Digital Image Processing HW1 Report

Royston Marian Mascarenhas

Table of Contents

Problem 1: Image Demosaicing and Histogram Manipulation

A. Bilinear Interpolation

I. Abstract and Motivation

What it does: Cameras have 1 sensor for each pixel and it is sensitive to only a certain wavelength of light, in our case, just red, blue and green. Bilinear Demosaicing helps calculate the other two colour wavelengths of light so that the pixel need not be biased to a particular wavelength.

Why use it?

- ➢ Without demosaicing, the image has only 1 component of a pixel at each location which quantizes the image very roughly resulting in a non aesthetic view.
- ➢ Bilinear interpolation demosaicing is not computationally intensive.

[To summarize] A **demosaicing** (also de-mosaicing,**demosaicking** or debayering) algorithm is a digital image process **used** to reconstruct a full color image from the incomplete color samples output from an image sensor overlaid with a color filter array (CFA) [1]

II. Approach

Theory and Procedure:

Bayer's array consists of a color filter array in the order given below:

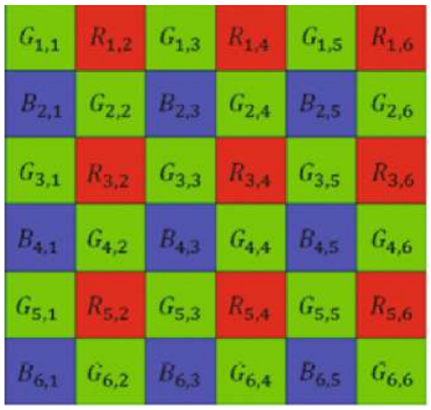

$G_{1,1}$ $R_{1,2}$ $G_{1,3}$ $R_{1,4}$ $G_{1,5}$ $R_{1,6}$
$B_{2,1}$ $G_{2,2}$ $B_{2,3}$ $G_{2,4}$ $B_{2,5}$ $G_{2,6}$
$G_{3,1}$ $R_{3,2}$ $G_{3,3}$ $R_{3,4}$ $G_{3,5}$ $R_{3,6}$
$B_{4,1}$ $G_{4,2}$ $B_{4,3}$ $G_{4,4}$ $B_{4,5}$ $G_{4,6}$
$G_{5,1}$ $R_{5,2}$ $G_{5,3}$ $R_{5,4}$ $G_{5,5}$ $R_{5,6}$
$B_{6,1}$ $G_{6,2}$ $B_{6,3}$ $G_{6,4}$ $B_{6,5}$ $G_{6,6}$

Figure 1: Bayer's array

At each pixel location, we calculate the values of the other two components (which are not captured) by taking a **local average** of immediate neighbouring pixels of the corresponding component.

For example, at R(3,4), the red component would be R^(3,4) = R(3,4) and the blue B^(3,4) and green G^(3,4) components of that pixel are calculated by taking the local average of neighbouring pixels of that color:

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

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

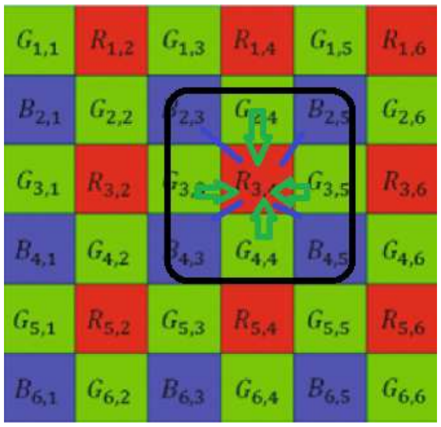

$G_{1,1}$ $R_{1,2}$ $G_{1,3}$ $R_{1,4}$ $G_{1,5}$ $R_{1,6}$
$B_{2,1}$ $G_{2,2}$ $B_{2,3}$ $G_{2,4}$ $B_{2,5}$ $G_{2,6}$
$G_{3,1}$ $R_{3,2}$ $G_{3,3}$ $R_{3,4}$ $G_{3,5}$ $R_{3,6}$
$B_{4,1}$ $G_{4,2}$ $B_{4,3}$ $G_{4,4}$ $B_{4,5}$ $G_{4,6}$
$G_{5,1}$ $R_{5,2}$ $G_{5,3}$ $R_{5,4}$ $G_{5,5}$ $R_{5,6}$
$B_{6,1}$ $G_{6,2}$ $B_{6,3}$ $G_{6,4}$ $B_{6,5}$ $G_{6,6}$

Fig. 2: Calculating all component values for pixel (3,4)

***Algorithm**:*

1. Read the raw image
2. Iterate for each pixel in the image
3. Classify the pixel on the basis of odd/even row and columns
   Odd row and column: Top Green
   Even row and column: Bottom Green
   Odd row, even column: Red
   Even row, odd column: Blue
4. Based on what color sensor it is classified into, keep the inherent color as the corresponding color component of the output and calculate local average of neighboring pixels of the other two color components.
5. Each classified pixel will have a fixed method of calculation for its components.

III. Results

As is observed,

- ➢ The original gray scale cat image is now RGB
- ➢ The components of each pixel is calculated from its neighbors since the original gray scale image did not have colors.

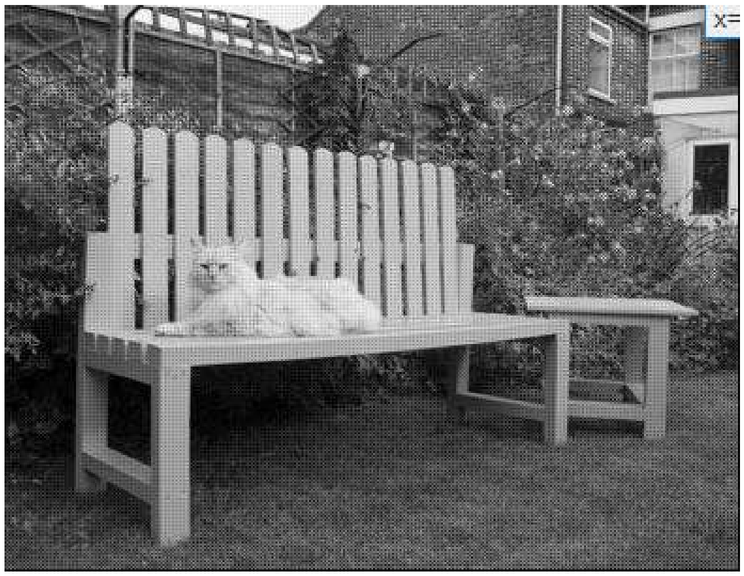


Figure 3: Grayscale Cat Image

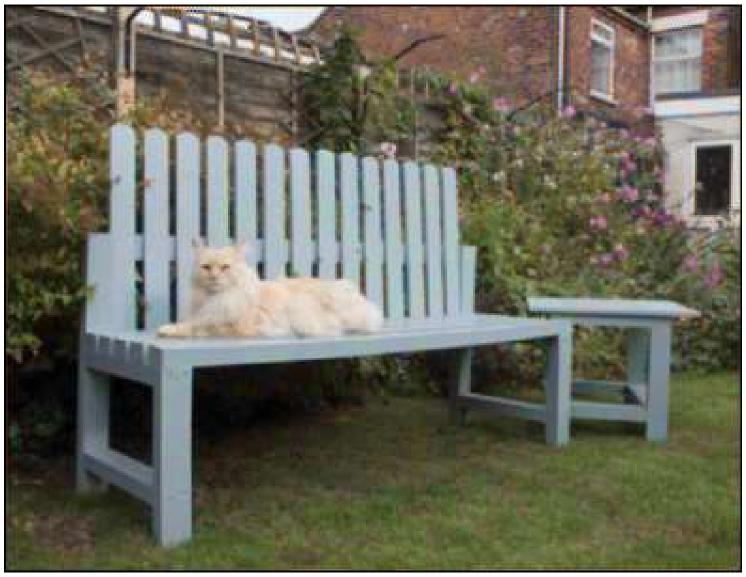


Figure 4: Demosaiced Cat Image via Bilinear Interpolatioin

IV. Discussion/Interpretation of results

Advantages of Bilinear Demosaicing:

1. Less computation
2. Good for non sophisticated images

The **disadvantages** of this method are that you tend to get artefacts in the output image. **These artefacts are caused by oversampling, especially at the edges.**

Pictures of a few artefacts are given below:

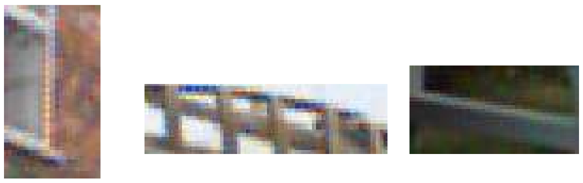


Figure 5 : Artefacts seen in the output image

This can be overcome by

- preventing oversampling by imposing relevant criterion
- taking into consideration more than just the immediate surrounding neighbours, as we shall see in MHC Demosaicing.

B. Malvar-He-Cutler (MHC) Demosaicing

I. Abstract and Motivation

What it does: Performs bilinear interpolation and adds a correction factor which considers pixels beyond immediately surrounding neighbours.

Why use it?

- Bilinear Interpolation helps demosaicing with lesser computation but performs poorly at edges. MHC overcomes this defect.

II. Approach

Theory and Procedure:

After performing bilinear demosaicing, add a correction term to the formulae involved. An example of this correction term for green in R(3,4) is given below:

$$\hat{G}(i,j) = \hat{G}^{bl}(i,j) + \alpha\Delta_R(i,j)$$

$$\Delta_R(i,j) = R(i,j) - \frac{1}{4}\big(R(i-2,j) + R(i+2,j) + R(i,j-2) + R(i,j+2)\big)$$

The end result is a filter that can be convoluted over the 5x5 image for green component at red pixels:

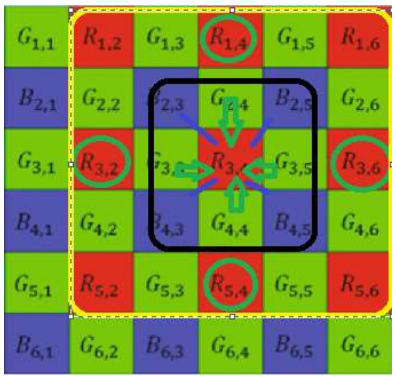


Fig.6: MHC demosaicing for R(3,4)

For different cases, after applying this formula, the following filters are obtained:

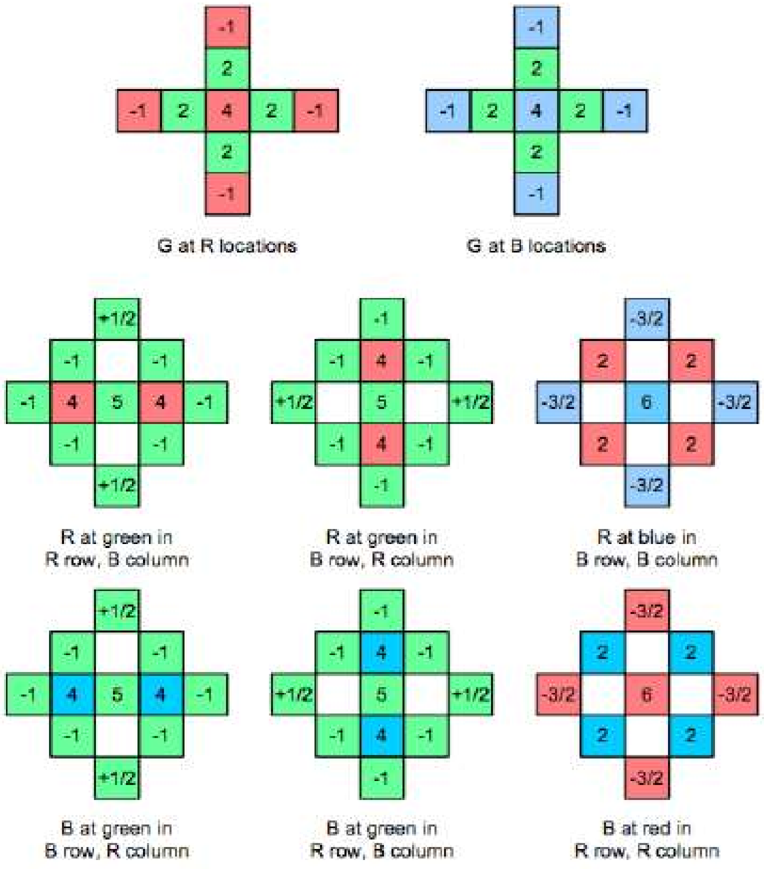

G at R locations
G at B locations
R at green in R row, B column
R at green in B row, R column
R at blue in B row, B column
B at green in B row, R column
B at green in R row, B column
B at red in R row, R column

Fig.7: MHC filter coefficients

Boundary extension:

Since this filter draws considerable from beyond the boundary for boundary pixels, boundary extension is carried out in the manner shown below:

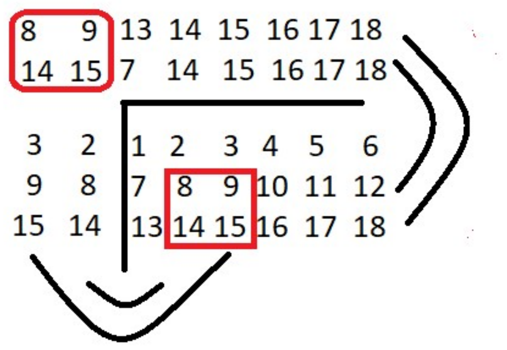
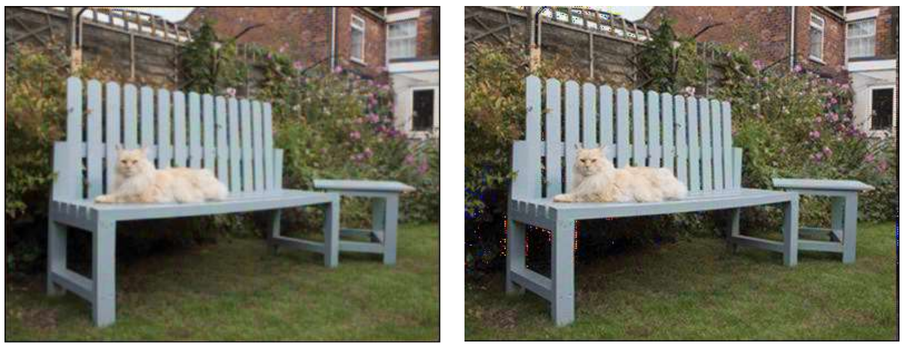


Fig. 8: Boundary Extension

***Algorithm**:*

1. Read input image
2. Iterate for each pixel in the image
3. Classify the pixel on the basis of odd/even row and columns
   Odd row and column: Top Green
   Even row and column: Bottom Green
   Odd row, even column: Red
   Even row, odd column: Blue
4. Based on what color sensor it is classified into, convolve the 5x5 neighborhood of that pixel with the corresponding filter coefficients.

III. **Results:**



a b

Fig. 8: Image of cat in (a) bilinear demosaicing (b) MHC demosaicing

IV. Discussion:

The following elicits the differences observed while implementing the two methods:

Bilinear	MHC
computationally less intensive	computationally harder on a relative basis but not on a penalty basis
blurred edges	better performance on edges
artefacts observed	comparatively lesser artefacts observed
no malfunctioned pixel components	a few pixels will have overflowing components which will cause a specific colour to dominate in that pixel
rapid change of colors causes artefacts resulting in non continuity of colors	helps maintain the consistency of colors in different regions especially when the change rapidly

(c) Histogram Manipulation:

I. Abstract and Motivation:

What it does: Sometimes, the brightness or the contrast of an image results in lack of perceptibility. This is corrected by histogram manipulation wherein the pixel values are redistributed in equitable measures to obtain an equilibrium in terms of image visibility.

Why it is done:

- increased visibility
- reveals concealed features

All in all, contrast enhancement.

II. Approach:

This is done using two methods:

1. Transfer function based histogram equalization: (method 1 or A)
   The frequency of each pixel is input into a transfer function wherein its probability of occurrence is considered before the function equalizes the image.

2. Cumulative Probability based histogram equalization: (method 2 or B)
   Counts the number of grayscale levels and assign equal number of pixels from the total number of pixels to each level before redistributing equalized pixels to the image.

   Algorithm for method 1:
   1. Read the input image file
   2. Iterate for every pixel
   3. Count the number of times each pixel intensity has occurred
   4. Calculate probability of each pixel intensity
   5. Calculate cumulative probability
   6. Map each probability value to 255

   Algorithm for method 2:
   1. Read the input image file
   2. Iterate for every pixel
   3. Find out the number of intensity levels
   4. Find out how many pixels go into each level for an equitable distribution
   5. Assign pixels in the original image to the corresponding level of intensity

III. Results:

Method 1: Transfer function based histogram equalization

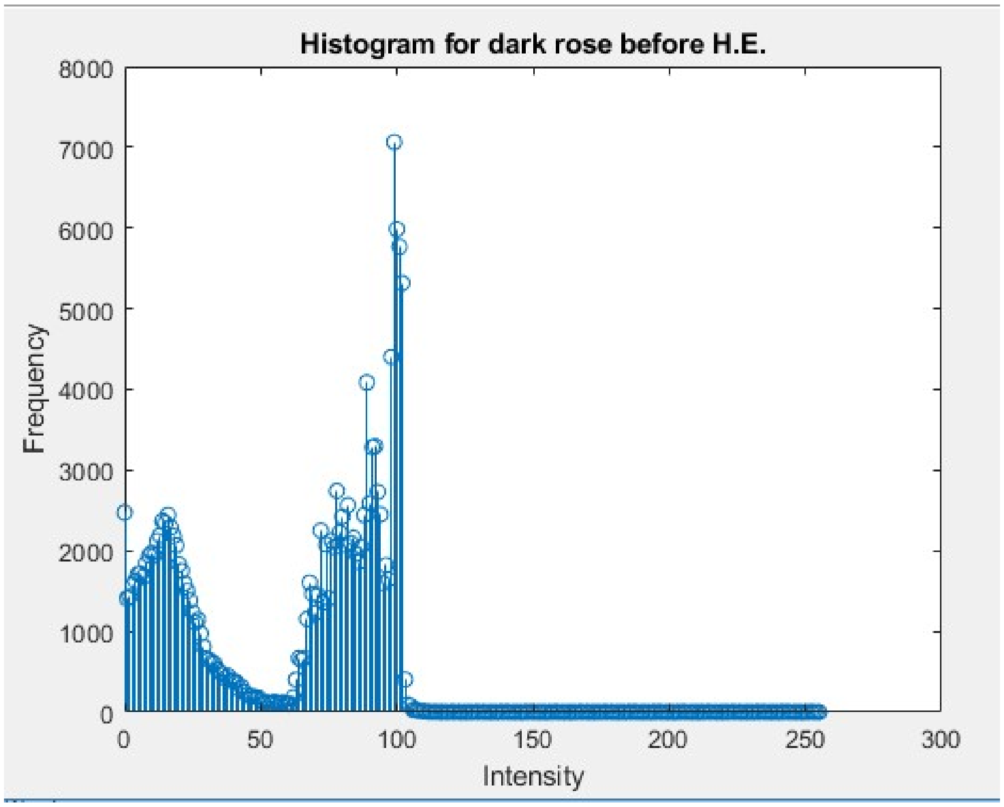

Histogram for dark rose before H.E.
Frequency
Intensity
0
1000
2000
3000
4000
5000
6000
7000
8000
0
50
100
150
200
250
300

Fig.9 : (c) (1)

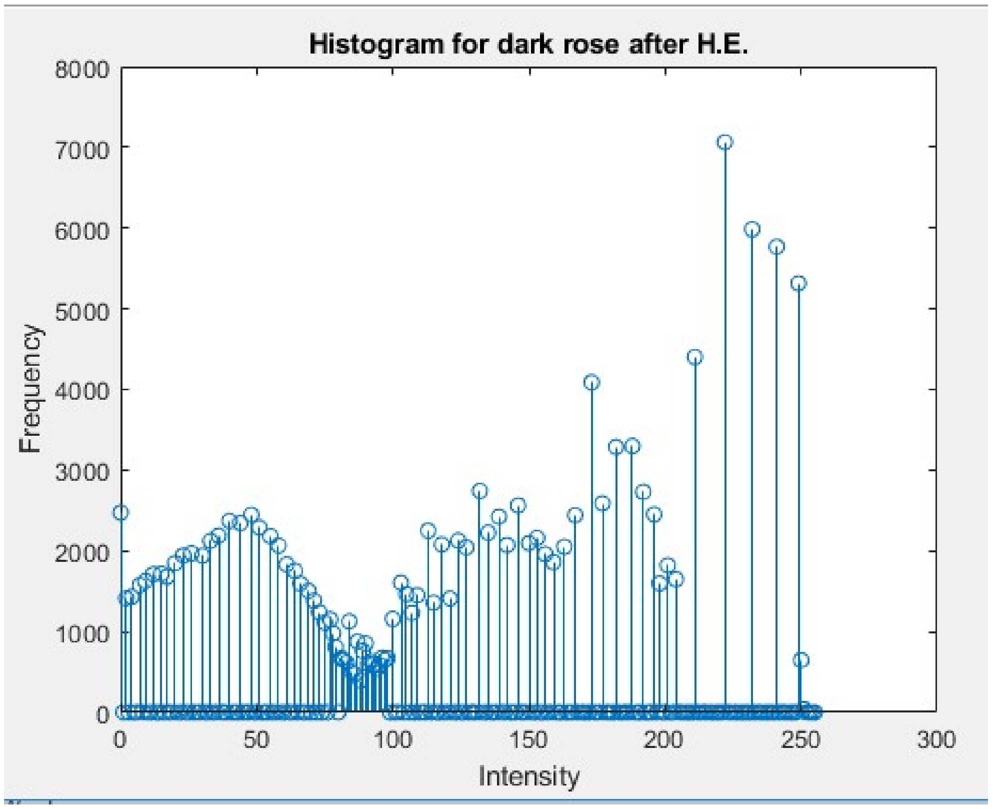

Histogram for dark rose after H.E.
Frequency
Intensity
0
1000
2000
3000
4000
5000
6000
7000
8000
0
50
100
150
200
250
300

Fig. 10: (c) (2)

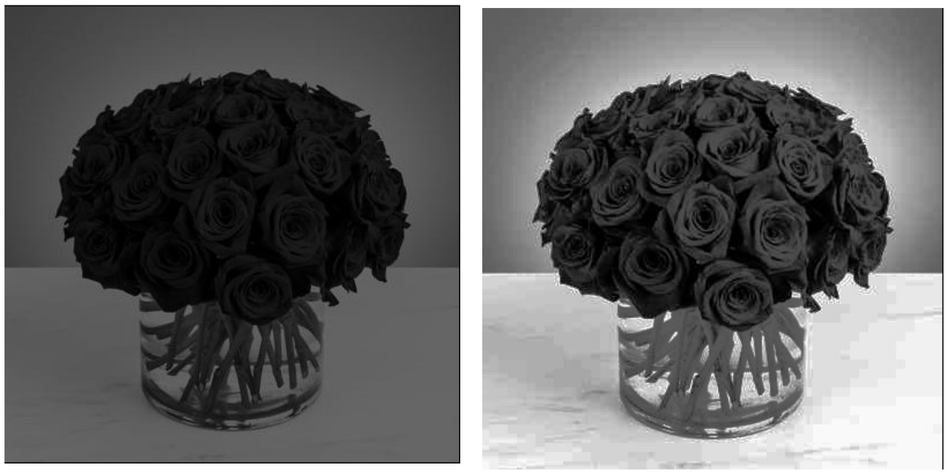


Fig. 11: (c) (2) : Dark rose before and after method 1

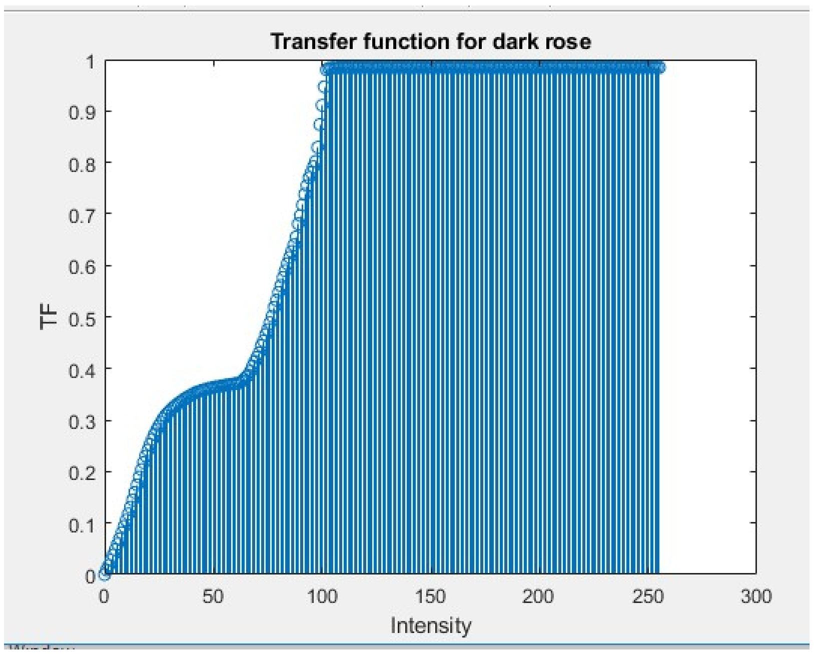


Fig. 12 (c) (2)

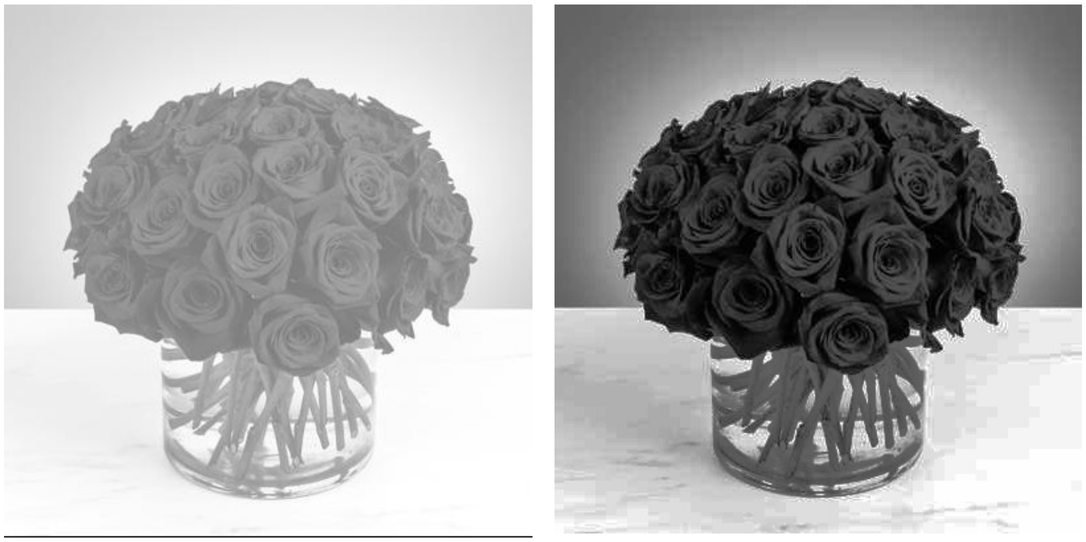
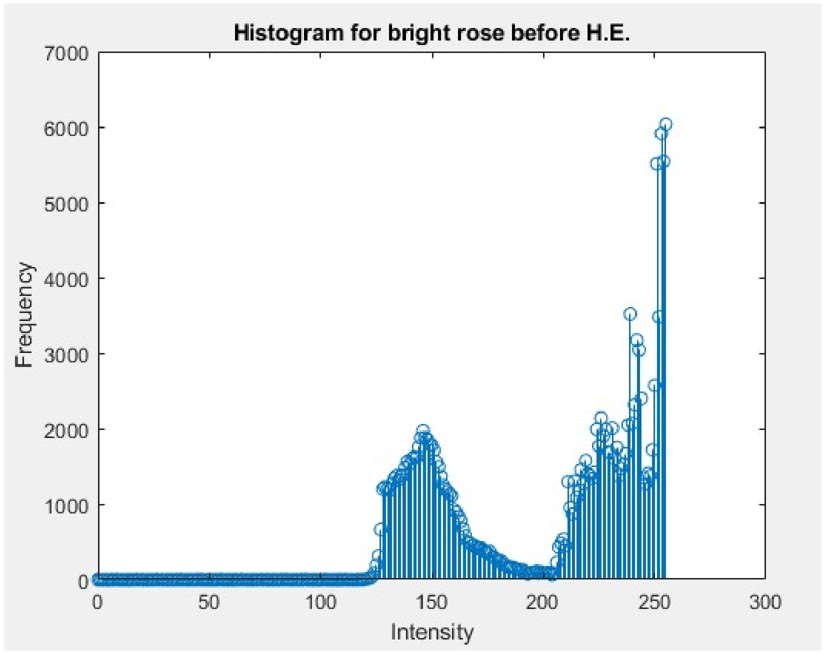
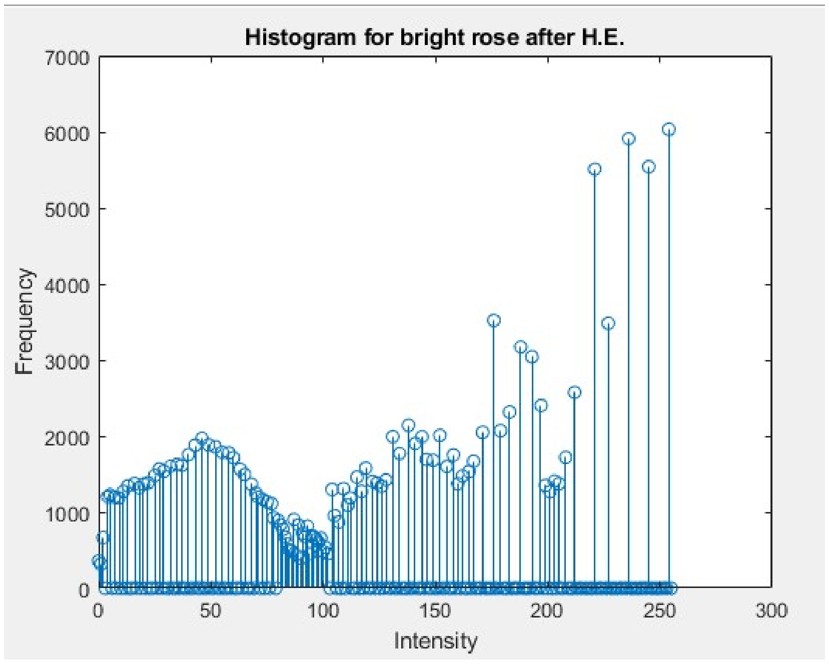


Fig. 13: (c) (2) : Bright rose before and after method 1





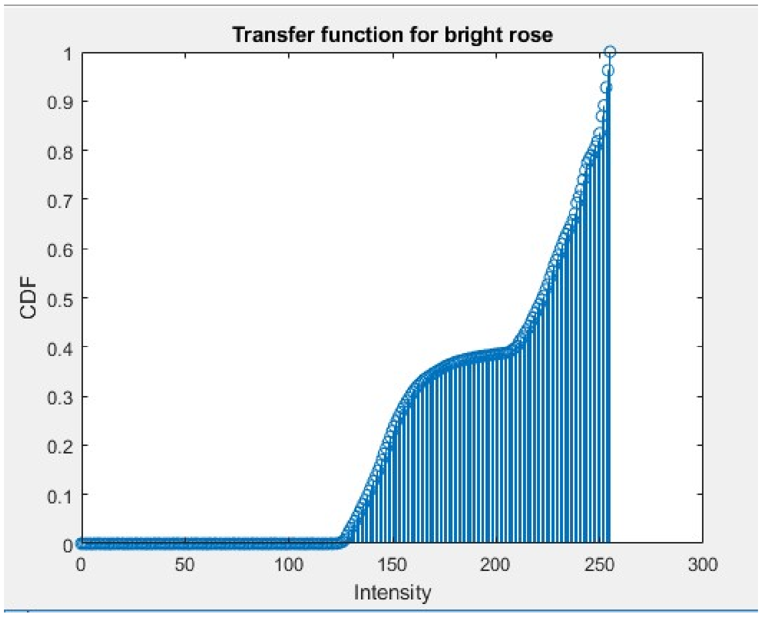


Fig. 14: Relevant paraphernalia for bright rose (c) (1) and (2)

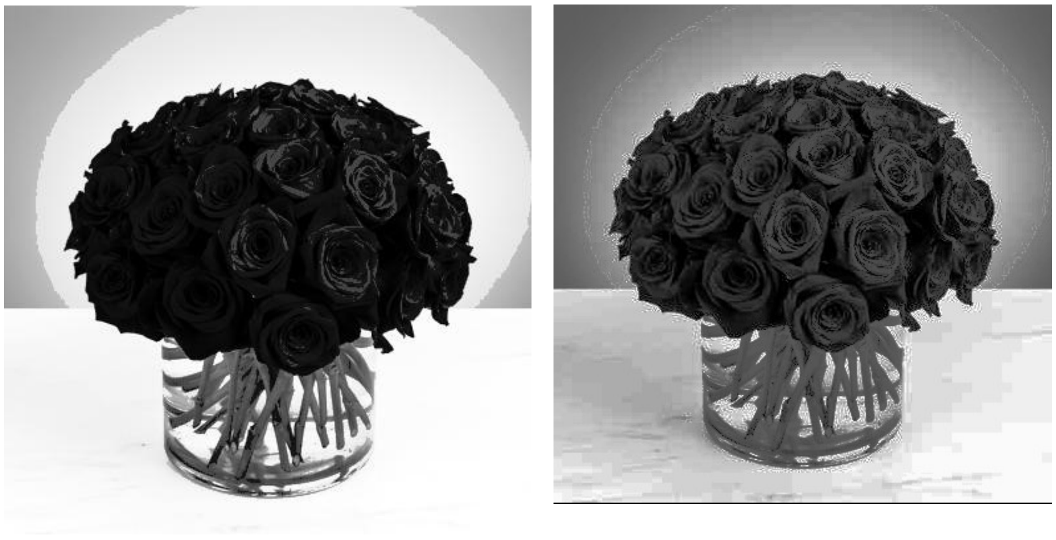


Fig. 15: Mixed rose before and after method 1

Method 2: Cumulative Probability based histogram equalization:

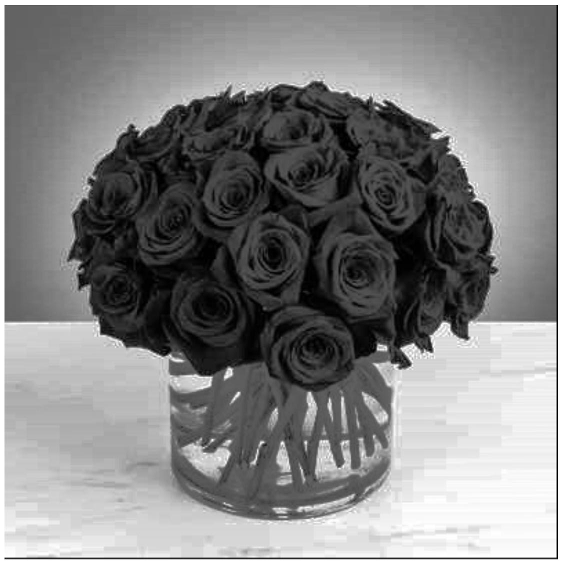


Fig. 16: Dark rose after method 2

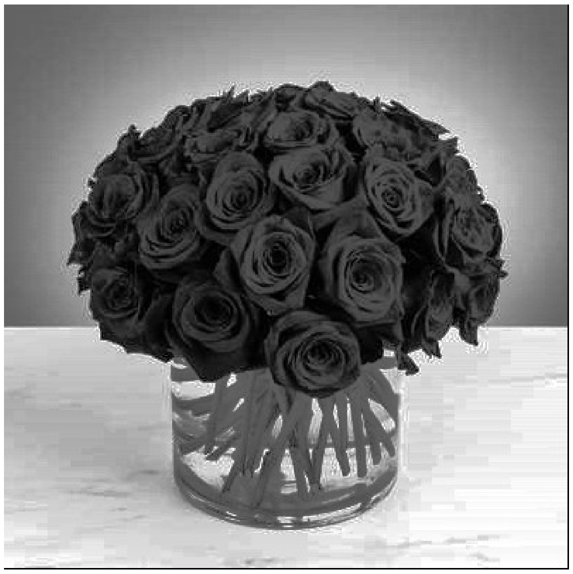
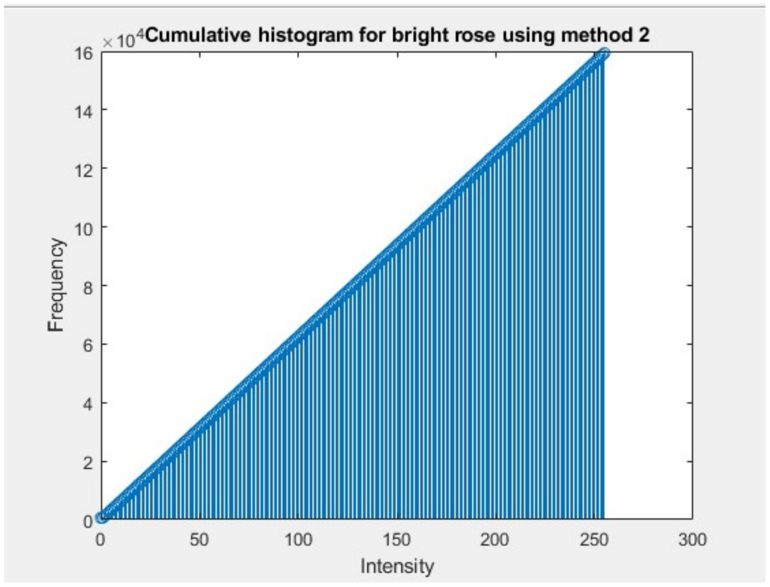
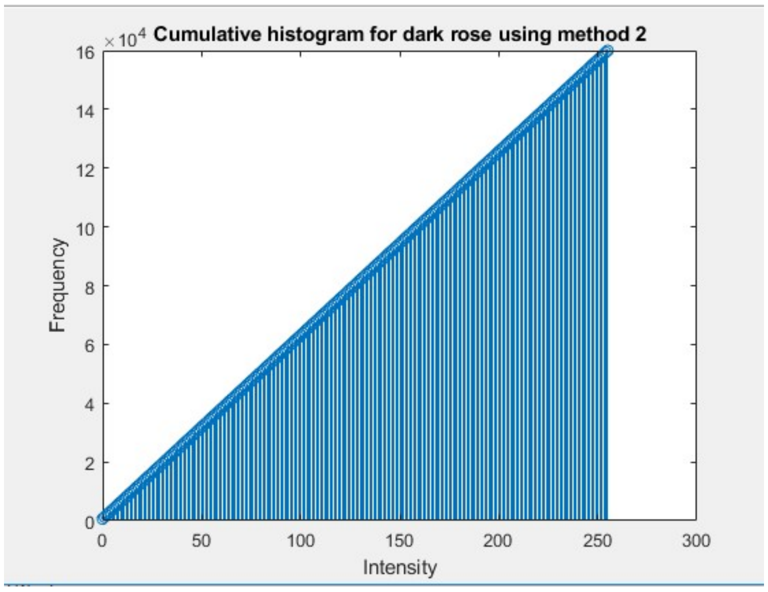


Fig. 17: Bright rose after method 2





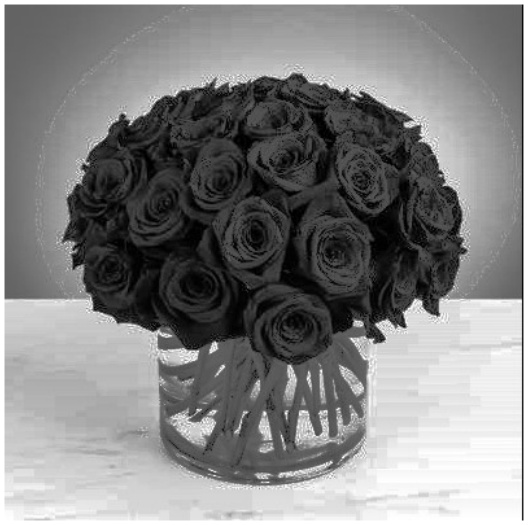


Fig. 19: Mixed rose after method 2

IV. Discussion:

- Method 2 is better than method 1 in terms of smoothness of color transitions and contrast enhancement, but marginally so.
- For example, in method 2, the background lighting is more pleasant and smoother and the rose petals stand out stronger.
- However, method 2 ignores the legitimacy of the value that the pixel holds by have equal sized bins. Therefore, some extra brightness persists which is not inherent to the original image.

- rose_mix does not give a similar result as method 1 and 2 equalize the contrast but do not fine tune it.
- However, this can be dealt with by increasing the number of bins in method 2 or semi quantizing the intensity levels in method 1.

Problem 2: Image Denoising

A. Gray Level Image

Abstract and Motivation:

Noise is an unnecessary signal. It distorts the image features. Denoising is a method which employs performing a certain operation using filters on the image pixels in order to purge it of its noise.

Here, we mainly deal with two types of noise: Salt and pepper noise (impulse noise) and uniform noise which is gaussian distributed.

Approach

We employ five types of filters to deal with the different types of noise:

1. Low Pass Filter (Uniform) : Removes uniform noise

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

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

where (k,l) is the neighboring pixel location within the window size of w1 x w2 centered around (i,j). I is the noisy image.

Algorithm:

1. Read input image file, perform boundary extension and iterate through each pixel
2. For each pixel, calculate its neighbourhood based on the size of the window and convolve it with the kernel to generate a weight
3. Multiply it with the intensity and take the aggregate
4. Divide by the sum of weights

2. Low Pass Filter (Gauassian): Removes uniform/gaussian noise

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

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

where sigma is the standard deviation of the Gaussian distribution.

Algorithm:

1. Read input image file, perform boundary extension and iterate through each pixel
2. For each pixel, calculate its neighbourhood based on the size of the window and calculate the weights as per formula and convolution.
3. Multiply it with the intensity and take the aggregate
4. Divide by the sum of weights

3. Median filter: Removes impulse noise (in next part of this problem)

Algorithm:

1. Read input image file, perform boundary extension and iterate through each pixel
2. Sort the neighbouring pixels into order based upon their pixel values
3. Replace the center pixel values with the median value from the list

4. Bilateral filter

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

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

Algorithm:

1. Read input image file, perform boundary extension and iterate through each pixel
2. For each pixel, calculate its neighbourhood based on the size of the window and calculate the weights as per formula and convolution
3. Multiply it with the intensity and take the aggregate
4. Divide by the sum of weights

5. Non Local mean filter

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

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

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

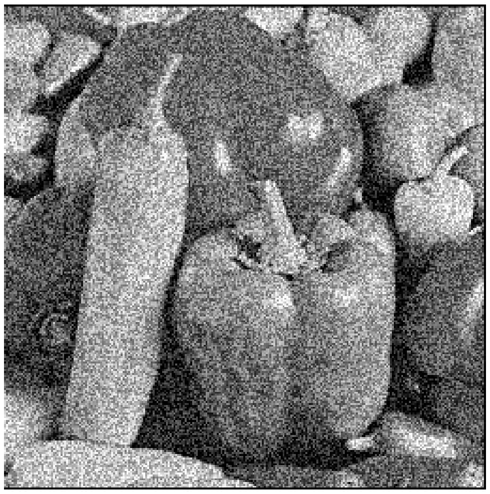
and

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

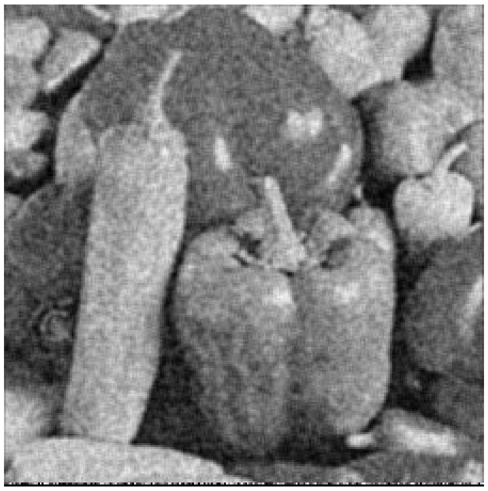
Algorithm:

1. Read input image file, perform boundary extension and iterate through each pixel
2. For each pixel, calculate its neighbourhood based on the size of the window N' M'
3. For each neighbourhood pixel check its neighbourhood in a smaller sized pixel window of n1 and n2 for similarity
4. This is done by implementing the formulae above
5. Multiply the weight with the intensity and take the aggregate
6. Divide by the sum of weights

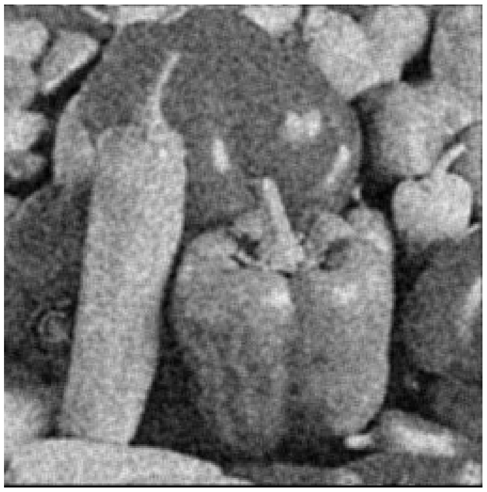
Results:

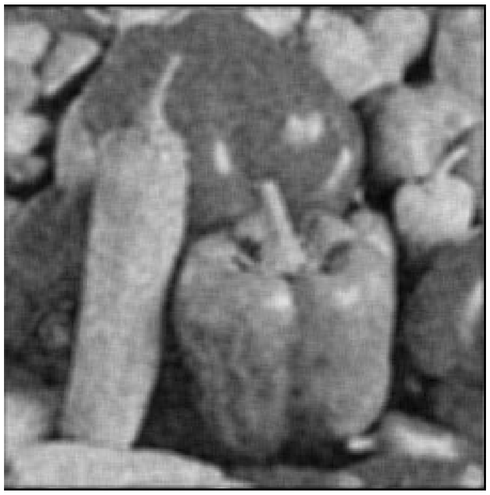


Noisy image

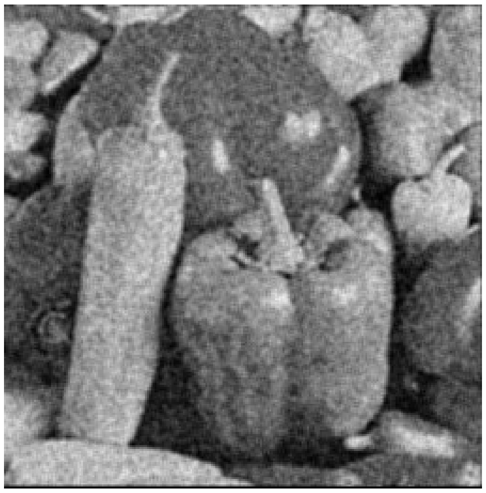


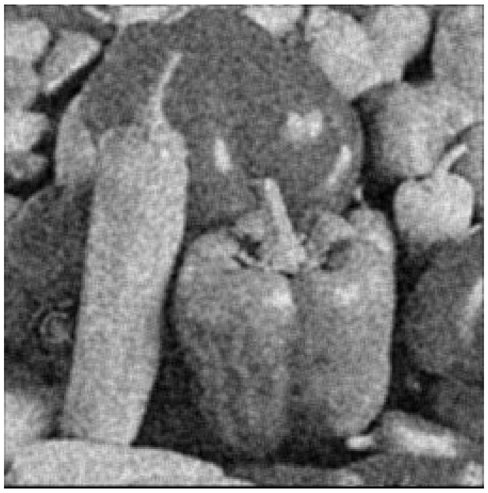
Uniform filter with window size 3(above)

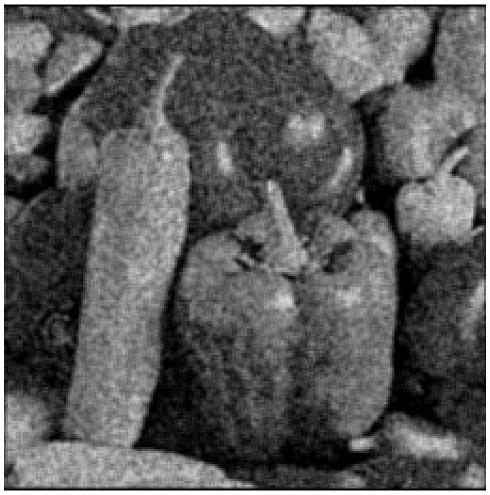




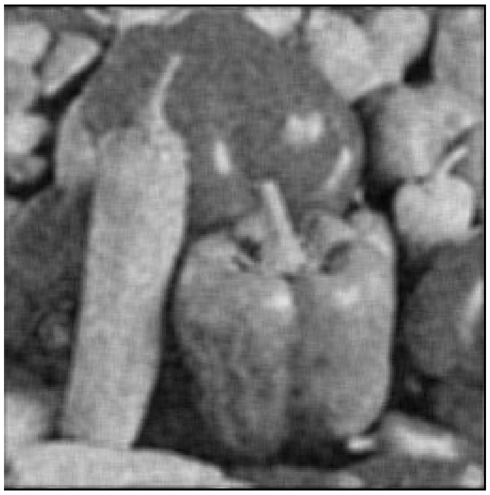
Gaussian filter with window sizes 3,5 and 7

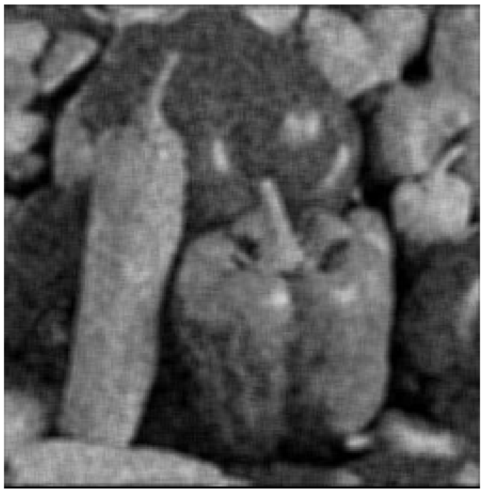




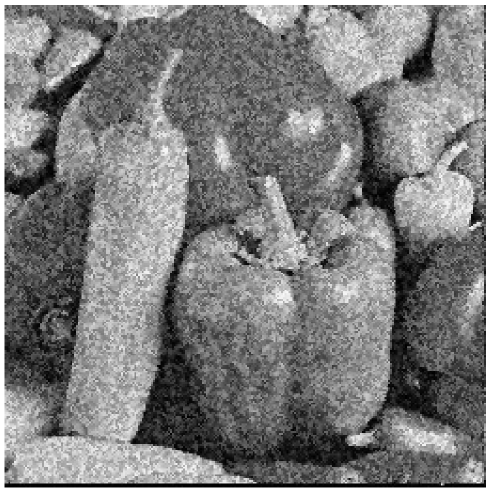


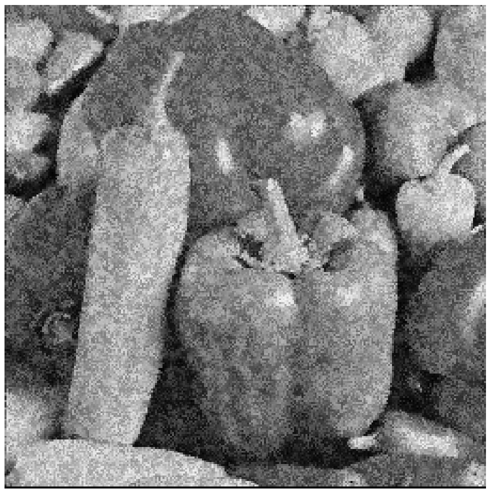
Gaussian filter with window size 3 and increasing variance

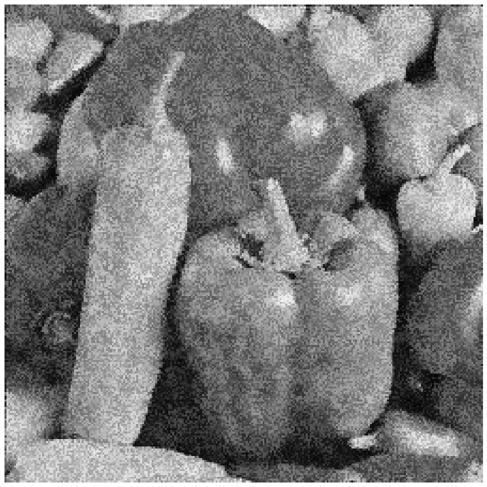




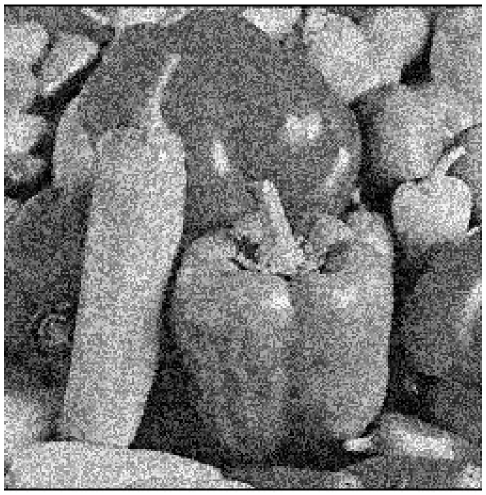
Gaussian filter with window size 5 and increasing variance

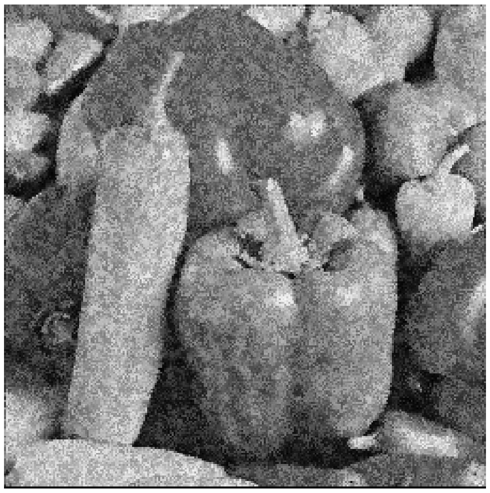




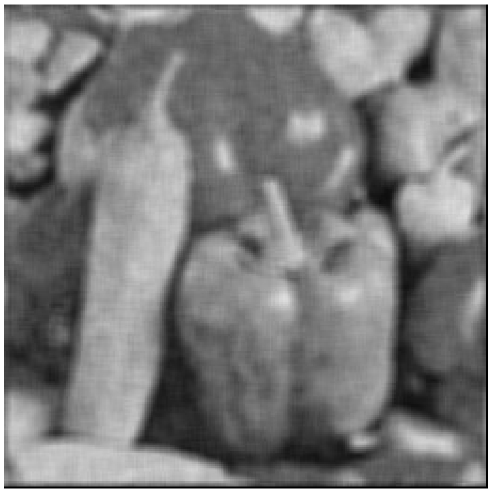


Bilateral filter with window sizes 3, 5 and 7





Bilateral filter with window size 5 and increasing sigmac and sigmas



Non local mean filter with larger window size 7 and smaller window size of 3

Non local mean filter(Ideal output)

<u>Discussion</u>

- For uniform filter, we see that although part of the noise is removed, there exists blurriness.
- For Gaussian filter, we observe that with increasing variance, the image gets clearer but the noise persist.
- For Gaussian filter, we observe that with increasing window size, the noise decreases drastically but the clarity reduces.
- For bilateral filter, we see that the performance is better on the edges.
- For non local mean filter, we can see that there is blurriness but the noise is removed. An idea NLM output would have oversmooth features.

B. Color Image

<u>Abstract and Motivation</u>

The image has impulse and uniform noise in each channel. The objective is to remove this noise using the right combination of filters.

<u>Approach</u>

The cascade of filters is a two step process:

It can be used in the following combinations:

Combination 1:

1. Gaussian filter: Removes uniform noise
2. Median filter: Removes impulse noise

Combination 2:

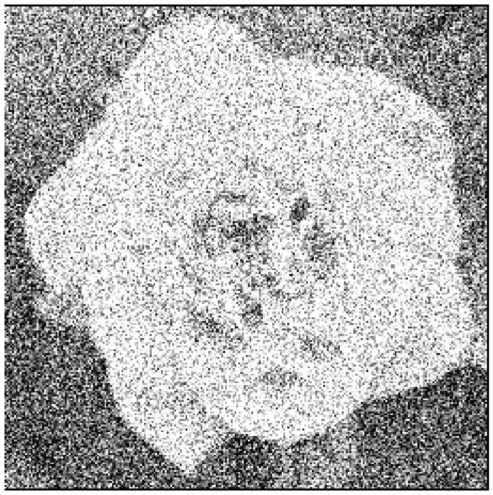
1. Uniform filter
2. Median filter

Switching the order: Combination 3:

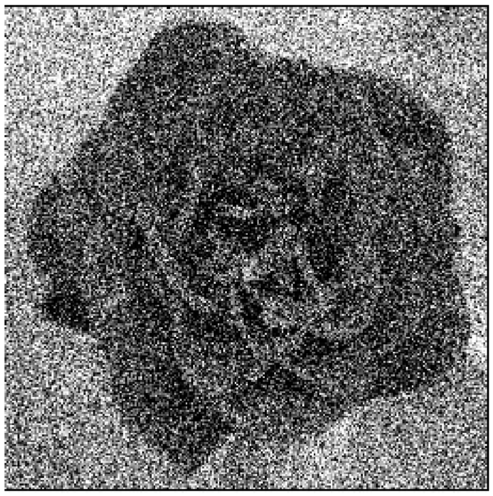
1. Median filter
2. Uniform filter

Switching the order: Combination 4:

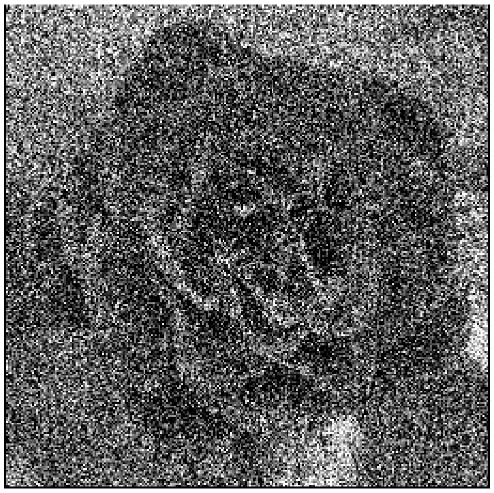
1. Median filter
2. Gaussian filter



Noise in R channel

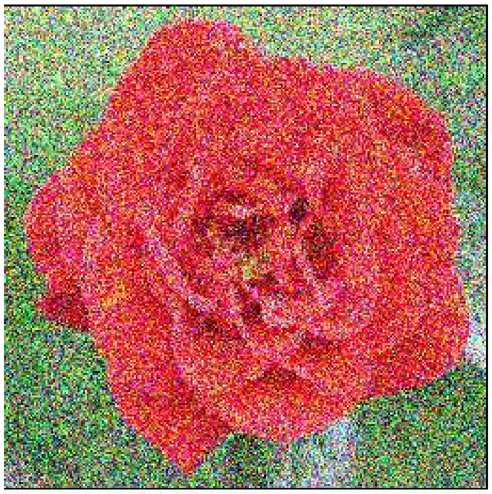


Noise in G channel

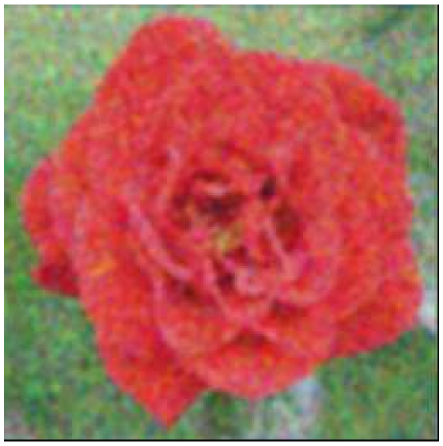


Noise in B channel

<u>Results</u>



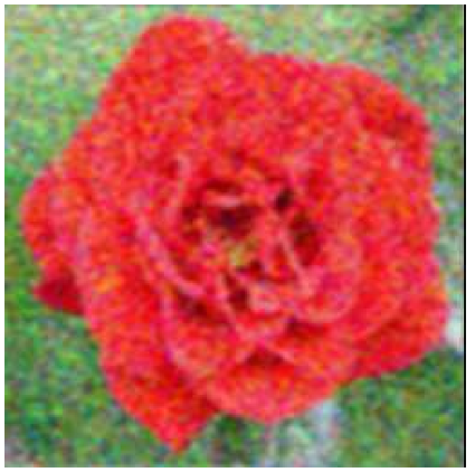
Original Noisy Rose Image (above)



Combination 1: After cascading filters and running channels through them -> filtered output

Combination 2: After cascading filters and running channels through them -> filtered output

Switching Cascaded filters: Combination 3: Median + Uniform



Switching Cascaded filters: Combination 4: Median + Gaussian

Discussion (Answers to questions in HW)

- Here we can see that blurriness is observed since the Gaussian filter and the mean filters do not work well on edges.
- This is the least computationally intensive compromisable combination
- The three channels were treated separately through the cascade of filters for better performance since each component of the colors behaves differently with impulse noise
- The sequence of the filters: the performance can be improved if the gaussian filter is added after the median filter as the median filter can remove the outliers and the gaussian filter can use an equitable gaussian weight to smoothen the image. The result is shown above.
- An alternative would be to implement the bilateral filter instead of the gaussian filter since it works better at the edges.