EE569 Digital Image Processing

HOMEWORK # 1

Aagam Shah

USC ID-8791018480

USC Email – aagamman@usc.edu

Submission Date – January 28, 2020

Problem 1: Image Demosaicing and Histogram Manipulation

(a) Bilinear Demosaicing:

1. ABSTRACT & MOTIVATION

A Demosaicing is basically a digital image processing technique which is used to reconstruct the entire RGB colored output image from the incomplete color samples in the input image obtained from an image sensor covered with a Color Filter Array (CFA). So therefore, Demosaicing is also know as the CFA interpolation technique.
Color Filter Array (CFA) is basically an array of tiny filters placed before the image sensor array of a camera and each color filter allows only a particular wavelength of light to pass which is pre-determined during the camera design as show in figure below.

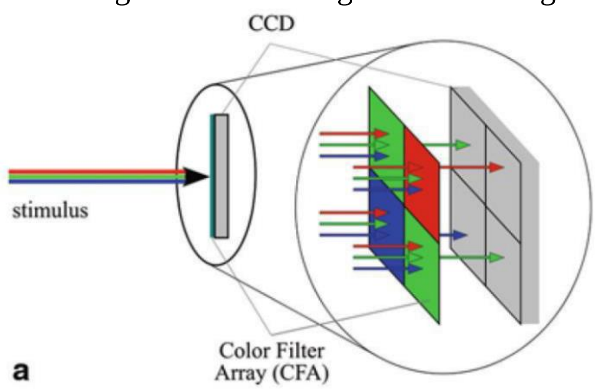


Fig.1.1

As we know that the modern-day digital image captured by digital camera has many CMOS sensors. So, in order to capture the color images, these CMOS sensors are arranged in form of Color Filter Array (CFA) which is also known as Bayer Pattern as shown in figure below.

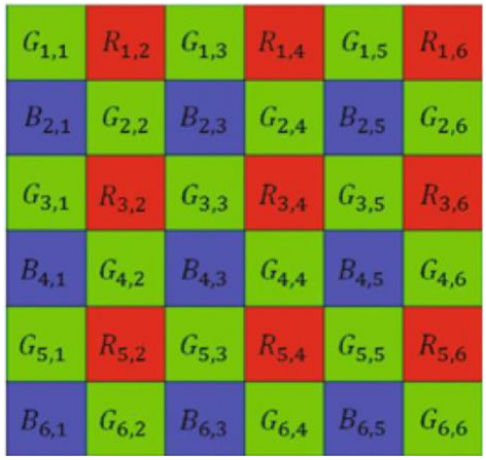


Fig.1.2

The interesting fact about the Bayer Pattern is that it uses twice the number of Green elements as compared to that of **Red** and **Blue** elements. This is because of the fact that both the **M** & **L** cone cells of the retina of human eye are more sensitive to the **Green** light.

The raw or the uncompressed output of the Bayer Pattern is called as the **Bayer Pattern Image** or the or the **Mosaiced Image.** Since we know that each sensor at each pixel location only allows a particular wavelength of light to pass out of the three primary colors i.e., (R, G, B). The remaining two colors have to be reconstructed by interpolating the intensity values of the neighboring pixels at that particular location to obtain the full colored image, which is known as **Demosaicing.**

There are four combinations of the Bayer Array Pattern such as GRBG, RGGB, BGGR and GBRG which can be used to approximate the values of the neighboring pixels at that particular location to obtain the full colored image. But here out of the four Bayer Array Pattern we will be using the GRBG configuration to demosaic the image.

There are several algorithms available for interpolating the missing color pixel value at a particular location, but here we will be using **Bilinear Interpolation** technique to get the desired output results. This is the easiest way to perform linear interpolation.

2. APPROACH & PROCEDURE

In this technique of Bilinear Interpolation, assuming the GRBG arrangement of the Color Filter Array, I calculated the missing color values of the remaining pixels at each pixel location by using the average of its two or four adjacent pixels of the corresponding color. So avoiding the first and the last row as well as column for now I started the interpolation technique from the pixel located at 2nd row and 2nd column up till the pixel located at 2nd last row and 2nd last column. This approach includes the loss of the information of the pixels located at the boundary of an image. This can be avoided by padding the first and last row as well as column with the alternate series of pattern and then perform the Bilinear interpolation to prevent the loss of the information of the pixels at boundary location. So, here starting from the pixel located at 2nd row and 2nd column which is G2 as per (G1-R-B-G2) pattern, I calculated the remaining R and B values by averaging out the neighboring two Red and Blue pixels for computing R and B values respectively. Similarly, the same technique was performed over the entire image with center pixel as R, B and G1. There are basically four types of pattern in the entire image with center pixel as either G1 or R or B or G2. The crucial part of this technique is to identify the pixel location at center for each type of pattern in the entire image to work upon. So, upon identifying I found the location of pixels as follows:

Pixel Type	Location
G1	Even Rows and Odd Columns
R	Odd Rows and Even Columns
B	Even Rows and Odd Columns
G2	Odd Rows and Odd Columns

So, after having the knowledge of the location of four types pixels from four different types of pattern present over the entire image, the pixel values of the remaining colors can be simply estimated using the simple linear averaging equations provided to implement the method. For example, to find the missing blue and green values at red pixel location, it can be calculated as follows:

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

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

Also, to calculate the missing red and blue value at green pixel location, is as follows:

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

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

3. EXPERIMENTAL RESULTS

As we can see that through the technique of Bilinear Interpolation the R, G and B colors have been interpolated and the final RGB colored resultant image has been generated from the RAW input image.

Although the image generated is not too sharp and has some visual distortion when compared with the ground truth color image, which is the artifact that was observed.

The cause for this artifact can be due to the consideration of only the nearest two neighboring pixels while averaging to find the missing color pixels rather than considering the 2nd neighborhood pixel locations to achieve better interpolation results.

Fig.1.3

4. DISCUSSION

The artifacts that are observed is as follows:

Fig.1.4

Fig.1.5

As we can see that when we zoom into the output image of *Dog* in Fig.1.4 we are able to observe random pixel color spots of R, G and B near the ears of the Dog which makes the image imperfect along with lot of random variations in pixel intensities thereby making an image unsmooth.
Similarly, in Fig.1.5 the boundary of the ball in the mouth of the Dog is not well defined as well as the hair Dog is not clearly visible. The sharp edges and texture in the output image generated are not clearly enhanced by the Bilinear Interpolation technique.
So, upon comparing the results with the original image of Dog, we conclude that Bilinear Interpolation technique is quite basic technique to demosaic an image as it does not clearly enhances the minute details of the image and leaves it blurred and unsharpened making it difficult to clearly identify the fine details of an image.

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

1. ABSTRACT & MOTIVATION

Malvar-He-Cutler is basically an improved version of bilinear interpolation Demosaicing algorithm. The quality of the Demosaicing result obtained by MHC algorithm is higher as compared Bilinear Interpolation technique due to the adding of a 2nd order cross channel correction term to the Bilinear Demosaicing result. The MHC algorithm is very much similar to the Bilinear Interpolation Technique with just the addition of the extra weight factor which controls how much correction has to be applied.

2. APPROACH & PROCEDURE

The MHC algorithm is nothing but derived as the modification of the Bilinear Interpolation. The algorithm for MHC can be explained as follows. Like in Bilinear Interpolation, MHC algorithm also follows the same Bayer Pattern as (G1-R-B-G2), thereby generating 4 different types of patterns as explained in Bilinear Demosaicing technique. But here there are in total 8 different orientations of kernels with center pixel as G1, R, B or G2. As compared to Bilinear technique instead of just using the adjacent pixel location for interpolation here we also use the diagonal pixel location i.e., 2nd neighborhood to estimate the missing colors at a particular location of pixel to get the RGB image. So, in this technique as well I avoided the first two and the last two rows as well as columns and started my analysis from pixel located at 3rd row and 3rd column up till 3rd last row and column. This approach again has some loss of information which can be prevented by doing padding across the edges of an image. Again, here the crucial part is to identify the pixel location at center for each type of pattern in the entire image to work upon as did for the Bilinear technique. So, for example,

- To find the **Green** component at **Red** pixel location is estimated as,

$$\mathbf{\hat{G}(i, j) = \hat{G}bl(i, j) + \alpha\Delta R(i, j)}$$

where ΔR is the discrete 5-point Laplacian of the red channel,

$$\mathbf{\Delta R(i, j) := R(i, j) - \tfrac{1}{4}(R(i - 2, j) + R(i + 2, j) + R(i, j - 2) + R(i, j + 2)).}$$

- To estimate a **Red** component at a **Green** pixel location,

$$\mathbf{\hat{R}(i, j) = \hat{R}bl(i, j) + \beta\Delta G(i, j),}$$

where ΔG is a discrete 9-point Laplacian of the green channel.

- To estimate a red component at a blue pixel location,

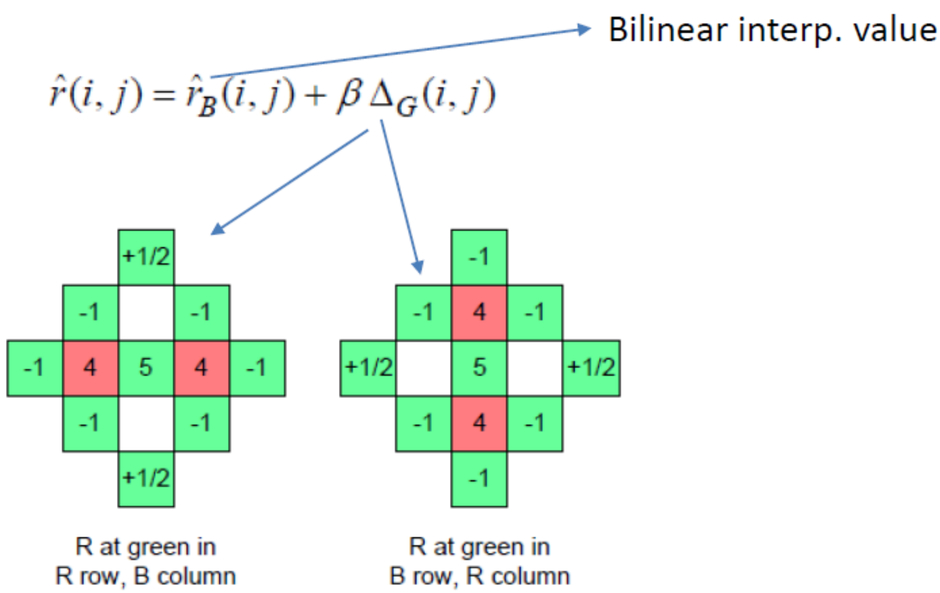
$$\mathbf{\hat{R}(i, j) = \hat{R}bl(i, j) + \gamma\Delta B(i, j),}$$

where ΔB is the discrete 5-point Laplacian of the blue channel.

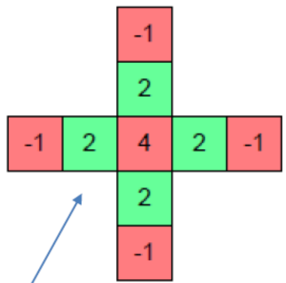
The parameters α, β and γ are basically the Laplacian Correction terms. The pre-computed values are as follows:

$$\mathbf{\alpha = 1/2,\ \beta = 5/8,\ \gamma = 3/4}$$

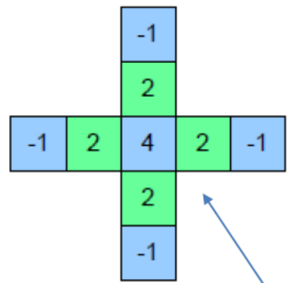
The 8 different types of pattern along with the formula to compute the value of missing color elements is as follows:



EE569 Digital Image Processing



G at R locations



G at B locations

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

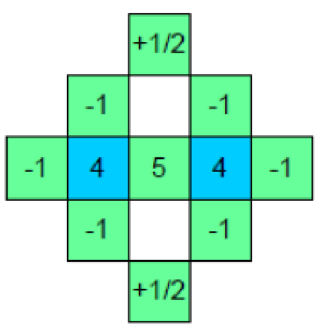
$$\Delta_R(i,j) \triangleq r(i,j) - \frac{1}{4}\sum_{(m,n)=\{(0,-2),\,(0,2),\,(-2,0),\,(2,0)\}} r(i+m, j+n)$$

$$\hat{g}(i,j) = \hat{g}_B(i,j) + \gamma\Delta_B(i,j)$$

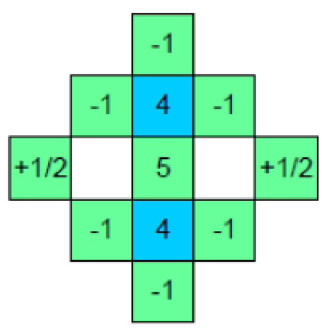
$$\Delta_B(i,j) = b(i,j) - \frac{1}{4}\sum_{\substack{(m,n)=\{(0,-2),(-2,0),\\(2,0),(0,2)\}}} b(i+m, j+n)$$

Bilinear interp. value

$$\hat{b}(i,j) = \hat{b}_B(i,j) + \beta\Delta_G(i,j)$$



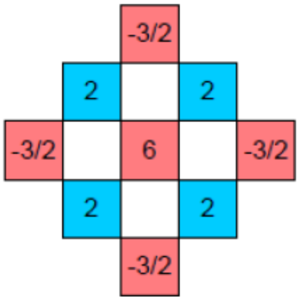
B at green in
B row, R column



B at green in
R row, B column

Bilinear interp. value

$$\hat{b}(i,j) = \hat{b}_B(i,j) + \gamma\Delta_R(i,j)$$



B at red in
R row, R column

3. <u>EXPERIMENTAL RESULTS</u>

Fig.1.6

MHC definitely works well compared to Bilinear Interpolation. The correction terms implemented in the equations of MHC enhances the minute details of an image even better and clearer. This technique basically evenly captures all the variations as that in an original image.

4. DISCUSSION

Fig.1.7 (Bilinear Demosaicing result)

Fig. 1.8 (MHC result)

As it is very clearly visible upon comparing the output images from Bilinear and MHC technique, that the image obtained from the MHC is sharper and clearer as compared to the Bilinear technique. Especially the edges in the MHC interpolated image are sharper than in case of bilinear filtering. The details are more enhanced in MHC output result. As in the above image it can be seen that the black spots of Dog in MHC result is having more clearer boundary as compared to that of the Bilinear filtering and also the details of the ball in the mouth of Dog is more clearly enhanced and visible in MHC result as compared to that of Bilinear result.

Thus, upon comparing the MHC and the Bilinear Demosaicing results the conclusion I have drawn is as follows:

i. The color contrast is brighter and appealing in the MHC output result as compared to that of Bilinear.
ii. The minute image details as in the detailing or the ball in the mouth of Dog or the black spots present on the body surface of the Dog are sharper with clear boundary in the MHC output result as compared to that of Bilinear output result.

iii. The sharp edges of the objects are preserved in MHC case and blurred in Bilinear case.

(c) Histogram Manipulation:

1. ABSTRACT & MOTIVATION

Histogram Equalization is a very useful image processing technique of contrast enhancement by adjusting the image intensities. Since we know that the human eye is sensitive contrast rather than pixel intensity values, so image with poor intensity is perceived less as compared to the image with better intensity distributions. In this method the areas of lower contrast gain a higher contrast. Through this kind of adjustment, the intensities of an image are evenly distributed over the histogram. Histogram equalization redistributes the image intensities in order to obtain as uniform image.

2. APPROACH & PROCEDURE

Basically, there are two methods to implement the histogram equalization techniques:

➢ METHOD-A (Transfer-function based):

In this method we change the probability distribution function to a more uniform distribution through cumulative distribution function. Steps to find histogram equalized image using Transfer function approach is as follows:

i. Firstly, being an RGB image we separate the R, G and B planes into separate 2-D matrix and work separately on each of the three planes. Now count the frequency of pixels of each grayscale value ranging from [0-255], hence the histogram is obtained.
ii. Next, we normalize the probability histogram i.e., we divide the histogram obtained in step 1 by the total number of pixels in an image.
iii. Next, we calculate the cumulative distribution function.
iv. Then according to the mapping rule, we again normalize the CDF by multiplying it with 255.
v. Lastly, we retrace back the new pixel intensities with corresponding frequency count. The output image now has the characteristics of the changed PDF.

➢ METHOD-B (Bucket Filling Approach):

In this method we basically convert the probability distribution function into uniform probability distribution function so that the image will have the intensities in all range.

i. Initially here too, being an RGB image we separate the R, G and B planes into separate 2-D matrix and work separately on each of the three planes. Now again count the frequency of pixels of each grayscale value ranging from [0-255], hence the histogram is obtained.

ii. Next, based on the range of intensities i.e., 256 buckets are created with each bucket assigned with different intensity values ranging from 0 to 255.

iii. Next, we start sorting the intensity values in ascending order along with fetching the location of each pixel which will be required during the remapping step.

iv. So now the pixels with the lowest intensity values will go into the lower intensity bucket until the bucket is filled. So, the capacity of each bucket is calculated by dividing the total number of pixels by the total number of buckets. If the bucket is already filled but still the pixels of one kind of intensity are left, then those pixels will go to the next bucket. This process continues till we fill in all the pixels values in all the buckets up to the last bucket with pixel value 255.

v. Now we remap these pixels to the corresponding intensities in their buckets along with their respective locations and thereby getting the final equalized output image.

3. EXPERIMENTAL RESULTS

- METHOD-A (Transfer-function based):

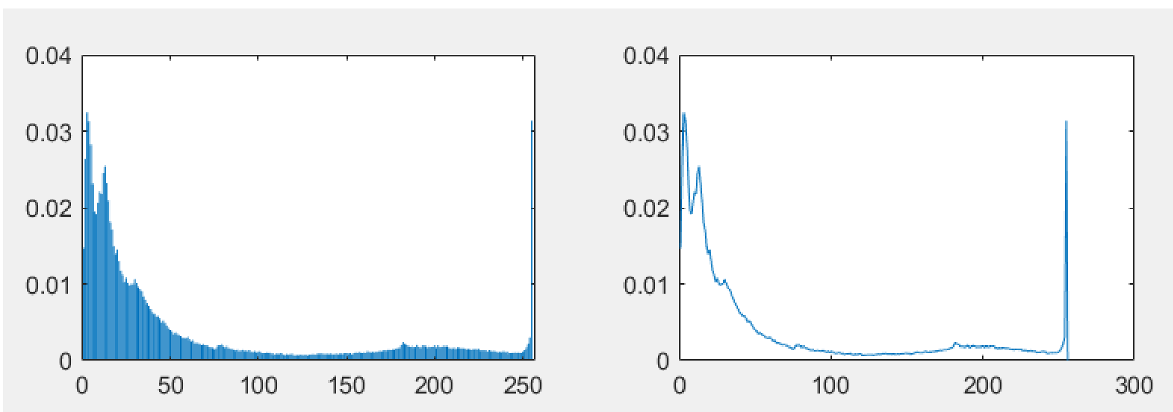


Fig.1.9 (Red Channel)

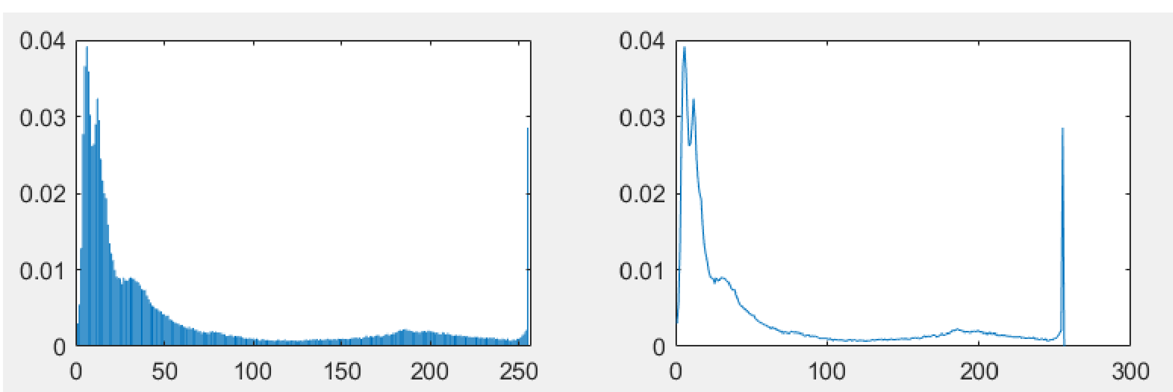


Fig.1.10 (Green Channel)

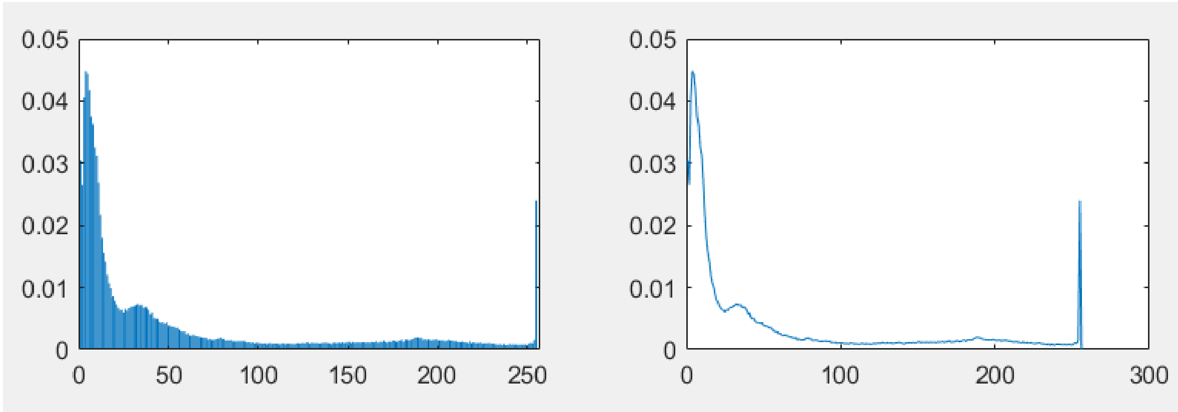


Fig.1.11 (Blue Channel)

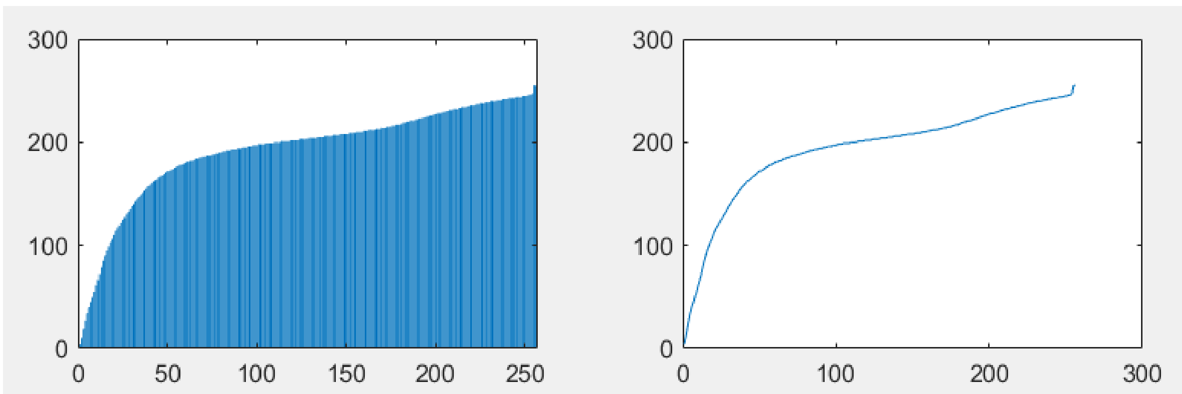


Fig.1.12 (Red Channel)

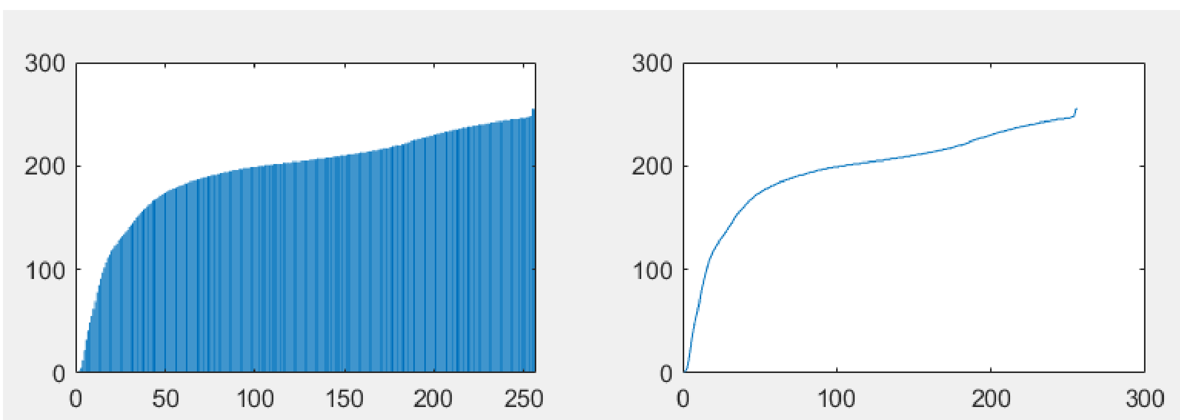


Fig.1.13 (Green Channel)

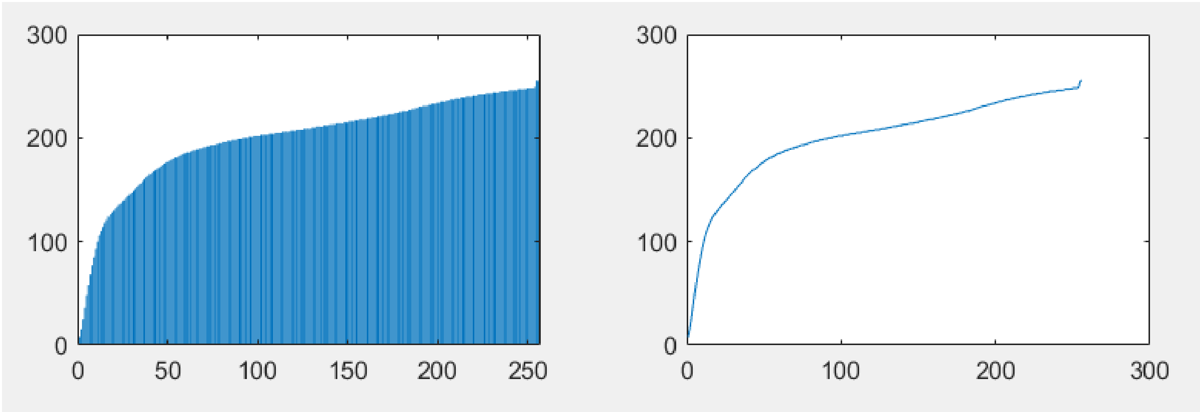


Fig.1.14 (Blue Channel)

Fig.1.15 (Original Image)

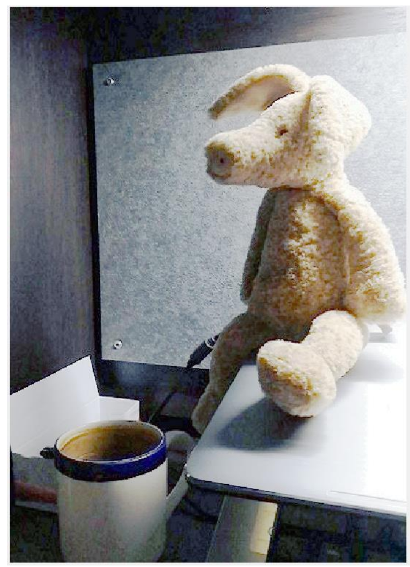


Fig.1.16 (Histogram Equalized Image)

As we can see that both the methods i.e., Method A as well as Method B can be used to compute the equalized histogram function and both the methods can be used for contrast enhancement. But the Method A (Transfer function approach) is computationally less intensive as compared to the Method B (Bucket Filling approach).

4. DISCUSSION

The enhanced image obtained after the histogram equalization in Method A i.e., Transfer Function based is more illuminated and brighter as compared to that of the ground truth image. As I could not generate the output for Method B so could not comment on the enhancement results obtained by Method B i.e., the Bucket filling algorithm. But for Method A the current results can be improved by using Adaptive Histogram Equalization technique which improves the local contrast and enhances the definitions of the edges in each region of an image. The other technique, Contrastive Limited Adaptive Equalization, can be used for contrast limiting in order to prevent the over amplification of noise which adaptive histogram equalization technique gives rise to.

Problem 2: Image Denoising

(a)Basic Denoising Methods:

1. ABSTRACT & MOTIVATION

Digital images play a very important role in our daily routine life like as they are used in wide are of science and technology, satellite television, intelligent traffic monitoring systems, signature validation, geographical information systems, astronomy, etc. But in digital image data acquisition systems introduce various types of noises and artifacts in the images, of which we need to get rid of. So, Image Denoising plays the most significant role in the entire Image Processing cycle. Noisy image produces poor quality images and lowers the visibility of low contrast objects. Hence Image Denoising becomes an essential step in Image Processing in order to enhance and preserve the fine details that are hidden in the image data. But in Image Denoising there is always trade-off between noise suppression and preserving of actual image fine details.

2. APPROACH & PROCEDURE

i. UNIFORM WEIGHT FUNCTION (MEAN FILTER):

Here in this approach we take 3*3 filter with uniform weights. This type of filter in known as Mean Filter. An example of mean filter with uniform weights is as shown below:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

In this technique the Mean Filter acts as sliding window over the noisy input image. So, at a particular instant each pixel of the filter is multiplied with the corresponding underlying pixel of the noisy image, followed by replacing the center pixel value of the noisy image with the average mean of all the pixel values of the product of window and noisy image. This process is carried out over the entire noisy image from left to right and from top to bottom in order to get the denoised output image.

ii. GAUSSIAN WEIGHT FUNCTION:

Gaussian filters are basically used to blur the images and remove the noise and details from a noisy image. Gaussian filter is non-uniform low pass filter. In order to work with images, we use 2D Gaussian function. Here in this approach N*N size of Gaussian filter can be used to denoise an image. Here I am using 5*5 Gaussian filter for denoising the Corn image. The Gaussian filter window can be designed using the given formula as below:

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

Where (i,j) represents the (row,column) pixel location of a noisy image and (k,l) represents the neighboring pixel location within the window centered around (i,j). So, once the Gaussian Window is designed it will act as sliding window in similar fashion over an entire noisy image. The denoised output image can be obtained by the given formula as follows:

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

Where I – Noisy input Image,
Y – Denoised output Image,
Sigma – Standard Deviation of Gaussian Distribution

3. EXPERIMENTAL RESULTS

Fig.2.1 (Uniform Weight Function)
(3*3 filter size)

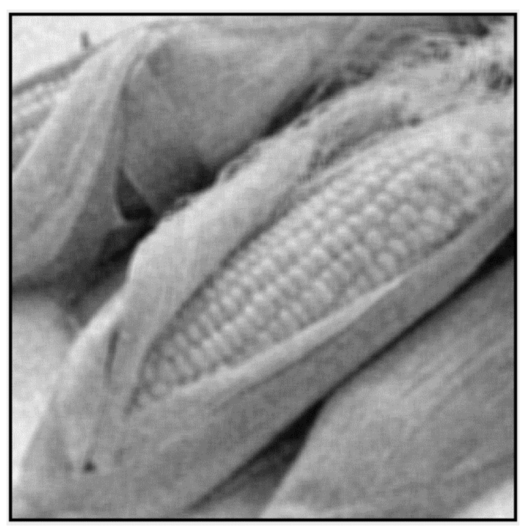


Fig.2.2 (Gaussian Weight Function)
(with sigma = 10)

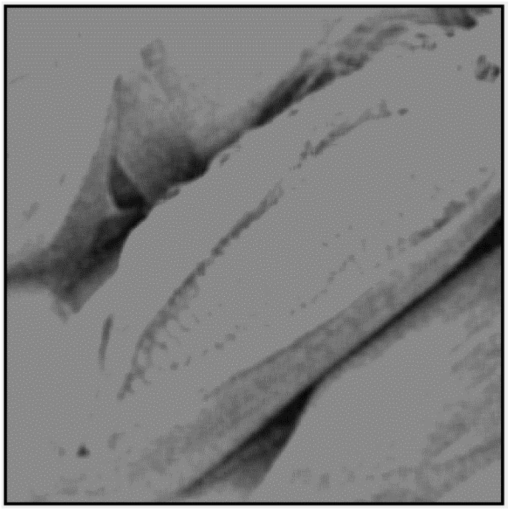


Fig.2.3 (Gaussian Weight Function)
(with sigma = 5)

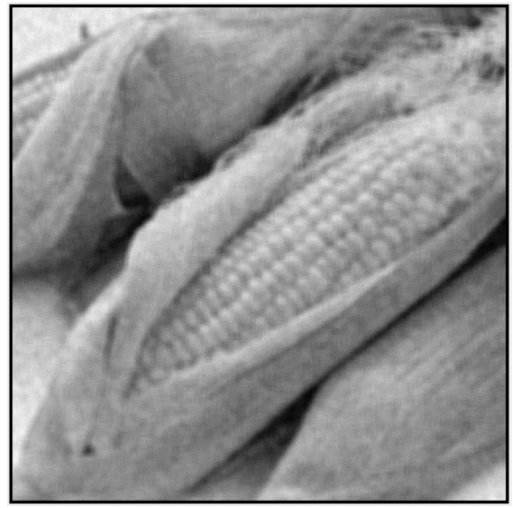


Fig.2.4 (Gaussian Weight Function)
(with sigma = 15)

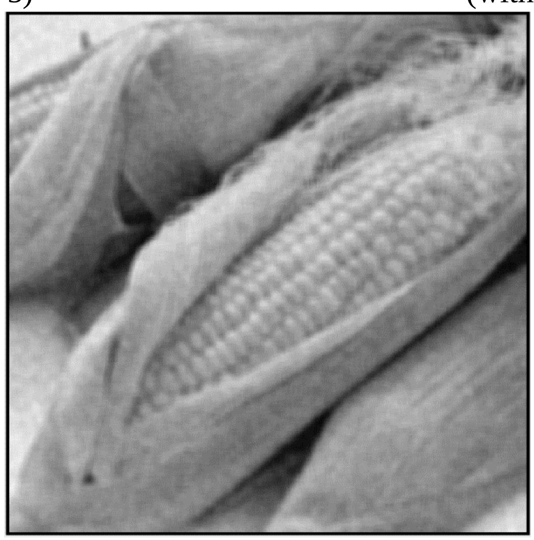


Fig.2.5 (Uniform Weight Function)
(5*5 filter size)

As it can be seen from the experimental results that the image is a bit smoothened but still noisy using the Uniform mean filter. But upon using Gaussian Filter the image is more smoothened and there is substantial amount of noise reduction as well. The cause for the same can be due to the increase in kernel size as well as the nonuniformity in the weight function of the Gaussian filter as compared to that of the Uniform Filter. So as we increase the sigma value the image gets more denoised and smoothened but with the loss of sharp edges and fine details in an image and if we decrease the sigma value then the image sharp edges are preserved somewhat but the contrast of an image is degraded making it difficult to identify the object in an denoised image.

4. DISCUSSION

The uniform filter is uniformly weighted so the denoising effect is substantially low as it evenly smoothens the noisy image in equal fashion throughout the image. Also, as you increase the filter size the image gets more blurred and denoised as well. The type of embedded noise is Additive White Gaussian Noise (AWGN).
Whereas, Gaussian function defines a probability distribution for noise. Basically, Gaussian filter is a non-uniform low pass filter. An example of a Gaussian filter is as follows:

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Gaussian function is basically a smoothening operator. The Standard deviation plays an important role in its own. Sigma is an important factor while designing a Gaussian kernel. The interesting fact about the Gaussian kernel is that the kernel coefficients diminish with increasing distance from the kernel's center, thereby the central pixel will have the highest weight than those in the neighborhood. This pattern is basically based on the Gaussian bell-shaped curve. Various other properties of Gaussian Kernel can be listed as follows:

- The Gaussian kernel is rotationally symmetric and depend on the value of sigma.
- Larger values of sigma give greater amount of blurring.
- The kernel size must increase with the increase in sigma values in order to maintain the nature of the Gaussian filter.
- Gaussian kernel allows fast computation.

Thus, Gaussian filter is used to remove the noise at the same time diminishing the fine details of the image. It is more effective in smoothening of an image. It can also be inferred that Gaussian filter is not that effective in removing salt and pepper noise.
Gaussian filter doesn't help in preserving the brightness of an image.

(b) Bilateral Filtering:

1. ABSTRACT & MOTIVATION

Bilateral filter is basically a non-linear filter used for denoising and smoothening the noisy images. Generally, the edges are preserved are preserved in such kind of filtering technique. It is basically an extended version of Gaussian filter. A discretized version of Bilateral filter is given as follows:

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

Where w(i,j,k,l) is given by,

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

Where, (i,j) represents the (row,column) pixel location of a noisy image and (k,l) represents the neighboring pixel location within the window centered around (i,j). Sigma(c) and Sigma(s) are spread parameters.

2. APPROACH & PROCEDURE

Bilateral filter basically preserves the sharp edges in the image without blurring them along-with the noise reduction. Here the approach used is as the Bilateral filter kernel is designed by the formula mentioned above, after that similar to the above techniques this kernel will act as a sliding window over the entire input noisy image and finally the output denoised image is generated using the formula mentioned in the above section. This technique basically compares the intensity of the neighboring pixels within the kernel size. It is actually nothing but an extended version of Gaussian filter which preserves the edges in the image.

3. EXPERIMENTAL RESULTS

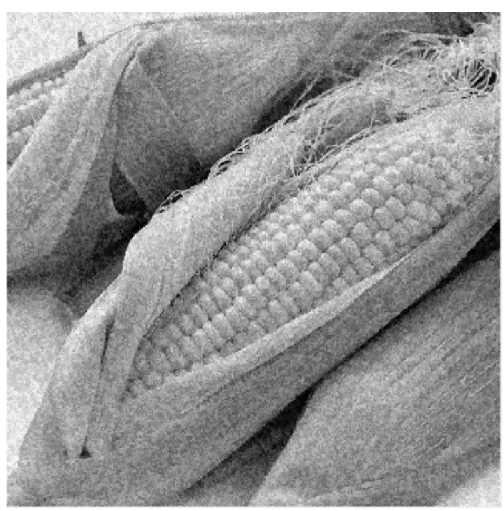


Fig.2.6 (Bilateral denoised output image)

As we can observe that the sharp edges in the image of Corn are preserved, along-with the image getting denoised. Although the result is visually almost similar to the Gaussian filter, but the quality of the result can be quantitatively analyzed by comparing the PSNR values of both the filtering methods of denoising.

4. DISCUSSION

As we can identify from the formula of Bilateral filter along-with the experimental results that the sharp edges of the Corn image are preserved accompanied by smoothening of the image i.e., denoising. The Bilateral filter is controlled by two spread parameters: Sigma(c) and Sigma(s).
So, upon increasing the range of Sigma(c) parameter it smooths the larger features and if the range of Sigma(s) is increased the Bilateral filter resembles close to the Gaussian Filter because Gaussian is flatter which means almost constant over the intensity level of an image.
The important characteristic of bilateral filtering is that the weights are multiplied, which implies that when the weight approaches to zero, smoothening of the filter gets minimized. Sigma(s) parameter preserves the contours present in an image and the term inside the norm indicates the Gaussian distance in an equation.

(c) Non-Local Means (NLM) Filtering:

1. ABSTRACT & MOTIVATION

Non-Local mean Algorithm is basically an image denoising technique in image processing. It is called Non-Local Mean filtering technique because its unlike local mean filters in which we used to take the mean value of a group of pixels surrounded by the center target pixel to denoise the image. In NLM filtering we will take the mean of all the pixels of an image weighing it by how much similar these pixels are to eth center target pixel. This technique actually results in greater clarity, smoothening and less loss of detail in the output denoised image.

2. APPROACH & PROCEDURE

In this filtering technique we basically compare the neighborhood of the center pixel with the other pixels present in the filter and the more weight is given to the pixels which has a similar neighborhood as that of the center pixel around which the filter is sliding. Also NLM algorithm not only compares the grey intensity levels at a single point but in geometrical configuration in the neighborhood. In this technique basically there are two different kernels (window inside window) which slides over the entire noisy image from left to right and from top to bottom to generate the denoised output image. The inner most kernel slides within the boundary of the outermost kernel in order to update the center pixel value of the outer kernel and then calculate the pixel value for the output denoised image using the formula given below.

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

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

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

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

Where N(x,y) is the window centered around (x,y) location and *h* is the filtering parameter. Here, n1 and n2 denotes the relative position in the neighborhood window. *a* is the standard deviation of the Gaussian kernel.

3. EXPERIMENTAL RESULTS

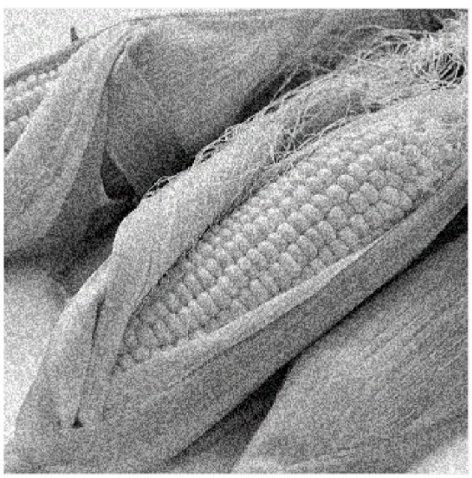


Fig.2.7 (Noisy image of Corn)

Fig.2.8 (Denoised image of Corn using NLM)

Non-local mean filter takes Gaussian weighted Euclidean distance between the block centered at target pixel and the neighboring pixel. As we see that NLM technique is much better than the previous techniques in terms of denoising as well as preserving the fine details and information present in the input image and resembling it closer to the ground truth image. NLM smoothens the image evenly throughout the region of interest.

4. DISCUSSION

As we know that the human eye is only able to perceive the quality of an image that has been improved by denoising method. Comparing the results with the local filtering techniques it shows that although NLM is computationally intensive, but it generates even toned image with good visual appearance, denoised and smoothened output image which still preserves the fine details present in the original image to convey the information. PSNR is the numerical measurement to calculate the quantitative performance of the filter rather than just relying on visual interpretation. *a* being the standard deviation, so as we increase the value of *a* the spread of the distribution increases thereby smoothening increases and the fine details of an image vanishes and if we decrease the value of *a* the spread decreases thereby resulting in uneven smoothening. *h* is nothing but the filtering parameter, so a *h* is directly proportional to the smoothening factor of an image.

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

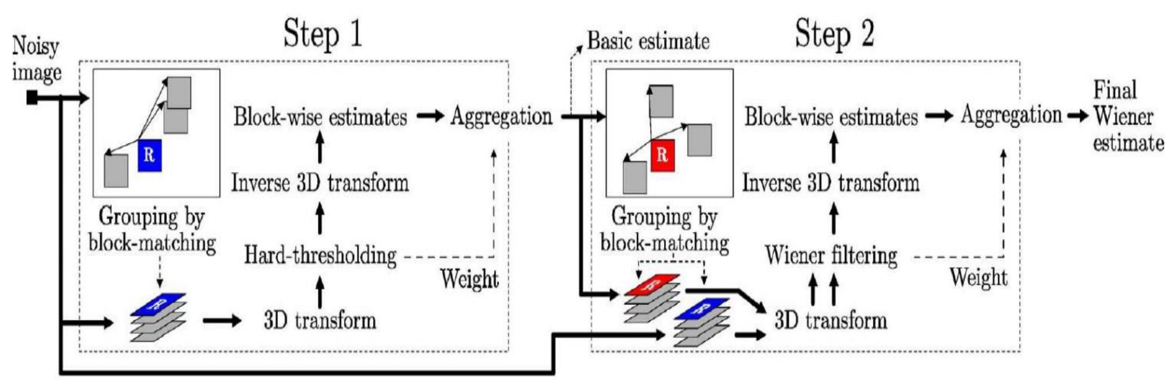
1. ABSTRACT & MOTIVATION

BM3D algorithm is basically a 3-D block matching which is primarily used for noise reduction in images. There are three steps in this technique:

i. Grouping: Here the image fragments are grouped together based on similarity like clustering. This grouping technique is called as block matching which is generally used to group similar image fragments of an image.

ii. Collaborative filtering: In this step filtering is performed on each and every fragment through 3D linear transformation followed by shrinkage of transform coefficients by Weiner filtering and then the linear transformed fragments are inverted back to reproduce the filtered fragments.

iii. Aggregation: This is the last step of the algorithm in which the image is transformed back in its 2D form in such a manner that the noise from the image is filtered out but still retaining the original information of an image.



2. <u>EXPERIMENTAL RESULTS</u>

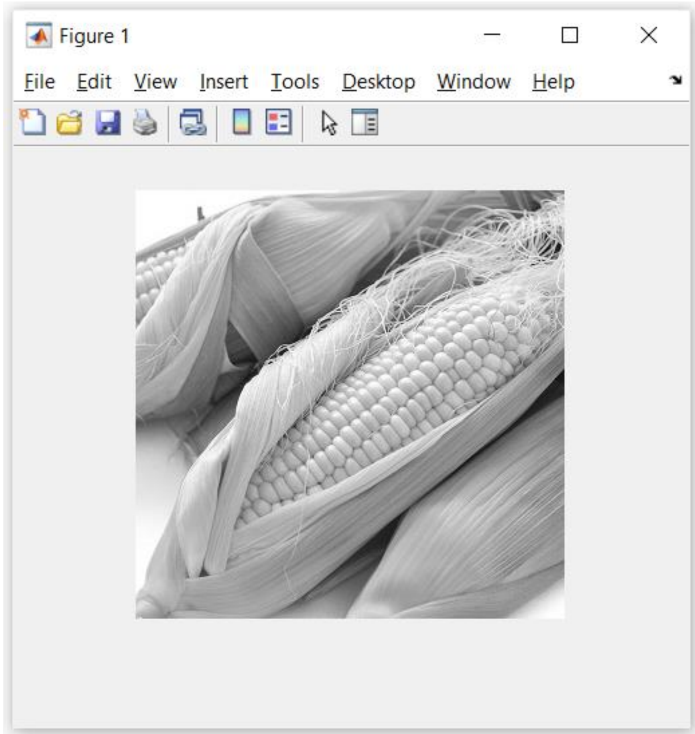


Fig.2.9 (Original Corn Image)

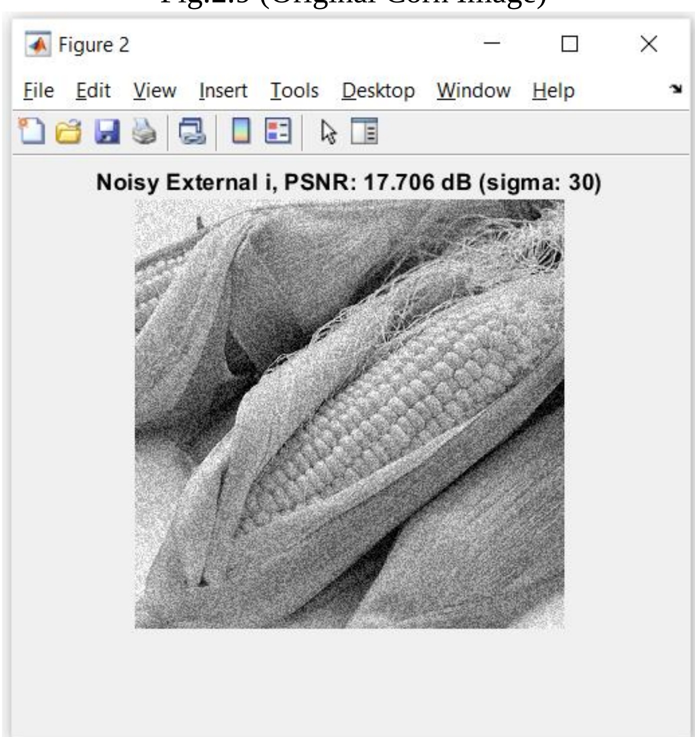


Fig.2.10 (Input Noisy Corn Image)

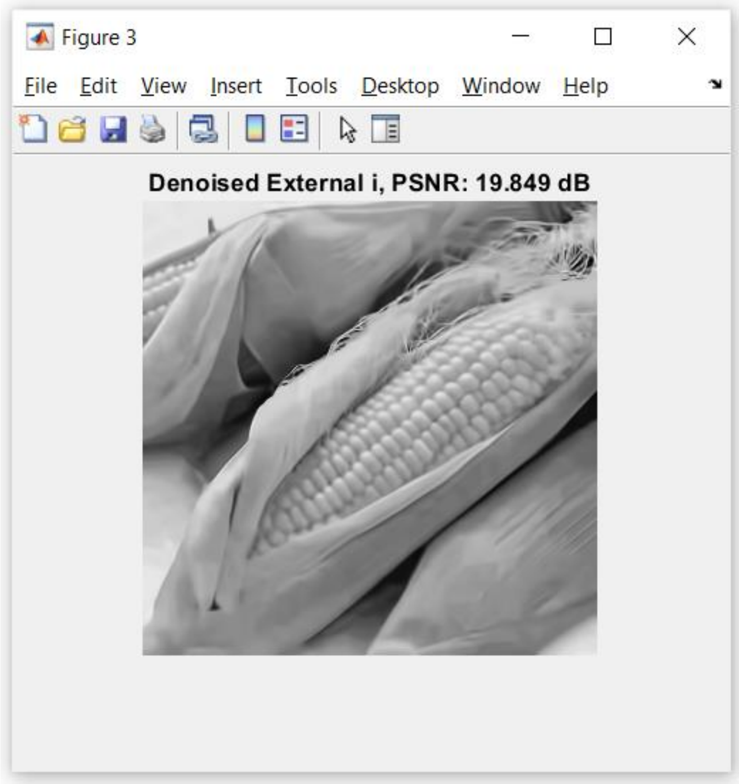


Fig.2.11 (Output Denoised Corn Image)

The inference that can be drawn from the output result which we got is that changing the sigma value changes the denoising profile of the output image as well as the PSNR values. If we decrease the sigma value the image resembles closer to the noisy image and has not been denoised substantially and if we increase the sigma value then the image is smoothened and the spread of the distribution increases thereby evenly toning the distribution and making image more smooth which might cause the loss of information present at the sharp edges in the image. So, we choose sigma value as 30 which is in the optimum range where the image is smoothened, denoised as well as it preserves the fine details and the information of the image to identify it.

3. DISCUSSION

Out of all the denoising methods implemented BM3D is the most powerful filtering technique. Although its computationally very intensive but the results generated by this technique closely resembles to the ground truth image. At the same time, it gives excellent result by implementing patch denoising based approach. It gives excellent deblurring and denoised output results along with preservation of the important information in the image.

(e) Mixed noises in color image:

DISCUSSION

1) There are basically two types of noises which can be identified upon human visualization, which are:
   i. Salt and Pepper Noise
   ii. Shot Noise or Film Grain Noise
2) Yes, we should perform the filtering techniques on individual channels separately for both types of noises in order for better understanding of noise distribution profile over the image and how to work upon it in less computationally intensive way to denoise it.
3) Salt and Pepper noise is basically an impulsive noise. Generally, image containing salt and pepper noise will have dark pixels in bright region and bright pixels in dark regions. Salt and Pepper noise can be eliminated using Mean or Median Filtering techniques.
   Shot noise or film grain noise is a signal dependent noise. It is usually in grainy structure distributed uniformly over entire image. Linear filters can be used to remove shot or film grain types of noises.
   Cascading of filters have to be taken care of when you work with linear as well as non linear filters, so here if we are working with Median filter which is a non linear filter and the other filter used is linear filter, so while cascading the order followed is generally Linear Filter followed by Non Linear filter.
   So the answer is No, we cant cascade these filters in any order if one of them is linear and the other one is nonlinear filter.

References:

1. https://stackoverflow.com/questions/25667140/how-can-i-read-in-a-raw-image-in-matlab
2. Pascal Getreuer," Malvar-He-Cutler Linear Image Demosaicking", Published in Image Processing On Line on 2011–08–14.
3. https://stackoverflow.com/questions/43127920/how-do-i-read-a-raw-rgb-image-in-matlab
4. https://www.youtube.com/watch?v=PD5d7EKYLcA
5. https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Gaussian%20Filtering_1up.pdf