**HOMEWORK #1**

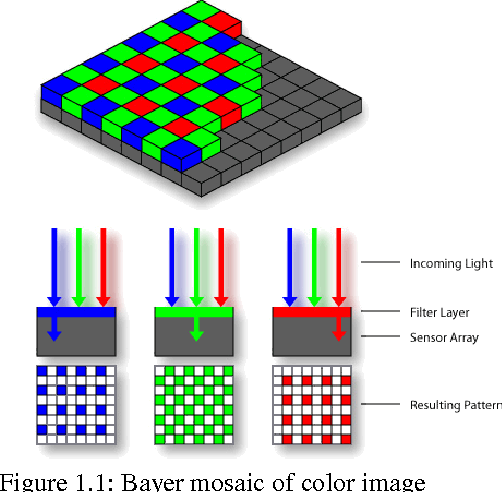
**Issued: 01/13/2020 Due: 01/28/2020**

**Problem 1: Image Demosaicing and Histogram Manipulation**

1. **Abstract and Motivation:**
2. ***Image Demosaicing***

The advent of digital cameras opened the gates of several research topics ranging from Image acquisition to compression. Till 1990s only film cameras were prevalent, but during the mid 1990’s consumer level cameras were manufactured and sold in an unprecedented way. Today, we have reached the age of digital SLR cameras. Digital cameras use Color Filter-Array capture the image and color interpolation technique is used to obtain full resolution color images. **Color Filter Array (CFA)** is an arrangement of sensors in a specific pattern to sense different wavelengths of color spectrum. The naive approach of stacking three sensors (each for red, blue, and green) to obtain full resolution color images is both cost and space inefficient. An efficient and pragmatic solution is to sense one channel and interpolate the missing color channels.

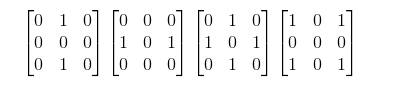
The most common and efficient CFA is the **Bayer pattern** where green sensors occupy 50% of the pixels and remaining 50% equally by red and blue. This is because the human visual system is sensitive to green wavelength. Figure (1.1) shows the schematic of image acquisition using CFA and Bayer pattern.



1. **Bilinear Demosaicing:** Bilinear Demosaicing is the simplest demosaicing algorithm where we interpolate the missing color channels using the neighbouring pixels. We do simple averaging operation of neighbouring pixels to interpolate respective color channel. Interpolating blue channel in red and green pixel location is given by Equations 1.1 and 1.2.

Similarly the red channel in green and blue pixel values can be interpolated. The equation for interpolating green channel in red and blue pixel values is given by Equation 1.3.

If we convert these equations to a matrix we will get following filter masks.



1. **Malvar-He-Cutler (MHC) Demosaicing:** Unlike bilinear interpolation technique MHC approach exploits the correlation between the channels. For example to interpolate red channel in a blue pixel, besides taking the average of the neighbouring pixels we also consider following conditions,

* If the center pixel value is equal to the average of neighboring pixels, then no correction is needed.
* If the center pixel value is less than the average, then we need to reduce the pixel value.
* If the center pixel value is more than the average, then we need to increase the green pixel.

So, based on the 2nd neighborhood information, we add a discrete n-point Laplacian of the channel.

The equation to interpolate G value at R channel is given by Equation 1.x

where ∆R (i,j) is the discrete 5 point laplacian of R channel

Th

Similarly, to interpolate Red channel at green pixel location is given by the Equation 1.x

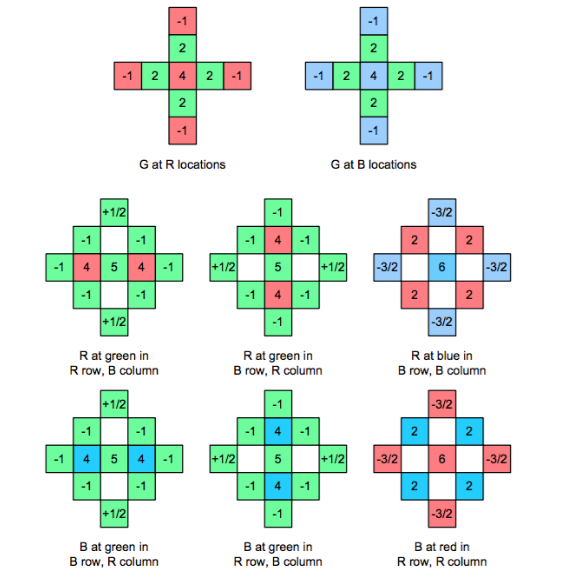
where ∆G (i,j) is the discrete 9 point laplacian of G channel

Finally, to estimate red channel at blue pixel location is given by the Equation 1.x

where ∆B (i,j) is the discrete 9 point laplacian of G channel

The values of are ½, ⅝, ¾ respectively. These values (hyper parameters) are gradient descent method.

The equations are generalised to a filter mask shown in Figure 1.2.

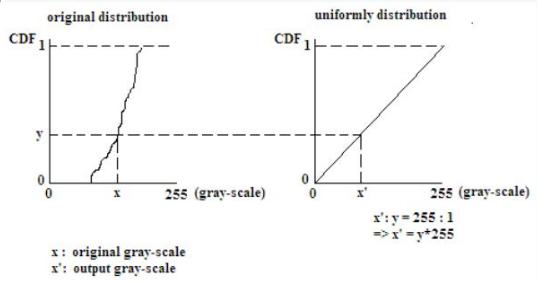


***2. Histogram Manipulation:***

Histogram equalisation is a method to enhance the overall contrast of the image by spreading the intensity values of the image over the wide range. For color images we have to operate with each channels (R,G,B) seperately and then stack them together to get the enhanced color image. Histogram equalization can be done using many techniques. In this report, we discuss two most common and popular histogram manipulation techniques.

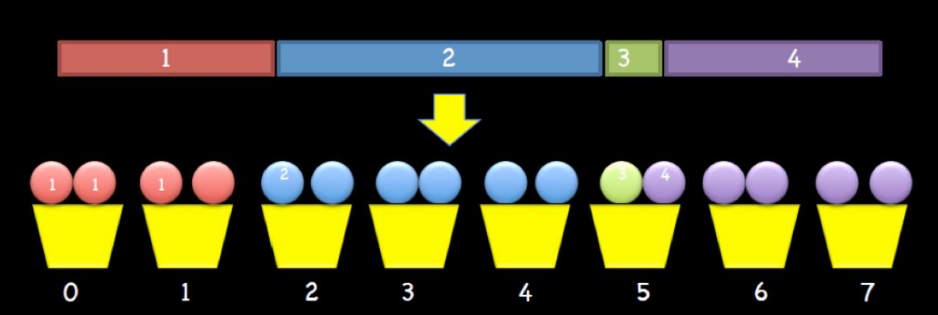
1. **Method A:**  **Transfer function based histogram equalization**

In this method, we obtain the histogram of red, blue and green channels separately and the plot CDF (Cumulative Distribution Function) of each channel. The desired transfer function is the integration of the probability density function (pdf) which is CDF. So, we have to transform the original CDF to CDF of uniform distribution. Figure 1.x shows the mapping.



1. **Method B: Cumulative Probability Based (Bucket Filling):** In this method, we arrange the pixels of each channel in ascending order and then distribute the pixels equally to each intensity component (0-255). The number of pixels in each channel is obtained be the equation below.

After arranging the pixels, distribute all the pixels equally to each bucket. The challenging part about the implementation is to determine which location in the image should be changed and which pixel location should be retained. Detailed Implementation is explained in the next section. The pictorial representation of this method is shown in figure 1.x

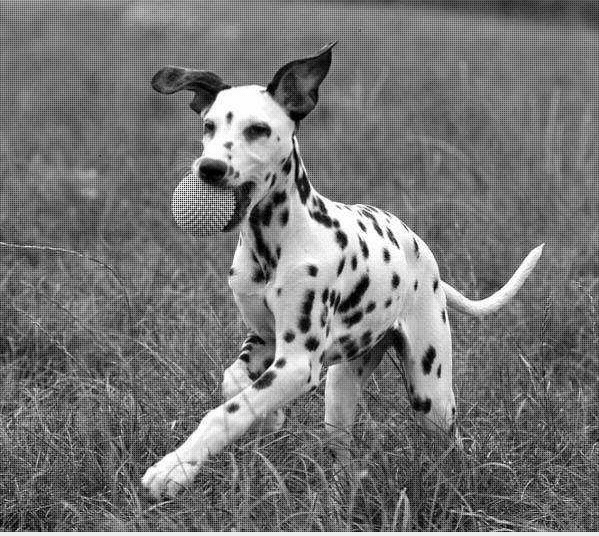


**II. Approach and Procedure**:

1. **Image Demosaicing:**
   1. **Bilinear Interpolation:**
      1. The input image to the bilinear interpolation is a 1 channel CFA pattern in 2 dimensions (Dog.raw).
      2. The input image is read byte by byte using file stream and stored in a 2D array in c++.
      3. The main operation in the implementation is convolution. Since all the filters for bilinear interpolation is 3\*3, we use 3\*3 convolution by moving the filter through each pixel.
      4. Reflection padding is used to preserve the Bayer pattern and to proceed with convolution.
      5. There are three cases. First is the pixel is green in **even rows, columns and odd rows, odd columns**. In this case we have to interpolate red and blue channel using corresponding filters by convolution. Second, the pixel is red in **even rows and odd columns.** Finally, the pixel is blue in **odd rows and even columns.**
      6. The filters in the convolution process will change based on the center pixel and the channel we are interpolating.
      7. After finding each channel in the particular pixel location, stack all the three channels and update the output 3D array.
      8. The output is stored in newDogBilinear\_op.raw which is a 3D array.
      9. The arguments to be passed are <./executable.out input\_image.raw output\_image.raw>
   2. **MHC Demosaicing:**
      1. The algorithm is similar to Bilinear interpolation except the filters.
      2. Since we are taking second order laplacian as correction term, the filter size will be 5\*5.
      3. There are 4 different filters for MHC demosaicing.
      4. Reflection padding is used to preserve the Bayer pattern.
      5. As in bilinear interpolation, there are three cases. First, the pixel is green in **even rows, columns and odd rows, odd columns**. In this case we have to interpolate red and blue channel using corresponding filters by convolution. Second, the pixel is red in **even rows and odd columns.** Finally, the pixel is blue in **odd rows and even columns.**
      6. After convolution, interpolate missing channels and stack them in a 3D array to construct full resolution color image.
      7. The arguments to be passed are <./executable.out input\_image.raw output\_image.raw>.
2. **Histogram equalization:**
   1. **Method a: Transfer function based**
      1. The input image is an RGB color image (Toy.raw).
      2. The raw image file is read byte by byte using file stream and stored in a 3D array in C++.
      3. For histogram manipulation of color images, we have to process each channel separately.
      4. Initially, the image is iterated through its width and height and pixel value from each channel is stored in a 1D array. By the end of iteration we have three 1D array of size HEIGHT\*WIDTH for each of the channels.
      5. To plot histogram of each channel, we have to calculate the count of each pixel value. For the **hashmap of <map> library** is used in C++.
      6. After hashing each pixel value, we get a key-value pair for each of 256 pixel values where key is the pixel intensity and value is the count.
      7. The key-value pair is exported to a csv file for plotting the histogram of the original image.
      8. From the histogram plot, we could observe that most of the pixels are towards the dark side of the intensity values.
      9. The purpose of histogram equalization is to make the input histogram uniform.
      10. We normalise the input histogram by diving each count value by the total number of pixels. Now, we have a probability density function.
      11. To find the cumulative distribution, we have to cumulatively sum the probabilities of each intensity value. Using the cumulative distribution we map the pixel values so that we get a uniform density function in the output image.
      12. We multiply each probability value in cdf by 255 and store it in an array. We iterate the input image and map the pixel values corresponding to the new cdf.
      13. We export the pixel values to plot the output cdf.
      14. The arguments to be passed are <./executable.out input\_image.raw output\_image.raw 3>.
   2. **Method b: Cumulative Probability based (Bucket filling).**
      1. Unlike Transfer function method, we use bucket filling technique to distribute the histogram of input image.
      2. The input image is an RGB color image (Toy.raw). The raw image file is read byte by byte using file stream and stored in a 3D array in C++.
      3. Initially, the image is iterated through its width and height and pixel value from each channel is stored in a 1D array. By the end of iteration we have three 1D array of size HEIGHT\*WIDTH for each of the channels.
      4. After getting 1D array for each channel we sort each array using bubble sort or inbuilt sort function in C++.
      5. Now we calculate the number of pixels per bucket using the equation mentioned in the above section. For the given input image the value is 875.
      6. Our task is to distribute the intensity distribution equally to 875 pixels per bucket.
      7. But the challenging part of the implementation is to find which part of the image location should be replaced with the corresponding value. The implementation is explained in the below steps.
         1. Initiate counter array of size 256 for each of R,G and B with 0.
         2. Iterate through the image using two for loops.
         3. Search for the nth (corresponding to the counter array) occurrence of (i,j) pixel value in the sorted array and find the index of that element.
         4. Now new image will be updated with the formula.
         5. But this is computationally intensive. The total number of searches for one (i,j) iteration in 3\*224000. So total number of searches will be **HEIGHT\*WIDTH\*3\*224000.** This takes approximately 7 minutes in C++.
         6. To mitigate this problem, we can use efficient data structure hashmap.
         7. As we did in the previous problem, we generate a key value pair of the sorted array.
         8. Now the index of nth occurrence of the pixel in the sorted array is determined by cumulative distribution with O(1) access for each iteration.
         9. The formula now changes to .
         10. Overall logic is preserved but we are operating through efficient data structure to reduce the execution time.
         11. Increment the counter array.
         12. This can be done simultaneously for each channel.
      8. The arguments to be passed are <./executable.out input\_image.raw output\_image.raw 3>.

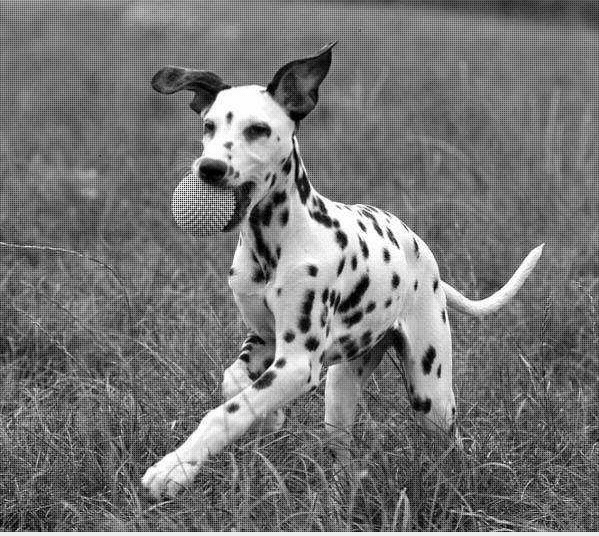
**IV. Results:**

1. **Bilinear Interpolation:** Bilinear Interpolation algorithm is implemented on the given image and output is observed. Below are the pictures of input and output image.

**Input Image Output of Bilinear demosaicing**

1. **MHC Demosaicing:** MHC demosaicing algorithm is implemented on the given image and output is observed. Below are the pictures of input and output image.



**Input Image Output of MHC Demosaicing**

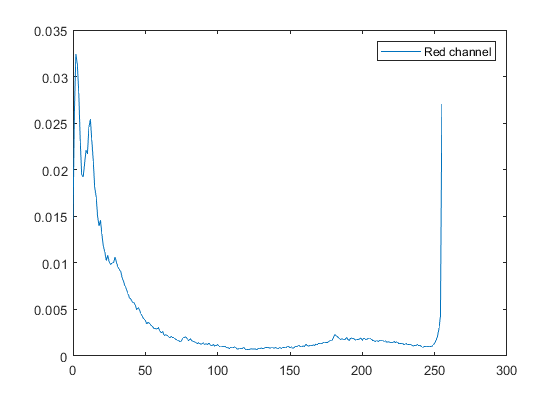
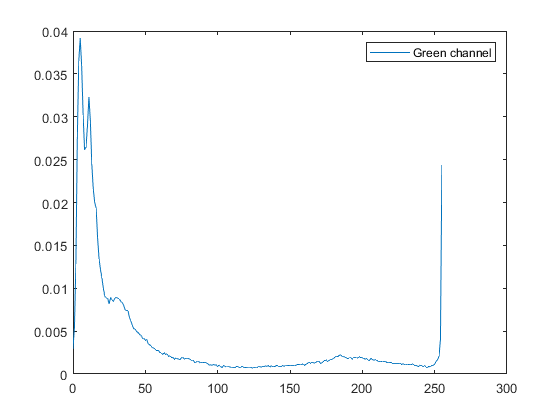
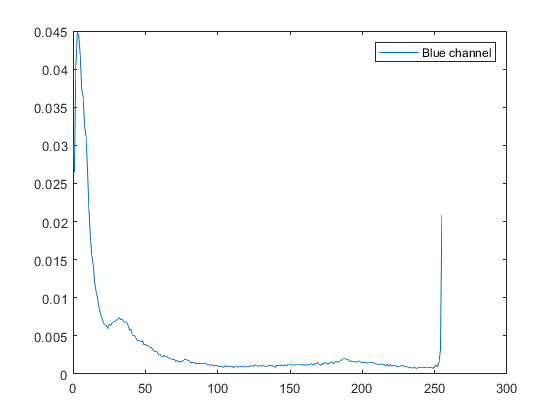
(See Discussion section for detailed comparison of results)

1. **Histogram Equalisation:**
   1. **Method a: Transfer function based method:** The results of this is as shown in the pictures below.

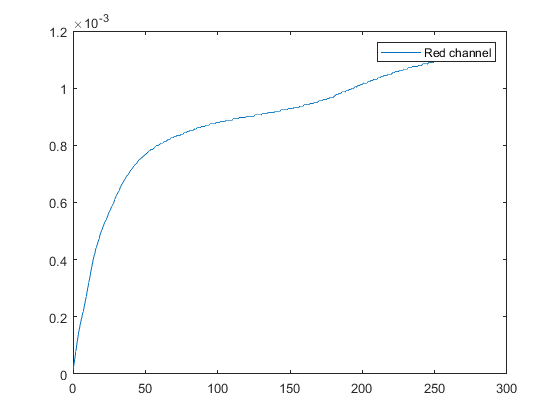
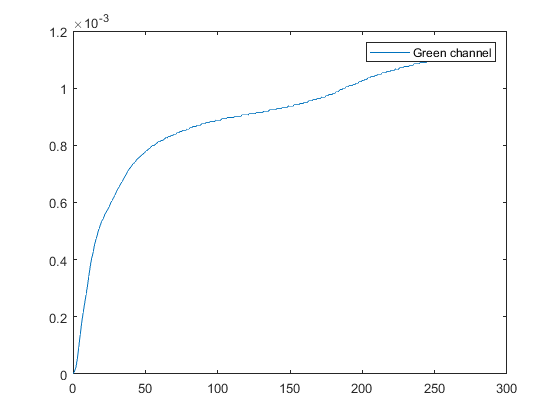
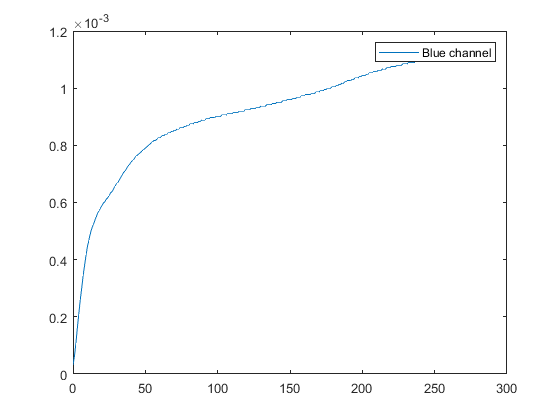


**Input Image Method-a output**

The histogram and CDF of input image are shown below.

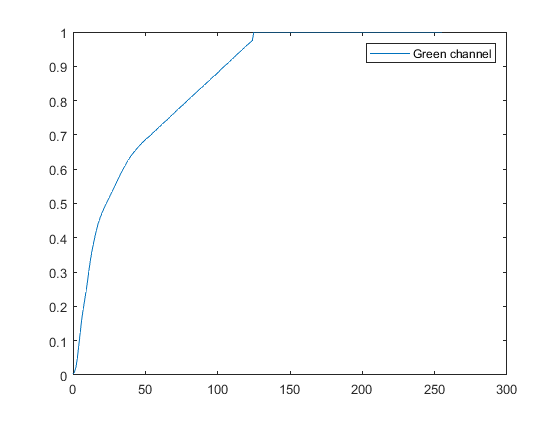
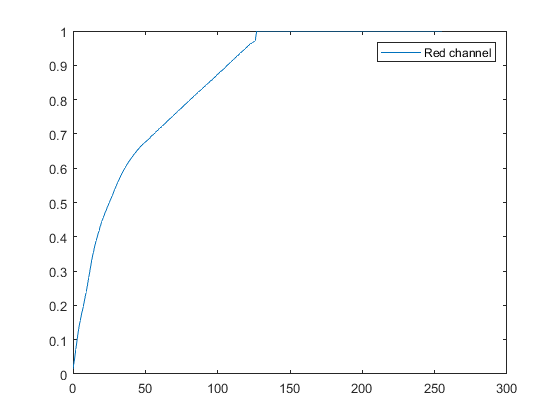
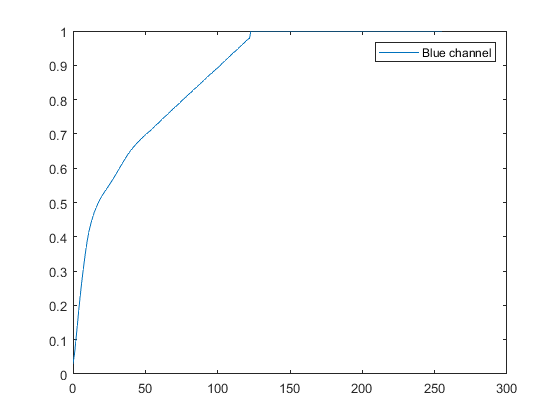


**Histogram (Normalized) of input image for each channel**



**CDF of input image for each channel**

The transfer function of the enhanced image is shown in figures below.



**Transfer functions of enhanced image (method a).**

* 1. **Method b: Cumulative distribution based (Bucket filling).**

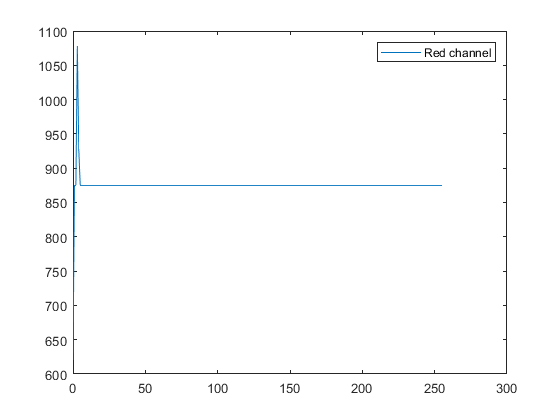
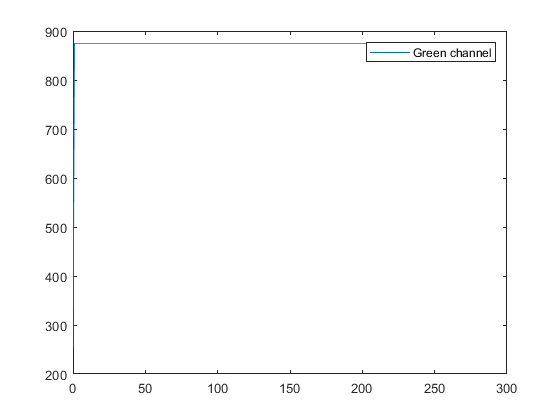
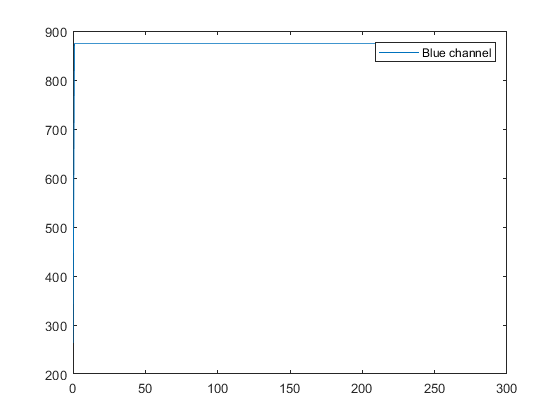
The input and output for Bucket filling method is shown below.



**Input and output image for Method B**

**(Detailed discussion in Discussion section)**

The cumulative histogram of the enhanced image using method b for each channel is shown below.



**Cumulative distribution of each channel for enhanced image using method b**

**IV. Discussions:**

1. **Bilinear Demosaicing:** The output image of bilinear interpolation is attached in the previous section. Upon closely observing the result and comparing it with the ground truth image following observations were made,
   1. Since Bilinear interpolation is a linear interpolation technique, it resembles low pass filter properties. This is the reason why the edges were not clear with sharp transition. The high frequency components at the edges are clipped off. This can be observed near the ears of the dog and sharp yellow shades of grass. This effect is called **Zipper effect.**
   2. In addition to this false color effects are observed and the image seems to be skewed towards red channel. Inconsistency among color channels is the main reason for this.
   3. Inorder to improve the quality, non linear approaches like bicubic method can be used.
2. **MHC Demosaicing:** The output image of bilinear interpolation is attached in the previous section. As we can see, the full resolution color image obtained using MHC demosaicing technique is clear and have high quality. Upon comparing the result with bilinear interpolation following observations are made,
   1. Unlike Bilinear Demosaicing, the color channel correlation is good.
   2. Due to the addition of 2nd order laplacian correction term, the zipper effect is nullified.
   3. The sharp edges are preserved.
   4. The tuning of alpha, beta and gamma parameters is very importance in this method because these coefficients dictates the amount of correction by exploiting the cross channel correlation.
3. **Histogram Equalisation:**  The transfer functions and output images of both the methods are added in the previous section. Following observations are made by comparing both the results,
   1. As we can see, the Histogram of the input image is skewed towards the darker section (lower values) of the intensity values.
   2. Using method A, we almost got a linear CDF for the enhanced image (non linear in 0-50 range). This indicates that the probability density function is not completely uniform.
   3. The contrast has been enhanced but there are some distortions when we zoom the image towards wall. This is because we are rounding off the floating values while mapping.
   4. Using method B, we got a complete uniform PDF (Linear CDF). Also, the contrast is slightly better than method A.
   5. In method B, the distortions are slightly more because we replace high frequency components with some other intensity values.
   6. Advanced algorithms like adaptive histogram equalization can be used to improve the image quality.

**Problem 2: Image Denoising**

1. **Abstract and Motivation**

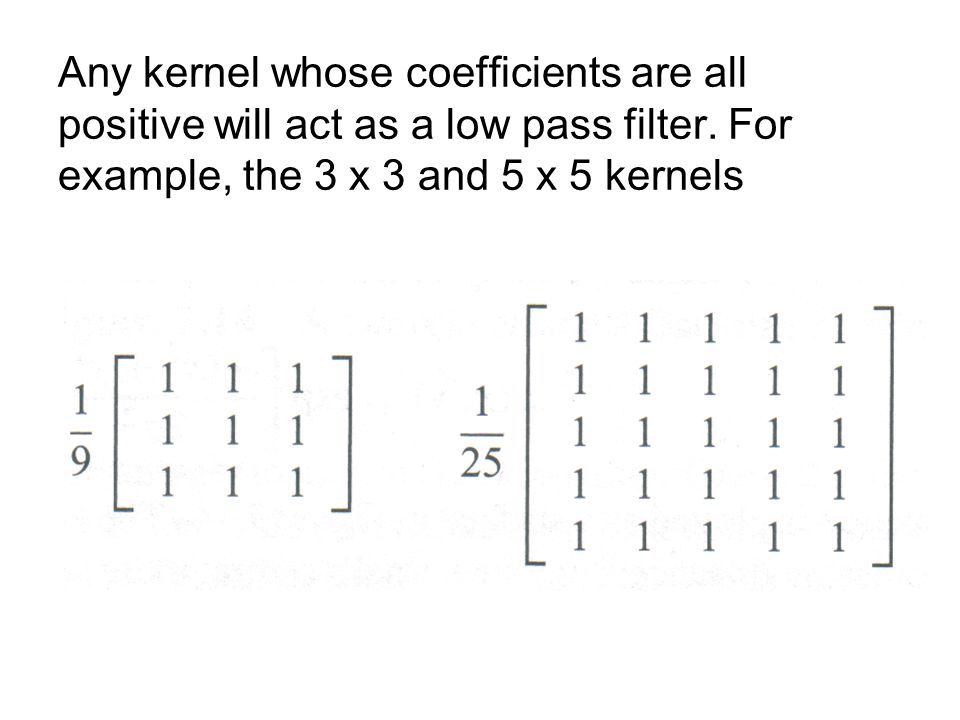
In Spite of the advancements in digital photography technology, we can’t avoid noise while acquiring the images. Denoising is a process of removing these unwanted signals to reconstruct the high quality image. Denoising is one of the low level computer vision problems which attracts the researchers even today. The ubiquitousness of imaging system - ranging from medical imaging to satellite imaging, made it very important to denoise the image without losing the information. The common types of noise are **Salt and Pepper Noise** and **Gaussian Noise**. Below figure depicts both kinds of noise.

**Salt and Pepper Noise Gaussian Noise**

There are many denoising techniques, some of them are implemented in this and their performances are quantitatively compared.

1. **Low pass filtering/ Averaging method (Basic denoising techniques):** One of the key properties of noise is that noise are high frequency components. By taking the average of the surrounding pixels, the image will be smoothened. This is because mean reduces the skewness in the data. This is a basic and simple denoising method which smoothens the image but can’t effectively remove the noise embedded in the image. We can use N\*N mask for convolution, where N can be 3,5 or 7. The filter masks are shown below.



Instead of uniform weight, we often use gaussian function to generate weights for the filter mask. The main reason behind using this kind of noise weights is because noise are not correlated and when we take an average of surrounding pixels, the signal will be preserved by eliminating the noise. The Gaussian function is given the equation below,



1. **Bilateral filtering:** Linear filters along with smoothing the image, it also degrades the edges. Inorder to avoid degradation in edges, we use non-linear filters which preserves the edges. Bilateral filter is one kind of filter. The equation of bilateral filtering operation is shown below.

)

As we can see from the above equations,

1. If the pixels are very far the first term of the function will be large, so weight will be small.
2. If the pixel values are very far (which indicates the edges) i.e., sharp variations, we give lesser weight to preserve edges.

**3) Non Local means denoising:** The main intuition behind non local means denoising is,

1. Repeated patterns of images have strong correlation.
2. Also, self similarity also corresponds to high correlation.
3. When we average these similar patterns noise gets eliminated by preserving the signal.
4. Since we look for patterns in the image which are not in the neighborhood, we call it as non local means.

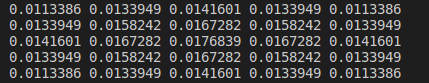
The equation of non local means is given by the following equation.

)

Basically, we iterate through the image using a window and define a neighborhood and find the similar patches. If the patches are highly correlated we give larger weight else we give lesser weight.

**II. Approach and Procedure:**

1. **Low pass filtering/ Averaging method (Basic denoising techniques):**
   1. Input image in read byte by byte and stored in a 2D array using fstream in c++.
   2. Generated a uniform mask of size N\*N.
   3. Reflection padding has been done to maintain Bayer pattern.
   4. Block by block convolution is done on the noisy image.
   5. Convolution results are stored in a 2D image which is called denoised image.
   6. PSNR value has been obtained by comparing the results with ground truth image.
   7. For **Gaussian mask,**  weights were generated and convoluted with the pixels in the loop itself. The gaussian mask generated is as shown in the picture below.



1. **Bilateral Filtering:**
2. Input image in read byte by byte and stored in a 2D array using fstream in c++.
3. Reflection padding has been done to maintain Bayer pattern.
4. Block by block convolution is done on the noisy image.
5. Weights were generated and convoluted with the pixels in the loop itself.
6. Convolution results are stored in a 2D image which is called denoised image.
7. PSNR value has been obtained by comparing the results with ground truth image.

**3) Non Local Means Filtering:**

a) The NLM open source in C++ implementation by Image Processing online has been cloned <https://github.com/npd/nlmeans>.

b) The source code is compiled using Cmake.

c) Output is obtained and compared with the ground truth data.

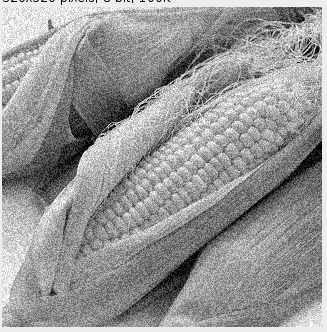
**4) BM3D Method:**

1. BM3D open source implementation in C++ by Image processing online is cloned <https://github.com/gfacciol/bm3d>.
2. The source code is compiled after installing all the necessary dependencies.
3. Output is obtained with the ground truth data.

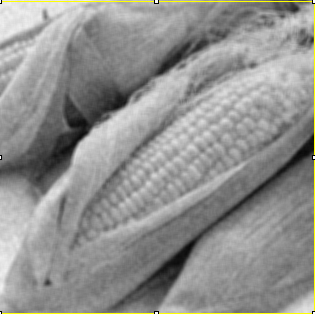
**Note:** The source code needs FFT library installed.

**III. Results:**

The denoised image using linear mask, gaussian weight, bilateral, non-local means and BM3D algorithms are as shown in the figure.



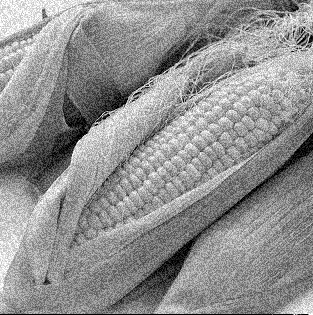
**Noisy Image**

****

**Denoised Image using Uniform filter**

****

**Denoised Image using Gaussian filter**

****

**Denoised Image using Bilateral filter.**

****

**NLM Denoising**

****

**BM3D**

**IV. Discussions**

1. **Linear filters using uniform mask:** Denoising is done on the given image (Corn\_noisy.raw) using averaging filters using linear masks. The type of noise embedded in the image is **Uniform Noise.** The noisy image is subtracted with the original ground truth image and the histogram of the result in plotted. The result was nearly uniform.

Following observations were made,

1. The averaging operation is done with different mask sizes and PSNR values were calculated.
2. With the mask size of 3\*3, the PSNR value was **19.3674db** which is slightly better than **PSNR** of Noisy image which is **17.45db.**
3. As we increase the mask size, the quality of image decreased with more smoothing.

**2) Linear filters using gaussian mask:** Denoising is done on the given image (Corn\_noisy.raw) using averaging filters using gaussian weights. Following observations were made,

1. With a 5\*5 gaussian mask the PSNR value was **19.89db** which is slightly better than linear filter with uniform mask.
2. As we increase the size of the filter to 7\*7, no changes observed in PSNR value and image quality.
3. As the value of sigma increased we noticed a slight decrease in the PSNR value, while no much visual changes were observed.

**3) Bilateral filtering:** A 5\*5 bilateral filtering is implemented and results were observed on Corn\_noisy.raw image. While, there is no significant improvement in the PSNR value, we can easily observe from the image that the sharp edges are preserved. The second coefficient is working as intended. The PSNR value was **19.9db.** I used a linear relationship between sigma\_c and sigma\_s and observed maximum PSNR value was noted for sigma\_c = 0.1 and sigma\_s = 1.

Bilateral filter produced more quality image than linear filters because it preserves edges.

4) **Non Local means:** The image was over smooth but there is a considerable jump in the PSNR value which was **22db.** The hyperparameter h is arbitrarily selected and tested. But as we observed, as h increases the PSNR value decreases. But when we used h=10\*a, spread factor, we got the maximum PSNR.

So ideally, h should be 10 times the standard deviation.

**5) BM3D:** Sadly, the output and PSNR values were similar to that of NLM algorithm.

**5) Mixed Noises in colored Images:**

1. Salt and pepper noise and Gaussian noise
2. Yes, we should perform denoising on individual channels .
3. We use cascading filters to remove mixed noise. First we remove salt/pepper noise using median filter and then we remove gaussian noise.

References:

1. Google Images.
2. Wikipedia.
3. Image Processing online journals.