EE 569: Homework #1

Issued: 1/7/2019 Due: 11:59PM, 1/22/2019

Name: Wenjun Li
USC ID: 8372-7611-20
Email: wenjunli@usc.edu
Submission Date: 01/22/2019

Problem 1: Image Demosaicing and Histogram Manipulation (50%)

(a) Bilinear Demosacing (10%)

a) Motivation

Digital camera sensors are designed in a so-called color-filter-array (CFA) way in order to obtain color images. And, Bayer Pattern, as shown in Figure 1, is one of the most popular choices for cameras companies. There are 3 kinds of color sensor (i.e. R, G, B.) to capture different light color. Therefore, when a raw image is captured, we need to display the full-color image using neighbor pixels, so that we can have R, G, B values at each pixel. And this is why it takes longer time to open a raw image file than a JPEG file.

As you can know from above paragraph, different demosaicing methods have different time-efficiency and display effect. So, finding a better demosacing method has always been a goal for many camera companies and researchers. Here we introduce 2 simples demosaicing methods: Bilinear demosaicing and Malvar-He-Cutler (MHC) demosaicing.

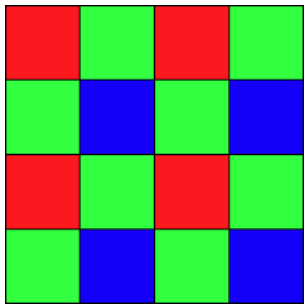


Figure 1. Bayer Pattern

b) Approach

(a) Bilinear Demosaicing

As you can know from the name, we interpolation the current pixel by linearly averaging its neighbor pixels. Let's denote the pixels in figure 1. From p1 to p16 as an order from top-left to bottom-right. Hence, we can represent the R and G values of p6 (the left blue pixel in 2nd row) as:

$$G_6 = (p2+p5+p7+p10)/4$$
$$R_6 = (p1+p3+p9+p11)/4$$

Similarly, we can implement this algorithm to p7 (the right green pixel in row 2),

whose surrounding scenario is different from p6.

$B_6 = (p6+p8)/2$

$R_6 = (p3+p11)/2$

For pixels at the boundary of the image, we can use boundary extension: computing the values of the boundary pixel using its neighbor pixels and mirror-reflecting neighbor pixels. So, before we start computing, we first add mirror-reflecting pixels to the original boundary pixels. When we obtain all the RGB values of each pixel in the original image, we then delete the extension pixels we added.

c) Results

Apply the bilinear demosaicing to the cat.raw image. Results are displayed below.

cat.raw	cat_bilinear.raw
	

d) Discussion

Prob.1.(a).(2).

Question: Do you observe any artifacts? If yes, explain the cause of the artifacts and provide your ideas to improve the demosaicing performance.

Although bilinear demosaicing is the most traditional way to process a raw image, bilinear demosaicing has its natural shortcomings. As you can see in Figure 2., suppose we have a grey image with only 2 different values for each pixel. The left part pixels are dark grey and the right part pixels are light grey. By Bayer Pattern, we then can assign a R/G/B value for each pixel, and correspondingly, the pixel value of R/G/B on the left part is greater than those on the right part. Hence, we have R_D (dark red), R_L (light red), G_D (dark green), G_L (light red), B_D (dark blue), B_L (light blue). Then we do bilinear demosacing to assign each pixel with RGB value, and we can have (d), (e), (f) if we only display one color. Here you can see 3 different pixel values in (d) and (f). Since we only have 2 pixel values for Red and Blue in (b), so, the bilinear interpolated image has a new pixel value, i.e. a new color, and this phenomenon is called False-color effect. Besides, you can see an interleaved pixel value change in (e), which is like a zipper. We called this phenomenon as Zipper

Effect. Therefore, the bilinear interpolated image we finally obtained is (c) and it has both Zipper Effect and False-Color Effect.

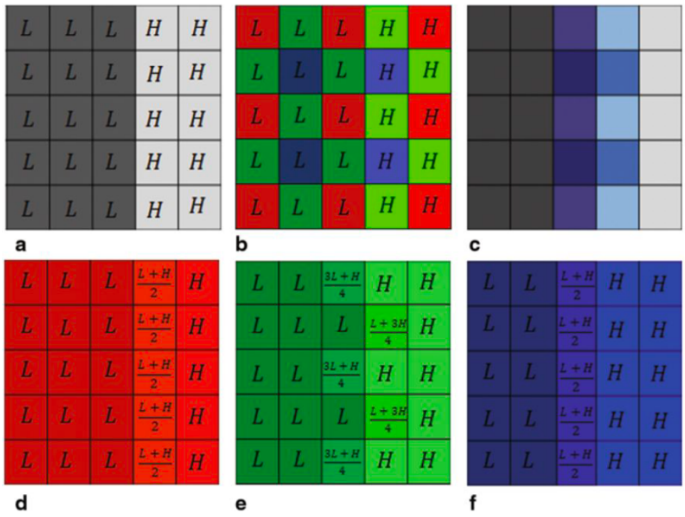


Figure 2. a grey image, b CFA samples of a, c bilinear interpolation result, d bilinear interpolated red plane, e bilinear interpolated green plane, f bilinear interpolated blue plane

So, when will bilinear demosaicing introduce these 2 effects that will influence the image quality? As Figure 2. (a) implies, when there are edges that differs in pixel grey value a lot, we will have Zipper Effect and False-Color Effect in the processed image. The fence in the original lighthouse image has the feature accords with above-mentioned reason, hence, we can observe some artifacts (false-color and zipper effect) in the fence area.

What we can do to avoid this? First, we can try to make our interpolation edge-sensitive. We can add higher order term to correct the error.

(b) Malvar-He-Cutler (MHC) Demosaicing (20%)

a) Motivation

Because bilinear demosaicing has some natural shortcomings when it comes to sharp edges, for example: false-color and zipper effect. So, we need better better methods to demosaicing images. Malvar et al. [2] proposed a new method by adding a 2^{nd}-order cross-channel correction term to the basic bilinear demosaicing output. Because of this 2^{nd}-order correction term, MHC performs better when the image has sharp edges.

b) Approach

Figure 2. gives the template coefficients of MHC 5x5 window demosaicing. This template is similar to a bilinear interpolation demosaicing template, but using four diagonal neighbor pixels instead of up/down/left/right neighbor pixels. And this template intend to miss some pixel in certain locations, which help to build strong relation between center pixel and its neighbors.

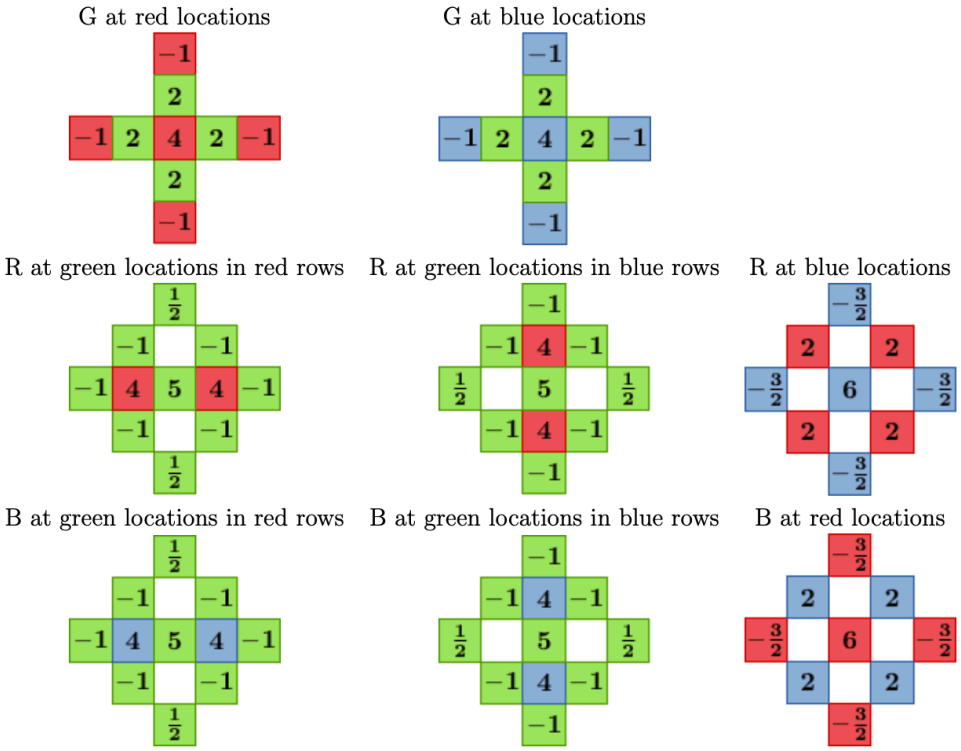


Figure 3 MHC 5x5 linear filter, the coefficient are sacled by 8

c) Results

cat.raw

car_mhc.raw

cat_bilinear.raw

d) Discussion

Overall, the performance of MHC is better than bilinear, especially in the region with sharp color changes, i.e. edges. In bilinear demosaicing, the interpolated pixel values are derived from neighbor pixels independently, the difference of center pixel and neighbor pixels near edges will produce zipper artifacts and color-false. However, in MHC demosaicing, thanks to the correction term and missing pixels at certain locations, it preserves some edge property and avoid false-color and zipper effect.

(c) Histogram Manipulation (20%)

a) Motivation

Sometimes images can be very dark or bright, i.e., not uniform. In order to improve image quality in this situation, we need to enhance the contrast of the image, i.e. make the bright area darker or dark area brighter. If we want to enhance the contrast, we have to know the original contrast, and this introduces grey-scale histogram. In histogram, we can know

what's the percentage of every different grey-scale value, and according to this, we can apply an equalization function to the original histogram and obtain an equalized histogram, which gives us a more uniform image.

b) Approach

Here are 2 approaches to equalize the histogram: transfer-function-based algorithm and cumulative-probability-based algorithm.

Method A: transfer-function based function:

Figure 4. shows how to do transform-function based histogram equalization. First, we calculate the histogram of the original image and then make it to be cumulative and normalized it by deviding the number of all the pixels (i.e. image_length*image_width). After doing this, now we obtain a cumulative histogram of original distribution CDF of image pixels. If we map this original distribution to a uniformly distribution CDF, then we can get a histogram equalized image.

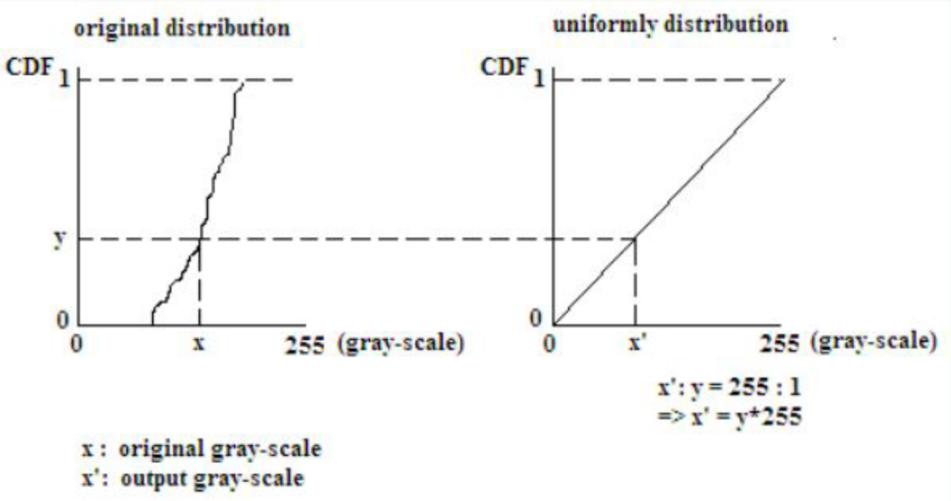


Figure 4 Transform-Function based Algorithm

Method B: cumulative-probability-based algorithm:

Method B cumulative-probability-based algorithm can be also called Bucket-Filling algorithm. First, we draw the histogram of the original image. Then, according to pixel intensity value, we list the pixel number of certain intensity value according to pixel intensity value from 0 to 255. Now, in order to make pixel number of each intensity the same, i.e. equalized, we put all the pixels into different intensity value in order. To do this, we first computer how many pixels should go to each intensity, and then assign the pixels with new intensity in order. Figure 5. shows a simplified version of "Bucket Filling".

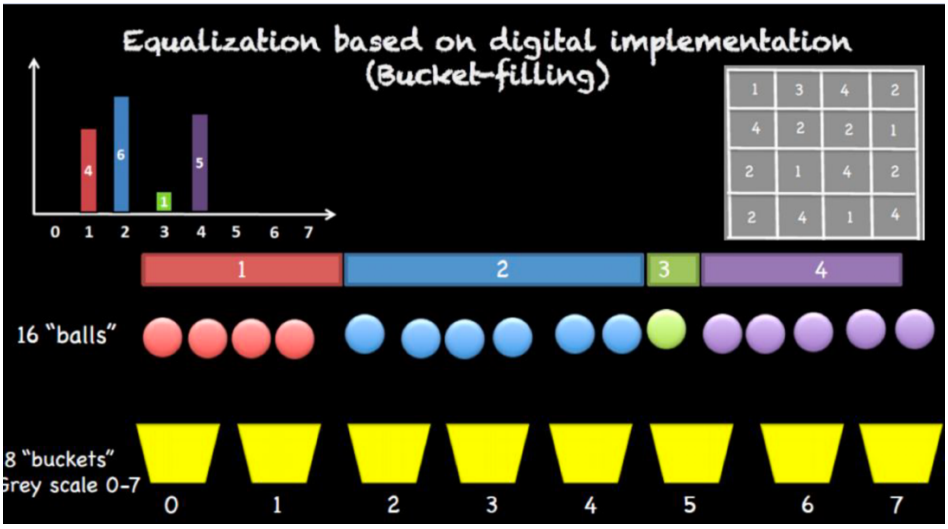
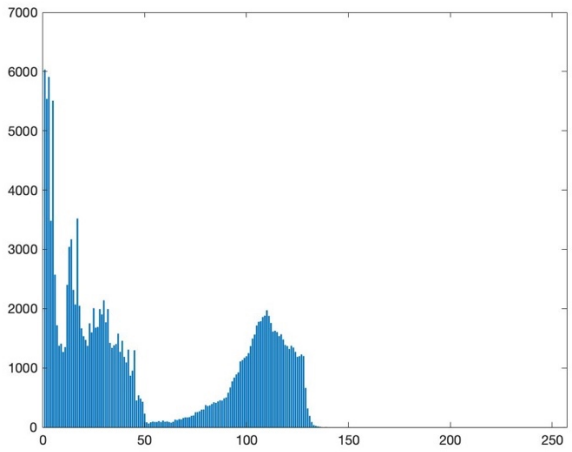


Figure 5. Cumulative-Porbability-Based Algorithm (Bucket-Filling Algorithm)

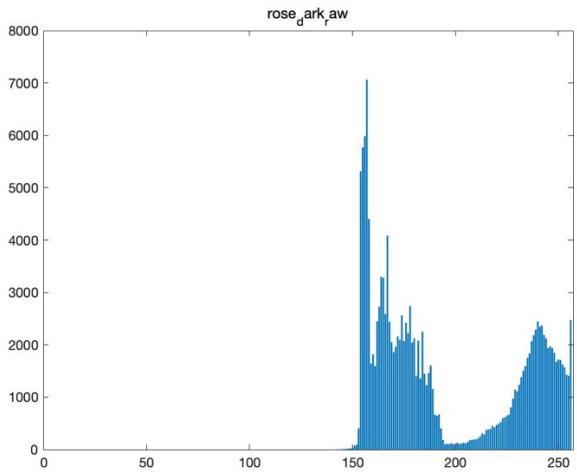
c) Results

(1)Plot the histograms of all images (2 input images and 4 output images).

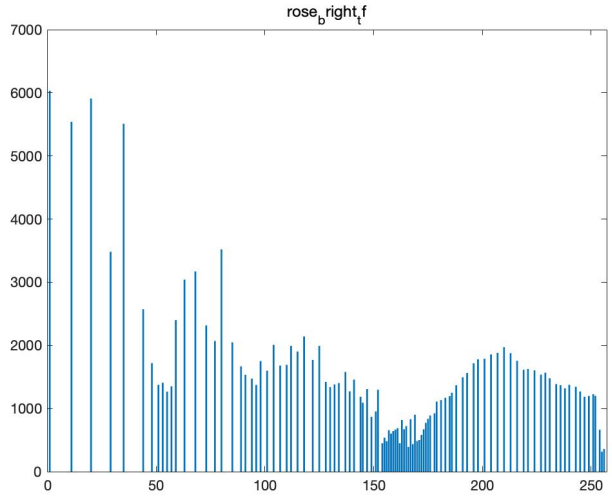
1.rose_bright.raw



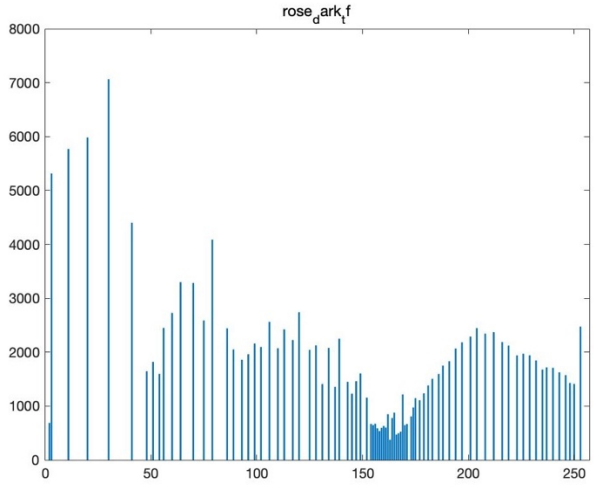
2.rose_dark.raw



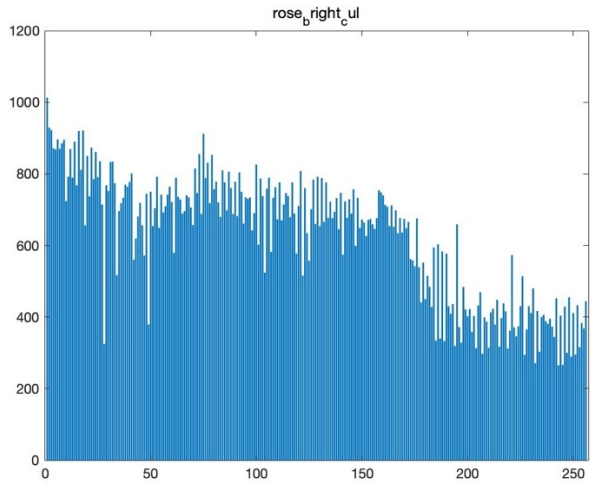
3.rose_bright_tf.raw



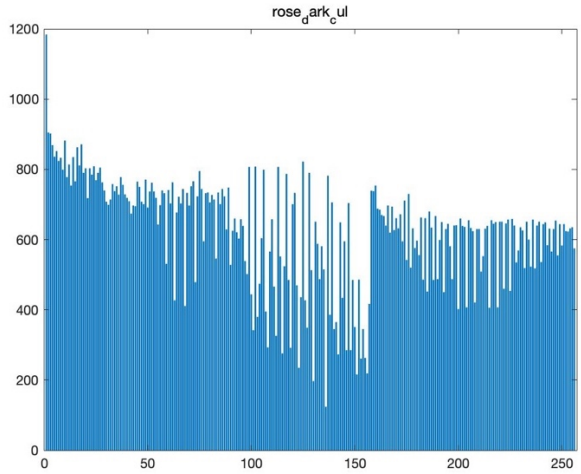
4.rose_dark_tf.raw



5.rose_bright_cul.raw



6.rose_dark_cul.raw



(2)Apply method A to rose_bright.raw and rose_dark.raw. Show original images and enhanced images. And plot the transfer function.

rose_bright_raw	rose_bright_cul.raw
	
rose_dark.raw	rose_dark_cul.raw
	

Corresponding transfer function

(3)Apply method B to rose_bright.raw and rose_dark.raw. Plot the cumulative histogram for each testing image.

rose_bright_raw	rose_bright_cul.raw
	
rose_dark.raw	rose_dark_cul.raw
	

Correspond cumulative histogram

d) Discussion

(4)Question: Discuss your observations on these two enhancement results. Do you have any idea to improve the current result?

Answer: we can set threshold to histogram equalization. If we set threshold from 20-230 (i.e. cut those grey intensity from 0-19 & 231-255), we can remove those extreme pixel intensity from our image. Hence, we can obtain a more uniform image.

(5)Question: Apply your implemented Method A and Method B to rose_mix.raw and show the result. Can you get similar result as in previous part? If not, how to modify your implementation? Please justify your answer with discussion.

Result Images:

Rose_mix.raw	Rose_mix_tf.raw	Rose_mix_cul.raw
		

Answer: The results we get from rose_mix.raw is not as good as in the previous part.

The first way is to achieve better histogram equalization result, we can split the original image apart, and do histogram equalization from part by part, and combine them together at last. For example, we can cut the rose region out and do TF histogram equalization to it first. Then, we do CUL histogram equalization to the whole image (including the rose region).

The second way to get better result is to assign intensity value to certain region. As we can see in the rose_mix.raw image, there is a dark region outside the top of rose. First, we write a code to return the intensity value of the while region, and then we assign the dark region with our result. In this way, we can get rid of the non-uniform part effectively.

Problem 2: Image Denoising (50%)

(a) Gray-level Image (20%)

1. Motivation

Due to various reasons, such as equipment defect, transmission error, storage damage and so on, there will be noises in images. There are 2 major kinds of noise: uniform noise and impulse noise. To achieve better image quality, people create lots method to denoise the image.

2. Approach

To filter uniform noise, people often use mean filter and gaussian filter. All the filters can be viewed as a small window. Mean filter calculate the mean value of pixels within an N*N window and return this value to center pixel. Then, the window center pixel will move to next position to repeat the above operation again until it comes to the last pixel in the image. Mean filter calculates the mean of the window, it does not consider the effect of distance form center pixel to its neighbor pixel. To conquer this shortcoming, people invent Gaussian filter. By assigning the pixel at different position with Gaussian weight value, we can better preserve image properties.

Figure 6. gives the kernel of a 3*3 mean filter (it will be scaled by 1/9 before output).

1	1	1
1	1	1
1	1	1

Figure 6. Mean Filter Kernel

The Figure 7. gives a 5*5 Gaussian filter kernel, where (i, j) is the index of the pixel in the context of the entire image and (k, l) is the index of the pixel in the context of a window of size w1*w2.

- Image denoising:
  - uniform: low pass filter (Gaussian)

$$Y(i,j) = \frac{\sum_{k,l} I(k,l)w(i,j,k,l)}{\sum_{\substack{k,l \\ w_1}} w(i,j,k,l)}$$

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

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

where σ is the standard deviation of Gaussian distribution

Figure 7. Gaussian Filter (5*5) Kernal

However, mean filter and Gaussian perform bad when it comes to impulse noise, because impulse noise gives extreme values in the filtering window and this affects the performance of mean and gaussian filter. To filter impulse noise, people come up with median filter, which can effectively remove the influence of extreme pixel value. As it can be implied from the name, this filter calculates the median value within the filtering window and returns it back to center pixel.

All of the filters introduced above perform bad when the region has sharp edges. So, how to achieve this? Similar to the idea in MHC interpolation, we add a term representing the intensity difference between pixels to the gaussian weight. Hence, now the gaussian weigh is determined not only by Euclidean distance of pixels, but also by the intensity value difference of pixels.

Here, we give the equation of Bilateral filtering. Sigma-c and sigma-s here represent the spread parameters.

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

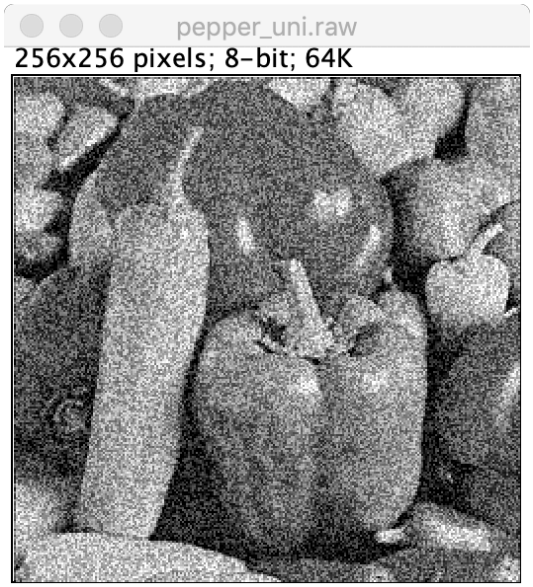
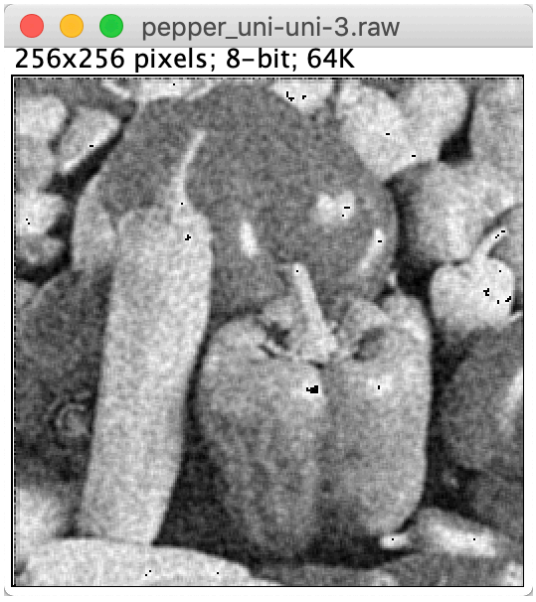
$$w(i,j,k,l) = exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_c^2} - \frac{\|I(i,j) - I(k,l)\|^2}{2\sigma_s^2}\right)$$

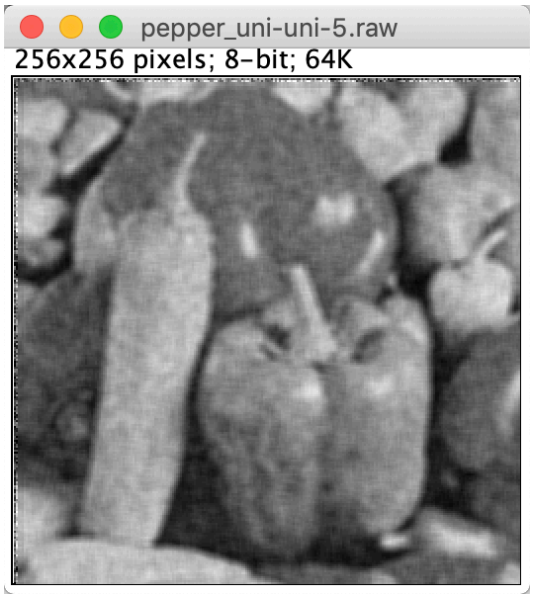
A novel filter called non-local mean (NLM) filter was proposed in 2005 and this filter perform even better than bilateral filter. It takes gaussian weighted Euclidean distance between the block centered the target pixel and the neighboring block. In other words, it tries to find a window which has similar scenario with current window, and use the similar window to help. This NLM filter can be very computationally demanding, so, to speed up the filtering processing, we often a smaller window rather than the entire image.

3. Result

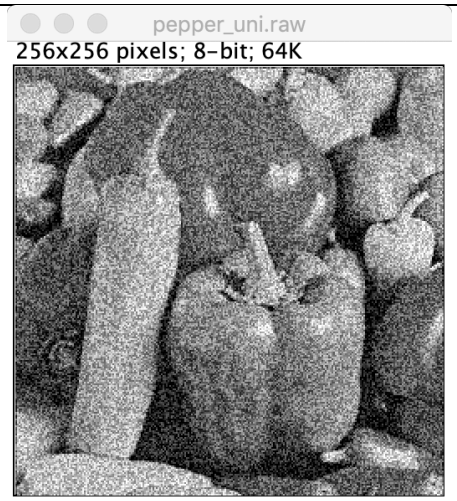
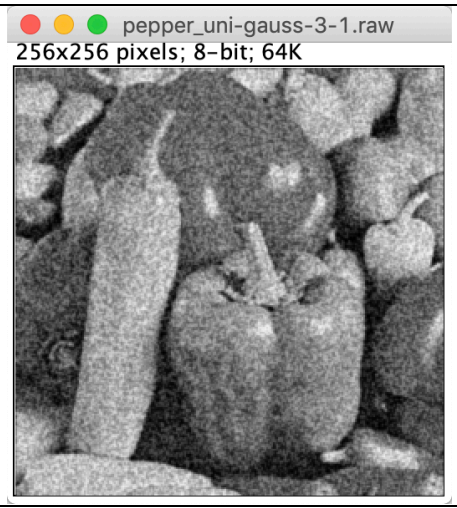
(1) Apply mean filter to pepper_noise.raw

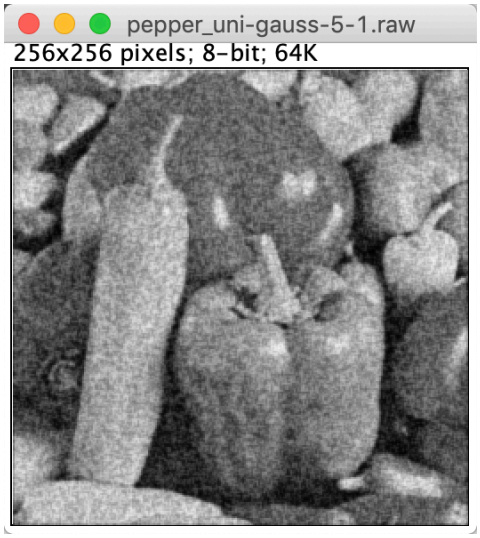
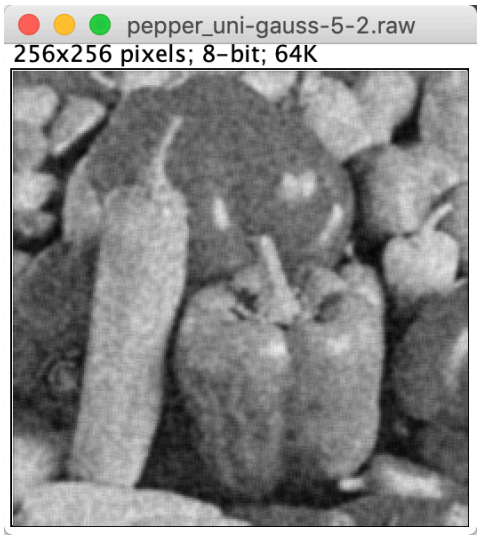
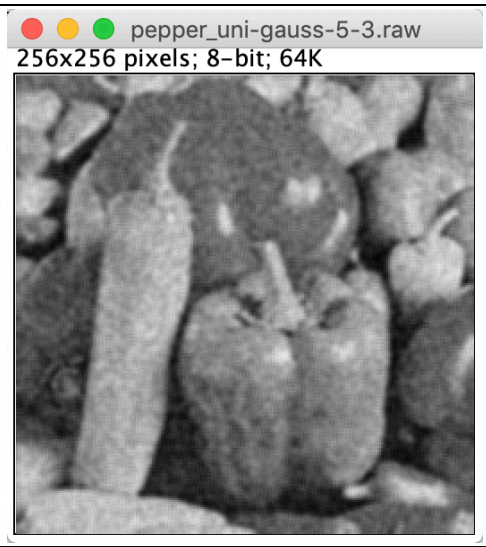
a. window size 3*3

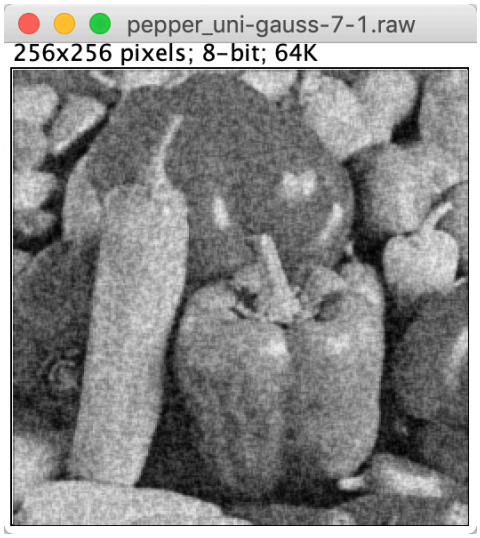
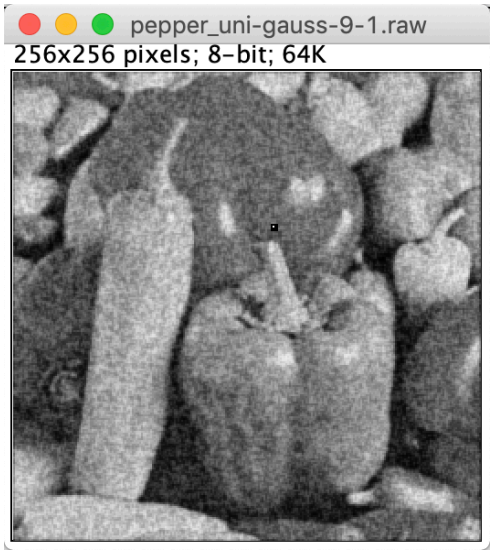
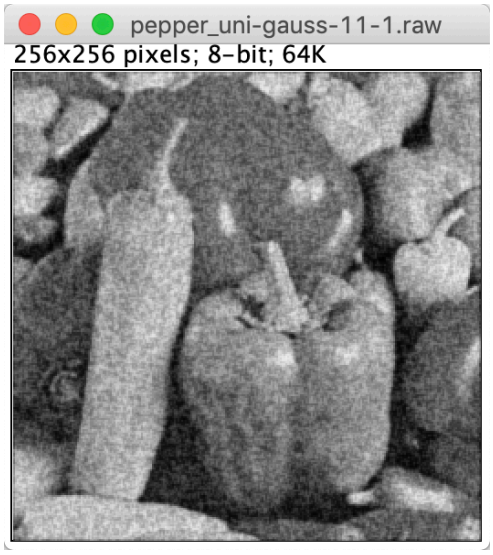
Pepper_noise (the original image)		
Pepper_noise_mean-3 (the last number 3 represents window size)	PSNR=15.80	

Pepper_noise_mean-5	PSNR=15.68	

(2)Apply Gaussian filter to pepper_noise.raw

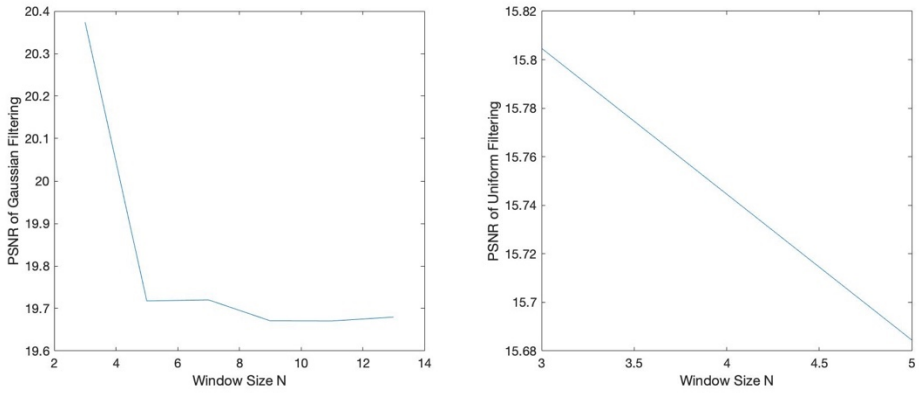
Pepper_noise		
Pepper_noise_gauss-3-1 (the first number represents window size; the second number represents sigma in STD gaussian distribution)	PSNR=20.37	

Pepper_noise_gauss-5-1	PSNR=19.72	
Pepper_noise_gauss-5-2	PSNR=17.89	
Pepper_noise_gauss-5-3	PSNR=17.58	

Pepper_noise_gauss-7-1	PSNR=19.72	
Pepper_noise_gauss-9-1	PSNR=19.67	
Pepper_noise_gauss-11-1	PSNR=19.67	

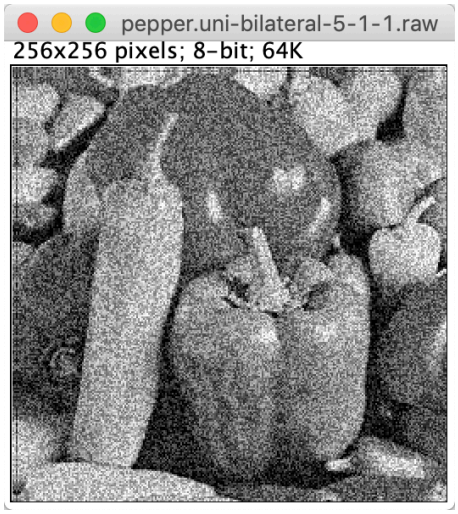
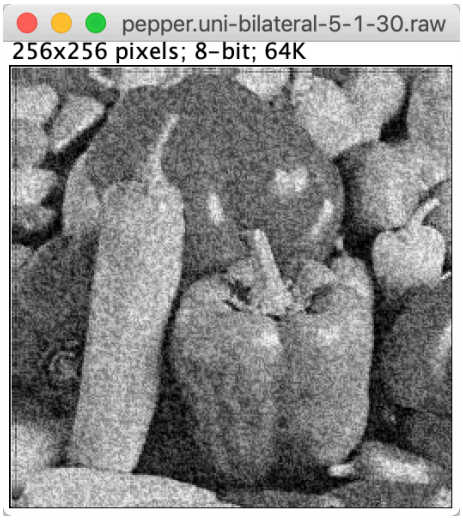
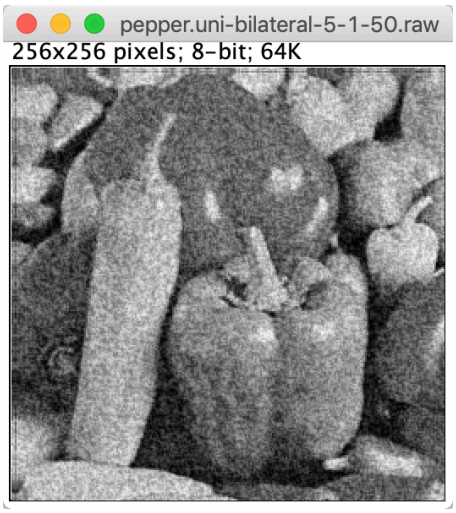
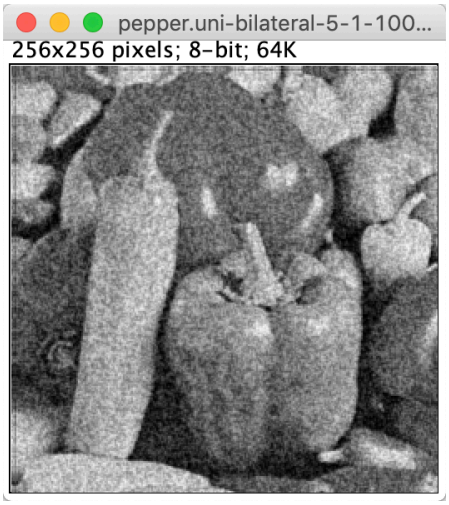
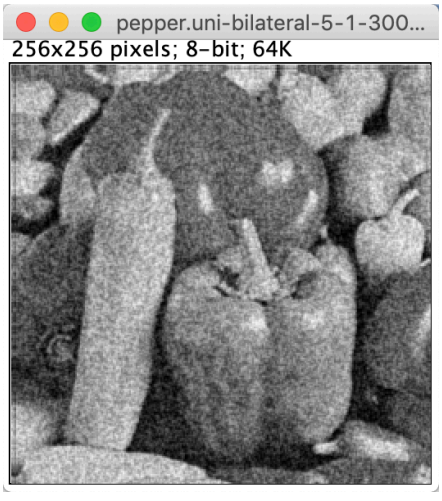
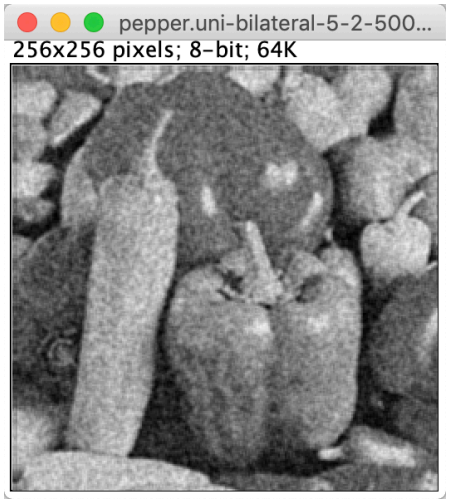
EE 569 Digital Image Processing: Homework #1

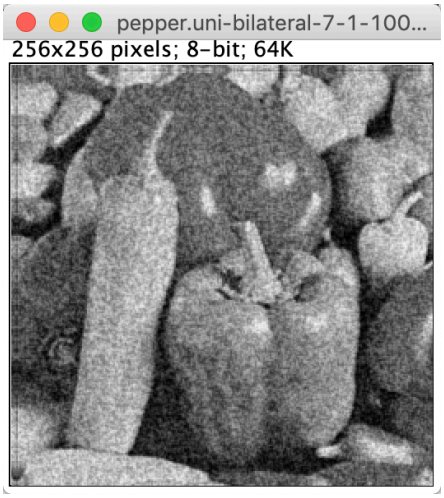
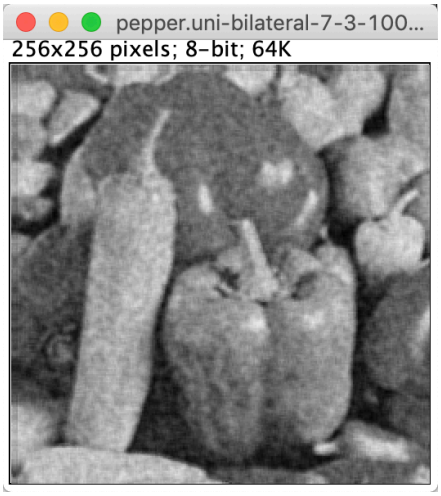
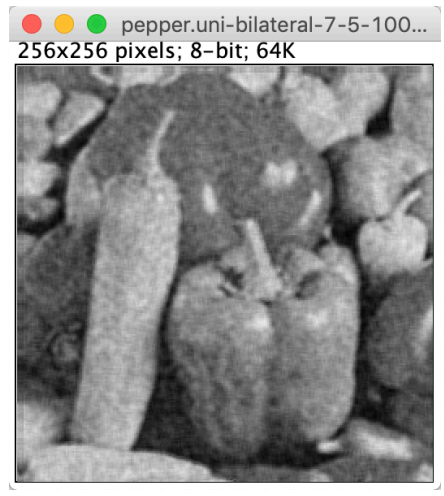
Plot of PSNR of mean filter & gauss filter in terms of window size:



(3) Bilateral Filter

*first number is window size; second number is sigma-c; third number is sigm-s

Pepper_bilqteral-5-1-1	Pepper_bilqteral-5-1-30	Pepper_bilqteral-5-1-50
		
Pepper_bilqteral-5-1-100	Pepper_bilqteral-5-1-300	Pepper_bilqteral-5-2-500
		

Pepper_bilqteral-5-7-100	Pepper_bilqteral-7-3-100	Pepper_bilqteral-5-1-1
		

(4) Non-Local Mean filter (NLM)

4. Discussion

(1)Question: What is the type of embedded noise in Figure 7(b)?

Answer: uniform noise.

(2)Question: the filtering outcome in terms of different window size & gaussian STD sigma

Answer:

For mean filter, the bigger the filtering window size is, the smoother the outcome the image is. For example, in mean filtering, the outcome image of window size 5 is apparently smoother than that of window size 3.

For gaussian filter, the smoothing effect of the window size gives to outcome image is smaller (compared to mean filter), because it has gaussian weighted scaler. The more distant the pixel from the center pixel is, the smaller gaussian scaler it is. In general, when the sigma in STD Gaussian becomes bigger, the outcome image will have less noise, but, at the mean time the outcome image is more blurred. Choosing a right window size and sigma is significant for Gaussian filter.

For bilateral filter, the sigma-c and sigma-s have more influence on the outcome image than the window size does. It is because that bilateral filter does not only consider the Euclidean distance, but also takes pixel intensity value into account. As we can see in above outcome images filtered by bilateral filter, the sigma-c has the largest effect to the filtered image. When the sigma-c becomes larger, the filtered image will become blurred quietly and lose some edge property. Therefore, we need to set sigma-c relatively small. For sigma-s, it considers the edge property and returns this property to the gaussian scaler, therefore, it acts positively in keeping the edge property. Under this context, we need to set the sigma-s reasonably, i.e. let sigma-s preserves the edge while not blurring the image. In my discussion, I set sigma-s between 100-300, and it performs good. As we can from all the filtered image, bilateral filter has the best outcome.

(b) Color Image (20%)

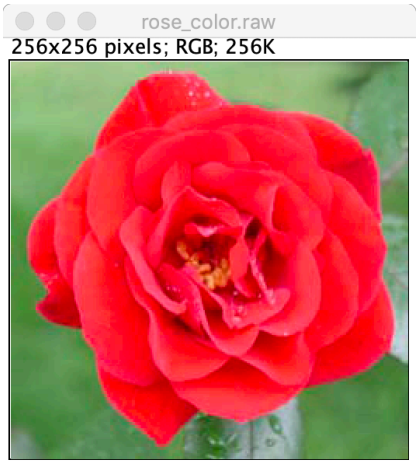
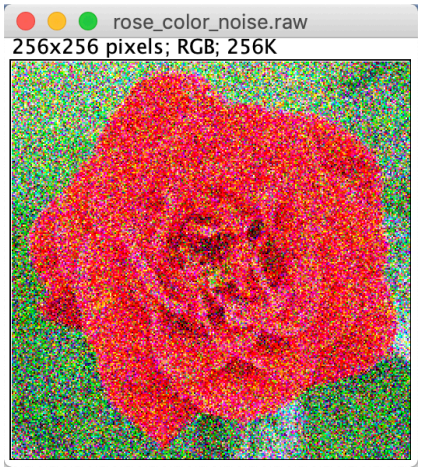
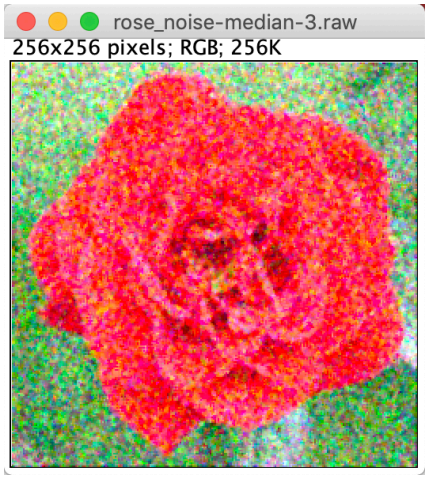
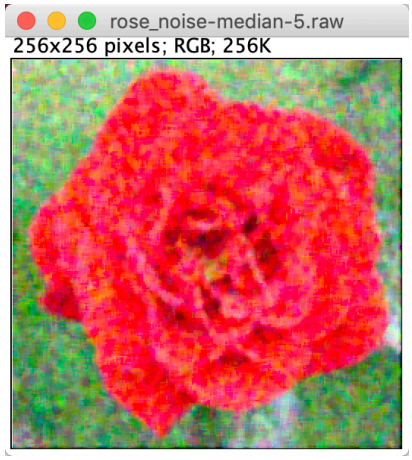
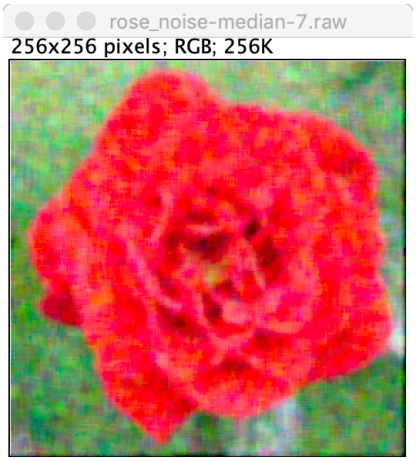
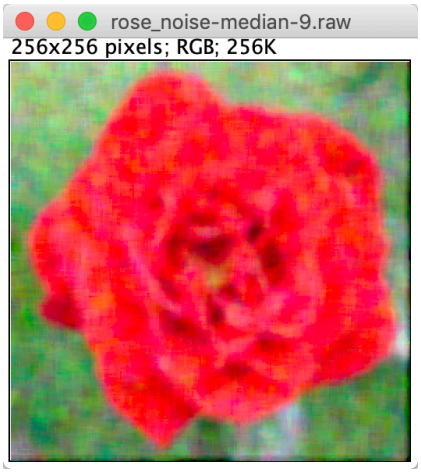
1. Motivation & Approach

The motivation and approach under color image denoising is similar to grey image denoising. The approach here just includes dealing with multiple channels. Please refer to above motivation and approach.

2. Results

*the order of filter used is stated in the file name; the first number after filter name is window size;

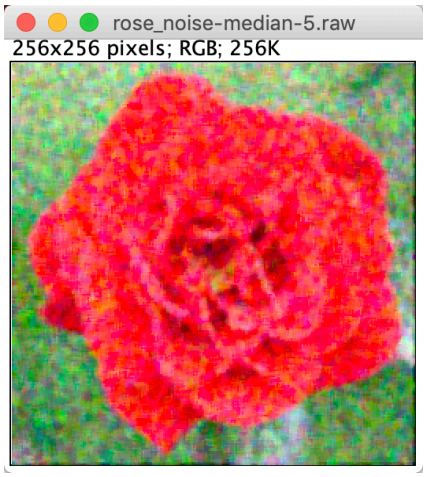
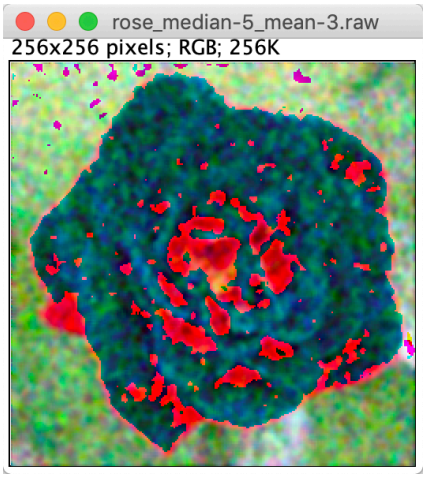
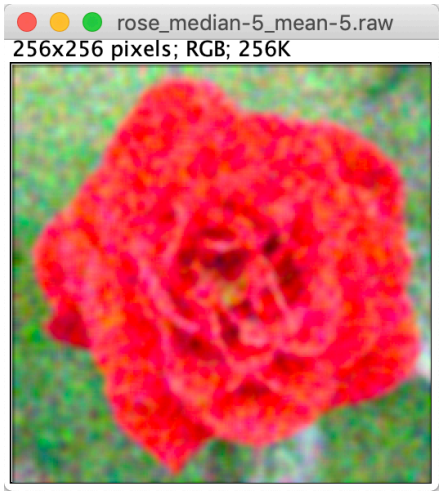
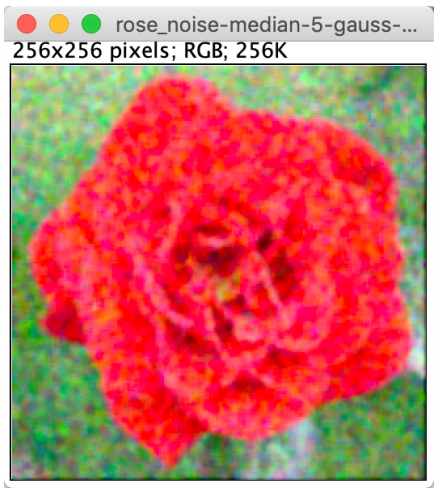
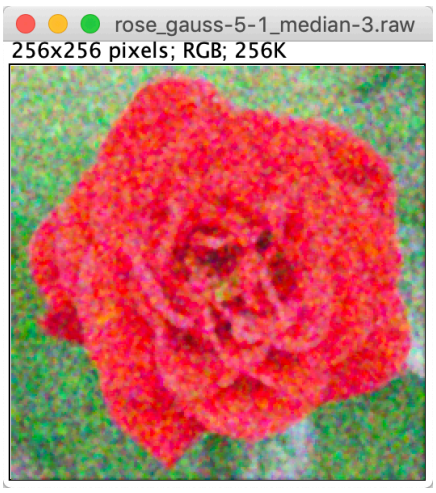
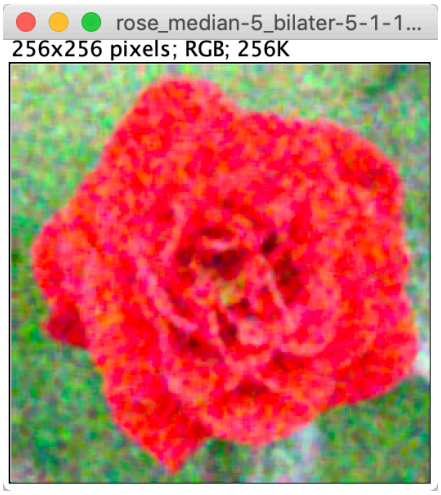
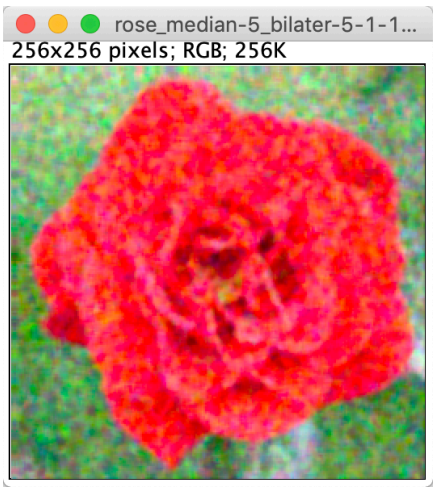
First, I implemented median filter to remove impulse. And the output image of median filter with window size 5 is relatively the best (considering the blur and edges).

Rose_color_original	Rose_color_noise	Rose_noise_median-3
		
Rose_median-5	Rose_median-7	Rose_median_gauss-5-1
		

EE 569 Digital Image Processing: Homework #1

Secondly, I implemented mean/gauss/bilateral filter to rose_median-5.raw.

*filter name, filter implemented order, window size are in file name

Rose_medain-5	Rose_median-5_mean-3	Rose_median-5_mean-5
		
Rose_median-5_gauss-5-1	Rose_gauss-5-1_median-3	Rose_gauss-5-1_median-5
		
Rose_median-5_bilater-5-1-100	Rose_median-5_bilater-5-1-150	Rose_median-5_bilater-7-1-100
		

3. Discussion

(1) Question: Should we perform filtering on individual channels separately for both noise types?

Yes. It has uniform noise and impulse noise in all R/G/B channels. So, we need to filter all the image channels.

(2) Question: What filters would you like use to remove mixed noise?

First, use median noise to remove impulse noise. Then, use gaussian filter and mean filter to remove uniform noise. Also, I used bilateral filter to replace gaussian & mean filter.

(3) Question: Can you cascade these filters in any order? Justify your answer

No. Median filter should be implemented first. Because we should remove impulse noise first, or it have bad influence on gaussian filtering and bilateral filtering due to the extreme impulse value.

To justify this, I implemented Gaussian filter to rose_noise.raw first and then implement median filter. The output images are displayed in the above table, the file names are “Rose_gauss-5-1_median-3” and “Rose_gauss-5-1_median-5”. Comparing this 2 output images to “Rose_median-5_mean-5”, “Rose_median-5_gauss-5-1”, and “Rose_median-5_bilater-5-1-100”, we can see the quality of those output images that are first implemented by median filter are better than “Rose_gauss-5-1_median-3” and “Rose_gauss-5-1_median-5”.

(4) Question: It could be difficult to remove uniform noise satisfactorily. Filters may blur the object’s edge when smoothing noise. A successful design of a uniform noise filter depends on its ability to distinguish the pattern between the noise and the edges. Please suggest an alternative to low pass filter with Gaussian weight function and discuss why such solution can potentially work better. (You are welcome to implement the filter, but it is not required)

4. Shot Noise (10%)

a) Motivation

b) Approach

c) Results

d) Discussion

References

[1] Emre Celebi, Michela Lecca, Bogdan Smolka: Color Image and Video Enhancement

[2] [1] M. E. Celebi et al. (eds.), Color Image and Video Enhancement