Homework One Report

Zifan Wang

January 22, 2019

1 Problem 1: Image Demosaicing and Histogram Manipulation

1.1 A. Bilinear Demosaicing

1.1.1 Abstract and Motivation

Color images are captured by a color filter array (CFA) in the camera. However, the CFA is made by CMOS sensors and each CMOS sensor is only able to capture one color (R,G,B). In order to obtain color images, people developed Bilinear Demosaicing method to reconstruct other two colors at a pixel location according to its nearby pixels' values.

1.1.2 Approach and Procedures

Bilinear demosaicing method is based on bilinear interpolation. The missing color value at each pixel is approximated by averaging its two or four adjacent pixels of the same color. For example, a green pixel is surrounded by two red pixels and two blue pixels. The red value at this green pixel's location can be calculated by averaging two surrounding red pixels' value. After constructing missing colors in each pixel, a color image can be constructed. For boundary pixels, a image is necessary to be extended to reconstruct missing colors.

1.1.3 Result

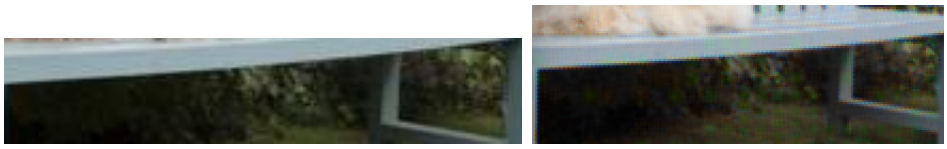
(a) Original Image (b) After Bilinear Demosaicing

Figure 1: Bilinear Demosaicing Result

The figure 1(a) is the output image of CFA. The figure 1(b) is the reconstructed color image by the Bilinear Demosaicing method.

1.1.4 Discussion

The short comings of bilinear demosaicing are zipper effect and false color effect. The zipper effect is unnatural changes of intensities over a number of neighboring pixels around edges. The figure 2(b) shows that the zipper effect happens in the edge of bench. Improper averaging of surrounding color values across edges is responsible for causing this effect.



(a) Edge in Original Image (b) Edge after Bilinear Demosaicing

Figure 2: Bilinear Demosaicing Result

The false color effect refers to wrong color interpolation. The figure 3(b) shows that bilinear demosaicing constructs wrong color to the flower. This effect is caused by inconsistency among the three color spaces. This inconsistency results color intensity changes.

(a) True Color in Original Image (b) False Color after Bilinear Demosaicing

Figure 3: Bilinear Demosaicing Result

One idea to improve the demosaicing performance is to use more neighboring pixels to recover missing colors.

1.2 B. Malvar-He-Cutler (MHC) Demosaicing

1.2.1 Abstract and Motivation

The 'zipper effect' and 'false color effect' casued by bilinear demosaicing affect the reconstructed image's quality. The Malvar-He-Cutler (MHC) is developed to reduce those two effects.

1.2.2 Approach and Procedures

To imporve bilinear demosaicing, Malvar et al added a second order cross channel correction term to the bilinear demosaicing reuslt. For example, a green component at a red

pixel location is given by:

$$G(i,j) = G^{bl}(i,j) + \alpha\Delta_R(i,j)$$

where $G^{bl}(i,j)$ is the bilinear interpolation result from bilinear demosaicing, α is a weight factor, and $\Delta_R(i,j) = R(i,j) - \frac{1}{4}(R(i-2,j) + R(i+2,j) + R(i,j-2) + R(i,j+2))$.
Red component at green pixels location:

$$R(i,j) = R^{bl}(i,j) + \beta\Delta_G(i,j)$$

Red component at blue pixels location:

$$R(i,j) = R^{bl}(i,j) + \gamma\Delta_B(i,j)$$

The weight factors $\alpha = \frac{1}{2}, \beta = \frac{5}{8}, \gamma = \frac{3}{4}$ control color corrections.
In addition, same to the bilinear demosaicing, the image extension is necessary before doing MHC algorithm due to boundary pixels.

1.2.3 Result

Figure 4: Color Image after MHC Reconstruction

The figure 4 shows the color image after MHC color reconstruction.

1.2.4 Discussion

(a) Original Image (b) Color Image after Bilinear Demosaicing (c) Color Image after MHC Reconstruction

Figure 5: Color Image Reconstruction

The images in the figure 5 show original color image, color image after bilinear demosaicing, and color image after MHC. By comparing figure 5(b) and 5(c) with 5(a), the MHC linear demosaicing has a better performance at edges and details than the bilinear demosaicing. The MHC method reduce the zipper effect and false color effect. For example, the zipper effect at bench is reduced and the color of flower is correct. The second order correction term in the MHC method preserves true color at the edge where colors change rapidly and smoothes colors change.

1.3 C. Histogram Manipulation

1.3.1 Abstract and Motivation

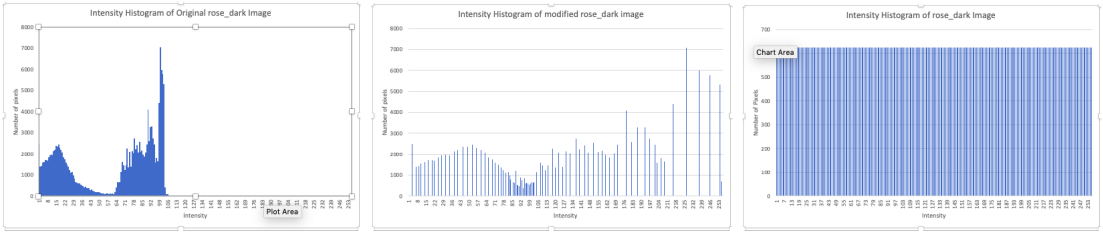
Contrast adjustment is important to camera imaging pipeline. This adjustment is able to make bright images look dark and dark images look bright. There are two methods can be used to modify the luminous intensity in order to make images more distinguishable. The first one is the transfer function based (TFB) histogram equalization method and the another is the cumulative probability based (CPB) histogram equalization method.

1.3.2 Approach and Procedures

A gray-scale digital image can make a intensity histogram according to its' intensity level from 0 to 255. The distribution function associates with each intensity level and the number of pixels with this intensity. The Transfer function based histogram equalization method normalizes input images histogram and maps to the the output images which has uniform intensity histogram.
To do the cumulative probability based histogram equalization method, a 1-D array with size 1*256 is created. Each position of array indicates number of pixels in a specific intensity. Then, distribute equal number of pixels to each intensity. The number of pixels is defined by $N_w * N_h/256$, where N_w is the width of the image and N_h is the height of the image.
After two processes, a distinguishable photo will be generated.

1.3.3 Results

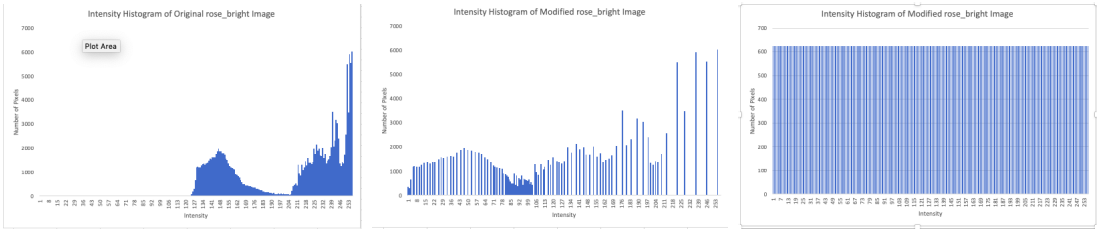
Histogram:



(a) Original Histogram (b) Output Histogram t-f-b (c) Output Histogram c-p-b

Figure 6: Histogram of Rose Dark Image

The figure 6 shows intensity histogram of rose dark image before and after histogram equalization. From the above figure, it is easy to notice that the intensity histogram spread out.

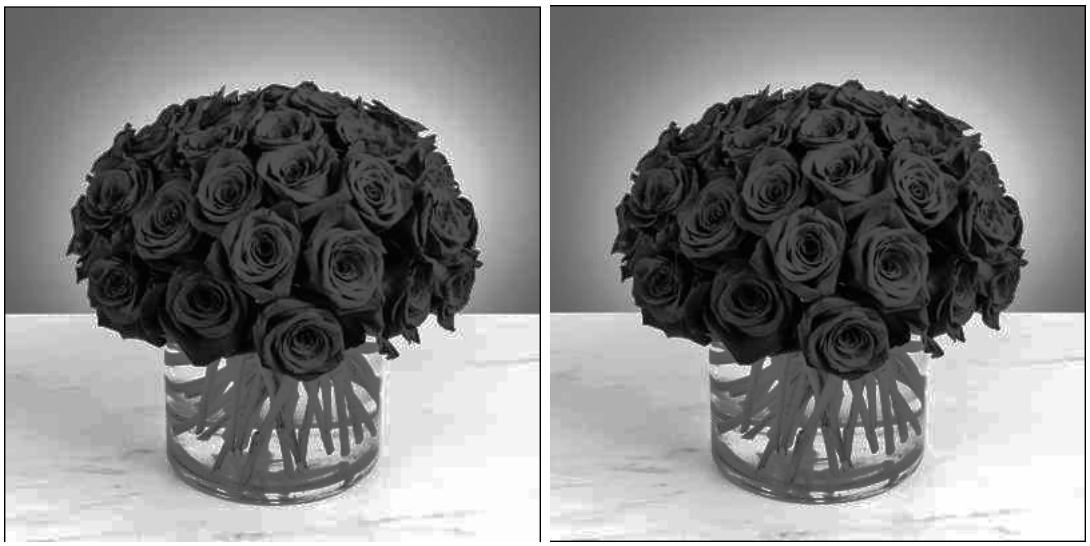


(a) Original Histogram (b) Output Histogram t-f-b (c) Output Histogram c-p-b

Figure 7: Histogram of Rose Bright Image

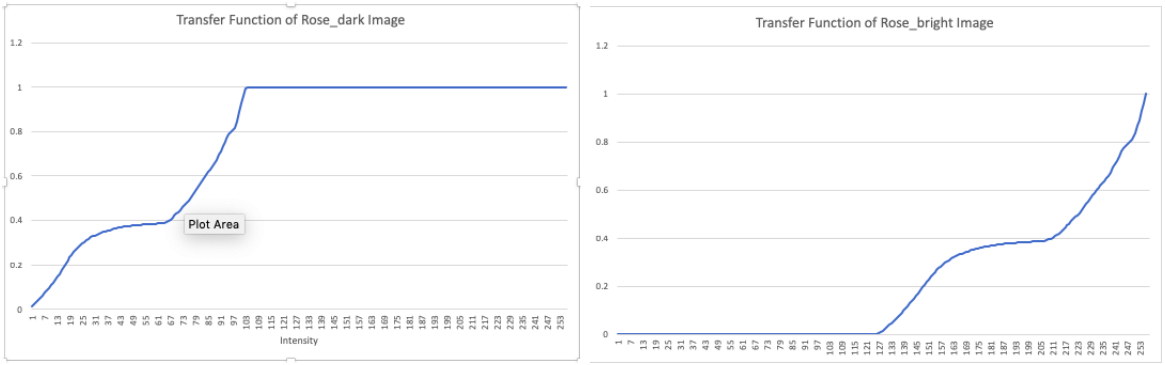
The figure 7 shows intensity histogram of rose bright image before and after histogram equalization. From the above figure, it is easy to notice that the intensity histogram spread out. This will cause photo to look nice.

Transfer-Function-Based Histogram Equalization Method Output:



(a) Output Rose Dark Image by TFB (b) Output Rose Bright Image by TFB

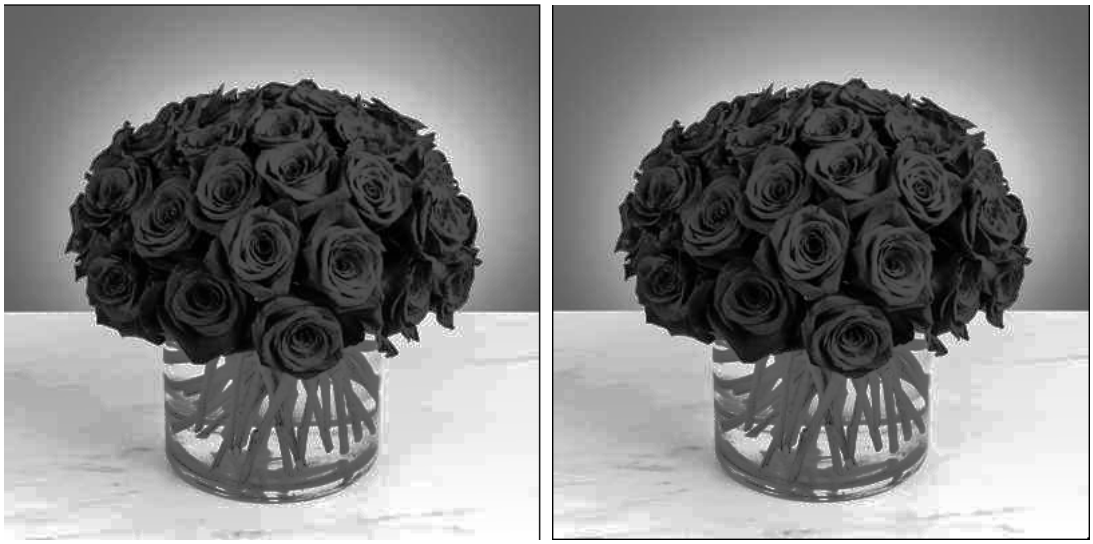
Figure 8: Output Image after Transfer Function Based Method



(a) Transfer Function of Rose Dark Image (b) Output Rose Bright Image by TFB

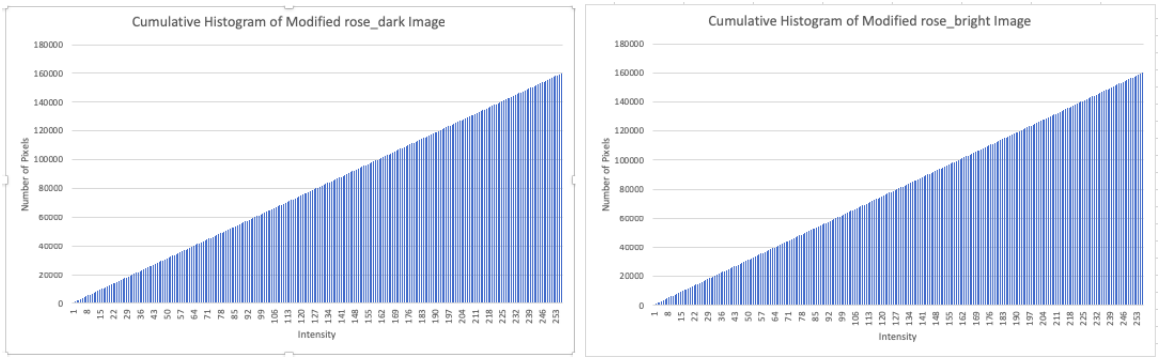
Figure 9: Transfer Function after Transfer Function Based Method

Cumulative-Probability-Based Histogram Equalization Method Output:



(a) Output Rose Dark Image by CPB (b) Output Rose Bright Image by CPB

Figure 10: Output Image after Cumulative Probability Based Method



(a) Cumulative Histogram of Rose Dark Image (b) Cumulative Histogram of Rose Bright Image

Figure 11: Cumulative Histogram after Cumulative Probability Based Method

1.3.4 Discussion

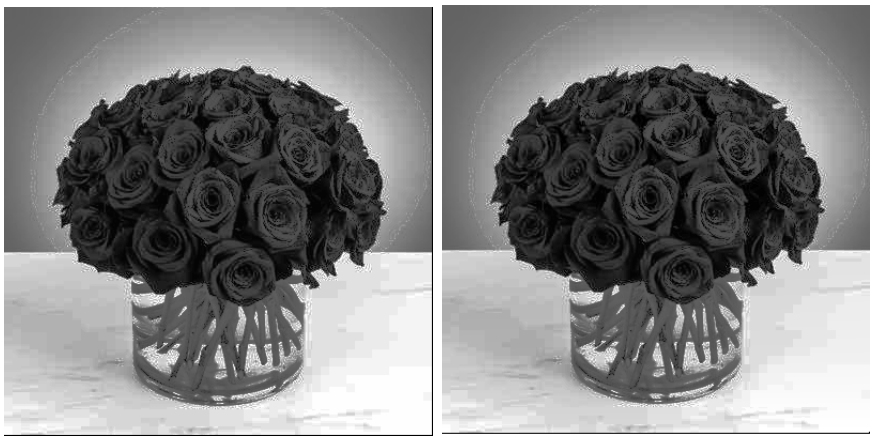
Observations on Two Enhancement Results:
Output images look the same after processing by two methods. The degree of contrast for the resulting images are the main difference between TFB equalization and CPB equalization. The CPB method distributes the equal number of pixels to each intensity value rather than different number of pixels to each intensity value. Therefore, the output images of CPB processing will have stronger contrast than the output images of TFB processing.

(a) Rose Dark Image (b) Rose Bright Image (c) Rose Image

Figure 12: Original Images

Comparing with images in the figure 12(a) and the figure 12(b), the images in the figure 8 and the figure 10 improve a lot after two contrast adjustment method. However, the the figure 12(c) looks better than the modified images, especially in the top dark region. One way to improve the current result is that taken the mean brightness of an input image during TFB processing. This method can preserve the mean brightness of the input images.

Rose Mix Implementation: The following figure 13 shows output images of rose mix after two methods. The figure 13(a) and the figure 13(b) look similar to resulting images



(a) Output Image of TFB (b) Output Image of CPB

Figure 13: Rose Mix

in the previous part. The main difference is that there is a distinguishable color change line in the top region of the images.

2 Problem 2: Image Denoising

2.1 Gray-Level Image

2.1.1 Abstract and Motivation

Taken by digital cameras, images will pick up noises from a variety sources. There are two types of noise: salt and pepper noise and Gaussian noise. In salt and pepper noise, pixels in the image are very different in color or intensity from their surrounding pixels. When viewing, images shows dark and white dots. In Gaussian noise, pixels in the image will be changed a small amount from their original value. For aesthetic purpose, these two types of noises have to de denoised before shown to people. Four filters are introduced here to denoise noisy images: the uniform weight filter, the Gaussian weight filter, the bilateral filters, and the non-local mean filter.

2.1.2 Approach and Procedures

The uniform weight filter is defined by:

$$Y(i,j) = \frac{\sum_{k,l} I(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = \frac{1}{w1 * w2}$$

where (k,l) is the neighboring pixel location within the filter of size $w1 * w2$ centered around (i,j), I is the noisy image, and Y is the denoised image.
The Gaussain weight filter is defined below:

$$Y(i,j) = \frac{\sum_{k,l} I(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

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

where (k,l) is the neighboring pixel location within the filter of size $w1 * w2$ centered around (i,j), σ is the standard deviation of Gaussian distribution, I is the noisy image, and Y is the denoised image.
The Bilateral filter is defined by:

$$Y(i,j) = \frac{\sum_{k,l} I(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

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

where σ_c and σ_s are the spread parameters.
The non-local mean filter is defined below:

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

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

$$(I(N_{i,j}) - I(N_{k,l}))^2_{2,a} = \sum_{n_1,n_2 \in N} G_a(n_1,n_2)(I(i-n_1, j-n2) - I(k-n_1, l-n_2))^2$$

where $G_a(n_1,n_2) = \frac{1}{\sqrt{2\pi}a} \exp(-\frac{n_1^2+n_2^2}{2a^2})$, and h is a filtering parameter.
A image extension have to be applied to noisy images before using filters to denoise images. This image extension is the same as image extension in Bilinear Demosaicing and MHC Demosaicing which simply copy the pixels near the boundaries.

2.1.3 Result

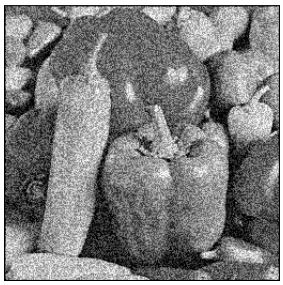


Figure 14: The Pepper Image with Noise

The figure 14 is the input image to filters and there are uniform noise in it.
The Uniform Weight Filter:
The figure 19(a) below shows denoised image after a 3*3 size uniform weight filter (UWF);

Figure 15: Denoised Image by 3*3 UWF

The Gaussian Weight Filter:
The figure 23(a) below shows denoised image after a 3*3 size Gaussian weight filter (GWF) with STD = 3;

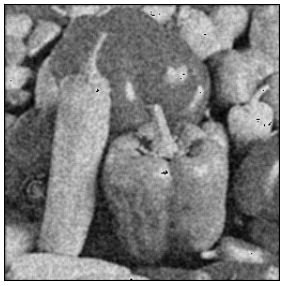


Figure 16: Denoised Image by 3*3 GWF with STD =3

The Bilateral Filter:
The figure 17 below shows denoised image after a 14*14 size Bilateral filter (BF) with $sigma_c = 5.5$ and $sigma_s = 45$;

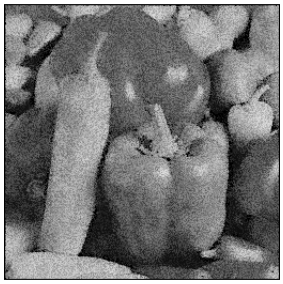


Figure 17: Denoised Image by 14*14 BF with $sigma_c = 5.5$ and $sigma_s = 45$

The Non-local Mean Filter:
The figure 18 below shows denoised image after a size non-local mean filter (NLM).

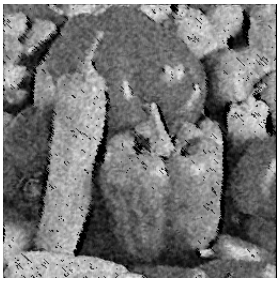


Figure 18: Denoised Image by a size 3*3 and searching window 7*7 NLM filter with $std = 45$ and $filtering - Parameter = 20$

2.1.4 Discussion

The Uniform Weight Filter:

(a) 3*3 UWF (b) 5*5 UWF (c) 7*7 UWF (d) 9*9 UWF

Figure 19: Denoised Pepper Image by UWF

The figure 19 shows that as the size of filter increases the noises decrease. However, images become more blurred after applying larger size filters. This change can be measured by the peak-to-signal-noise (PSNR) value. The following figure 20 shows a relationship between PSNR and filters' size:

Figure 20: PSNR by UWF

The Gaussian Weight Filter:

(a) 3*3 GWF (b) 5*5 GWF (c) 7*7 GWF

Figure 21: Denoised Pepper Image by GWF with STD =3

The figure 21 shows that when keeping the standard deviation of Gaussian distribution as constant, increasing the size of Gaussian filter causes images become more unreadable. The following figure 22 shows a relationship between PSNR and filters' size:

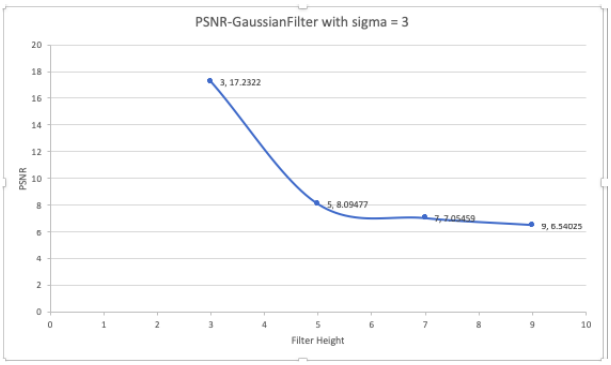
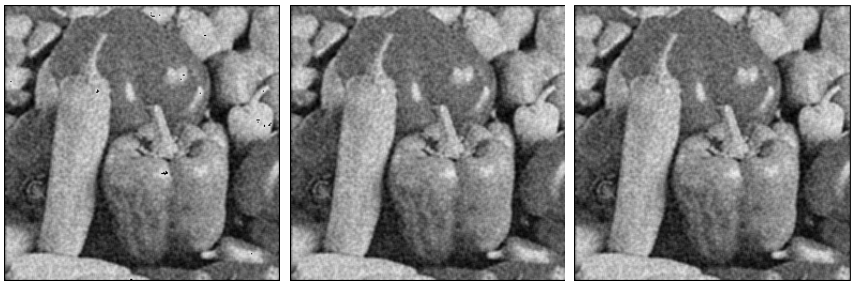


Figure 22: PSNR by GWF with STD = 3

The Guassian filter has two parameters. The one is the filter's size. Another is the standard deviation (std). The following figure 23 shows the relationship between the std with the quality of images.



(a) std=3 GWF (b) std=5 GWF (c) std=7 GWF

Figure 23: Denoised Pepper Image by GWF with size 3*3

The main improve from $std = 3$ to $std = 5$ is that the black dots in figure 23(a) are removed. The following figure 24 shows a relationship between PSNR and std value:

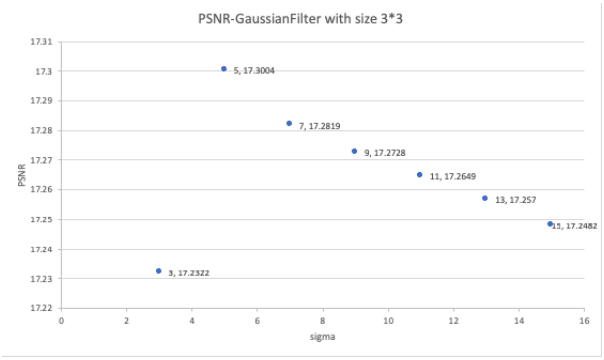
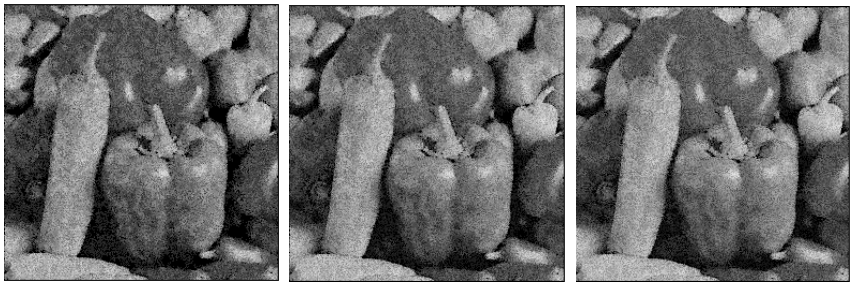


Figure 24: PSNR by GWF with size 3*3

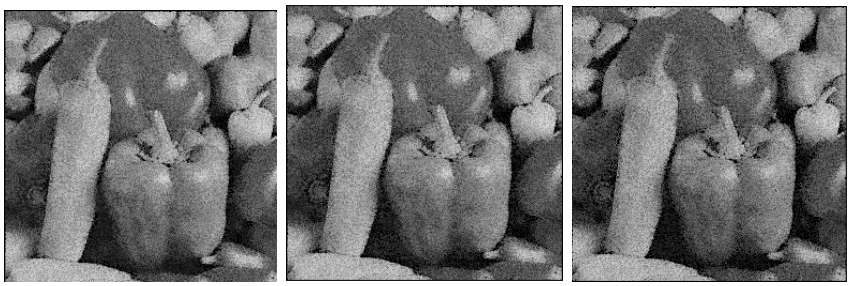
The Bilateral Filter:
The Bilateral filter has three parameters: filter's size, distance variance (σ_c), and intensity variance (σ_s). Each parameter will contribute the quality of denoised images. The figure 25 shows that keeping distance variance and intensity variance as constants, the quality of denoised images increases as filters' size increase.



(a) 6*6 BF (b) 8*8 BF (c) 10*10 BF

Figure 25: Denoised Pepper Image by BF with $\sigma_c = 3.5$ and $\sigma_s = 45$

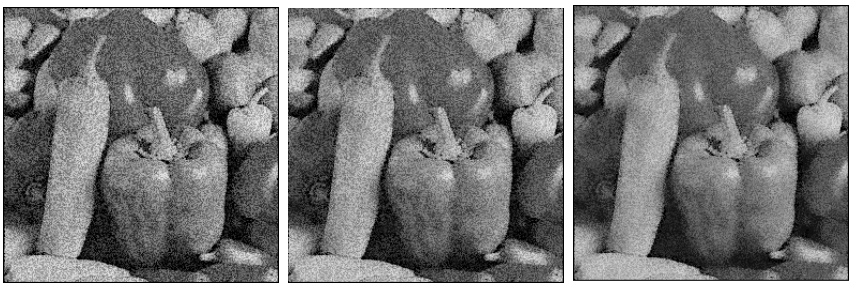
The figure 26 shows that keeping filter size and intensity variance as constants, the quality of denoised images increases as distance variance increases. However, the images will be blurred when distance variance increases.



(a) $\sigma_c = 7$ BF (b) $\sigma_c = 40$ BF (c) $\sigma_c = 50$ BF

Figure 26: Denoised Pepper Image by BF with $Size = 14 * 14$ and $\sigma_s = 45$

The figure 27 shows that keeping filter size and distance variance as constants, the quality of denoised images increases as intensity variance decreases.



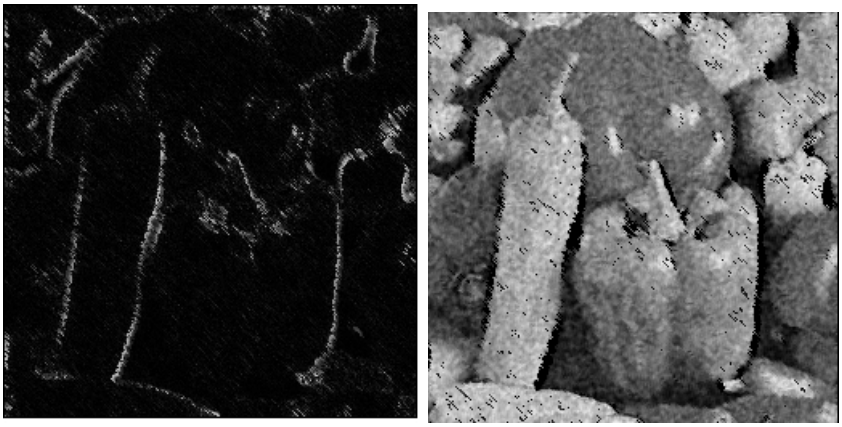
(a) $\sigma_s = 25$ BF (b) $\sigma_s = 35$ BF (c) $\sigma_c = 50$ BF

Figure 27: Denoised Pepper Image by BF with $Size = 14 * 14$ and $\sigma_c = 7$

In conclusion, bilateral filter is better than uniform weight filter and Gaussian filter. Compare to Gaussian filter, Bilateral filter's weights depend on Euclidean distance of pixels and the difference on the pixel values. This will make it preserve sharp edges and become less blurred.

The Non-local Mean Filter:

The following figure 28 compares the filter size and the searching windows size:

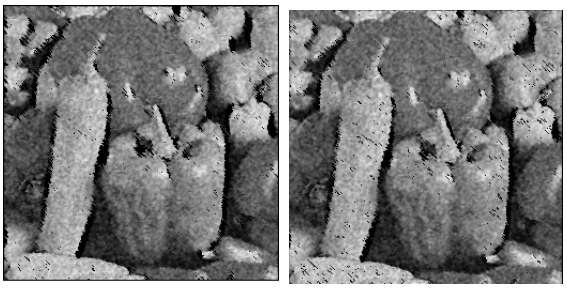


(a) Small Searching Window Size (b) Small Filtering Size

Figure 28: Denoised Pepper Image by NLM Filter with Different Filter Size

The result shows that it is better to make filter size smaller than the searching window

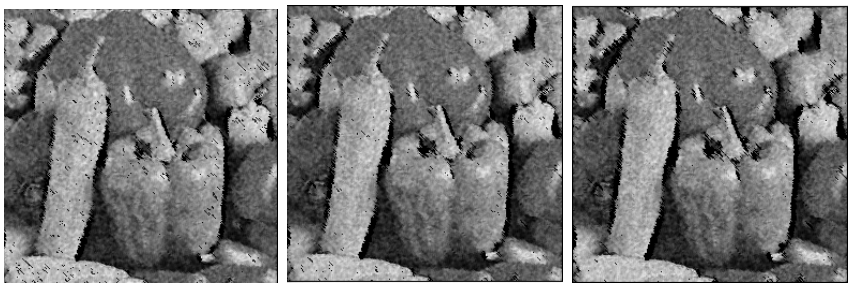
size. For non-local mean filter, the smaller filter size will make denoised image look better. The following figure 29 compares the standard deviation value of Gaussian distribution with images' qualities:



(a) STD = 40 (b) STD = 45

Figure 29: Denoised Pepper Image by NLM Filter with Different STD

The following figure 30 compares the filtering parameter with images' qualities:



(a) fp = 20 (b) fp = 19 (c) fp=18

Figure 30: Denoised Pepper Image by NLM Filter with Filtering Parameter

In conclusion, non-local mean filter is an advanced filters to remove uniform noise. It performs better than the uniform weight filter and Gaussian filter.

2.2 B. Color Image

2.2.1 Abstract and Motivation

Noisy color images contain impulse noises and uniform noises. To remove noises in color images, filters must be used in three color channels.

2.2.2 Approach and Procedures

Since the noisy rose colored image contains impulse noises and uniform noises, two types of filters have to be used in order to remove both type of noises. Impulse noises can be removed by median filter and uniform noises can be removed by all filters implemented in section 2.1.

2.2.3 Result

The following figure is the result of first applying median filter to noisy image and then applying uniform filter.

(a) UWF (b) GF (c) BF

Figure 31: Denoised color rose image

2.2.4 Discussion

All three color channels contain both impulse noises and uniform noises. The figure 32 below shows the denoised image only remove two channels' impulse noises:

(a) BG (b) BR (c) GR

Figure 32: Denoised color rose image with two channels' impulse noises removed

The figure 32 clearly shows that when remove two channels' of impulse noises the unmodified channel's impulse noises will show in the graph. As a result, there are impulse and uniform noises in all color channels.
After generating the figure 31, calculate the PSNR values between denoised image and original image without noise. The uniform weight filter generates the figure 31(a) and returns 23.7342 PSNR value. The Gaussian filter generates the figure 31(b) and returns 21.5509 PSNR value. The Bilateral filter generates the figure 36(a) and returns 15.8158 PSNR value. As a result, Bilateral filter performs will in denoising color image. The following figure 33 compares the order of applying Bilateral filter:

(a) BF first, MF second (b) MF first, BF second

Figure 33: Different order of using Bilateral Filter

Applying Bilateral filter first, will return 18.7182 PSNR value. However, applying medium

filter only returns 15.8158 PSNR value. The impulse noises keep adding up during the Bilateral filter filtering the noisy image, because Bilateral filter cannot removes impulse noises.
The showing figures above do not remove noises satisfactorily. One way to improve the result is that add Bilteral filter's algorithm to non-local mean algorithm. Here is the equation:

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

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

$$(I(N_{i,j}) - I(N_{k,l}))_{2,a}^2 = \sum_{n_1,n_2 \in N} G_a(n_1,n_2)(I(i-n_1, j-n2) - I(k-n_1, l-n_2))^2 + \frac{n_1^2+n_2^2}{2\sigma_s^2}$$

where $G_a(n_1,n_2) = \frac{1}{\sqrt{2\pi}a} \exp\left(-\frac{n_1^2+n_2^2}{2a^2}\right)$, h is a filtering parameter, and σ_s is distance variance. Adding a distance difference term will help non-local mean algorithm decide whether the neighborhood regions are similarly to the computed region. This will improve edges' reconstruction and result less blurred.

2.3 C. Short Noise

2.3.1 Abstract and Motivation

When taking photos in the dark environment, shot noises will be generated. This type of noise originates from the discrete nature of electric charge. There are two method to remove short noises: Gaussian filter and BM3D filter.

2.3.2 Approach and Procedures

The way to implement Gaussian filter is similarly to the section 2.1. For short noise, a size 5 by 5 Gaussian filter with std = 7 is used.
The BM3D method involves grouping, collaborative filtering, and aggregation. First, images fragments are group together based on similarity. Then, apply filtering in each fragments group. Last, aggregate image fragments back to denoised images.

2.3.3 Result

The following figure 34 is denoised image generated by Gaussian filter with PSNR = 17.1757:

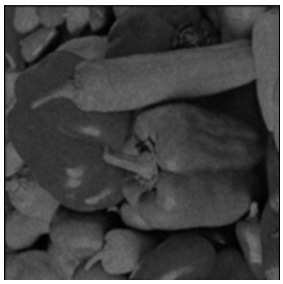


Figure 34: Gaussian Filter

The following figure 35 is denoised image generated by BM3D filter with PSNR = 38.577 dB.

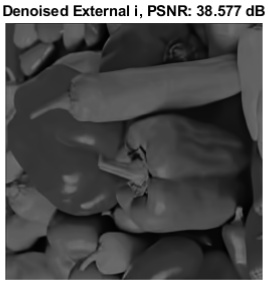
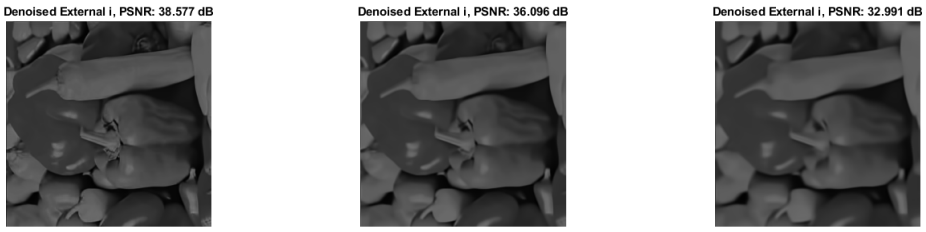


Figure 35: BM3D

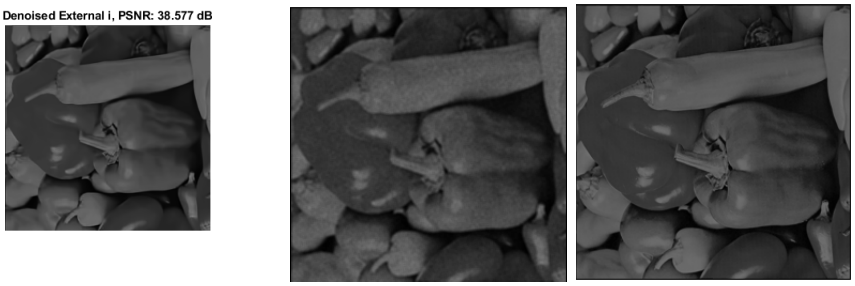
2.3.4 Discussion



(a) BM3D10 (b) BM3D20 (c) BM3D30

Figure 36: BM3D with Different STD

The figure 36 shows that the image will be more blurred as the std value in the BM3D filter increases.



(a) BM3D10 (b) GFdark (c) Original Image

Figure 37: Filter Comparison

The figure 37 denoised images after two types of filters with the original image. From figures and PSNR values, the BM3D filter performs better than the Gaussian filter.

2.4 Reference

[1] Makitalo, Markku, and Alessandro Foi. "Optimal inversion of the Anscombe transformation in lowcount Poisson image denoising." IEEE transactions on Image Processing 20.1 (2011): 99-109.
[2] BM3D Algorithm: http://www.cs.tut.fi/ foi/GCF-BM3D/
[3] Filter Algorithm: https://courses.uscden.net/d2l/le/content/15358/viewContent/243727/View