EE569 - Hw1

Mozhdeh Rouhsedaghat

2726554211 rouhseda@usc.edu

Problem 1: Image demosaicing and histogram manipulation

Abstract and Motivation:

Digital cameras can only capture one of the 3 main colors per sensor at each pixel location, so their initial output has only one value per pixel and other two colors should be constructed based on the neighbor values to have a complete RGB picture. There are some methods to do it: bilinear demosaicing, Malvar-He-Cutler (MHC), … [1]

Sometimes a digital image is too dark or too bright. In this case its luminance is concentrated in one side of the luminance histogram. To correct this, there are several methods to correct the image through manipulating the histogram. One of them is called transfer-function-based histogram equalization method and another is the cumulative-probability-based histogram equalization method.

Approach and Procedures:

bilinear transformation

For applying bilinear transformation, at each pixel, we obtain the missing values of a main color through computing the average of present neighboring values of the same color. As an example, for an input with the Bayer pattern the missing blue and green values at pixel R3,4 are estimated as:

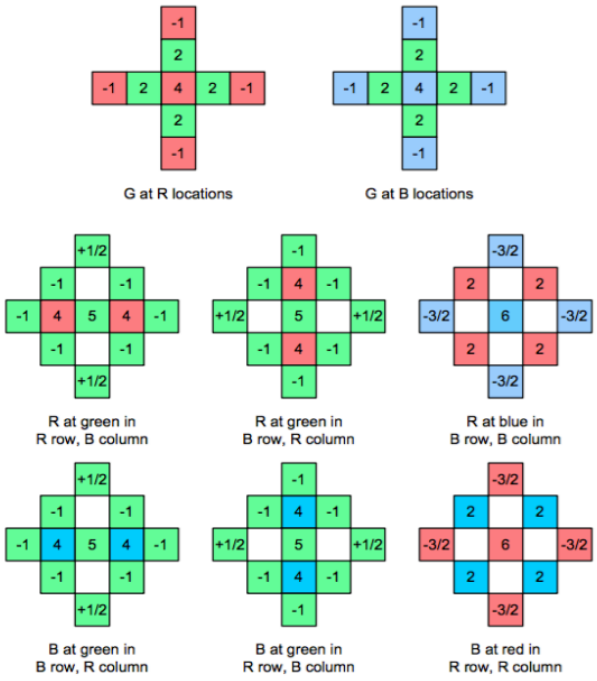
$$\hat{B}_{3,4} = \frac{1}{4}(B_{2,3} + B_{2,5} + B_{4,3} + B_{4,5})$$
$$\hat{G}_{3,4} = \frac{1}{4}(G_{3,3} + G_{2,4} + G_{3,5} + G_{4,4})$$

For pixel $G_{3,3}$ the blue and red values are calculated as:

$$\hat{R}_{3,3} = \frac{1}{2}(R_{3,2} + R_{3,4})$$
$$\hat{B}_{3,3} = \frac{1}{2}(B_{2,3} + B_{4,3})$$

MHC demosaicing

For applying MHC demosaicing, at each pixel, we obtain the missing values of a main color through computing the weighted average of present neighboring values. In fact, it adds a 2th order term to the bilinear demosaicing result. the mentioned weighs are as below:



Transfer-function-based histogram equalization method

For applying the transfer-function-based histogram equalization method, first we have to obtain the histogram by counting the number of pixels with each gray-scale value. Then we must calculate the CDF by adding the number of pixels of a specific value with the number of pixels with a less value and at last we have to do a one to one mapping from the CDF of the image to the uniform CDF to obtain the new gray scale value for each set of pixels with equal gray scale value. The equal formula for this conversion is

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

Where M*N is the number of pixels in the image and L-1 is the max gray scale value (255). [2]

Cumulative-probability-based histogram equalization

For applying the cumulative-probability-based histogram equalization method, we aim to put equal number of pixels in each gray scale value(bucket). To do this, we can first sort the location of pixels according to their gray scale values and then, starting from the beginning of this sorted array, we can assign each 400*400/256 =625 pixel to the corresponding gray scale value and in this way assign pixels in the original image to the corresponding bucket.

Results:

a)1) see fig 1.

Fig 1: The result of applying bilinear demosaicing to the “cat” image (left) The result of applying MHC demosaicing to the “cat” image(right)

a)2) Yes. In the place of edges, a zip shape (zipper artifact) can be seen which is because edges are of high frequency while this transformation is a low pass filter and can’t capture edges correctly. One way to improve this, is to use higher order filters like MHC.

b)1) see fig 1.
b)2) By comparing the result of applying the bilinear and MHC demosaicing we can easily see that the output of MHC is sharper and less blur. This is because considering a weighted average of the surrounding pixels to find the edges more clearly instead of simply computing a uniform average which leads to a blur image.

c)1)

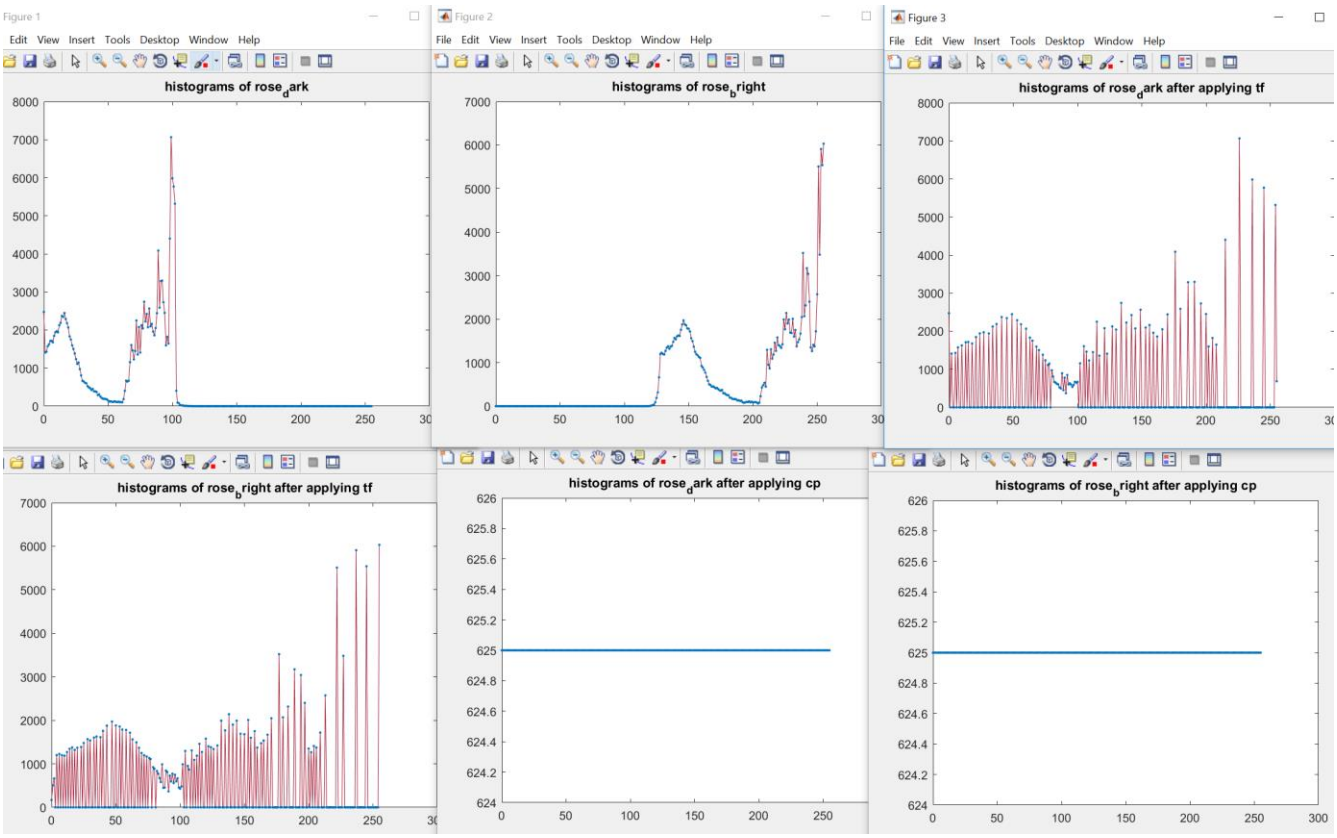


Fig 2: The histogram of all images

c)2)

Fig3: The result of applying transfer-function-based histogram equalization method to the “dark rose” image(left)and to the “bright rose” image(right)

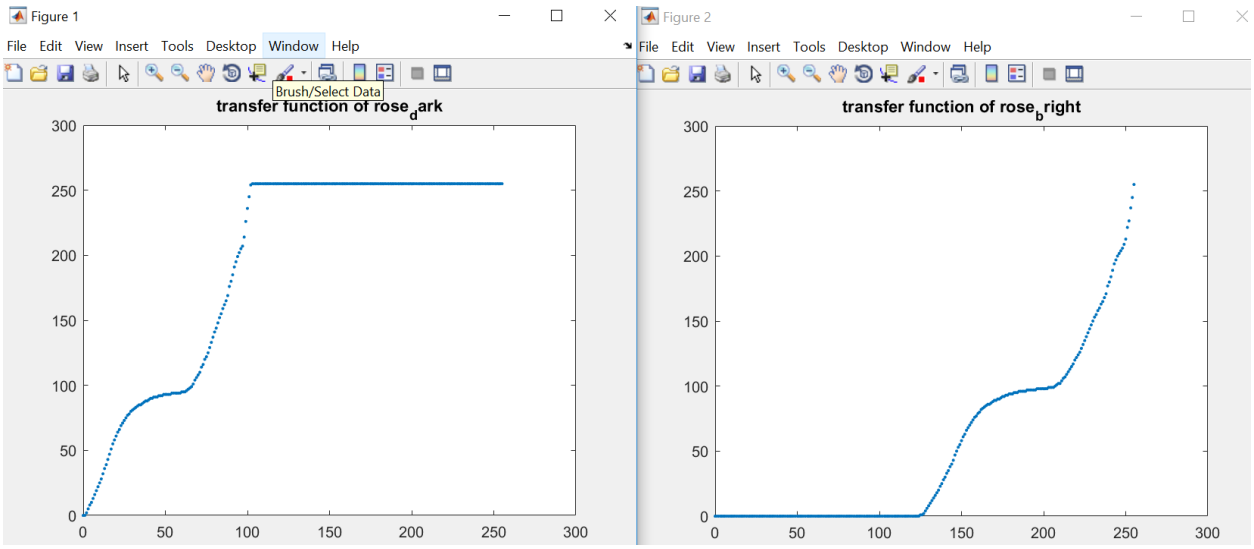


Fig 4: The transfer function of rose_dark and rose_bright

c)3)

Fig 5: The result of applying cumulative-probability-based histogram equalization method to the “dark rose” image(left)and to the “bright rose” image(right)

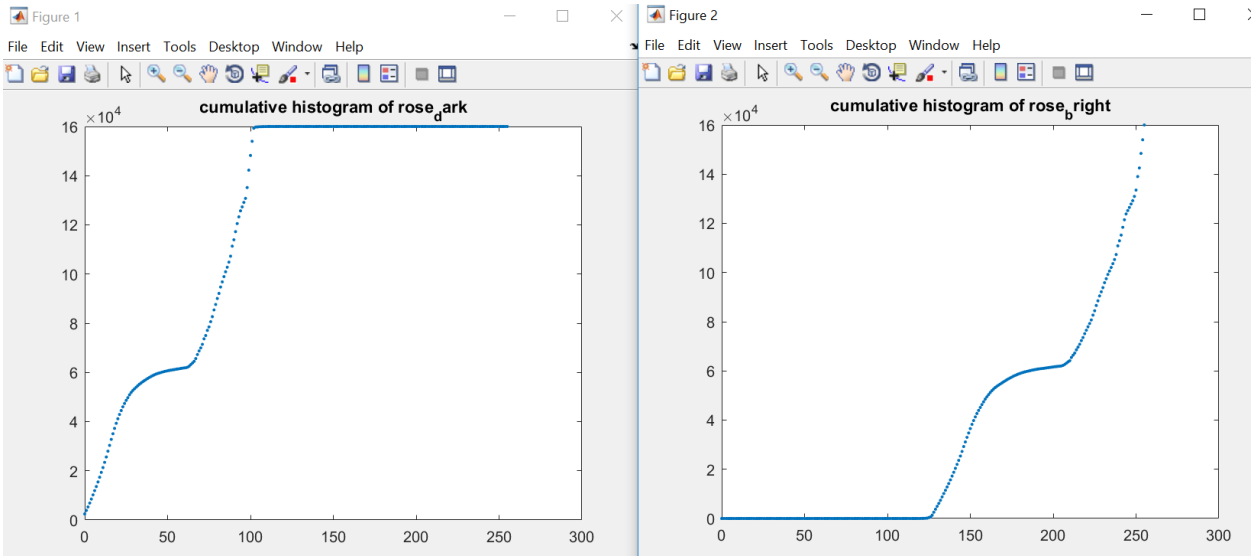


Fig 6: The cumulative histogram of rose_dark and rose_bright

c)4) The used methods enhance extreme darkness or brightness with the same quality as we can barely see a difference in the resulting images, whether they were dark or bright before the transmission. We can also see that the results of the second method are a little smoother.

About the first technique, by removing the jumps in the diagram we can have some improvements. For example, if some gray-scale values in the diagram correspond to a high number of pixels (more than a specific threshold) we can divide this pixel among neighboring gray-scale values and in this way flatten the diagram.

About the second method we can do the selection part (to fill the buckets)from pixels with the same gray-scale value randomly and have more smoothness in the resulting image.

c)5)

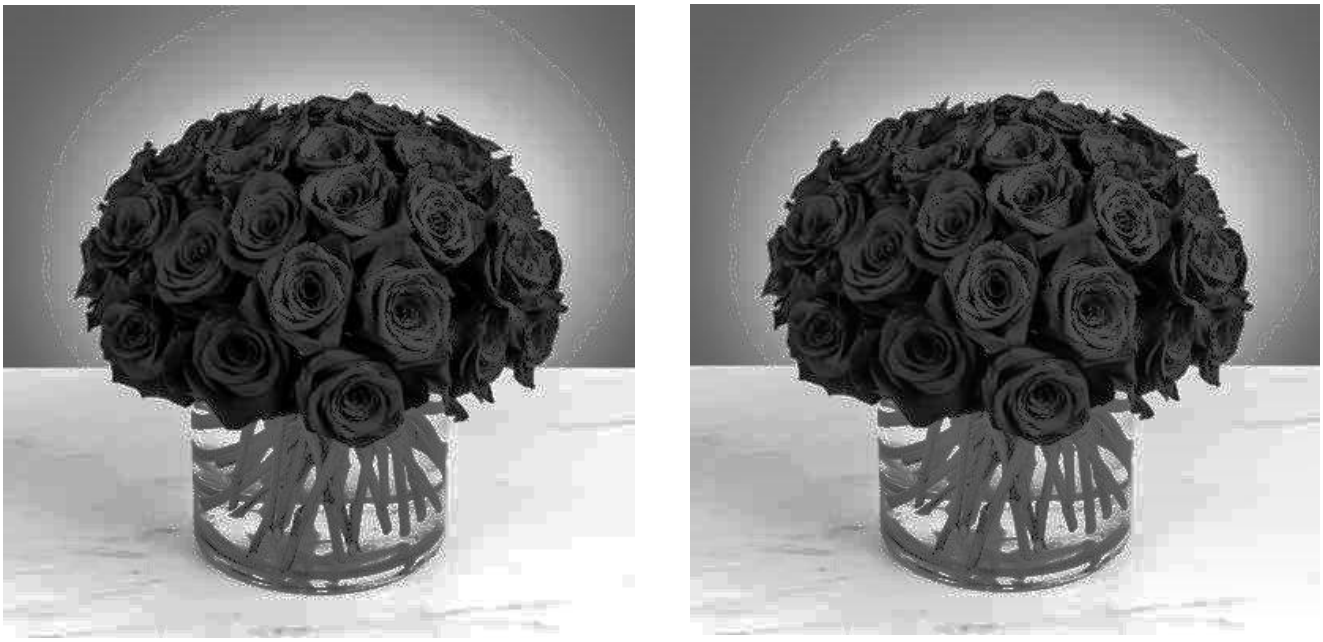


Fig 7: The result of applying transfer-function-based histogram equalization method to the “mix rose” image(left)and the result of applying cumulative-probability-based histogram equalization method to the “mix rose” image(right)

The resulting image of the cumulative-probability-based histogram equalization method is a little smoother and we can improve it using the method I explained in part c)4).

We can also see that because of a sharp edge in the image between dark and bright parts the output image still has a circle shadow of that edge. By applying a low pass filter before the transmission, we smoother the sharp edge and can improve the result.

Discussion

By noticing the output images of the first method we can see there are small blocks of pixels with the same color in each image which can be distinguished from neighbor blocks. This is because at first all this pixels had a very high (or low) gray-scale value and after the transmission they are mapped to a middle gray scale value, at the same time their neighbor block which had a close value to them in the bright (or dark) initial image, now has a gray scale value far from them and so this blocks are visible separately.

By comparing the result of applying transfer-function-based histogram equalization and cumulative-probability-based histogram equalization method to images we can find out that the output of the cumulative-probability-based is smoother (for example notice the background of the vase in each case: transfer-function based(left) and cp based (right))

This is because in the first method there are jumps in histogram while in the second method we don't have any jumps and pixels values are uniformly distributed which leads to a smoother looking image.

Problem 2: Image Denoising

Abstract and Motivation:

Noisy images are created because of different reasons such as malfunction of sensors in a camera. An image may contain different types of noise such as uniform, gaussian, and salt and pepper. Image denoising is an important concept which tries to remove the noise on a image with preserving the information of the image (edges, …) as much as possible. Several methods for image denoising are mean filter, median filter, Bilateral filter, and non-local mean filter.

PSNR value is a good metric for comparing different methods of denoising over each other for a specific noisy image.

Approach and Procedures:

<u>part a)</u>

For this part I have used several methods to remove the uniform noise from an image.

The easiest method is using mean filter. In this approach for each pixel we compute the average of gray scale value of pixels in a neighboring window and assign the result to the pixel.

I have tried mean filter with size 3*3, 5*5, and 7*7, compared the resulting PSNR, and found out a 5*5 window size gives the highest PSNR.

Another method is Gaussian weight filter which is in fact taking a weighted average of neighboring window at each pixel location. I tried different values for N and omega and I got the highest PSNR (25.73) with N=5 and omega=1.2

Bilateral and non-local mean filter are two other filters for removing uniform noise which better preserve edges and lead to a less blur image.

By try and error I reached to these parameters for bilateral filter N=5 Omegac=1.3 and Omegas=200 which gave the PSNR of 25.74

By choosing these parameters for NLM filter I reached a PSNR of 26.56: N=7, Nn=5, h=100, and a=1.4

Part b)

In this part we have a noisy color image as input. Its noise includes both uniform and impulse noise. We can remove the impulse noise by a median filter. This filter takes the median of the gray-scale values of the neighboring window at each pixel as the gray scale value of that pixel.

We can also use the filters explained before to remove the uniform noise. I have tried different filters with different parameters and at last NLM filter gave me the best PSNR value for each channel: 25.38 for red channel, 25.32 for green channel and 25.17 for blue channel. The parameters for the red channel are N=7, Nn=5, h=60, and a=1.9, for the green channel N=7, Nn=5, h=80, and a=1.4 and for the blue channel N=7, Nn=5, h=80, and a=1.7.

Part c)

In this part we must remove shot noise from the image. About the shot noise there is a transformation for each pixel which the resulting image of applying it has a gaussian noise of unit variance which is stated below:

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

So, after applying this transformation, we can use a gaussian low pass filter to remove the noise and then we can use the inverse transform of the initial transformation to reconstruct the image. I use a gaussian filter with N=3 and Omegac=1 and reach a PSNR of 35.43

Results:

a)1) Uniform noise

a)2

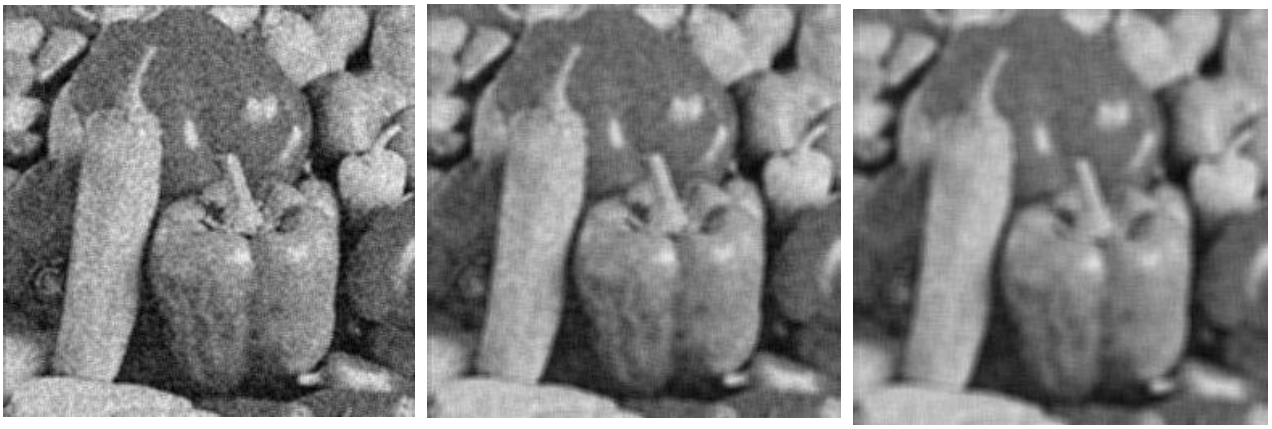


Fig 8: The output filtered image by uniform filter (left image: N=3, PSNR=24.73 middle image: N=5, PSNR=24.87 right image: N=7, PSNR=23.75)

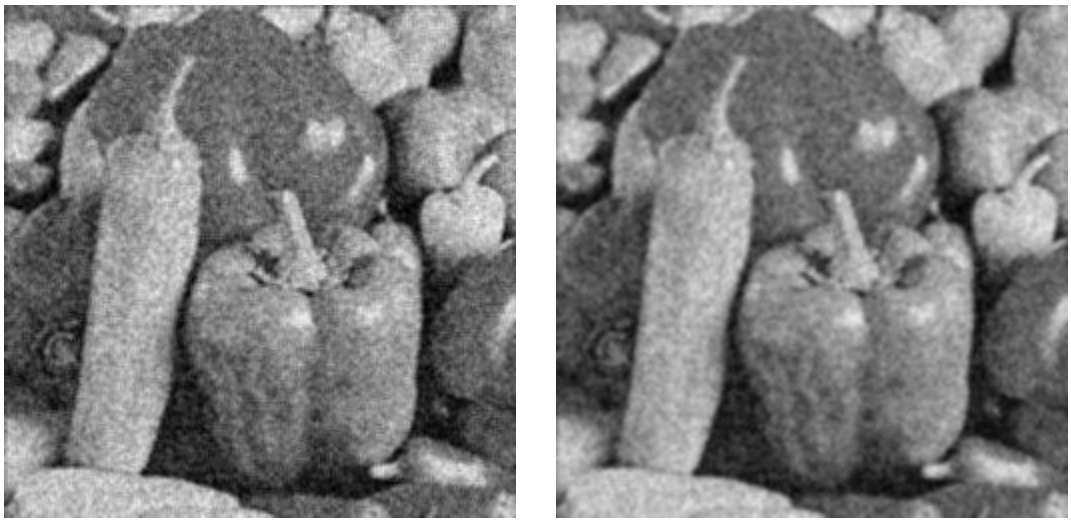


Fig 9: The output filtered image by gaussian filter (upper left image: N=3,O=1, PSNR=24.75
upper right image: N=5, O=1, PSNR=25.62 lower left image: N=3, O=1.5, PSNR=24.8
lower right image: N=5, O=1.2, PSNR=25.73)

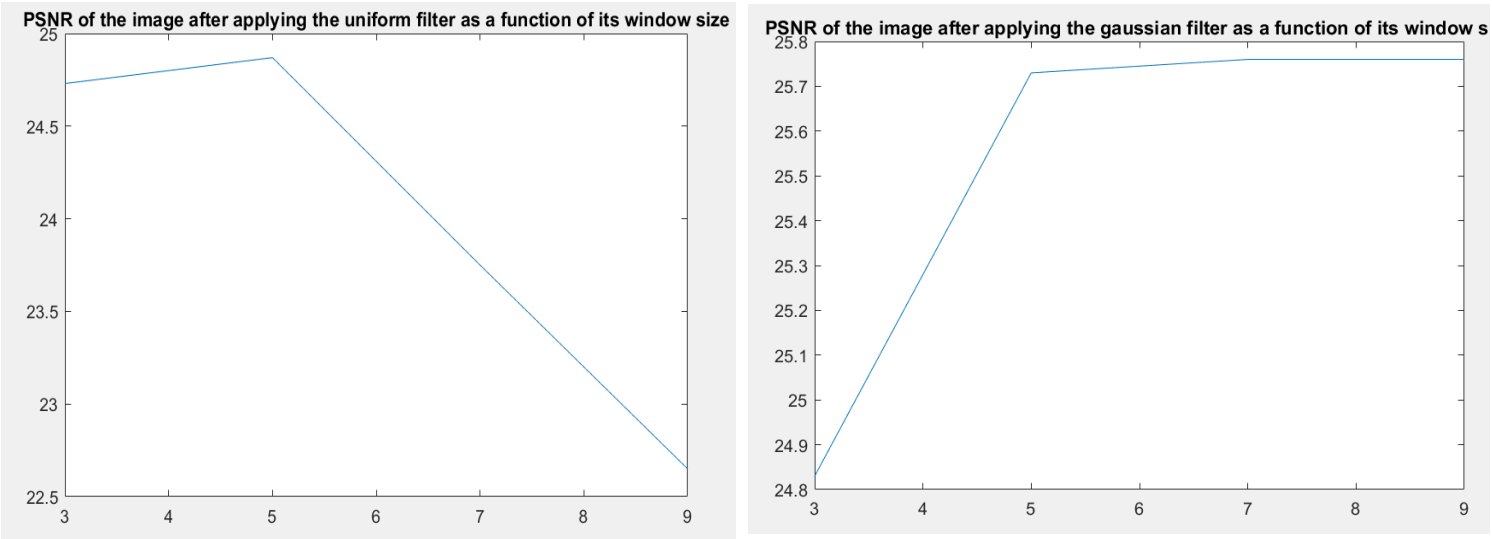


Fig 10: PSNR of the image as a function of its window size (N)

According to the experimental results the gaussian filter can reach a better PSNR and a higher quality output. By increasing the window size in uniform filter after N=5 the PSNR drops and we see a decrease in the quality of the image and the output image will be blur, but in the case of gaussian filter increasing the window size won't decrease the PSNR as the weights of far pixels will be very little in comparison with near pixels.

a)3)

Fig 11: The output filtered image by bilateral filter (N=5, Oc=1.3, Os=200, PSNR=25.74)

a)4)

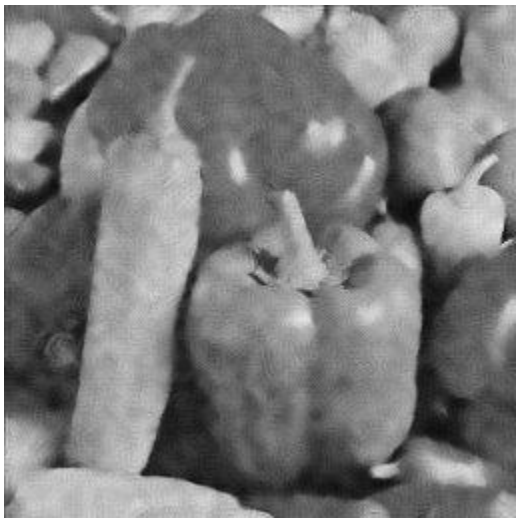


Fig 12: The output filtered image by NLM filter (N=7, Nn=5, h=100, a=1.4, PSNR=26.56)

b)

Fig 13: the original rose image (left) the output filtered image (right)

1) No, for impulse noise there is no need to do it, as the only filter parameter is the window size which N=7 gives the best result for all channels. But for removing the uniform noise NLM filter with different parameters for each channel gives the best result.

b)2) I tried median filter in addition to gaussian, bilateral, and NLM filters and NLM gave the best result: PSNR value for each channel: 25.38 for red channel, 25.32 for green channel and 25.17 for blue channel. The parameters for the red channel are N=7, Nn=5, h=60, and a=1.9, for the green channel N=7, Nn=5, h=80, and a=1.4 and for the blue channel N=7, Nn=5, h=80, and a=1.7.

b)3) No, as these filters are not linear so changing their order will change the output image. If we apply the median filter first, we will have a better result as it removes the invalid pixel values (outlier gray-scale values) at the first place.

b)4) We can check the value of neighboring pixels and if they are far away from the value of the central pixel it is better to ignore them and take the weighted average of other pixels in the neighboring window. In this way we can show the pixels which belong to each side of the edge more clearly.

c) 1)

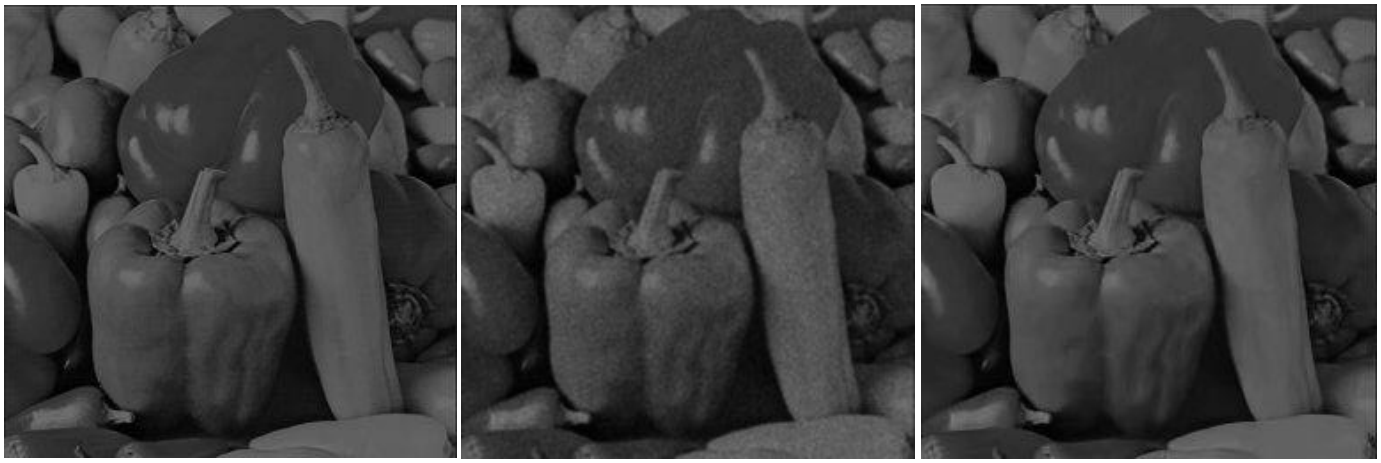


Fig 14: The original image (left, without noise), the denoised version using gaussian filter with parameters N=3 and Oc=.8 (middle, PSNR=35.43), and the denoised image using BM3D filter (right, PSNR=54.33)

2) BM3D is much more effective and gives a very sharp and clear result and a much higher PSNR in comparison with a gaussian filter.

Discussion:

In denoising uniform noise from images, uniform filter is the simplest algorithm which gives the worst results. The resulting image will be blur and this is not pleasant for the viewer.

For this noise NLM filter gives a relatively sharp and clear image but its computation time is high. Especially by increasing Nn the computation time increases significantly, it also has various parameters to set which I have determined them by try and error method, but overall it is not easy to find a local maximum for a function with a lot of parameters.

Bilateral filter has fewer parameters to set and it has a less computational time, so it is easier to use this approach and its output image is also acceptable.

For removing the impulse noise median filter is a simple and also very effective approach and with a reasonable computational time increases the PSNR significantly.

References:

[1] Digital Image Processing: PIKS Inside, Third Edition. William K. Pratt

[2] https://en.wikipedia.org/wiki/Histogram_equalization