HOMEWORK #0

University of Southern California

Lecturer: C.-C.JayKuo   Issued: 1/7/2019   Due:1/22/2019

Report

1 Problem 1: Image Demosaicing and Histogram Manipulation

1.1 Image demosaicing

1.1.1 Abstract and Motivation

Most digital image devices use bayer array sensor. The basic form of this kind of sensor is as follows: Each pixel of the Bayer array sensor is a monochrome pixel. The adjacent 2×2 pixels form a basic bayer pattern. Each array contains two pixel capturing only green values, one carrying blue, and one carrying red. This is a common method used to capture colour images while CMOS devices are used as light sensors.

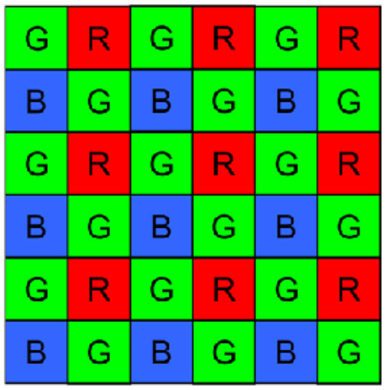

Figure 1: Bayer Pattern

In these cases, each pixel only carries value of one channel, which greatly saves the cost. Thus, demosaicing becomes a enssencial problems that need to be solved. I used bilinear demosacing method to re-constructed the full colour image based on their neighbourhood value. That is, the missing value of the pixel can be estimated by caculating the average of the adjacent pixels of the same colour. This is called the bilinear interpolation. Precisely, the value of pixel (i, j) can be caculatede by the following equation:

$$F(i,j) = \sum_{n=1}^{N} w_n p_n \tag{1}$$

Where p_n stands for the value of the adjacent pixels, and w_n stands for the weight of each pixel.

Figure 1 shows the origianl image of *cat.raw* with bayer pattern. My task is to apply the bilinear demoscaicing to the *cat.raw* and recover the colour image.

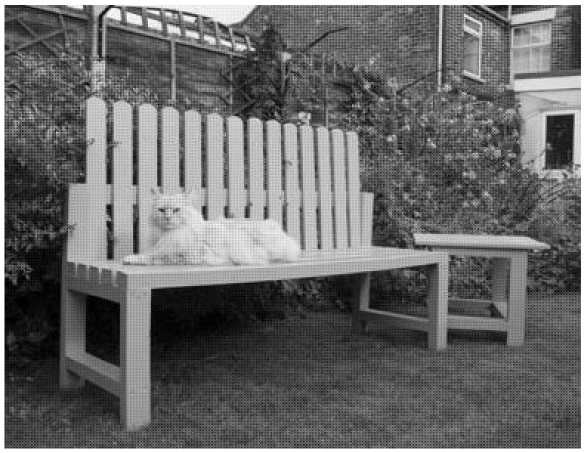

Figure 2: *cat.raw*

1.1.2 Approach and Procedures

At the first part of this problems, we only caculate the pixel containing the same color to the object pixel. For instance, the missing value in the blue channel is approximitely estimated by the average of the adjacent four or two blue pixels.
In order to preserve the same size as the input image, firstly I extension the image in to a $(Row+2)$ and $(Column+3)$ matrix by reflecting the entries along the four edges of original image. I simply do this transformation in order:

$$\begin{aligned}\hat{F}\left(\hat{i},\hat{j}\right) &= F\left(i,2-j\right) &(2)\\ \hat{F}\left(\hat{i},\hat{j}\right) &= F\left(2-i,j\right) &(3)\\ \hat{F}\left(\hat{i},\hat{j}\right) &= F\left(2\times Row-i,j\right) &(4)\\ \hat{F}\left(\hat{i},\hat{j}\right) &= F\left(i,2\times Column-j\right) &(5)\\ & &(6)\end{aligned}$$

After the first step, I finally get a $(Row+2)$ and $(Column+3)$ matrix. By observing the matrix, I have found out that green A pixels only appear at the even rows and columns. And so do other pixels appear the samilar property. Using this property, I distinguish every color pixels and caculate their R,G,B values.

In terms of objectively evaluation of image quality, Peak signal-to-noise ratio(PSNR) is used as the evaluation basis in this report. PSNR is the most common and widely used measurement method for evaluating image quality. Its mathematical definition is as follows:

$$MSE=\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left[I(i,j)-K(i,j)\right]^2 \tag{7}$$

$$PSNR=10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{8}$$

Here, MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255.

1.1.3 Experimental Result

The re-constructed image *cat_demosaicing.raw* by bilinear interpolation is shown in Figure 3. The re-constructed image *cat_MHC.raw* by MHC linear demosaicing method is shown in Figure 4. The original image *cat_ori.raw* is shown in Figure 5.

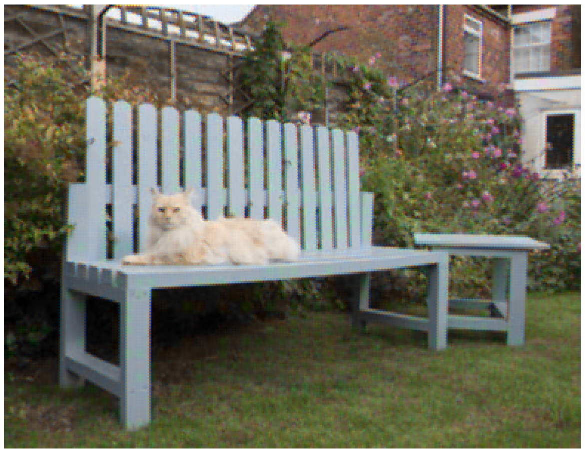

Figure 3: *cat_demosaicing.row*, PSNR = 30.8030

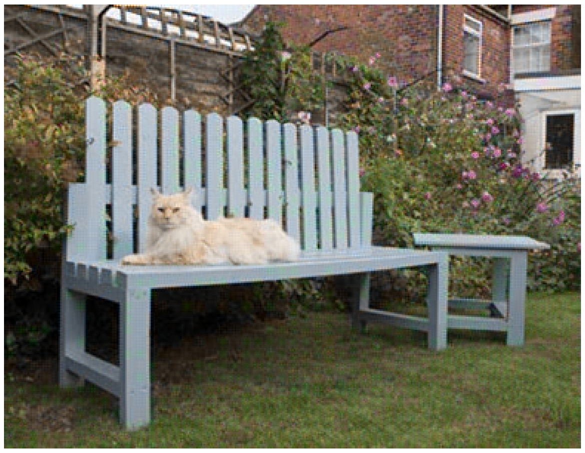

Figure 4: *cat_MHC.row*, PSNR = 34.7528

1.1.4 Discussion

Figure 3 shows the colour image that re-constructed by bilinear interpolation method. Figure num show the original image *cat_ori.row*. Compare to the original image, we can observe some artifacts patter appear at the bench obviously, where the color frequently change from white to grass green. Besides, this phenomenon also appeare at other edges of the object, such as cat and windows.

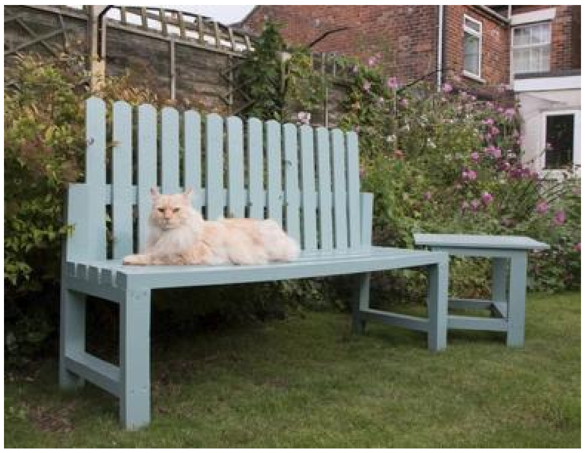

Figure 5: $cat\_ori.row$

As far as I am concerned, the cause of the artifacts is that I apply the bilinear interpolation to the whloe image without any restriction. However, this method only considers the influence of the colour values of the four or two immediate neighbors around the target pixel. And it does not take the influence of the rate of colour changing values between adjacent points into account, so it has the properties of a low-pass filter.

To resolve this problem, we take more entries into caculation to help estimating the target value. Therefore I use MHC-demosacing method to reduce the artifects.

Figure 4 shows the colour image re-constructed by MHC linear demosaiciing method. Compare to the Figure 3, we can clearly observe that the edges of the objects in the image re-constructed by MHC linear demosaicing method become much smoother, also the PSNR is much higher. And the artifact reduces efficiently. Thus, MHC method prossess higher calculation accuracy because it add the adjacent pixels to estimate the target value.

1.2 Histogram Manipulation

1.2.1 Abstract and Motivation

A good photo should give the audience much detail of the objects. Photos that are too dark or too bright will lose some details. Therefore, we need to adjust the white balance of the digital image. Histogram equalization is a method to process images in order to adjust the contrast of an image by modifying the intensity distribution of the histogram. The objective of this technique is to give a linear trend to the cumulative probability function(cdf) associated to the image. In this section, we use two methods to adjust the histogram of the image.

The method A is transfer-function based. The processing of histogram equalization relies on the use of the cumulative probability function ($cdf(x)$). The $cdf(x)$ is a cumulative sum of all the probabilities lying in its domain and defined by:

$$cdf\,(x) = \sum_{k=-\infty}^{x} P\,(k) \tag{9}$$

That is to say, by applying a transfer function to the original histogram, a linear $cdf(x)$ is associated to the uniform histogram that we want the resulting image to have.

$$\begin{aligned} g &= (g_{max} - g_{min})cdf(x) + g_{min} \qquad (10) \\ &= (255 - 1)cdf(x) \qquad (11) \end{aligned}$$

The method B is the cumulative-probability-based histogram equalization method. It is similar to method A in some aspects. The difference lies where we even the density so that every gray scale intensity is prossessed by the same number of pixels rather than caculate the transfer fucntion.

1.2.2 Approach and Procedures

Firstly, I caculate the histogram of the previous image and draw the histogram. Figure 6 and Figure 7 show the histogram of previous image.

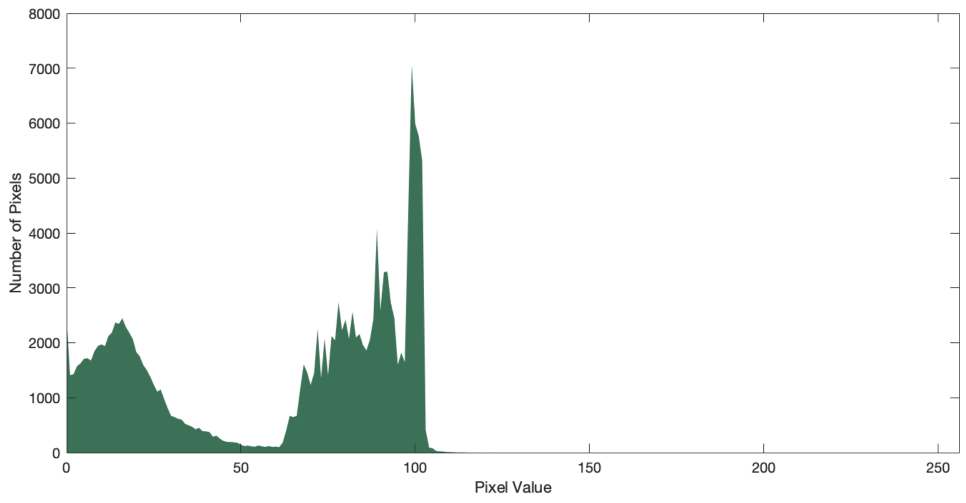

Figure 6: Histogram of *rose_dark.raw*

While looking at the associated histogram where we can see that the histogram are located on the left side which correspond to the darkest gray levels, and on the right side which correspond to the brightest. Secondly, I operate two method to modify the histogram of two images.

i Method A
In method A, to be able to perform further transformations on the image, I have to normalize the histogram by simply deviding the total number of pixels ($Row \times Column$). The normalization will give the properties of a probability density function $P(k)$. Therefore, I can integral the $P(k)$ to get the $cdf(x)$. The histogram transformation can be obtainede in approximate form by replacing the discrete probabaility distributions. The resulting approximation can be caculated by the transfer function (8).

ii Method B
Method B is much simplier than method A. Without using the transfer function, I just caculate

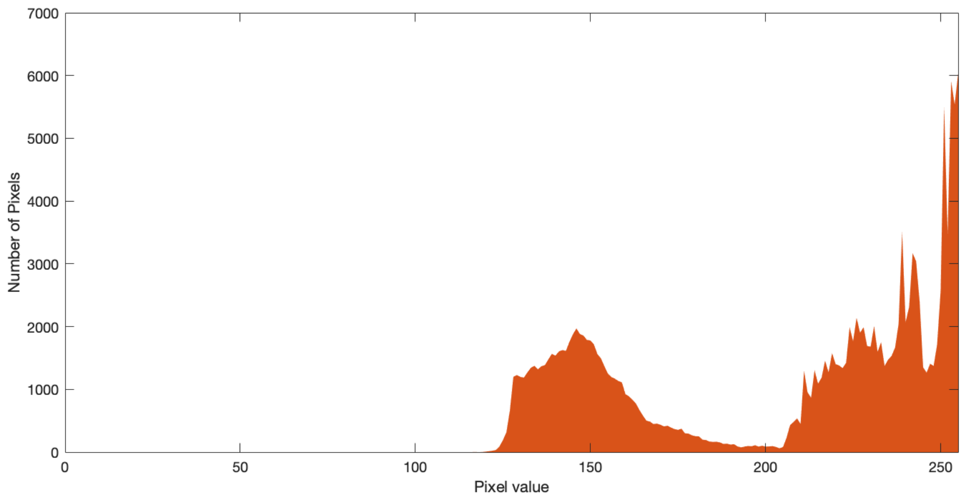

Figure 7: Histogram of $rose\_bright.raw$

the histogram of the previous image and sort the pixels value from 0 to 255. Then the pixels are assigned new value in order to ensure that each value is obtained by $\frac{Row \times Column}{256}$ number of pixels.

1.2.3 Experimental Result

Figure 8 shows the origianl image $rose\_ori$. Figure 9 and 10 shows the images that are too dark and too bright. Figure 11 to 17 shows the image that modified by two methods.

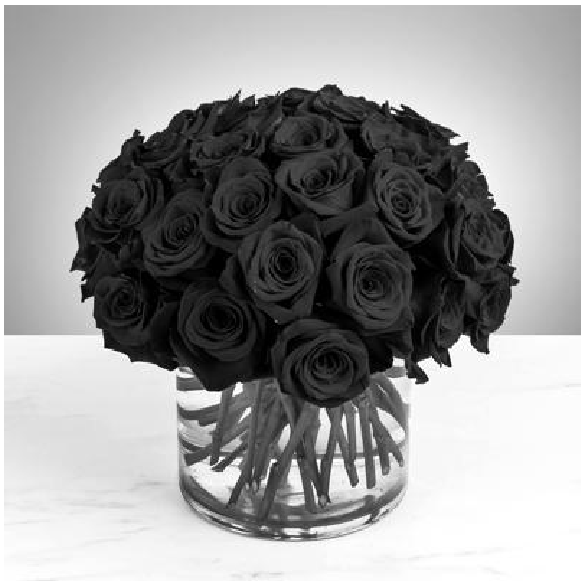

Figure 8: $rose\_ori.raw$

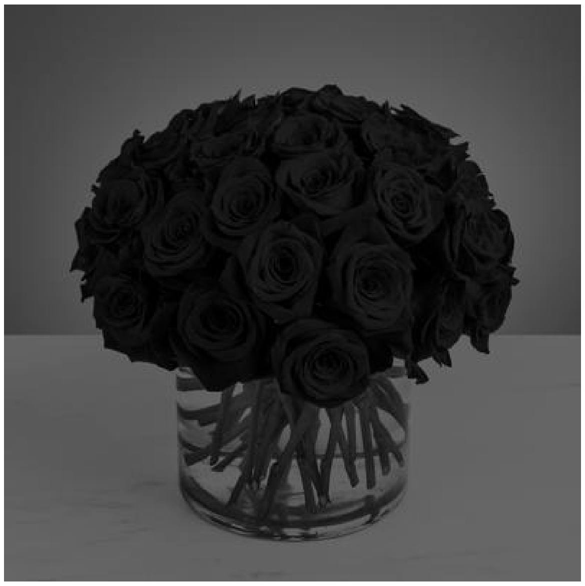

Figure 9: *rose_dark.raw*

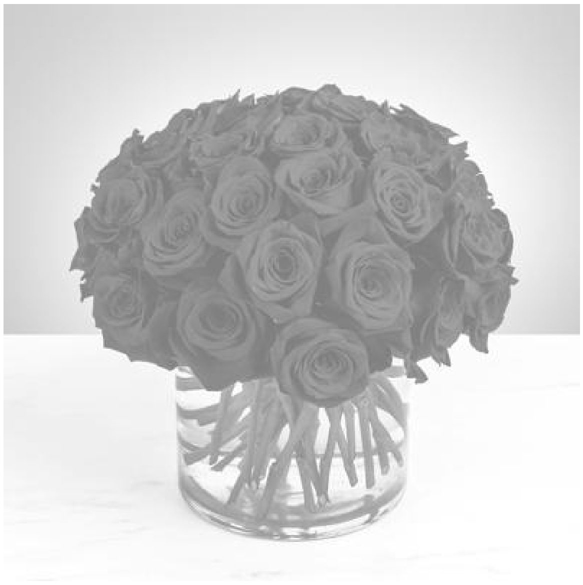

Figure 10: *rose_bright.raw*

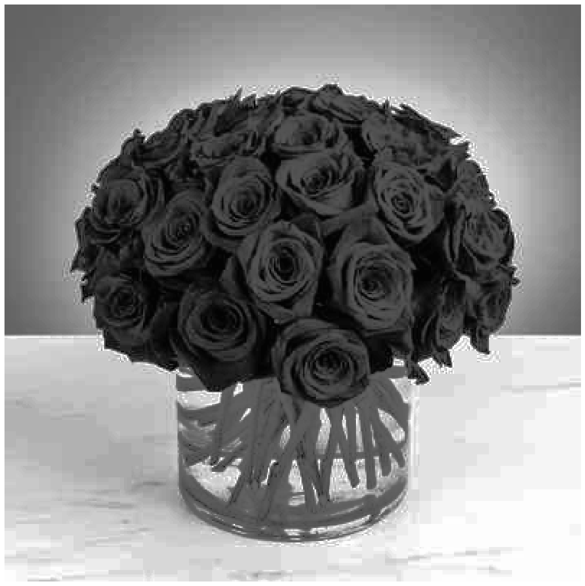

Figure 11: *rose_dark.raw* Modified by Method A

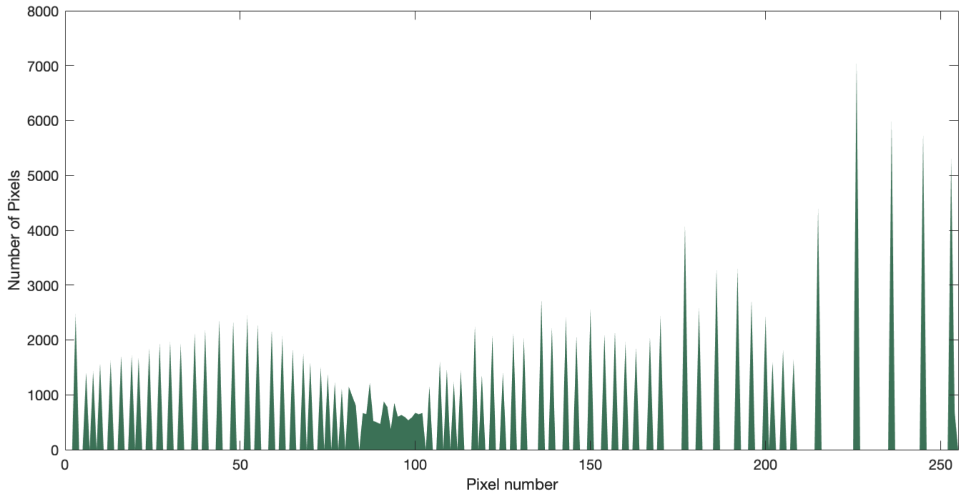

Figure 12: Histogram of Figure 11

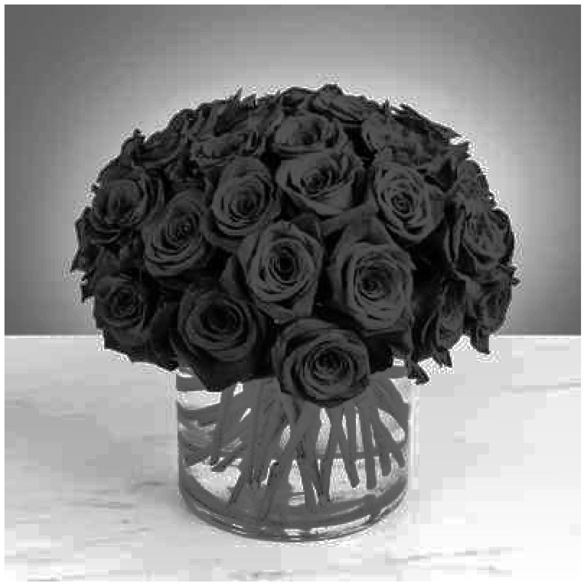

Figure 13: *rose_dark.raw* Modified by Method B

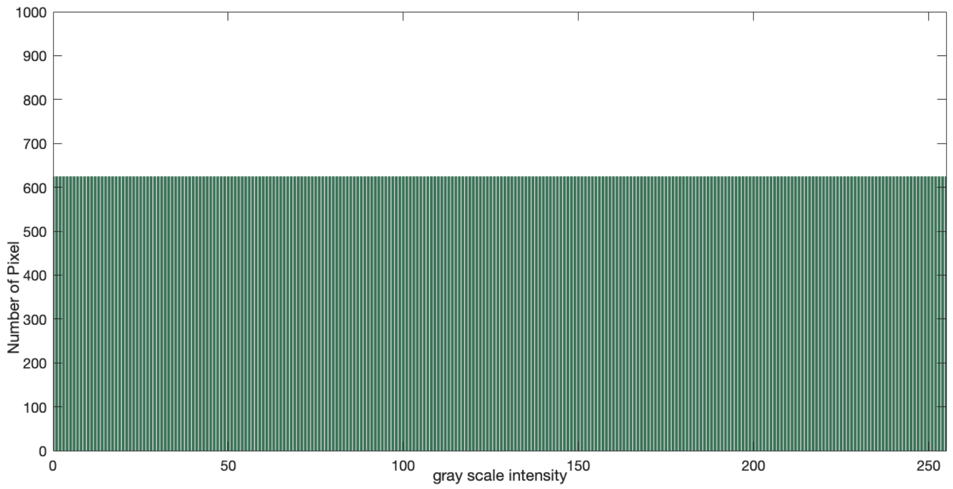

Figure 14: Histogram of Figure 13

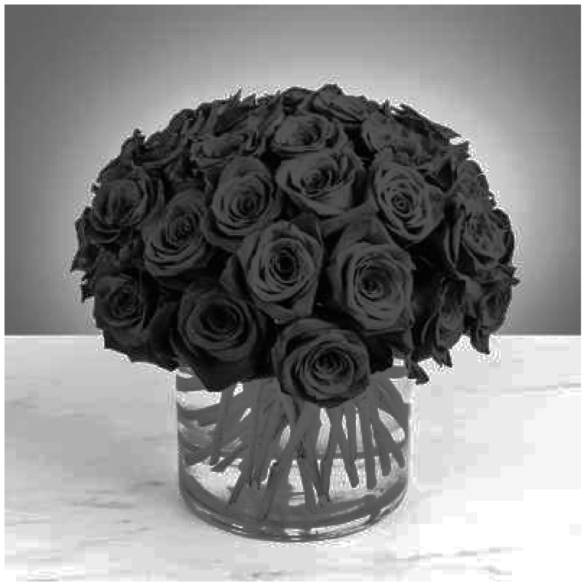

Figure 15: *rose_bright.raw* Modified by Method A

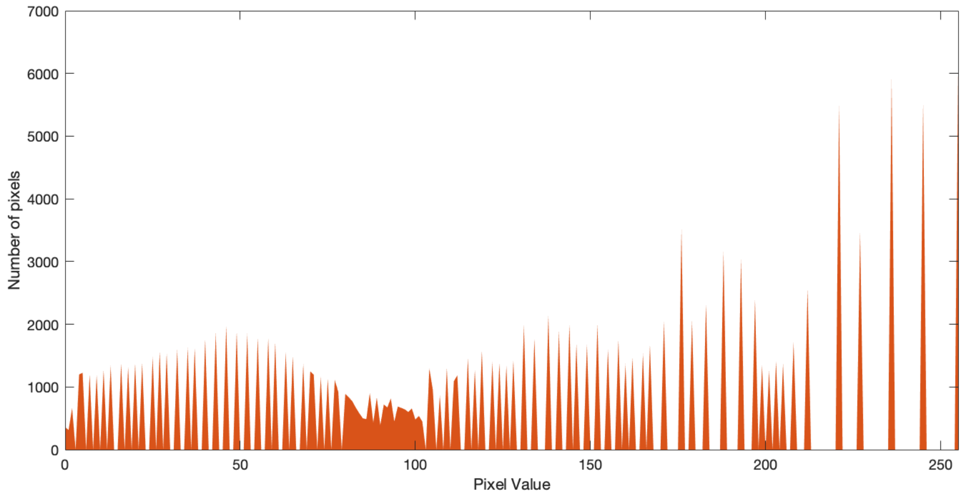

Figure 16: Histogram of Figure 15

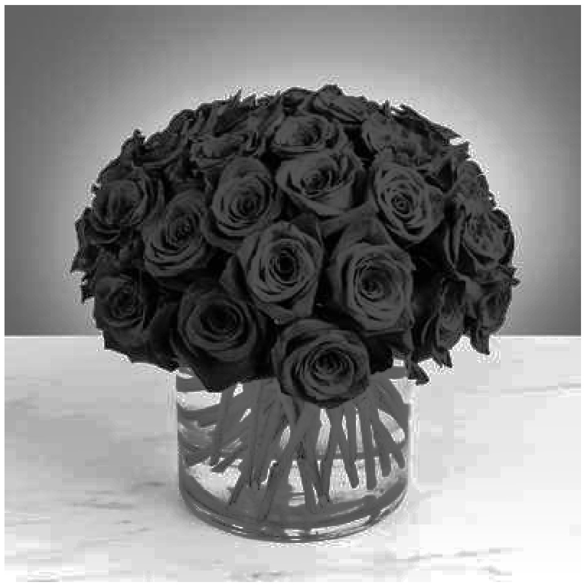

Figure 17: *rose_bright.raw* Modified by Method B

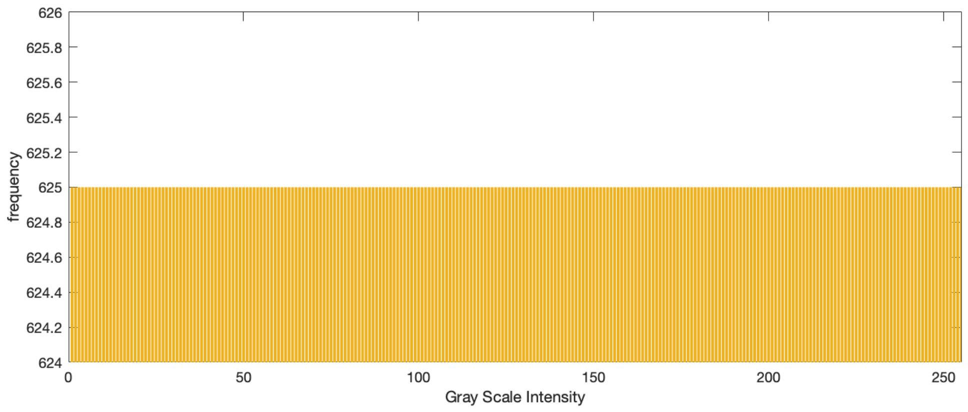

Figure 18: Histogram of Figure 17

1.2.4 Discussion

As we can see in the previous figure, if we compare the images we can see that the contrast of the image has clearly been enhanced by performing modification. Figure 19 and 20 are the transfer function, by which the output density is forced to be the uniform density. Figure 21 and 22 are the cumulative histogram that equalized by method B.

The success of the procedure is proven by the histogram associated to each image, showing that it's range has been stretched to occupy the whole spectrum of gray scale intensity. However, there is some mosaic pattern in all result images, especially in the white desk and the gray wall. That is because the histogram become discrete after modification.

Although the histogram of the grayscale distribution of the output image is close to uniform distribution, there may still be a large difference between the value and the ideal value which may not be the optimal. Moreover, The gray scale level of the output image may be excessively combined. That is, pixels with different gray scale intensity may become the same after processing, forming a slice with same gray level. There is obvious boundaries between the areas, resulting in a false outline.

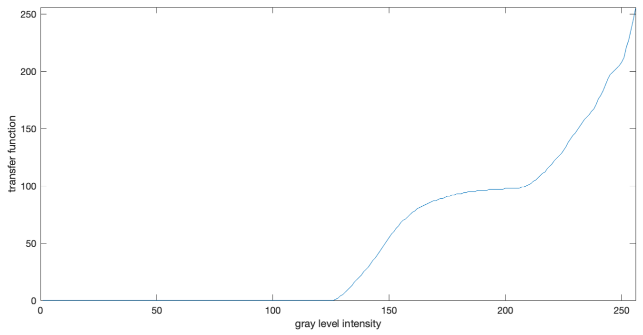

Figure 19: transfunction of *rose_bright.raw*

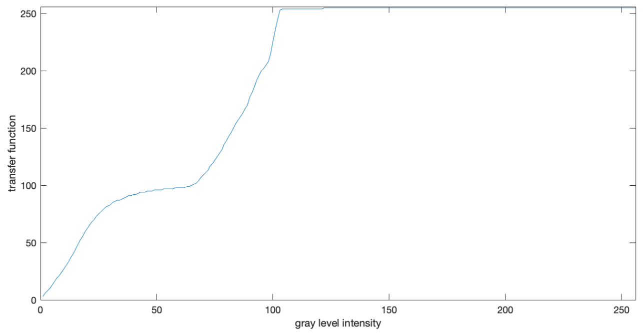

Figure 20: transfunction of *rose_dark.raw*

To reduce this phenomenon, we could block the test image and modify the histogram block by block. Similar to the kernel convolution method, taking the pixel to be processed as the center of the kernel, do the method A in the kernel and processing the result only to the central pixel. Block

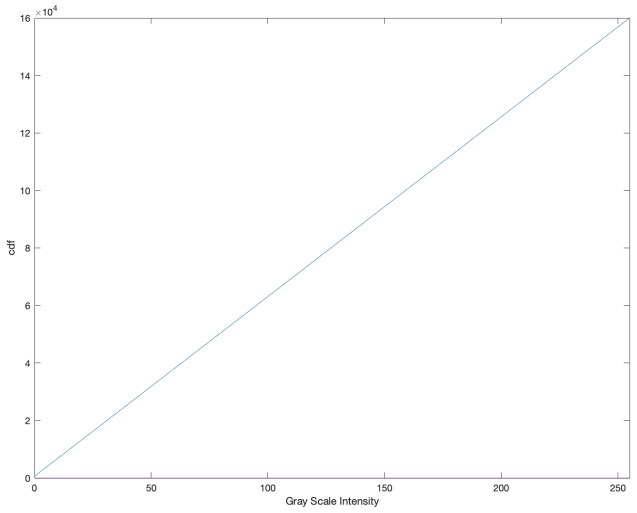

Figure 21: Cumulative Histogram of Figure 13

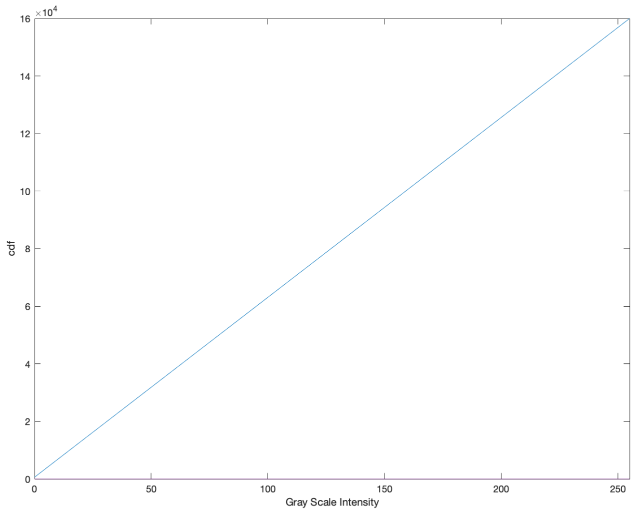

Figure 22: Cumulative Histogram of Figure 17

could be elimnated in this way. But this algorithm requires a histogram processing for every pixel, which is inefficient.

The PSNR value of each images are shown in Table 1. Comparing the method A and method B, The PSNR value of both *rose_dark.raw* and *rose_bright.raw* modified by method A are slightly better than method B, althought the visual quality does not distinguish a lot.

Table 1: PSNR of Two Method

	Original	Method A	Method B
rose_bright.raw	11.3634	16.8120	16.6478
rose_dark.raw	7.6299	16.8463	16.6363

While apply Method A and Method B to *rose_mix.raw* show in Figure 23, I cannot get the similar result as in previous part. The result images is shown in Figure 25 and 26 respect to the two methods. There is a quite obvious fake edge formed around the rose. Inside the circle, there is mosaic pattern. Figure 24 shows the histogram of *rose_mix.raw*. The density get larger both around 0 and 255. To modify my implementation, I suggest that the dark part and bright part should be modified respectively. Taking the index of the pixels into account, do the histogram modification in the dark region and bright region.

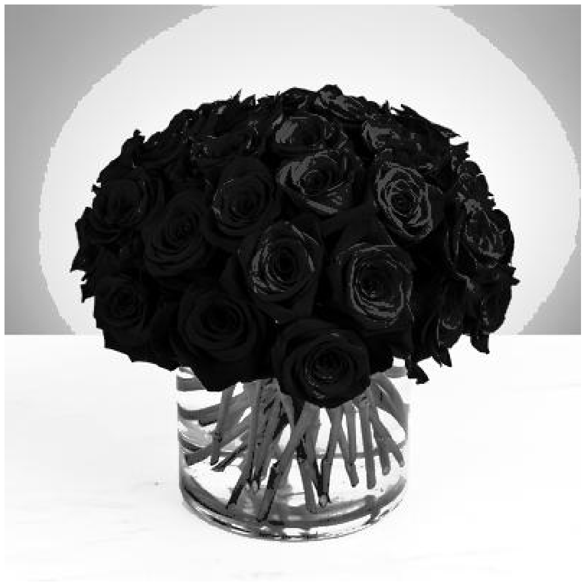
Figure 23: *rose_mix.raw*

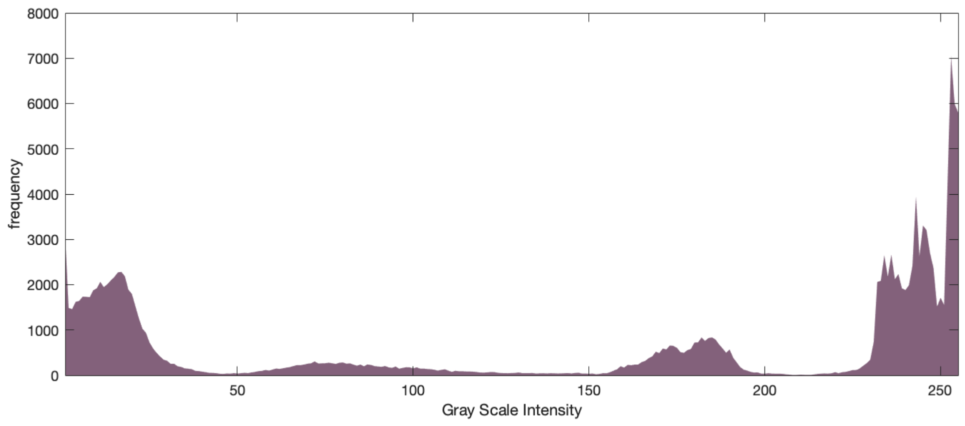

Figure 24: Histogram of *rose_mix.raw*

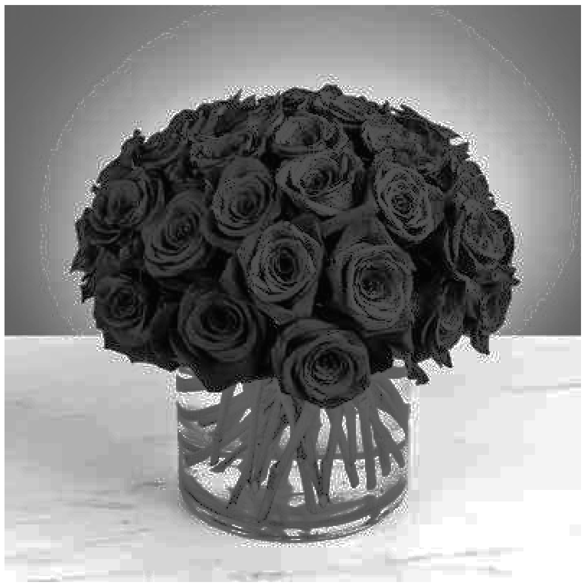

Figure 25: *rose_mix.raw* modified by method A

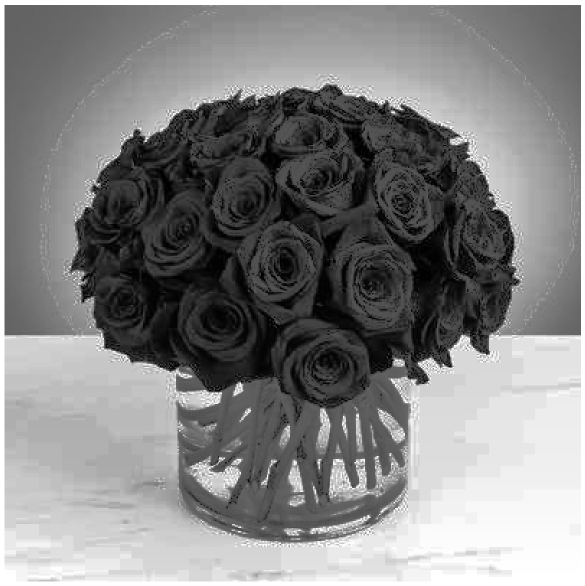

Figure 26: *rose_mix.raw* modified by method B

2 Problem 2 Image Denoising

2.1 gray-level image

2.1.1 Abstract and Motivation

Digital images may be contaminated by noise during acquisition and transmission. Common noises are Gaussian noise and salt and pepper noise. Precisely, Gaussian noise is mainly generated by the camera sensor components. The salt and pepper noise is mainly black and white bright and dark noise generated by image cutting. "Pepper" means black noise and "salt" means white noise. In this section, several linear and nonlinear techniques that have proved useful for noise reduction are performed to recover the image *pepper_uni.row*.

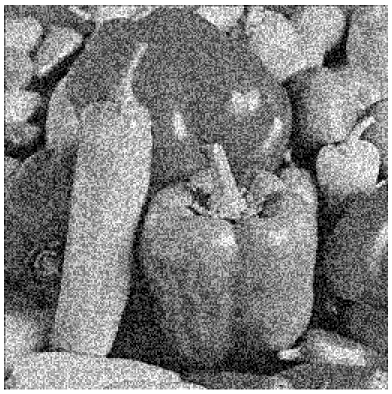

Figure 27: *pepper_uni.raw*

2.1.2 Approach and Procedure

Before starting denoising, the image should be expend to a larger matrix in order to maintain the same size as the input image, Using the same method in Section 1.1.2. Then I start denoising the image with several filters.

1 Linear Filter

i Uniform weighted Filter
The idea of uniform weighted filtering is simply to replace each pixel value in an image with the average value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Uniform weighted filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Here is a 3×3 kernel often used in uniform filter:

$$\boldsymbol{H} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{12}$$

ii Gaussian Filter
Gaussian kernel coefficients are sampled from the 2D Gaussian function. We know a 2D Gaussian function is defined as follows:

$$G\left(i,j\right) = \frac{1}{2\pi\sigma^2} exp(-\frac{i^2+j^2}{2\sigma^2}) \tag{13}$$

Where σ is the standard deviation of the distribution. The distribution is assumed to have a mean of zero. We need to discretize the continuous Gaussian functions to store it as discrete pixels. Then convolution process is applied to get target value for each pixel.

$$G\left(i,j\right) = \frac{1}{2\pi\sigma^2} exp(-\frac{(i-k)^2+(j-l)^2}{2\sigma^2}) \tag{14}$$

Where (k, l) is the relative distance from the center of the kernel.

2 Non-linear Filter

i Bilateral Filter
Bilateral filter has very similar convolution kernel as Gaussian filter. However, it does not consider the relative position of target pixel and neighboring pixel, but also consider the difference between their values. In Gaussian filter, the closer these two pixels are, the more weight neighboring pixel would have. However, In bilateral filter, the more 'similar' these two pixels are, the more weight neighboring pixel would have. In bilateral filter template, the value of convolution kernel is defined by two factors $l(i, j, k, l)$ and $r(i, j, k, l)$:

$$w(i,j,k,l) = l(i,j,k,l) \cdot r(i,j,k,l) \tag{15}$$

These two factors are defined as:

$$r(i,j,k,l) = exp(-\frac{||I(i,j) - I(k,l)||^2}{2\sigma_s^2}) \tag{16}$$

$$l(i,j,k,l) = exp(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_c^2}) \tag{17}$$

Then we get $w(i,j,k,l)$:

$$w(i,j,k,l) = exp(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_c^2} - \frac{||I(i,j) - I(k,l)||^2}{2\sigma_s^2}) \tag{18}$$

For each pixel (i,j), its value is determined by its neighboring pixels (k,l):

$$Y(i,j) = \frac{\sum\limits_{k,l} I(k,l)w(i,j,k,l)}{\sum\limits_{k,l} w(i,j,k,l)} \tag{19}$$

$I(k,l)$ is the gray sacle value of pixel (k,l).

ii Non-local Mean Filter
Compare with bilateral filter, Non-local mean filter utilize the pixel value from a larger region instead. The non-local mean filter use Gaussian weighted distance to measure bothe the similarity and the distance between two kernels. The Gaussian weighted Euclidian distance between window $I\left(N(i,j)\right)$ and $I\left(N(k,l)\right)$ is defined as:

$$\|I\left(N_{i,j}\right) - I\left(N_{k,l}\right)\|_{2,a}^2 = \sum_{n_1,n_2 \in N} G_a(n_1,n_2)(I(i-n_1,j-n_2) - I(k-n_1,l-n_2))^2 \tag{20}$$

and

$$G_a(n_1,n_2) = \frac{1}{\sqrt{2\pi}a} exp(-\frac{{n_1}^2 + {n_2}^2}{2a^2}) \tag{21}$$

Where $N_{i,j}$ denote the window center around the pixel (i,j), n_1,$n_2 \in N$ denotes the relative position in the neighbourhood window. a is the standard deviation of the Gaussian kernel. Therefore, for each pixel (i,j), its value is determined by the similar kernel around it:

$$Y(i,j) = \frac{\sum_{k=1}^{N'} I(k,l)f(i,j,k,l)}{\sum_{k=1}^{N'}\sum_{l=1}^{M'} f(i,j,k,l)} \tag{22}$$

$$f(i,j,k,l) = exp\left(-\frac{\|I\left(N_{i,j}\right) - I\left(N_{k,l}\right)\|_{2,a}^2}{h^2}\right) \tag{23}$$

2.1.3 Experimental Result

The original pepper image and the noise image are showed in Figure 28 and Figure 29. Noise distribution of $pepper_uni.raw$ and the output image filtering by low-pass filter is shown in Figure 30. Figure 31 to 34 show the output image denoised by several filters.

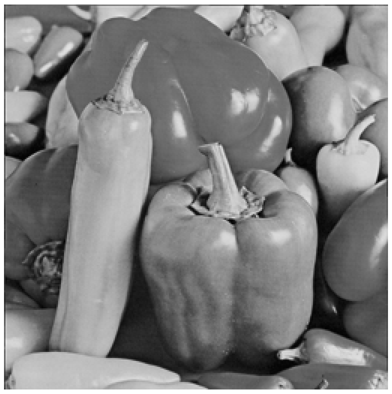

Figure 28: *pepper.raw*

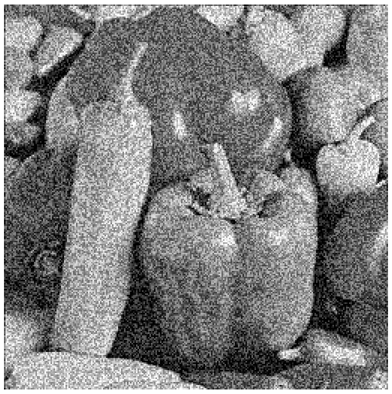

Figure 29: *pepper_uni.raw*

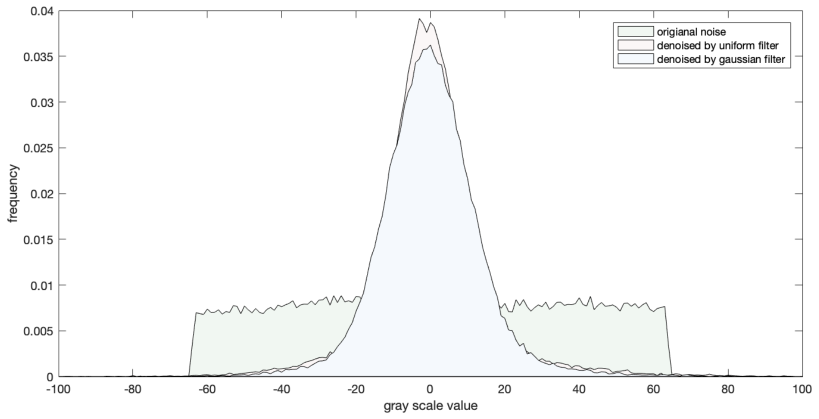

Figure 30: Noise Distribution

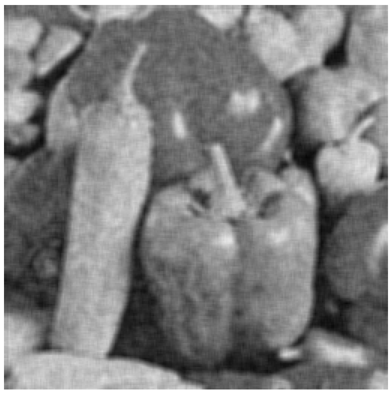
Figure 31: *pepper_uni.raw* Denoised by Unifrom Weighted Filter ($N = 5$)

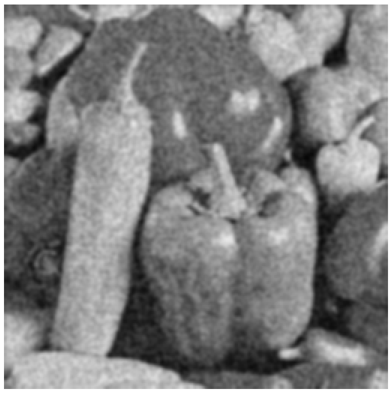
Figure 32: *pepper_uni.raw* Denoised by Gaussian Weighted Filter ($N = 11, \sigma = 0.8$)

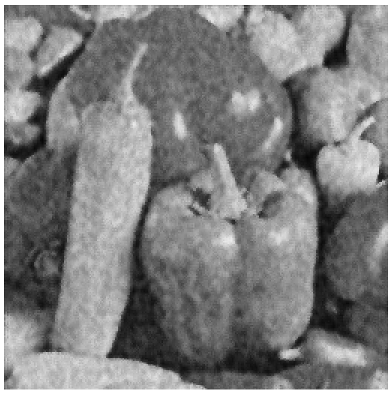
Figure 33: *pepper_uni.raw* Denoised by Bilateral filter ($N = 5, \sigma_c = 10, \sigma_s = 100$)

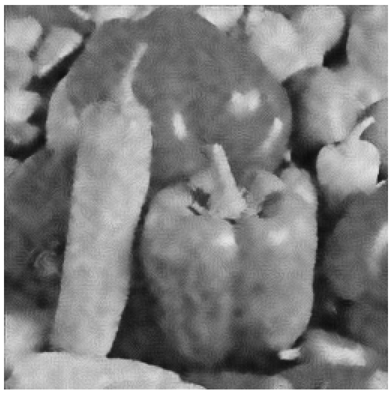

Figure 34: *pepper_uni.raw* Denoised by N-L Mean Filter($N = 3, M' = N' = 6, h = 100, a = 1$)

2.1.4 Discussion

It could be clearly observed that the visual quality of the denoised image is improved. The noise reduces at the cost of bluring the edges of objects. The PSNR value as a function of N for uniform weighted and Gaussian weighted filter is plotted in Figure 35.

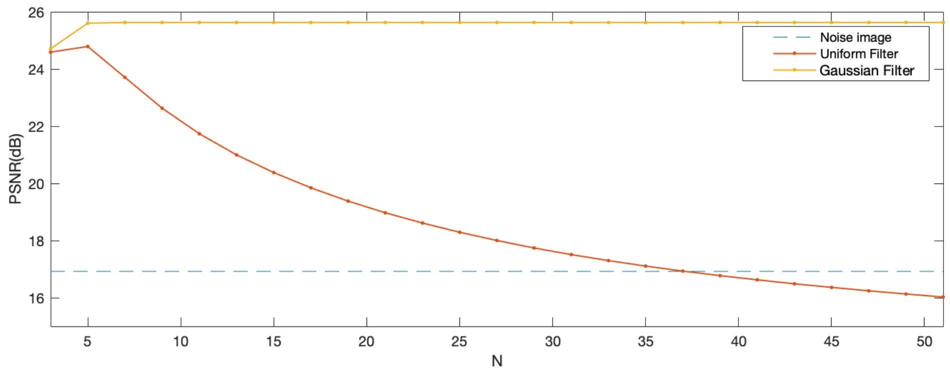

Figure 35: PSNR value sd a function of N

The PSNR value of noise image is 16.9218. As uniform filter, the PSNR value reaches the highest point when $N = 5$, and goes down by the increacing of N. Unilike the uniform filter, after reaching the peak PSNR value, it remains at the peak value when N increases. This difference is caused by the entries in Gaussian kernel decrease from center to the edges. Althought the keneral size increase, the entries at the edges of the kenerl contribute less to the denoising process. Thus, the quality of output image decrease while enlarging the window of uniform filter kernel.

When it comes to non-linear filters, such as bilateral filter and non-local mean filter, the noise is efficiently reduced and the edges are preserved. non-local mean filter perform better than bilateral filter. It not only greatly reduce the noise but remain the object's edges clearly.

2.2 Colour Image

2.2.1 Abstract and Motivation

Figure num is a noisy colour image *rose_color_noise.raw*, where each chanenel is corrupted by both impulse noise and uniform noise. The noise should be romove orderly with the combination of different filters. Filtering should be performed on channels *separately* for both noise type.

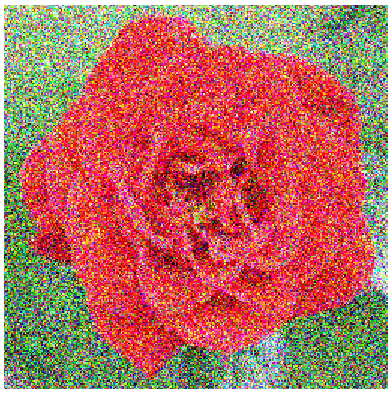

Figure 36: *rose_color_noise.raw*

Impule noise is sudden white and black pixels, which remind one of salt and pepper sprinkled over a photo. To remove the impule noise, I implement median filter. The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighbouthood entries. The pattern of neighbors is called the 'window', which slides, entry by entry, over the entire signal. For 2D images, the window size normally has an odd number of entries, like 3, 5 or 7. In this section, I implement median filter and bilateral filter to reduce the mixed noise. I will cascade these filter in different order to test the capabiliaty of denoising.

2.2.2 Approach and Procedure

Figure 37 shows the noise distribution in three channels. The figure implies that the noise distribution distinguishes in three channels. And we should perform filtering on each channel. I try different kind of combination to reduce the uniform noise and impulse noise. As we claim in previous section, low-pass filter will blur the edges obviously. Therefore, the combination of the two filters will enhance the blur effect. Considering the algorithm complexity and the efficience, I choose the bilateral filter and median filter to denoise.

2.2.3 Experimental Result

Figure 38 is the origianal image *rose_color.row*. I try cascading Bilateral filter and median filter in two order. The results are shown in Figure 39 and 40.

2.2.4 Discussion

We can observe that the Bilateral + Median performs better the other one. We can conclude that the order of filter matters a lot. Because the pixels value with impulse noise is either too large

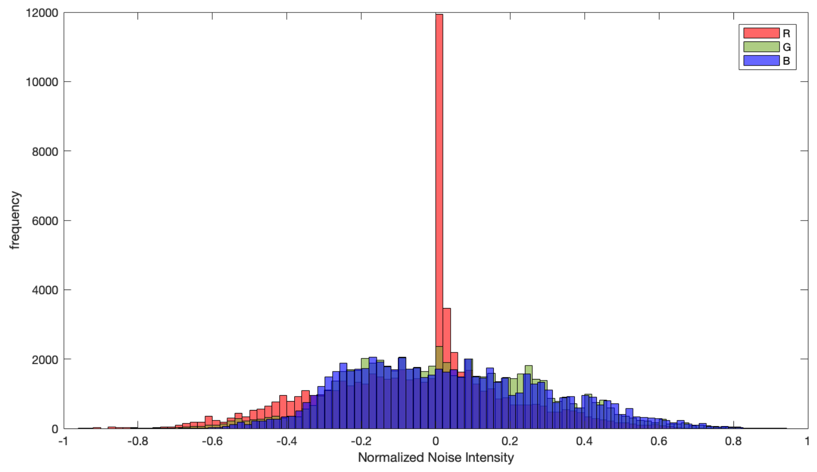

Figure 37: Noise Distribution in Three Channels

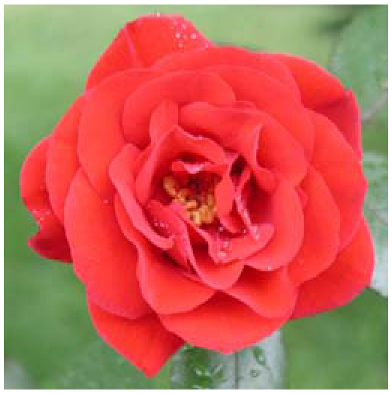

Figure 38: $rose\_color.raw$

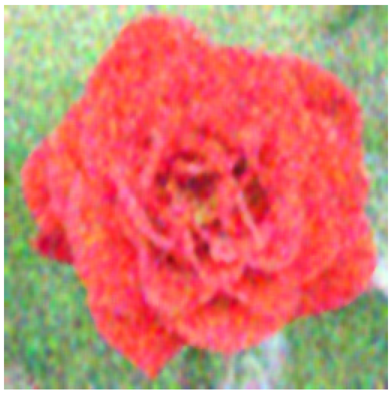

Figure 39: Median Filter($N = 5$) + Bilateral Filter($N = 9, \sigma_c = 60, \sigma_s = 50$), PSNR=18.7884

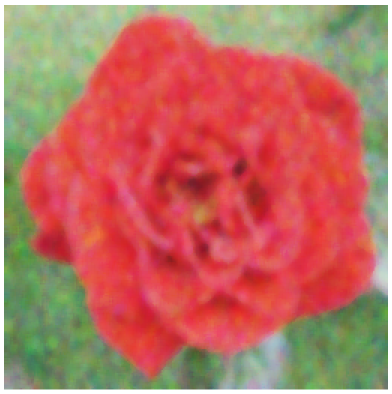

Figure 40: Bilateral Filter($N = 9, \sigma_c = 60, \sigma_s = 50$) + Median Filter($N = 5$), PSNR=22.4929

or too small, thus contaminating the neighbourhood pixels while performing a linear filter to the image. If we use a linear filter to the image, it should go after the median filter.

Nevertheless, the object's edge still blurred after two rounds filtering. To alleviate this phenomenon, we can add edge detection factor to measure the weight of the filters. The edge gradient is used as the threshold of color segmentation, which could be estimate with the neighborhood of the original image. The gradient can reflect the edge of the image. In general, if there is a continuous color transition interval in the original image. That is, the variation of the gray value of the pixel is relatively slow within a certain range. Similar to the interpolation mathod, I choose four points in four directions as the coefficients of four equations. In the stationary region, two sets of equations determine two sub-stationary regions. At the steep point, the two sets of equations determine two discontinuous regions with discontinuous lines as common edges. In this way can the edges be approximately detected. Adjust the kernel's coefficient by using the estimated edges as the dividing line. To increase accuracy, more directions can be added to more accurately estimate the direction of the edges.

2.3 Shot Noise

2.3.1 Abstract and Motivation

Shot noise, the time-dependent fluctuations in electrical current caused by the discreteness of the electron charge, is well known to occur in solid-state devices, such as tunnel junctions, Schottky barrier diodes and p-n junctions. Noise per pixel depend on the original pixel value itself. Each pixel z follows poisson distribution as shown below:

$$P(z) = \frac{\lambda^z e^{-\lambda}}{z!} \tag{24}$$

The mean and the variance are not independent:

$$E[z] = \lambda \tag{25}$$

$$var(z) = \lambda \tag{26}$$

In order to remove the possion noise, I apply Anscombe transformation. By applying Anscombe transformation, Poissonian data was transform to approximately Gaussian data with mean larger than 4 and unit variance. Then I use low-pass filter (Gaussian filter) and BM3D filter for futher denoising.

2.3.2 Approach and Procedure

At the first step, I applay Anscombe tranformation to the test image *pepper_dark_noise.raw*. The Anscombe forward transform is defined as followed:

$$D = f(z) = 2\sqrt{z + \frac{3}{8}} \tag{27}$$

Then the poisson distributed noise is transfrom to Gaussian noise. I apply Gaussian filter and BM3D filter sepreately. After denosing, There are two inverse transformation.

$$Z' = f^{-1}(D) = \left(\frac{D}{2}\right)^2 - \frac{3}{8} \quad (biased) \tag{28}$$

$$Z' = f^{-1}(D) = \left(\frac{D}{2}\right)^2 - \frac{1}{8} \quad (unbiased) \tag{29}$$

The main step of BM3D is grouping and collaborative filtering. Image fragments are grouped together based on similarity, but unlike standard k-means clustering and such cluster analysis methods, the image fragments are not necessarily disjoint. This block-matching algorithm is less computationally demanding and is useful later-on in the aggregation step. Fragments do however have the same size. The fragments whose distance (i.e., dissimilarity) from the reference one is smaller than a given threshold are considered mutually similar and are subsequently grouped.

Collaborative filtering is done on every fragments group by Shrinkage in Transform Domain. Specifically, a $d+1$ dimensional linear transform is applied, followed by a transform-domain shrinkage such as Wiener filtering, then the linear transform is inverted to reproduce all (filtered) fragments.

2.3.3 Experimental Result

The origianal image *pepper_dark.raw* and the image with shot noise are shown in Figure 41. Figure 42 and 44 are the output image filtering by gaussian filter(unbiased and biased). The noise distribution of the privious image, image after transformation and the Gaussian filtered output image are shown in Figure 47.

I also implement BM3D. Figure 45 and 46 are the result images. First I do the transformation. The I put the transformed image into BM3D function. The shot noise become Gaussian noise with unit variance, so I set noise variance equal to 1.

[PSNR, pepper_dark_BM3D] = BM3D(1, pepper_dark_trans,1,'lc',0);

After filtering, I transform The PSNR of biased and unbiased is 38.9023 and 39.674.

2.3.4 Discussion

The PSNR value of the output image denoised by BM3D filter is slightly higher than Gaussian filter. The shot noise is efficiently removed by filters. Comparing the biased transformation and

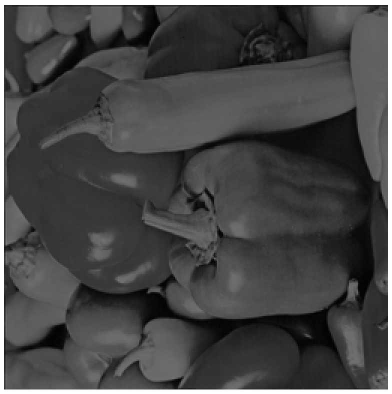

Figure 41: *pepper_dark.raw*

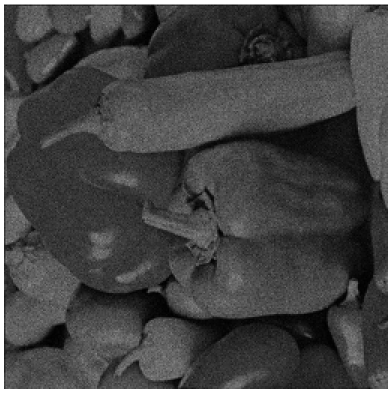

Figure 42: *pepper_dark_noise.raw*

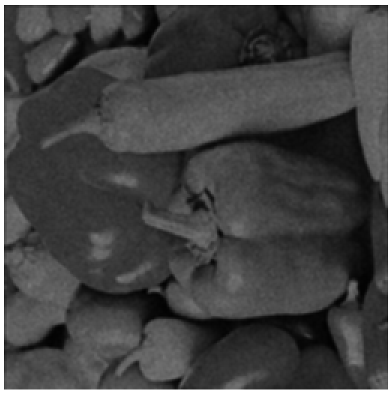

Figure 43: Denoised by Gaussian Filter (Unbiased transformed)

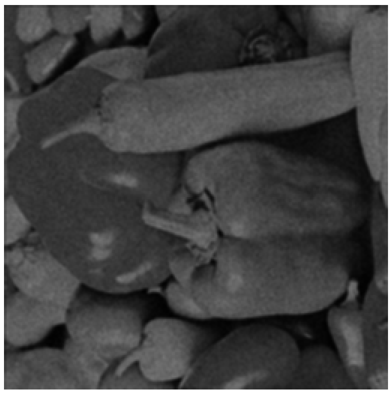
Figure 44: Denoised by Gaussian Filter (biased transformed)

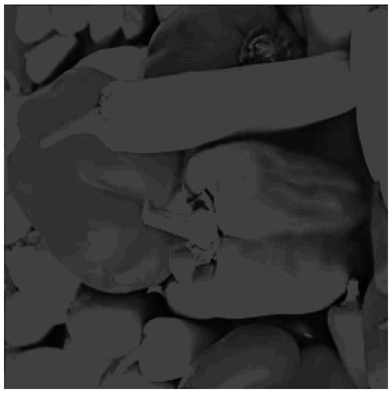
Figure 45: Denoised by BM3D Filter (biased transformed)

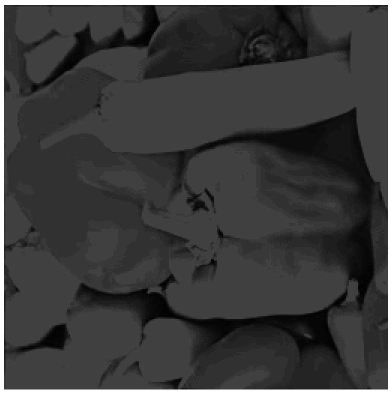
Figure 46: Denoised by BM3D Filter (unbiased transformed)

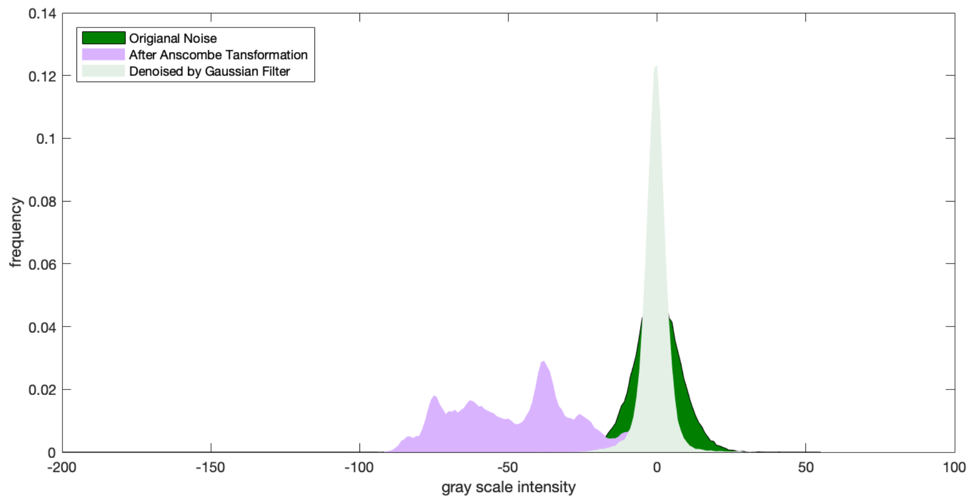

Figure 47: Noise Distribution

unbaised transformation, the PSNR value of unbaised transformation is higher. On the other hand, the biased transformation image becomes 'darker' than unbaised transformation. The brightness of the biased transformation image is somewhat distorted. Figure 48 is the Noise distribution after filtering by Gaussian filter. It shows that the noise mean of biased invers transformation is slight move to the negative axis. On the contrast, the mean of unbiased inverse transformation is apprpximately equal to 0.

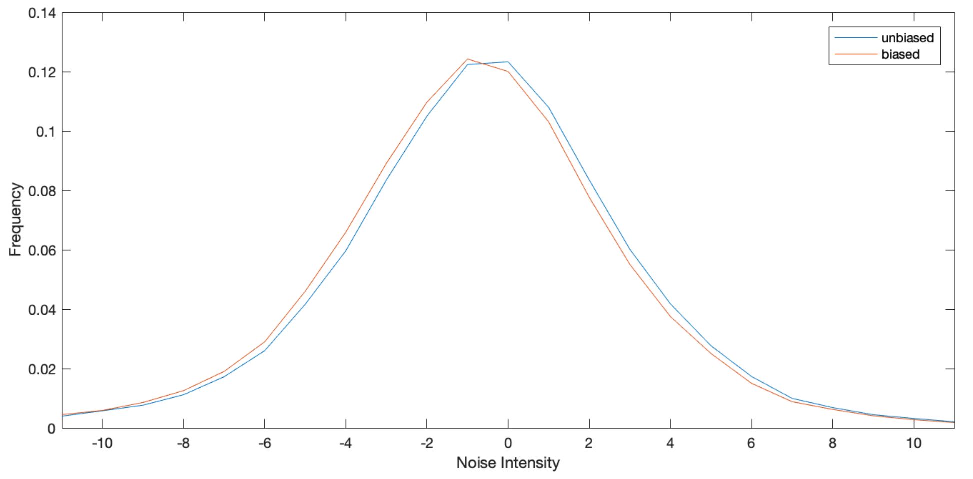

Figure 48: Noise Distribution