Name: Shreya Kate

USC ID : 2334973997

Email : shreyak@usc.edu

Submission Date: 01/28/2020

HOMEWORK #1

Issued: 01/13/2020 **Due: 01/28/2020**

Problem 1:

(a) Bilinear demosaicing

Abstract:

In this, the 8-bit gray image of the dog was to be converted into a 24-bit color (RGB) image by using the concept of bilinear demosaicing.

Procedure:

The procedure used to do so is as follows:

1. Boundary extension

First, 4 additional rows were added to the image (2 at the top and two at the bottom). 4 additional columns were also added (2 to the left and 2 to the right of the image). This boundary extension was done so that the pixels at the corners and edges would have information of the neighboring pixels to get a proper color RGB image. The rows and columns that were added were taken to be reflections of the 1st, 2nd, last and second last rows and columns of the gray image.

2. Equations to get RGB information at all pixel locations

The sensors capture only one color at each pixel location. To get the information of all three colors (Red, Green, Blue) at all pixel locations, the information of the other two colors is taken from the neighboring pixels.

The Bayer array is arranged in the following manner:

G R G R G R G R

B G B G B G B G

G R G R G R G R

B G B G B G B G

Therefore, if Red pixel is at the center, the pattern is like this:

B G B

G R G

B G B

The Red pixel will get the information of Green and Blue pixels from the neighboring locations.

The red component will be the pixel itself.
The blue component will be (1/4)*(image(i-1,j-1) + image(i-1,j+1) + image(i+1,j-1) + image(i+1,j+1))
The green component will be (1/4)*(image(i,j-1) + image(i-1,j) + image(i,j+1) + image(i+1,j))

Similarly, I generated equations for when the Green and Blue pixels will be at the center.

Results:

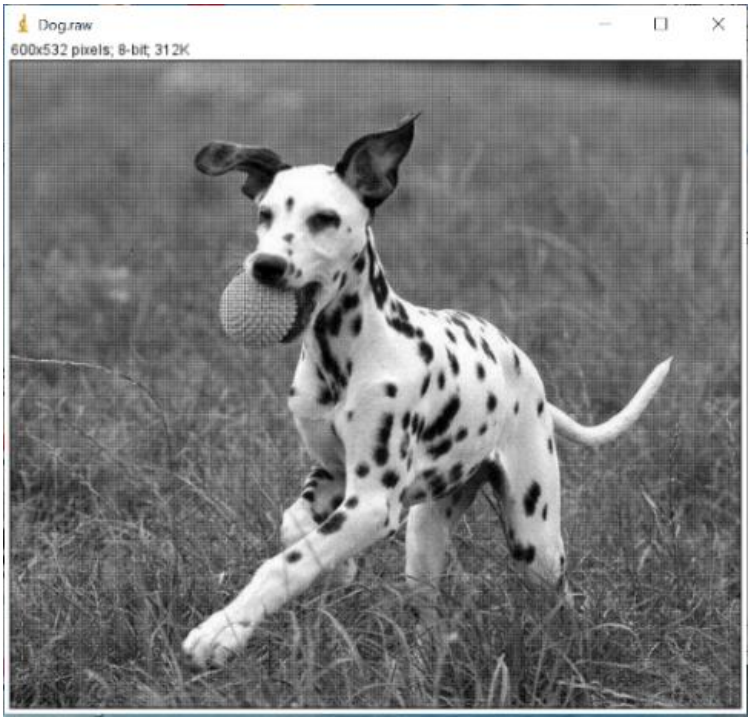

Fig. 1: Dog.raw (gray image)

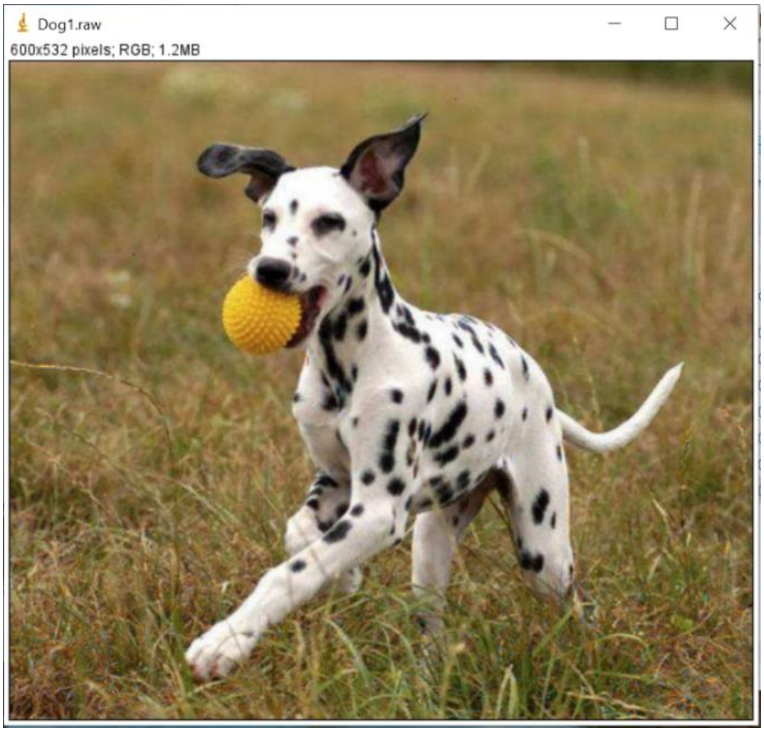

Fig. 2: Dog1.raw (after bilinear demosaicing)

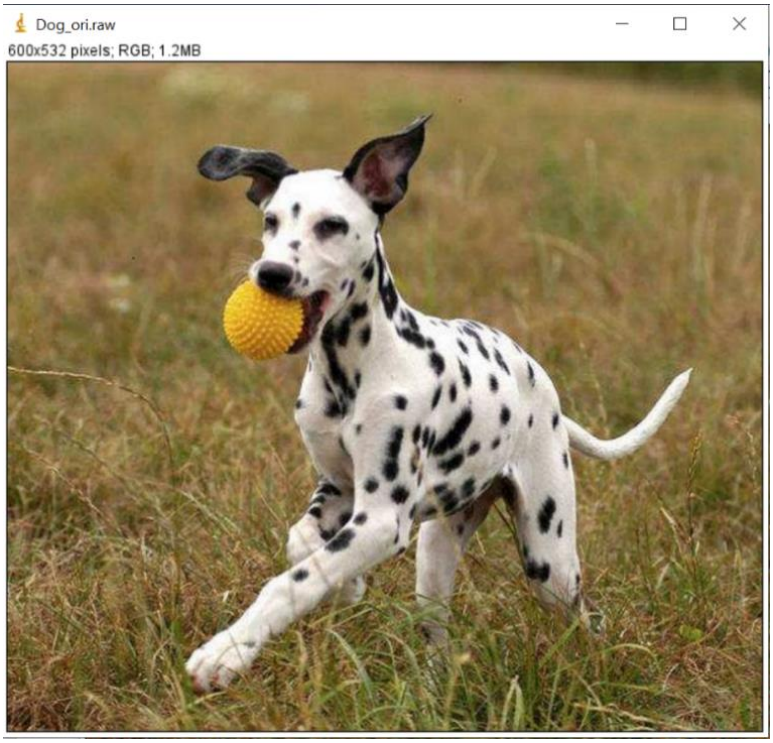

Fig. 3: Dog_ori.raw (original color image)

Discussion:

Comparing the demosaiced image with the original color image, certain artifacts can be observed. The original image is clearer.

The spokes on the ball of the original image are clearer than on the demosaiced image. The blades of grass can also be seen more clearly in the original image. The original image has slightly brighter colors too.

The cause of these artifacts is that we have used only the 4 neighboring pixels while demosaicing the image. If we use more number of neighboring pixels to get the information of R,G,B components then the demosaiced image will give better results (a more clear and brighter image).

(b) MHC demosaicing

Abstract and procedure:

MHC demosaicing adds a correction term to the equations of bilinear demosaicing. It involves eight filters with which convolution is done on the gray image depending on which pixel (R, G or B) is at the center and what its neighboring pixels are.

Results :

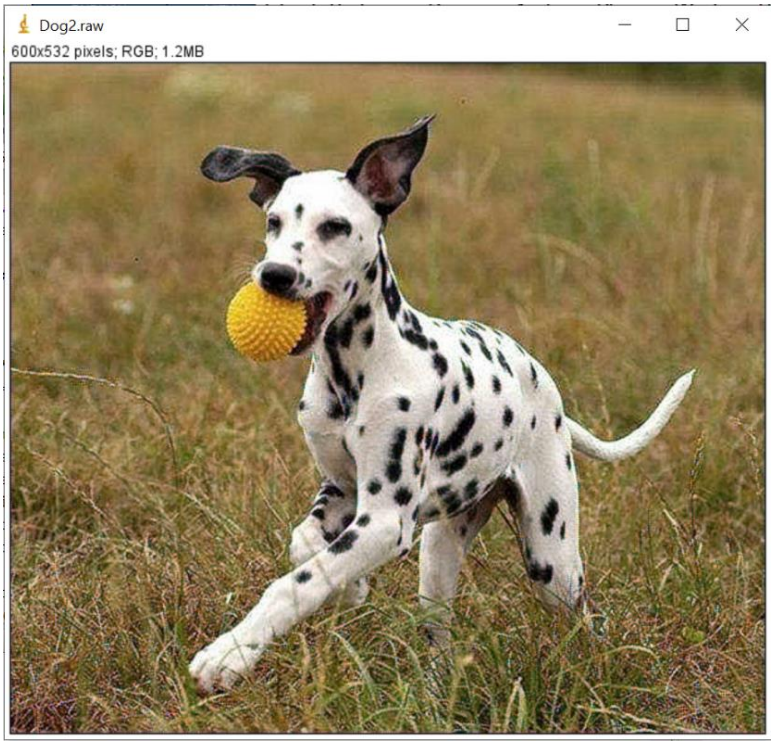

Fig. 4: Dog2.raw (after MHC demosaicing)

Discussion:

After looking at the results of bilinear demosaicing and MHC demosaicing I could observe that the resultant image of MHC demosaicing is clearer and the colors are brighter in MHC demosaiced image.

The grass in the lower half of the image is evidently more detailed in the MHC demosaiced image.

This difference occurs due to the correction term added in the MHC demosaicing equations.

I feel like the missing color components in the bilinear demosaicing have been added/corrected in MHC demosacing by the correction term.

(c) Histogram Manipulation

Abstract and procedure:

The intensity of the pixels vs no. of pixels is plotted in the histogram of the red, green and blue channels. As we can see in the histogram of the given Toy.raw image, the number of pixels having intensities near 0 is very high as compared to the pixels having intensities near 255. This means that the image is dark since 0 means black and 255 means white.

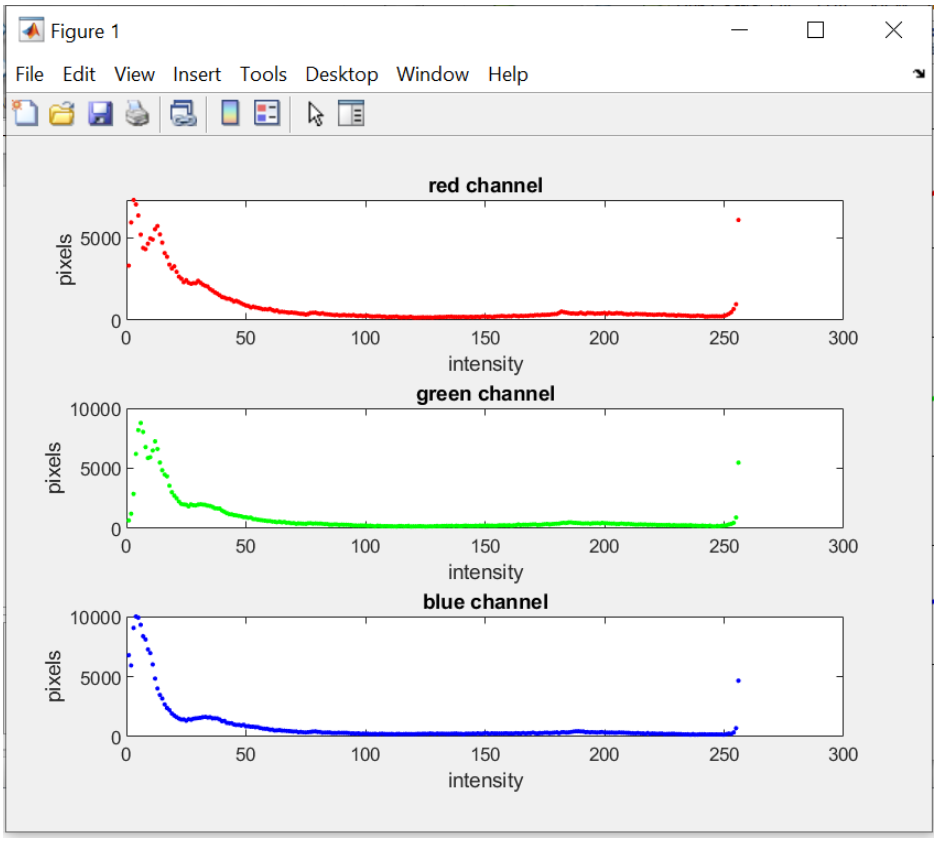

Fig. 5: Histogram of red, green and blue channel

Method A: Transfer function based histogram equalization method

In this method the intensities of the image are equally divided between 0 to 255 to enhance the image so that the image is brighter, and we can see the objects in the image and identify them.

In this method, I first found the probabilities of each of the intensities.

Next, I found the cumulative probabilities of the intensities.

After that, I multiplied the cumulative probabilities by 255 (L-1), L being the number of intensity values. L = 256. I mapped the pixels and the intensity values so that the intensity values of the pixels could be changed in histogram equalization.

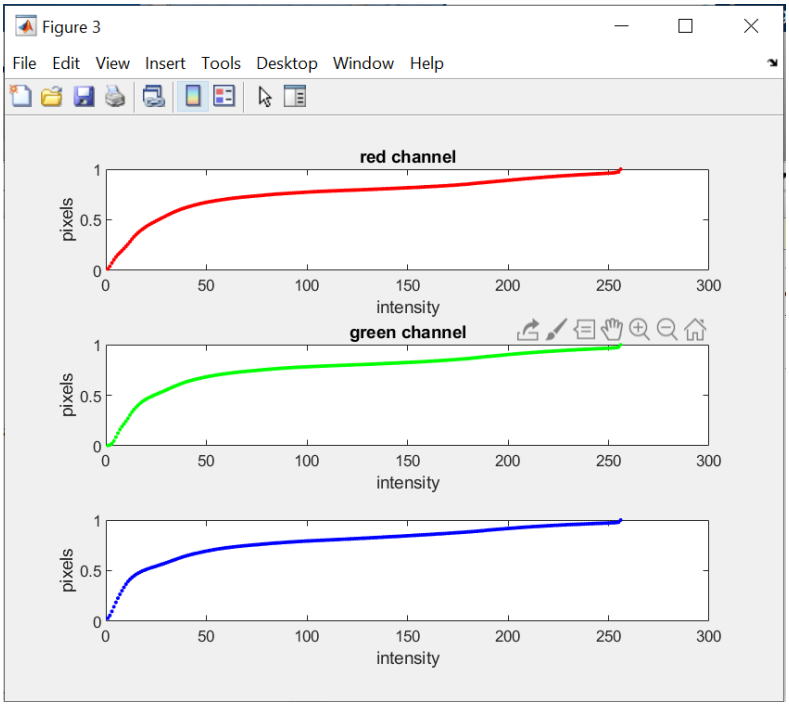

Fig. 6: Transfer function, CDF of red, green and blue channel

Method B: Bucket-filling method

In this method we aim to have equal number of pixels at all the intensity values from 0-255. The number of pixels in the Toy.raw image is 2,24,000 pixels. Therefore, each intensity will have 2,24,000/256 pixels (875 pixels).

We will start with 0 intensity and fill 875 pixels in it. After 875 pixels have been assigned to 0, we will move to intensity 1 and start assigning intensity value 1 to the next 875 pixels. If the number of pixels of intensity 0 are originally less than 875 then we will assign the intensity value of 0 to the pixels that had intensity 1. We do this so that each intensity value has equal number of pixels.

The equal distribution of pixels between all intensities enhances the image and makes the image clearer and brighter.

The equal distribution of the pixels across all intensities can be seen in the histogram plot below: in Fig. 7

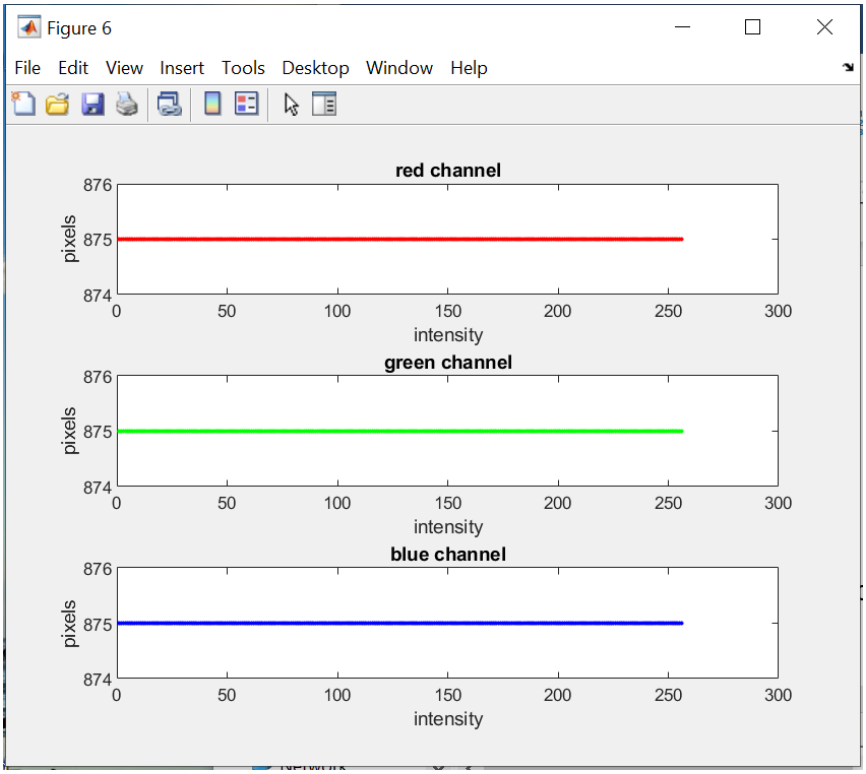

Fig. 7: Histogram of red, green, blue channels after bucket-filling method

Results:

Fig. 8: Toy.raw

Fig. 9: Toy1.raw (enhanced image by Method A)

Fig. 10: Toy2.raw (enhanced image by Method B)

Discussion:

After histogram equalization using method A the enhanced image can be seen in Fig. 9. As we can see, the image is brighter and the objects in the image can be seen clearly.

After histogram equalization using method B the enhanced image can be seen in Fig. 10. Here, we notice that the image is brighter and has truer colors than what were seen after using Method A.

The enhancement can be improved if there is some contrast on the right-hand side of the image. The colors can be seen better if the intensity of bright pixels was lower.

Problem 2:

(a) Linear and Gaussian denoising

The type of embedded noise in Figure (Fig. 11 in the report) is salt and pepper noise. This is because the image has dark pixels in bright locations and bright pixels in dark locations.

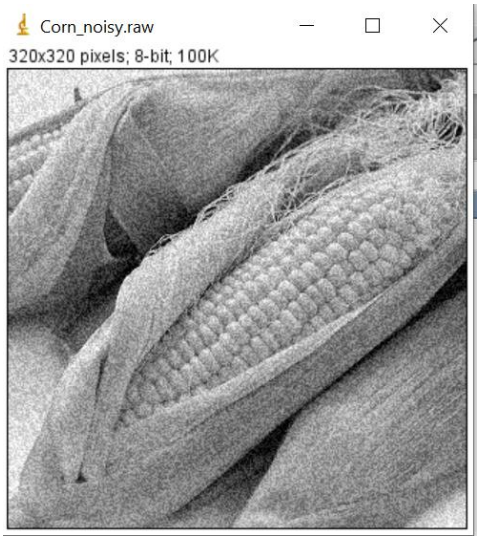

Fig. 11: Corn_noisy

1. Linear Filter

To removed noise using linear filter, I applied a 3x3 linear filter on the image. That is, I did convolution using a 3x3 linear filter. The linear filter looks like this:

1 1 1

1 1 1

1 1 1

After convolution, the denoised image (Fig. 12) was obtained and then I subtracted the denoised image from the original image to get the noise (Fig. 13). The noise was uniform.

2. Gaussian filter

I used a 5x5 gaussian filter to denoise the image using gaussian filter. I created the gaussian filter by using its equation:

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

The filtered image using gaussian filter can be seen in Fig. 13.

The filtered image subtracted from the noisy image gives the image of noise as can be seen in Fig. 15.

Results:

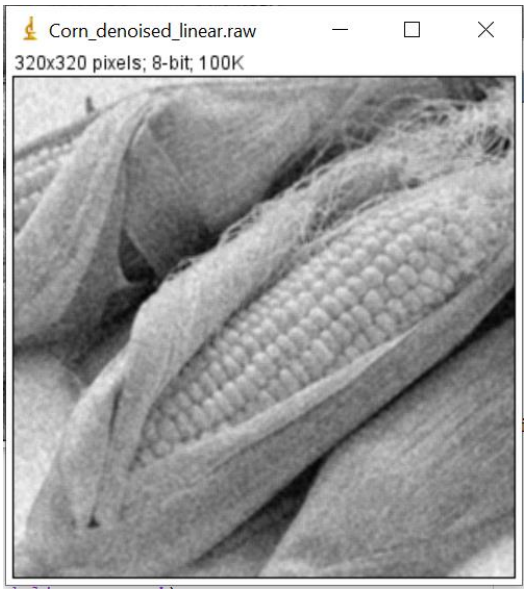

Fig. 12: Corn_denoised using linear filter

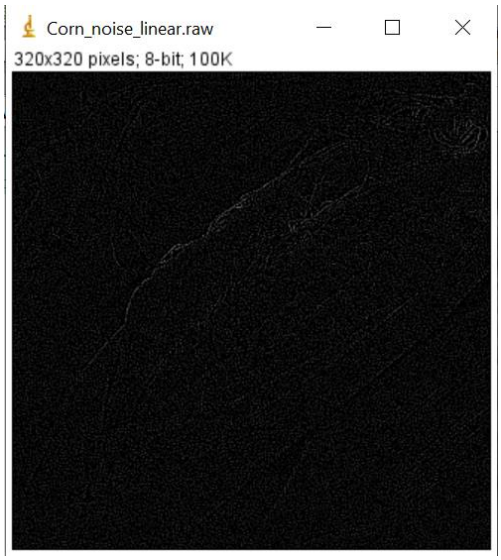

Fig. 13: noise

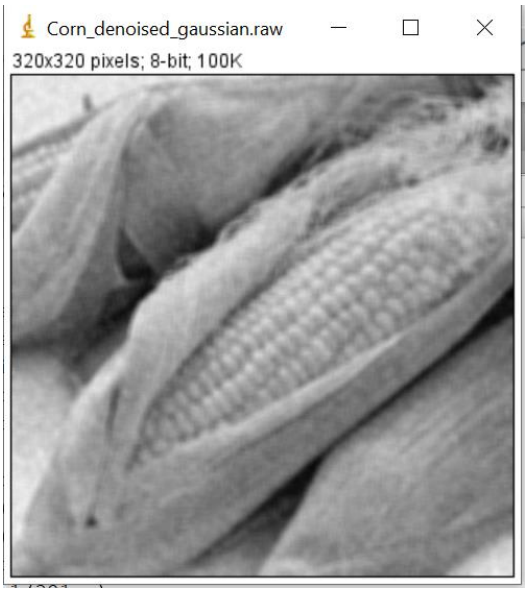

Fig. 14: Corn_denoised using gaussian filter

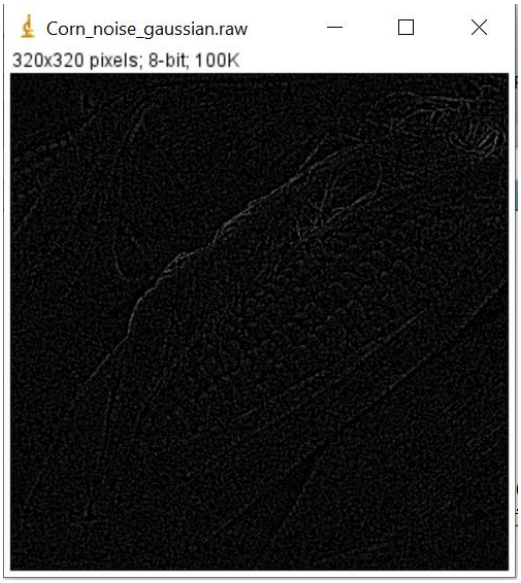

Fig. 15: noise

As can be seen from the results, the gaussian filter removed more noise than the linear filter. The PSNR value of gaussian filter is 19.42 dB and the PSNR value of linear filter is 19.18 dB (as can be seen in Fig. 15). Higher PSNR value means less noise in the image.

(b) Bilateral filter denoising

The equation for the bilateral filter was used to make the bilateral filter:

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

Then, convolution was done using the bilateral filter on the image.

When sigma c and sigma s are less than 1 (when sigma c = 0.4, sigma s = 0.6) the PSNR was 17.7062 db.

So the least noise is removed when the values are less than 1.

When sigma c = 40, sigma s = 50 the PSNR was 19.7202 db.

When sigma c = 50, sigma s = 40 the PSNR was 19.6400 db.

When sigma c = 50, sigma s = 50 the PSNR was 19.7201 db.

When sigma c = 100, sigma s = 10 the PSNR was 18.0196 db.

When sigma c = 10, sigma s = 100 the PSNR was 19.5392 db.

Also, another thing that can be observed is that when the values of sigma c and sigma s are close to each other, the noise removal is better.

When the values are not close to each other then the PSNR is lower.

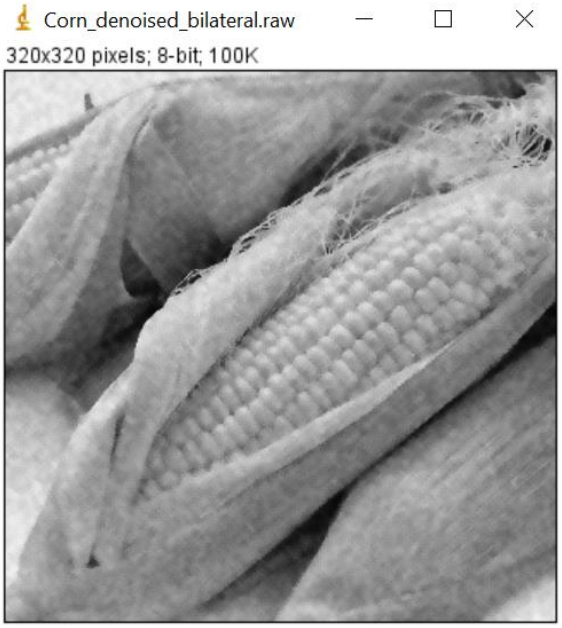

Fig. 16: Corn_denoised using bilateral filter

The bilateral filter performs better than the linear filter since it gives a clearer image with less noise. This can be inferred when we look at the PSNR values. The bilateral filter always gives a higher PSNR as compared to the linear filters. Which means that it removes more noise from the noisy image.

```
>> psnr
    Retrieving Image Corn_gray.raw ...
    Retrieving Image Corn_denoised_bilateral.raw ...
    Retrieving Image Corn_denoised_gaussian.raw ...
    Retrieving Image Corn_denoised_linear.raw ...
psnr for bilateral filter:
   19.7202

psnr for linear filter:
   19.1888

psnr for gaussian filter:
   19.4243
```

Fig. 17: PSNR for linear, gaussian and bilateral filtered images

(c) Non-Local Means Filter

Abstract and Procedure:

In all the filters that we have used till now we have been averaging by using the neighboring pixels and getting the denoising done. In non-local means filter, as the name suggests, we do not use the neighboring pixels for averaging. The pixels to be used for the target pixel are chosen based on how similar the pixels are to the target pixel.

While implementing the algorithm weights are assigned to the pixels depending on how similar they are to the target pixel.

Then the value of the target pixel is estimated by averaging the pixel values multiplied by their weights. Hence, we get the value of intensity at the target pixel.

This process is repeated for all the pixels in the image and noise is removed.

Results:

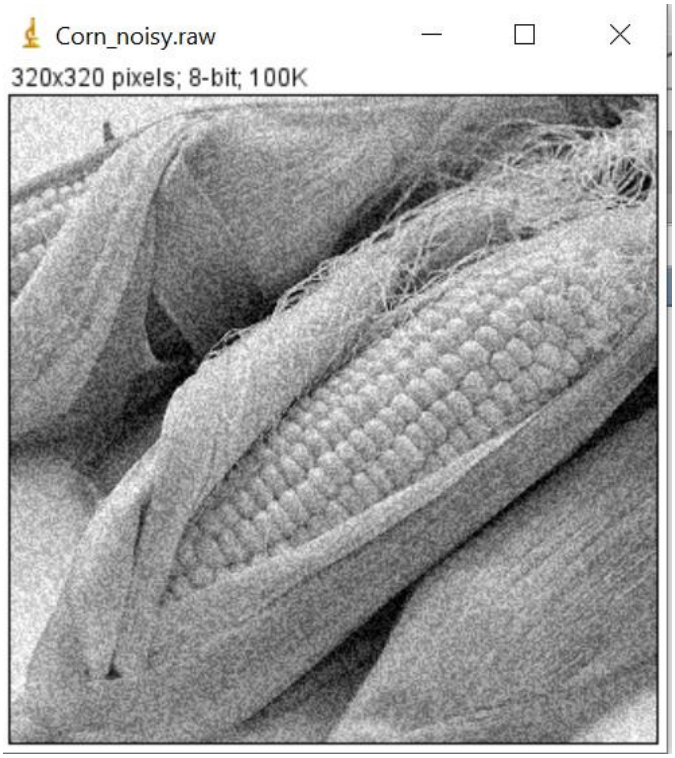

Fig. 18 : Corn_noisy.raw

Fig. 18: After applying NLM filter

Discussion:

The NLM filter gives better results than the linear, gaussian and bilateral filters. This is due to the fact that is uses more number of pixels from the image to estimate the true value of the target pixel. But the output of NLM is way better than the filters used in 3(a) and 3(b).

The denoised image using NLM filter looks extremely smooth and noise-free.

The value of sigma taken was 25. When the value of sigma was very low, the denoising was not effective at all. So the parameter to be aware of was sigma having a higher value.

The source code for Non-Local Means filter implementation is taken from the website mentioned the references. [1]

(d) BM3D filter (Block Matching and 3-D transform filter

Abstract and Procedure:

The following steps are carried out for applying the BM3D filter:

1. Block matching and collaborative filtering is done. In block matching, the noisy image is scanned and processed, and blocks are made of the noisy image. These are called reference blocks. After this, similar blocks are found and are stacked in the form of a 3-D array. Collaborative filtering is performed on the stack and then the values are returned to their original locations in the 2-D form.

2. After the first step, there are more than one estimates of the information present at each pixel. So, all the information is averaged out to get the final information at each pixel. This removes the noise from the picture. And due to the overlap in the blocks the noise is removed effectively since the algorithm goes over the same pixels multiple times.

Results:

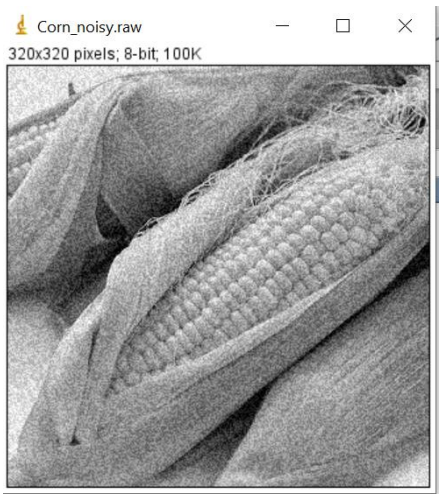

Fig. 17 : Corn_noisy.raw

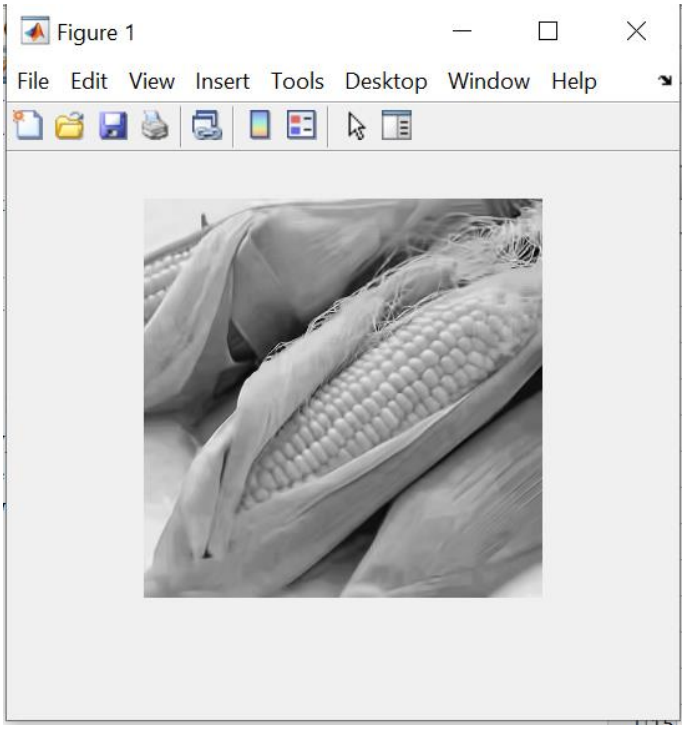

Fig. 18: After applying BM3D filter

Discussion:

The output image after the BM3D filter is applied is very clear and almost all the noise is removed from the Corn_noisy.raw image file. The image looks smooth. The filter gives better results than linear, gaussian and bilateral filters. This is because it goes over each pixel of the image multiple times since there is overlap in the blocks that were chosen. So, we have more information of each pixel which can be averaged out and it gives us better noise removal and a more accurate output which is closer to the original image.

The source code for BM3D filter implementation is taken from the website mentioned the references. [2]

(e)

(1) The types of noises in the image are salt and pepper noise(impulse noise) (since there are dark pixels at bright locations and bright pixels at dark locations in the image). The noise is also uniformly distributed over the entire image, so the image has Gaussian or Uniform noise as well.

(2) The filtering needs to be performed on the individual red, green and blue channels separately to remove both the noises. The filtering needs to be performed individually since it is a color image and each color (R,G,B) has intensities from 0-255 which have noise associated with them. So, for effective removal of noise, the filter needs to be applied individually to all the channels for both the noises.

(3) The types of filtering that can be used to remove the noise is mean filtering i.e. convolution (linear filters work well with both impulse as well as gaussian noise). Gaussian filtering can also be used to remove the impulse as well as the gaussian noise. The filters can be used in cascade to have better noise removal.

References:

[1] NLM source code: https://github.com/NVlabs/SNN/tree/master/NLM

[2] BM3D source code: http://www.cs.tut.fi/~foi/GCF-BM3D/