EE 569 Digital Image Processing Homework 1

-Lubdha Pimpale

Image Demosaicing and Histogram Manipulation

(a) Bilinear Demosaicing

Motivation:

The aim of a demosaicing algorithm is to reconstruct a full color image (i.e. a full set of color triples) from the spatially under sampled color channels output from the CFA. Bilinear demosaicing is the simplest of all.

Approach:

First, I extended the boundary of image by 1 pixel on all 4 borders considering border layer as mirror. Then, the missing color value at each pixel is approximated by bilinear interpolation using the average of its two or four adjacent pixels of the same color. To give an example, the missing blue and green values at pixel $R_{3,4}$ are estimated as:

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

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

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

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

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

Results:

Discussion:

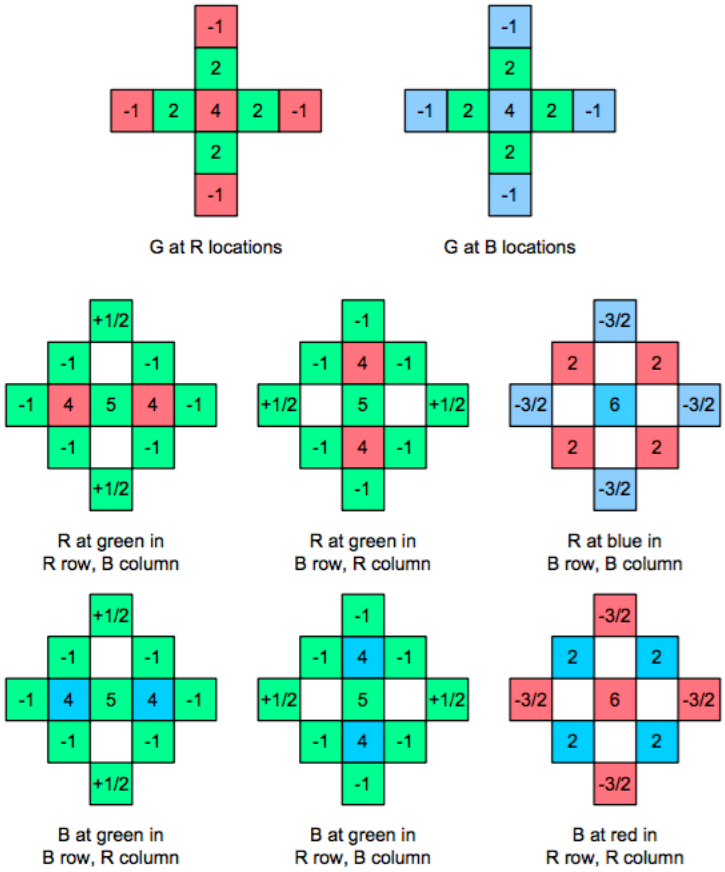
Artifacts like Zipper Effect and False-color Effect are seen.

The zipper effect refers to the abrupt or unnatural changes of intensities over a number of neighboring pixels, manifesting as an “on–off” pattern in regions around edges
The false colors are spurious colors which are not present in the original image. They appear as sudden hue changes due to inconsistency among the three color planes.

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

Motivation: In an attempt to reduce these artifacts, we use **Malvar-He-Cutler (MHC) Demosaicing.** It yields a higher quality demosaicing result by adding a 2nd-order cross-channel correction term to the basic bilinear demosaicing result.

Approach:
We increase boundary by 2 on all boundaries. And use the following filters:



Results:


Discussion:
We can see MHC has better contrast. Zipper effect has almost disappeared.

(c) Histogram Manipulation

Motivation:
Histogram Equalization is used to improve contrast in images.

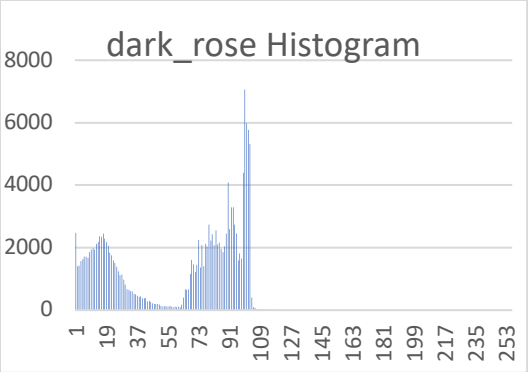
Approach:

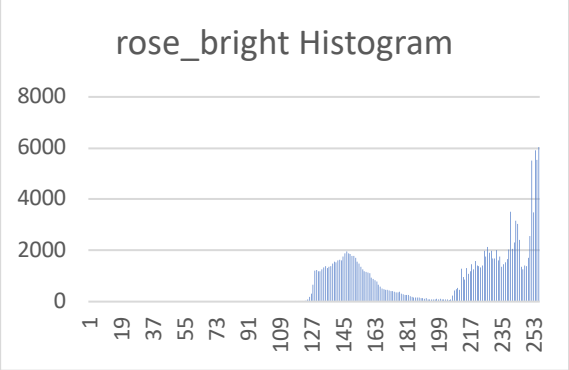
Method A: the transfer-function-based histogram equalization method :
Step 1: Obtain the histogram
■ Count the frequency of pixels of each grayscale value (0~255)
Step 2: Calculate the normalized probability histogram
■ Divide the histogram by total number of pixels
Step 3: Calculate the CDF
Step 4: Create the mapping-table
■ Mapping rule: x to CDF(x)* 255

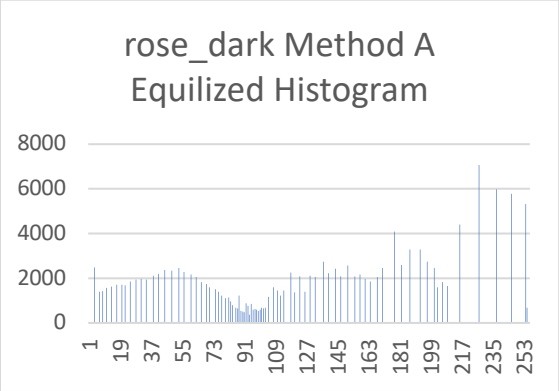
Method B: the cumulative-probability-based histogram equalization method
Step 1: Find out how many pixels go in each bucket(gray scale value)
Step 2: Put them all in an array. Order doesn't matter.
Step 3: Assign each bucket equal number of pixels.
Step 4: Assign new bucket-value to old one.

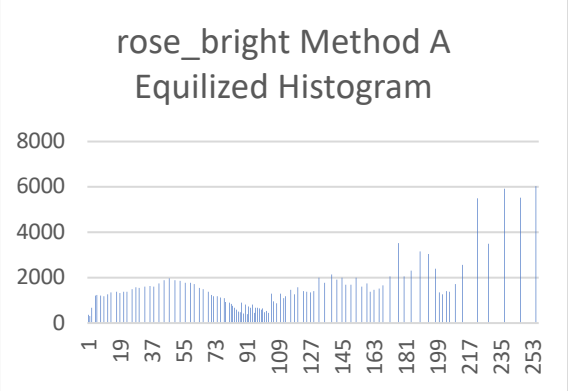
Results and Discussion:

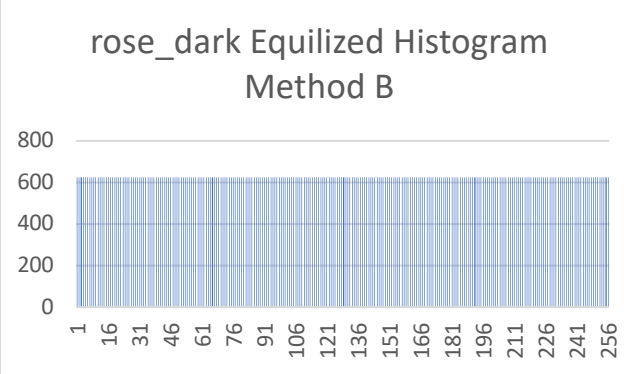
1. Histograms

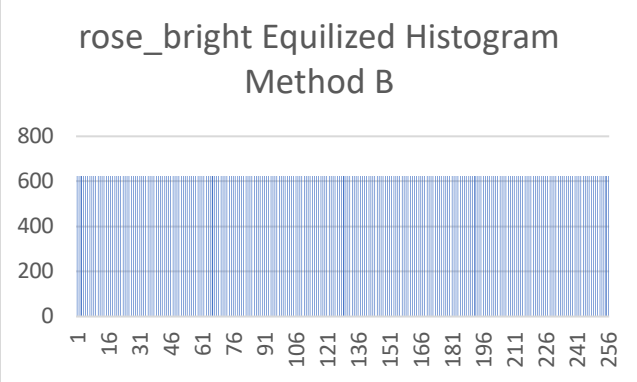

dark_rose Histogram
8000
6000
4000
2000
0
1 19 37 55 73 91 109 127 145 163 181 199 217 235 253


rose_bright Histogram
8000
6000
4000
2000
0
1 19 37 55 73 91 109 127 145 163 181 199 217 235 253

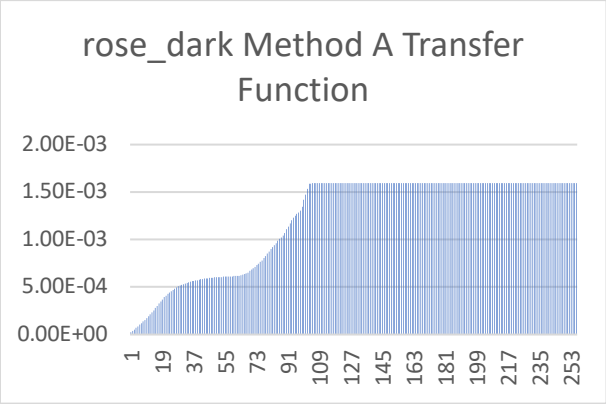

rose_dark Method A
Equilized Histogram
8000
6000
4000
2000
0
1 19 37 55 73 91 109 127 145 163 181 199 217 235 253

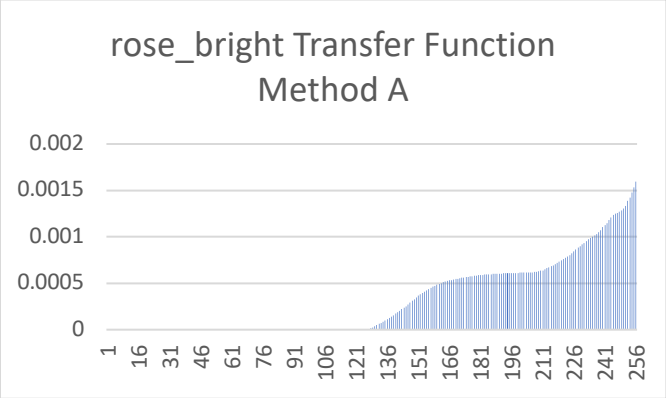

rose_bright Method A
Equilized Histogram
8000
6000
4000
2000
0
1 19 37 55 73 91 109 127 145 163 181 199 217 235 253


rose_dark Equilized Histogram
Method B
800
600
400
200
0
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211 226 241 256


rose_bright Equilized Histogram
Method B
800
600
400
200
0
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211 226 241 256

2. Transfer function and CDF





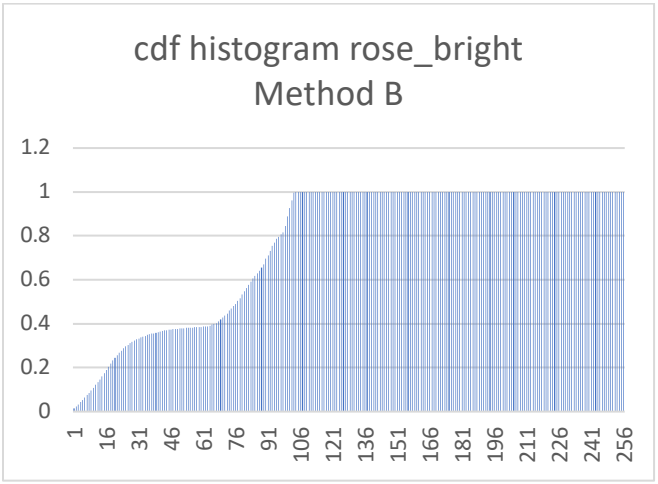
Rose_dark method A

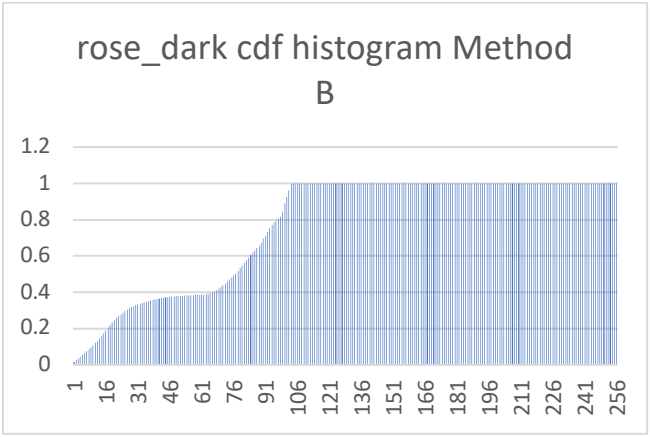
rose_bright method A

3. Method B

Rose_dark method B

rose_bright method B





4. Both look very similar.

5. Rose_mix

rose_mix method A

rose_mix method B

Problem 2: Image Denoising

Motivation:

Low pass filters remove high frequency components. Noise is high frequency. But so is edge. We try different filters to get a good trade-off between removing noise and blurring the image.

Approach:

Uniform Weight function:

- uniform: low pass filter (uniform)

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

$$w(i,j,k,l) = \frac{1}{w_1 \times w_2}$$

where (*k*,*l*) is the neighboring pixel location within the window of size *w*1x *w*2centered around (*i*,*j*)

Gaussian weight function:

- uniform: low pass filter (Gaussian)

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

$\frac{1}{273}$

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

where σ is the standard deviation of Gaussian distribution

Bilateral:

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

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

where σ_c and σ_s are the spread parameters of your choice

Non-local :

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

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

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

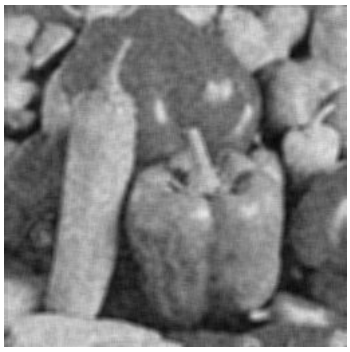
and

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

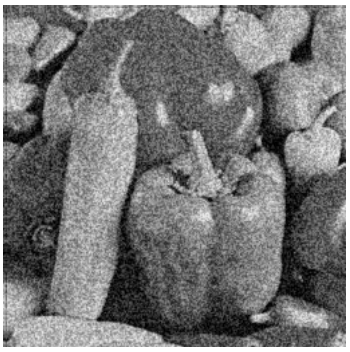
Result and Discussion:

1. 1. Type of embedding noise:
   Uniform.

2. Linear filter:

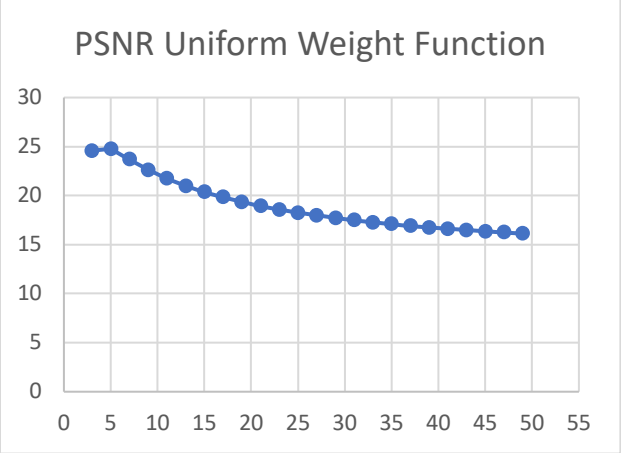


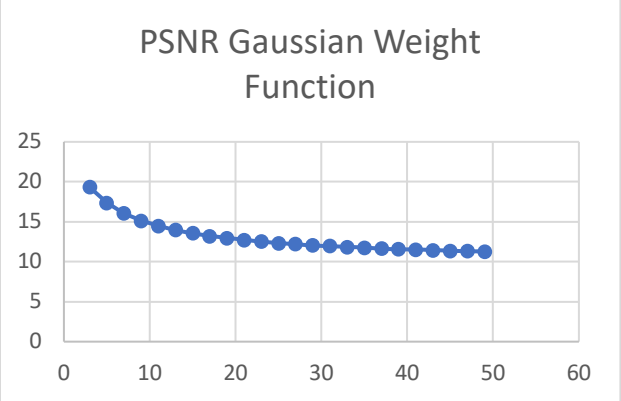
pepper_uni uniform weight function.



pepper_uni gaussian weight function

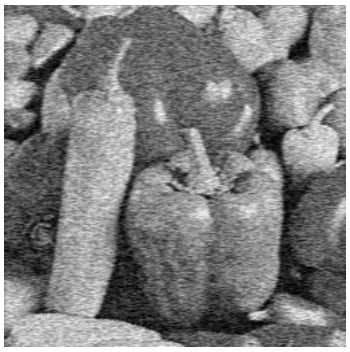
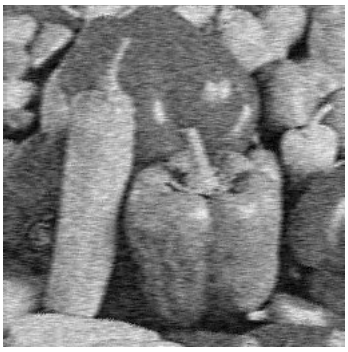
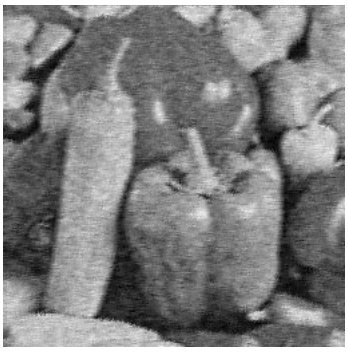
We can see that uniform removes more noise but also blurs the image. Gaussian reduces noise and is better to look at than uniform one.





We can observe from PSNR that 5 is the best window size for uniform. 3 is best for Gaussian

3. Bilateral :

Both Spread^2=1 Spread^2=10 Spread^2= 100

Image starts to get clear and blur as we increase the spread.

1. preserve sharp edges
2. weights depend not only on Euclidean distance of pixels, but also on the difference on the pixel values.

4. Non-local

I chose window of 21x21 and and filter of 5x5 to optimize the time for non-local mean filter.

(b) Color image:

Approach:
I used Gaussian filter and then median filter.
impulse: median filter (non-linear)
1. consider each pixel in the image
2. sort the neighboring pixels into order based upon their pixel values

1. Satisfactory amount of noise is removed.

(c) Shot noise

Result and Discussion:

A very good amount of noise is cleared from the image.