Samwoo Seong
7953-6137-66
samwoose@usc.edu

EE569 Homework #1
January 25, 2020

1.

1.1 Abstract and Motivation

(a), (b) In any digital camera, it can only capture intensity of the reflected light. Therefore, we obtain gray scale image. However, many digital cameras have color filters for red, green and blue before cavity array where it captures reflected light. This form of filters called color filter array and the image with the filter array is called color filter array image as known as CFA image [1].

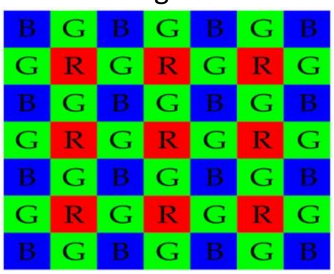


Fig 1. Color Filter Array [2]

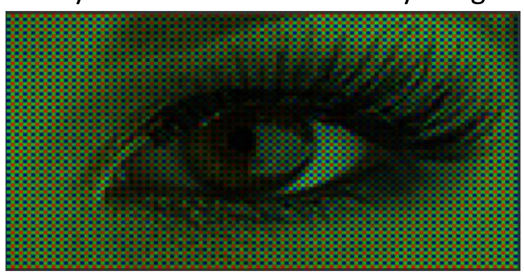


Fig 2. CFA image (a.k.a Bayer image) [3]

With CFA image, intensity value of each color that is not captured in CFA image can be estimated by several color interpolation techniques (i.e., Image Demosaicing) such as bilinear demosaicing and Malvar-He-Cutler (MHC) demosaicing. All these two techniques are based on an idea that unknown intensity value on each pixel should be somehow related to near pixel where its intensity for certain color that is captured by color filter array.

(c) In real life, our environments have sufficient light to be reflected on objects when people take pictures of them especially at night. It may cause a dark image in Fig3.


Fig3. Dark image example [4]

This type of dark image can be enhanced by manipulating intensity histogram of the image. This image results from the fact that its intensity histogram is not uniformly distributed. Therefore, basic idea of the histogram manipulation is to make a non-uniform intensity histogram be a uniform intensity histogram while keep maximum value of cumulative distribution function as 1.

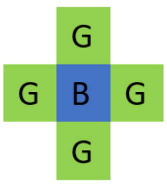
To achieve the goal, two histogram manipulation algorithms are used such as transfer function method and bucket filling method.

1.2 Approach and Procedures

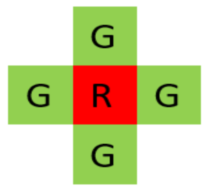
(a)

(a-1) Approach: fundamental idea of this color interpolation algorithm is that unknown intensity for target color can be estimated by neighbor pixels where have intensity value of the target color. In the algorithm it only uses nearest pixels for each color estimation. For each color such as red, blue, and green, the following pixels are used.

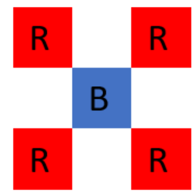
(1) Green at blue



(2) Green at red

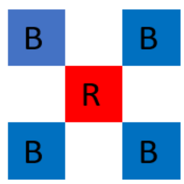


(3) Red at blue



(4) Red at green

(5) Blue at red



(6) Blue at green

For the edge and side, we can extend boundary of our image by size of a filter used and perform bilinear demosaicing algorithm. There are many boundary techniques are available. e.g., zero padding, mirror reflection, etc.

(a-2) Procedure: procedure of bilinear demosaicing is shown below as a flow chart.

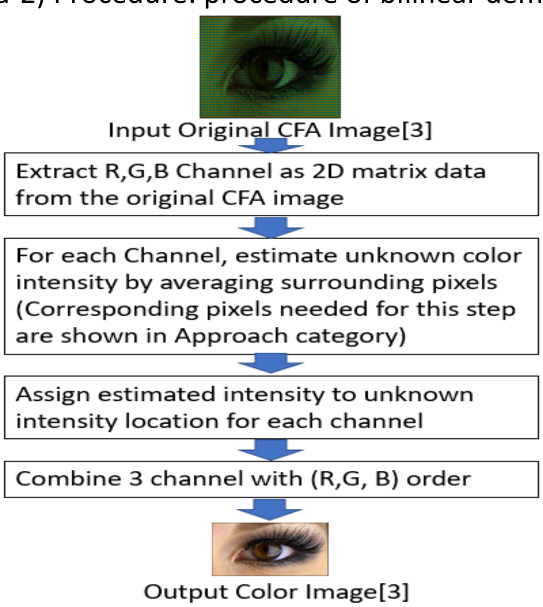


Fig4. Bilinear Demosaicing procedure flow chart

(b)

(b-1) Approach: even though performance of bilinear demosaicing algorithm is pleasant to human eyes, it is common to have artifact that pure color dots occur along side edges. This is because the demosaicing technique performs in each separate color channel meaning it possibly ignore correlation among 3 colors [5]. This effect can be reduced by considering correlation between two color when unknown intensity for each color is estimated. It can be done by adding correction terms to formula for bilinear demosaicing as follows [5].

$$\hat{g}(i,j) = \hat{g}_B(i,j) + \alpha \Delta_R(i,j)$$

Where,

$$\Delta_R(i,j) \triangleq r(i,j) - \frac{1}{4} \sum_{(m,n)=\{(0,-2),\,(0,2),\,(-2,0),\,(2,0)\}} r(i+m, j+n)$$

Formula 1. Green interpolation at an R location [5]

Here, gradient term is called correction term taking into account correlation between green and red. Coefficient alpha decides how much correction term will correct the estimation value. This coefficient is chosen by provided value in this homework.

Similarly, to interpolate green, red, and blue at all possible locations, 8 pre-calculated filters can be applied.

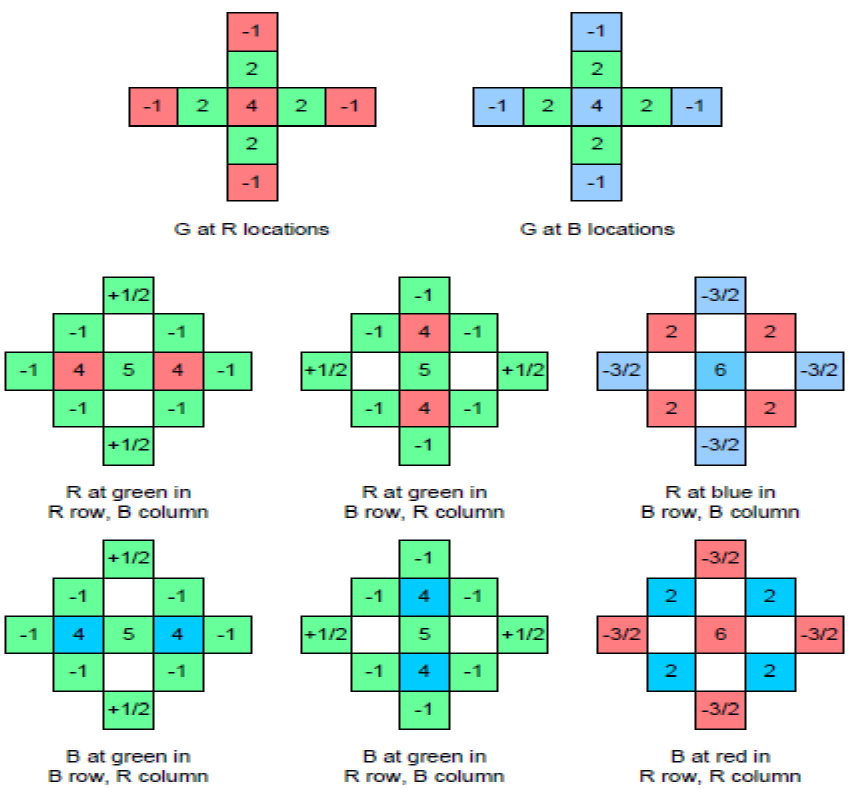
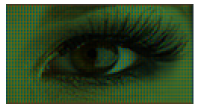


Fig5. 8 filters for MHC demosaicing algorithm [5]

(b-2) Procedure: procedure for MHC algorithm can be summarized by a following flow chart.



Input Original CFA Image[3]

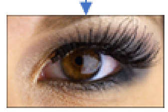
Step1. Pick a filter from 8 filters based on pixel location where I want to interpolate.

Step2.
A = Sum of all values in a chosen filter
B = weighted intensity of center pixel of a filter
C = weighted sum of target color intensity except for center
D= weighted sum of correlated color intensity
$\Rightarrow$ Estimated target color intensity
(1/A)x(B+C+D)

Step3. Apply above step for all pixels of unknown intensity

Similarly, repeat step 1~3 for rest of channels

Combine 3 channel with (R,G, B) order



Output Color Image[3]

Fig 6. Flow chart for MHC demosaicing algorithm

(c)
(c-1) Approach:
(c-1-1) Method A: In this method, probability of each intensity plays a key role. If we consider each intensity value of pixel as random variable, we can change the random variable that has increase value as well as its integration value of probability density function is also one. This successfully increasing intensity values of pixels uniformly not only causes effect that brighten a dark image, but also makes its intensity histogram relatively uniformly distributed.

(c-1-2) Method B: Generally, a dark image has a histogram that has high appearance values of few low intensities. Basic idea of the method is to manipulate this type of non-uniformly distributed histogram to a uniformly distributed histogram. It can be achieved by making all intensities appear same amount of time.

(c-2) Procedure:

(c-2-1) Method A: procedure for method A can be summarized as follows.

Extract R,G,B Channel as 2D matrix data from the original dark image

↓

Step1. Make a storage that has all numbers of intensities that appear in an input image.

↓

Step2. Using the storage from step1, calculate probability of each intensity by dividing the number of appearance by total number of pixels in the input image. Also, store all these probability in an array.

↓

Step3. Using the array from step2, calculate cumulative distribution function by adding up each value from probability of 0 to probability of 255. Save all these CDF in an array.

↓

Step4. Multiply number of gray scale value(i.e., 256) to each CDF of intensity. Save these value in an array. It becomes a transfer function that maps old intensity values to enhanced intensity values

↓

Step5. With transfer function from step4, map old intensity values to enhanced intensity values to output image data. Make sure save them onto corresponding location to locations of old intensity values

↓

Combine 3 channel with (R,G, B) order as a colorful image

Fig 7. Procedure for method A [6]

(c-2-2) Method B: procedure for method B is shown below.

Extract R,G,B Channel as 2D matrix data from the original dark image

Step1. For each channel save a (intensity, row, column) pair that tells us where the intensity is located in original input image in an array. The array will have (width x height of image) pairs.

Step2. For each channel, sort the array by intensity value in ascending order (i.e., 0~255)

Step3. Find three values below
(1)Total number of pixels = width of image x height of image
(2)Number of buckets = number of gray scale values(intensities)
(3)Number of balls per bucket = total number of pixels / number of buckets

Step4. For each channel, assign each intensity as many as number of balls per bucket from array from step2. The number of each intensity will be equal to number of balls per bucket.

Step5. For each channel, using the array from step4, change the intensity at corresponding row and column in image

Combine 3 channel with (R,G, B) order as a colorful image

Fig 8. Procedure for method B [7]

1.3 Experimental Results

Fig 9. Original dog image

Fig 10. CFA dog image

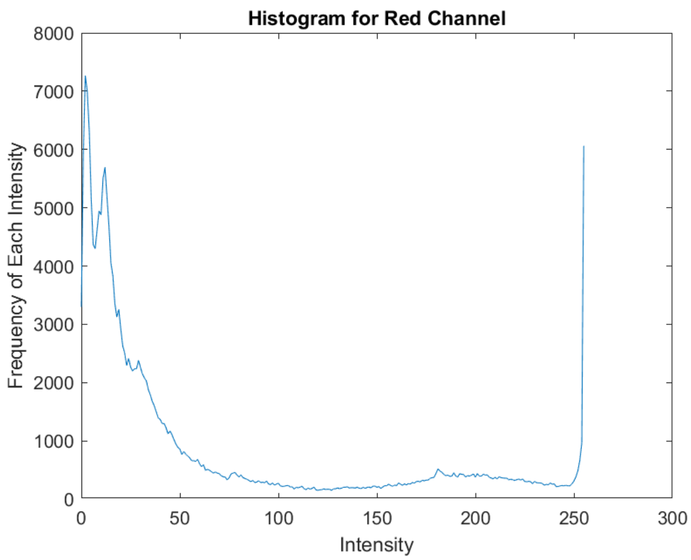
(a)

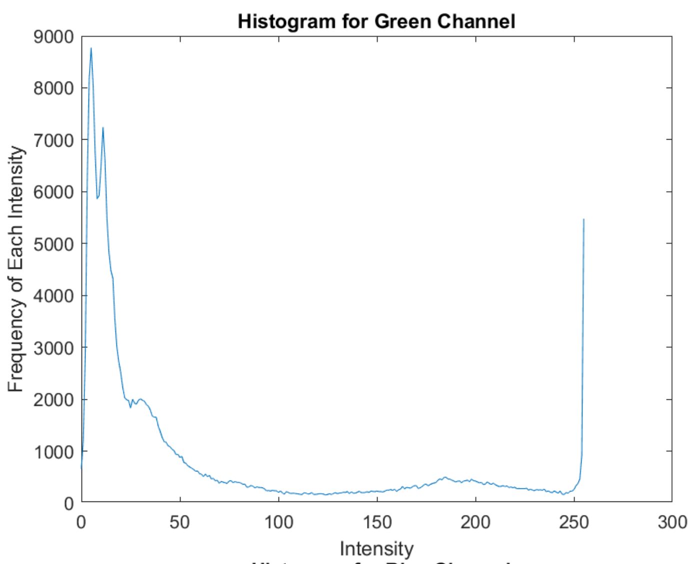
Fig 11. Interpolated dog image by bilinear demosaicing algorithm

(b)

Fig 12. Interpolated dog image by MHC demosaicing algorithm

(c)


Histogram for Red Channel
Frequency of Each Intensity
Intensity
0
1000
2000
3000
4000
5000
6000
7000
8000
0
50
100
150
200
250
300


Histogram for Green Channel
Frequency of Each Intensity
Intensity
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
0
50
100
150
200
250
300

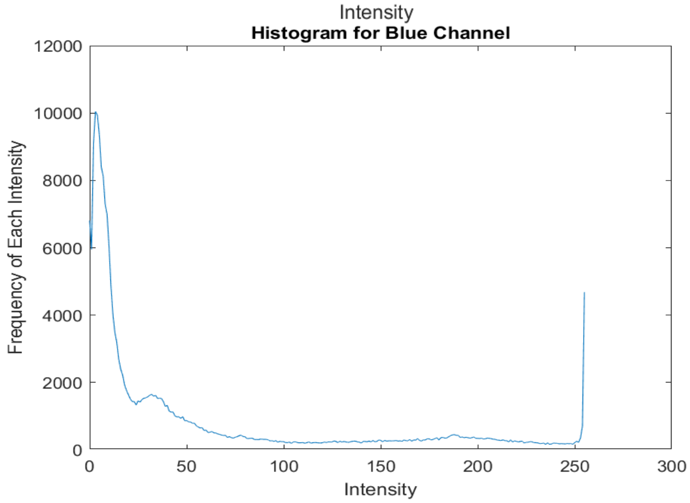

Histogram for Blue Channel
Frequency of Each Intensity
Intensity
0
2000
4000
6000
8000
10000
12000
0
50
100
150
200
250
300

Fig 13-1. Histogram for each channel
(Red Channel (first), Green Channel (Second), Blue Channel (Third) order)

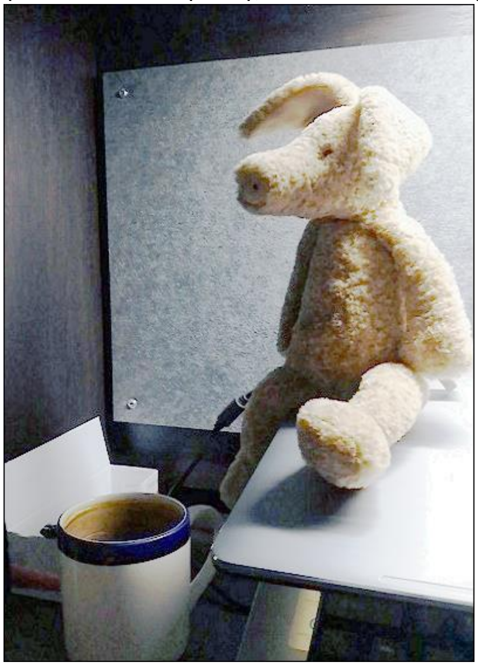
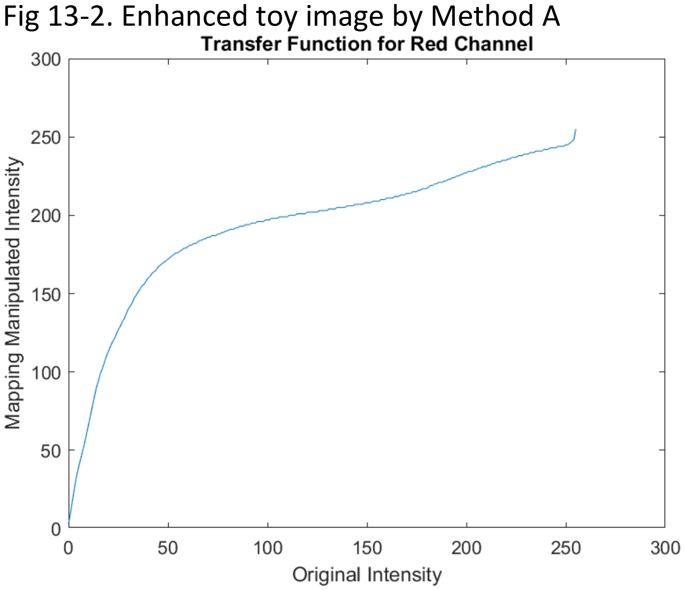


Fig 13-2. Enhanced toy image by Method A



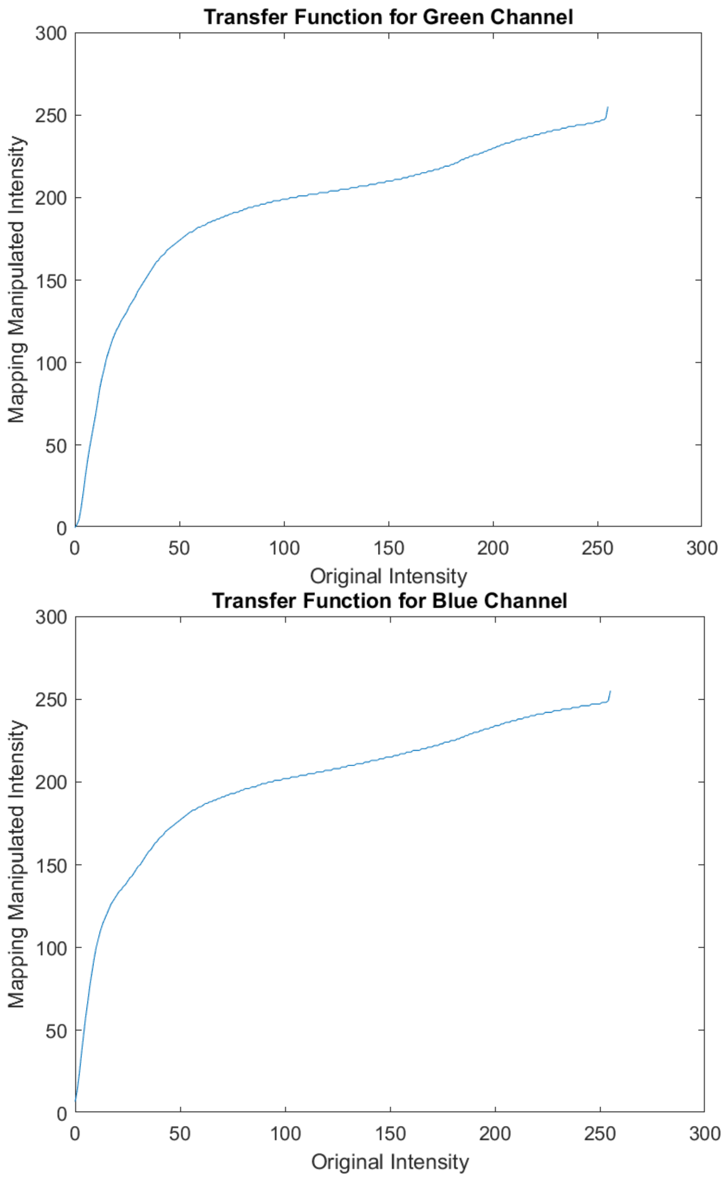


Fig 13-3. Transfer function for each channel
(Red Channel (first), Green Channel (Second), Blue Channel (Third) order)

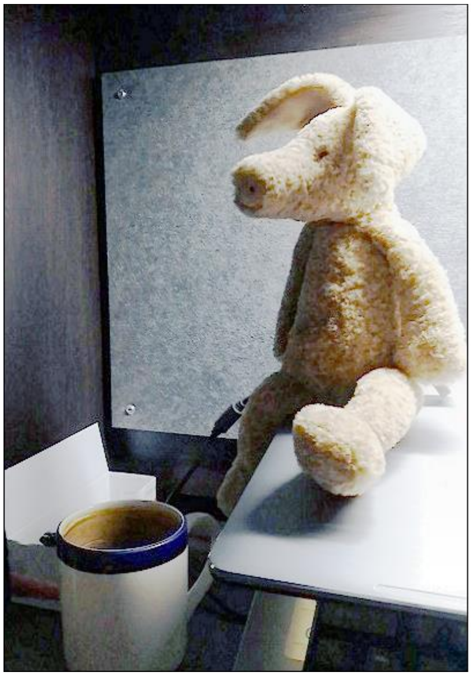
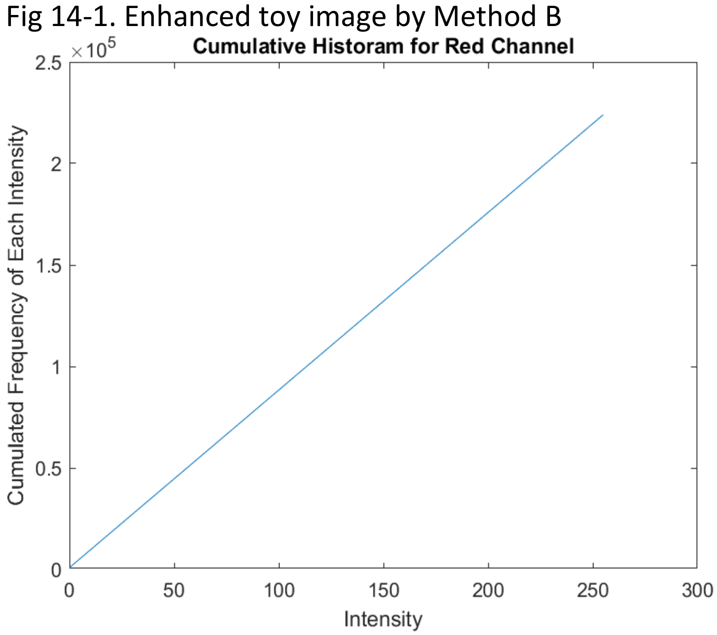
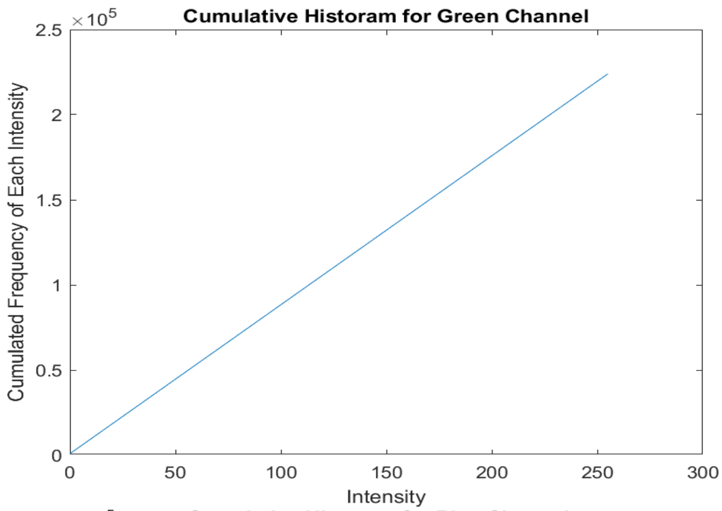


Fig 14-1. Enhanced toy image by Method B





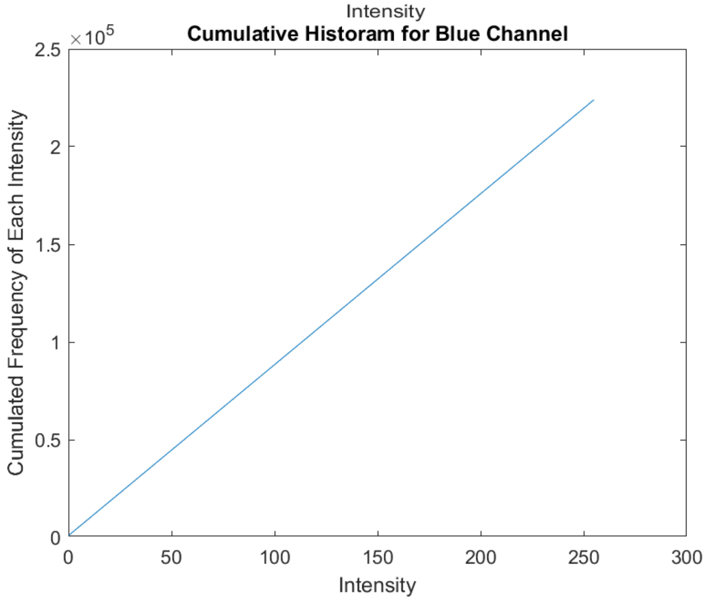


Fig 14-1. Cumulative histogram for each channel
(Red Channel (first), Green Channel (Second), Blue Channel (Third) order)

1.4 Discussion

(a)

Fig 15. Zoomed in interpolated dog image by bilinear algorithm

Fig 16. Zoomed in original dog image.

Even though the interpolated dog image by bilinear algorithm looks pleasant in general, we can observe some artifacts when we zoom in the result as it is shown in figure 15. This happens along edges of objects (e.g., dog's body and back-ground, grass and grass, etc.) or high frequency in image because bilinear demosaicing algorithm does not consider correlation between colors. To reduce these artifacts at edges, we can actually use MHC demosaicing technique which has correction terms to handle correlation between target color and surrounding color.

(b)

Fig 17. Zoomed in interpolated dog image by MHC demosaicing algorithm

We can observe many artifacts that are shown in figure 15 are reduced in figure 17. This is because weight coefficients (i.e. alpha, beta, and gamma) and gradient terms in correction terms play a key role when the algorithm estimate intensity value at edges by including information of non-target color.

(c)

Since fundamental ideas to enhance a dark image (i.e. image with non-uniformly distributed histogram) are the same in both methods, the results from the two methods are quite similar. We would be able to observe huge difference in result images if we have an image that is taken in completely dark place. This is because, not like in method A, in method B, it will always result in same number of balls per buckets regardless condition of an input image.

We can also see some artifacts in terms of color in figure 18. It occurs because each channel is processed separately. This procedure can lead us to this color artifact. To improve result, we can save locations where we manipulate intensities of pixels in one channel and manipulate the same location in rest of channels.

2.

2.1 Abstract and Motivation

(a), (b), (c), (d)

In digital images, when they are produced by digital camera, random noise could be added to the images by several reasons e.g., heat loss, quantization error, etc. This type of random noise can be reduced in image by averaging multiple same images. However, in reality, taking a lot of same pictures is hard to be done in cost sense. This is reason why Mean, Gaussian, Bilateral, Non-Local Mean, and BM3D filters are invented. Fundamental idea of all filters is to use property of the random noise that it will be converge to zero on certain pixels and only noise free intensity value will remain if we average a lot of random noises. To achieve this, different denoising filters use different strategies.

(e)

Beside random noise, there is another type of noise called impulse noise in digital images. This noise is caused due to malfunctioning sensor because of temperature, any type of disturbances in atmosphere, etc [8]. These two noises, random noise as known as uniform/Gaussian noise and impulse noise are highly likely mixed in one image. We can remove this mixed noise from the picture by applying proper filters in certain order.

2.2 Approach and Procedures

(a)

(a-1) Approach:

There are two low pass filters that can be used to remove uniform/Gaussian noise. One is called mean filter and the other is called Gaussian filter. Underlying principle of these filters are quite similar. We will average values of 25 pixels covered by one of filters (i.e., 5x5 size filters are applied to a noisy image) to replace centered pixel. In this way, the calculated value of the pixel ends up with less noise because of property of random noise stated in the Abstract and Motivation category. For each filter, pre-calculated masks can be found with each formula below.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

5x5 Mask

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

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

where (k,l) is the neighboring pixel location within the window of size w_1 x w_2 centered around (i,j), I is the noisy image, Y is the output image.

Fig 18. Pre-calculated mask and denoising intensity formula for specific pixel for mean filter [7]

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

5x5 Mask for Gaussian filter where sigma = 1

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

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

Fig 19. Pre-calculated mask and denoising intensity formula for specific pixel for Gaussian filter[7]

(a-2) Procedure: since mean filter and Gaussian denoising filter can have pre-calculated mask for any size of filter, procedure can be shown in one flow chart. Only difference is a chosen filter for denoising.

Step1. Read raw file image and save image data into 2D matrix
Each (row, column) will represent location of pixel.

↓

Step2. Locate center of a chosen filter to location where target pixel is.

↓

Step3. Perform elementwise multiplication, sum them up, and normalize the sum(i.e. average elementwise multiplication values)

↓

Step4. Repeat step2&3 throughout all pixels in 2D matrix image data(i.e. Perform convolution operation with the chosen filter)

↓

Step5. Save filtered 2D matrix image data as raw file image.

Fig 20. Procedure flow chart for mean denoising filter and Gaussian denoising filter

(b)

(b-1) Approach: as we will see in experimental results below, Gaussian filter has flaw that it could make edge blurry during convolution operation. This is why Bilateral denoising comes up. In this algorithm there is an additional term $\left\|I(N_{i,j}) - I(N_{k,l})\right\|$ and its value can be controlled by sigma s. The role of this term is that balance total value for exponential term which is weight value w(I,j,k,l) to preserve edges. For example, if value of sigma is large total weight become large considering minus sign in exponential term. That means it will try to preserve edges by weighting more value. However, this case PSNR could be lower because denoising term (i.e., $(i-k)^2 + (j-l)^2$), which is governed by sigma c, will contribute less to the weight value meaning it remove less random noise. This leads us to interesting observation. It will be discussed in experimental result and discussion category.

The denoised intensity value can be found by following formula.

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

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

where σ_c and σ_s are parameters of your choice.

Fig 21. Denoising intensity formula for specific pixel for Bilateral denoising filter

(b-2) Procedure: Bilateral denoising process can be achieved by following steps

Step1. Read raw file image and save image data into 2D matrix
Each (row, column) will represent location of pixel.

↓

Step2. For each pixel, apply a provided formula above.

↓

Step3. Repeat step2 throughout all pixels in 2D matrix image data

↓

Step4. Repeat step2&3 for several values of a and h and calculate PSNR to find optimized values of sigma c and s.

↓

Step5. Choose a (sigma c, sigma s) pair that gives the best PSNR

↓

Step6. Save filtered 2D matrix image data that is filtered with the chosen (sigma c, sigma s) as raw file image.

Fig 22. Procedure flow chart for Bilateral denoising filter

(c)

(c-1) Approach: fundamental idea of Non-Local Mean denoising filter is that it tries to find similar regions in an image using large and small neighbor windows and averages them up to reduce random noise. For example, let's assume large window is located at middle of image. Small window starts searching for similar regions within large window and average them up. This process can reduce random noise because of property of random noise. For the similarity measurement while small window is moving through range n_1 and n_2, $\|I(N_{i,j}) - I(N_{k,l})\|_{2,a}^{2}$ term comes to play a role. Since every denoising process for each pixel need pixel values from the input image, it is impossible to find a pre-calculated mask. Formula is provided below.

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

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

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

where $N_{x,y}$ is the window centered around location (x,y), and h is the filtering parameter. $\aleph$ denotes the local neighborhood centered at the origin, $n_1, n_2 \in \aleph$ denotes the relative position in the neighborhood window. a is the standard deviation of the Gaussian kernel.

Fig 23. Denoising intensity formula for specific pixel for NLM denoising filter [7]

(c-2) Procedure: NLM can be performed as follows.

Step1. Read raw file image and save image data into 2D matrix. Each (row, column) will represent location of pixel.

↓

Step2. For each pixel, one large window within image and small neighbor window within the large window will be used. The small neighbor window searches for similar regions throughout area covered by the large window and average them up a using provided formula.

↓

Step3. Repeat step2 for all pixels in the noisy image and save denoised pixel values into same size 2D output matrix at corresponding (row, column).

↓

Step4. Repeat step2&3 for several values of a and h and calculate PSNR to find optimized values of hyperparameter a and h.

↓

Step5. Choose a (a, h) pair that gives the best PSNR

↓

Step6. Save filtered 2D matrix image data that is filtered with the chosen (a, h) as raw file image.

Fig 24. Procedure flow chart for NLM denoising filter

(d)

(d-1) Approach:

This algorithm can be expressed with 2 big steps. At the first step, first, using search window find similar patched in the search window and group them together. Second, take transform onto groups. In frequency domain, high frequency corresponds to noise component in general. To remove this component in frequency domain, we can shrink coefficient with low pass filter. Third, take inverse transform and it becomes a filtered image.

At the second step, using the filtered image from the first step and original noise image, perform Wiener filtering. [9]

(d-2) Procedure: procedure for BM3D filtering can be shown below.

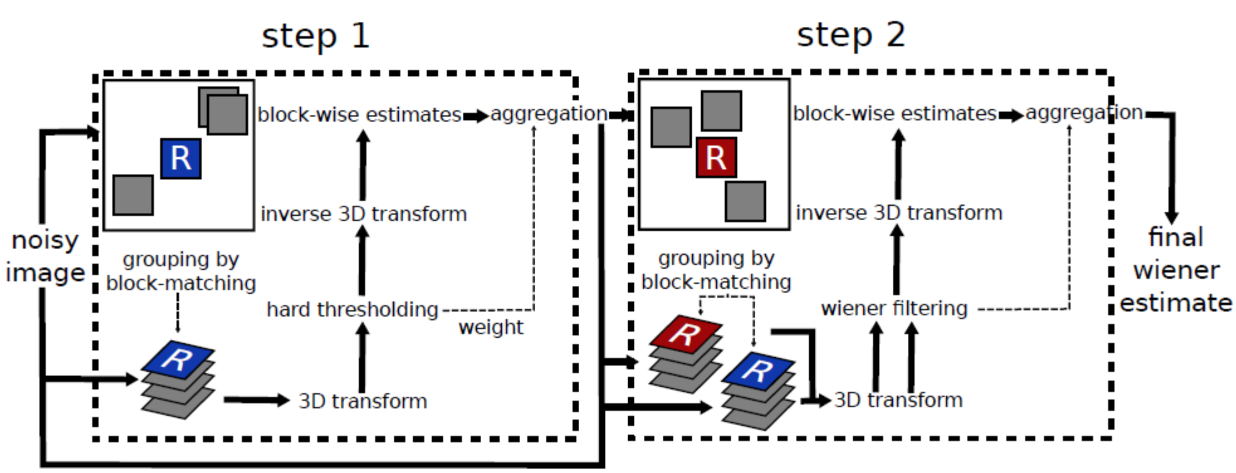


Fig 25. Procedure flow chart for BM3D[9]

(e)

(e-1) Approach: for a mixed noise image (i.e., Mixed noise = impulse noise + uniform/gaussian noise), we can employ two filters in a row for purpose of each noise removal.

(e-2) Procedure: Filtering mixed noise in image can be done by following procedure

Step1. Read raw file image and save image data into 2D matrix. Each (row, column) will represent location of pixel.
↓
Step2. Apply median denoising filter to remove impulse noise for one channel
↓
Step3. Performing BM3D denoising filtering to remove random noise for one channel
↓
Step4. Repeat step2&3 for rest of channels
↓
Step5. Combine 3 channel with (R,G, B) order and save it as raw image file

Fig 26. Procedure flow chart for mixed noise filtering

2.3 Experimental Results

Filter size for mean, Gaussian, and Bilateral filter is 5x5.

(a)

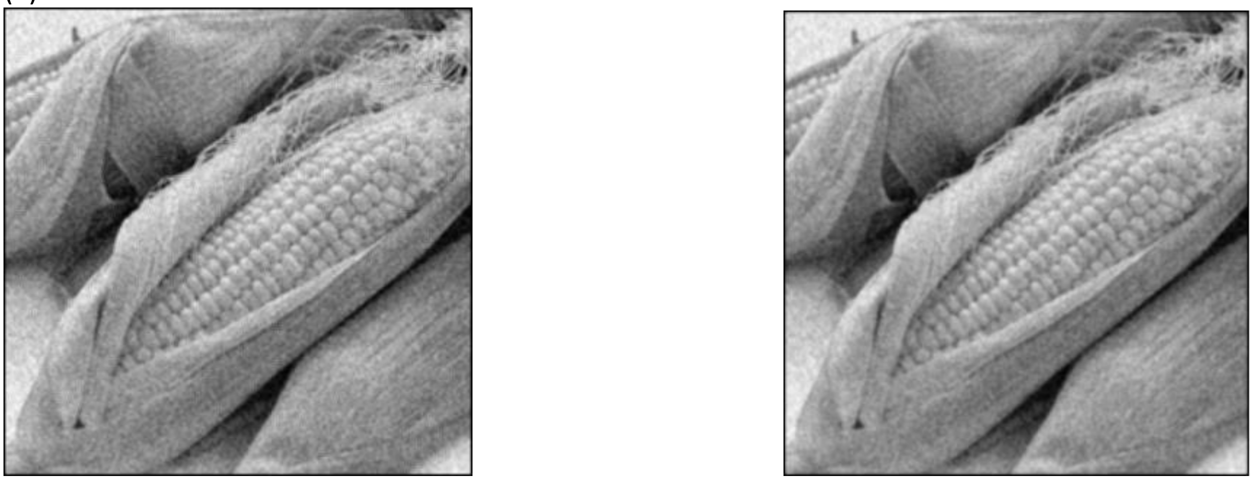


Fig 27. Filtered images (left by mean, and right image by Gaussian, sigma = 1)

(b)

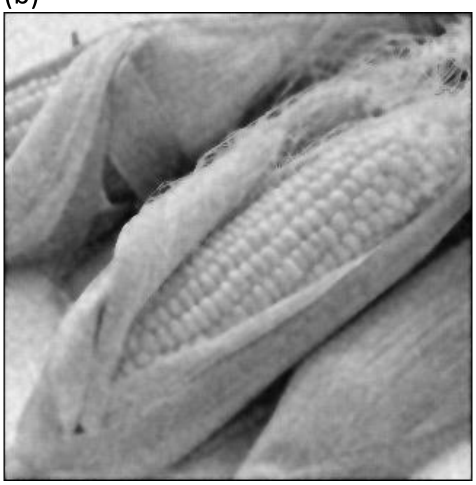


Fig 28. Filtered image by Bilateral denoising algorithm. sigma c=100, sigma s= 100

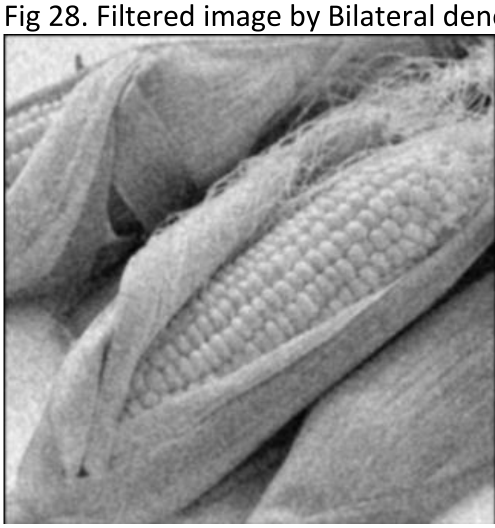


Fig 28-1 Filtered image to explain sigma s role, sigma c = 1, sigma s = 1000

(c)

Fig 29. Filtered image by NLM denoising algorithm. a=20, h=10

(d) (online open source is used)

Fig 30. Filtered image by BM3D. Sigma = 20

2.4 Discussion

(a)To identify what type of random noise is added into image, we can use peak-signal-to-noise-ratio (PSNR) metric. If added noise is uniform random noise, PSNR with mean filter would be higher. Otherwise, PSNR with Gaussian filter would be higher. As a result, PSNR values are 18.5035(dB), and 19.0981(dB) for filtered image by mean filter and filtered image by Gaussian filter respectively. Therefore, we can say added noise could be Gaussian random noise.
As already discussed, image processed by Gaussian filter may look more pleasant in terms of perception. However, it loses some characteristics at edges due to property of edges.

(b)Purpose of this algorithm is to preserve edges while remove the random noise by adding an additional term $\frac{\|I(N_{i,j})-I(N_{k,l})\|}{\sigma_s}$ in weight term. As I explained the roles of σ_s and σ_c, the larger σ_s, the better pleasant result in terms of edge preservation. However, it lowers the PSNR indeed.

In figure 28 and 28-1, $\sigma_s$value is much higher compared to $\sigma_c$in figure 28-1. Even though PSNR somewhat lower than figure 28, it looks better to human eyes. This could happen because of different roles of σ_s and σ_c. Speaking of comparison between results from (a) and (b), depending on value σ_s and σ_c bilateral can perform slightly better.

Some PSNR values are provided below.

	Sigma c= 100 Sigma s=100 (Final parameters)	Sigma c=100 Sigma s=200	Sigma c=100 Sigma s=300	Sigma c=100 Sigma s=400	Sigma c=100 Sigma s=500	Sigma c=100 Sigma s=600	Sigma c=100 Sigma s=700
PSNR (dB)	19.3614	18.8417	18.6676	18.5986	18.5652	18.5466	18.5353

Fig 31. Some PSNR values during optimization process for Bilateral filtering

(c)To choose the best parameter values for a and h, iteration optimization method is used. Range of each value is from 10 to 100. In this method, I repeat NLM filtering process with different pairs of (a, h) and then select a pair (a, h) as optimized parameters that give the highest PSNR. The final parameters are 20, and 10 for a and h respectively.
As we expect, the result image by NLM filtering seems less noisy compared to results from (a) and (b). This is because random noise becomes smaller while similar multiple regions in the large window are averaged. However, edges are relatively blurry because random noise and edges in image, both are corresponding to high frequency component and it is likely reduced during filtering process. Plus, because of concept of the algorithm, if large window size is closed to image size, filtered image become significantly blurry.
Some PSNR values are provided below.

	a=100, h=100	a=50, h=10	a=30, h=40	a=10, h=40	a=20, h=10 (Final parameters)
PSNR (dB)	18.5035	18.5084	18.5041	18.5052	18.5122

Fig 32. Some PSNR values during optimization process for NLM filtering

(d)
Since it is an improved version of Non-Local Mean filter, the result from the BM3D filter seems legitimate compared to result from (a), (b), (c). It preserved edges quite well while successfully removed random noise. However, I could observe the result image become blurry as sigma increase. This algorithm is relatively sensitive to value of sigma when compared to other algorithms.

Some PSNR values are provided below.

	Sigma =10	Sigma =20 (Final parameters)	Sigma =30	Sigma =40	Sigma =50	Sigma =60	Sigma =70
PSNR (dB)	18.34	19.95	19.85	19.69	19.51	19.43	19.37

Fig 33. Some PSNR values during optimization process for BM3D filtering

(e)
We can observe white and black dots are relatively uniformly distributed. Therefore, we can assume there are impulse noise and uniform random noise in the image.
Since we only have denoising filters for one channel, we need to do the filtering process for each channel and combine at the end. More specifically, we can apply two filters in a row to remove impulse noise and uniform random noise. First of all, median filter can be used to remove impulse noise. Second, I will choose BM3D filter to remove uniform random noise because BM3D showed the most pleasant visual perception and highest PSNR in denoising random noise process. Choice of filters for each step can be varying depending on characters of noisy image.

References

[1] CAMBRIDGE in COLOR, “DIGITAL CAMERA SENSOR,” [Online]. Available:

https://www.cambridgeincolour.com/tutorials/camera-sensors.htm

[2] ResearchGate, “Color Filter Array (Bayer Pattern),” [Online]. Available:

https://www.researchgate.net/figure/Color-filter-array-Bayer-pattern_fig1_325772221

[3] RED, “THE BAYER SENSOR STRATEGY,” [Online]. Available:

https://www.red.com/red-101/bayer-sensor-strategy

[4] Catesby Holmes, THE CONVERSATION, “The science of street lights: what makes people feel safe at night,” [Online]. Available:

http://theconversation.com/the-science-of-street-lights-what-makes-people-feel-safe-at-night-103805

[5] H. Malvar, L.-W. He, and R. Cutler, “High-quality linear interpolation for demosaicing of Bayer-patterned color images,” *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*.[Online]. Available:

https://www.researchgate.net/publication/4087683_High-quality_linear_interpolation_for_demosaicing_of_Bayer-patterned_color_images

[6] Santosh Poudel, “Histogram Equalization,” [Online]. Available:

https://www.youtube.com/watch?v=PD5d7EKYLcA

[7] Zhiruo Zhou, “2020_EE569_Discussion_1&2,” [Course Material]. Available for students in the course:

https://courses.uscden.net/d2l/le/content/17205/viewContent/297015/View?ou=17205

[8] J. Harikiran, B. Saichandana, and B. Divakar, “Impulse Noise Removal in Digital Images,” *International Journal of Computer Applications*, vol. 10, no. 8, pp. 39–42, Oct. 2010. [Online]. Available:

https://pdfs.semanticscholar.org/f04b/23a8d12faac337ad2a7a533d8612a350f2cf.pdf

[9] Marc Lebrun, “An analysis and Implementation of the BM3D Image Denoising Method”, Aug. 2012. [Online] Available:

https://www.ipol.im/pub/art/2012/l-bm3d/article.pdf