HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

EE569-HW1

Problem 1: Part A: Image Demosaicing

• Abstract and Motivation:

In an analogy to human eye, it consists of different cords to infer different wavelength signals, which in turn we classify them as red, green, blue spectrum. In digital cameras, we use a Bayer filter sensor which is a grid of different photo-sensitive elements. Since a position is occupied by only one sensor, hence we are able to collect fraction of information of the incident light. Bayer filter array is designed to occupy 50% green, 25% blue and rest 25% red channel sensors. We complete the information of each position in grid with the help of neighboring pixels. This process of filling up the missing information with the help of neighboring pixels and obtaining RGB values at each position is known as Image Demosaicing. The mathematics of how to use the neighboring pixels give rise to Bilinear, MHC and various other demosaicing methods. We implement Bilinear Demosaicing on the input 8-bit grayscale image and obtain 24-bit RGB image.

• Approaches and Procedures:

Before applying Bilinear Demosaicing, we pad our image by adding a reflected row and column on each side of the image. This is known as boundary extension and helps us dealing with the boundary pixels.
We now implement Bilinear Demosaicing by leveraging row and column index of padded image. We define different filters for the respective cases. Let i = row_idx % 2; j = col_idx % 2, then

Filter_diagonal = $\begin{pmatrix} 1/4 & 0 & 1/4 \\ 0 & 0 & 0 \\ 1/4 & 0 & 1/4 \end{pmatrix}$ **Filter_cross** = $\begin{pmatrix} 0 & 1/4 & 0 \\ 1/4 & 0 & 1/4 \\ 0 & 1/4 & 0 \end{pmatrix}$

Filter_vertical = $\begin{pmatrix} 0 & 1/2 & 0 \\ 0 & 0 & 0 \\ 0 & 1/2 & 0 \end{pmatrix}$ **Filter_horizontal** = $\begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 0 & 0 \end{pmatrix}$

Sr. No.	(i,j)	Place holder	R	G	B
1	(0,1)	Blue	Filter_diagonal	Filter_cross	-
2	(1,0)	Red	-	Filter_cross	Filter_diagonal
3	(0,0)	Green	Filter_vertical	-	Filter_horizontal
4	(1,1)	Green	Filter_horizontal	-	Filter_vertical

We convolve our padded image with the respective filters according to the row and column index. We thus obtain a 3-channel 24-bit array matrix.

- **Experimental Results:**

  Here is the output after implementing bilinear demosaicing.

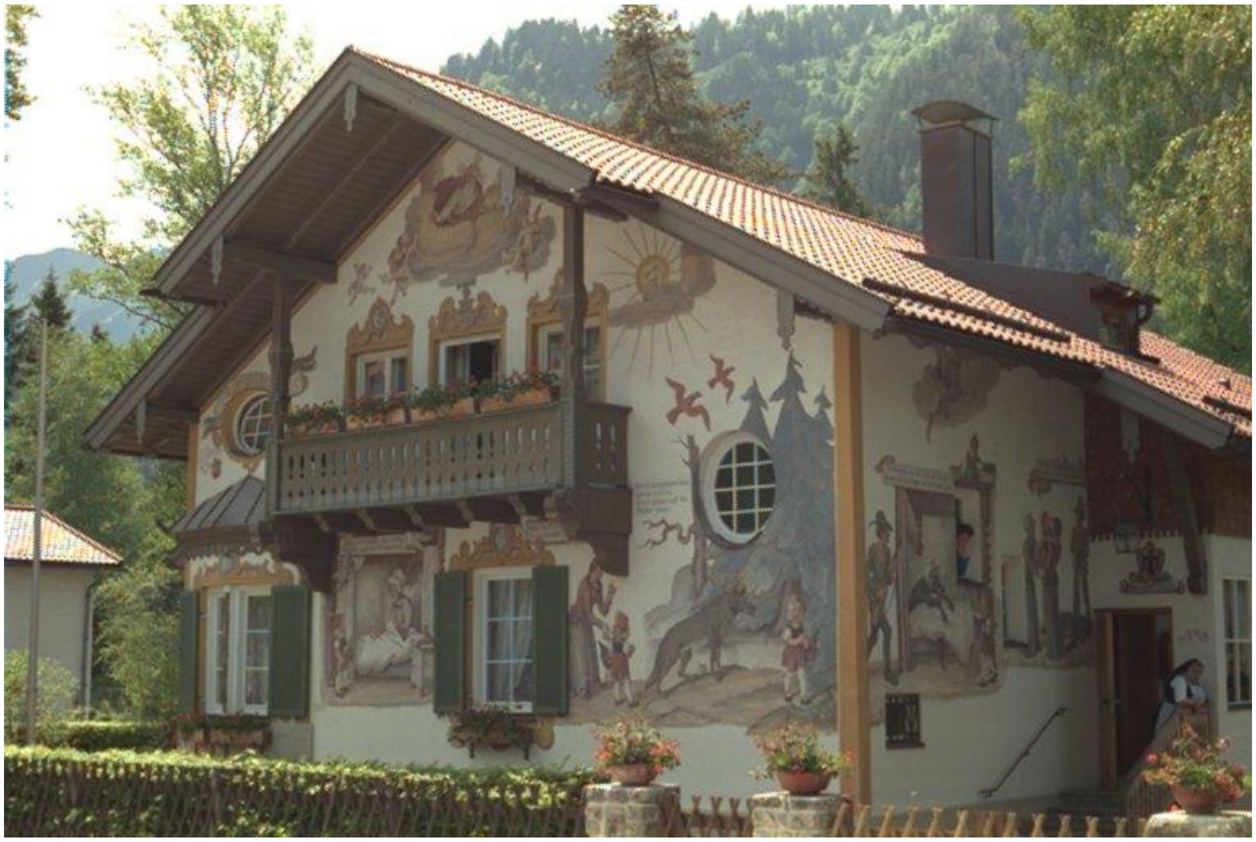


Figure 1 Bilinear Demosaicing

HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

- **Discussion:**

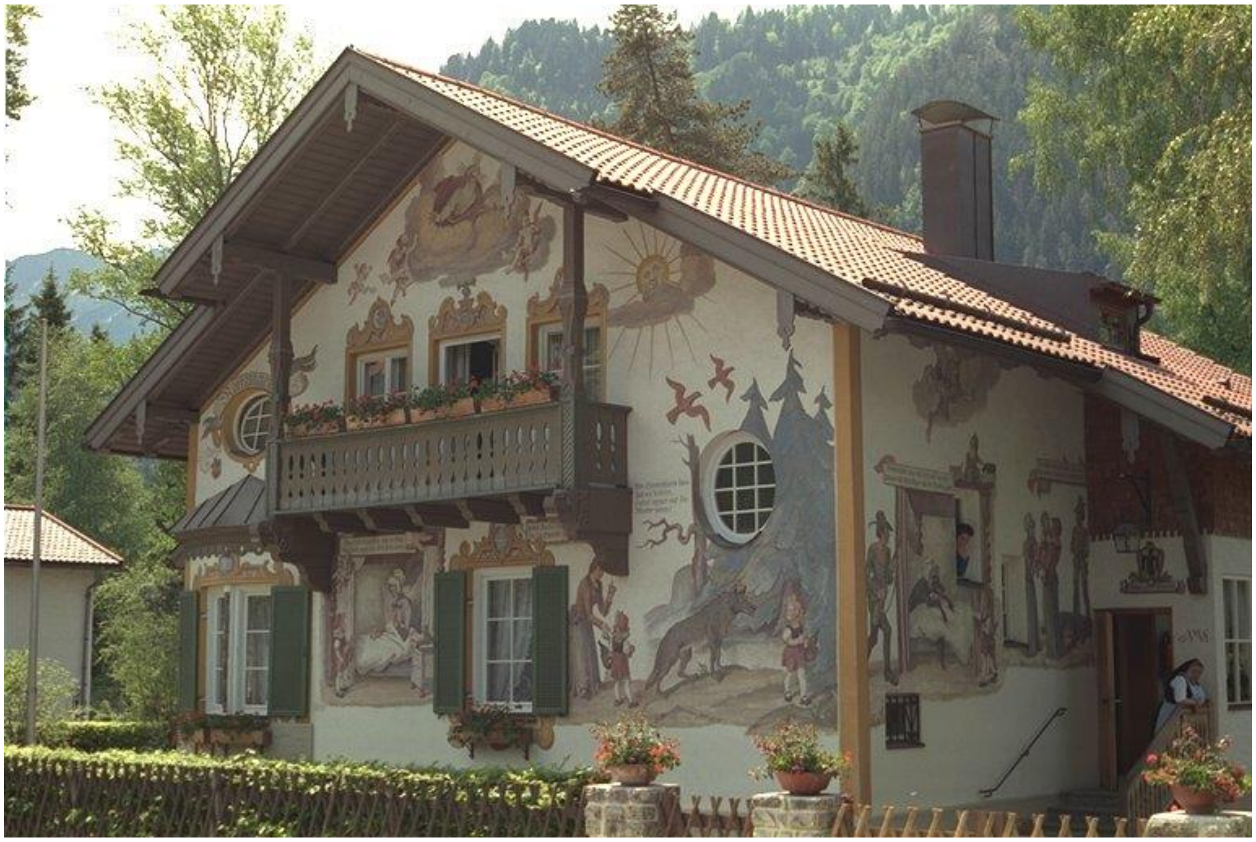


Figure 2 House_ori (advanced demosaicing)

Our implementation of Bilinear demosaicing seems to have done fairly well as our obtained output looks quite similar to House_ori which is obtained from advanced demosaicing. This in turn means, that our simple implementation comes at very low cost and does fairly well. Easy to implement and easy for human interpretation.

Although, I do observe some artifacts:

- ➔ The portion which are less dense in similar object type in the neighborhood seem to be distorted/noisy. For example, the pattern on the roof, the area of leaves (tree on left side in image).
- ➔ Also, seems like our output image is blurred a bit.

Ideas for improvement:

- ➔ Instead of one entire grid for the complete image, how about multiple grids of smaller sizes.
- ➔ Also, all these grids need not follow the same 2:1:1 (G:R:B) place holder ratio.
- ➔ Also, instead of R,G,B sensors, how about CMY combination which theoretically proves to have improved light absorption property.

HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

Problem 1: Part B: Histogram Manipulation

- **Abstract and Motivation:**
  We often encounter low-light/bad-light while clicking a photo. The features in resultant image are not quite visible. Say, we are scanning a document and the room lamp is obstructed by our body which in turn drops shadow on the document. Technically speaking, this leads to a histogram spread across a small range of values. Hence, the contrast is low as the difference in maximum and minimum intensity value is less. This motivates us to do histogram manipulation which will create contrast and in turn the image is more clearly visible.

- **Approaches and Procedures:**
  We implement 2 methods.
  Method A: Transfer-function based histogram equalization method.
  In this method, we compute the histogram of the input image. We then divide the histogram values by total pixels to give PDF/PMF. A cumulative addition of these values (0-255) will give us the CDF. Now we generate mapping rule x → cdf*255 which thus scales the intensity values over the entire range of 0-255. The relationship between the old intensity values and new intensity values is know as the transfer function.

  Method B: cumulative probability-based histogram equalization method.
  In this method, we decide the number of bins (pixel intensity values) over which we want to spread our histogram. We then decide on the approximate number of pixels each bin should contain. Once we know the bin size, we start distributing the pixels amongst these bins. This will lead to a uniform distribution of pixels values over the entire range.

- **Experimental results:**

Q1_b (1)

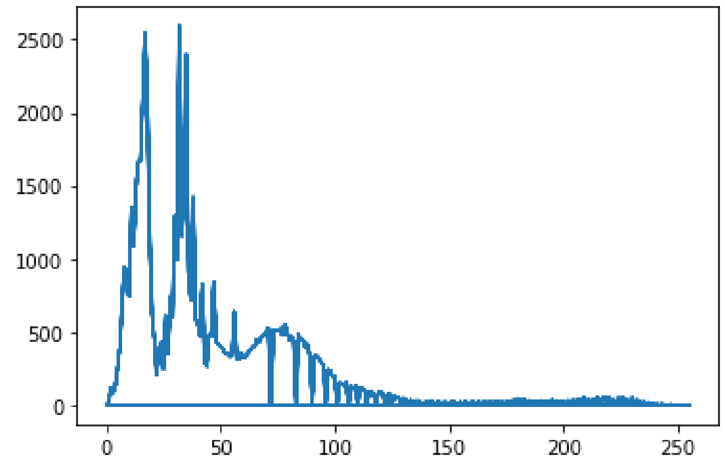
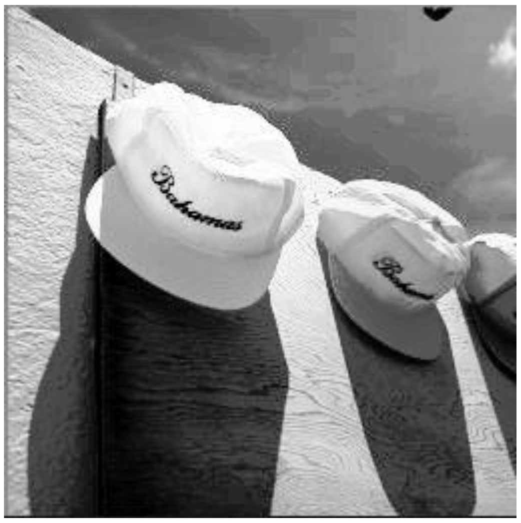


Figure 3 Histogram of hat_original

Q1_b (2)



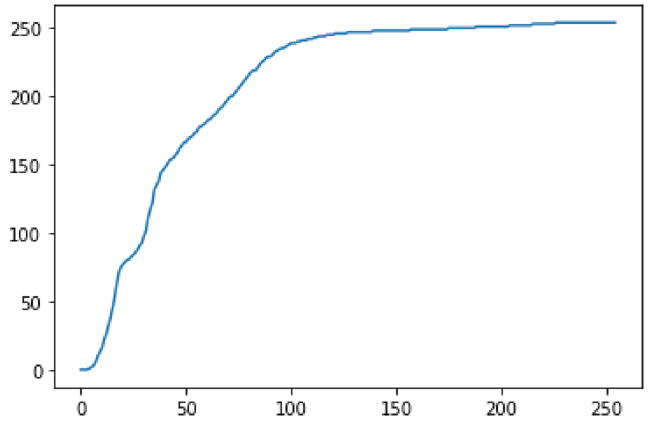


Figure 4 Method A, Enhanced Image

Figure 5 Method A, Transfer Function

HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

Q1_b (3)

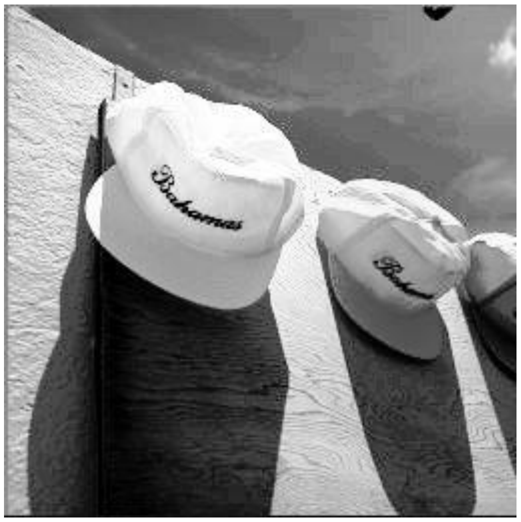


Figure 6 Method B Enhanced Image

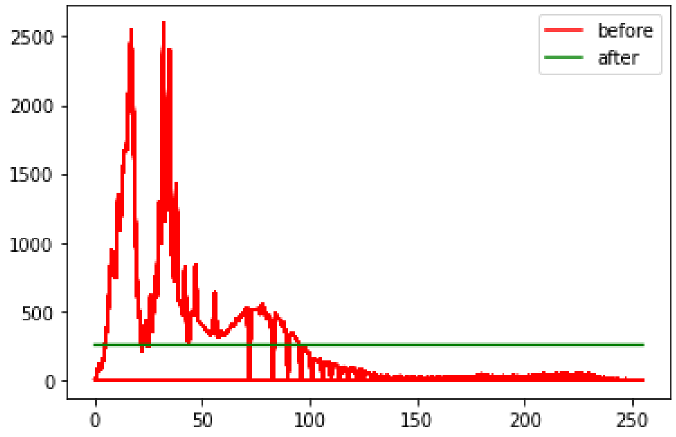


Figure 7 Before and after histogram

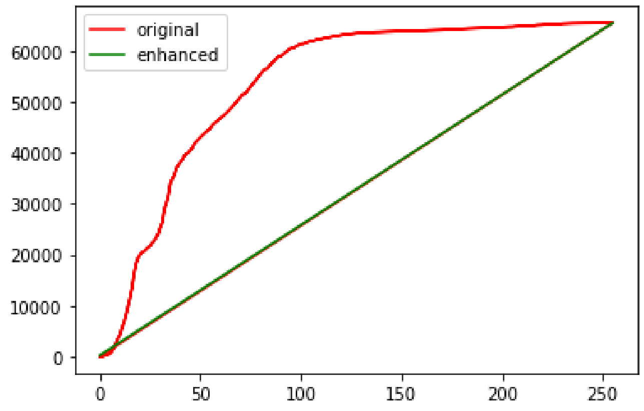


Figure 8 CDF (before and after)

- **Discussion:**

  To the naked eye, both enhanced images look quite similar. But when we check the histogram of both the images, there are gaps in values of grayscale for Method A. This is not the case in Method B. Method B tends to produce more smoother image compared to Method A, which in turn results into low gradient change at edges/boundaries of objects in image.

HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

Problem 1: Part C: CLAHE

- **Abstract and Motivation:**
  Light source has different quantity of incidence on the image. This results into certain areas brighter and certain areas darker. When we apply histogram equalization over the entire image, we tend to blur out the edges. This gives motivation to implement adaptive histogram equalization. In this we break the image into several grids and then apply histogram equalization over these grids with their respective local histograms. This helps in countering the effect of low light in certain specific part of the image. But in doing so, one problem arise which is there are high chances of amplification of noise. Here, CLAHE comes into picture which is a type of adaptive histogram equalization method. In CLAHE, before computing CDF and transfer function, we first clip the histogram value (hyperparameter). This limits to contrast amplification and hence noise amplification. One of the applications of this is haze removal.

- **Approaches and Procedures:**
  We repeat the previously discussed two histogram equalization methods in addition to CLAHE. The procedure for implementing all of these remain the same and is stated as below
  1) We convert the image to YUV color space, in order to work with only the luminance channel Y.
  2) We apply Method A, Method B, CLAHE over this Y channel and receive an enhanced Y channel, one by one.
  3) The enhanced Y channel is re-grouped with the original U,V channels. This results into our final output image which is enhanced image and does the work of haze removal in case of CLAHE.
  4) For the CLAHE method, we need to set the hyperparameter, cliplimit. I have chosen the values as ‘4’. Also, I have used the open-source code for implementing CLAHE using OpenCV function.

HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

- **Experimental results:**

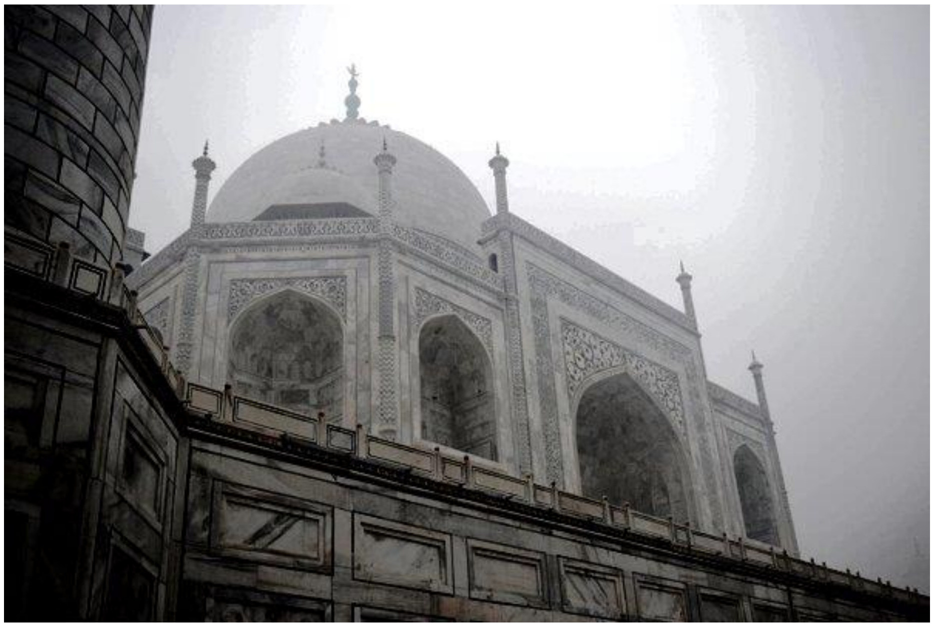


Figure 9 Haze removal_MethodA

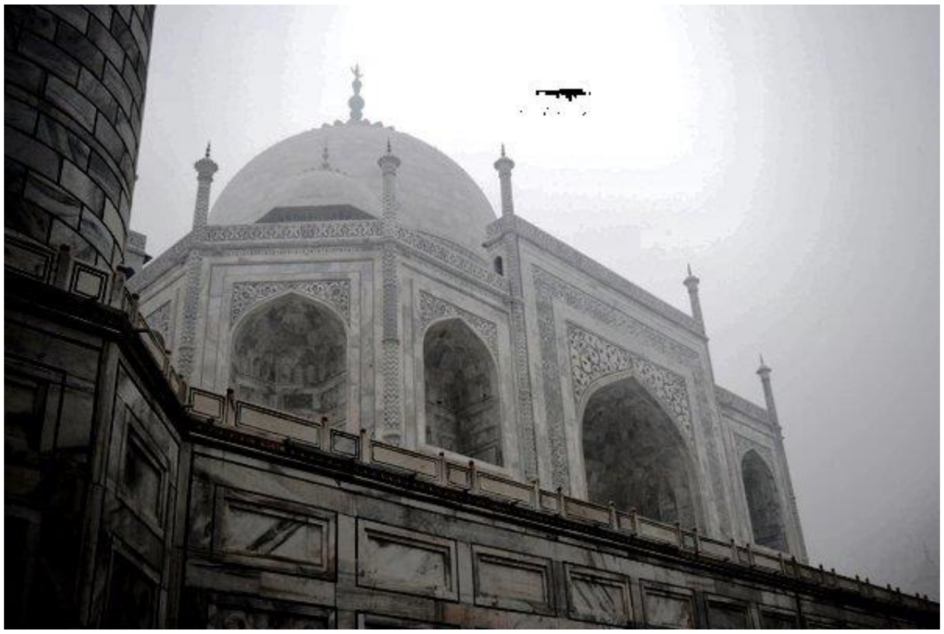


Figure 10 Haze_Removal_MethodB

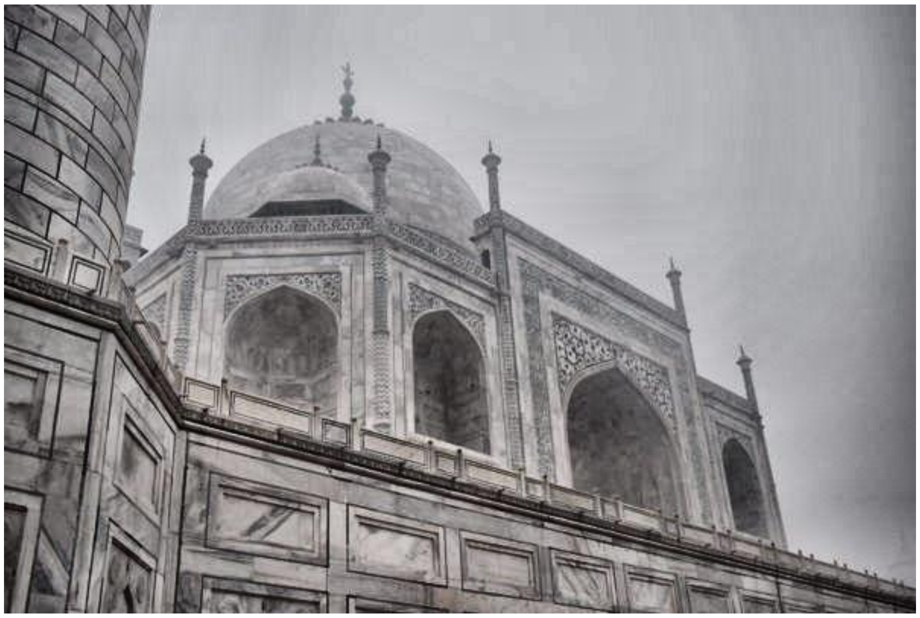


Figure 11 Haze_removal_CLAHE

- **Discussion:**

  CLAHE outperforms the other 2 methods by a big margin. Its quite evident that CLAHE has been able to do haze removal. At the same time, it has taken care that noise is not amplified.

  Method A and Mehod B have made certain portions in the image darker. Although the fog near the tomb of the Taj Mahal has been filtered out for these both methods. Method A and Method B seem to be subjected to some noise addition.

HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

Problem 2: Image Denoising

- **Abstract and Motivation:**

  In the Digital Image Processing pipeline, there are various stages which give rise to noise addition. For example, low quality sensors may add noise due to inefficient data collection. Another example would be due to ageing effect of appliances. One more source of noise could be ambient conditions. These all cannot be prevented. This gives the motivation of Image Denoising once the digital image is captured and handled by us. Noises have been categorized according to their probability density, namely gaussian, uniform, impulse etc. These noises can be attached through various filters such as Gaussian, Uniform, Median, Bilateral etc.

- **Approaches and Procedures:**

  Metric: Peak Signal to noise ratio (PSNR) is the metric used to determine the quality of denoised image. Higher the value of PSNR, better is the result.
  We also, do image padding in order to deal with the boundary pixels. For filter size of 3x3 we add 2 reflected rows and columns and for filter size of 5x5 we add 4 reflected rows and columns

1) Mean filter: $\frac{1}{9} * \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ takes average of the surrounding pixels. It has high effect of outliers which in turn can be impulse noise.
2) Gaussian filter: $\frac{1}{16} * \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$ gives more weightage for spatially closely located pixels. As size increases, image might get more blur.
3) Bilateral filter:

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

   As, it can be seen that, it has 2 components in the weights. One is the spatial closeness, closer the pixel, more is the weight assigned to that neighboring pixel. The second component checks for the similarity of the intensity value. It assigns more weight to that pixel whose intensity value is closer to the centered (or origin) pixel.
   Sigma_c & Sigma_s are the normalizing factors. They are user-defined. I have tried Sigma_c=Sigma_s=1; Sigma_c=Sigma_s=0.5 and Sigma_c=0.1 Sigma_s=0.05. Eventually I obtain best results with Sigma_c=Sigma_s=1.
4) NLM filter: As the name suggests, we do not take the mean of local neighborhood pixels rather, take mean of all pixels of image weighted by their similarity in intensity

values. We use the open-source code to implement this filter. We have 3 hyperparameters to set:

a) Degree of Smoothing =11
b) Search for window Size =17
c) Comparison Window Size = 7

Computationally, NLM filter is expensive.

Actually, we have 2 smoothing parameters. One each for Big window and neighboring window.

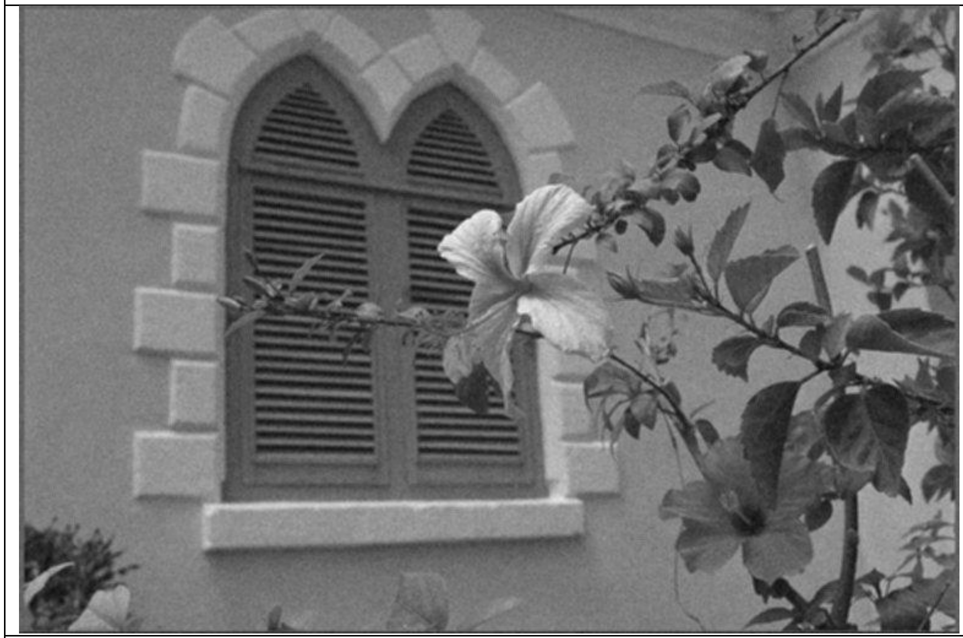
Decrease the Gaussian smoothing parameter of neighboring window in order to improve detection of weak features.

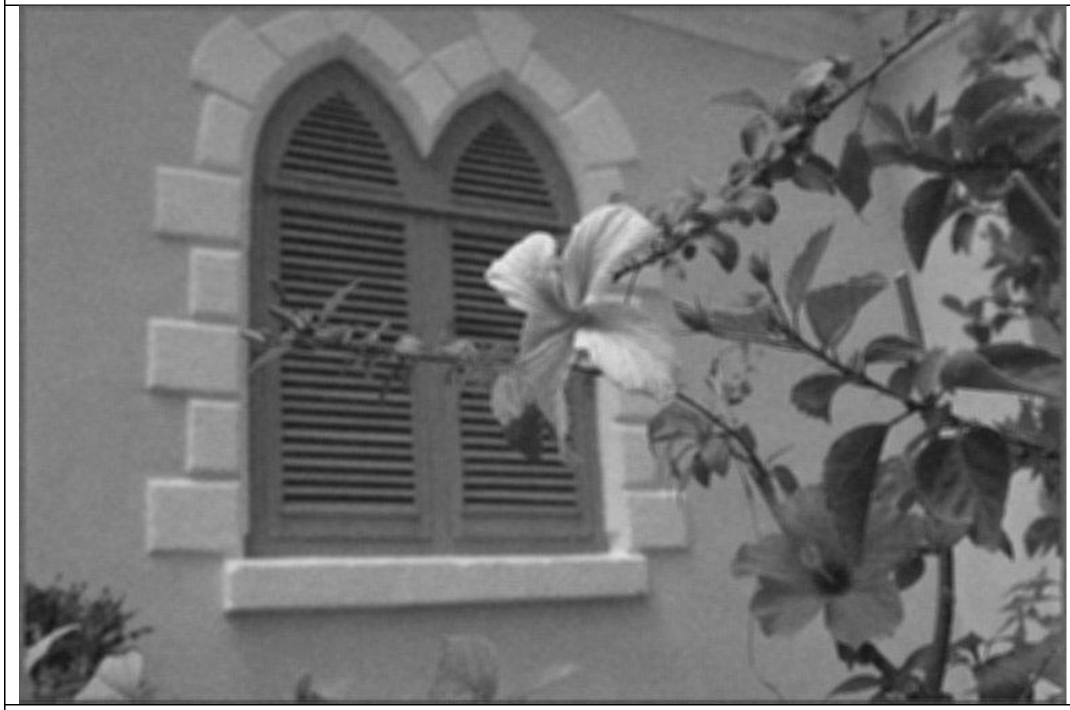
Choosing an optimum higher value of big window, will reduce the chances of being affected by any outliers.

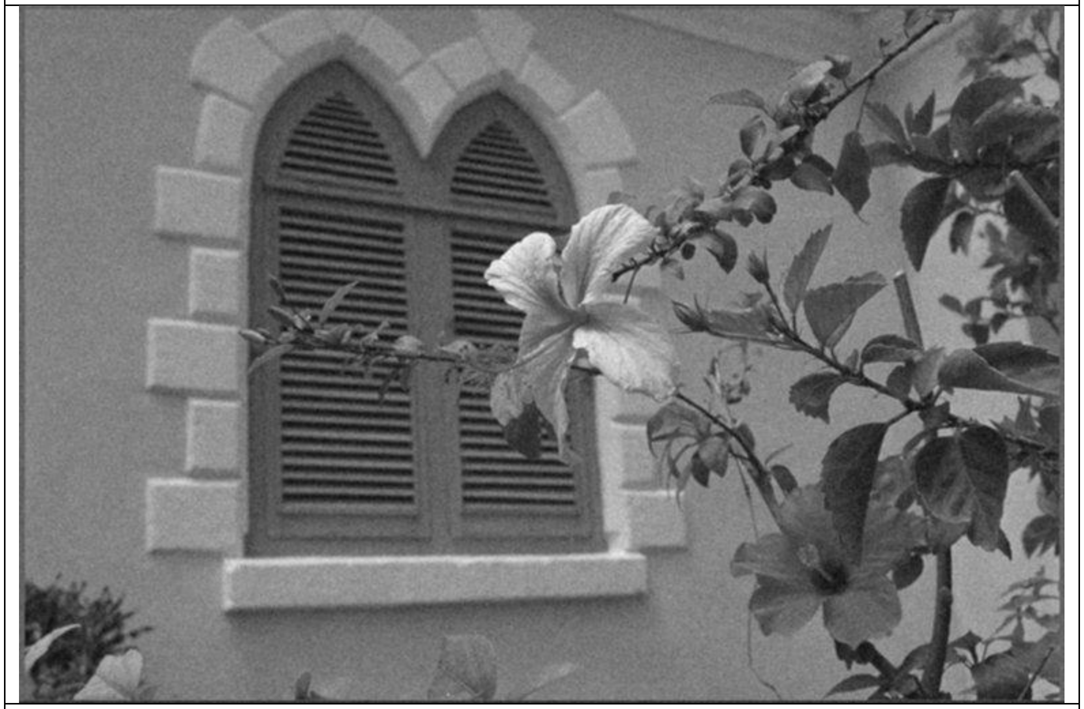
Closer the Euclidean distance between the neighboring window and Big Searching window (similarity) more the weight given to that neighboring window.

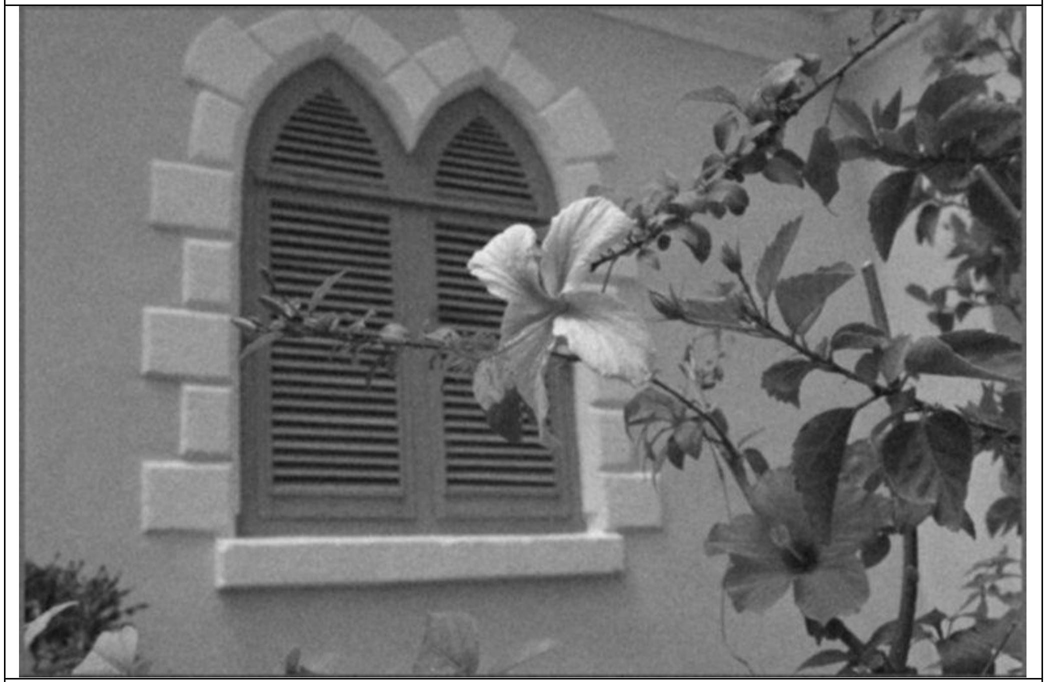
5) Mixed Noises: Color Image
We identify the noises by plotting the histogram of noise signal for each R, G, B channel.

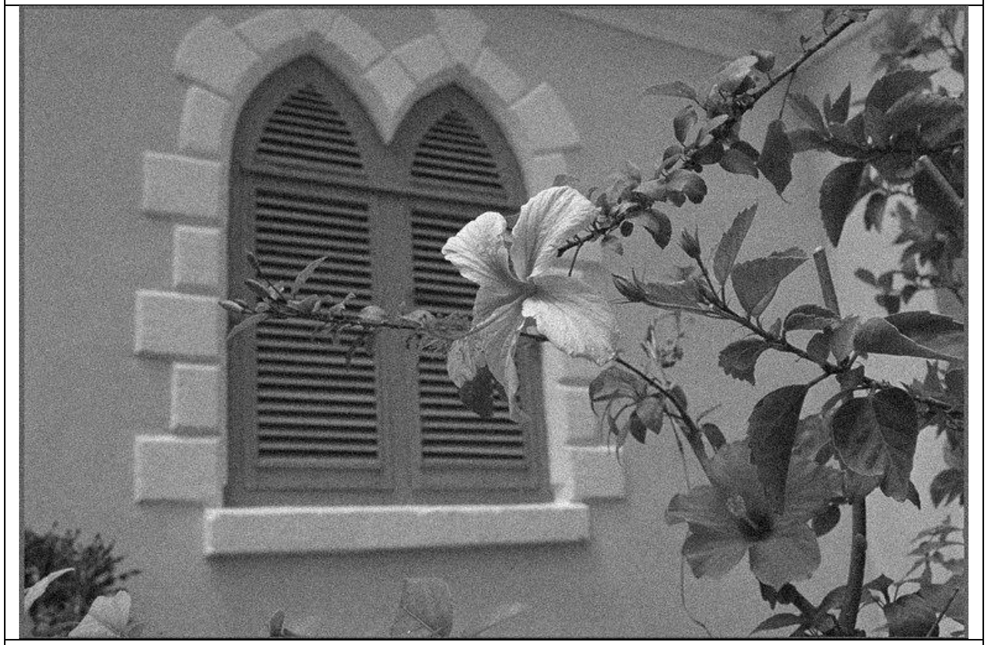
- **Experimental results:**

Mean 3x3

PSNR = 30.4358

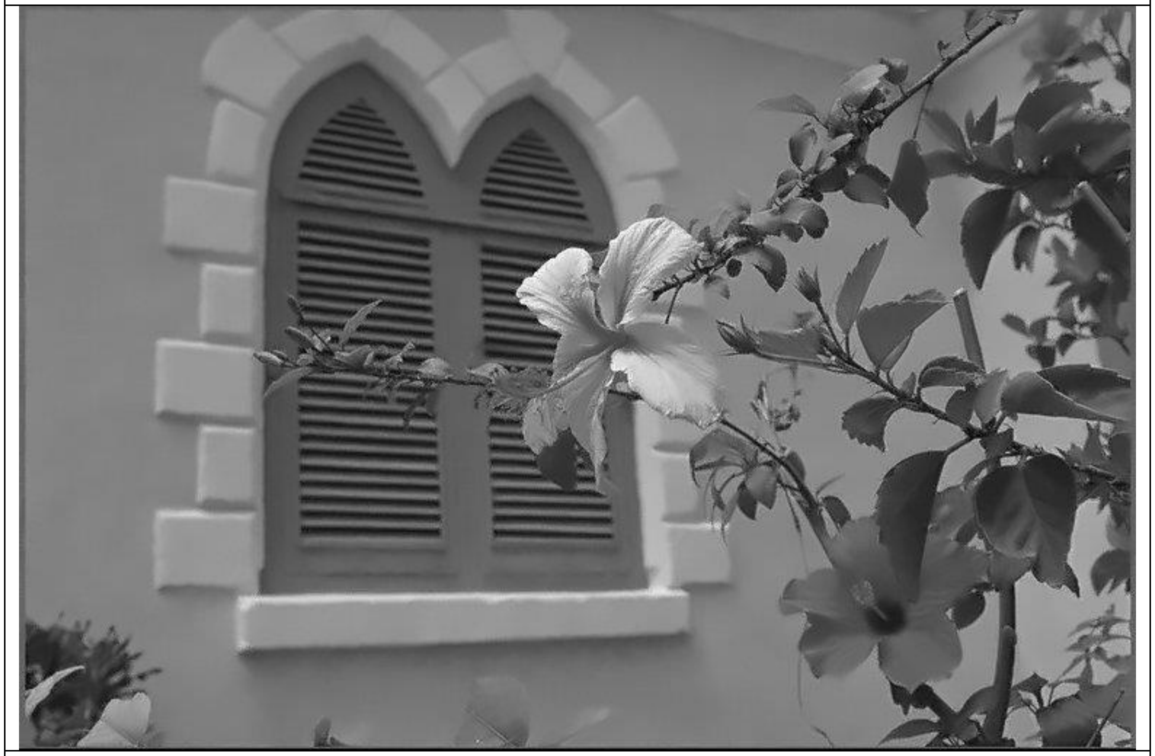
Mean 5x5

PSNR= 27.6176

Gaussian 3x3

PSNR= 31.9256

Gaussian 5x5

PSNR= 30.5181

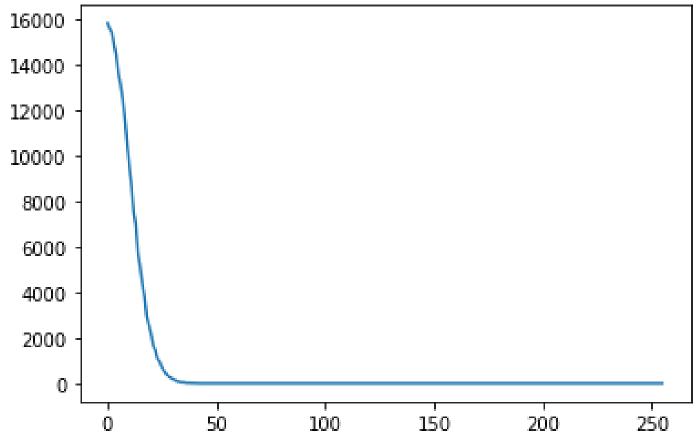
Bilateral

PSNR= 28.1384

HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

NLM

PSNR= 34.4741

- **Discussion:**

Q2_a_(1):



Type of noise: **Impulse** noise. From the plot of the histogram of noise signal

Noise = Noisy signal – Noise Free signal

Q2_a_(2): As, we can see that when we increase the kernel size from 3x3 to 5x5, PSNR value decreases. This is because, more smoothing happens and the image gets blur. This is more visible and evident in case of Uniform filter compared to Gaussian Filter.

Q2_b_(2): Sigma_c & Sigma_s are the normalizing factors. They are user-defined. Sigma_c is associated with spatial closeness. Increasing Sigma_c will lead in increase of weight.
Similarly, Sigma_s is associated with similarity (intensity) of pixels. If Sima_s is decreased, better chances of detecting weaker features.

Q2_b_(3): Bilateral filter performed better than only 5x5 mean filter. PSNR values justify the same.

Q2_c_(1): NLM filter: We have 3 hyperparameters to set:

d) Degree of Smoothing =11
e) Search for window Size =17
f) Comparison Window Size = 7

Computationally, NLM filter is expensive.

Actually, we have 2 smoothing parameters. One each for Big window and neighboring window.

Decrease the Gaussian smoothing parameter of neighboring window in order to improve detection of weak features.

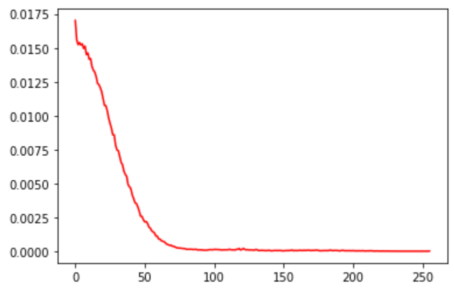
Choosing an optimum higher value of big window, will reduce the chances of being affected by any outliers.

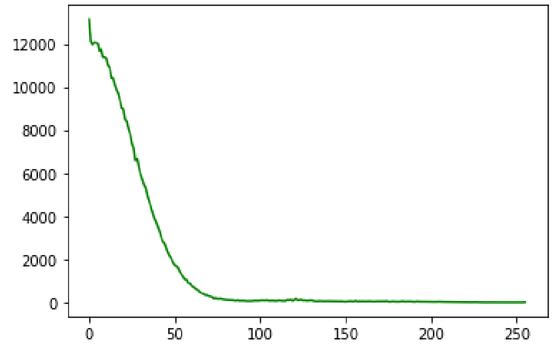
Closer the Euclidean distance between the neighboring window and Big Searching window (similarity) more the weight given to that neighboring window.

Q2_c_(2): NLM filter performed the best in comparison to all other filters. It obtained the highest PSNR value of 34.47. The resultant image seems to be noise-free to naked eye. Also, the edges are preserved.

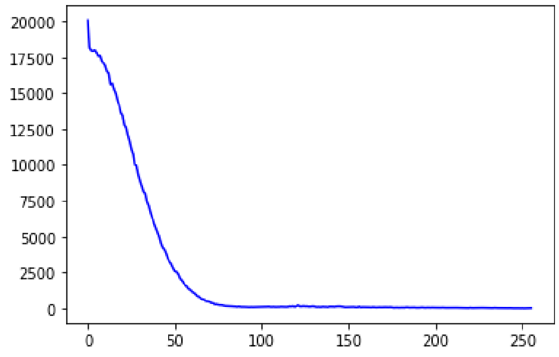
Q2_d_(1):

Red channel :





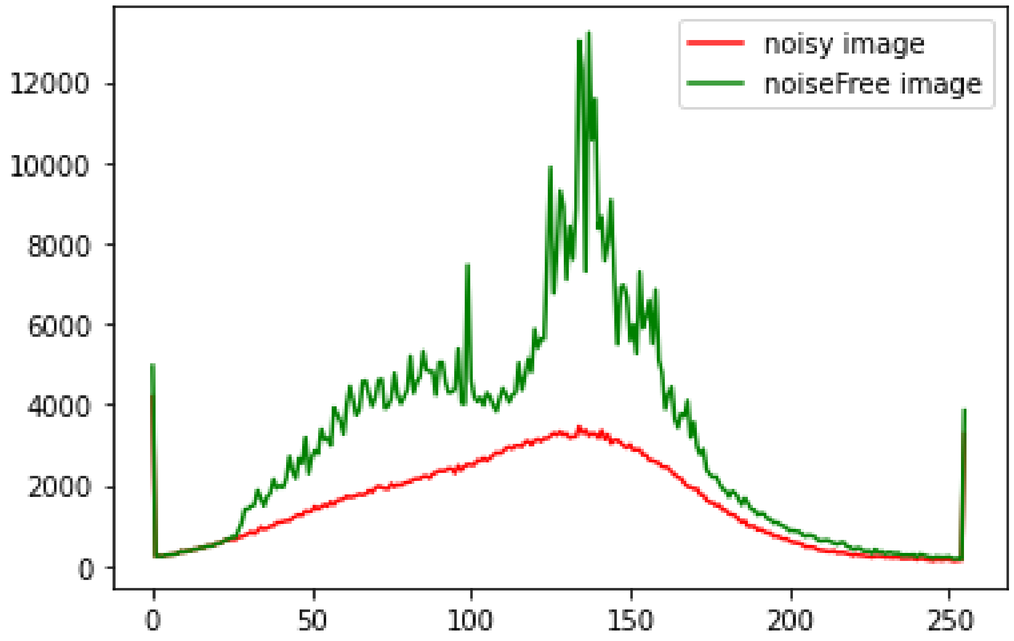
Green channel:



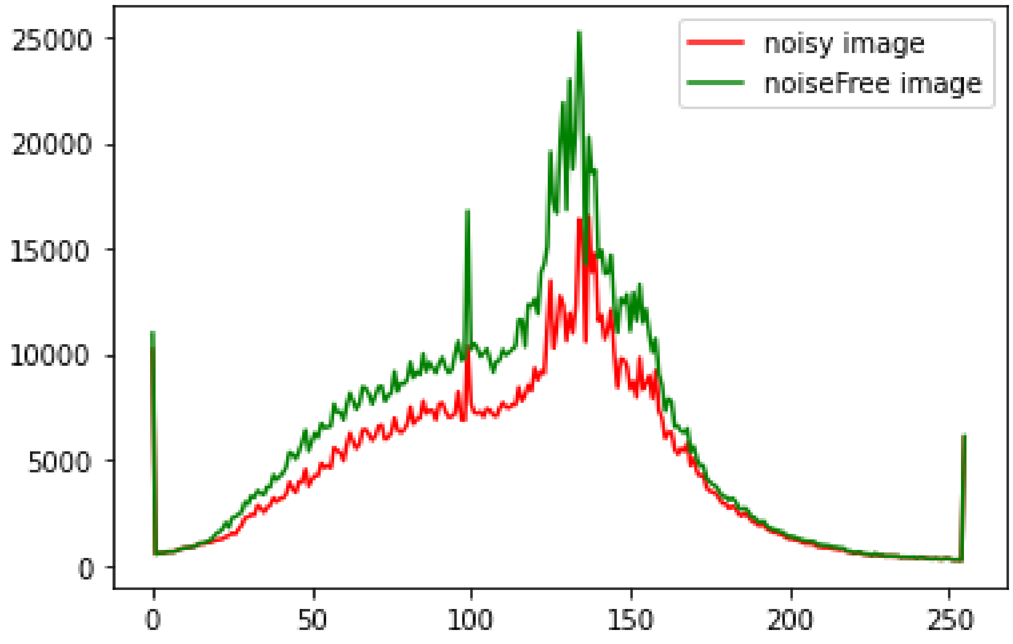
Blue channel:

All these noises seem to be Gaussian noises.

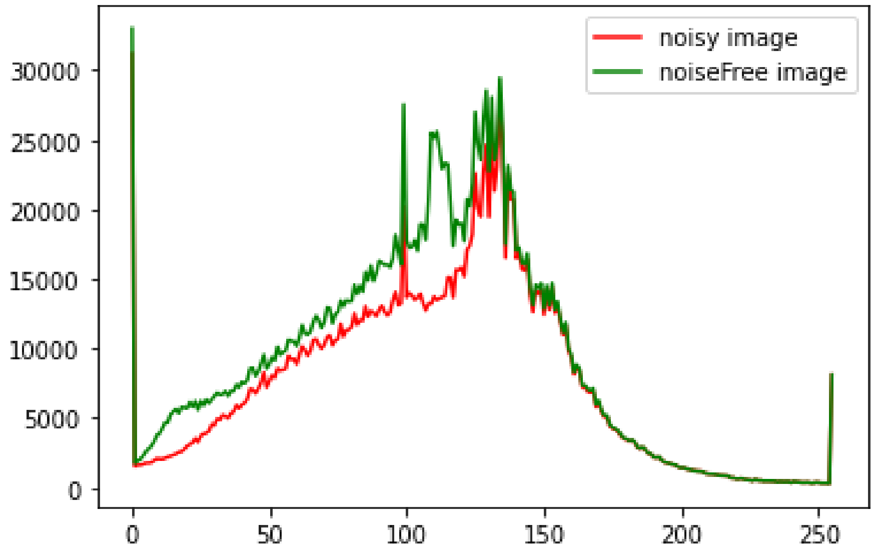
Red channel: Seems like there is Impulse noise around intensity value 150.



Green channel: Seems like uniform Noise distribution over green channel



Blue channel: Seems like, impulse noise around 110.



Q2_d_2: For impulse noise: Median filter,

For gaussian noise: Gaussian Filter

For uniform noise: Gaussian filter/Mean filter.

First, I would like to apply Median filter and then Gaussian filter. No we cant cascade these filters in any order. Median filter has to be first applied to remove impulse noise and then we can apply Gaussian filter and then Mean filter.

Q2_d_3: I treat each noise channel separately. As identified, I'll apply median filter over Red channel first and then apply Gaussian filters over all channels.

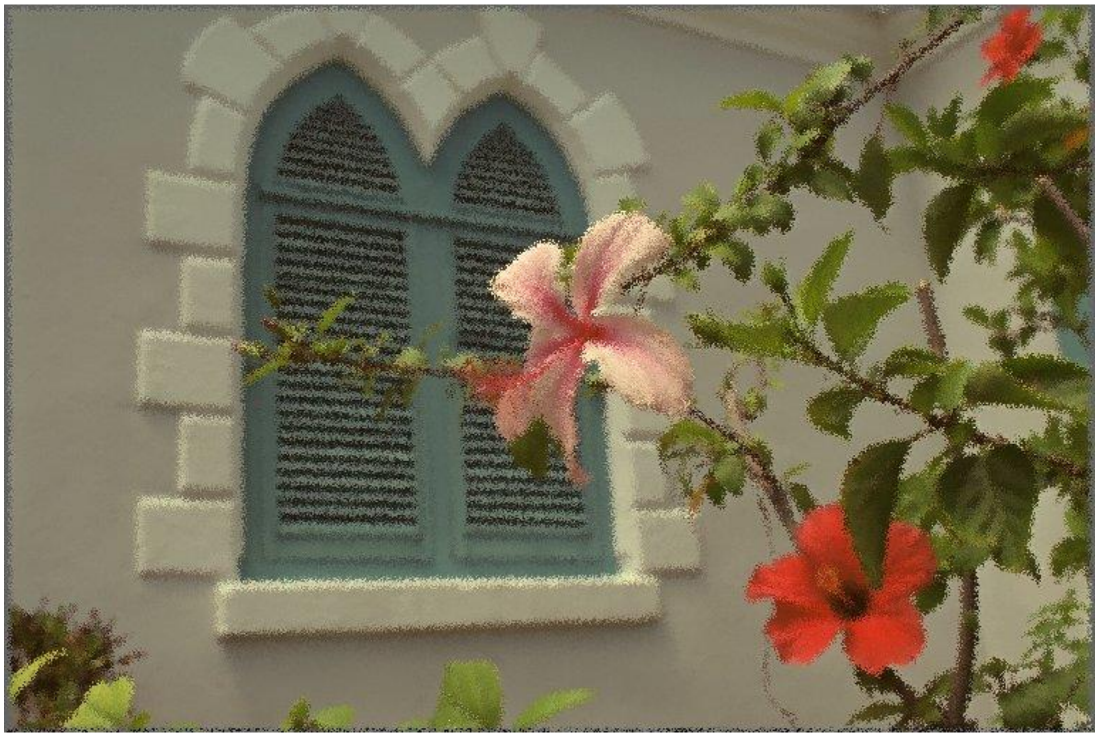
HARDIK PRAJAPATI
HPRAJAPA@USC.EDU
2678294168
1/30/22

Problem 3: Special Effect image filter: Frosted Glass effect

- **Abstract and Motivation:**
  Mobile applications like snapchat have brought in various filters and designs. One such effect is Frosted Glass effect. Its an interesting effect which has glass patterns which is often used in windows to make it unable to seen through although light can transmit.

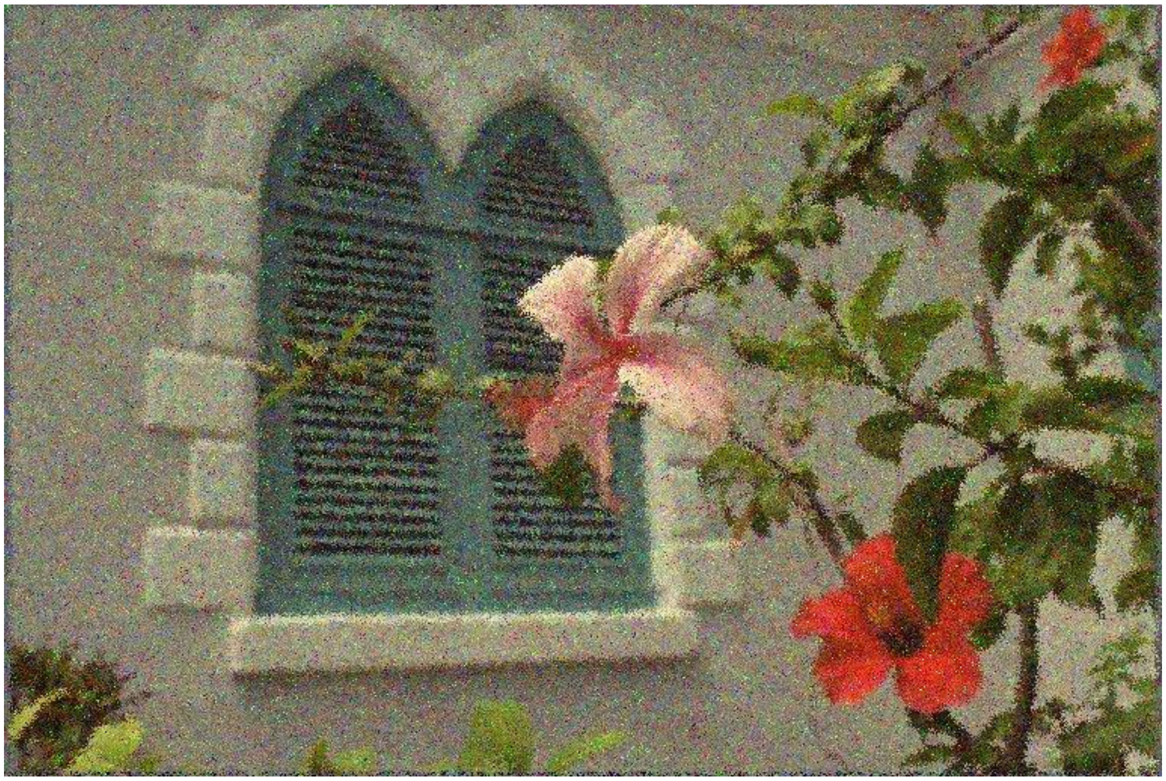
- **Approaches and Procedures:**
  I pad the image with 4 reflected rows and columns. Then I initialize srand(time(0)) which will a random number generator with seed defined as current time. Hence giving different numbers every time I run the program. Then I generate 2 integers in range of [-2,-1,0,1,2] and term them row_idx and col_idx. With this index, I assign the value of this pixel to my target pixel.
  This is similar to 2d convolution. The same index is used for all 3 channels.

- **Experimental results:**
  Q3_1



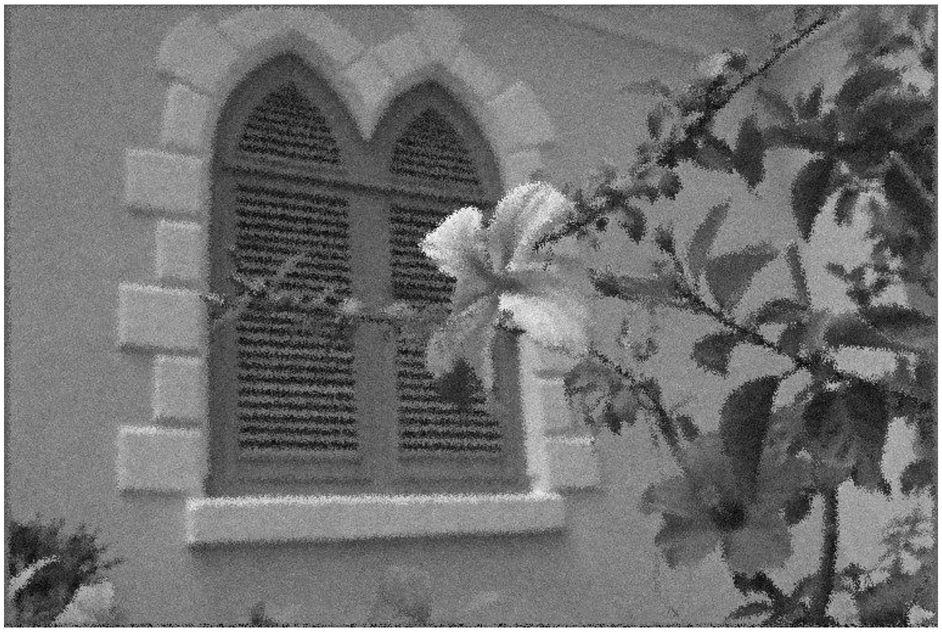
EE569-HW1

Q3_2

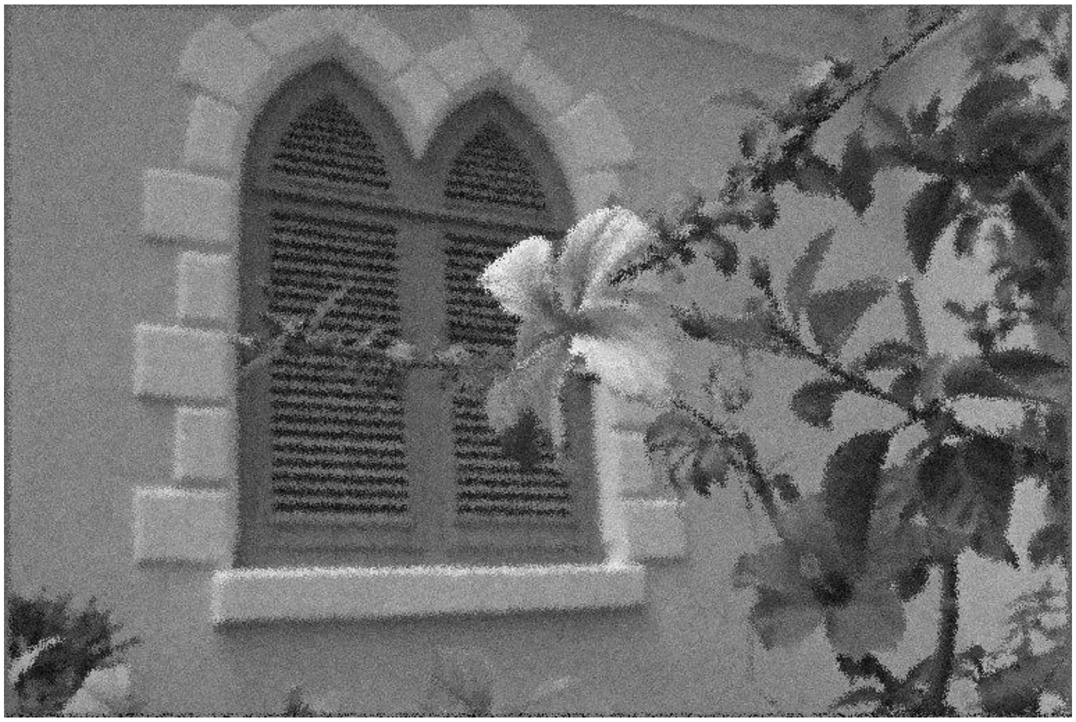


It seems that noise has been amplified. The frosted glass effect is not so much effective now. The frosted glass effect on Noise-free image looks much better.

Q3_3_a



Q3_3_b



Interchanging the sequence does not produce same outputs. In the second (3_3_b)Clearly, the amount of noise amplification that happen on first applying glass effect filter could not be nullified by later denoising it.

- **Discussion:**
  I feel using a better random number generator can produce better results. Also, increasing size of the filter would create more effect on image.