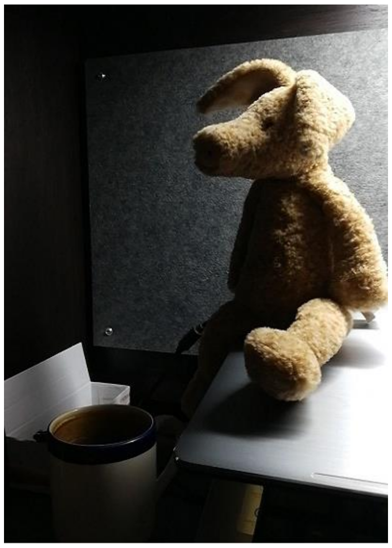
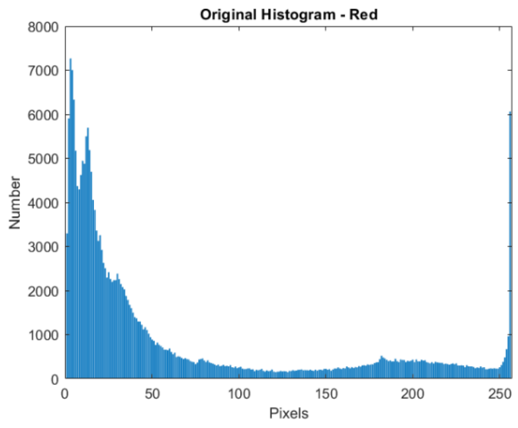
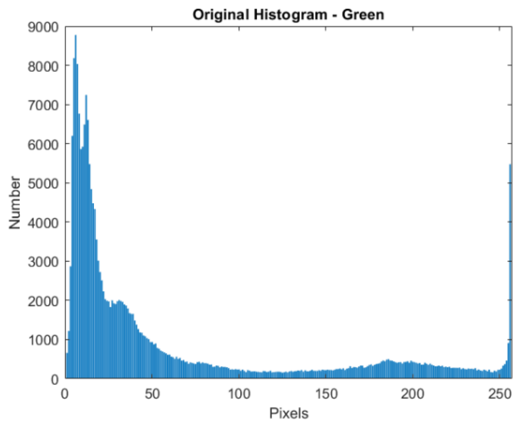


Fig 1.5 Original Image





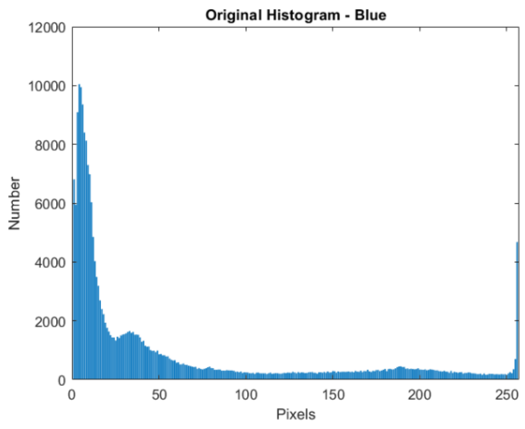


Fig 1.6 Histogram of Each Channel of Original Image

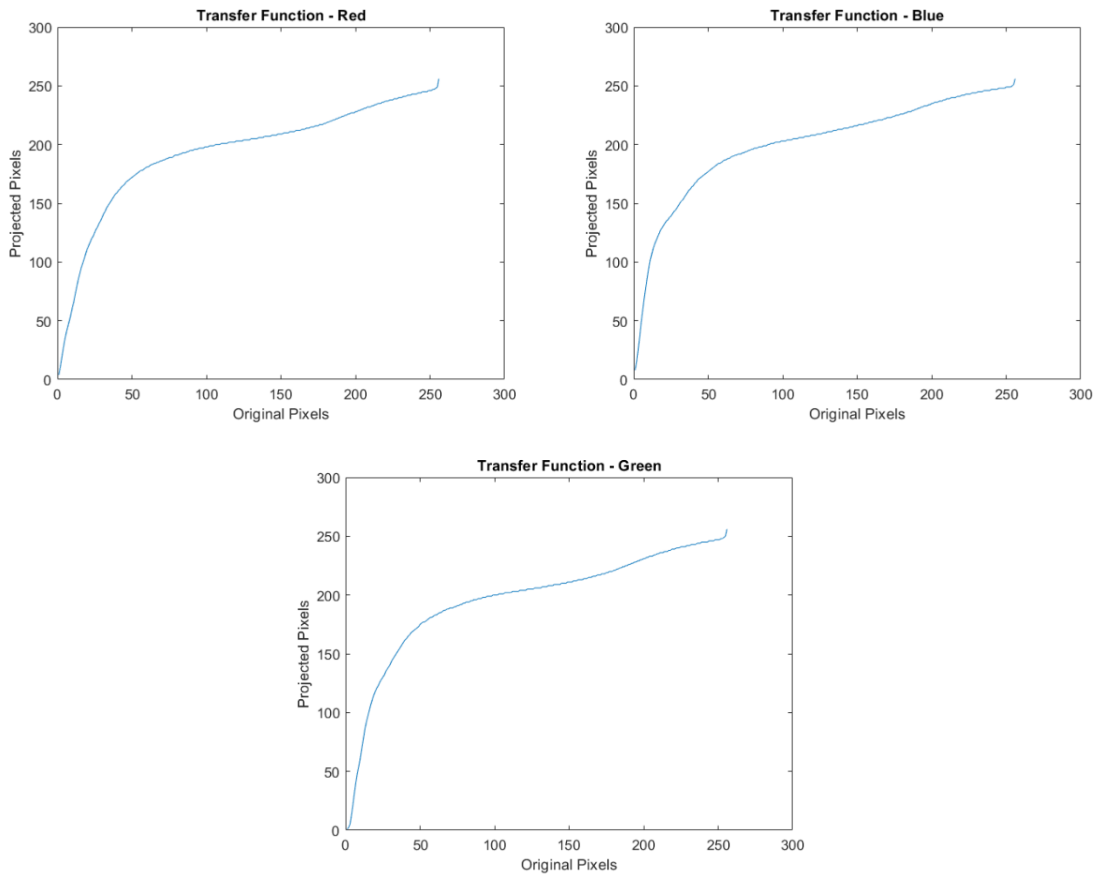


Fig 1.7 Transfer Function of Each Channel

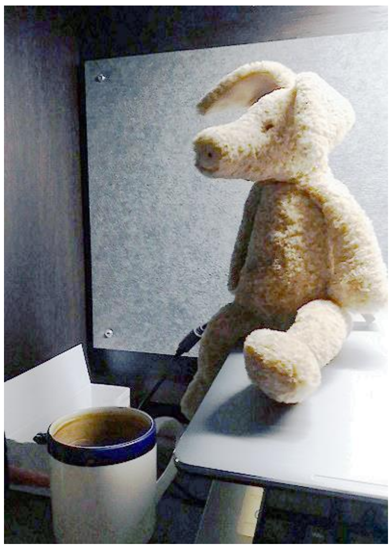


Fig 1.8 Refined Image by Method 1

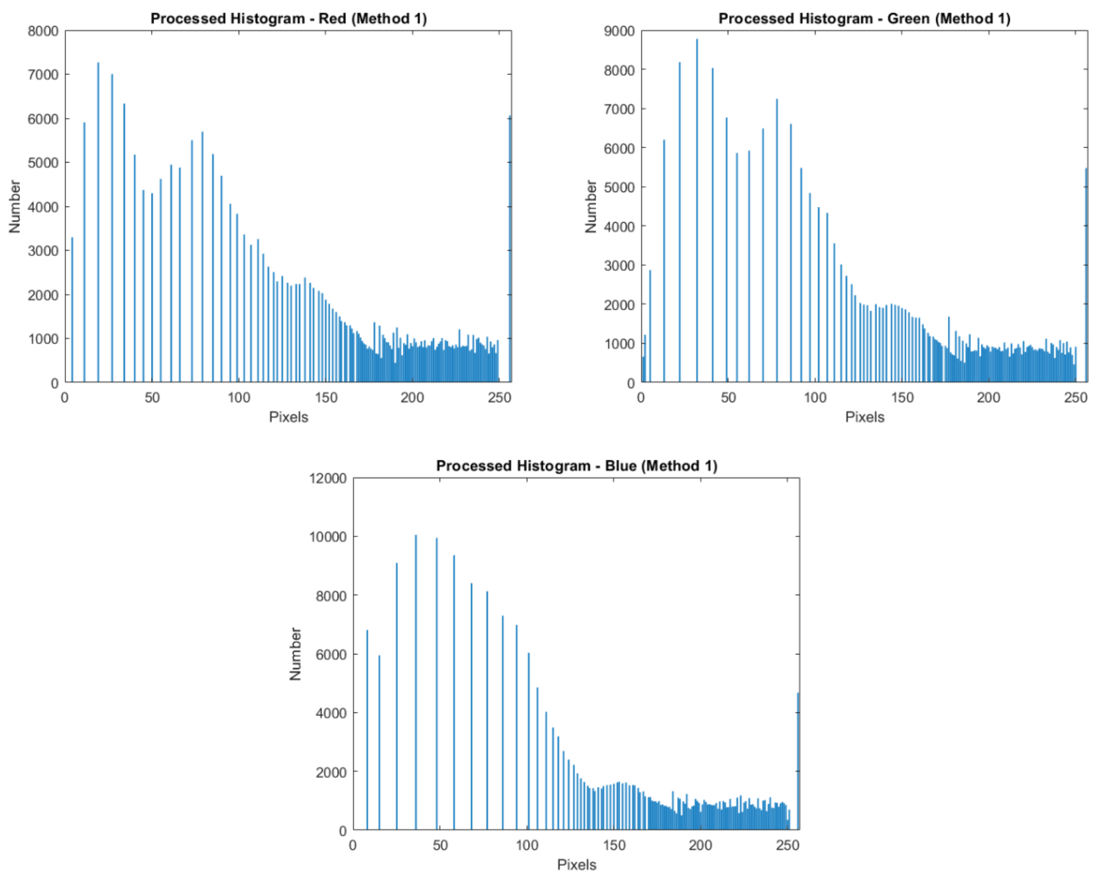


Fig 1.9 Histogram of Each Channel of Refined Image by Method 1

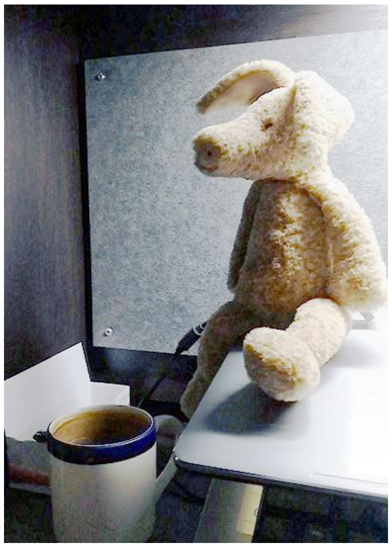


Fig 1.10 Refined Image by Method 2

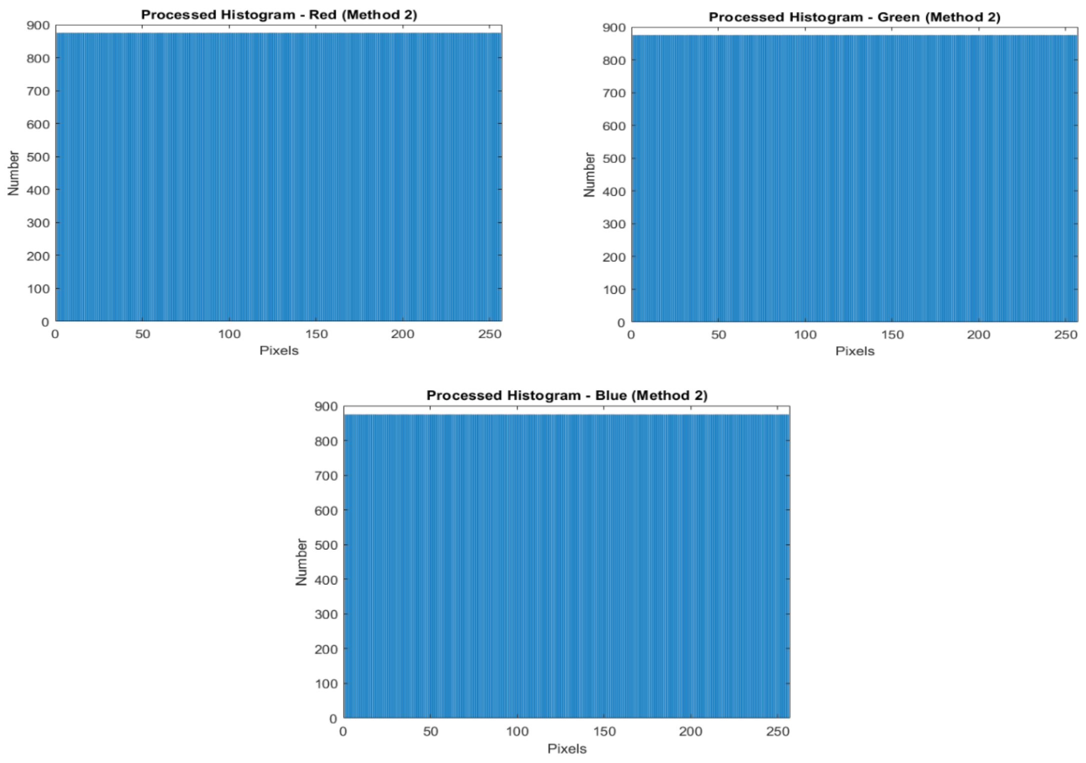


Fig 1.11 Histogram of Each Channel of Refined Image by Method 2

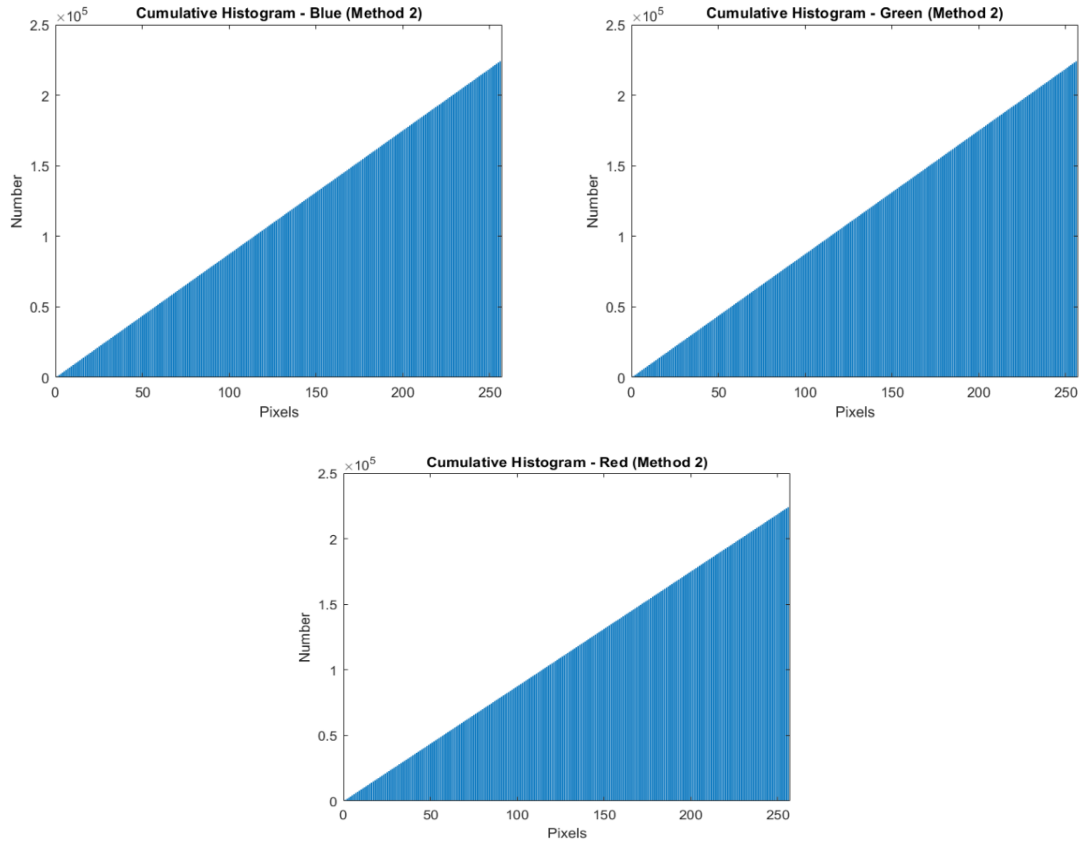


Fig 1.12 Cumulative Histogram of Each Channel of Refined Image by Method 2

IV. Discussion

(c)-(4): From figure 1.8 and 1.10 (the two reconstructed image), the two images are almost the same, both of them could enhance the original image effectively, but through the pixel distribution histogram, the two images have different pixel values. The image processed by method 1 have a less uniformly distributed than method 2, but method 1 could project every single value of pixels to the same value, which could make the image flatter and the relationship between points are preserved better, but as the histograms show, there are more gaps between each pixels than the original image, which may add some contours throughout the whole image, especially where the pixels change abruptly. Method 2 could lead to an absolutely uniformly distribution, but because the points to be changed are selected randomly, so there may be some slightly difference among experiments. This method is a convenience way to enhance image, but it changed the relative relationship between points, the structure of the image.

Both of the method mentioned in lecture do not take noise into account, and they will enhance noise equally. An idea is that we could set a threshold, cut off the points with pixel exceeding the threshold, and then calculate the area being cut in histogram, then the cut area could be equally reassigned to the whole range of pixels. This way, the noise could not be enhanced and the structure of image is preserved better.

Problem 2:

Abstract and Motivation

Noise always exists in images, the most common one is AWGN, with zero means and a certain standard deviation. For this kind of noise, there are a lot of ways to denoise and get a better quality of images. The most basic filter just takes pixels around the target point into account, weighted by the locations of pixels around it or using uniform weight. Such kind of filter could smooth image effectively, but it could also make the image blurry, losing edge information severely. An improvement of this filter is bilateral filter, this filter not only take the relative location around the target point into account, but also take the similarity of the value of these pixels into consideration, if the pixels around the target point are closer to the target pixels, the weight is heavier. In this way, the noise could be inhibited more effectively, however, the edge information is also harmed. Both of the filters mentioned above do not take global information into consideration. So Non-Local

Mean (NLM) filter is proposed. This kind filter uses three windows to calculate the value of pixels, a large one and two small one with the same size. The large window could give a range on the image for a certain point to search for, and one small window is centered at the target point, another small window traverses the large window, the Gaussian distance between the two window is calculated for each point in the large window. All of the value is weighted and the target pixel is calculated. Another way to solve the local problem is BM3D Algorithm, which is an advanced denoising algorithm, projecting the 2D image into 3D space and process, then the 3D data is projected back to 2D space. BM3D algorithm could get a better result than all the algorithm above, but the computational complexity is relatively high. Another kind of noise is impulse noise, this kind of noise is caused by error in DAC. This kind of noise could be denoised by outlier detection or medium denoising. If an image is mixed with impulse noise and Gaussian noise, the impulse noise should be gotten rid of in the beginning, for the existence of impulse noise will have influence on the statistic distribution of Gaussian noise.

(a) Basic denoising methods

I. Approach and Procedures

Because denoising problem also need to get pixels outside image, the first thing to do is to padding image. The method here is the same as the one in Problem 1 (a)(b), that is, the image is not actually enlarged, instead, a function is used to proceed the index out of boundary, and get the same result as reflection padding. Then, in this section, two filters are used, one is uniformly distributed, the other is Gaussian distributed, the Gaussian weight is calculated using the relative position of pixels to the center of filter. Then, denoised image is calculated by 2D convolution, different sizes of filter and Gaussian standard deviations are tried. Because the side length of filter must be odd, the radius of filter is defined as follows:

$$r = \frac{(s-1)}{2}$$

Where s stands for side length of filter, and r stands for radius of filter. The center of filter is obviously (r + 1).

Noise could be obtained by subtract original image from noised image.

II. Experimental Results

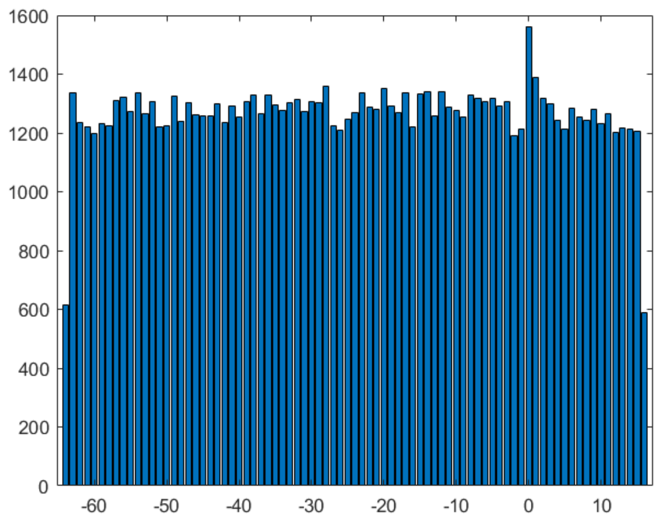


Fig 2.1 Distribution of Noise

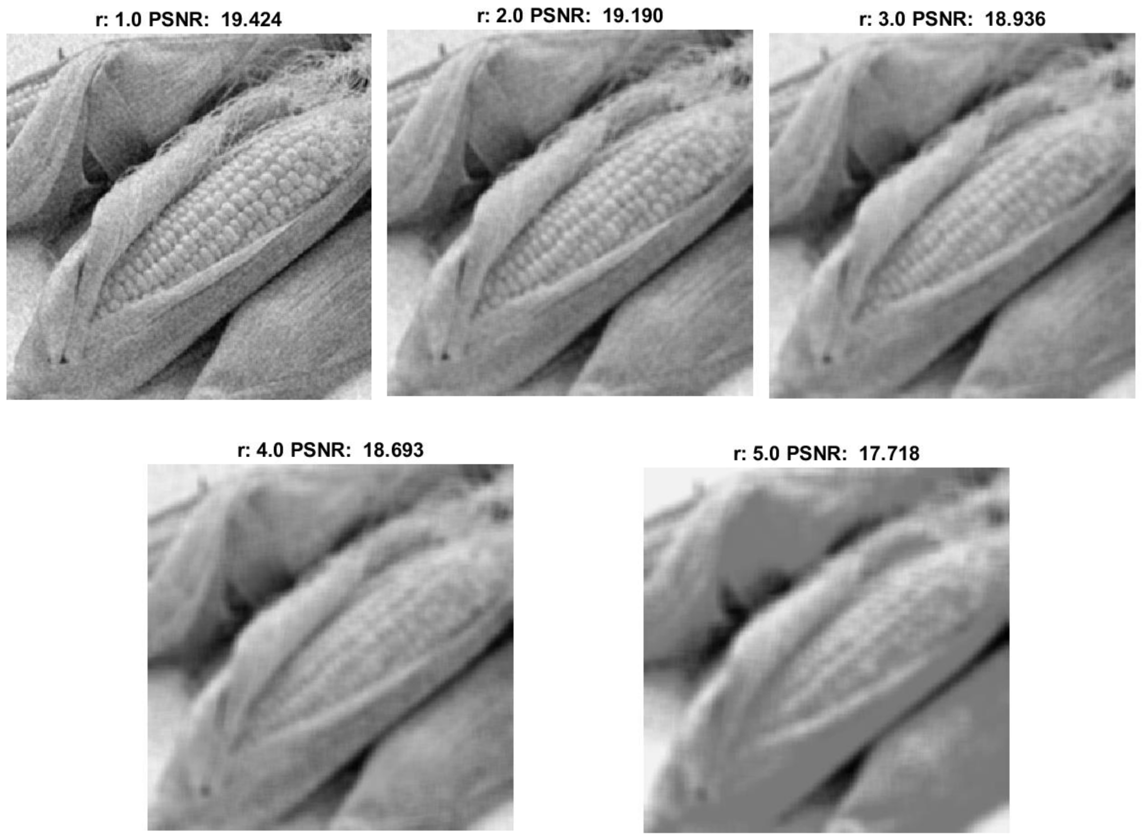


Fig 2.2 Images Processed by Basic Denoising Methods (Uniform Weight)

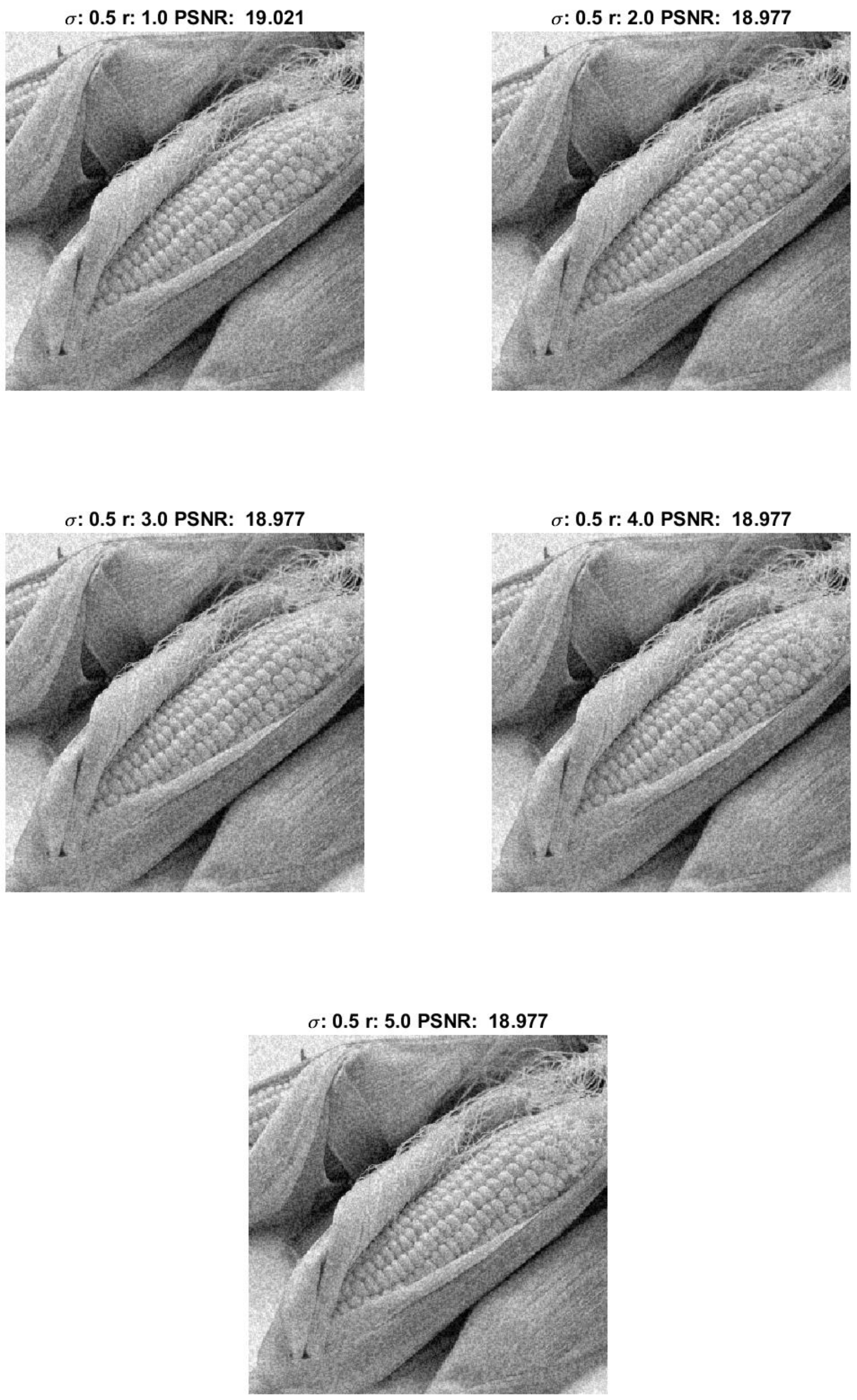

σ: 0.5 r: 1.0 PSNR: 19.021
σ: 0.5 r: 2.0 PSNR: 18.977
σ: 0.5 r: 3.0 PSNR: 18.977
σ: 0.5 r: 4.0 PSNR: 18.977
σ: 0.5 r: 5.0 PSNR: 18.977

Fig 2.3 Images Processed by Basic Denoising Methods (Gaussian Weight, σ= 0.5)

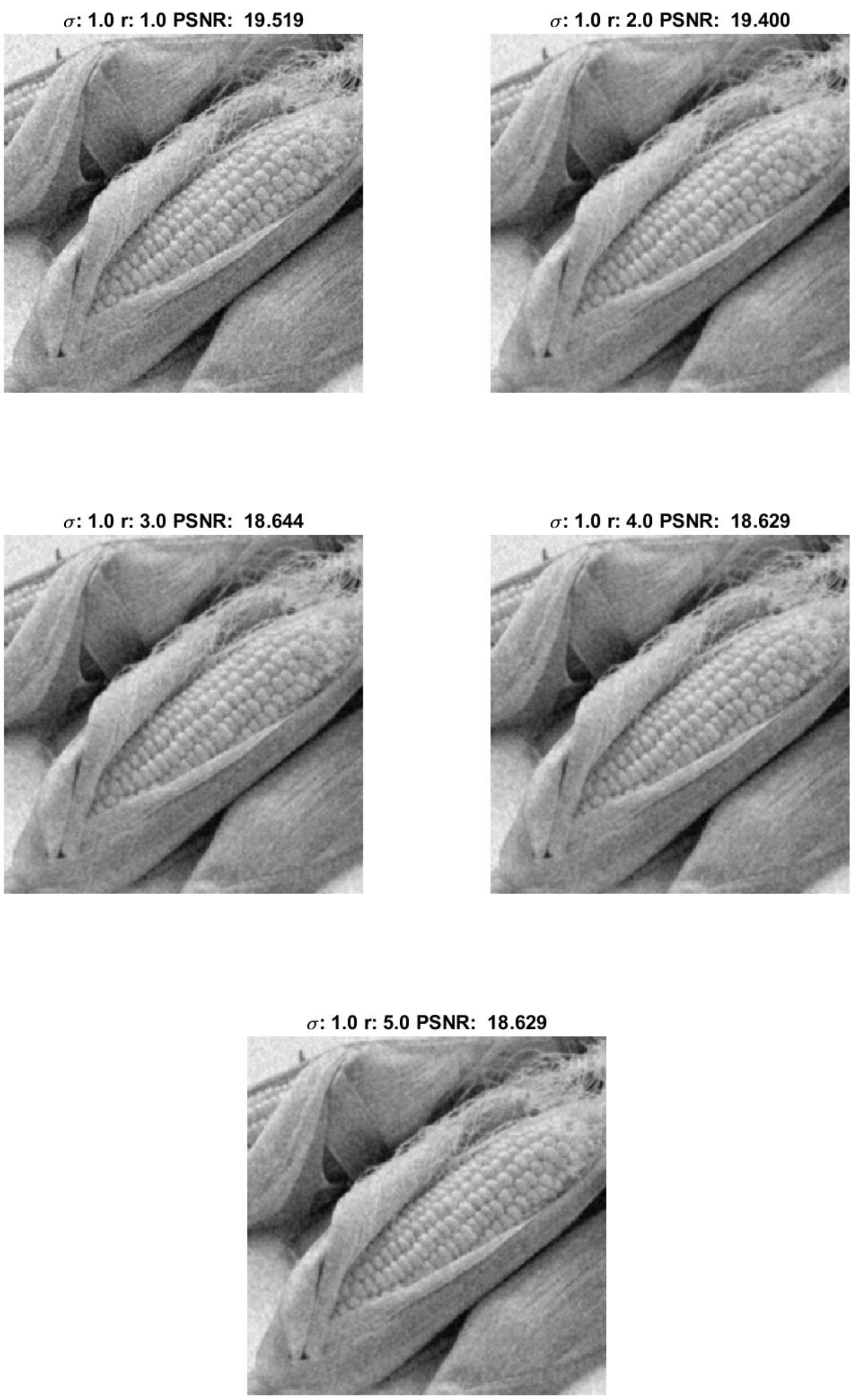

σ: 1.0 r: 1.0 PSNR: 19.519
σ: 1.0 r: 2.0 PSNR: 19.400
σ: 1.0 r: 3.0 PSNR: 18.644
σ: 1.0 r: 4.0 PSNR: 18.629
σ: 1.0 r: 5.0 PSNR: 18.629

Fig 2.4 Images Processed by Basic Denoising Methods (Gaussian Weight, σ= 1.0)

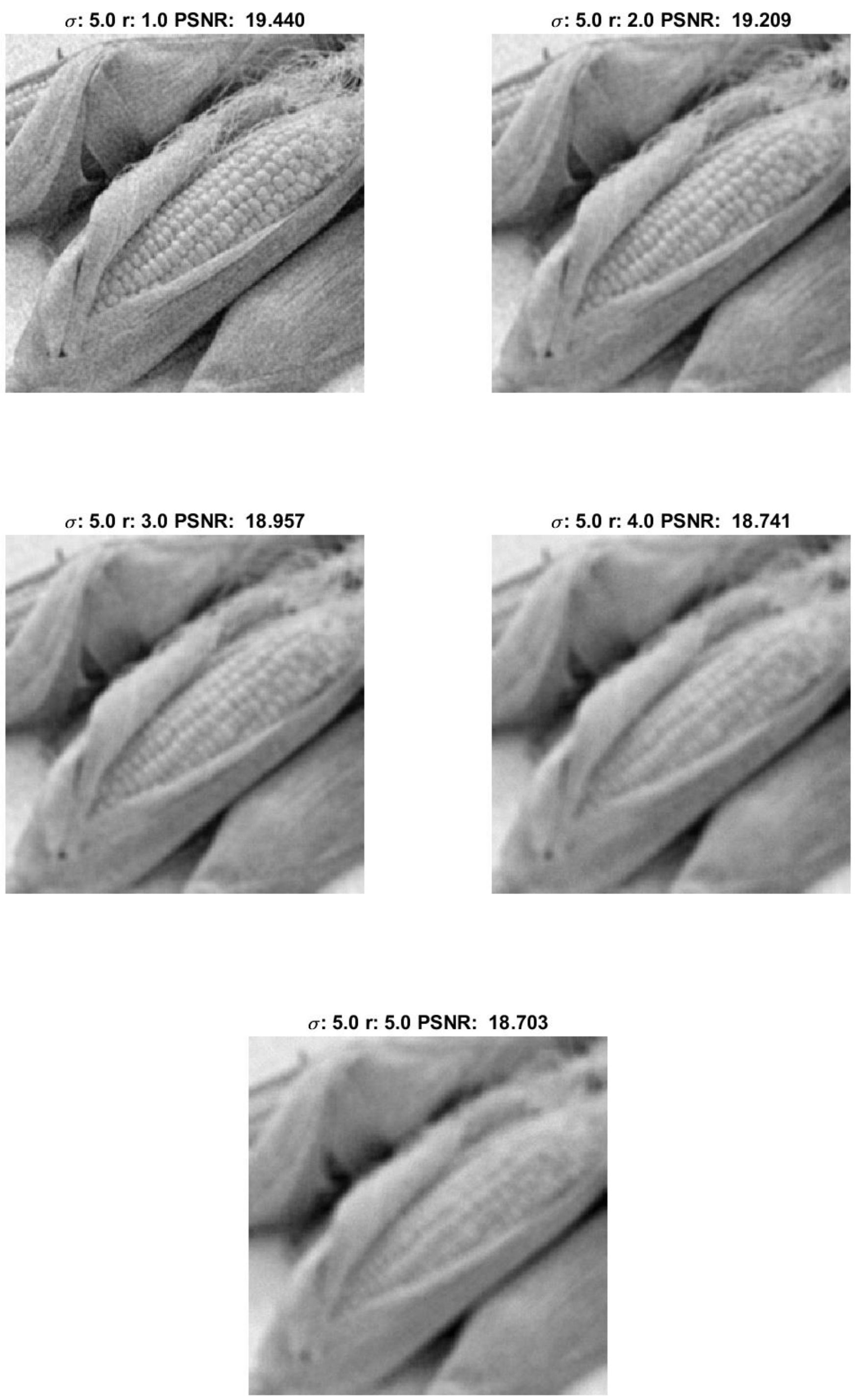

σ: 5.0 r: 1.0 PSNR: 19.440
σ: 5.0 r: 2.0 PSNR: 19.209
σ: 5.0 r: 3.0 PSNR: 18.957
σ: 5.0 r: 4.0 PSNR: 18.741
σ: 5.0 r: 5.0 PSNR: 18.703

Fig 2.5 Images Processed by Basic Denoising Methods (Gaussian Weight, σ= 5.0)

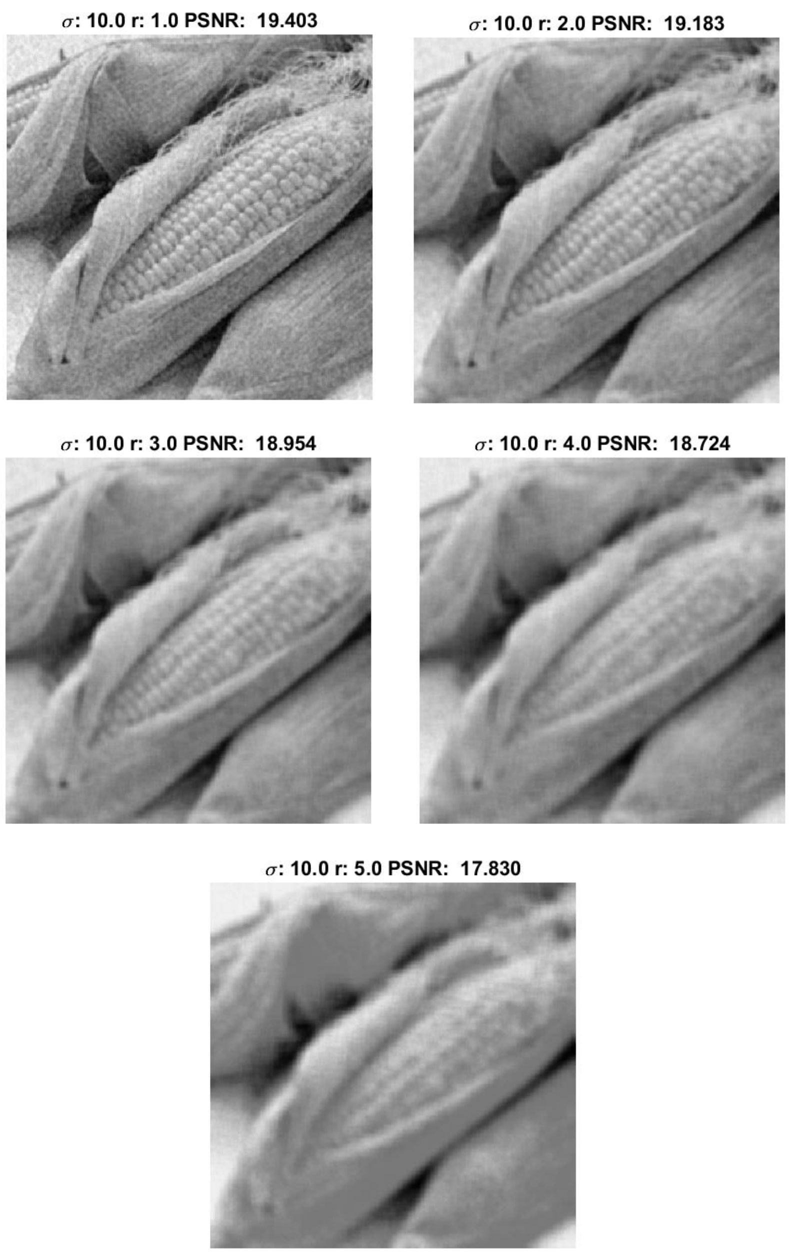

σ: 10.0 r: 1.0 PSNR: 19.403
σ: 10.0 r: 2.0 PSNR: 19.183
σ: 10.0 r: 3.0 PSNR: 18.954
σ: 10.0 r: 4.0 PSNR: 18.724
σ: 10.0 r: 5.0 PSNR: 17.830

Fig 2.6 Images Processed by Basic Denoising Methods (Gaussian Weight, σ= 10.0)

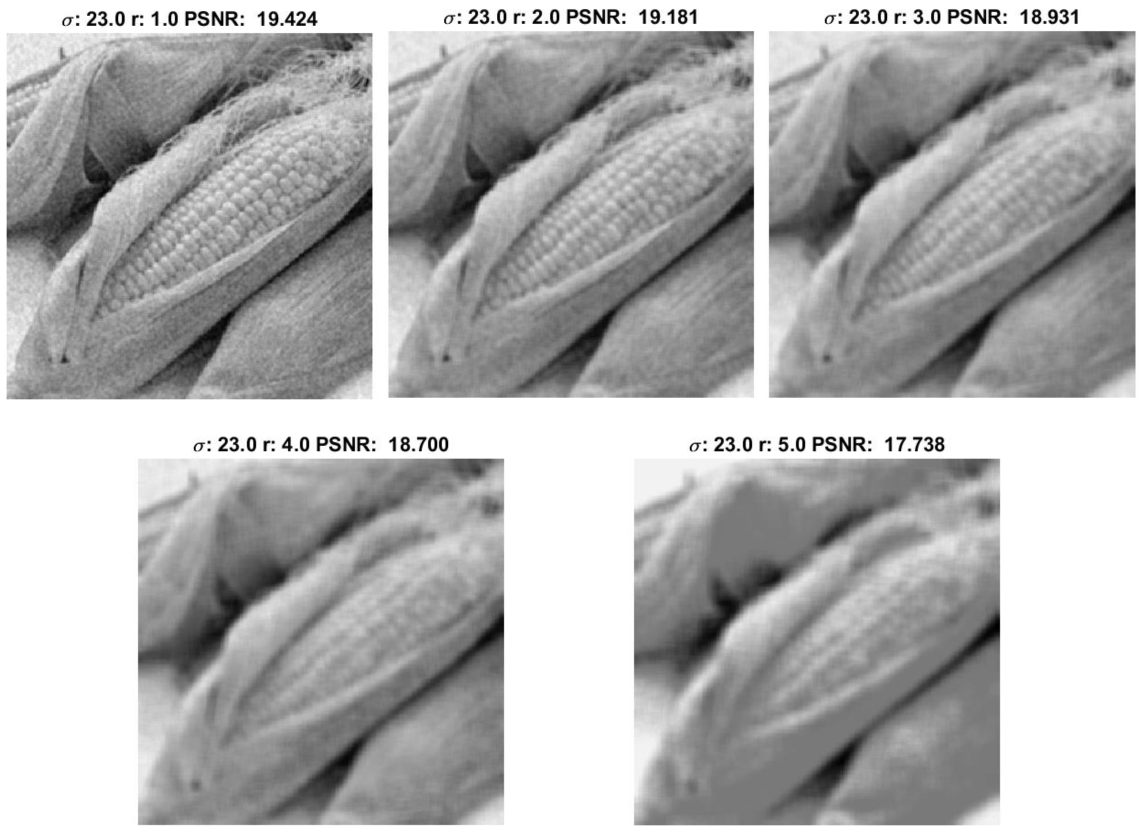

σ: 23.0 r: 1.0 PSNR: 19.424
σ: 23.0 r: 2.0 PSNR: 19.181
σ: 23.0 r: 3.0 PSNR: 18.931
σ: 23.0 r: 4.0 PSNR: 18.700
σ: 23.0 r: 5.0 PSNR: 17.738

Fig 2.7 Images Processed by Basic Denoising Methods (Gaussian Weight, σ= 23.0)

	sigma	Size of window				
		3	5	7	9	11
Uniform Weight		**19.424**	19.190	18.936	19.693	17.718
	0.5	19.021	18.977	18.977	18.977	18.977
	1	**19.519**	19.400	18.644	18.629	18.629
Gaussian Weight	5	19.440	19.209	18.957	18.741	18.703
	10	19.403	19.183	18.954	18.724	17.830
	23	19.424	19.181	18.931	18.700	17.738

Table 2.1 PSNR (dB) of denoised image processed by basic filter

III. Discussion

(1) The noise added into Figure 7(b) is a uniformly distributed noise with standard deviation 22.948 and average 24.

(2) The experimental results show that the general effect of basic method is not very well. when r = 1, the smallest 3×3 filter is applied, so the PSNR is decreased as the size of filter increases. When standard deviation of Gaussian filter is 1, the PSNR could get the maximum in these experiments with r = 1. So, adjust standard deviation of Gaussian filter could give a better result.

(b) Bilateral Filtering

I. Approach and Procedures

The function is designed almost the same with (a), only the weight used in filter is redesigned, adding the pixel revise item covered in lecture. In the experimental result part, the size of window is fixed to 5 to show the impact of two different standard deviation.

II. Experimental Results

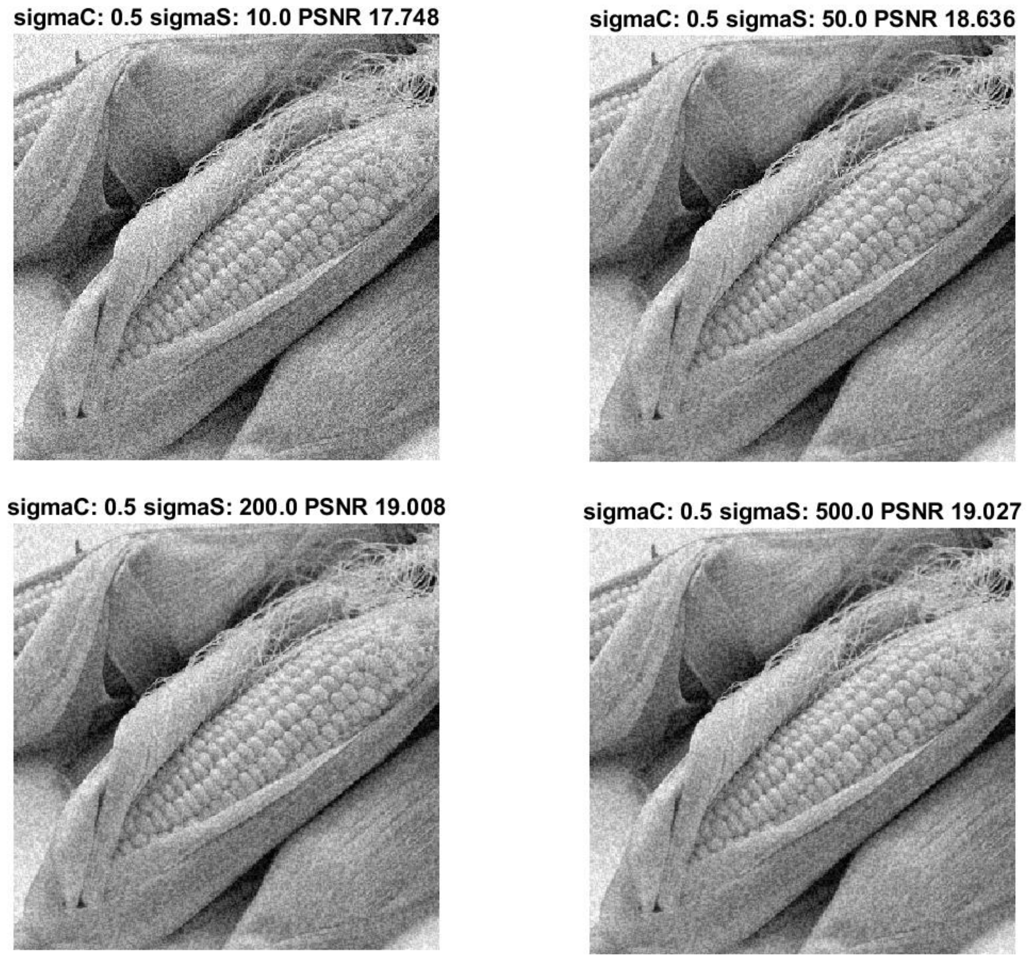

sigmaC: 0.5 sigmaS: 10.0 PSNR 17.748
sigmaC: 0.5 sigmaS: 50.0 PSNR 18.636
sigmaC: 0.5 sigmaS: 200.0 PSNR 19.008
sigmaC: 0.5 sigmaS: 500.0 PSNR 19.027

Fig 2.8 Images Processed by Bilateral Filter (σ_c= 0.5)

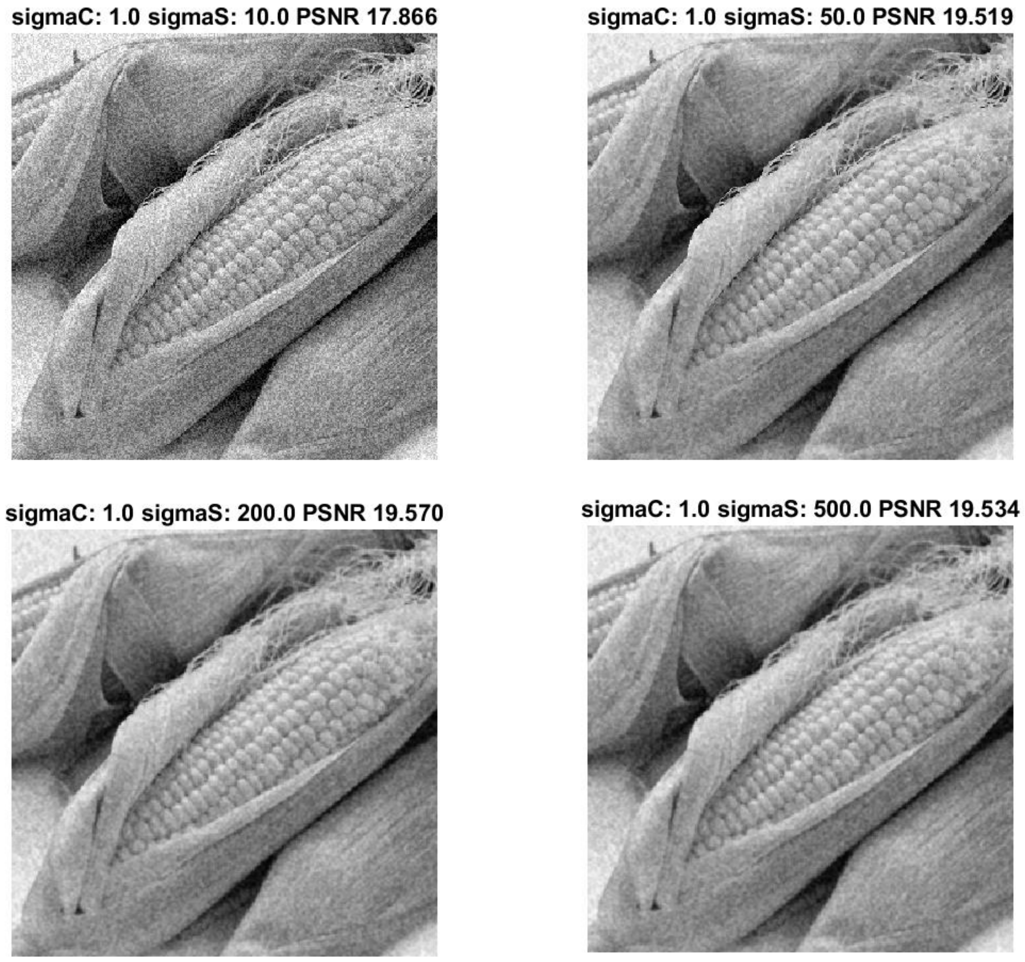

sigmaC: 1.0 sigmaS: 10.0 PSNR 17.866
sigmaC: 1.0 sigmaS: 50.0 PSNR 19.519
sigmaC: 1.0 sigmaS: 200.0 PSNR 19.570
sigmaC: 1.0 sigmaS: 500.0 PSNR 19.534

Fig 2.9 Images Processed by Bilateral Filter (σ_c= 1.0)

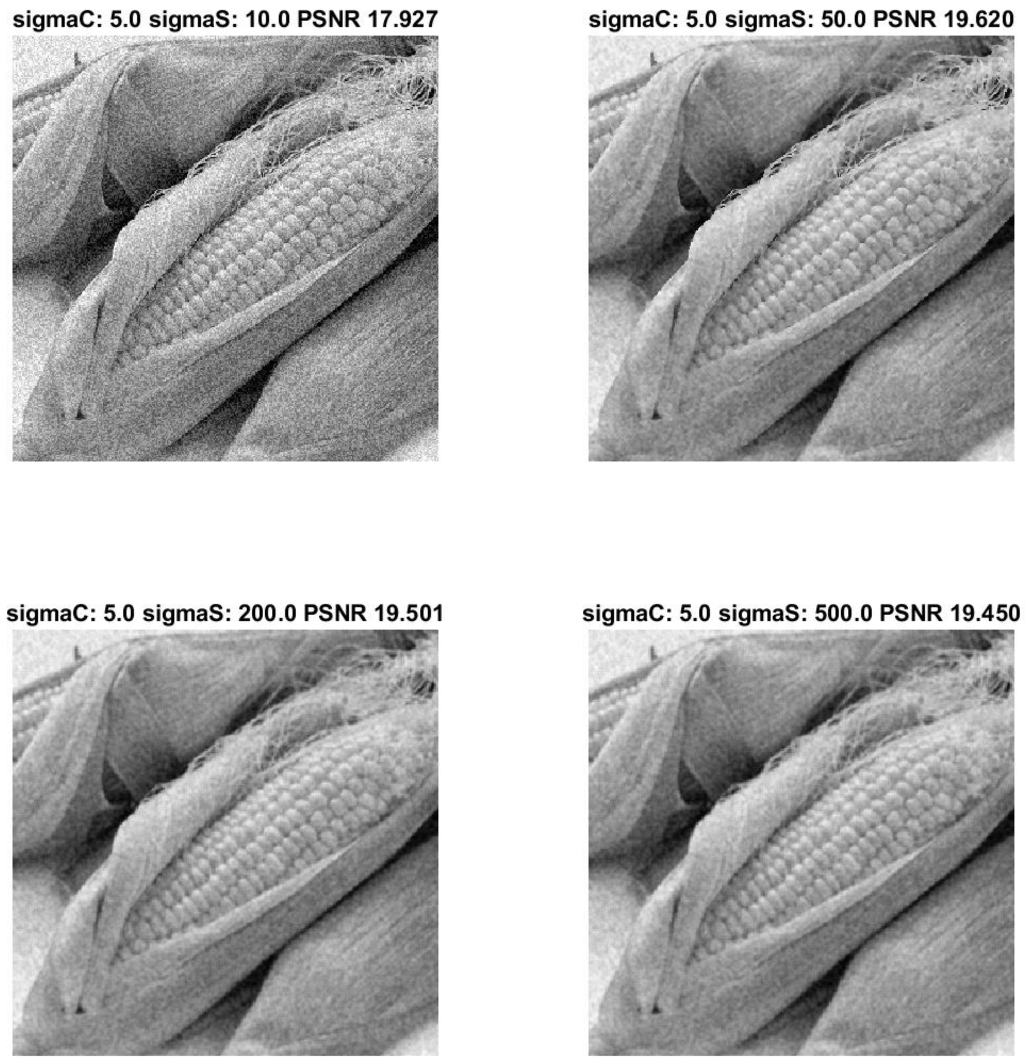

sigmaC: 5.0 sigmaS: 10.0 PSNR 17.927
sigmaC: 5.0 sigmaS: 50.0 PSNR 19.620
sigmaC: 5.0 sigmaS: 200.0 PSNR 19.501
sigmaC: 5.0 sigmaS: 500.0 PSNR 19.450

Fig 2.10 Images Processed by Bilateral Filter (σ_c= 5.0)

Sigma_C	Sigma_S			
	10	50	200	500
0.5	17.748	18.636	19.008	**19.027**
1.0	17.866	19.519	**19.570**	19.534
5.0	17.927	**19.620**	19.501	19.450

Table 2.2 PSNR (dB) of denoised image processed by Bilateral Filter

III. Discussion

(2) As the experimental result shows, when the value of σ_s is fixed, the PSNR becomes larger, and then smaller as σ_c increases. Also, when the value of σ_c is fixed, the PSNR becomes larger, and then smaller as σ_s increases, which means there is a most suitable value for both of the two parameters to get a best result when the size of filter is fixed.

(3) Bilateral Filter does perform better than basic filter, from the experimental result, bilateral filter could obtain a higher PSNR than basic filter for bilateral filter takes value of pixel into account, which could preserve the edge information better. Bilateral filter could be regarded as a revised version of Gaussian basic filter adding pixel value term.

(c) Non-Local Means (NLM) Filtering

I. Approach and Procedures

There are three windows are applied in this algorithm, it is more complicated than the filter above. So, in this part, the image is enlarged by reflection padding, Then, every point in the image is traversed. For every point, a large window and a small window are specified, both of them are centered at this point, then, for every point in the large window, a small window is used to traverse them and the weight of the window is calculated, besides the points are too close to edges of large window that the small window cannot find enough point to be accommodated. The small window is weighted by Gaussian weighted Euclidian distance, the large window is uniformly weighted. The difference between small windows is calculated, and the target pixel could be obtained by the formula mentioned in lecture. For the effect of Gaussian standard deviation has been discussed in (a) and (b), so in this part, only size of small window, size of large window and parameter h will be discussed, and Gaussian weight parameter is fixed to 30 to get a relatively finer result. LW and SW stand for side length of large window and small window respectively.

II. Experimental Results

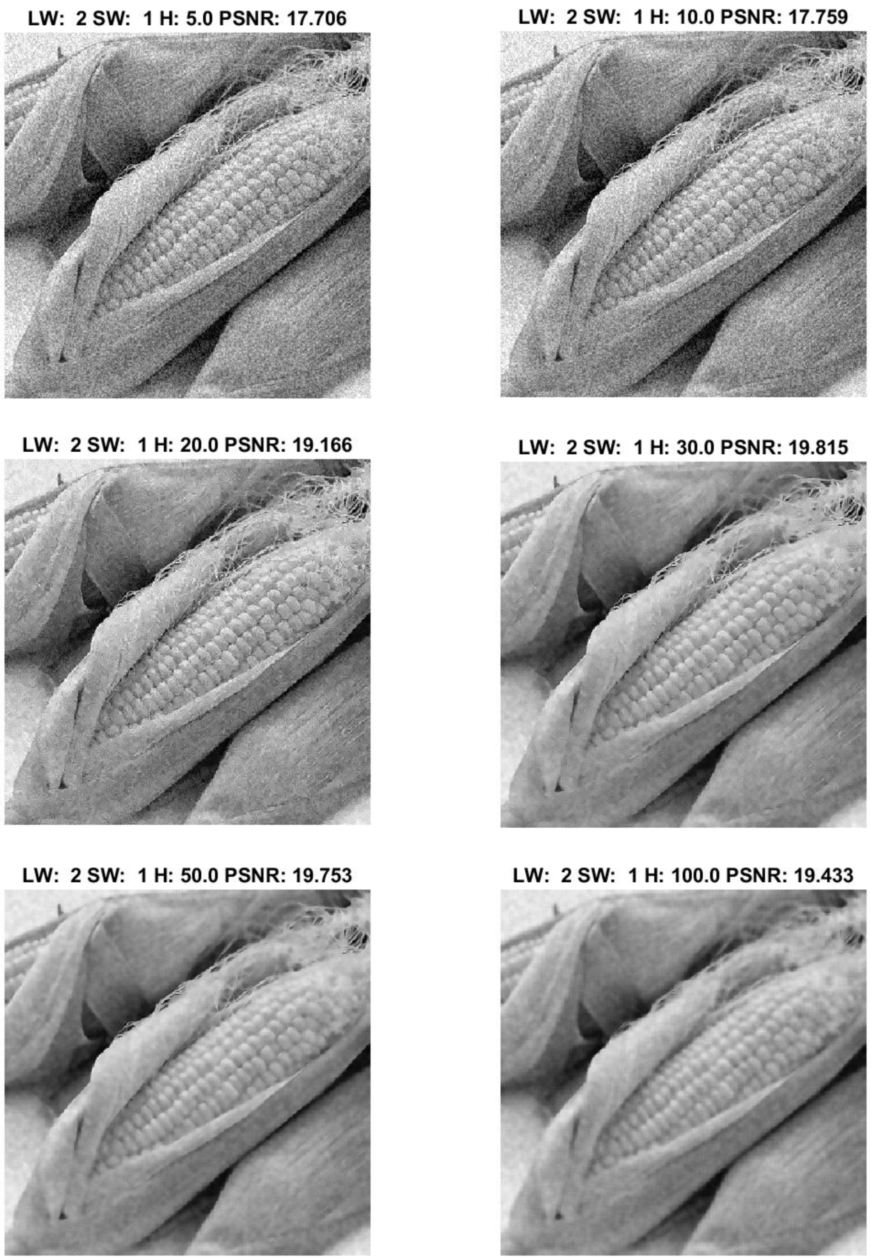

LW: 2 SW: 1 H: 5.0 PSNR: 17.706
LW: 2 SW: 1 H: 10.0 PSNR: 17.759
LW: 2 SW: 1 H: 20.0 PSNR: 19.166
LW: 2 SW: 1 H: 30.0 PSNR: 19.815
LW: 2 SW: 1 H: 50.0 PSNR: 19.753
LW: 2 SW: 1 H: 100.0 PSNR: 19.433

Fig 2.11 Images Processed by NML Filter (LW = 5, SW = 3)

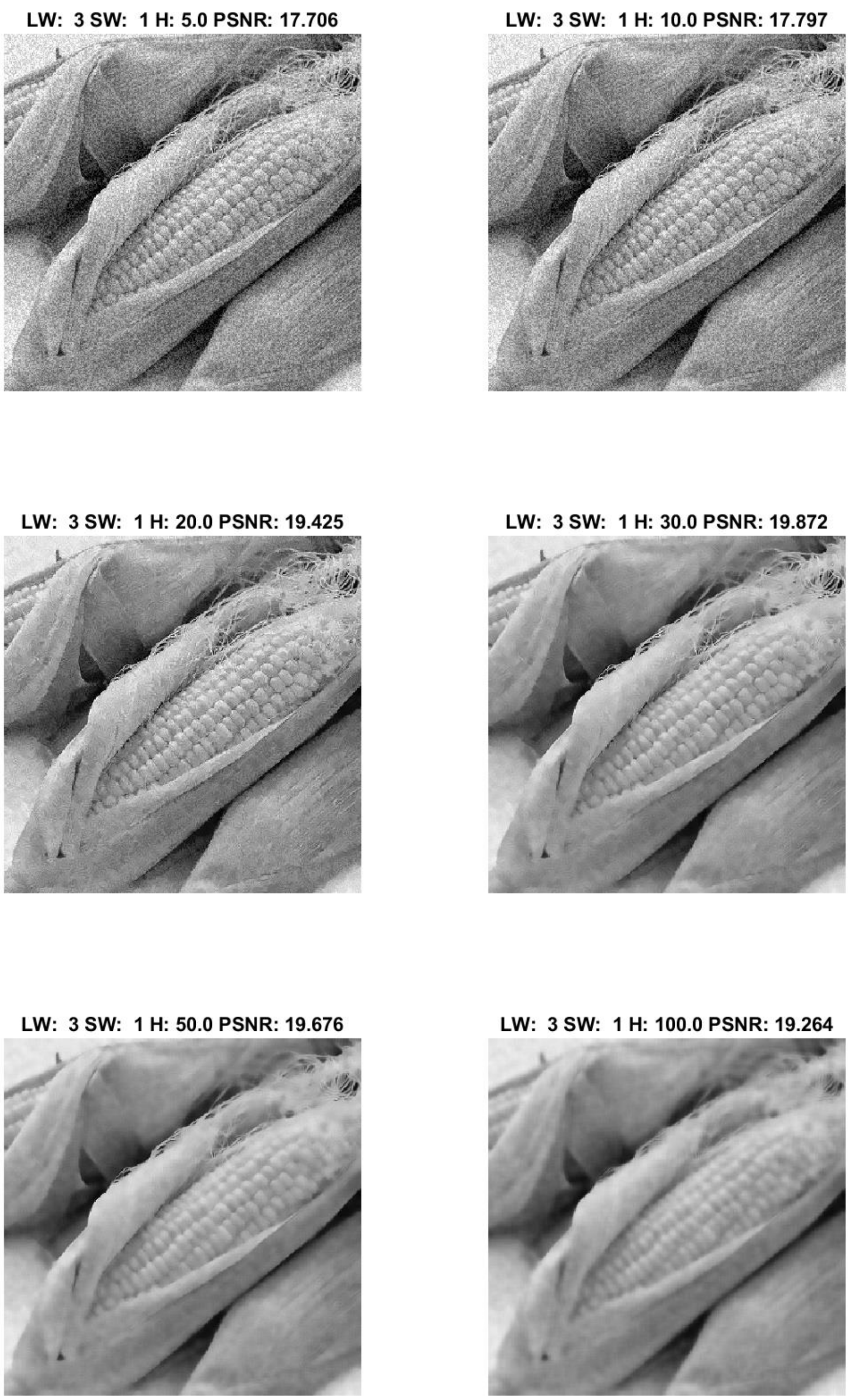

LW: 3 SW: 1 H: 5.0 PSNR: 17.706
LW: 3 SW: 1 H: 10.0 PSNR: 17.797
LW: 3 SW: 1 H: 20.0 PSNR: 19.425
LW: 3 SW: 1 H: 30.0 PSNR: 19.872
LW: 3 SW: 1 H: 50.0 PSNR: 19.676
LW: 3 SW: 1 H: 100.0 PSNR: 19.264

Fig 2.12 Images Processed by NML Filter (LW = 7, SW = 3)

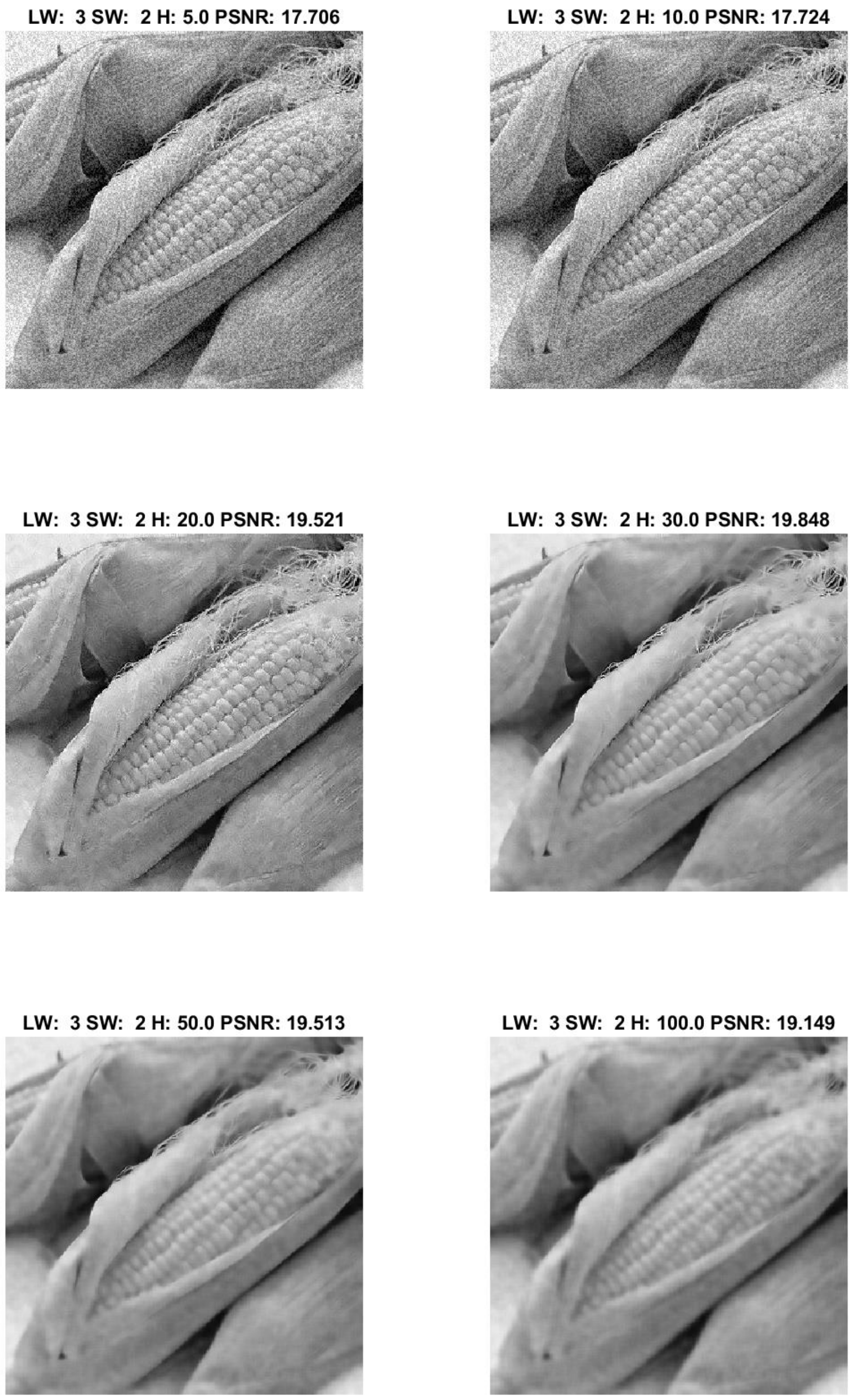

LW: 3 SW: 2 H: 5.0 PSNR: 17.706
LW: 3 SW: 2 H: 10.0 PSNR: 17.724
LW: 3 SW: 2 H: 20.0 PSNR: 19.521
LW: 3 SW: 2 H: 30.0 PSNR: 19.848
LW: 3 SW: 2 H: 50.0 PSNR: 19.513
LW: 3 SW: 2 H: 100.0 PSNR: 19.149

Fig 2.13 Images Processed by NML Filter (LW = 7, SW = 5)

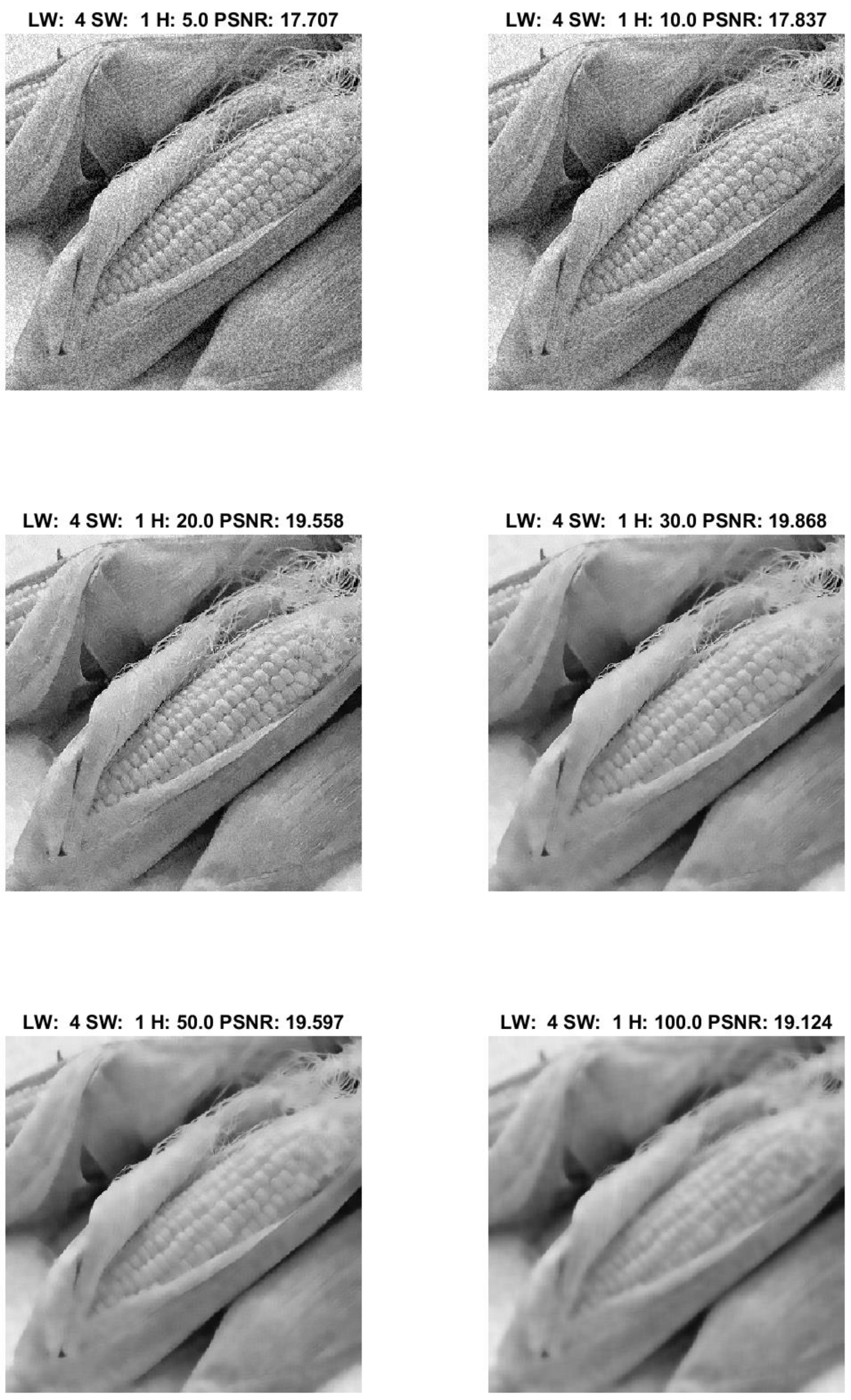

LW: 4 SW: 1 H: 5.0 PSNR: 17.707
LW: 4 SW: 1 H: 10.0 PSNR: 17.837
LW: 4 SW: 1 H: 20.0 PSNR: 19.558
LW: 4 SW: 1 H: 30.0 PSNR: 19.868
LW: 4 SW: 1 H: 50.0 PSNR: 19.597
LW: 4 SW: 1 H: 100.0 PSNR: 19.124

Fig 2.14 Images Processed by NML Filter (LW = 9, SW = 3)

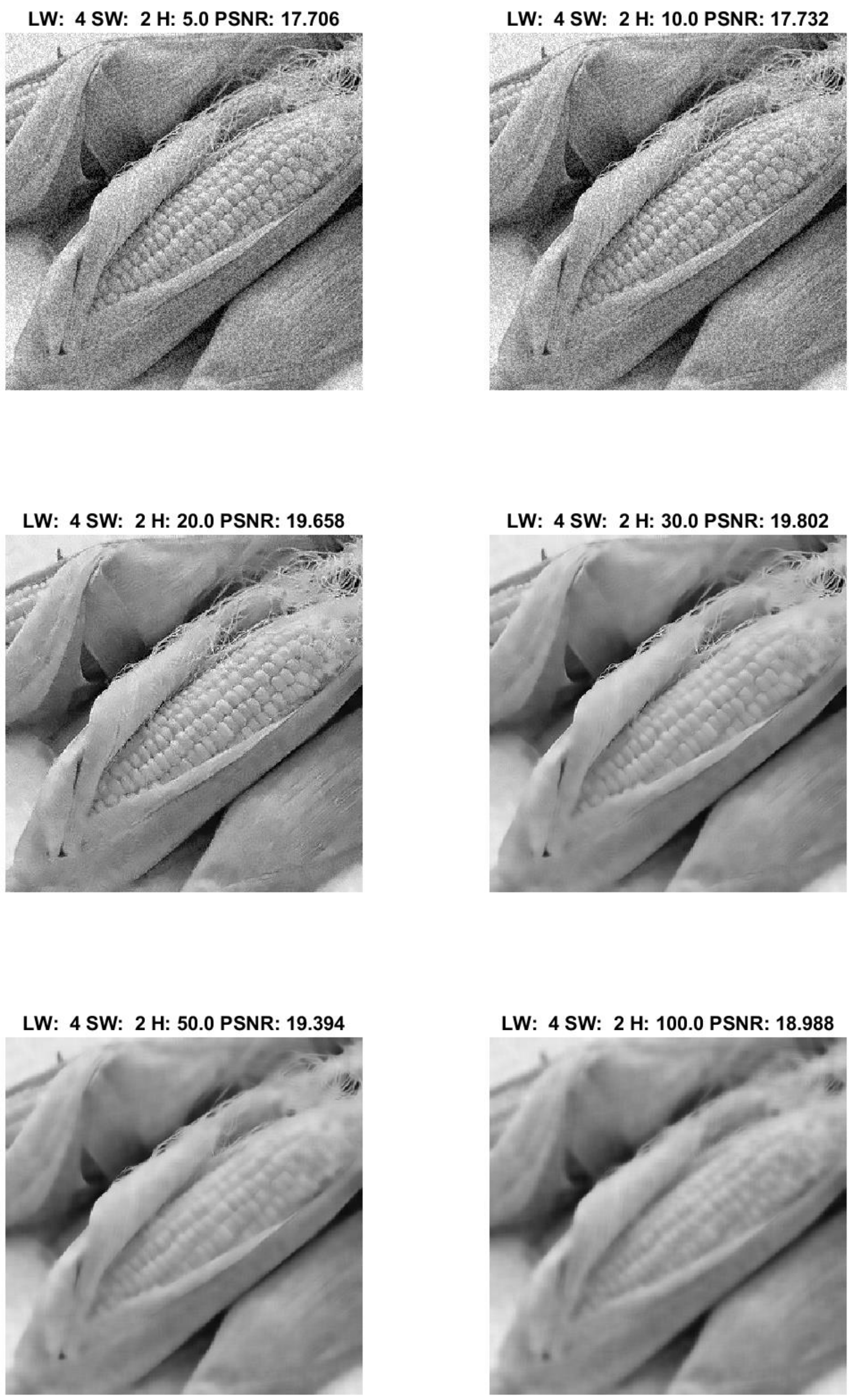

LW: 4 SW: 2 H: 5.0 PSNR: 17.706
LW: 4 SW: 2 H: 10.0 PSNR: 17.732
LW: 4 SW: 2 H: 20.0 PSNR: 19.658
LW: 4 SW: 2 H: 30.0 PSNR: 19.802
LW: 4 SW: 2 H: 50.0 PSNR: 19.394
LW: 4 SW: 2 H: 100.0 PSNR: 18.988

Fig 2.15 Images Processed by NML Filter (LW = 9, SW = 5)

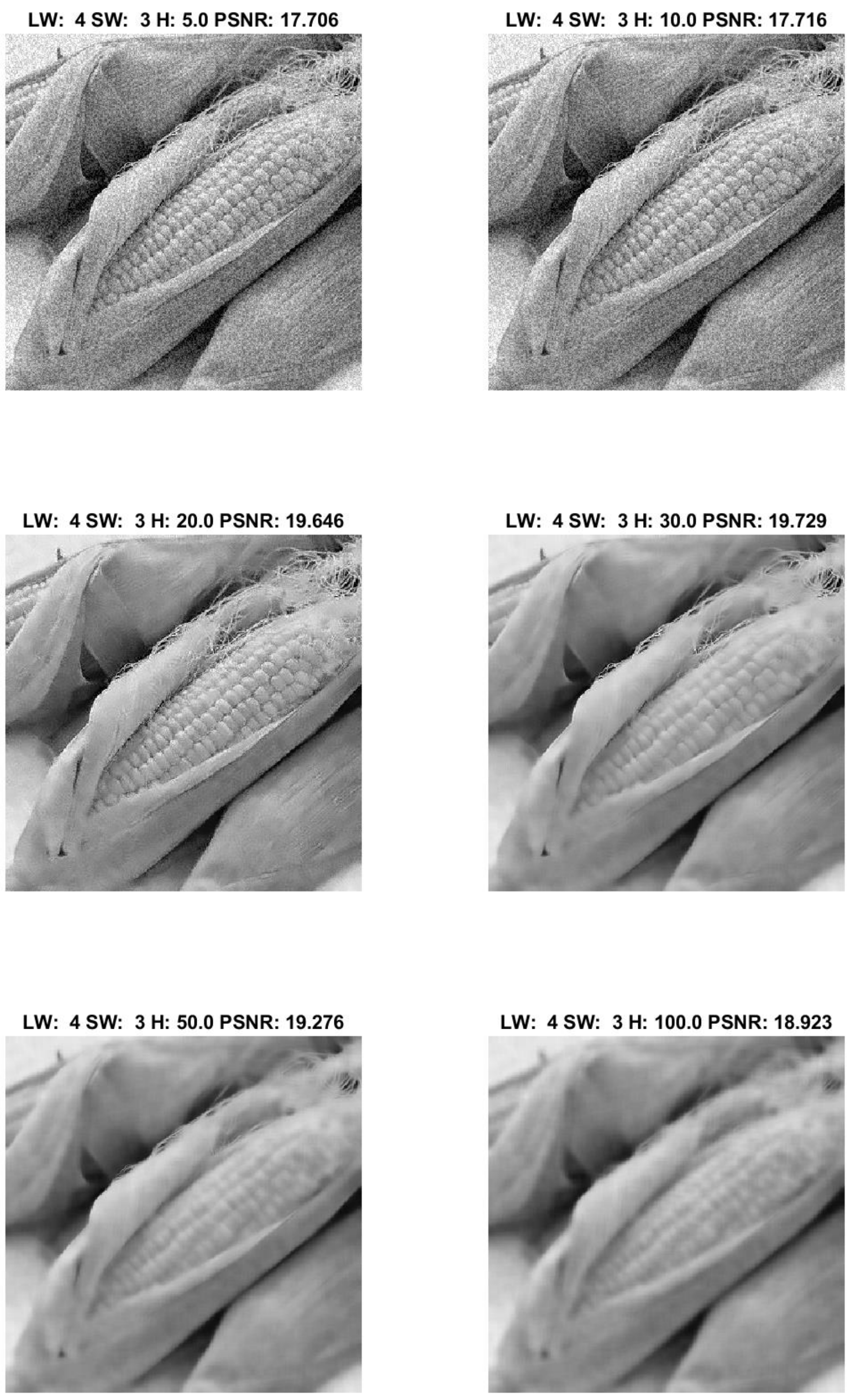

LW: 4 SW: 3 H: 5.0 PSNR: 17.706
LW: 4 SW: 3 H: 10.0 PSNR: 17.716
LW: 4 SW: 3 H: 20.0 PSNR: 19.646
LW: 4 SW: 3 H: 30.0 PSNR: 19.729
LW: 4 SW: 3 H: 50.0 PSNR: 19.276
LW: 4 SW: 3 H: 100.0 PSNR: 18.923

Fig 2.16 Images Processed by NML Filter (LW = 9, SW = 7)

Large Window	Small Window	H					
		5	10	20	30	50	100
5	3	17.706	17.759	19.166	19.815	19.753	19.433
7	3	17.706	17.797	19.425	**19.872**	19.676	19.264
	5	17.706	17.724	19.521	19.848	19.513	19.149
9	3	17.707	17.837	19.552	19.868	19.597	19.124
	5	17.706	17.732	19.658	19.802	19.394	18.988
	7	17.706	17.716	19.646	19.729	19.276	18.923

Table 2.3 PSNR (dB) of denoised image processed by NLM Filter

III. Discussion

(2) As shown in experimental result, the denoising effect of NLM algorithm is better than the algorithm in (a) and (b), and the image seems more similar to the original image with some certain parameter choices. However, NLM algorithm cost much more time to calculate than the former algorithm and there are more parameters to choose, which means to get the best result, the choice of parameter is a tough problem to be solve. Regarding reading material, I chose seldom other parameters recommended by the thesis, such as h is set to (10 * standard deviation) and size of large (search) window and small (comparison) window are set to 21 and 5 or 35 and 7 respectively, the outcome PSNR is not better than the best parameter listed above. For the sake of computational time, the additional parameter mentioned above is carried out with the help of open source function imnlmfilt(). The outcomes of my own NLM algorithm are also verified by imnlmfilt(), they are almost the same, the slight difference may be caused by different padding method. So, from my result, I will make the small window with side length 3 and large window with side length 7, the parameter H is set to 30.

(To set imnlmfilt(), J = imnlmfilt(G, 'DegreeOfSmoothing', H, 'SearchWindowSize', LW, 'ComparisonWindowSize', SW, where H, LW, SW are the parameters mentioned above and G is the image matrix to be denoised, J is the matrix representing denoised image.)

(d) Block matching and 3-D (BM3D) transform filter

I. Approach and Procedures

In this part, BM3D algorithm is employed the open source in the reference in HW_1 document.

II. Experimental Results

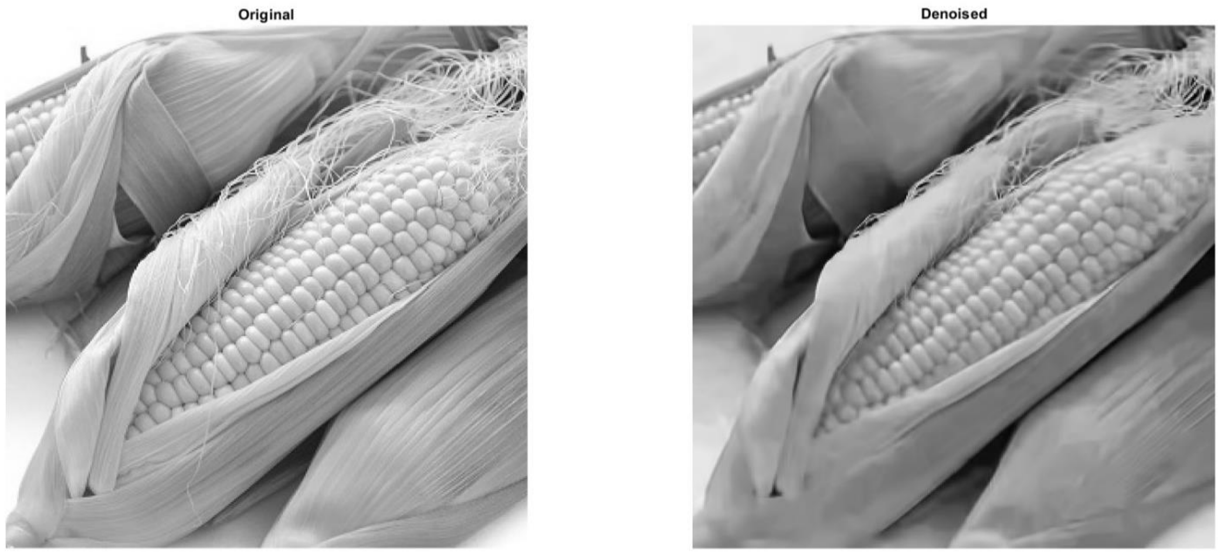


Fig 2.17 Original Image and Images Processed by BM3D

III. Discussion

(1) BM3D is a more advanced algorithm than the other three. There are two steps in general. The first step is basic estimation, and the other step is final estimation. In first step, every pixel block is grouped by their similarity. Then, for every target pixel block, up to a certain number of most similar blocks are chosen. In order to get rid of the impact of noise and find the most similar blocks to the target block, a local 2D transformation (DCT, DFT, etc.) is employed. Then, hard threshold operator is used to set all component which is smaller than a parameter to 0, the number of components set to 0 is recorded. Then, all the blocks in 3D space is processed by the inverse operator of the local 2D transformation just used. Then, all the blocks are put into the original place on the image, the values of pixel could be calculated by the corresponding block, weighted by their position and the number of components forced to 0 in hard threshold operation. After the first step, the noise is eliminated in general. In the second step, the basic-estimated image and the original noised image are used to generate 3D arrays separately, so two 3D arrays are obtained. For the array obtained by basic-estimated image, a Wiener filter with the energy spectrum of the basic estimates as pilot energy spectrum is applied. Then, Wiener coefficient is calculated. Then all the blocks containing noise is multiplied by Wiener coefficient as weight and then put back to the original place in 2D image, the denoised image is obtained.

(2) The PSNR processed by BM3D is 19.926, the standard deviation is set to 0.1 and could obtain a relative nice result, the final result is shown in part II, from the image, the effect of BM3D is much better than method in (a), the noise is almost eliminated and the detail of image is preserved better than any algorithm above. The detail of image is also recovered well.

(e) Mixed noises in color image

I. Discussion

(1) There is impulse noise which seems like black dot (0, 0, 0) and white dot (255, 255, 255), and Gaussian noise which interrupting the color slightly on the image.

(2) For the impulse noise, both processing by single channel and in the three channels in total work. For Gaussian noise, the image should be filtered by single channel.

(3) For the impulse noise, outlier filter and median filter could be used, for the Gaussian noise, the filter in the former questions could be used. But the sequence should be that outlier filter should be employed ahead of filters for Gaussian filter, for the impulse noise will have influence on statistic characteristic of Gaussian noise. To justified this, a solid color image added with both Gaussian noise and impulse noise is constructed, then outlier filter and basic filter for Gaussian filter is used in different sequence. In the experimental result, if the image is processed by outlier filter in the first hand then the basic filter, the image will be more like the original solid image than the wrong sequence which applies basic filter prior to outlier filter, and the PSNR of image in the right sequence is larger than that of wrong sequence.

II. Experimental results

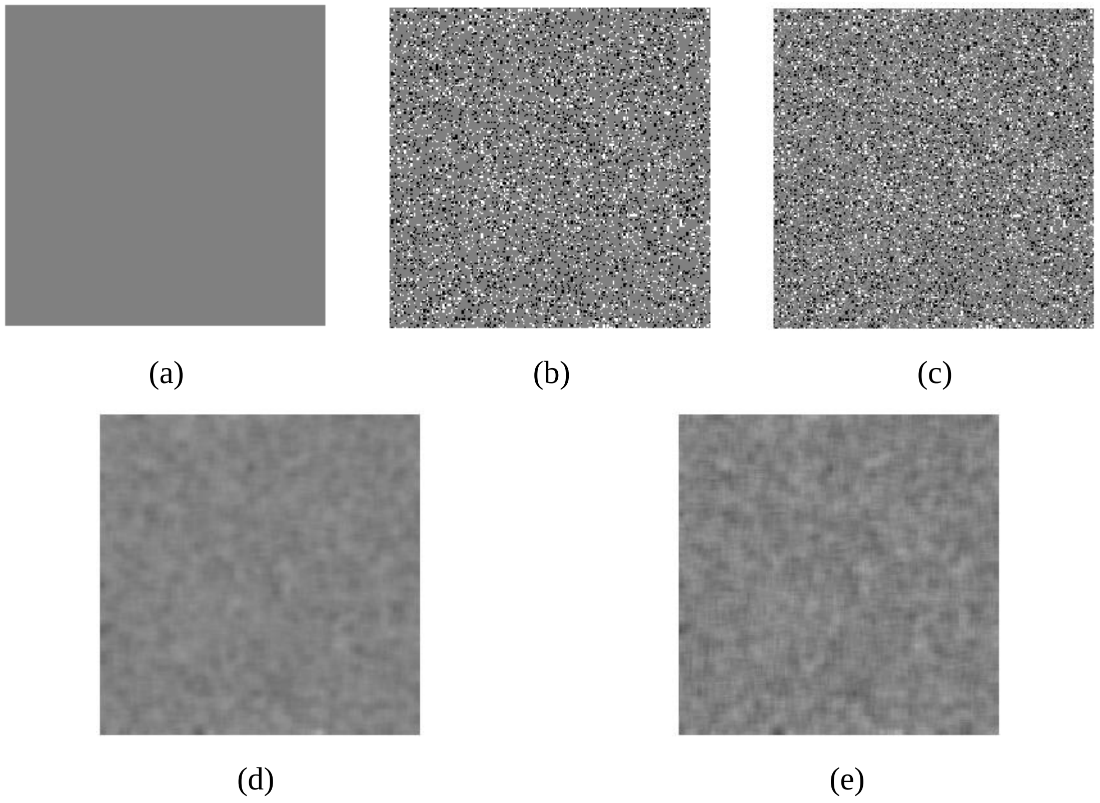


Fig 2.18 Mixed Noise processing

(a): original solid image, (b): image with impulse noise, (c): image with impulse noise and Gaussian noise, (d): image filtered in the right sequence, PSNR = 32.174dB (e): image filtered in the wrong sequence, PSNR = 28.933dB