EE 569 HOMEWORK 1

Nazim Shaikh

8711456229

nshaikh@usc.edu

01/22/2019

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

I. Abstract and Motivation

a. Image Demosaicing

The need for Image Demosaicing arises from the fact that most digital cameras use color filter arrays over their sensors which usually gives out incomplete color images. To display such raw images, their data needs to be demosaiced.

Image Demosaicing in layman terms, is a process of converting grayscale/raw images into full color images

A practical way to record these full color images is by arranging these color filters in a pattern. This pattern of filters is called Bayer Filter pattern. It is an alternate row of red and green filters with a row of blue and green filters also known as Bayer Array. At each pixel location, sensor is placed which captures one of the 3 primary colors. Other 2 colors are reconstructed based on their neighbor pixel values resulting in full color. This process of interpolating missing color values from nearby pixels is called Demosaicing.

b. Histogram Manipulation

Histogram Manipulation is a technique of adjusting image contrast using its histogram. Here, we try to obtain a homogenous histogram, resulting in an image with optimal contrast (i.e. where all gray values are equally present). Hence, also known as Histogram Equalization. This is accomplished by spreading out most frequent pixel values. As a result of this, pixel intensities are better distributed, and we get a better contrast in areas of low local contrast.

II. Approach and Procedures

a. Image Demosaicing

There are 2 approaches to demosaicing:

1. Bilinear Interpolation: In this method, missing color value at each pixel is interpolated by taking average of 2 or 4 adjacent pixels of same color. For getting adjacent pixel values, 3x3 neighbourhood is considered.
   Image is first extended by 1 row/column to handle boundary pixels. The extended is row/column is computed by reflecting rows/columns just next to image border. Now, using simple averaging each missing color pixel is interpolated from the adjacent pixels within 3x3 region. Once all the pixels are processed, the image is then cropped back to original size to get demosaiced image.
2. Malvar-He-Cutler (MHC) Demosaicing: This is an improved linear interpolation demosaicing algorithm.

In this method, a 5x5 neighbourhood in the form of filters is considered. As shown in homework, there are eight different filters for interpolating the different color components at different locations and missing color value at each pixel is interpolated by convolution with these set of linear filters.
Image extension is done just as in Bilinear, except, image is extended by 2 row/columns to accommodate for 5x5 filter.

b. Histogram Manipulation

There are 2 approaches to this:

1. Transfer function based
    1) Start with obtaining histogram by getting pixel count for each of 0~255 grayscale value.
    2) The histogram is then normalized, and CDF is calculated for each pixel value
    3) Next, mapping rule is created as: x to CDF(x)*255
    4) Once all above is done, old pixel values are replaced as per above mapping rule, resulting in an optimal image.
2. Cumulative probability based (bucket filling)
    1) In this, first the image pixels are sorted in an array in ascending order.
    2) The image array is divided into bucket of size equal to image size/no. of gray scale levels.
    3) Next, for each old pixel we find in which bucket number it falls in.
    4) Old pixel values are replaced with bucket number to give optimal image

III. Experimental Results

a. Image Demosaicing

Fig 1: Original Image	Fig 2: Demosaiced Image by Bilinear Interpolation
	

Fig 3: Malvar-He-Cutler (MHC) Demosaicing

b. Histogram Manipulation

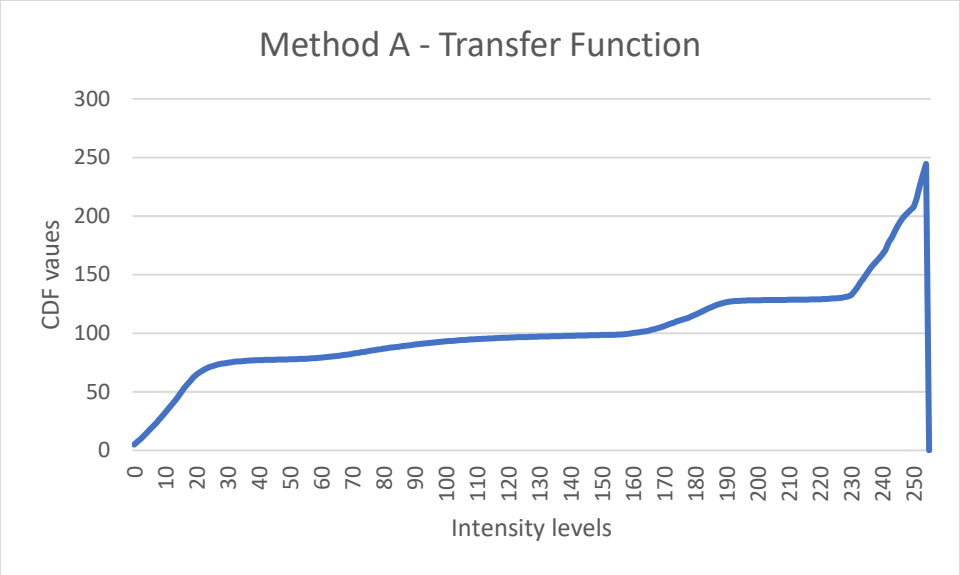
1. Transfer Function based O/P

Fig 4: Rose Dark

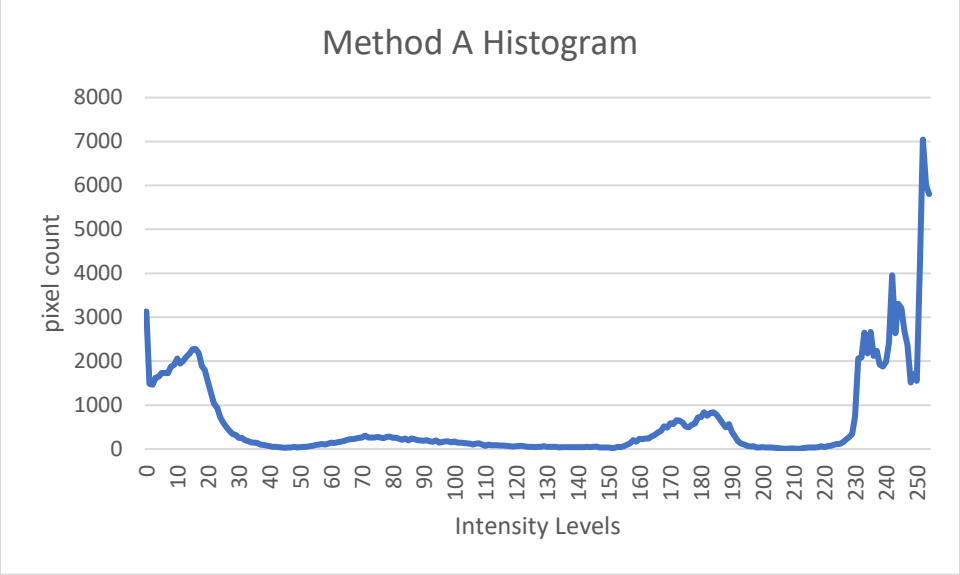
Fig 5: Rose Bright

Fig 6: Rose Mix

Transfer function plot:



Histogram:



2. Bucket Filling O/P

Fig 7: Rose Dark	Fig 8: Rose Bright
	

Fig 9: Rose Mix

Transfer Function plot:

Histogram:

IV. Discussion

a. Image Demosaicing

From section (a) in experimental results part, fig 2 is the output after applying bilinear interpolation and fig 3 is the output after applying MHC.

Do you observe any artifacts? If yes, explain the cause of the artifacts and provide your ideas to improve the demosaicing performance.
Compare the MHC and the bilinear demosaicing results and explain the performance differences between these two algorithms in your own words

- ➔ As seen from the two output, we observe that, although the output of Bilinear is same as original it has artifact such as blur edges. This may be due to zipper effect, caused by improper averaging of neighboring color values across edges. To improve this, we make use of MHC technique.
- ➔ MHC gives clearer image as compared to bilinear interpolation. However, MHC output has small artifact highlighted by green color in fig 3. This may be due to pixel overflow while processing filters across the image.

b. Histogram Manipulation

- ➔ Fig 4-6 shows the result of applying transfer function method and fig 7-9 shows the result of applying bucket filling operation.
- ➔
- ➔ We see that, in both cases the output is the same. Compared to original image, the contrast is fairly balanced in both methods. For rose_mix image as well, we get similar result after implementing method A and method B.

Problem 2: Image Denoising (50%)

I. Abstract and Motivation

Image Denoising is the process of removing noise from image. It is an important preprocessing step for further image analysis. Different types of noise that can corrupt image are Gaussian noise, uniform noise, impulse noise, shot noise and so on. To deal with each of them we have different filters. Here we try to implement Linear filters, Bilateral filter, non-local mean filters etc. on gray-level as well as color images and observe the performance with each of them.

II. Approach and procedure

a) Gray-level Image

The input gray level image is corrupted with uniform noise. It is passed through below filters to remove noise

Linear Filter: Linear filter with uniform function is implemented first. It is the simplest filtering operation, also known as mean filter. It is ideally used to remove impulse noise. The center pixel value is replaced with the average value of its all nearest pixels within the filter region of size N.

Next, Linear filter with gaussian function is used. It is basically a weighted mean filter, where higher weights are assigned to pixels that closer to the center pixel in the filter region of size N

Bilateral Filter: linear filter often leads to degradation of edges. So, to preserve the edges, non-linear filter such as bilinear filter is being used. It is given by as below:

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

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

where (k, l) is the neighboring pixel location within the window centered around (i, j), I is the image with noise, Y is the filtered image. σ_c and σ_s are the spread parameters.

Non-local mean filter: It is similar non-linear filter as above; however, it utilizes pixel from a larger region rather than mean of a local window centered around target pixel. One small block is placed around center pixel and a similar block is placed around the neighboring pixel in the above chosen large region for mean operation. After that, gaussian weighted Euclidean distance between block centered around target pixel and neighboring block is calculated. It is given by as below:

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

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

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

and

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

where *I, Y* are the noisy and filtered images respectively, N'xM' is the size of window centered around location (*x*, *y*), μ is the mean operation, σ is the filtering parameter.

b) Color Image

Here, we take color image which is corrupted by uniform and impulse noise.
First, gaussian linear filter is applied to remove uniform noise which is then followed by median filter to remove impulse noise.
Median Filter: In this, the value of corrupted pixel in noisy image is replaced by median value of corresponding filter window. First, median is calculated all pixel values within the window are sorted in ascending order and then the pixel being calculated is replaced with middle pixel value.

c) Shot Noise

These types of noise are found in the images captured with electronic camera image sensor such as CMOS sensor. There are caused to statistical quantum fluctuations and mainly present in the dark parts of the captured image. Shot noise is Poisson distributed.
To remove short noise:

1. Apply Anscombe root transformation on each pixel z of original image. This will result in additive gaussian noise.

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

2. Then, convolutional gaussian linear filter is applied to remove additive noise.
3. Finally, biased Anscombe inverse transform is applied to denoised image.

$$z' = f^{-1}(D) = \left(\frac{D}{2}\right)^2 - \frac{3}{8}$$

4. At the end, block matching is done and BM3D transformed is applied to get final image.

III. Experimental results

a) Gray-level Image:

Fig 1: Pepper image using linear filter with uniform weight function

Fig 2: Pepper image using linear filter with gaussian weight function

Fig 3: Pepper image using bilateral filter

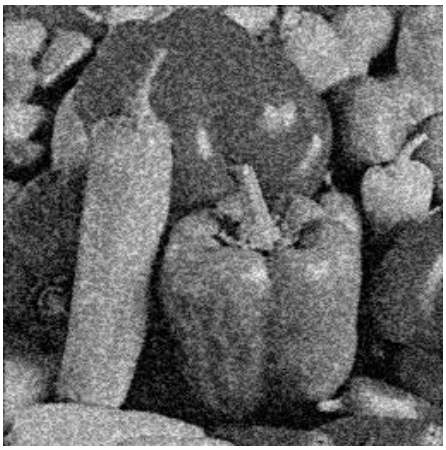
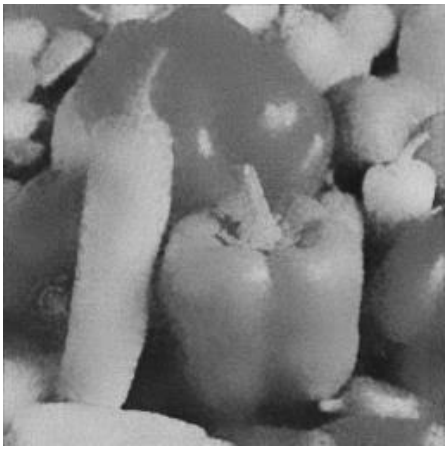


Fig 4: Pepper image using non-local mean filter



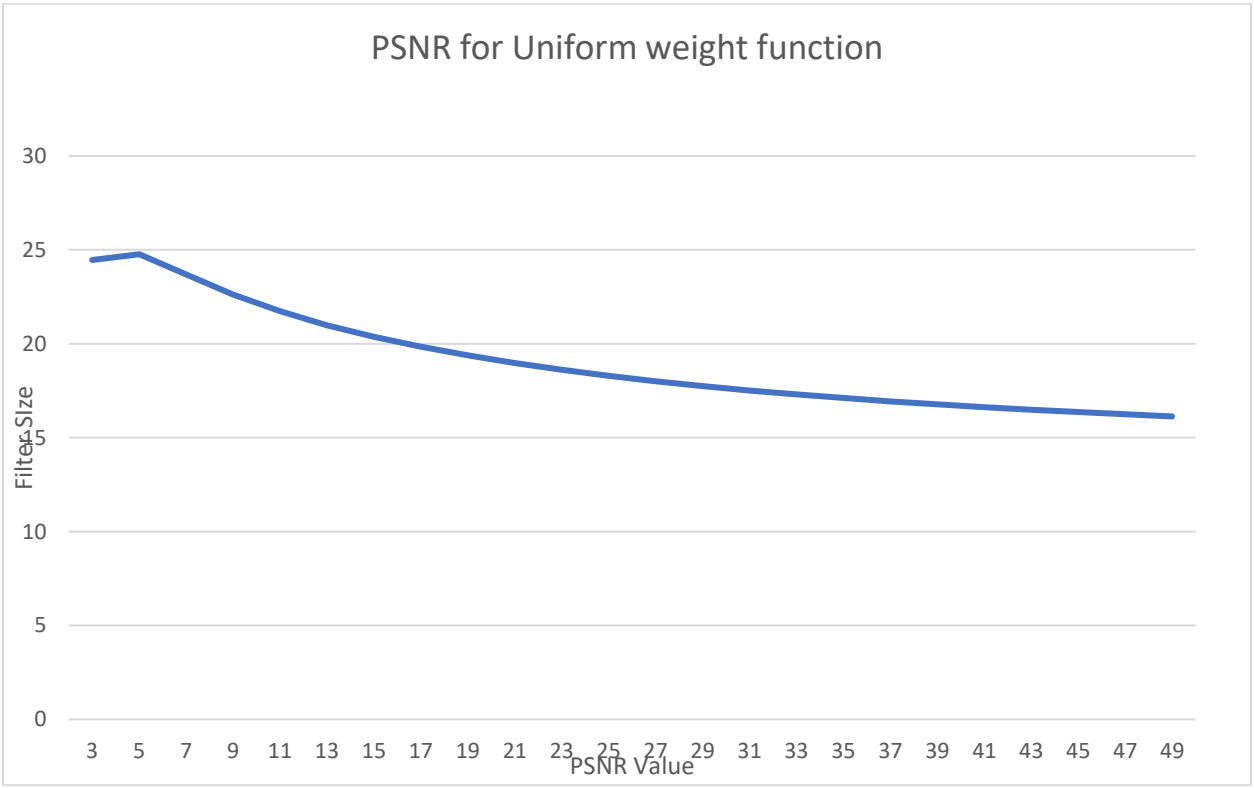


Fig (a): psnr plot (uniform weight)

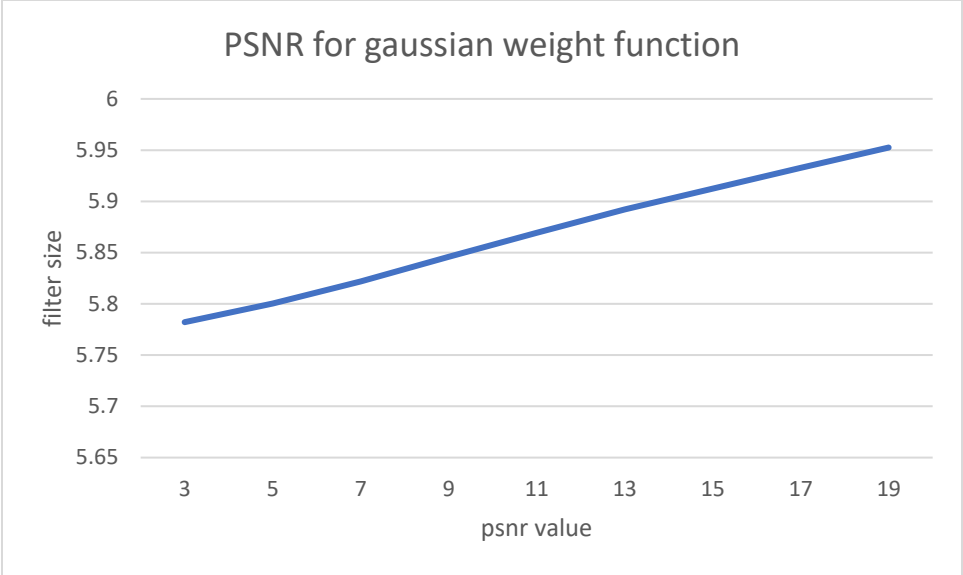


Fig (b): psnr plot (guassian weight)

b) Color Image:

Fig 5: rose image with gaussian and median filter

c) Shot Noise:

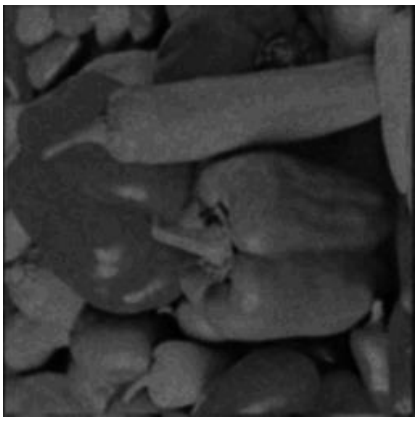


Fig 6: pepper image with shot noise removed

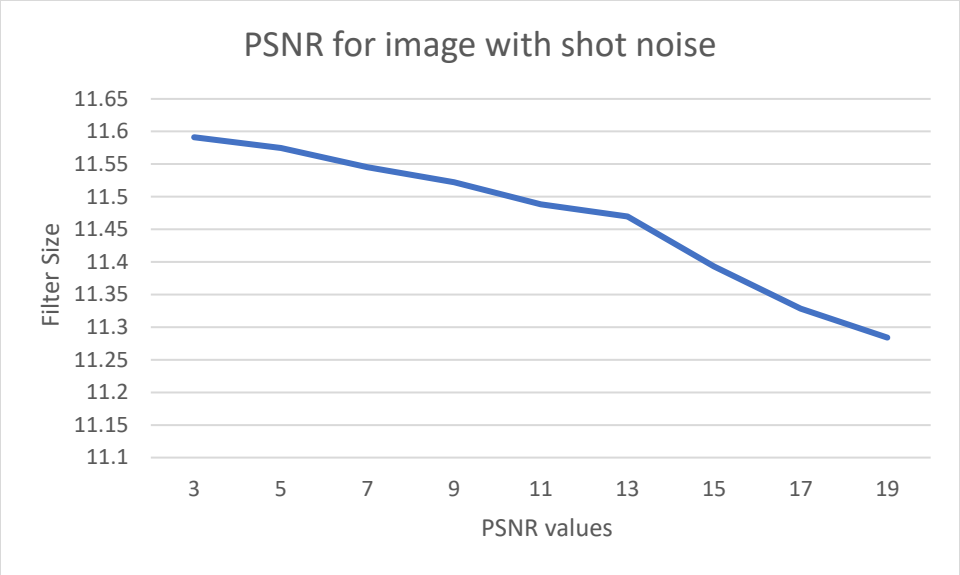


Fig 7: shot noise psnr

IV. Discussion

a) Gray level Image

1) What is the type of embedded noise in Figure 5(b)?

➔ The type of embedded noise applied to the mentioned image is UNIFORM NOISE

Answer to rest of questions:

➔ Fig 1,2,3,4 are noise-removed pepper image using different filters. We observe that in case of linear filters (fig 1, 2), noise is not removed effectively as well as few image details are being lost. Fig 3, 4 is obtained by applying non-linear filter. We see that image obtained in fig 3 is a little better, although the noise is not removed entirely. Fig 4 which is obtained by applying non-local mean filter, gives a clearer image with the noise being removed entirely.

➔ The psnr plot for linear filter with gaussian and uniform function is shown in fig (a) and fig(b). Filter size range used is from 3 – 49 for uniform and 3-19 for gaussian. We see that with low filter size psnr is better. As the filter size decreases, psnr goes down in turn reducing image filtering quality.

b) Color Image

1) Should you perform filtering on individual channels separately for both noise types?

➔ Yes, filtering should be done on individual channels separately for both noise types.

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

➔ Gaussian low pass filter is used to remove uniform noise and median filter is used to remove impulse noise.

3) Can you cascade these filters in any order? Justify your answer.

➔ Yes, since we are removing one noise at a time, order doesn't matter. We can apply median filter first followed by gaussian filter or vice versa. Fig 5. Shows rose image on which gaussian filter is applied first, followed by median filter.

4) 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 your solution to problem 2(a)(2) and 2(b)(2) and

discuss why such solution can potentially work better. (You are welcome to implement the filter, but it is not required)

➔ We can make use of one of non-linear filters such as bilateral or non-local mean as they are able to reduce noise levels without blurring the edges.

c) Shot Noise

Fig 6 shows dark pepper image with shot noise removed. On comparing with original image, we see that most of noise is removed without affecting image details.

The psnr plot for shot noise filtering for different filter size is shown. Filter size range used is from 3 – 19. We see that with low filter size psnr is better. As the filter size decreases, psnr goes down in turn reducing image filtering quality.