EE569: Introduction to Digital Image Processing

ASSIGNMENT 1

Image Demosaicing

Contrast Enhancement using Histogram Equalization

Image Denoising

Suchismita Sahu | 22nd Jan 2019 | USCID: 7688176370

Problem 1: Image Demosaicing and Histogram Manipulation

(A) BILINEAR DEMOSAICING

I. Abstract and Motivation

The idea of this question is to Implement the simplest demosaicing method based on bilinear interpolation. In this method, the missing color value at each pixel is approximated by bilinear interpolation using the average of its two or four adjacent pixels of the same color as represented in Bayer Pattern

Most modern digital cameras acquire images using a single image sensor overlaid with a CFA (Color Filter Array) i.e. the red, green and blue values are not sampled at the same position, so we need to use demosaicing to convert these images into a viewable format of a color image.

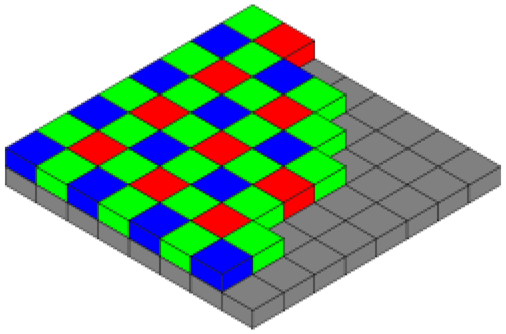


Figure 1: The Bayer arrangement of color filters on the pixel array of an image sensor. Each two-by-two cell contains two green, one blue, and one red filter.
(Source:https://en.wikipedia.org/wiki/Demosaicing)

The Bayer Pattern has alternating Red, Green filters for each odd row and alternating Green and Blue filters for every even row. Since, the human eye is most sensitive to green color, there are twice the number of green filters than red or green. I.e. 50% of the pattern is Green filters.

Since each pixel of the sensor is behind a color filter, the output is an array of pixel values, each indicating a raw intensity of one of the three filter colors. Thus, an algorithm is needed to estimate for each pixel the color levels for all color components, rather than a single component. We implement the bilinear interpolation algorithm here, for estimating the R, G and B components of each pixel, to render the color image output.

II. Approach and Procedure

Algorithm for Bilinear Interpolation Demosaicing:

1. Read Input Image.
2. Create a New Image data array to implement Boundary Extension by mirror-reflecting two rows and two columns each side across the input image border.
3. Implement Bilinear interpolation by averaging the neighbouring pixels according to the formulae: Corresponding to different locations

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

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

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

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

Equations for : B value at (3,4) , (b): G value at (3,4), (c): R value at (3,3) and (d): B value at (3,3)

4. Save the interpolated data into 3D new image data created.

III. Experimental Results

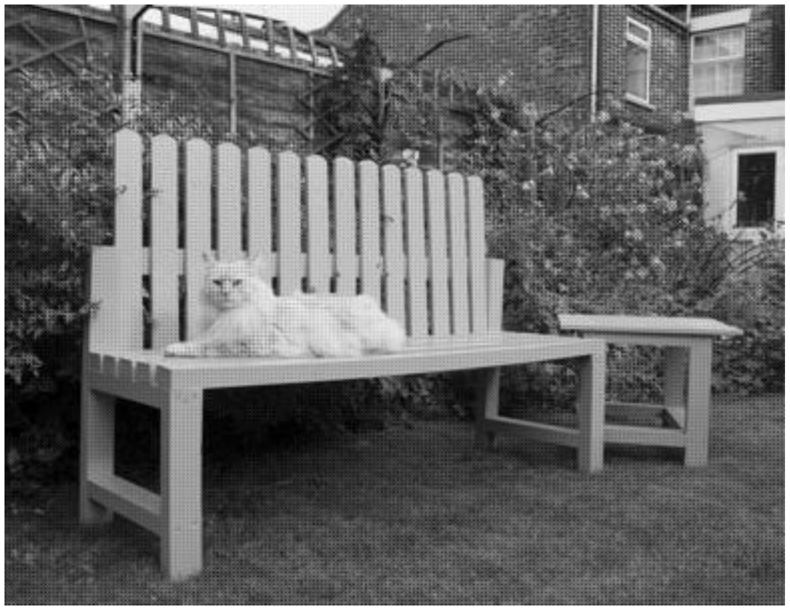


Figure 2 Input Raw Black and White Cat Image (cat.raw)

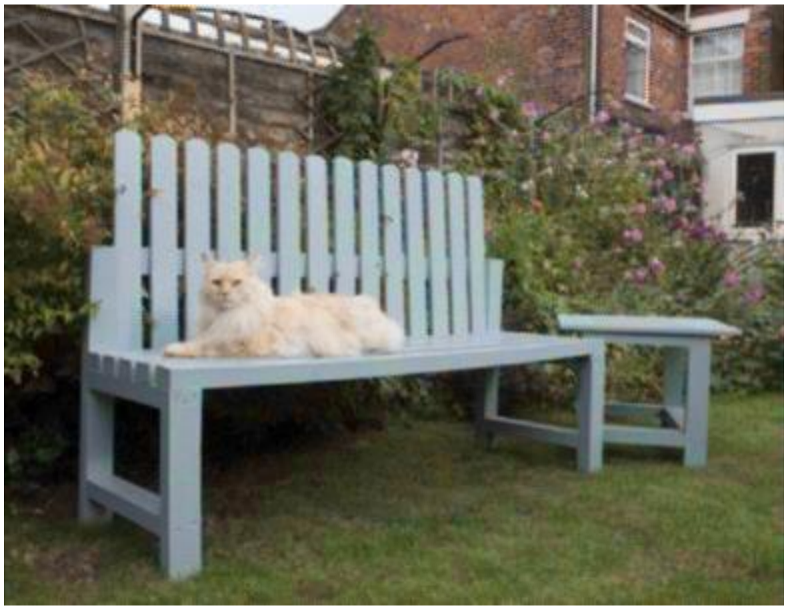


Figure 3 Output Cat Image after Demosaicing using Bilinear Interpolation

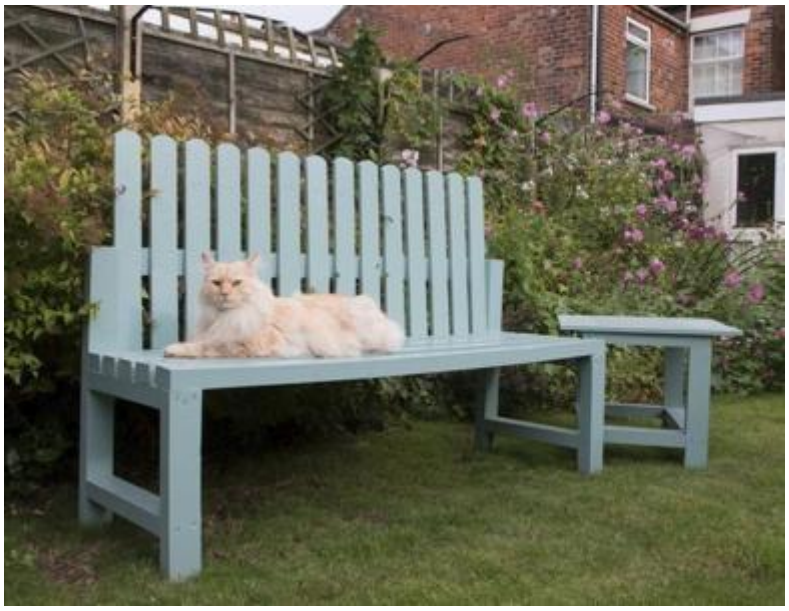


Figure 4 Original RGB Cat Image

IV. Discussion

We see certain abberations in the output image since visual quality is generally poor.

1. False-color Effect : The cat image looks odd colored. And pixelated. Similarly the leaves also show false color.

2. Zipper Effect: The output is blurred at edges because of averaging i.e. there are misalignments near the edges. It is properly evident in regions of high frequency. (e.g.. Edge of desk)

We might be able to improve the performance implementing non-linear methods and consider correlation among the RGB channels to avoid False color effect.

(B) MALVAR-HE-CUTLER (MHC) DEMOSAICING

I. Abstract and Motivation

As we find a burred and slightly color distorted image in bilinear interpolation, we implement MHC Demosaicing that uses larger neighborhood and defined pixels. The algorithm implements these fiters:

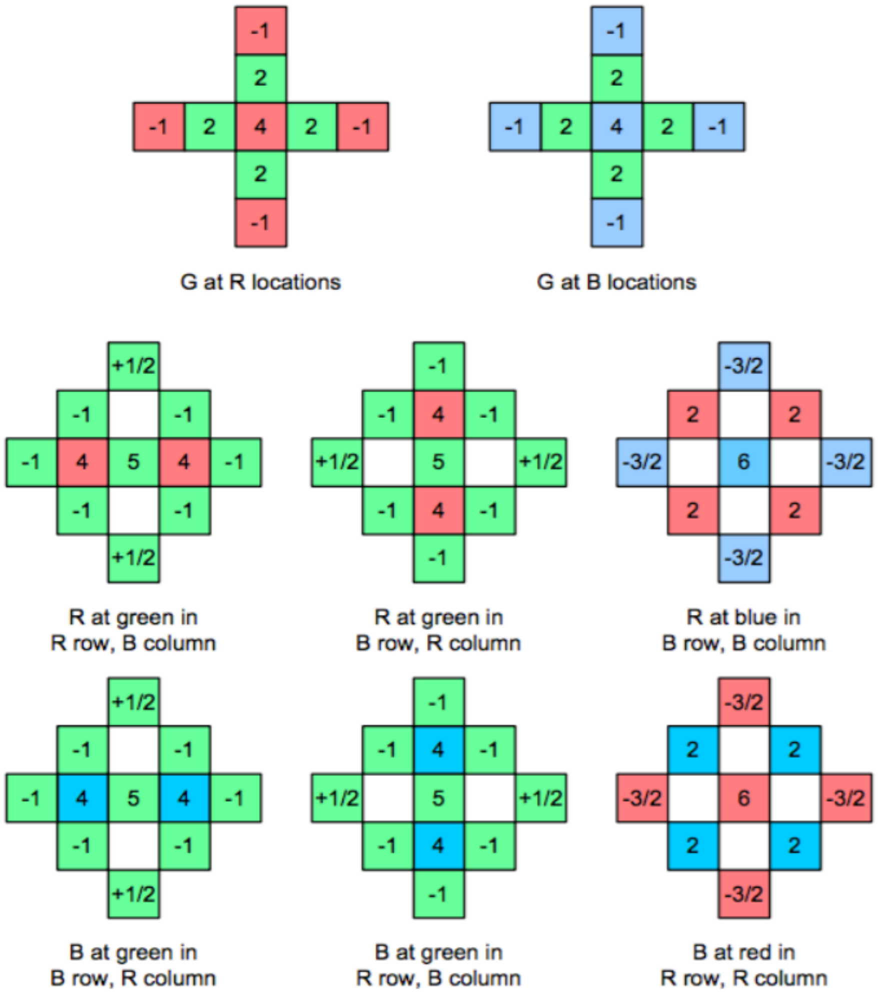

G at R locations
G at B locations
R at green in R row, B column
R at green in B row, R column
R at blue in B row, B column
B at green in B row, R column
B at green in R row, B column
B at red in R row, R column
-1
2
4
5
6
+1/2
-3/2

Figure 5 Filters used in MHC Algorithm

II. Approach and Procedure

Similar to Bilinear interpolation, we use the defined filters for each position of the pixel and take weighted average. The output is expected to be better than that of bilinear interpolation.

III. Experimental Results

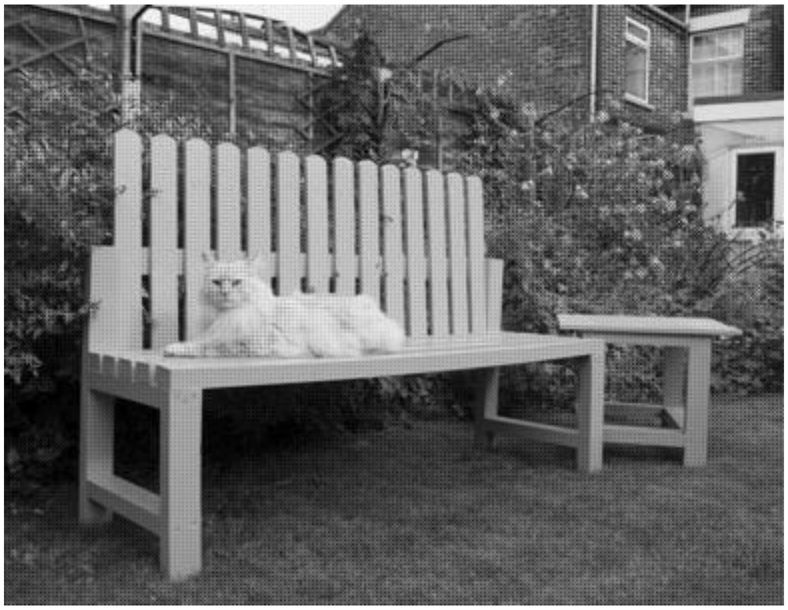


Figure 6 Input Raw Black and White Cat Image (cat.raw)

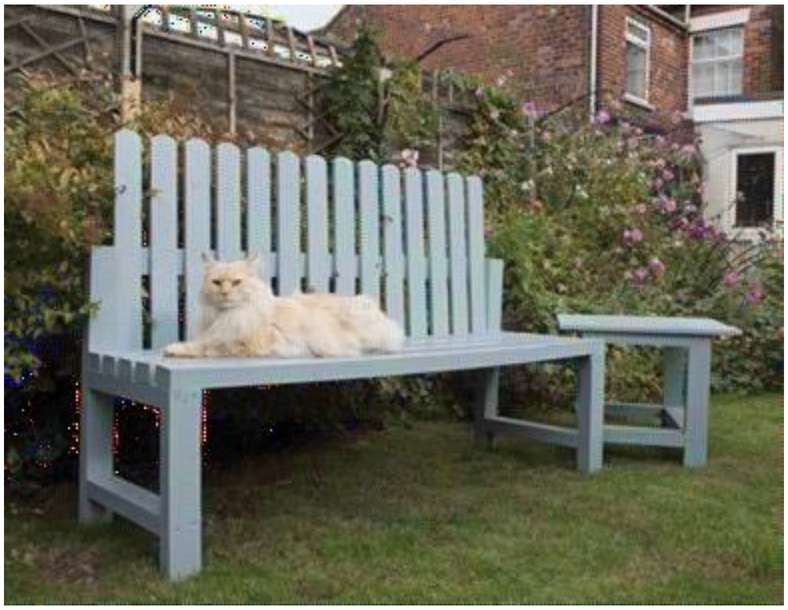


Figure 7 Output Cat Image after Demosaicing using MHC Algorithm

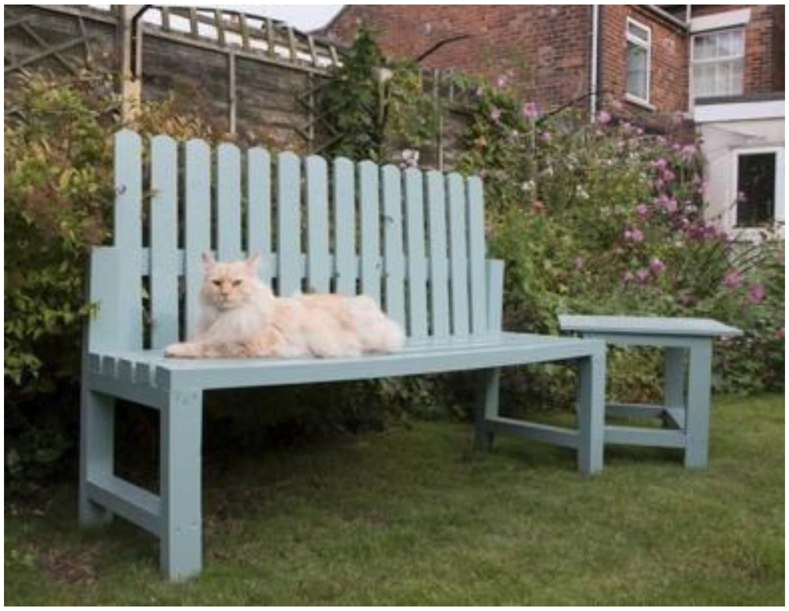
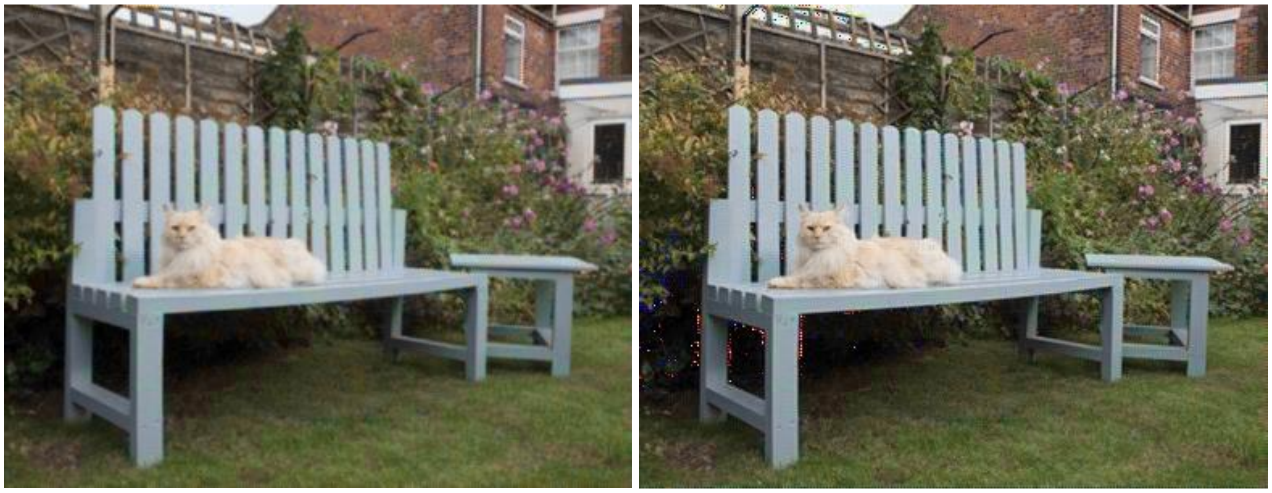


Figure 8 Original RGB CAt Image

IV. Discussion



Bilinear Interpolation vs MHC Demosaicing

The output of MHC Demosiacing is much sharper than that of Bilinear Interpolation. But we see some aberrated pixel colors near the edges. Like the Blue and Green spots near the legs of the desk.

Further, the zipper effect is reduced and high frequency regions have better visual quality.

(C) HISTOGRAM MANIPULATION FOR CONTRAST ENHANCEMENT

I. Abstract and Motivation

The idea of this question is to understand and implement Histogram equalization methods to enhance the image terms of contrast and overall visualization by the human eye. Since Image enhancement basically depends on visual evaluation which may vary from person to person, we don't have concrete methods to evaluate the performance of an Image Enhancement technique than visual appeal. The more soothing an Image is to the eye, better it is.

Histogram Equalization techniques have applications in X-ray photography and to enhance and recover images in too dark (e.g. night photography) or too bright photography situations where exposure to light is not moderated.

In histogram manipulation techniques, we handle the pixel intensities of an image, which are responsible for contrast of an image. We change the pixel intensity value distribution of an image so that the histogram of an Image is more spread out, than being concentrated to some pixel intensity regions.

We implement two methods here:

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

In this method, we consider the complete dynamic range of pixel intensities of the input image. The steps include cumulatively adding the histogram probability distributions and then mapping it on to the output image to get equalized output using the transfer function.

Probability of occurrence of each gray level is calculated by :

$$Pr(g(k)) = \frac{(nk)}{N} \quad for\ k = 0 \ldots L-1$$

Where,

k is the number of gray scale

N is the total number of pixels

L is the number of all possible intensity values

The transfer function that represents the CDF based equalized image is given by

$$T(k) = floor\left((L-1) \sum_{j=0}^{k} \frac{nj}{n} \right) k = 0 \ldots L-1$$

The output image is obtained by mapping each pixel value with grey scale k using the transfer function. Hence, the New Pixel value is renormalized as:

$$I(r,c) = CDF\big(I(r,c)\big) * 255$$

2. Method B: The cumulative probability-based (Bucket Filing) histogram equalization.

Method A might tends to introduce aberrations like extra contours and edge blurring. Also, the equalization is not defined properly.

To overcome these, we use the Bucket filling method. Here, first a map is created to store locations of pixel values and their intensities. The sorted pixels are then divided into buckets or bins according to this formula:

$$Number\ of\ pixels\ per\ bin = \frac{Total\ number\ of\ pixels}{256}$$

II. Approach and Procedure

Method A:

Algorithm for Transfer Function Based (CDF) Histogram Equalization:

1. Calculate the Histogram Data of the Input Image by counting the number of pixels of each intensity value in the grayscale image. i.e. (0-255).
2. Normalize the Histogram to obtain the Normalized probability distribution of the Image pixel intensity distribution.
3. Calculate the Cumulative Probability Distribution of the data from the Normalized Probability Distribution data.
4. Create Mapping matrix from the Cumulative Probability distribution by multiplying levels (255) and map the values to each pixel of the Image.

Method B:

Algorithm for Cumulative Probability Based (Bucket Filing Method) Histogram Equalization:

1. Calculate the Histogram Data of the Input Image by counting the number of pixels of each intensity value in the grayscale image. i.e. (0-255).
2. Sort the Histogram data in an ascending order and store the address of each pixel.
3. Divide the data into the grayscale levels i.e buckets (0-255) of (ImageLength*ImageWidth/256) i.e. 160000/256 = 625 pixels each.
4. Assign values from 0-255 to each bucket in order i.e 0 to first bucket, 1 to second bucket and so on.

5. Map the new assigned value pixels in their original location from the input image, to the output image.

III. Experimental Results

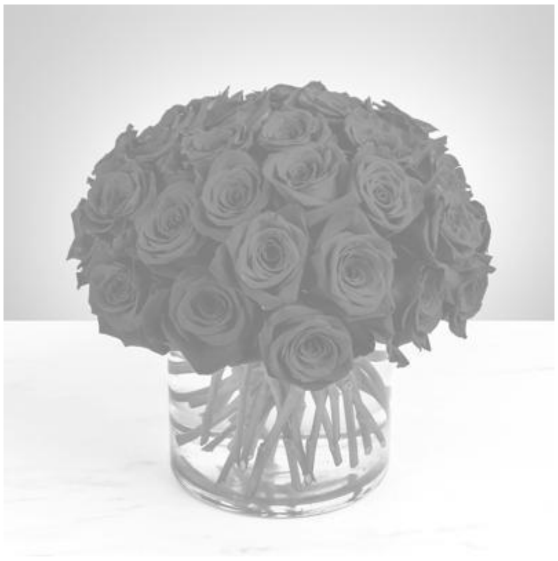


Figure 9 Input Image rose_bright

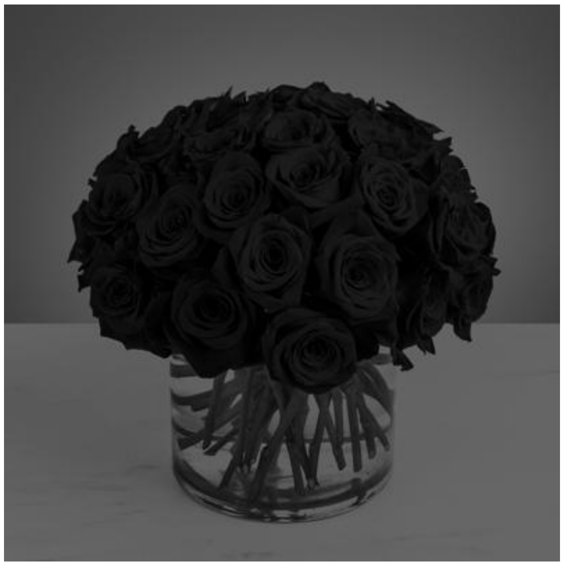


Figure 10 Input Image rose_dark

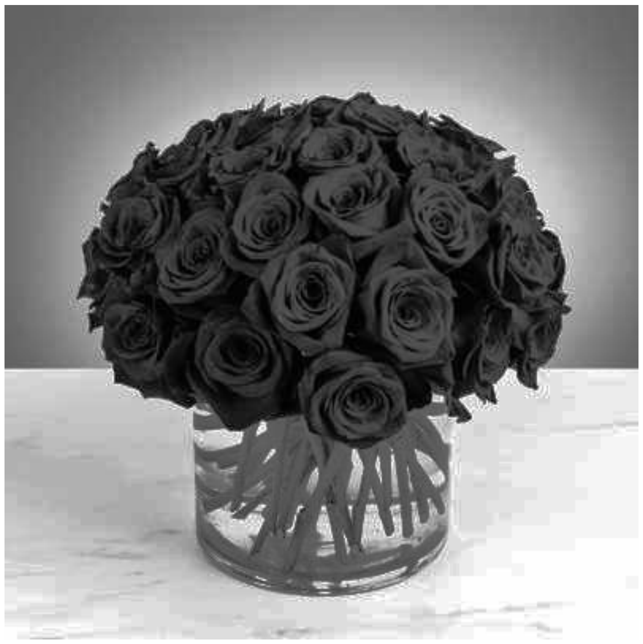


Figure 11 Method A Output on rose_bright

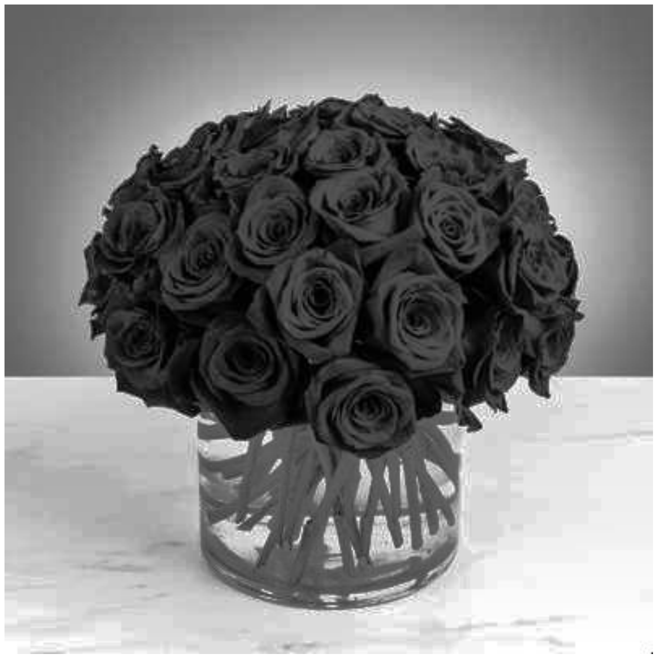


Figure 12 Method B Output on rose_bright

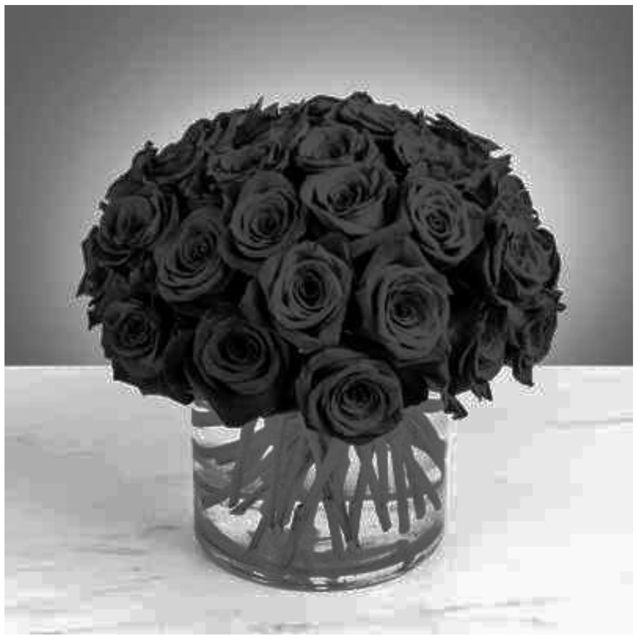


Figure 13 Method A output on rose_dark

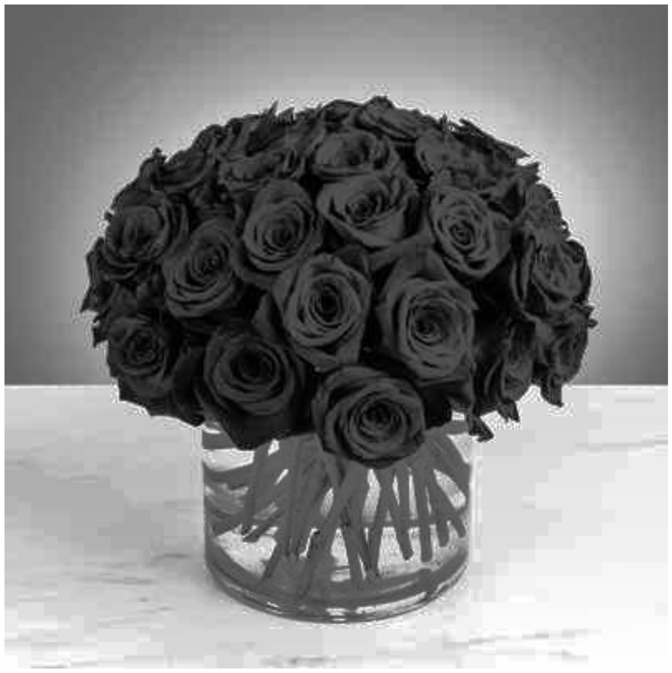


Figure 14 Method B Output on rose_dark

Histogram and Transfer Functions plots:

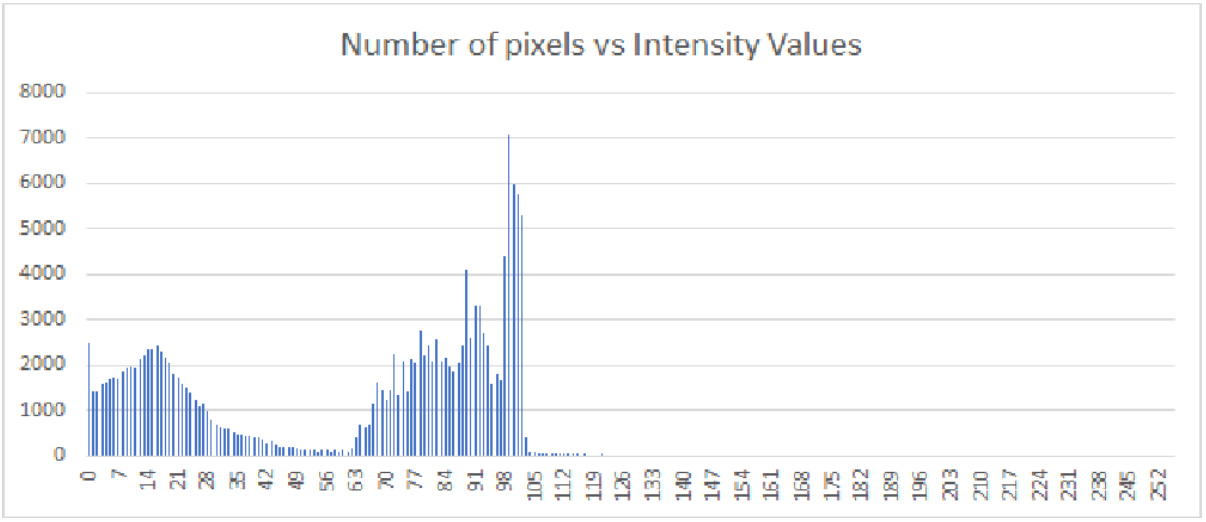


Figure 15 Histogram of Input Image rose_dark.raw

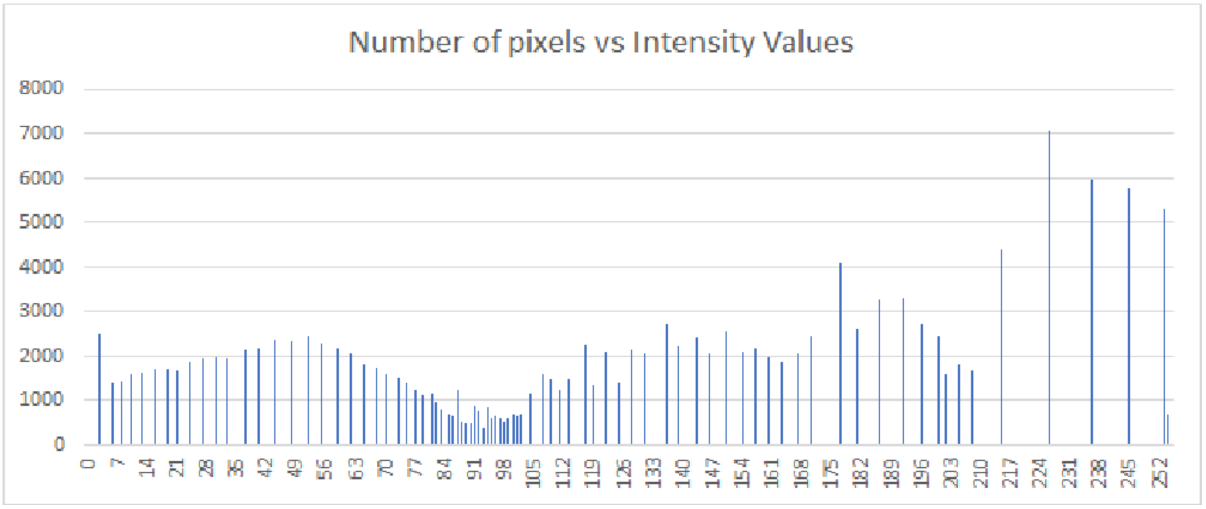


Figure 16 Histogram of Output of rose_dark.raw using Method A

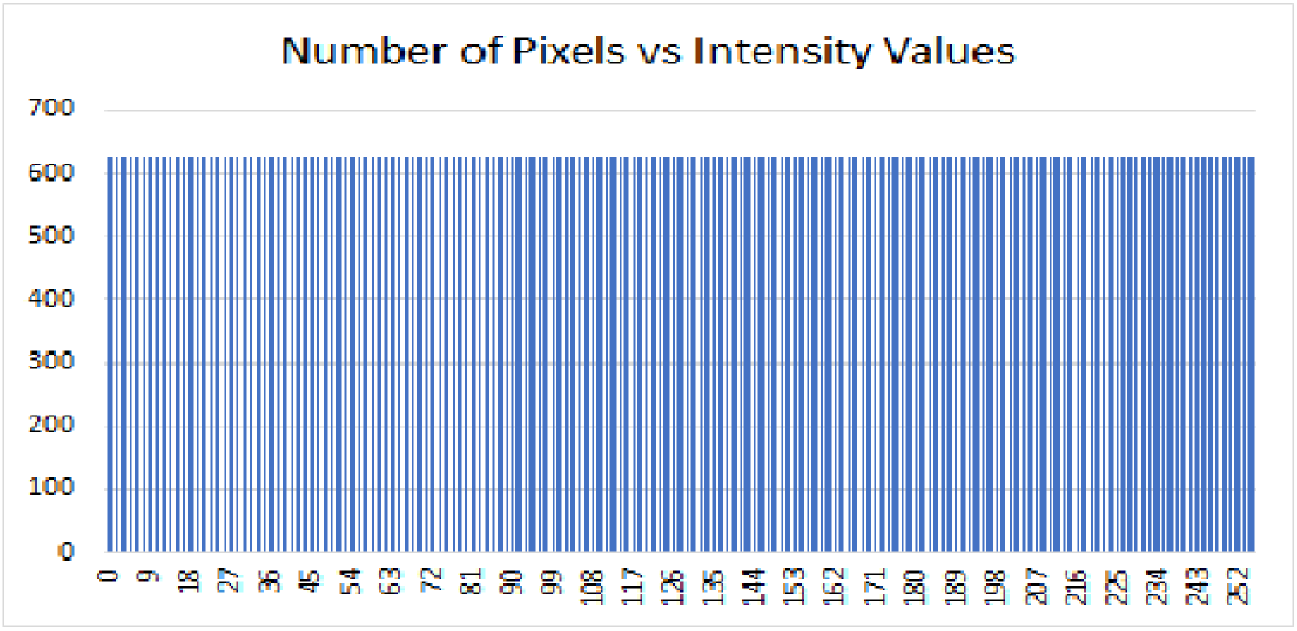


Figure 17 Histogram of Output of rose_dark.raw using Method B

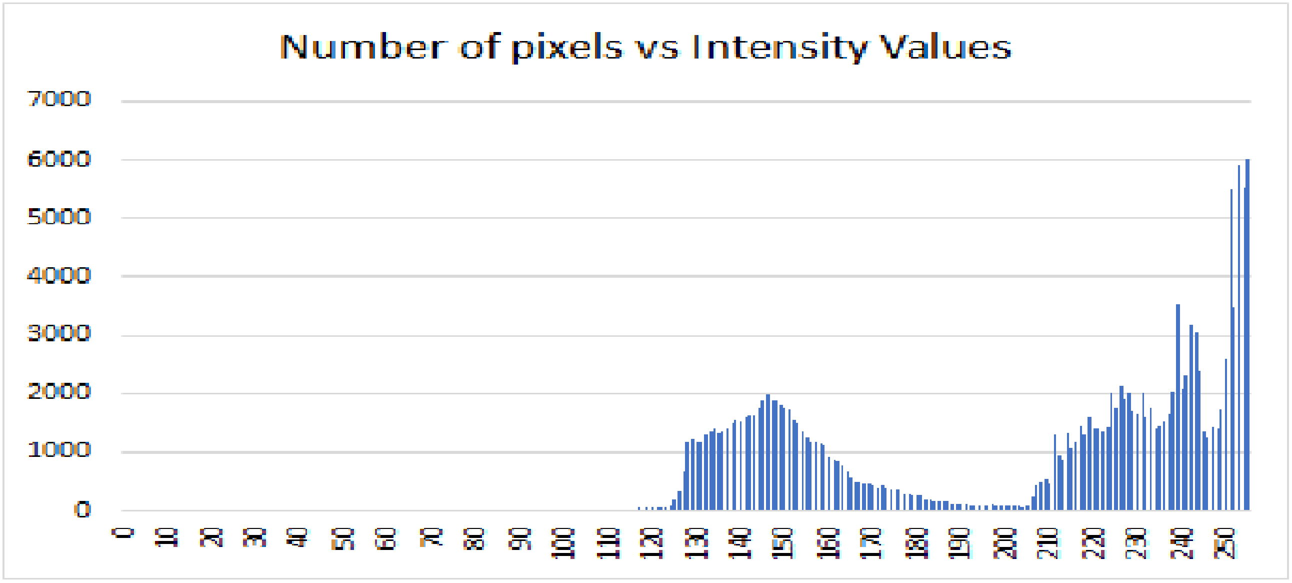


Figure 18 Histogram of Input Image rose_bright.raw

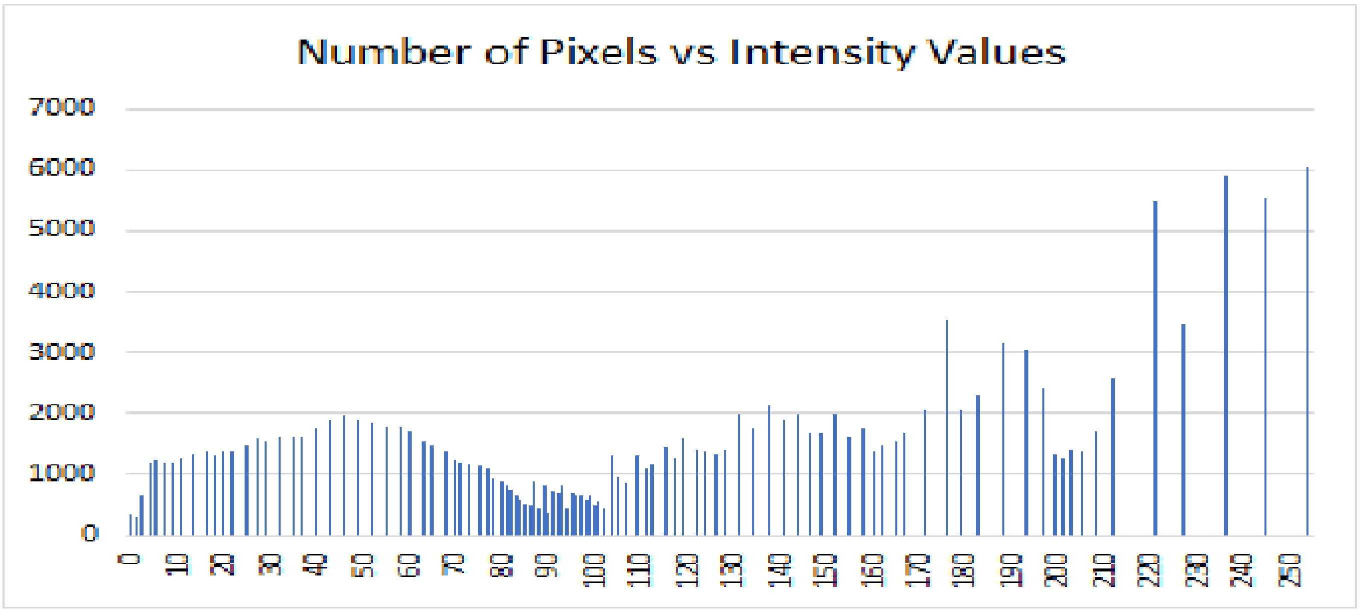


Figure 19 Histogram of Output Image rose_bright.raw using Method A

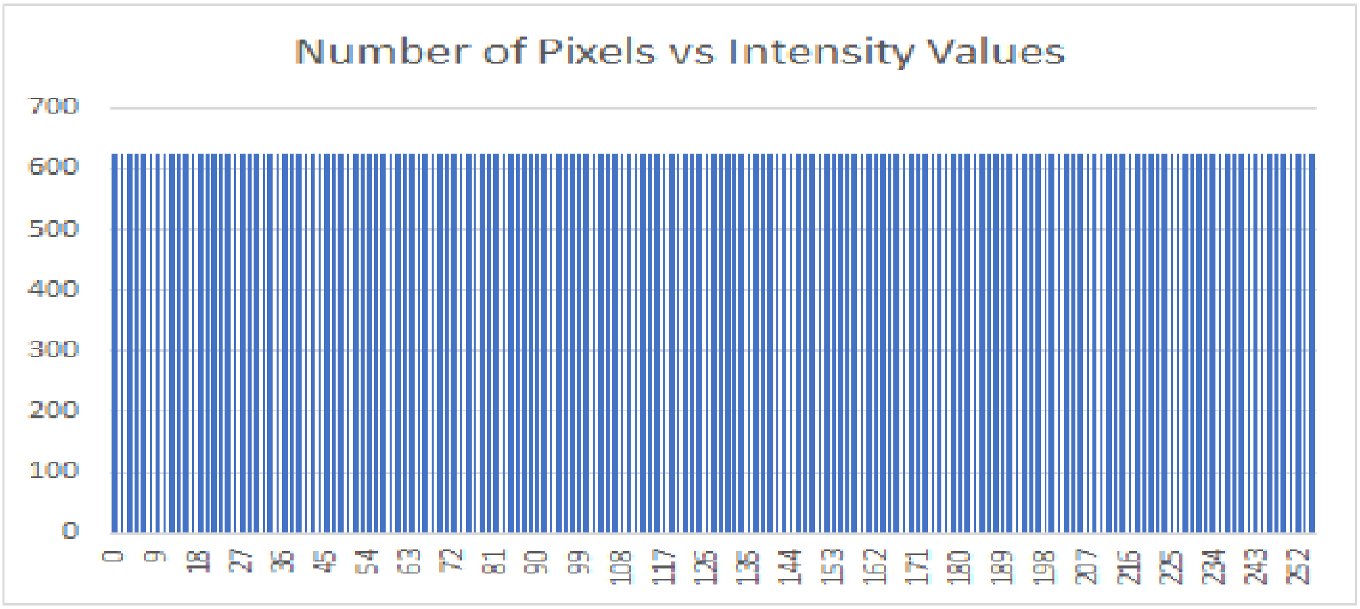


Figure 20 Histogram of Output Image rose_bright.raw using Method B

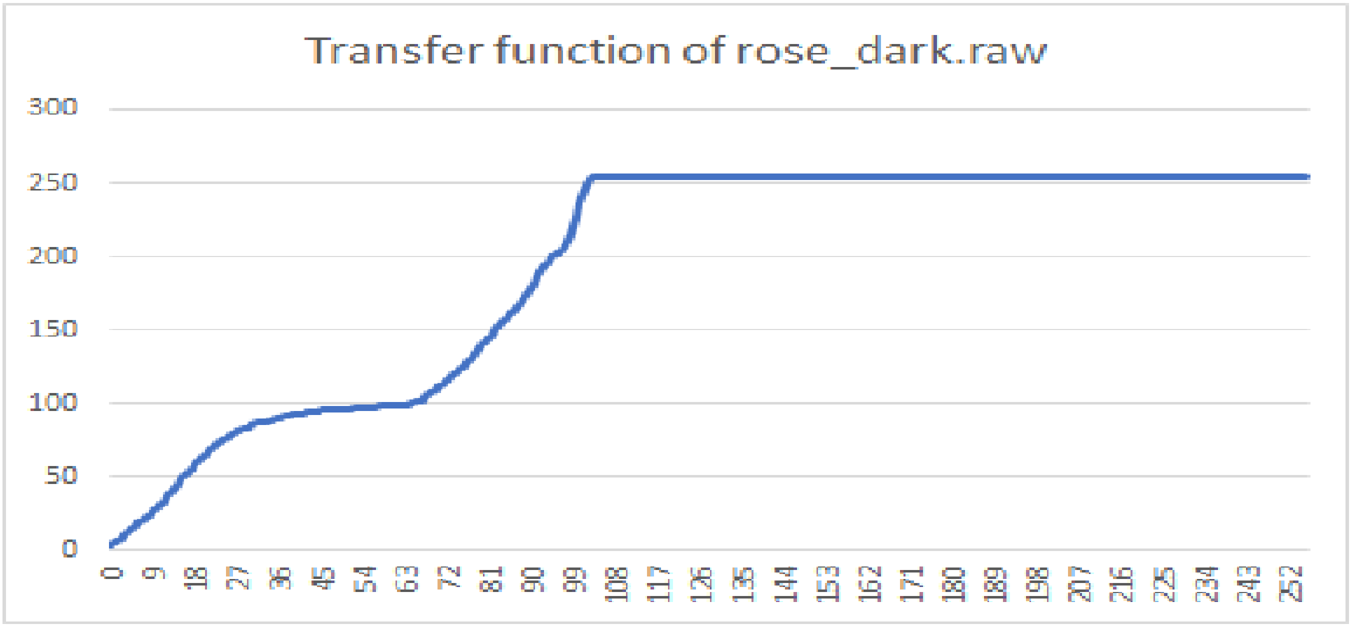


Figure 21 Transfer Function of rose_dark.raw

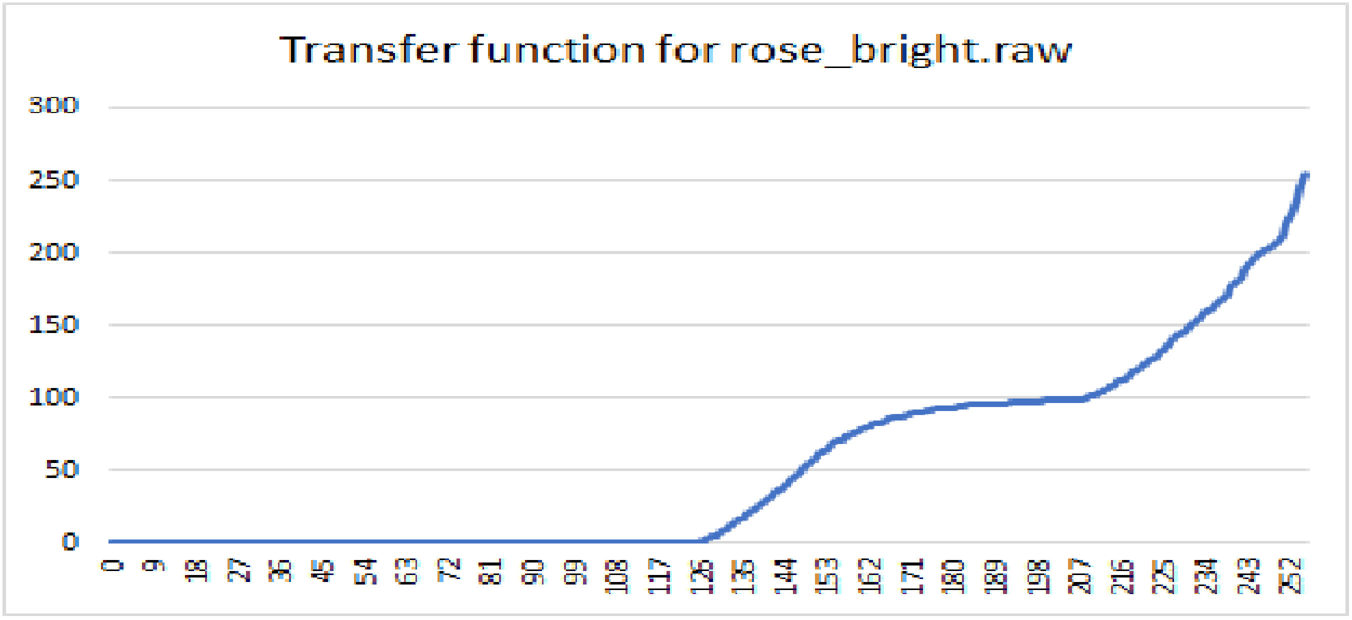


Figure 22 Transfer Function of rose_bright.raw

IV. Discussion

From the histogram equalized output images obtained for both rose_bright and rose_dark images, and the corresponding histogram plots, we can see that rose_bright has changed from being a brighter image to a darker image and similarly rose_dark has changed from being a darker image to a brighter image. Both the output Images are more soothing to the eye and the details of the image are clearly observed.

Both Method A and Method B both give an enhanced contrast image output of the input dark or bright image.

Even though Method A improves the distribution and the dynamic range of the pixels and gives improved contrast, the pixel intensities still don’t cover the entire range from 0-255. It either makes the histogram extend towards the right or left. Thus, the image still has pixel intensities concentrated towards the darker or brighter regions of the intensities. Thus, the image doesn’t have perfectly equalized output.

Method B on the other hand, makes the histogram perfectly equalized, thus giving a better enhanced image than given by method A. Here, every gray intensity has at least one pixel in it’s bucket, hence the distribution is more spread out.

Advantages in Method B output:

1. Better Contrast
2. Image more soothing to the eye
3. Sharper Edges
4. Lesser Distortion of Image
5. Linear Transfer Function for the entire range

Application of Method A and Method B to Mixed Image : rose_mix

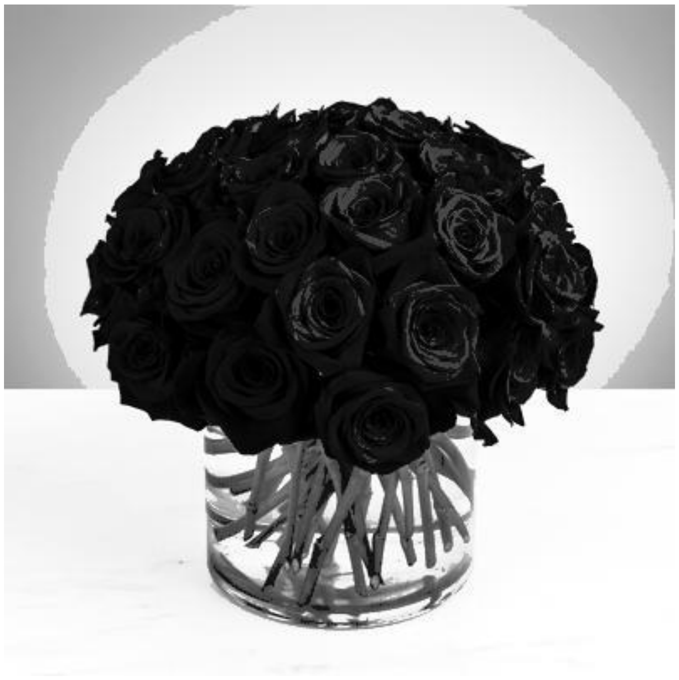


Figure 23 Input Mix Image rose_mix

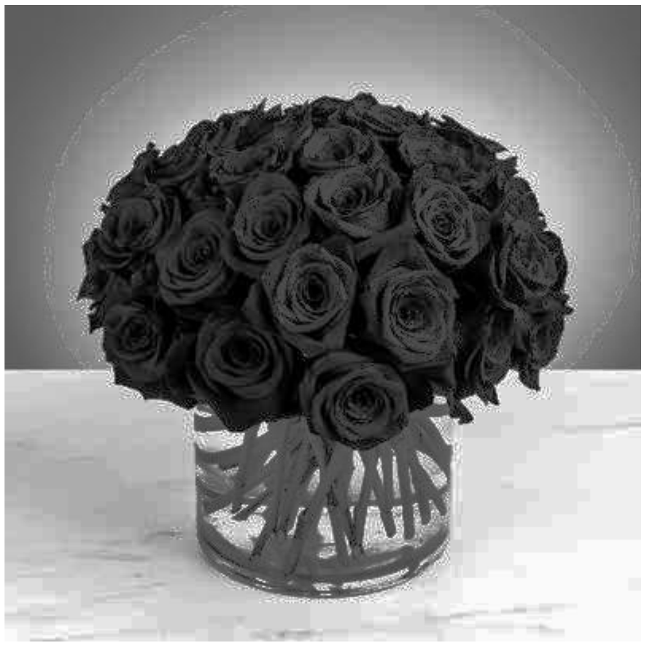


Figure 24 Method A output on rose_mix

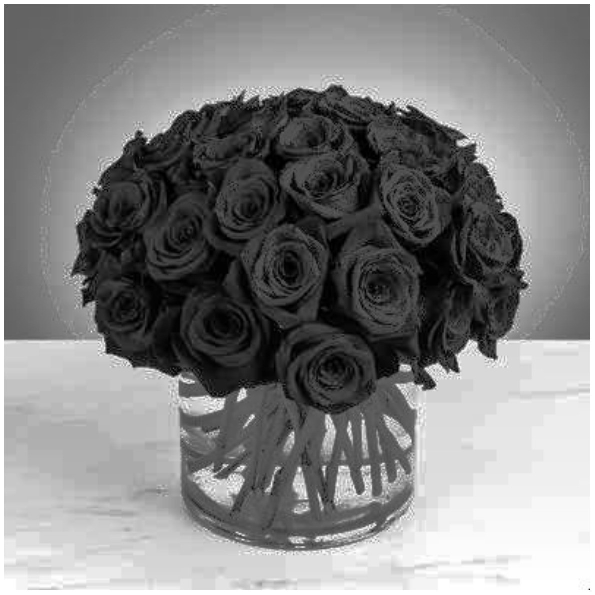


Figure 25 Method B Output on rose_mix

From the above outputs, we observe that Method B performs better on a mixed exposure image having intensities on both extremities. The output image obtained by method B is more soothing since the intensity values are perfectly equalized by bucket filling. Method A, even after improving the dynamic range of pixel intensities, fails to distribute the intensity pixels having extremities at both ends.

Method A performs desirable when there's extremity at one end of the intensity distribution. So, a way to make Method A work on mix image might be to separate out two curves of the histogram, each at one extremity of intensity values and equalize them separately to get the final output image.

Problem 2: Image Denoising

(A) GRAY-LEVEL IMAGE

I. Abstract and Motivation

Denoising is an important concern in any Signal Processing Pipeline. Noise distorts Image which might lead to loss of information and other issues. Noises can be of many types. The most common type of noises are uniform noise, impulse noise or salt and pepper noise, gaussian noise, etc.

The idea of this section is to understand the type of distortions in images caused by noises of different type and try to denoise the images using different filters and algorithms.

The input image given has salt and pepper noise. There's no said thumb rule for deciding the filter to remove a particular kind of noise, or that certain filters only remove a particular kind of noise. Hence a lot of experimentation is needed to properly remove noise from an input image. Further, we can use a mix and match and cascading of filters to have end output image with least value of noise intensities.

Filters:

1. MEAN FILTER
   In this filtering technique, the target pixel is replaced by the average of the pixels in its neighborhood pixel space depending on the kernel window size specified. This averaging process increases signal intensity and decreases noise intensity by making use of the uncorrelation between the image pixel intensities and noise pixel intensities. The output image of this filter is governed by the following set of equations :

$$Y(i,j) = \frac{\sum_{k,l} I(k,l)w(i,j,k,l)}{\sum_{k,l} \underset{w_1}{w}(i,j,k,l)}$$

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

   A typical 3*3 kernel for implementing Mean Filter is:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

2. GAUSSIAN FILTER
   In this filtering technique, the same concept as mean filter is followed, but the filter coefficients are obtained from the gaussian function:

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

   Example of 5*5 Gaussian Filter :

$\frac{1}{273}$

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

   I obtained Kernel values for gaussian filter from this website:
   http://dev.theomader.com/gaussian-kernel-calculator/

3. BILATERAL FILTER
   Since Linear filtering techniques like gaussian and mean filter cause blurring of edges, we use advanced filtering technique to overcome this. Bilateral Filter is such an option.
   The weights for the filter are defined by the following equation :

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

Where the tuning parameters for improving the performance of the filter are sigma_c and sigma_s.

4. NON-LOCAL MEAN FILTER
This is another non-linear advanced filtering technique. This filtering technique makes use of the fact that if two pixels have a similar neighborhood, they have higher probability of being similar. The filter is governed by the following equations:

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

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

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

and

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

PSNR(Peak Signal to Noise Ratio)
This quality metric is used to assess the performance of the denoised output image.
The formula used are:

$$\text{PSNR (dB)} = 10\log_{10}\left(\frac{\text{Max}^2}{\text{MSE}}\right)$$

$$\text{where } \text{MSE} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(Y(i,j) - X(i,j)\right)^2$$

X : Original Noise-free Image of size $N \times M$

Y : Filterd Image of size $N \times M$

Max: Maximum possible pixel intensity $= 255$ (Source: Lecture notes)

II. Approach and Procedure

The first step is to chose a 2D window size for capturing the neighboring pixels. The filter traverses through the image and replaces the center pixel with the new pixel value created using the filter.

Boundary Extension of Images: For implementing the filter windows on the edge pixels of the image, we need to pad dummy pixels on the boundaries of the image depending on the size of the filter window selected. There are four possible methods of boundary extension:

1. Zero Padding

2. Pixel Replication
3. Reflection
4. Linear Extrapolation.

Here, I implement pixel Refllection on the image borders as shown in the example figure.

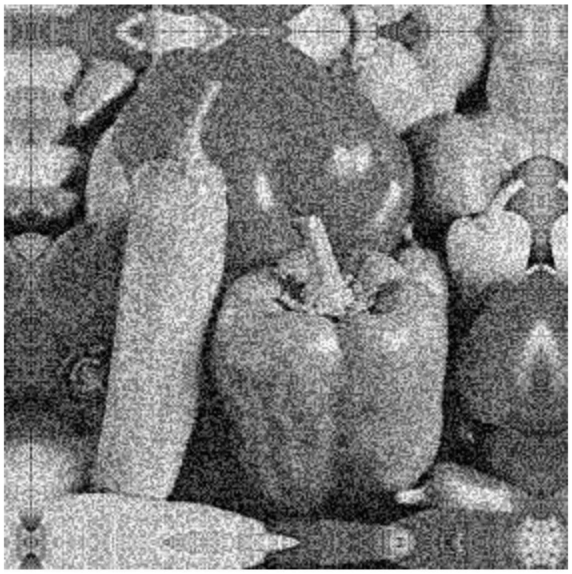


Figure 26 Example of Boundary Extended Image with Mirrored Rows and Columns for filtering operation (length+26,width+26)

Algorithm for Implementing Denoising :

1. For Gaussian and Linear Filters :
    a. Define Linear and Gaussian Filter Kernels
    b. Augment the image i.e implement boundary extension.
    c. Define a convolution function that convolves each input pixel with the corresponding filter. The final output pixel is the weighted average of the filter kernel convolution with the pixel.
    d. Iterate through the entire image pixel, convolve it with the corresponding filter kernel and store the new pixel in another output image data created.
2. For Bilateral Filter:
    a. Define Bilateral Filtering Function
    b. Augment the image i.e implement boundary extension.
    c. Iterate through the entire image pixel, convolve it with the corresponding bilateral filtering function and store the new pixel in another output image data created.
3. For NLM Filter
    a. Define NLM filter Function as per the equation.
    b. Augment the image i.e implement boundary extension.
    c. Iterate through the entire image pixel, convolve it with the corresponding NLM filter kernel and store the new pixel in another output image data created.

III. Experimental Results

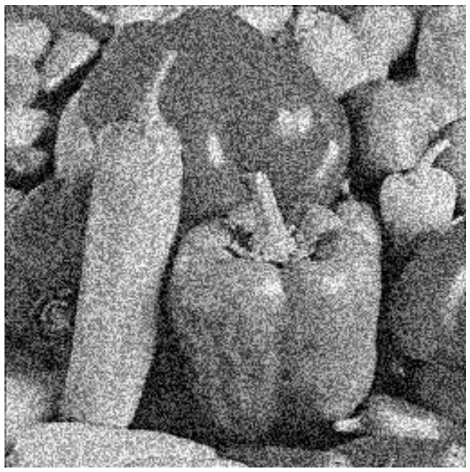


Figure 27 Input Image with Uniform Noise, pepper_uni.raw

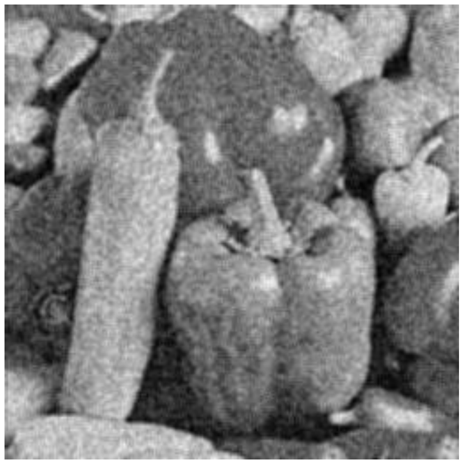


*Figure 28 Denoised Image Using Mean Filter 3*3*

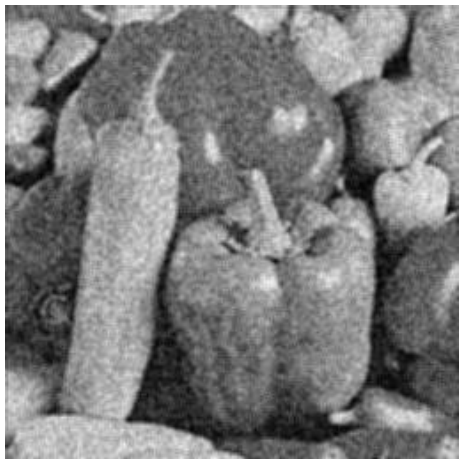


*Figure 29 Denoised Image using Mean Filter 5*5*

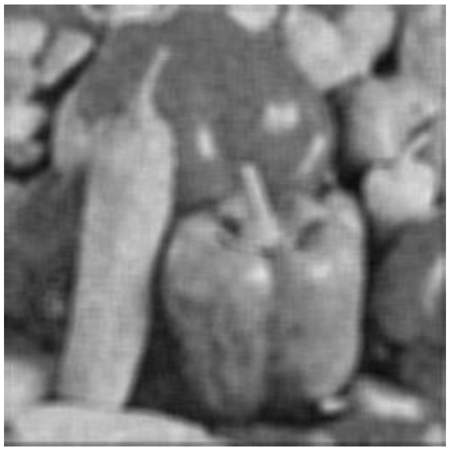


*Figure 30 Denoised Image using Mean Filter 7*7*

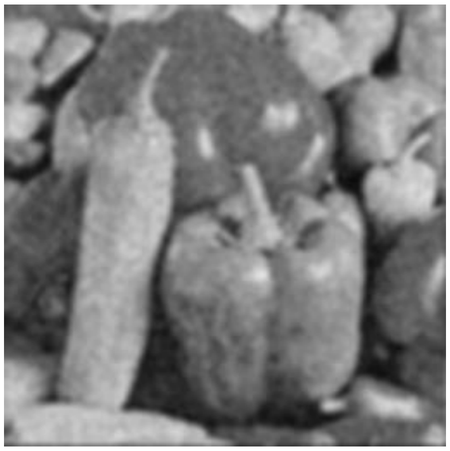


*Figure 31 Denoised Image using Gaussian 3*3 Filter (Sigma 1)*

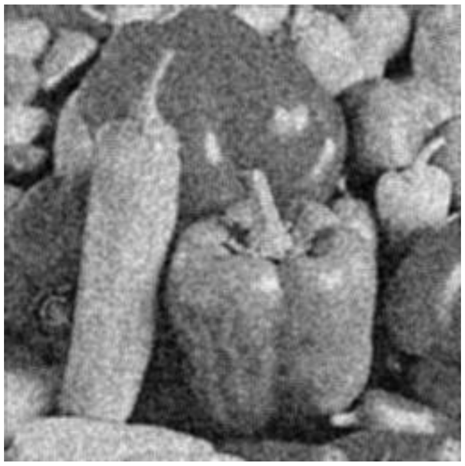


*Figure 32 Denoised Image using Gaussian 3*3 Filter (Sigma 2)*

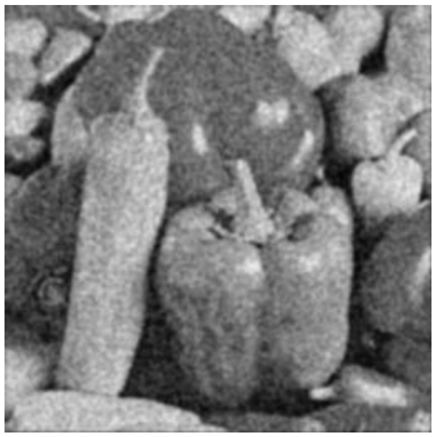


*Figure 33 Denoised Image using Gaussian 5*5 Filter (Sigma 1)*

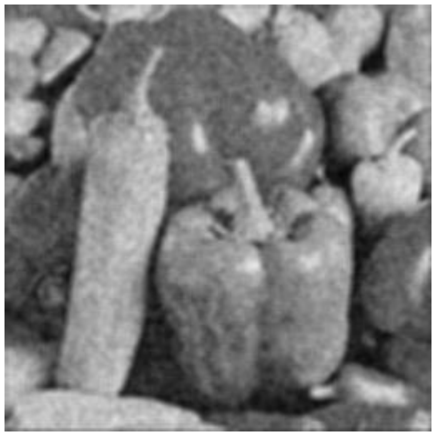


*Figure 34 Denoised Image using Gaussian 5*5 Filter (Sigma 2)*

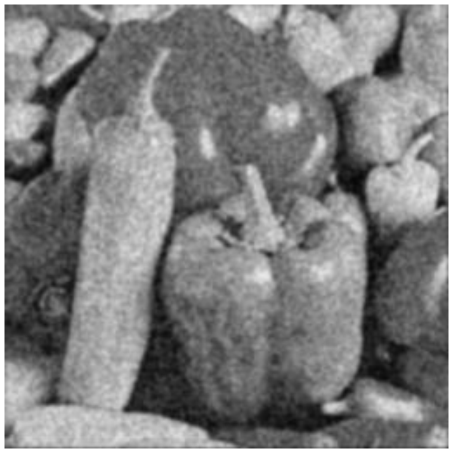


*Figure 35 Denoised Image using Gaussian 7*7 Filter (Sigma 1)*

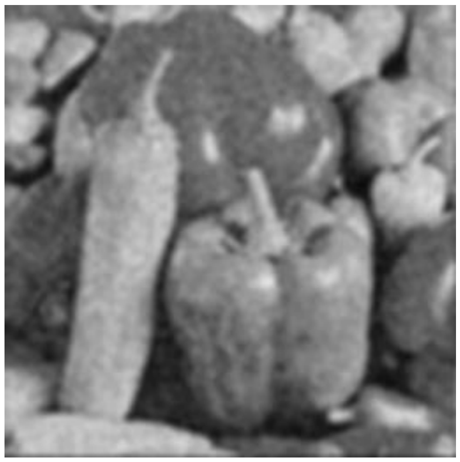


*Figure 36 Denoised Image using Gaussian 7*7 Filter (Sigma 2)*

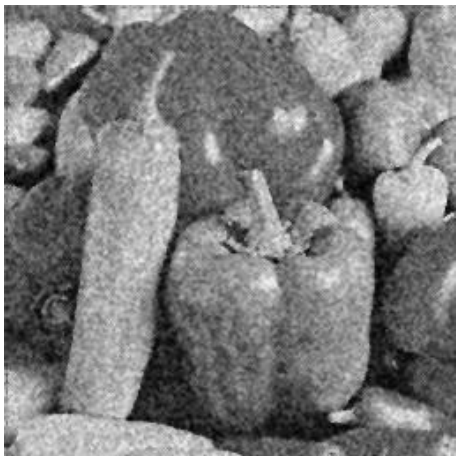


*Figure 37 Denoised Image using Bilateral Denoising Filter (3*3)*

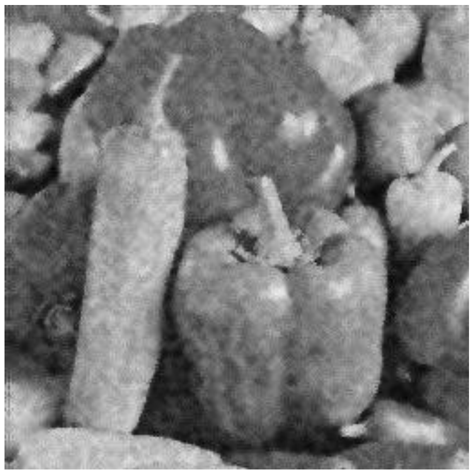


*Figure 38 Denoised Image using Bilateral Denoising Filter (5*5)*

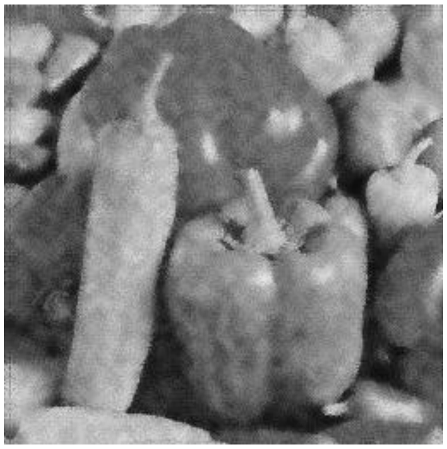


*Figure 39 Denoised Image using Bilateral Denoising Filter (7*7)*

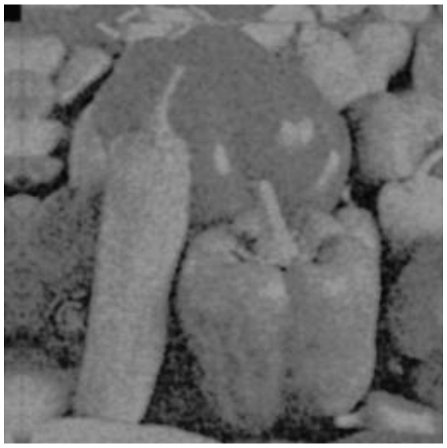


Figure 40 Denoised Image using Non-Local-Mean Filter Method

PSNR table:

N = 3 (Mean Filter)	PSNR 24.96
N = 5 (Mean Filter)	PSNR 24.37
N = 7 (Mean Filter)	PSNR 23.77
N = 3 (Gaussian Filter)	PSNR 24.91
N = 5 (Gaussian Filter)	**PSNR 28. 31**
N = 7 (Gaussian Filter)	PSNR 24. 76

IV. Discussion

We know that, higher the PSNR, better the Denoising. From the above results, we can see that, the filtering is best for Filter Size :

Further, window size plays an important role in denoising. On increasing the window size, we get better denoised output. But, it also comes at the cost of more deblurring in the image.

Higher PSNR indicated better denoising, but it doesn't take into consideration burring effect. So, we might have an image which is more blurred but it also might have high PSNR. Further, window size affects sharpness of Image. We get more blurring of image as we increase the window size

(B) COLOR IMAGE

I. Abstract and Motivation

The idea of this section to remove noise from a three channel RGB Image. Denoising for RGB images is slightly different from denoising for Grayscale images because of the different intensity distribution in corresponding each channel R,G and B.

The Image given here has both back-white spots i.e. Impulse(Salt and Pepper) Noise and gaussian Noise. Hence we need to implement a cascade of filters handling different filtering techniques.

II. Approach and Procedure

To remove salt and pepper noise, I am implementing median filter of different window sizes. The filter is implemented on each channel separately.

III. Experimental Results

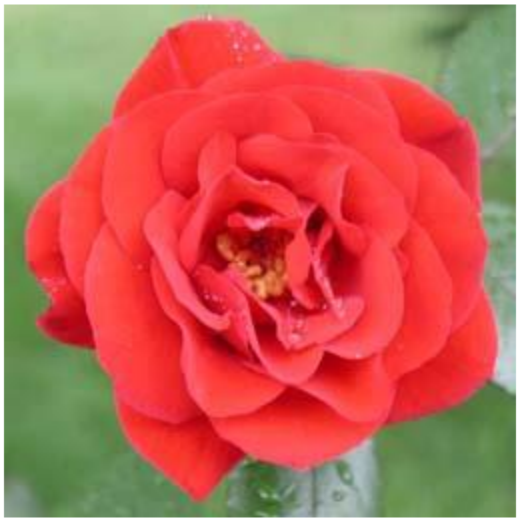


Figure 41 Original rose_color.raw Image

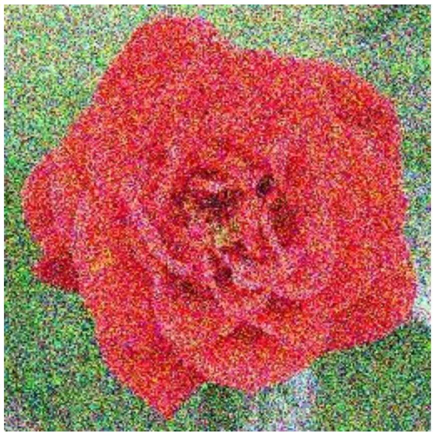


Figure 42 Input Noisy rose_color_noise.raw Image

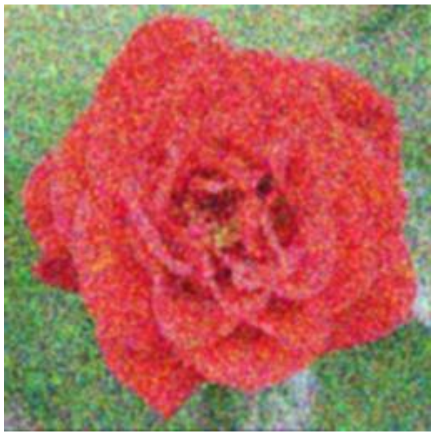


*Figure 43 Denoised rose_color_noise Image using Gaussian Filter(5*5)*

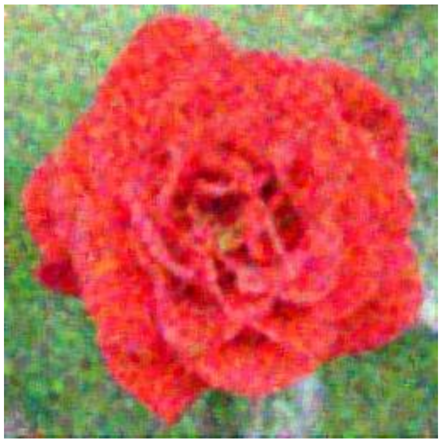


*Figure 44 Denoised rose_color_noise.raw Image using Cascaded Gaussian Filter(5*5) and Median Filter(5*5)*

IV. Discussion

On observing the histogram of the input noisy rose image, the channels containing mix noise tend towards having bell curve distribution. In conclusion, B channel has majorly gaussian noise whereas R ang G have majorly impulse noise.

For getting the best denoised output, I implemented Gaussian Filter followed by Median Filter. Median filter removes impulse noise easily whereas gaussian filter removes gaussian noise easily. Thus the combination works well. I obtained better results for Gaussian -> Median than Median -> Gaussian. Hence the order of cascading matters.

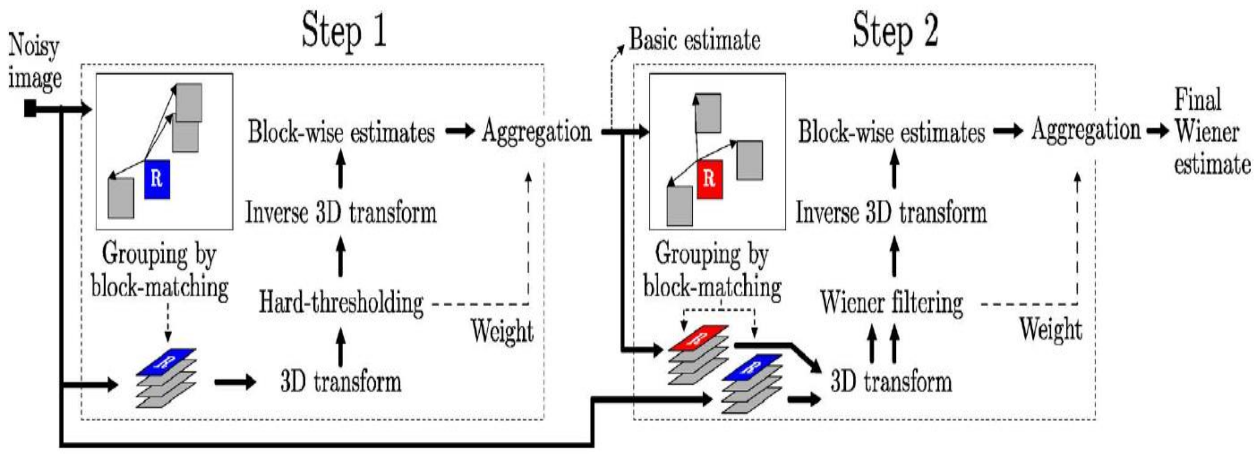
Further, window size plays an important role in denoising. On increasing the window size, we get better denoised output. But, it also comes at the cost of more deblurring in the image.

(C) SHOT NOISE

I. Abstract and Motivation

The idea behind this section is to familiarize oneself with shot noise and implement techniques to denoise shot noise image. Theoretically shot noise is the hardest to remove.

BM3D algorithm is a collaborative filtering technique used in which we divide input image into various 2D blocks and similar blocks are then stacked together to form another 3D stacked group. The next steps include thresholding coefficients after transform, then converting back to 2D stacks to give estimate of image. In the next step weiner filtering is implemented which performs really well for denoising.



BM3D algorithm

II. Approach and Procedure

For removing Shot Noise, we use the Anscombe Transform of Image to transfer the image to Anscombe Domain, implement filtering operation in the Anscombe domain, then implement Inverse Anscombe Transform on the image to get back the denoised output Image.

Also, we implement BM3D transform on the image to compare the performances.

III. Experimental Results

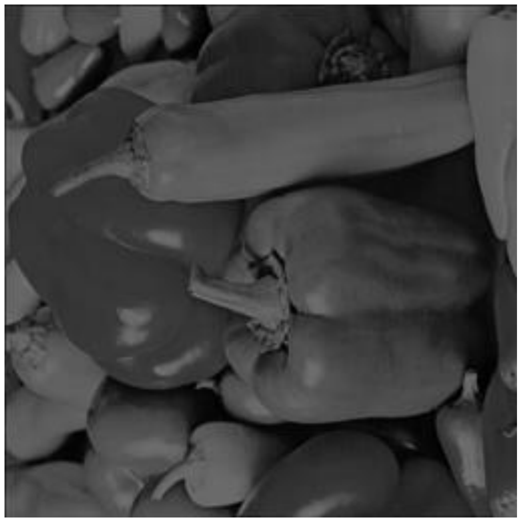


Figure 45 Original pepper_dark Image

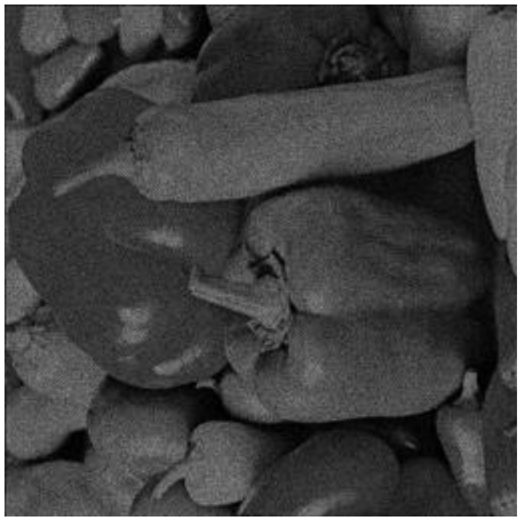


Figure 46 Input Noisy Pepper_dark Image

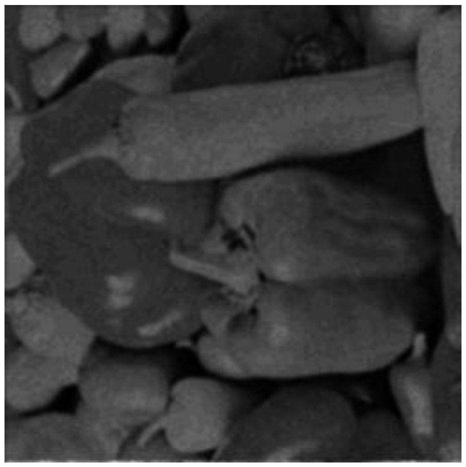


Figure 47 Output of Gaussian Filtering Before Inverse Anscombe Transform

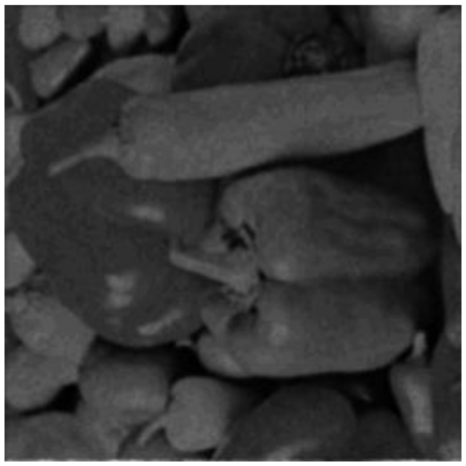


Figure 48 Denoised Image Using Gaussian Filter, Sigma 1 and Biased Inverse Anscombe Transform

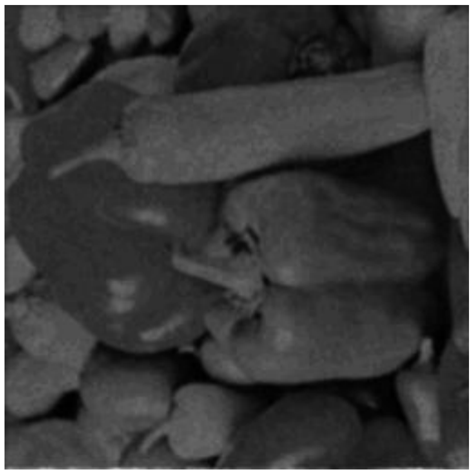


Figure 49 Denoised Image Using Gaussian Filter, Sigma 1 and Unbiased Inverse Anscombe Transform

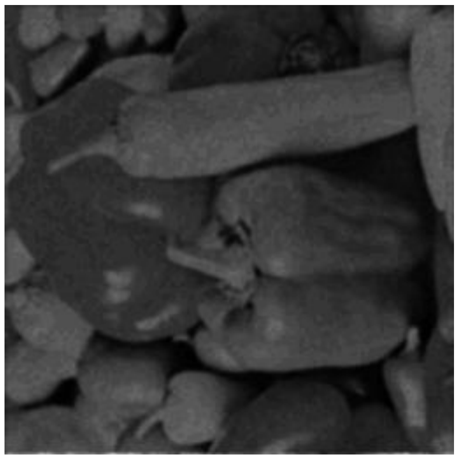


Figure 50 Denoised Image Using Gaussian Filter, Sigma 2 and Biased Inverse Anscombe Transform

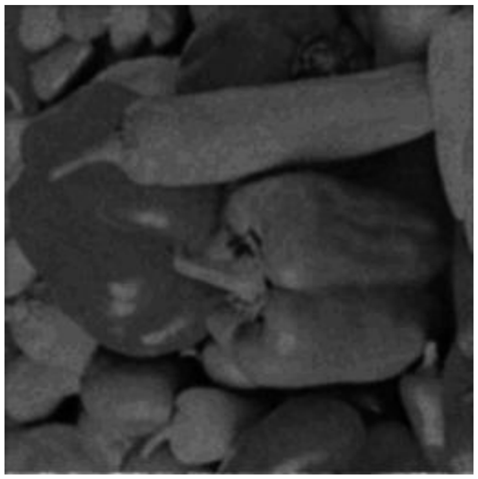


Figure 51 Denoised Image Using Gaussian Filter, Sigma 2 and Unbiased Inverse Anscombe Transform

IV. Discussion

Best Denoising is obtained for 5*5 filters combination.

Further, on comparing the output with BM3D output, we see that BM3D works best on denoising since it is both spatial domain as well as frequency domain filter.

Higher PSNR indicated better denoising, but it doesn't take into consideration burring effect. So, we might have an image which is more blurred but it also might have high PSNR. Further, window size affects sharpness of Image. We get more blurring of image as we increase the window size.

REFERENCES:

1. https://en.wikipedia.org/wiki/Demosaicing
2. http://www.ipol.im/pub/art/2011/g_mhcd/?utm_source=doi
3. http://dev.theomader.com/gaussian-kernel-calculator/
4. https://www.math.uci.edu/icamp/courses/math77c/demos/hist_eq.pdf
5. http://www.cs.tut.fi/~foi/GCF-BM3D/
6. http://www.librow.com/articles/article-1
7. http://www.cplusplus.com/reference/
8. https://stackoverflow.com/questions/6533570/implementation-of-non-local-means-noise-reduction-algorithm-in-image-processing