Problem 1

1.1 Motivation

A digital image scaling (reduced or enlarged) can be realized via bilinear interpolation and one famous painting from Vincent van Gogh is to be enlarged by this method. Another topic is about CFA. It is usually used in digital camera to capture color image. Reconstruction of a full color image from incomplete color samples from Bayer array is realized in this problem.

1.2 Approach

Image resizing: The input image is 512x512 and the target size is 650x650. First of all, transform the 650x650 pixels to 512x512 coordinate system. Then use bilinear interpolation to assign the value for each pixel in the output image.

Color reconstruction: Two methods are used, bilinear demosaicing and MHC linear demosaicing. For the marginal pixels in the image, the image is extended by reflection.

All programming use C++.

1.3 Results

(a) image of size 512x512 (b) image of size 650x650

Figure 1: Original and resized The-starry-night images

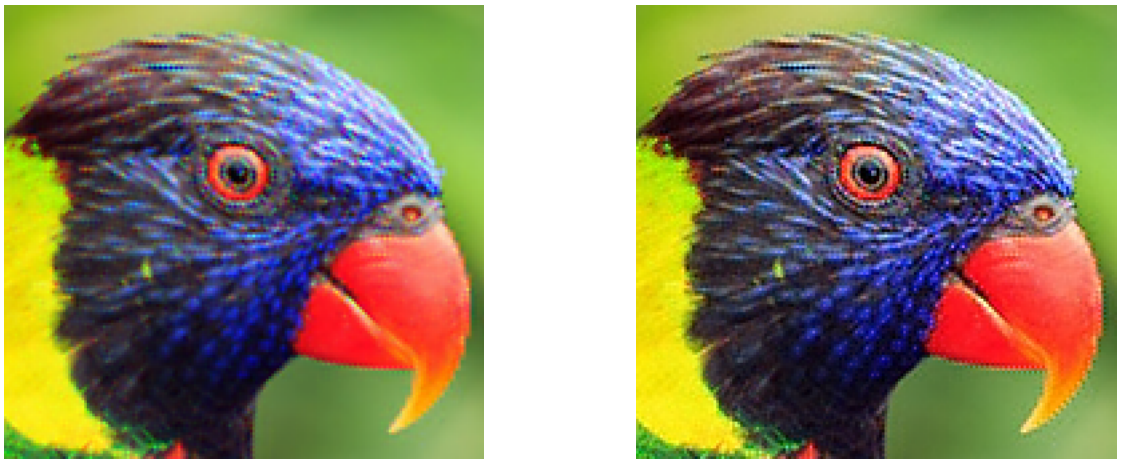
(a)parrot CFA image (b) bilinear demosaicing (c) MHC demosaicing

Figure 2: Demosacing results of parrot image

1.4 Discussion

(1)

Bilinear demosaicing is **easy to be implemented with less computation**. It is good at **previewing an image**, especially for images which have **simple or less various color**. However, this method performs badly **at edges or details in the image and can bring artifacts and alias in this region**. In Figure 3, there exists more blurring in (a) compared with (b). According to sampling theory, when applying bilinear interpolation and spatially sampling the pixels, any pixel (different color channel) frequency above the **Nyquist frequency** will cause aliasing when image is reconstructed. So artifacts appear in this method.



(a) bilinear demosaicing result (b) MHC demosaicing result

Figure 3: Local details of parrot image

(2)
MHC linear demosaicing **performs better at edges or details** in the image compared with Bilinear demosaicing. For example, in Figure 3, the feathers around the parrot's eyes in (b) is more vivid than that in (a). Since MHC use **2^{nd} order correction term** to Bilinear method, it preserves the true color, especially in the region where colors change rapidly. For instance, when estimating a red component, if green domains in this area, the correction term can make the red value smaller (also for blue value) to preserving the details and edges between different colors.

(3)
One important issue I meet when applying MHC is that since the correction term use plus or minus coefficients, it will make some values below 0 or above 255. The special values result in wrong color in the image. I deal with these values by assigning 0 to the below 0 value and 255 to the above 255 value.

Problem 2

2.1 Motivation

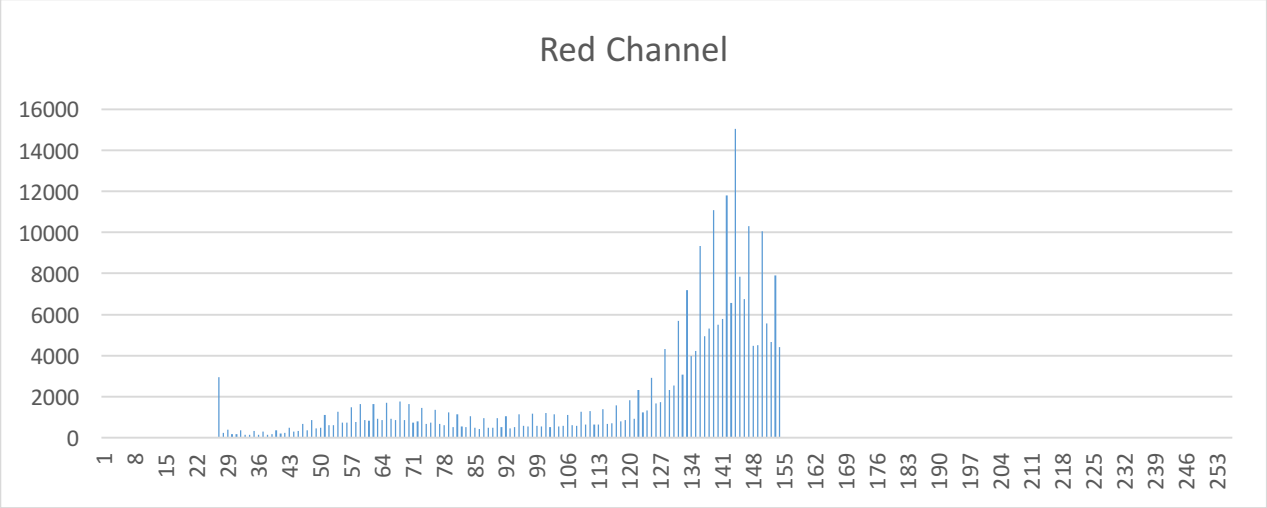
Contrast is key factor to make the objects in the image distinguishable. Two methods are used to change the luminance in order to make the jet in image more distinguishable. Image filtering can bring lots of effect to the original image and oil painting is one of that effects. By changing the parameters, the effect differs.

2.2 Approach

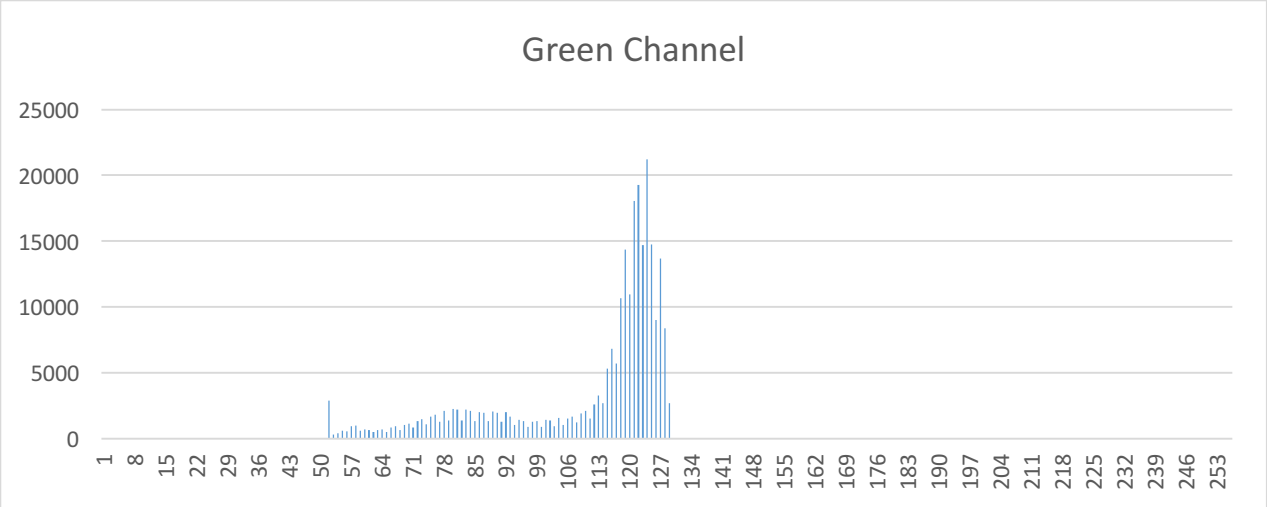
Contrast: transfer-function based histogram equalization and cumulative-probability-based histogram equalization are used. The key idea is to rearrange the pixel values to change the luminance.

Image filtering: Oil painting effect is realized by transfer 256 colors to 64 colors and then assign the most frequent color to a group of pixels in an area to achieve oil painting effect.

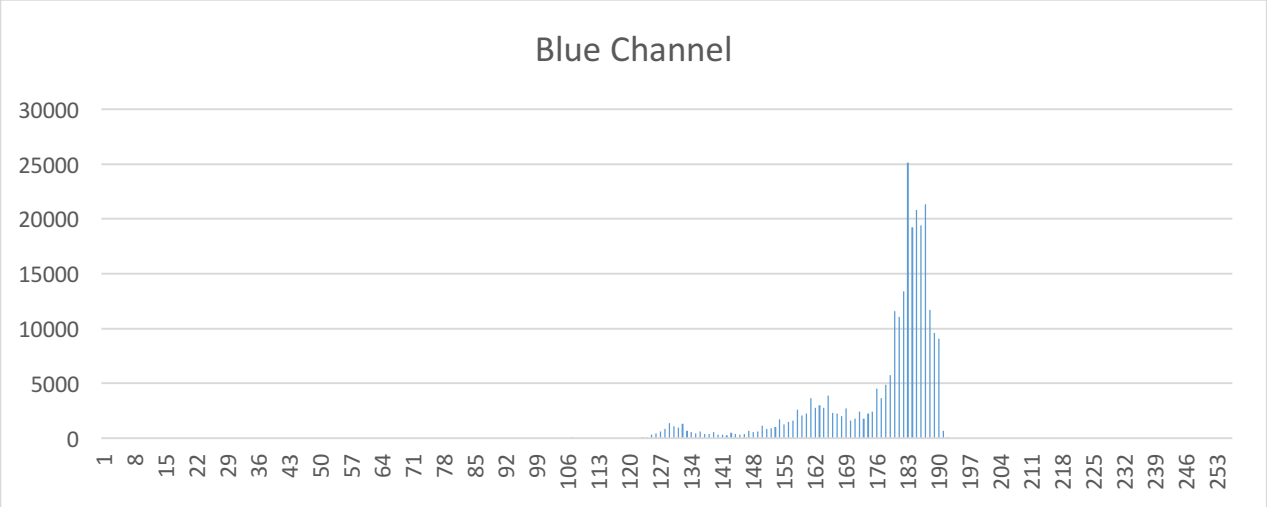
2.3 Results



(a) histogram of the red channel



(b) histogram of the green channel



(c) histogram of the red channel

Figure 4: Histograms of the RGB channels of the original image

(a) cumulative-probability-based

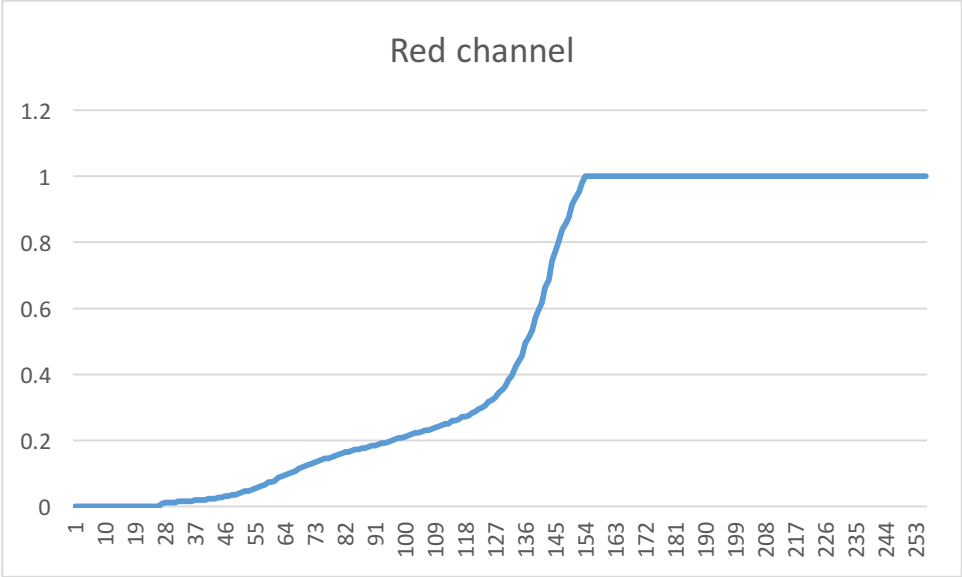
(b) transfer-function-based

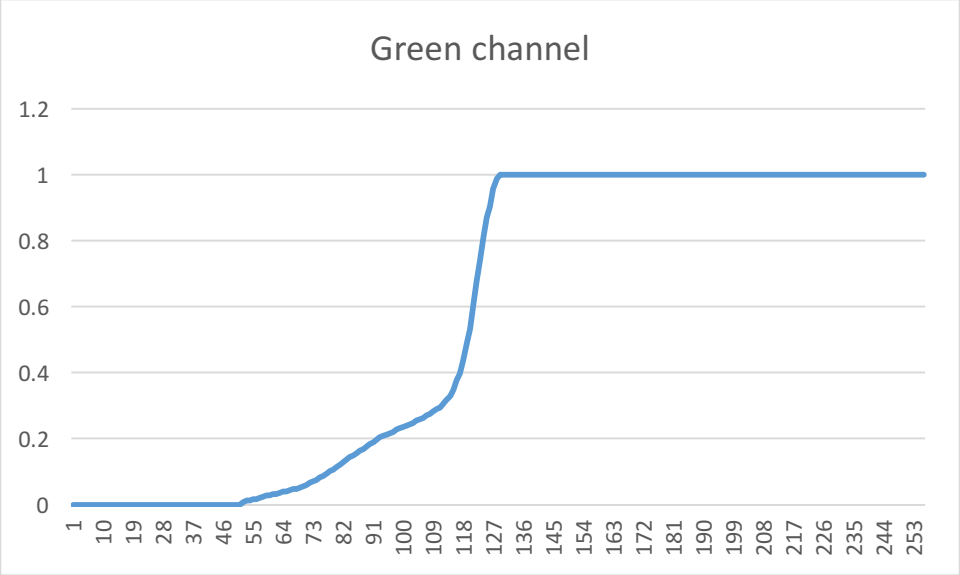
(c)original jet image

Figure 5: resulting images based on different method

From Figure 5 above, both methods increase the global contrast compared with (c). **For (b), the contrast is better than that in (a)**, **especially the local contrast**. For example, the cloud on top of the image in (b) is better than (a). The reason is that for transfer-function method, **peaks in the histogram of pixels are widened while the valleys are compressed.** However, for cumulative method, it **ignores the "information" about pixel value**, divides all the pixels to the same size bin. It may enhance the background contrast but lose the useful information's contrast. Some inappropriate luminance still exists.

For both methods, the histogram equalization is applied on the RGB channels respectively, it will result in some **singular value**. **Applying this method on the YUV channels can improve the performance**.





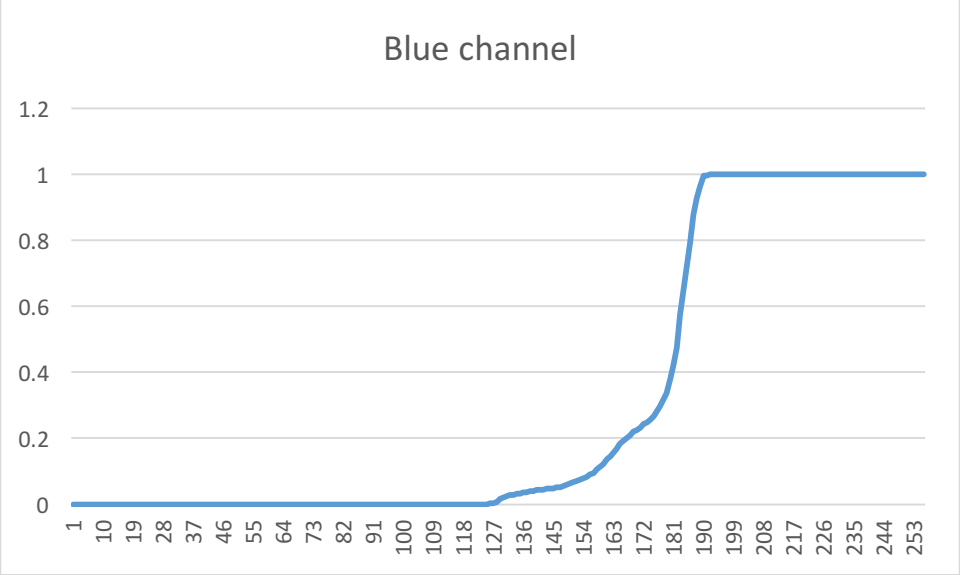


Figure 6: transfer function for RGB channels
(x-axis stands for the intensity value and the range is from 0 to 255. y-axis stands for the CDF. The output intensity should be 255*y for each pixel in Figure 4.)

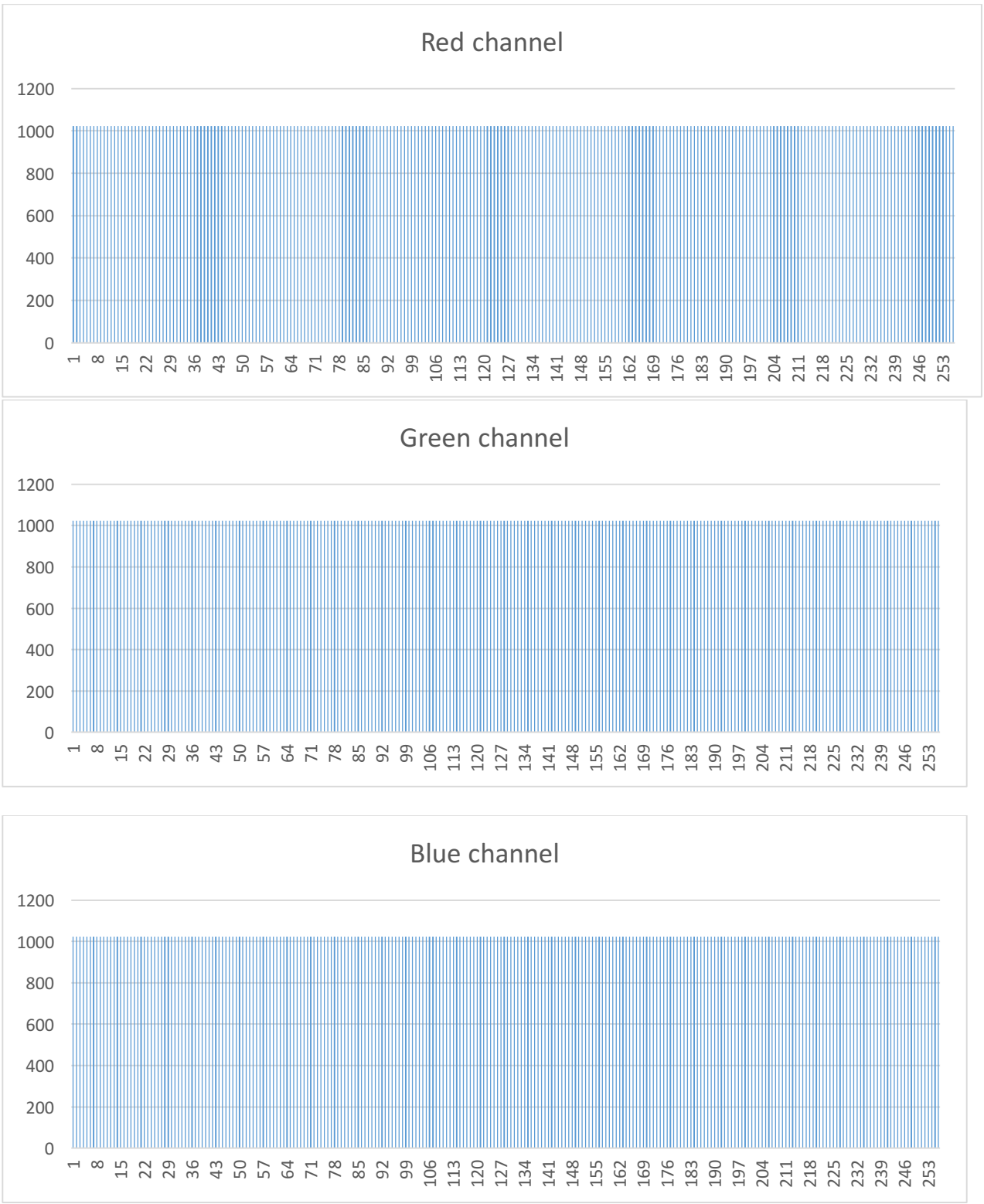


Figure 7: cumulative histogram for each channel
(x-axis is from 0 to 255)

(a)original barn image

(b)64-color barn image

(c)original coliseum image

(d)64-color coliseum image

Figure 8: 64-color version of barn and coliseum images

All pixels are integrated according to their intensity in each channel and divided into four bins with the same size. **The threshold for each bin correspond to the last pixel's intensity in the bin**. For each bin, the intensity for each pixel is the **mean value of the intensity of all pixels in this bin**.

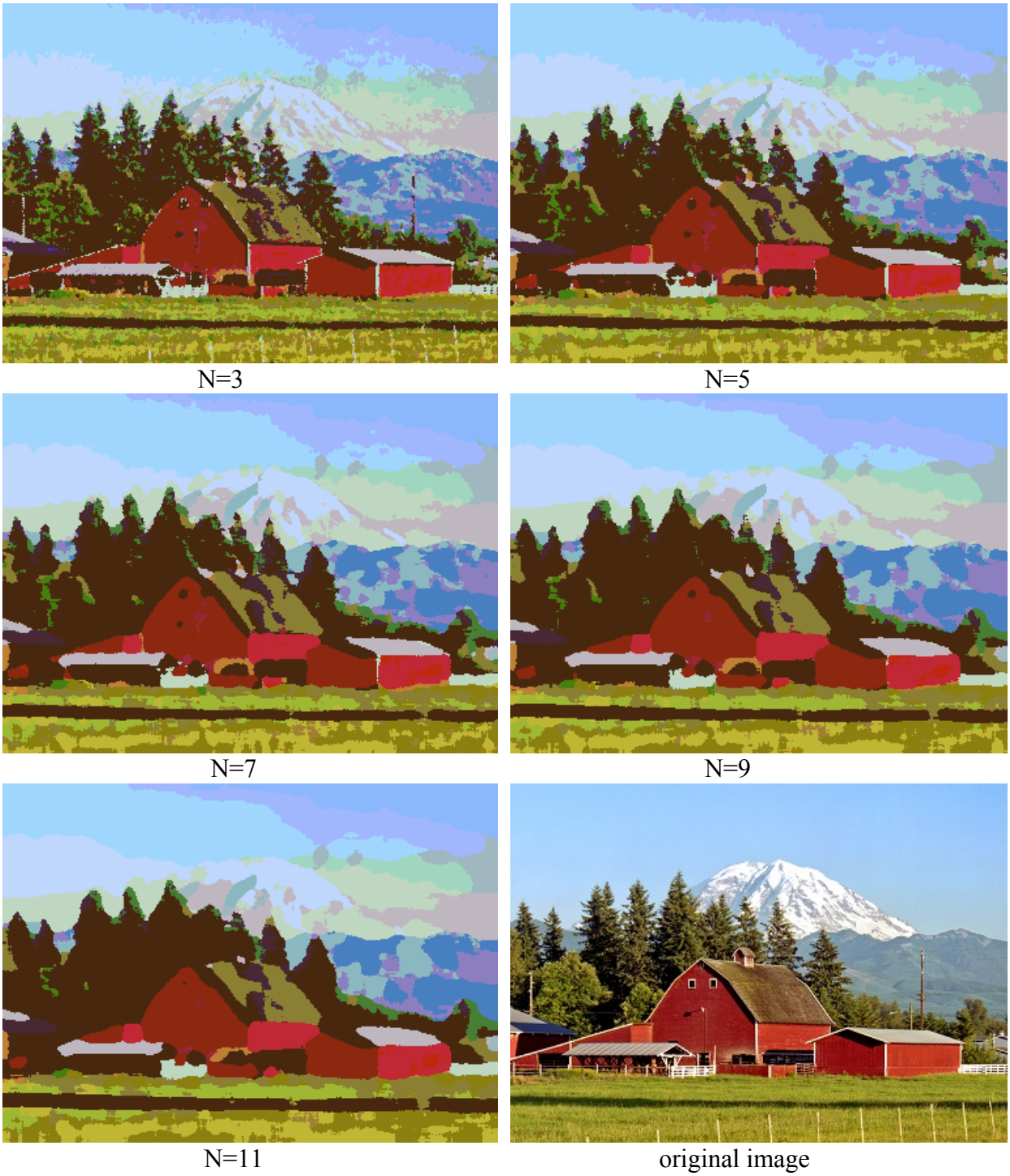


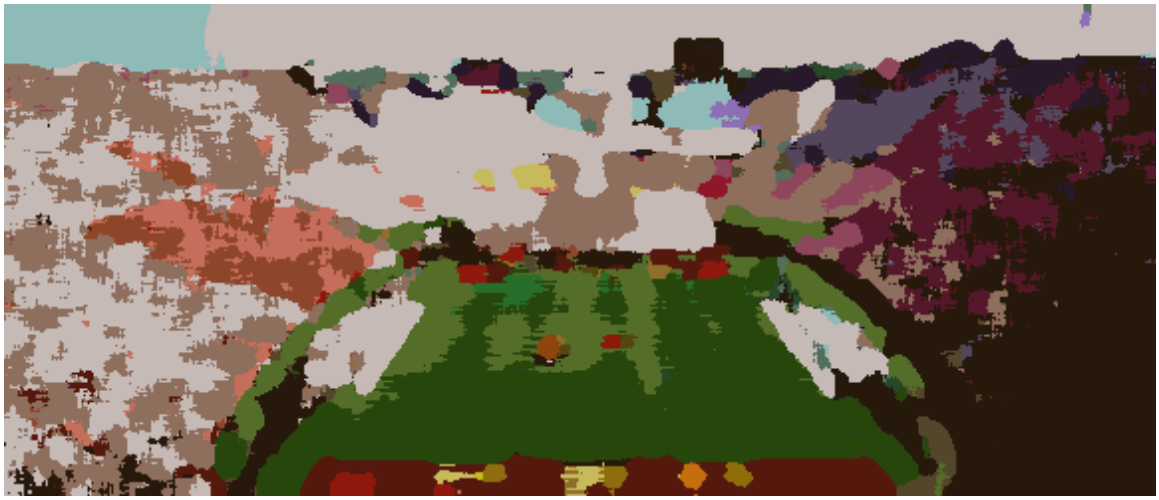
Figure 9: oil painting effect on barn image with different N

N=3

N=5

N=7

N=9



N=11

Figure 10: oil painting effect on coliseum image with different N

From Figure 9 and 10, as **N goes larger, the oil painting effect becomes more significant** and the **details in the image become less**, because a **larger window size results in larger areas with the same representation color**. Some details and edges about the objects in the image disappear.

N=5 filter, 64-color input image

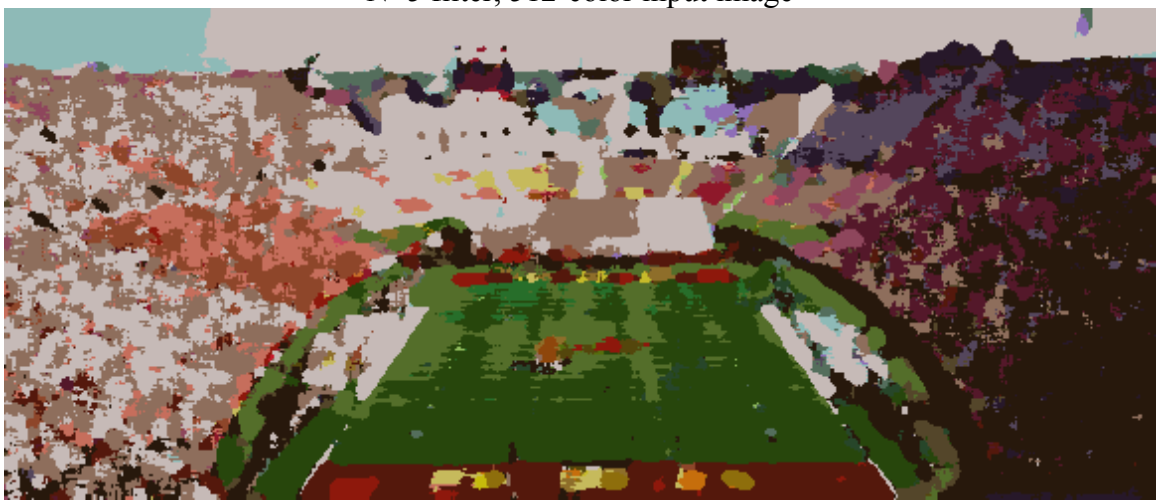
N=5 filter,512-color input image

N=7 filter, 64-color input image

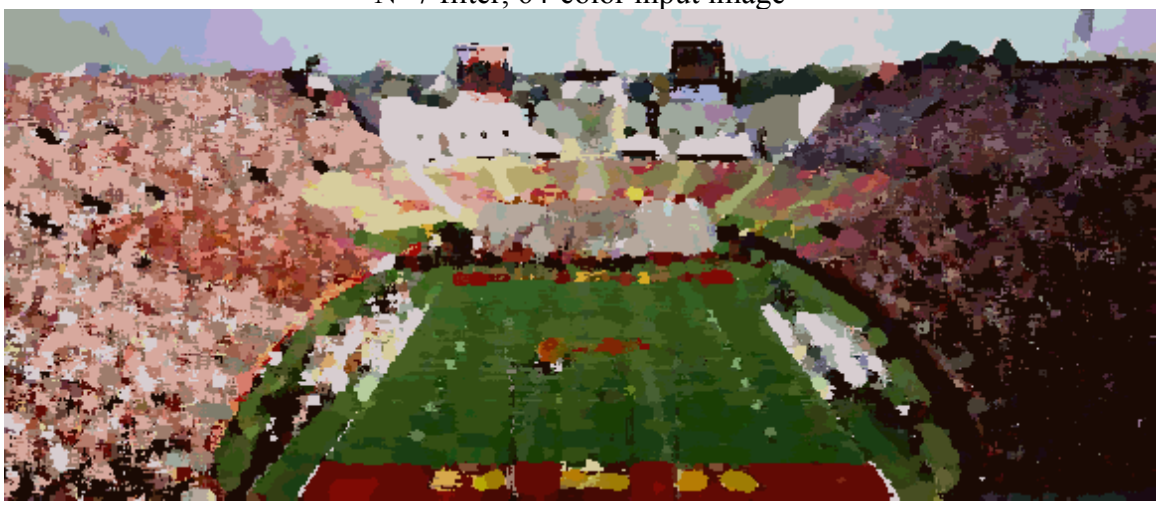
N=7 filter,512-color input image

N=5 filter, 64-color input image

N=5 filter, 512-color input image



N=7 filter, 64-color input image



N=7 filter, 512-color input image

Figure 11: oil painting effect on 512-color image (barn and coliseum)

From Figure 11, compared the oil painting effect on 512-color image with 64-color, the former one **is more colorful and has more details**. Because it has a **more abundant color set to assign the most frequent color (representation color) to an area**.

2.4 Discussion

(1) When getting the histogram of RGB channels of the image, I scan the image row by row. And for pixels belonging to each intensity, they are arranged in order of the index in the image. In cumulative method, they are assigned to bucket according to this order. But this order is not the only choice, another order leads to new distribution of luminance and the local contrast in the same parts in an image can be different.

(2) For oil painting effect, find the most frequent color as of combination of RGB values instead of finding the most frequent in RGB channels respectively.

Problem 3

3.1 Motivation

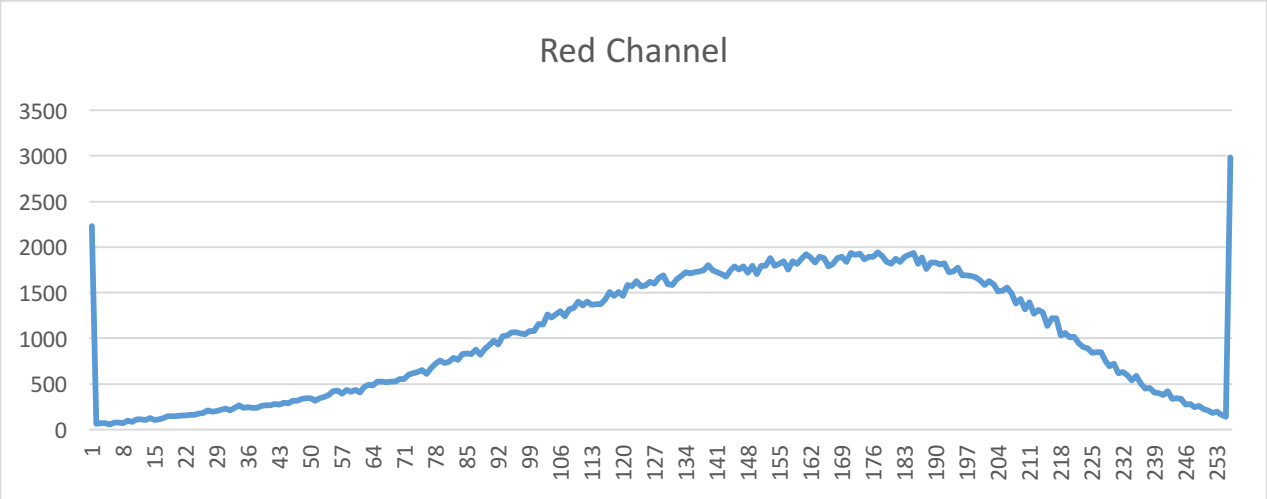
Noise prevents people from understanding the information of an image correctly. Usually, in probability and statistics perspective, noise is treated as a random error. In this problem, several methods are implemented to reduce the noise.

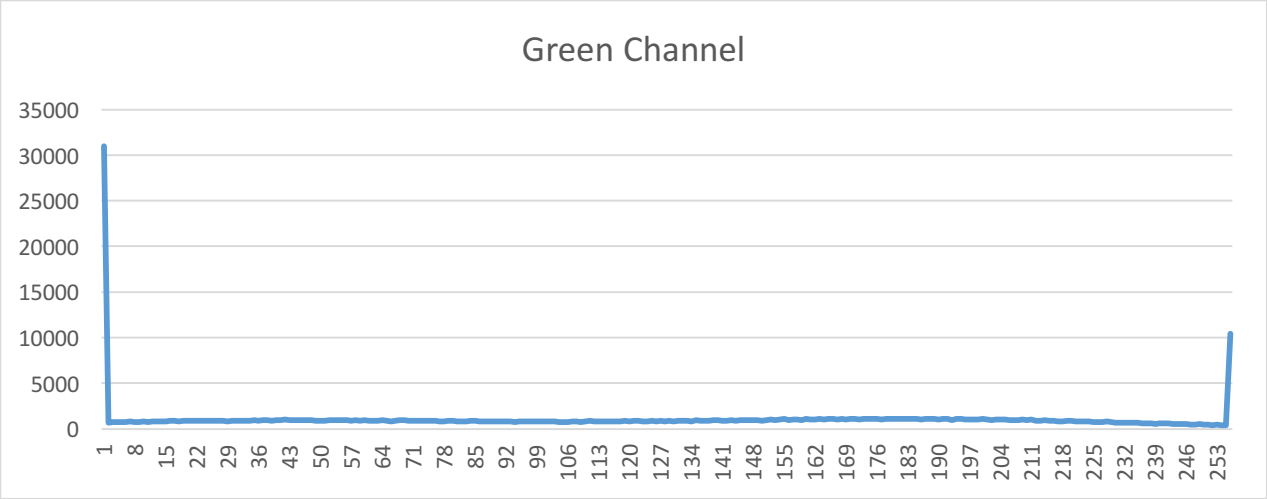
3.2 Approach

Filters involve Medium filter, Gaussian filter, Bilateral filter, and Guided filter. BM3D method is also discussed. The related parameters are provided in the 3.3.

In this problem, most results use *single-factor authentication* to compare the performance of different filters. But in some situations, single-factor authentication is violated. For example, when comparing the Gaussian filter with Bilateral filter, the sigma for intensity closeness only appears in Bilateral filter, so it violates the single-factor authentication.

3.3 Results





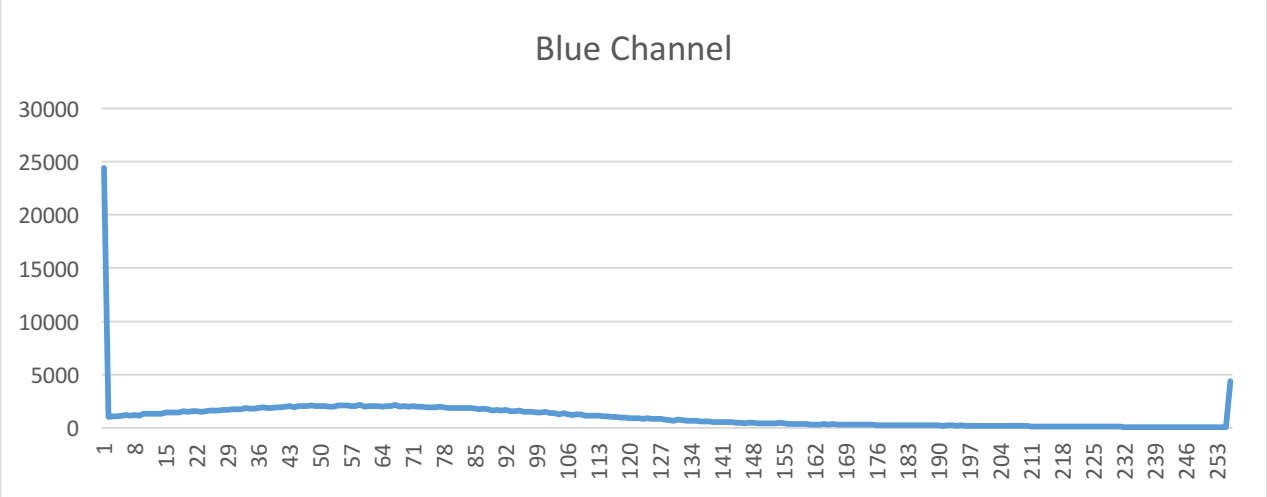
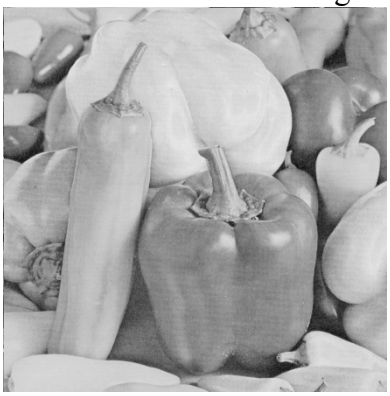
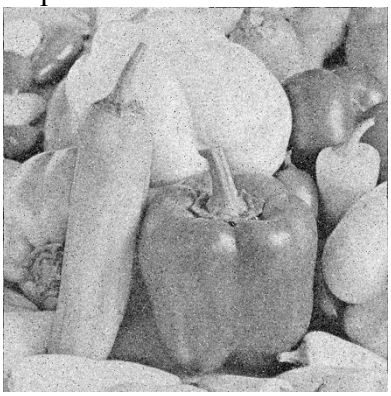


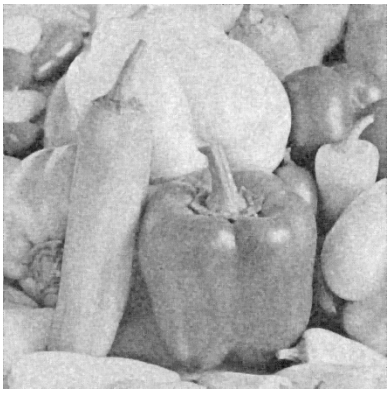
Figure 12: pixels distribution on different channel



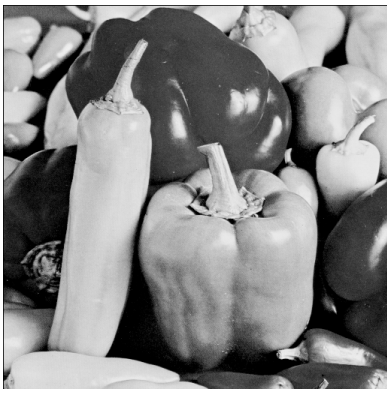
original R channel

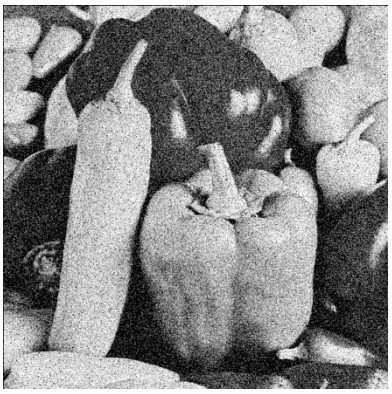


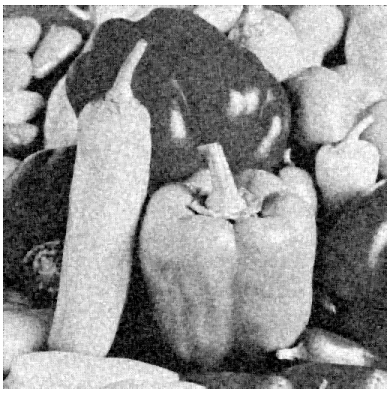
R channel with noise

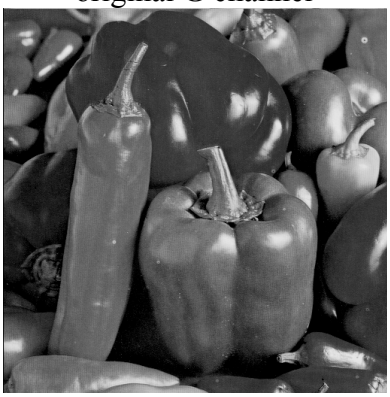


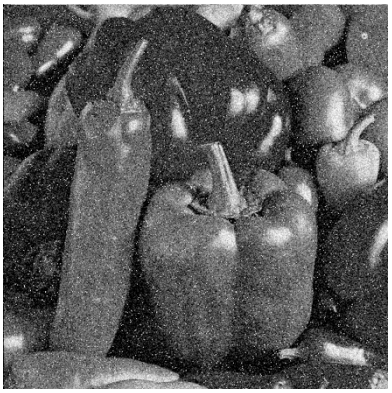
R channel without impulse noise


original G channel


G channel with noise


G channel without impulse noise


original B channel


B channel with noise

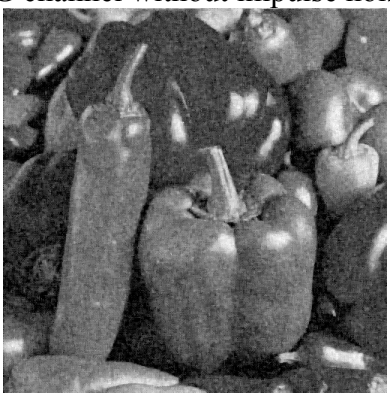

B channel without impulse noise

Figure 13: Noise on different channel

Form Figure 12, there are pepper and salt noise in three channels, and clearly there is Gaussian noise in red channel. But to determine whether there is Gaussian noise in green or blue channel is hard, so I obtain the images for 3 channels without impulse noise respectively in Figure 13. In general, there exists **impulse and Gaussian noise**, **and each channel should be filtered separately**.

For impulse noise, I use medium filtering. For Gaussian noise, I use mean filtering (Gaussian filter):

Noise type	Filtering method	Parameter
Impulse noise	Medium filter	Window shape and size
Gaussian noise	Gaussian filter	Window size and sigma

If there is only one noise type in each channel, the cascading order doesn't matter. When there is mixed noise, there are two cascading order of these filters:
Order A: Applying medium filter at first, then Gaussian filter.
Order B: Applying Gaussian at first, then the medium filter.

According to PSNR table below, the PSNR for order B is a little bigger than order A. In figure 14, the order A gives a more similar image with the original image compared with order B. **PSNR** is just one measurement of assessing the quality of the resulting image, I also calculate the **SSIM** using the function ssim from MATLAB. The SSIM for order A is 0.94 while for order B is 0.93(discussed in 3.4(2) in details). The selected **cascading order is order A**. The reason is

that **if applying Gaussian filter at first, the impulse noise will be added to the target pixel because Gaussian filter calculates the mean value of the surrounding pixels** (discussed in 3.4(1) in details). Eliminating impulse noise at first can deal with this problem.

order A order B original image

Figure 14: resulting images in different cascading order (N=7, sigma=3)

	Order A	Order B
PSNR	24.48	24.75

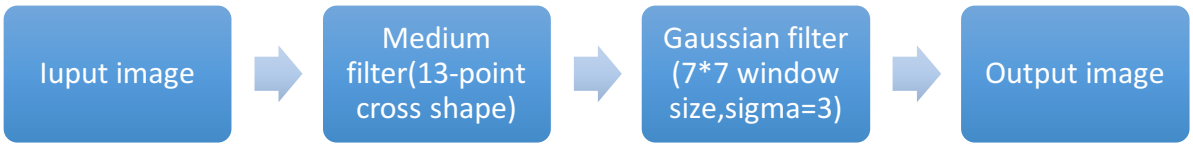
As shown in Figure 15 and PSNR table below, when **N increases, the resulting image becomes more blurred** and the **running time becomes longer**. When the window size is 7, the PSNR is the best compared with N=5 or N=9. Generally, the PSNR becomes smaller as N becomes larger. When **N is very large**, more pixels are used in the filter window to estimate the target pixel, but more information of the image is also lost, for instance, in the medium filter, the medium value can be very different from the true value if it is a pixel far from the target pixel. When **N is very small**, the mean value of not enough neighboring pixels for the target pixel is far away from the true value, so noise still exist.

	N=5	N=7	N=9
PSNR	23.93	24.48	24.36

N=5 N=7 N=9

Figure 15: noise removal images with different window size (sigma=3)

The final method to remove noise:



The output image is shown in Figure 16-(a). Gaussian filter can remove the noise, but edges of the objects in the image are not preserved. The **pixel intensity closeness** should also be considered as a weighting factor, because on both sides of the edges, the pixel value changes rapidly. To improve this performance, I use **bilateral filter** instead of Gaussian filter in step3 above, and the resulting image is shown below.

(a)Gaussian filter (b)bilateral filter

Figure 16: final resulting images and the improved image (N=7, sigma=3, sigma for intensity closeness= 30)

	Gaussian filter	Bilateral filter
PSNR	24.48	25.33

According to Figure 16 and the PSNR above, the resulting image by bilateral filter is better. The edges and details are displayed more explicitly. Because for bilateral filter, each pixel is more subjective to the surrounding pixel with the similar intensity value which is close to it. It can efficiently preserve the edges and eliminate the noise simultaneously.

Guided Image Filtering

The Guided Image filter uses a guidance image and set a local linear model between the output image and the guidance image. The idea is that a point can be expressed of the neighboring points linearly so in 2-D case (image), a pixel can be calculated from the surrounding pixels linearly in a certain window ω_i ($q_i = a_k I_i + b_k$). Since the relationship is linear, so the gradient of the both sides (input and output) exists, that's why this filter can preserve the edge. The criterion function in window ω_i is $q_i = p_i - n_i$. To minimize n_i using ridge regression, the corresponding optimal coefficients $a_k and\ b_k$ can be found. Since the estimate pixel (q_i) is involved in the overlapping window, so the final value is the average of all possible values for this pixel.[1]

In this problem, I use the algorithm above to realize the guided filter. In each window, calculate the variance of guidance image and covariance of input and guidance image. Because the two images are identical, so the variance and covariance are the same. It simplifies the computation. Then it is straightforward to calculate coefficients a and b for each pixel as the center point of the window ω in guidance image. Next step is to get $\bar{a}\ and\ \bar{b}$, by averaging all a's and b's in the window ω centered at that pixel. Finally, use $q_i = \bar{a}I_i + \bar{b}$ to produce the output image. One computational complexity is the convolution of filter with image, it can be solved by O(N) time using *integral image* method or *moving sum* method.

(2)

r=5 $\epsilon = 450$ | r=5 $\epsilon = 650$ | r=5 $\epsilon = 850$

r=7 $\epsilon = 450$ | r=7 $\epsilon = 650$ | r=7 $\epsilon = 850$

r=9 $\epsilon = 450$ | r=9 $\epsilon = 650$ | r=9 $\epsilon = 850$

Figure 17: resulting images with different r and ϵ

From the Figure 16, the optimal parameter is $\boldsymbol{r = 9}$ **and** $\boldsymbol{\epsilon = 450}$. Generally, the resulting image becomes more blurred as r (N) or ϵ increases. $\boldsymbol{\epsilon}$ **is the regularizer**, when ϵ is small, it makes coefficient a bigger and vice versa. Whether a is too big or small, the performance of the resulting image is bad. The **size of window** also matters because the number of pixels in the window determines $\bar{a}$ and $\bar{b}$. Too many pixels result in over smoothing while not enough pixels can't remove noise.

The guided filter performs better than Gaussian filter (linear filter). Guided filter not only preserves the edges and details of object in the image, but also avoid over smoothing. Because Gaussian filter treat the pixels at edges the same as other pixels and calculate the mean value, so the edges are not preserved.

cartoon matting effect:

original image | 1st time | 2nd time

3rd time | 5th time | 6th time

Continue 20 times…

20^{th} time

Figure 18: cartoon matting effect when apply guided filter many times

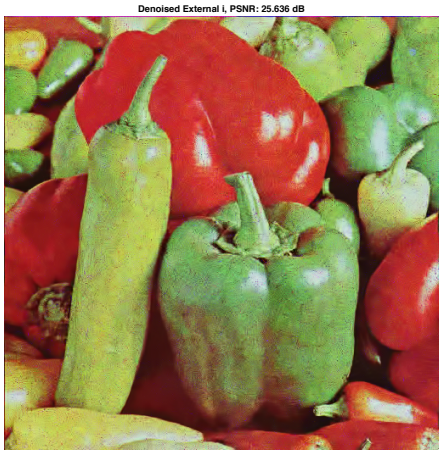
Block Matching and 3-D (BM3D) Transform Filter

BM3D algorithm:

(1) For a noisy image, partition it into several blocks and each block is 2-D dimensional. For one particular block, find the rest blocks similar to it and stack all of them forming a 3-D array (this is called grouping). Apply 3-D transform to the 3-D group and give hard-threshold to the transform coefficients. Invert the transformed 3-D array and the resulting is a group of estimates of the 2-D blocks. Then return the estimate blocks to their position to form a "basic estimate" image. If the blocks are overlapping, use the weighted averaging.

(2) Partition "basic estimate" image into several blocks and for one particular block, find the similar blocks location in "basic estimate" image by block-matching method. According to these location, find related blocks in the noisy image. So far, group the blocks from "basic estimate" image (group A) and the blocks from noisy image (group B) into two 3-D arrays. Apply 3-D transform to group A and B and use the energy spectrum of group A as standard to perform Wiener filtering on group B. Invert the filtered B and return the estimate blocks to their position. In the end, obtain the final image by weighted averaging the estimate blocks. [2]

Here **I use MATLAB code from** http://www.cs.tut.fi/~foi/GCF-BM3D/ . [3]
peppers.raw is used as the original image and *peppers_noise.raw* is used as the noisy image.



sigma=30 PSNR=25.64 dB

sigma=50 PSNR=27.96 dB

sigma=70 PSNR=27.53dB

sigma=90 PSNR=27.20 dB

Figure 19 Resulting images with different sigma

Sigma stands for standard derivation of the noise. Although the distribution of the noise is unknown, when sigma is around 50, the resulting image is the best (Figure 20). Either too small or large sigma will give a worse resulting image. Sigma tells the filter to eliminate how much noise, so it can determine the performance of the filter. When sigma approximates the standard derivation of the noise, the filter can result in a better noise removal image.

Although PSNR for **high profile** is higher, it's hard to tell the difference between the two images in Figure 21. But **normal profile** is faster.

	PSNR	Running time
Normal profile	27.96 dB	12.8s
High profile	28.06 dB	53.7s

normal profile high profile

Figure 20 Resulting images with normal profile and high profile

(2)

BM3D is spatial and frequency domain filter. Because it involves partition the images into blocks in spatial domain and filter the group in frequency domain to remove the noise such as collaborative Wiener filtering.

(3)

bilateral filter

guided filter

BM3D

Figure 21 Three methods for noise removal

Gaussian filter can be used as a noise removal smoothing filter for images, but it loses many edges of the objects. Bilateral filter improves this problem and is good at edges preserving, but it can sometimes lead to over smoothing and losing some textures in the image. Guided filter is efficient and effective in handling the problems above. From Figure 21, guided filter produces better image than bilateral filter. BM3D is the best among these algorithms.

3.4 Discussion

(1)

medium filtering: the pepper and salt noise is distributed at random pixels but have nearly same value, and medium filtering select appropriate value for the noisy pixel, so it can remove pepper and salt noise efficiently. However, the mean value of pepper and salt noise is not zero, so the mean filtering doesn't perform well.

mean filtering: Since Gaussian noise is normally distributed and exist at each pixel. Every pixel has noise and the medium filter will not perform well. The mean of Gaussian is zero, so mean filtering can eliminate the noise efficiently.

(2)

Sometimes human can't tell which is better between two similar images, PSNR is used for the fidelity of the image representation after removing the noise. A higher PSNR indicates more noise removed, but it is also biased towards to blurring effect, so an image with high PSNR may have lots of blurring. The textures or structure should also be considered to evaluate an image. In this problem, I also use SSIM to access the performance of the filer on noise removal, this measurement takes both noise removal and edge preserving into account.

(3)

For each filter above, I eliminate the impulse noise of the input image *peppers_noise* before filtering. Because the impulse noise is the "fake" intensity value.

References

[1]He, Kaiming, Jian Sun, and Xiaoou Tang. "Guided image filtering," IEEE Transactions on Pattern Analysis and Machine Intelligence, 35.6 (2013): 1397-1409.

[2] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," Image Processing, IEEE Transactions on, vol. 16, no. 8, pp. 2080–2095, 2007.

[3] [Online]. Available: http://www.cs.tut.fi/~foi/GCF-BM3D/