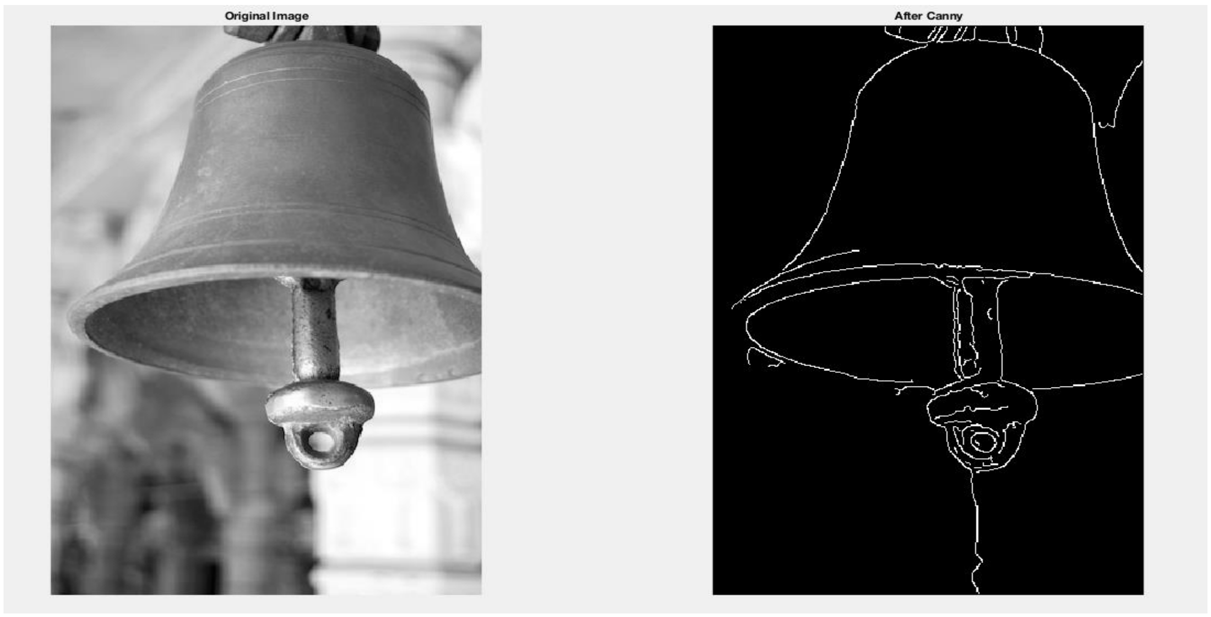
DIP Assignment 2

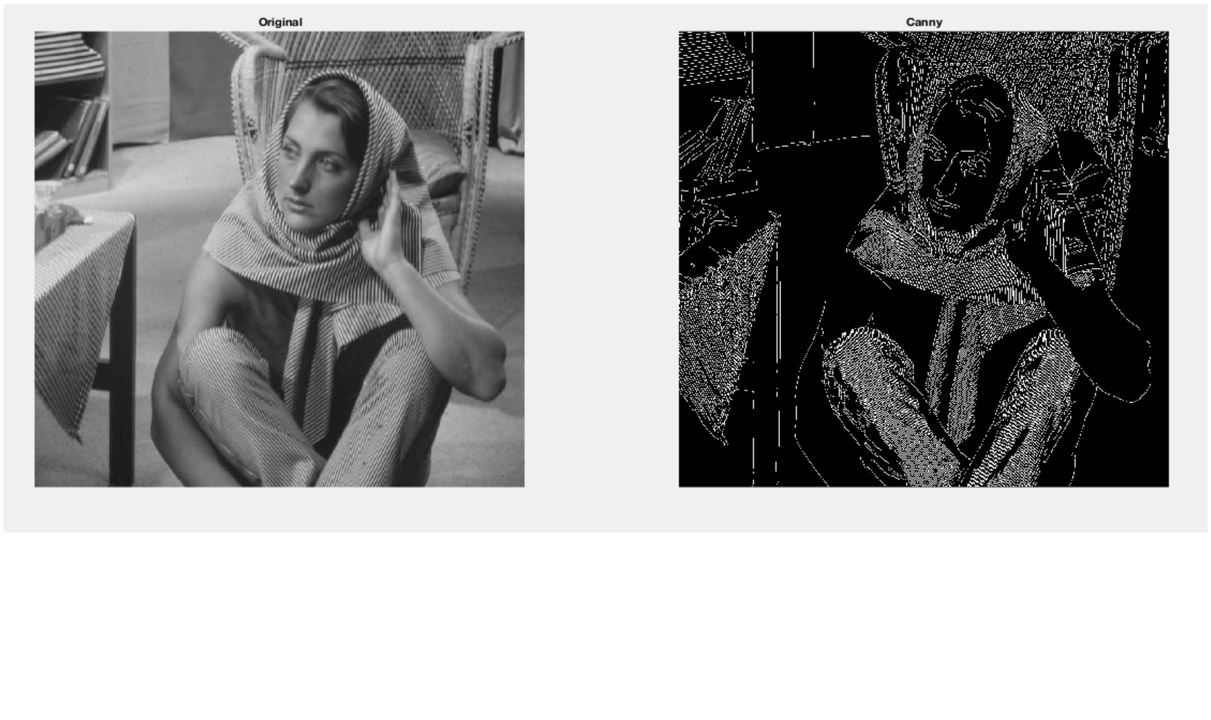
20171157

Harsh Sharma

Question 1:

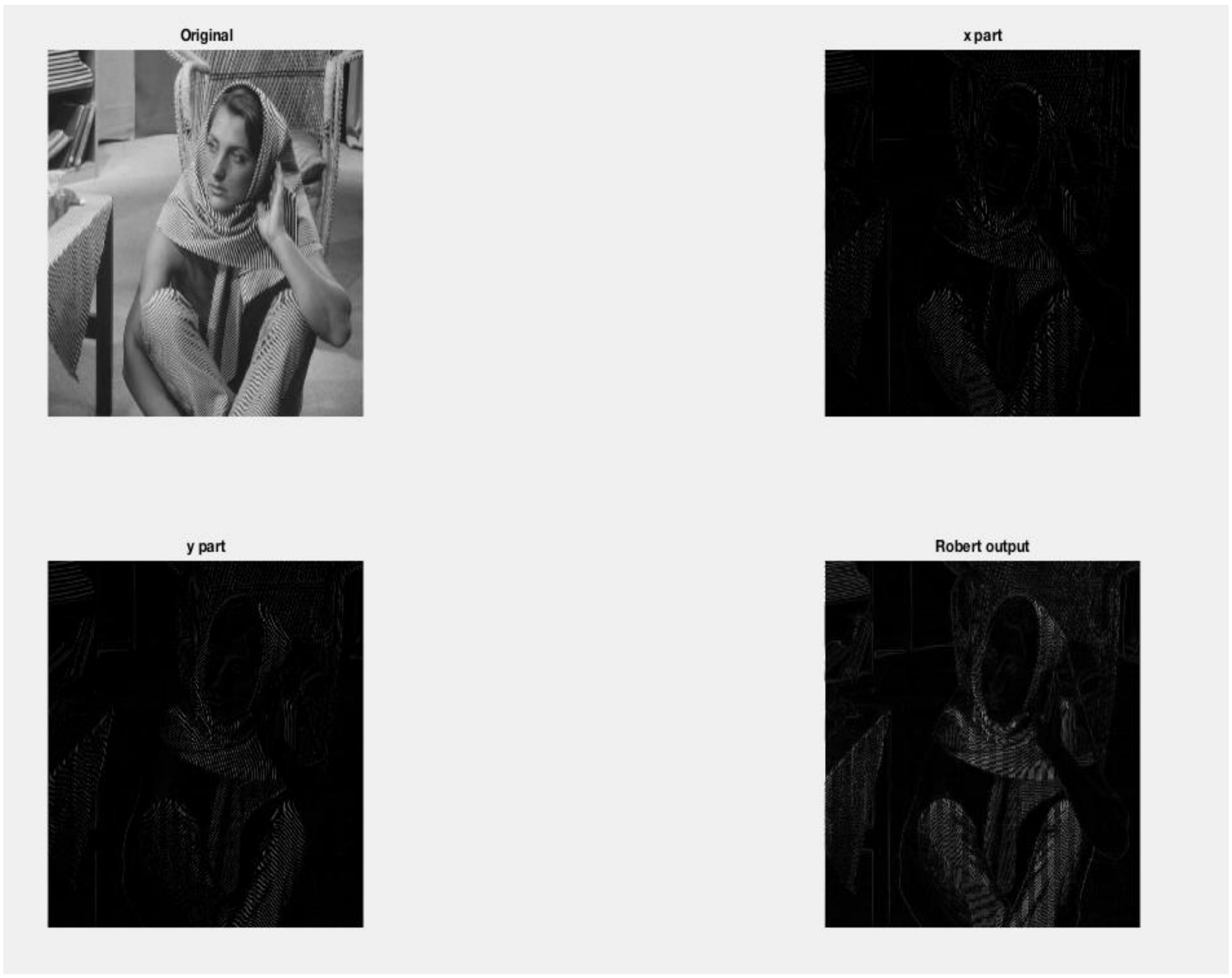
Part_1: As given in question that we can use edge function of matlab so by using this and setting [min,max] as [0 0.23] for 'bell.jpg' and for 'cubes.jpg' as [0 0.10].



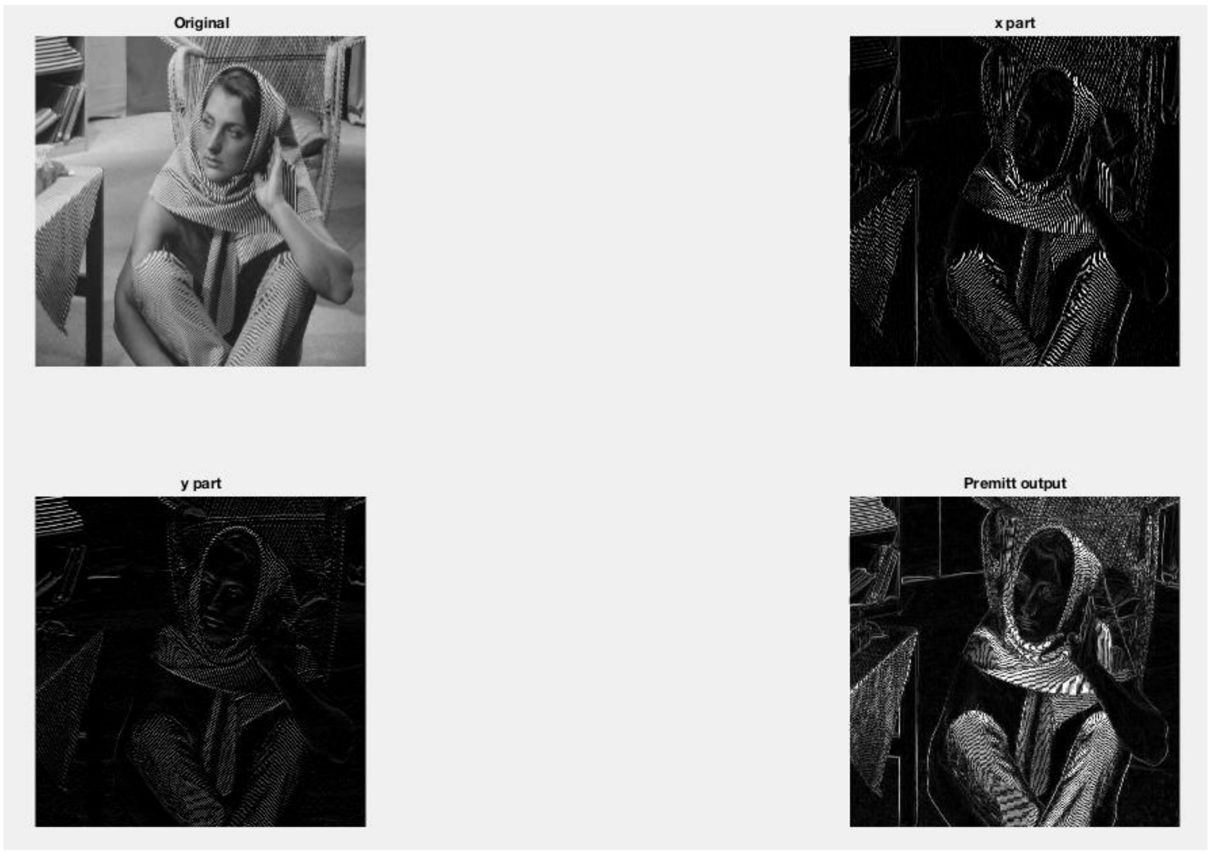


Part_2: In this part firstly, we have to build the filter and then we have to do convolution of the filter with the image. As we have given the matrix for all the filters so we just create the matrix and then run the convolution with x part and y part and then we combine them by using sqrt($x^2 + y^2$).

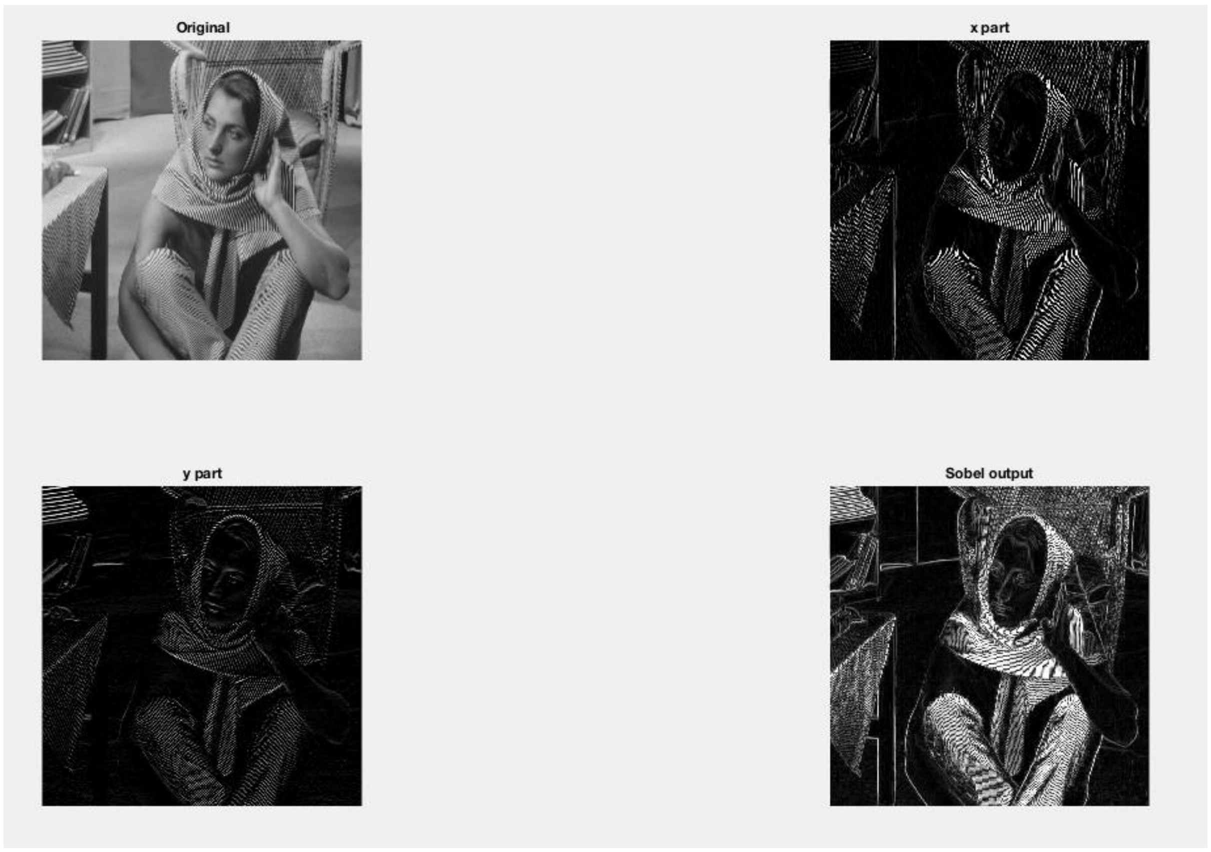
So, the output image and the original image is given below:



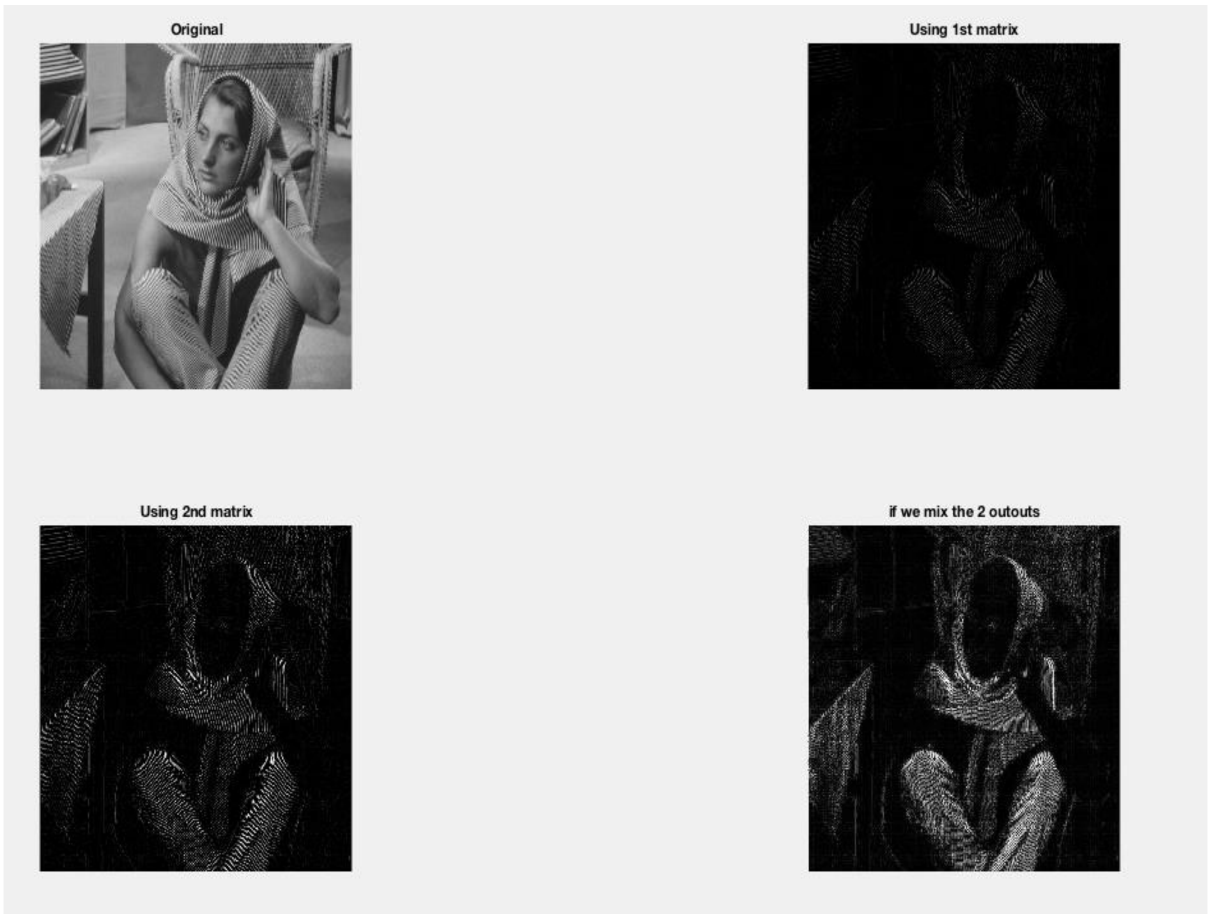
Robert filter



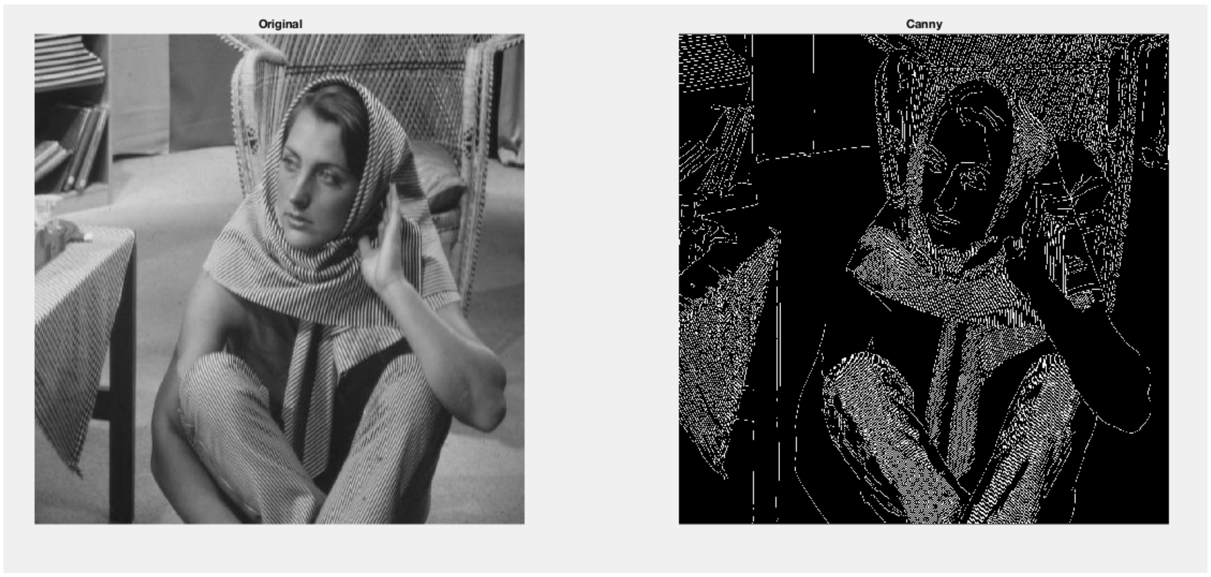
Perwitt filter



Sobel filter



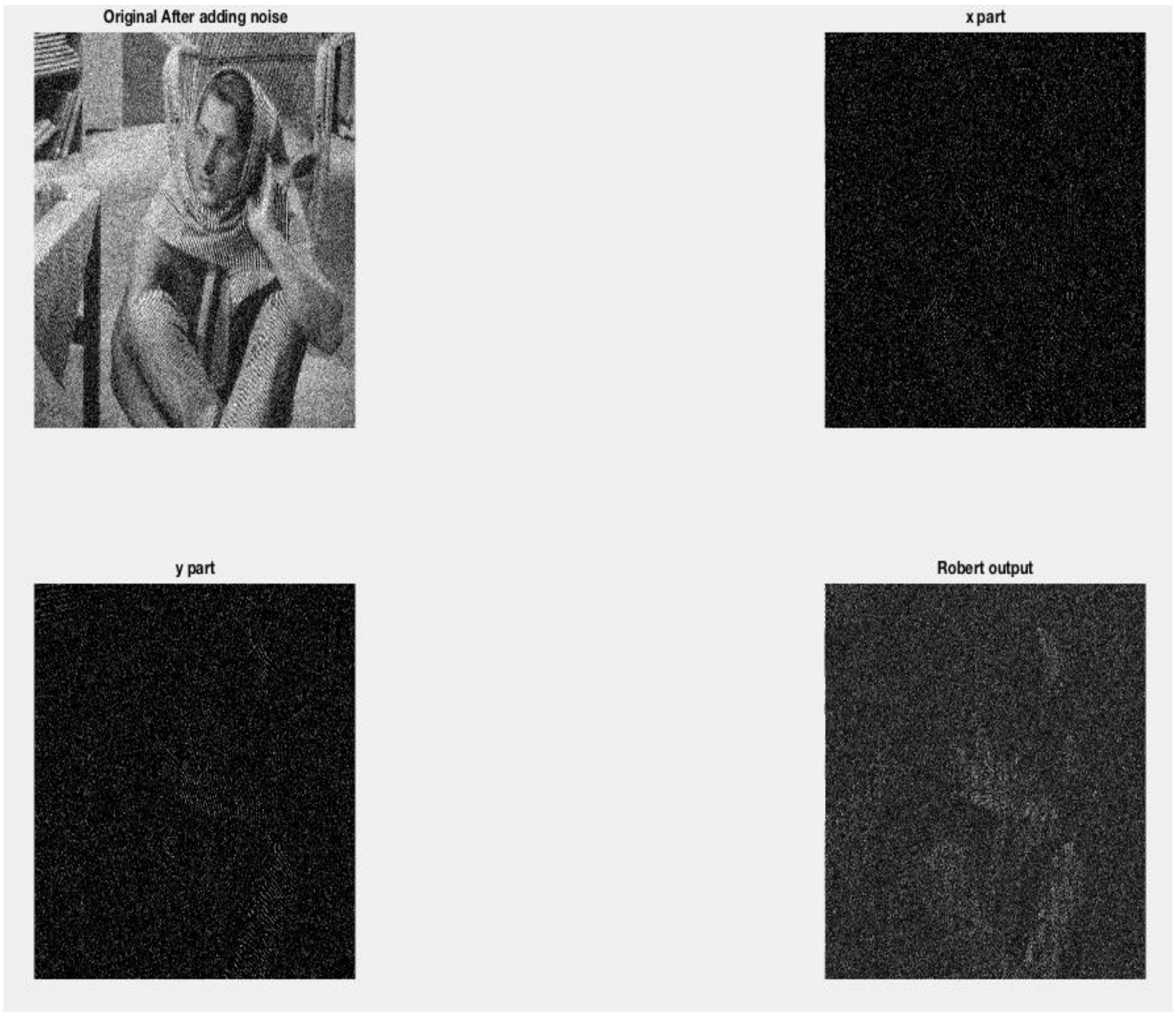
Laplacian filter



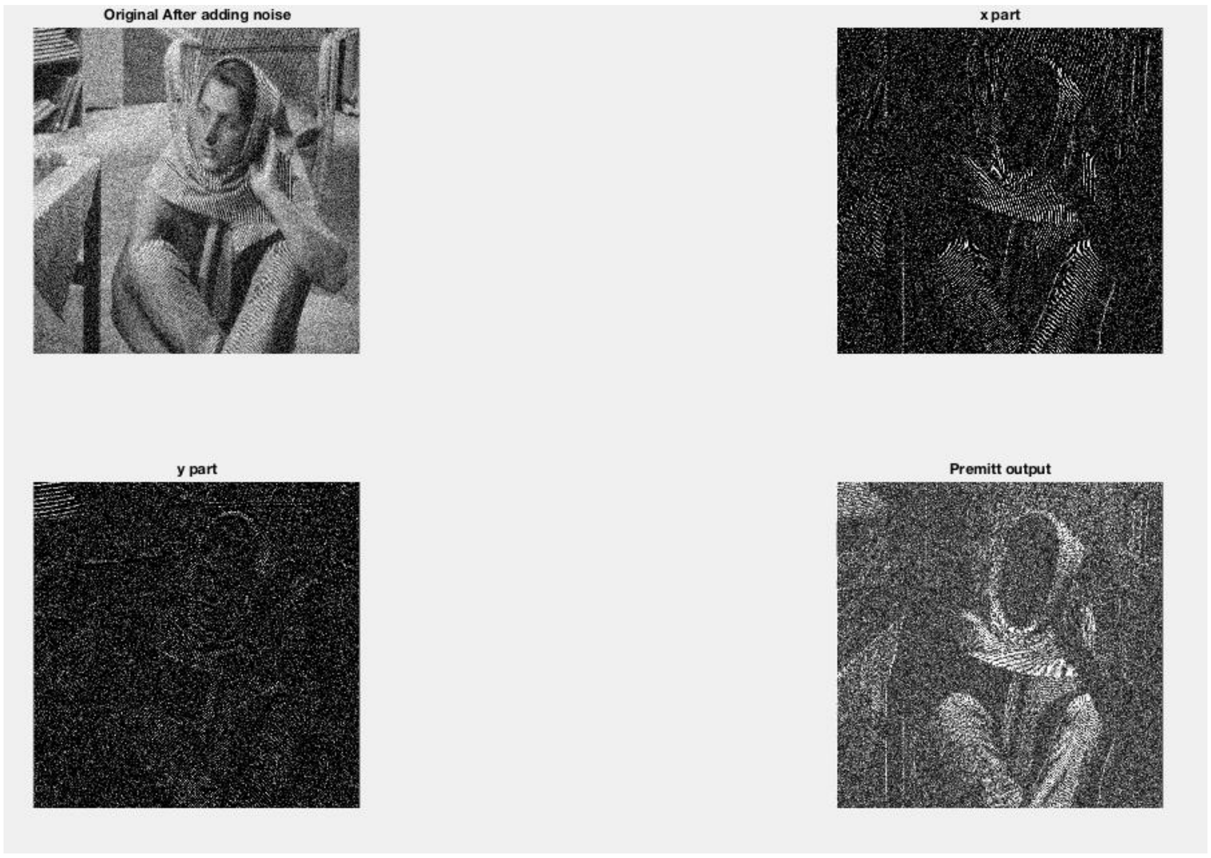
Canny filter

The output for Canny edge detector is shown and on comparing it with others, we see that while it shows only the edges in an image in a certain threshold, while other filters give much smoother outputs. Also, as we can see that the sobel filter is giving the best output among them.

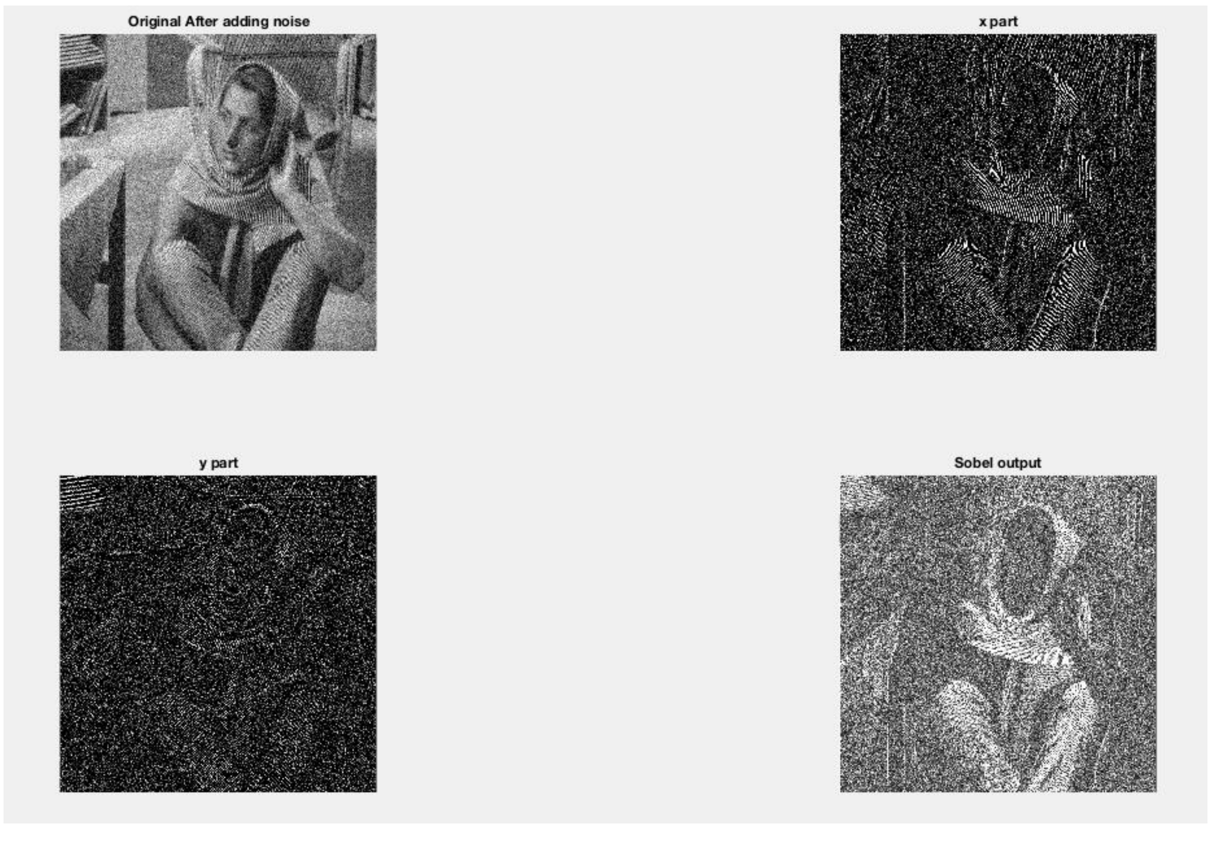
Part_3: In this part we have to add gaussian noise and then we have to apply the above filter on this new image. The output images of each filter with the noisy image is shown below:



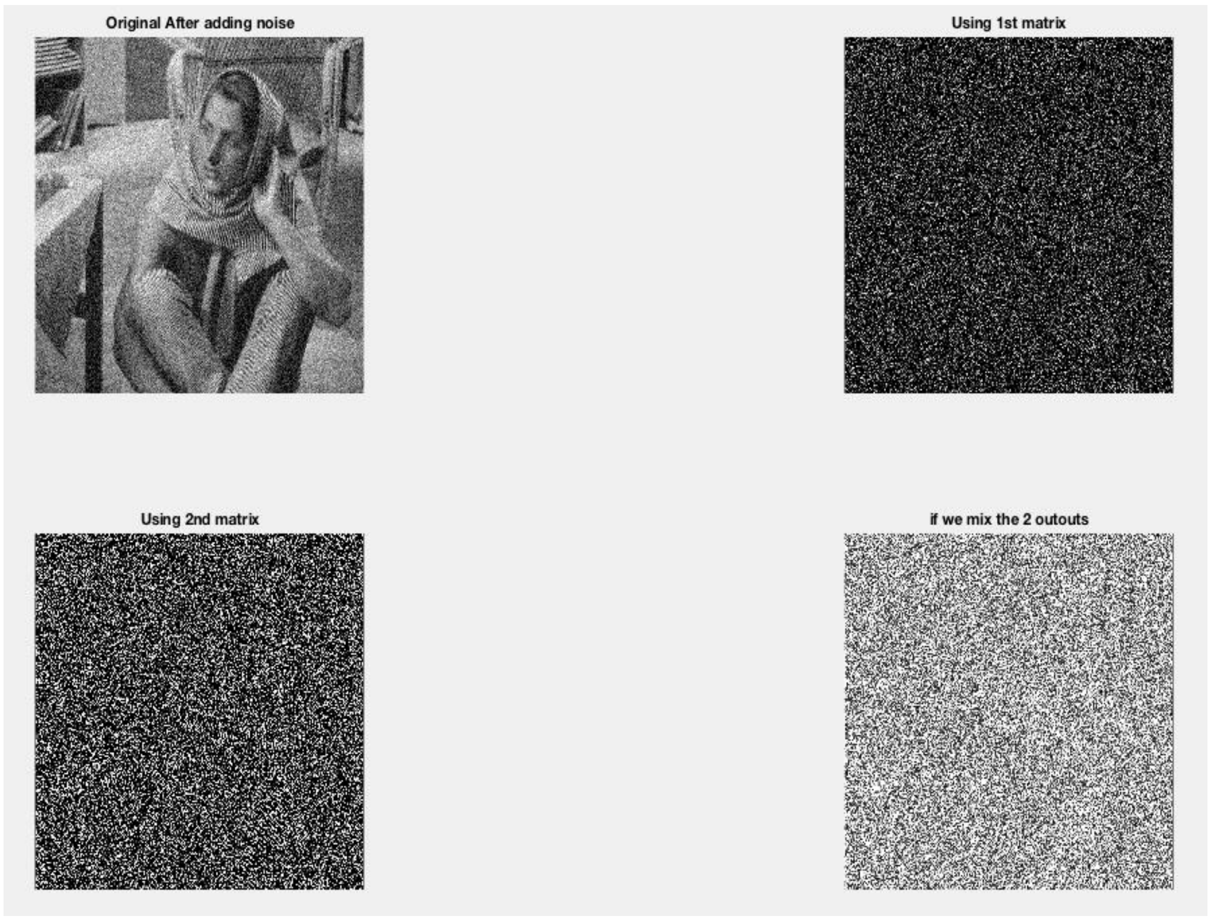
Robert filter



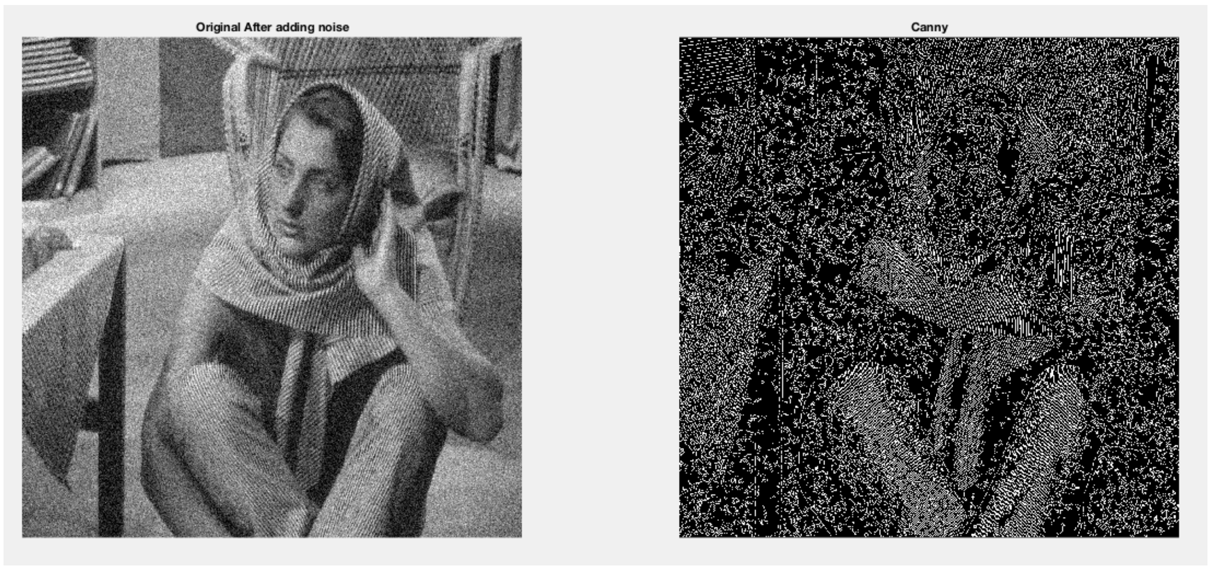
Permitt filter



Sobel filter



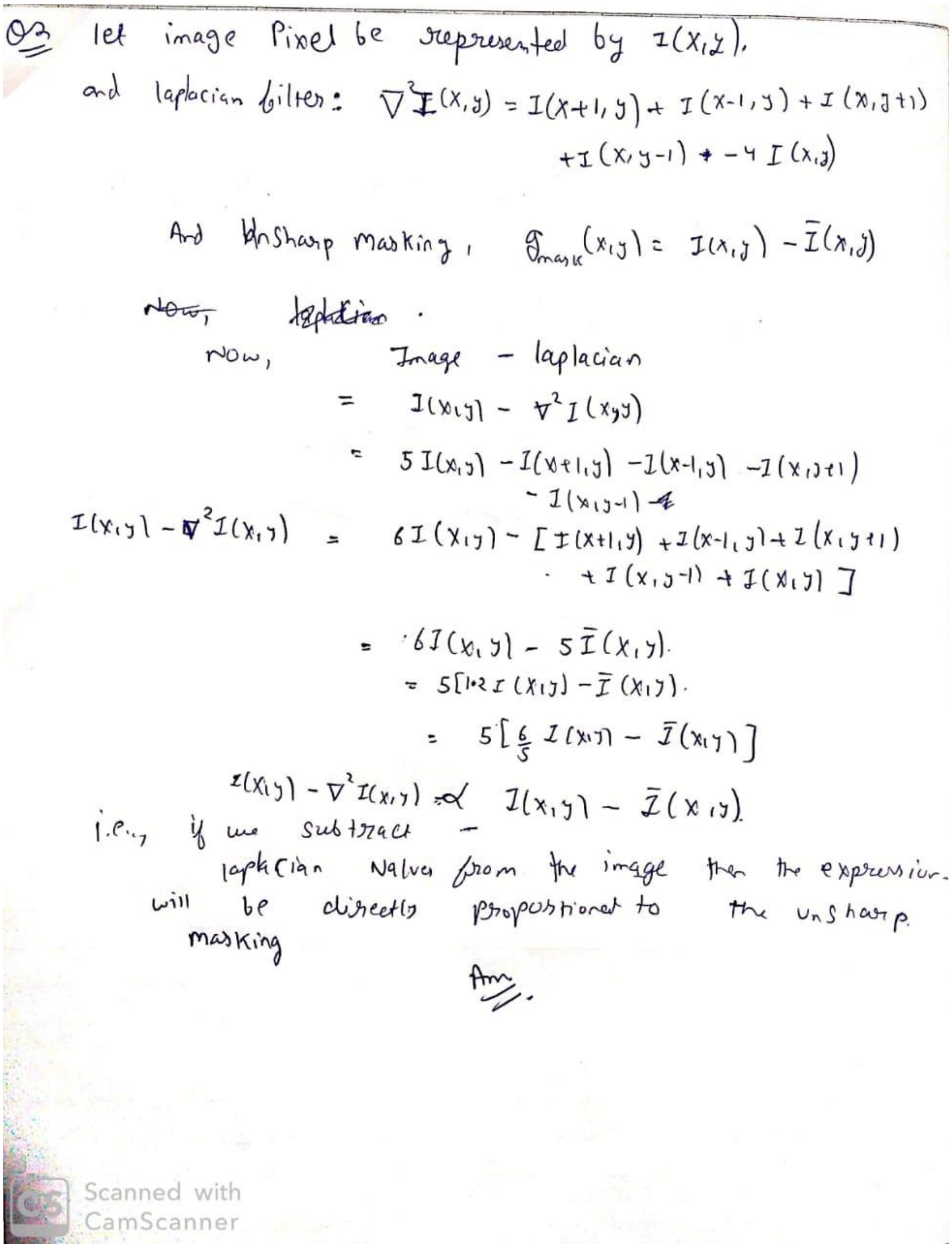
Laplacian Filter



Canny filter

Here the image is getting distorted because of the noise so the output image will also be going to be distorted.

Question 2:



Q3 let image Pixel be represented by $I(x,y)$.

and laplacian filter: $\nabla^2 I(x,y) = I(x+1,y) + I(x-1,y) + I(x,y+1)$
$+ I(x,y-1) \cancel{+} -4\, I(x,y)$

And Unsharp masking, $g_{mask}(x,y) = I(x,y) - \bar{I}(x,y)$

~~Now,~~ ~~laplacian~~ .

Now, Image − laplacian

$= I(x,y) - \nabla^2 I(x,y)$

$= 5I(x,y) - I(x+1,y) - I(x-1,y) - I(x,y+1)$
$- I(x,y-1)$

$I(x,y) - \nabla^2 I(x,y) = 6I(x,y) - [I(x+1,y) + I(x-1,y) + I(x,y+1)$
$+ I(x,y-1) + I(x,y)]$

$= 6I(x,y) - 5\bar{I}(x,y).$

$= 5[1.2\, I(x,y) - \bar{I}(x,y).$

$= 5[\frac{6}{5} I(x,y) - \bar{I}(x,y)]$

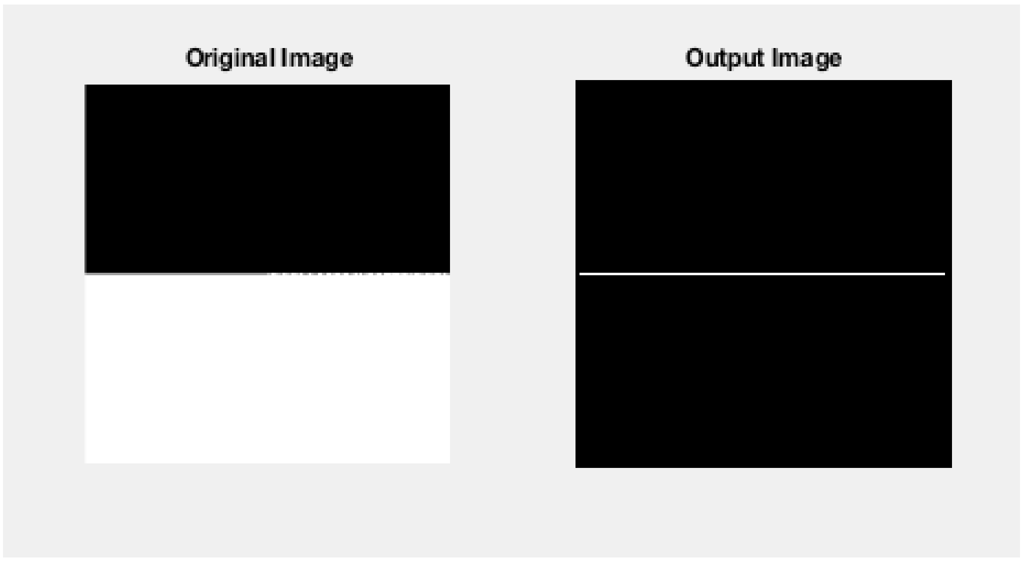
$I(x,y) - \nabla^2 I(x,y) \propto I(x,y) - \bar{I}(x,y).$

i.e., if we subtract laplacian values from the image then the expression will be directly proportional to the unsharp masking

Ans.

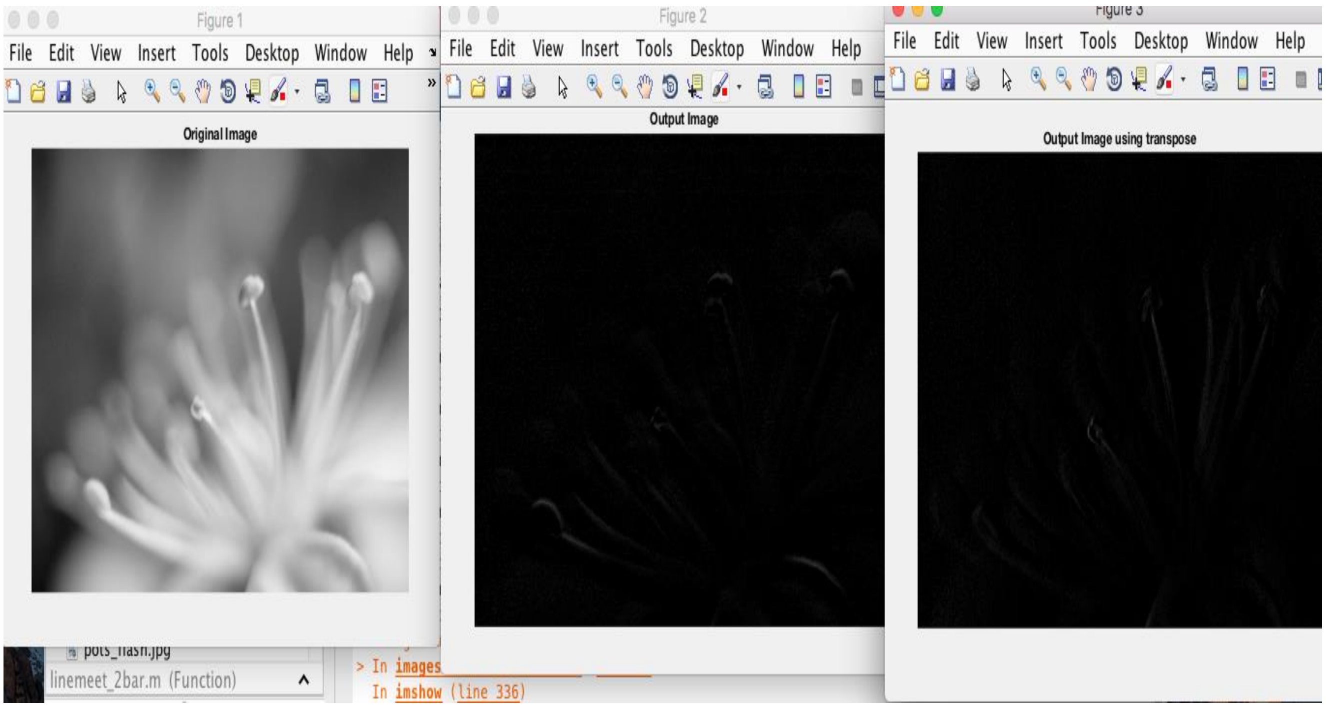
Question 3:

Part_1: In this question we have to create the matrix and then we have to take the convolution if the image with the matrix. To identify the horizontal boundary i have the filter that can full fill the requirements as mtx=[1,1,1;0,0,0; -1,-1, -1]; The input and output image is shown below:



As we can see the in the output image, we can see a white boundary which is representing the line where white and black meets.

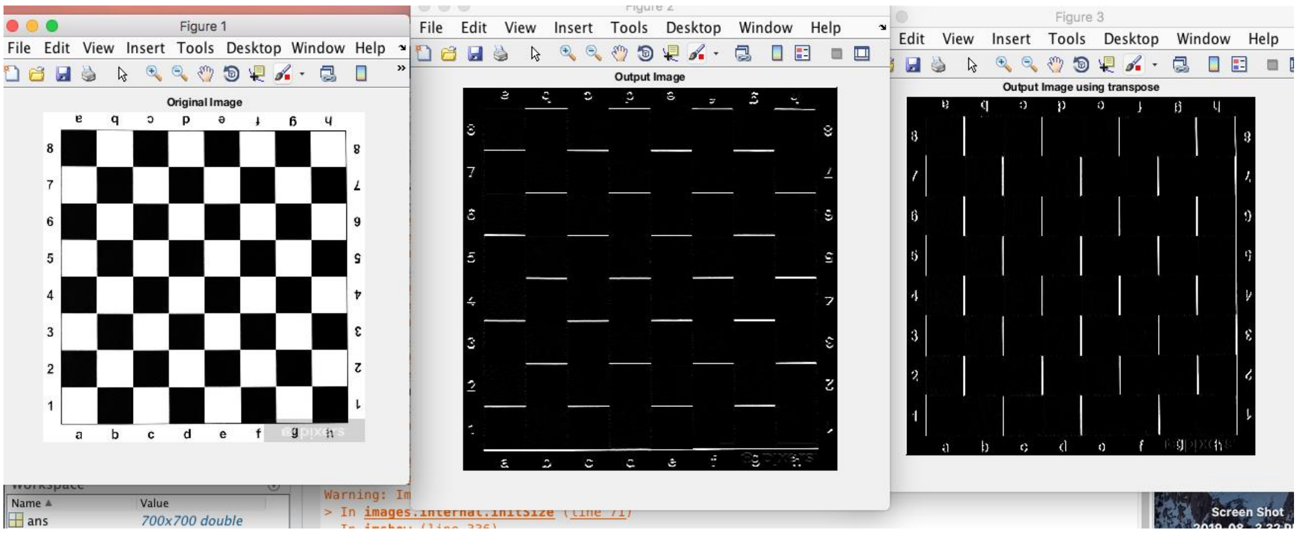
Part_2: Taking the matrix as above and the image as blurr,jpg, and then after taking the first output we have to take the transpose of the matrix(because it's the requirements of the question). The output image is shown below:



1st one is the input; middle one is the output without taking transpose and the last one is the the output image after taking transpose.

As we can see that the matrix is acting as a horizontal boundary detection and while its transpose is working as the vertical boundary detection. And this result comes can be seen as when we are not taking the transpose it is acting as horizontal detector but if take the transpose of the matrix i.e., we rotate the matrix so now for it, the vertical boundary is as a horizontal detector.

Part_3:



The above image is of a chess board where we can see that our filter is working fine i.e., its detecting the horizontal part while its transpose is detecting vertical part.

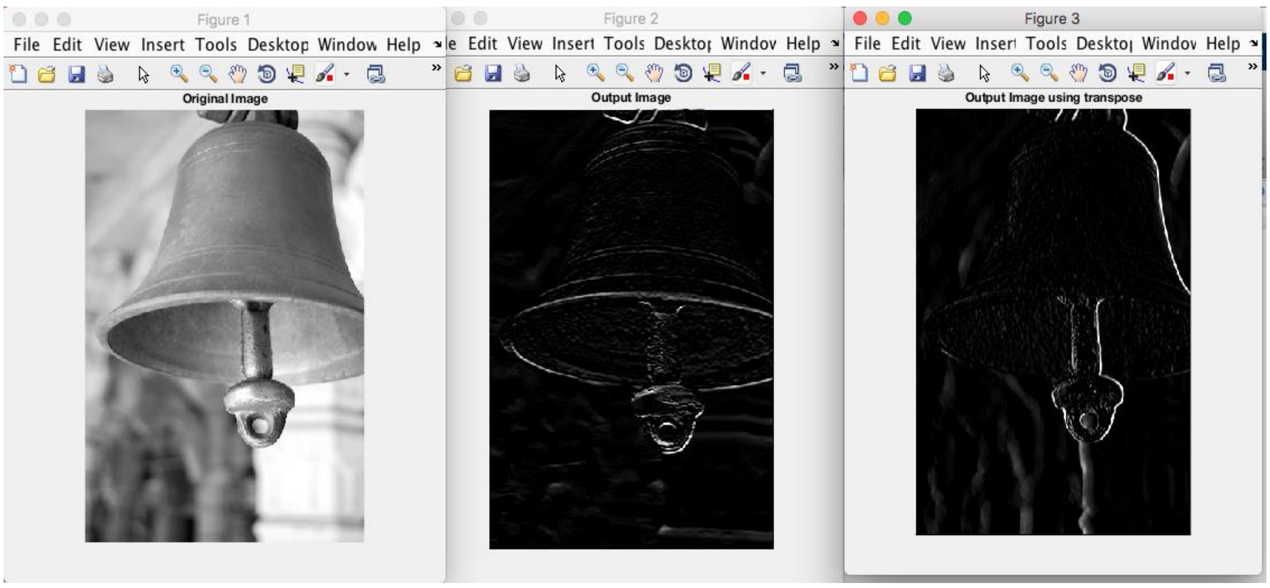
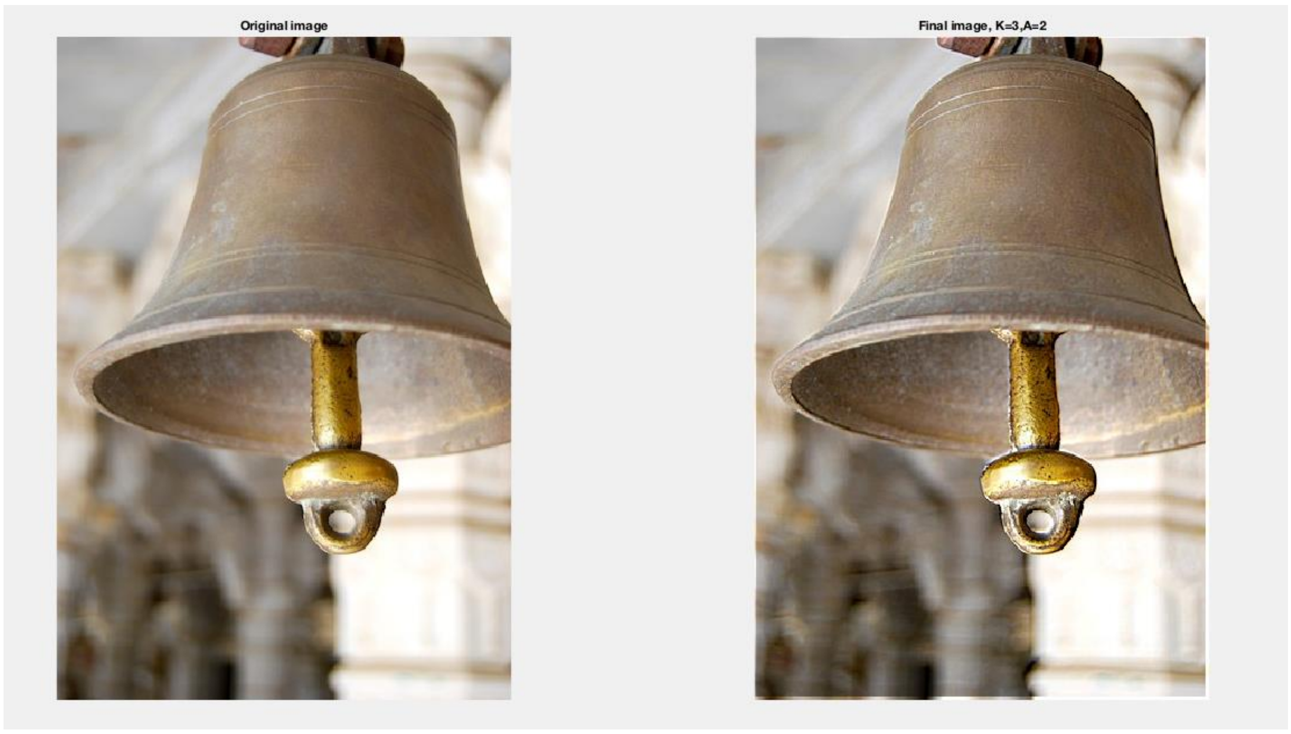
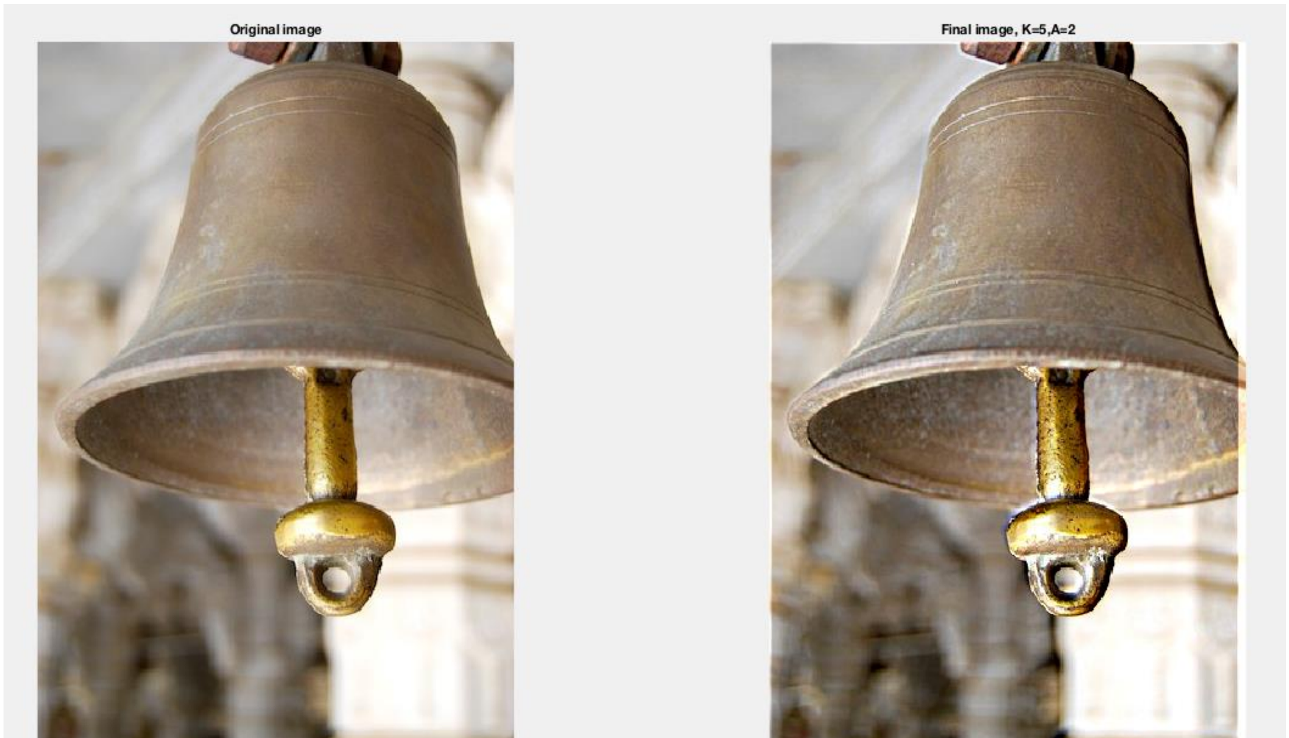


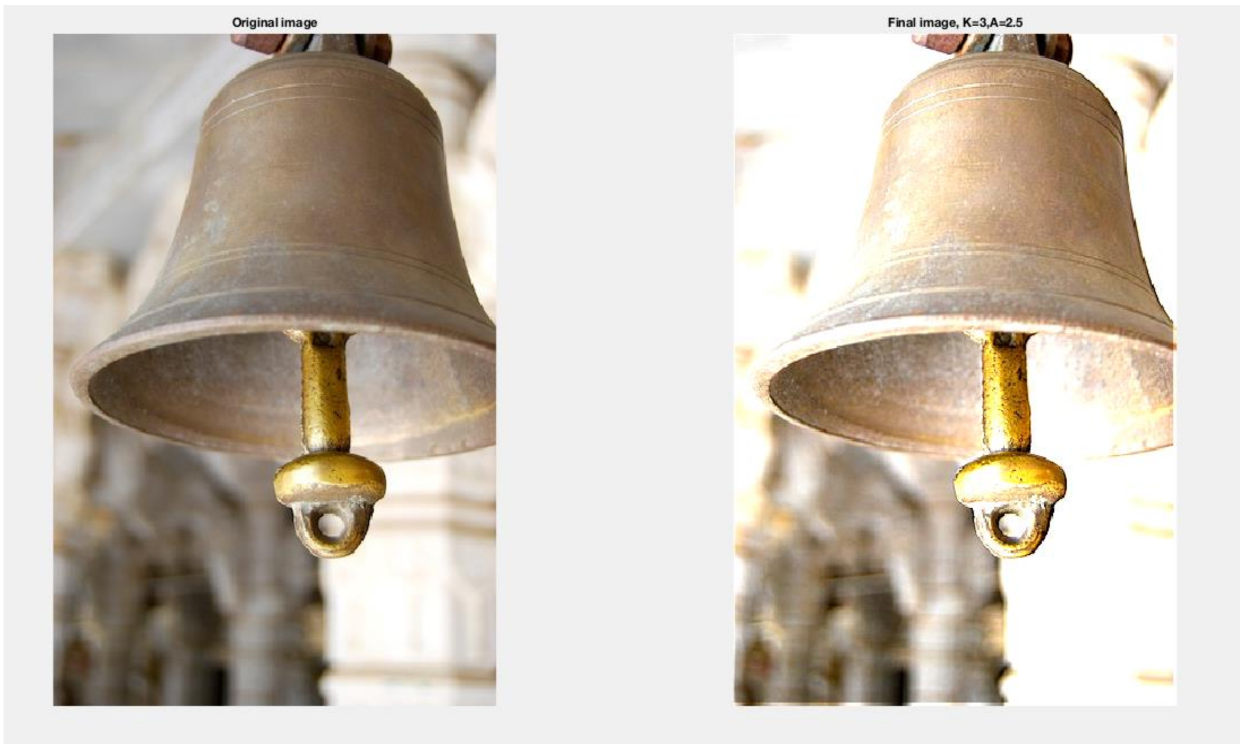
Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
Figure 2
Output Image
Figure 3
Output Image using transpose

Question 4:

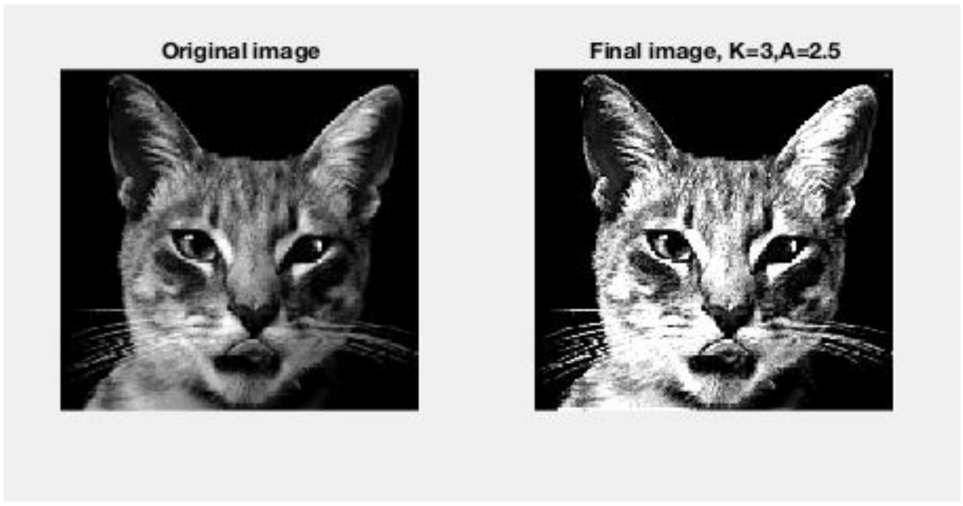
Part_1: In this part we have to implement high boost filter so there are two ways of doing that, one is that by multiplying A with the image and subtracting the low pass output from image i.e., A*Img – lowpass, and the other is multiplying A-1 with image and adding high pass output, i.e., (A-1)*Img + high_pass. I have implemented the lowpass one because it is easier to implement that than the high pass one because in high pass one we have to make sure that all the intensity values are coming in the range of 0 to 255 so if values is not in this range then we have to map the intensity values to [0 to 255]. While we don't have to do such things in low pass one. I have varied the values as (A =2, k=3), (A = 2, k=5), and (A = 2.5, k =3).

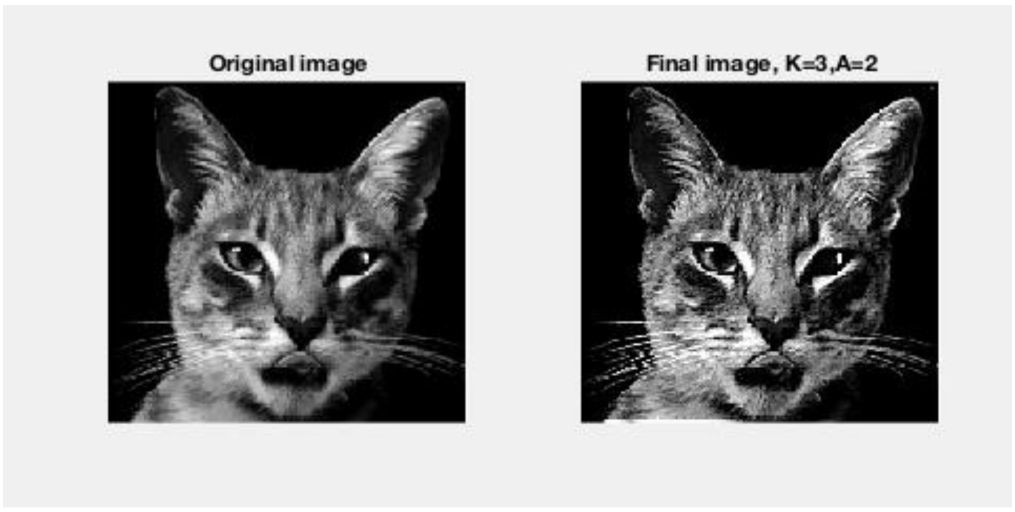

Original image
Final image, K=3,A=2

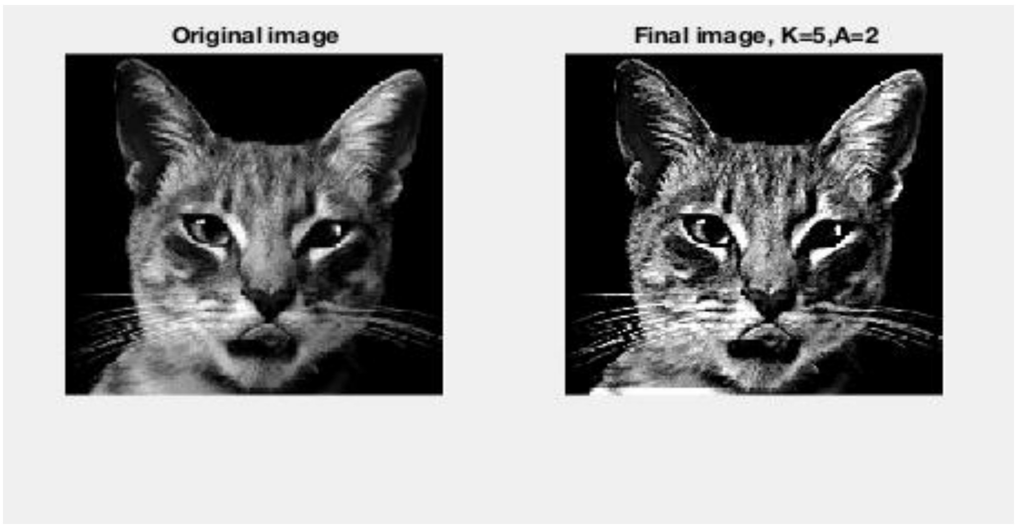

Original image
Final image, K=5,A=2



Part_2: I have varied the values as (A =2, k=3), (A = 2, k=5), and (A = 2.5, k =3).





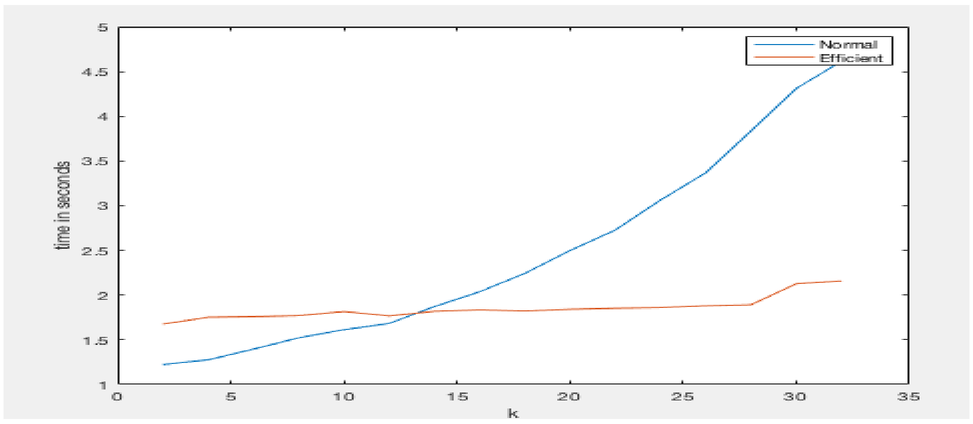


Part_3: It is often desirable to emphasize high frequency components representing the image details (by means such as sharpening) without eliminating low frequency components representing the basic form of the signal.

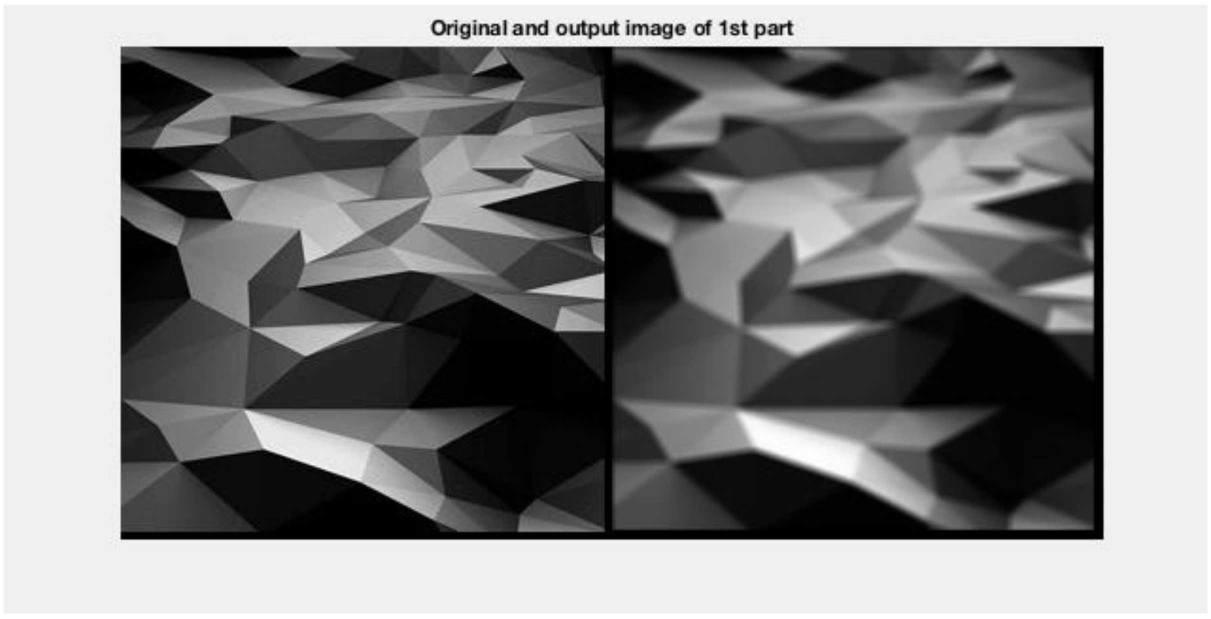
A **bilateral filter** is a non-linear, edge preserving, and noise reducing smoothing filter for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution. Crucially, the weights depend not only on Euclidean distance of pixels, but also on the radiometric differences (e.g., range differences, such as color intensity, depth distance, etc.). This preserves sharp edges.

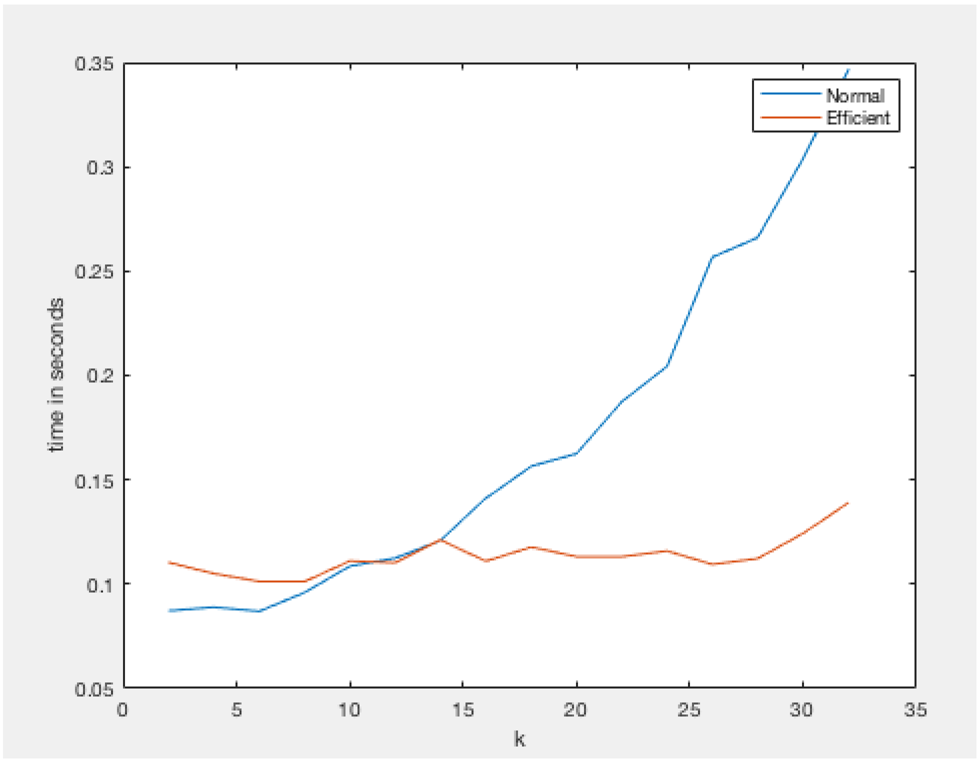
Question 5:

Part1_and_2: I have implemented the averaging filter that behaves as low pass filter and it causes blurring effect. And also, as we are using [k k] matrix and then we are convoluting it with the image so its taking time so process the output so to minimise the time or to implement the efficient version, I used the information stored in previous cell to avoid recalculating some values again. For any row (except the first one), the value is previous row minus sum of top row values used in previous row plus sum of bottom row values used in current row. Similarly, for columns we do the same.



Size of Image: 1024x1024x3





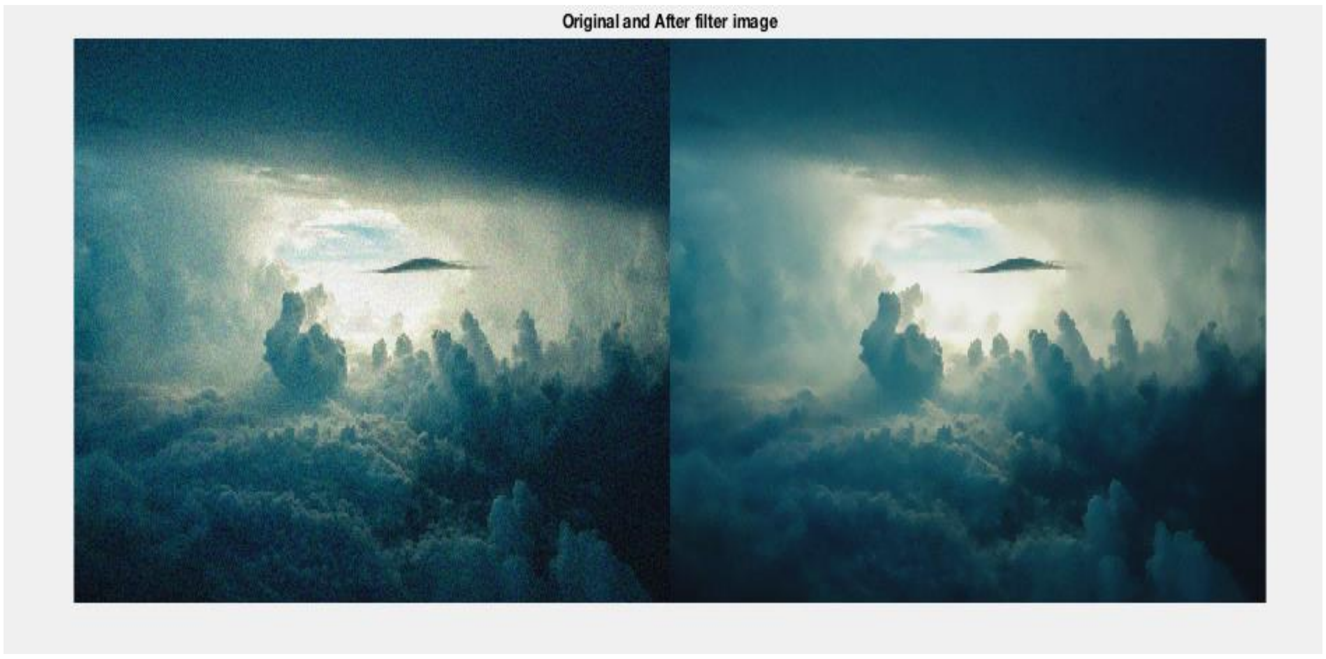
Size of image:256x256x3

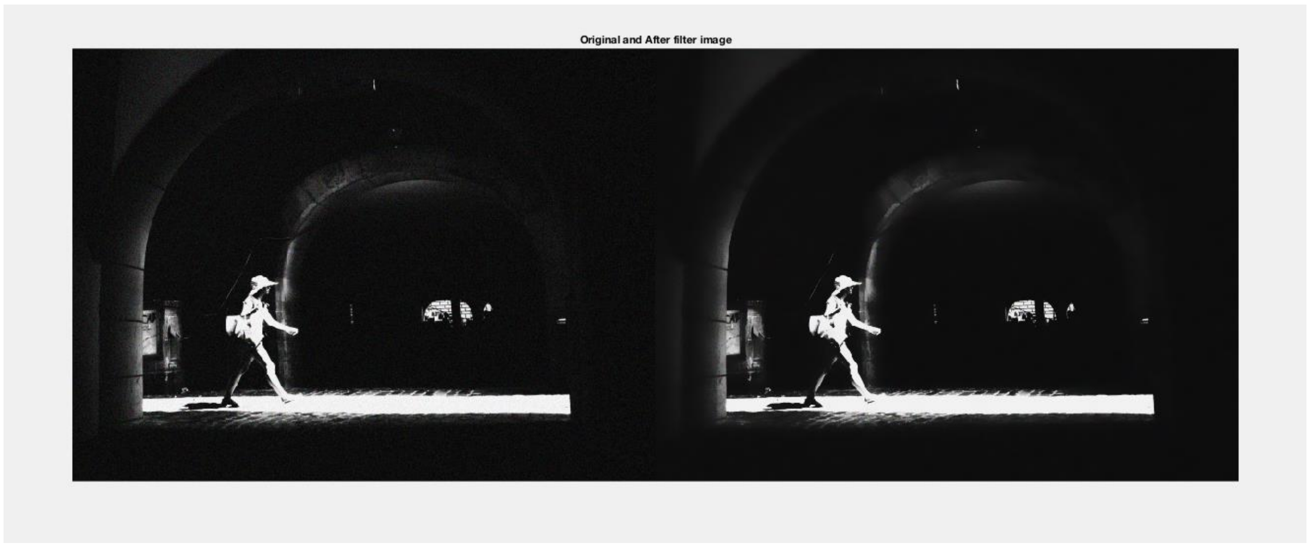
Part_3: One way is the way which I did in question 8.To optimise the median function we have firstly implemented the function which similar like im2col function and then we sort the the matrix and then take the median and then we resize it to get the desired result.

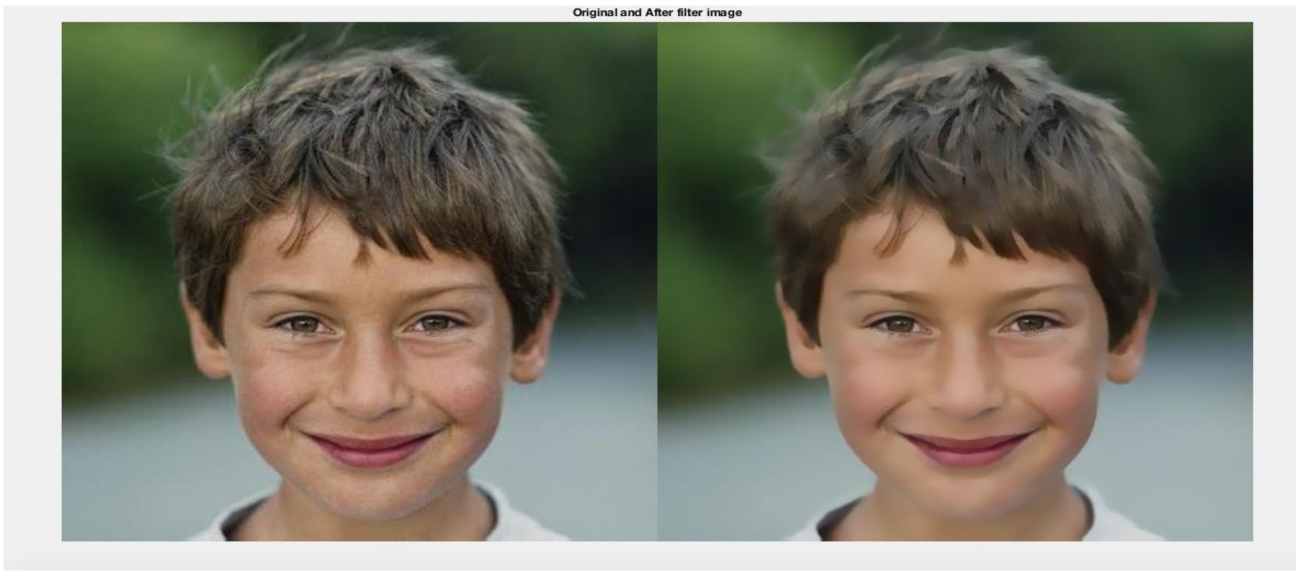
Question 6:

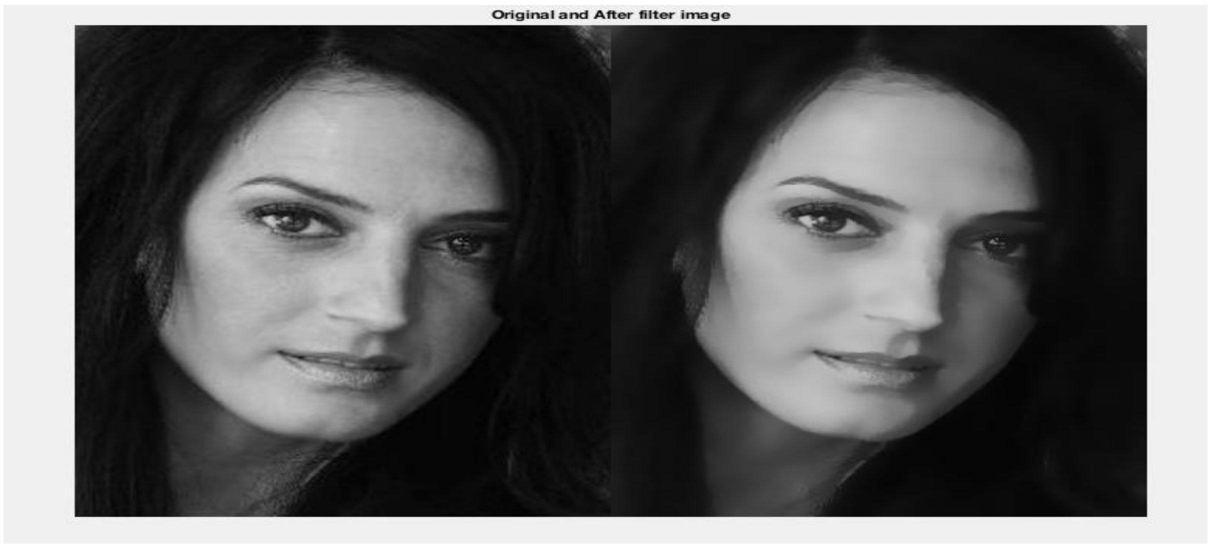
Part_1: It is implemented by the use of domain filter and range filter. Like firstly we have to find the matrix for domain and range filter and then we multiply them to get the bilateral filter. In this, we firstly stored all the edges and then we try to blur and try to do sharp the image and then we again add the edges to the image to get the final output.

Output:

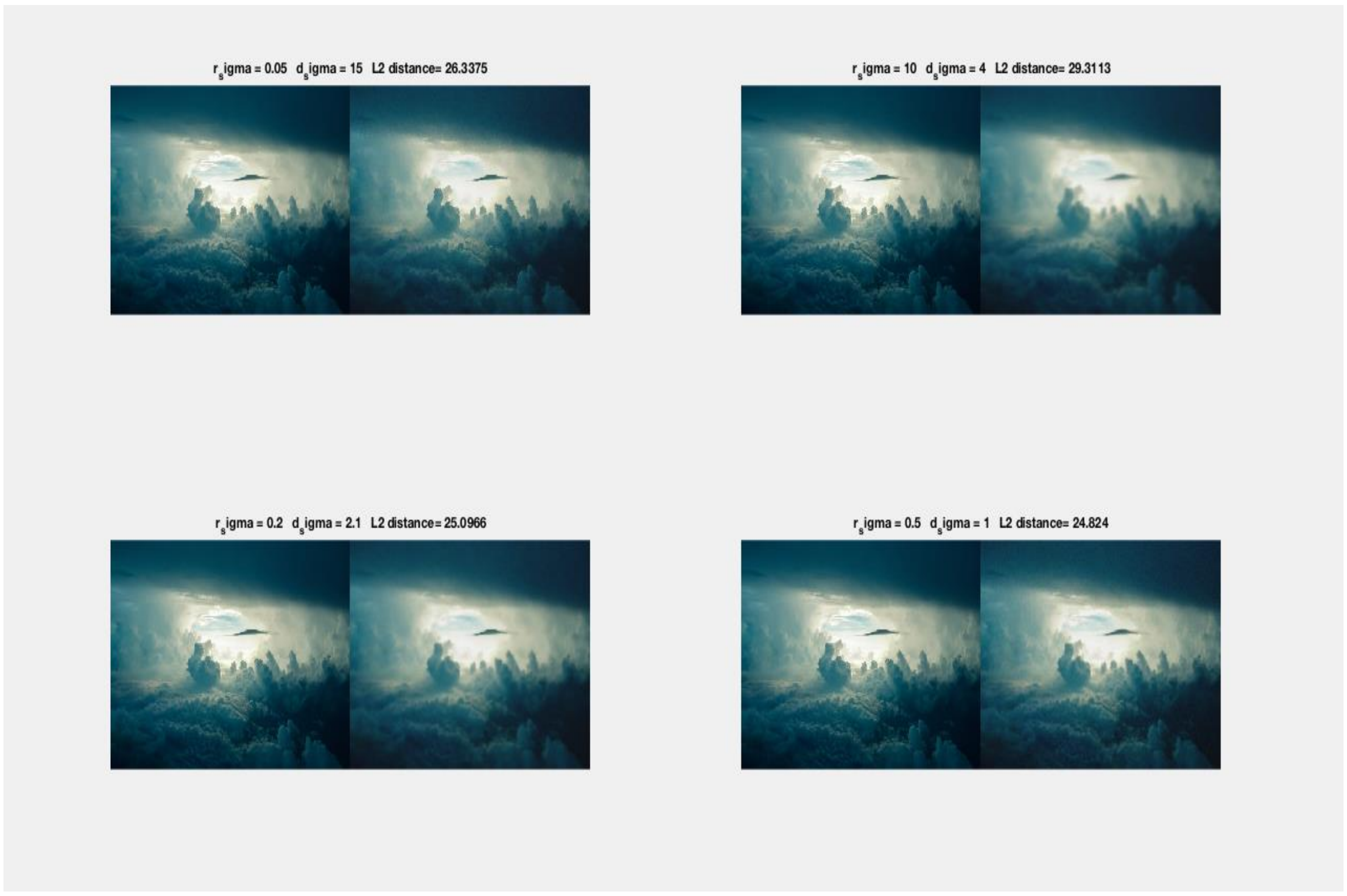






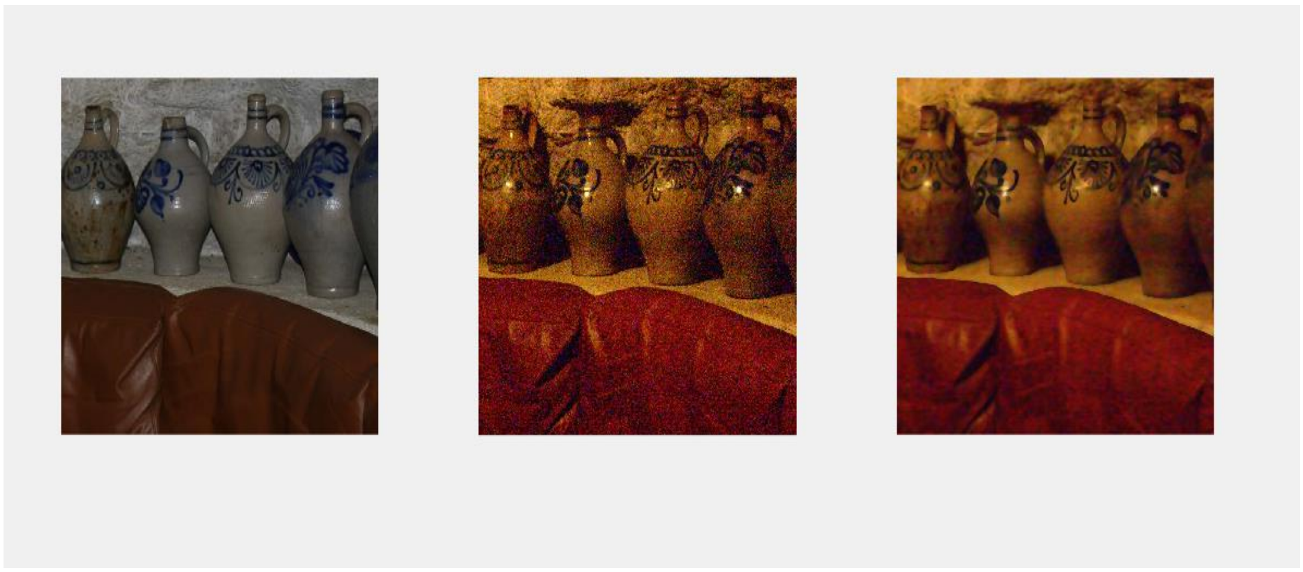


Part_2: On varying the domain sigma and range sigma we take some observation and the find the L2 distance for all observation. The output of each observation with the values is shown below:

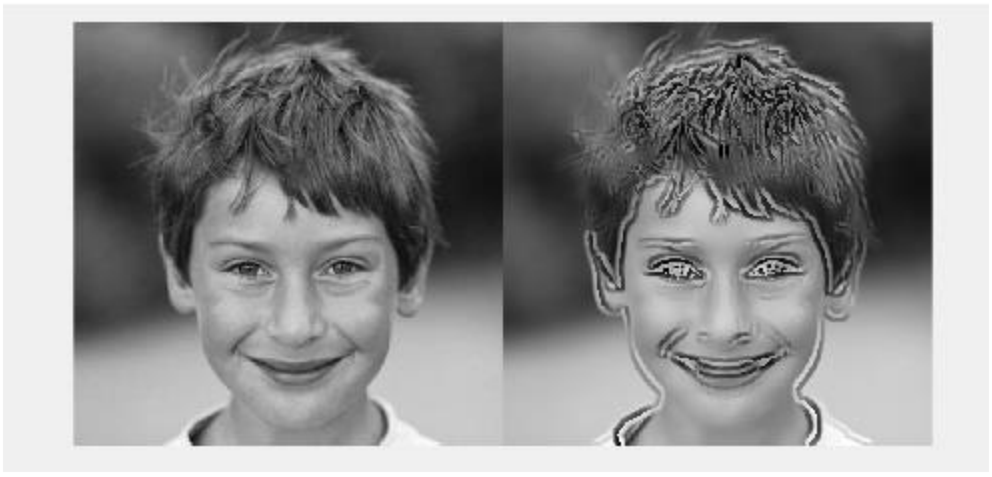

r_sigma = 0.05 d_sigma = 15 L2 distance= 26.3375
r_sigma = 10 d_sigma = 4 L2 distance= 29.3113
r_sigma = 0.2 d_sigma = 2.1 L2 distance= 25.0966
r_sigma = 0.5 d_sigma = 1 L2 distance= 24.824

Question 7:

Part_1: Bilateral Filter performs 2 functions i.e. Preserving edges and Gaussian Blurring to reduce the noise. However, a variant of Bilateral Filter called Joint or cross bilateral Filter can be used on 2 different images where the range filter is applied to one image while domain filter on the other image. This idea can be used in case of Low Light Imaging where one filter is applied on a NO FLASH Image while the other one on FLASH Image. We find that it is coming good for window size = 5, sigma_r = 0.05 and sigma_d = 15.

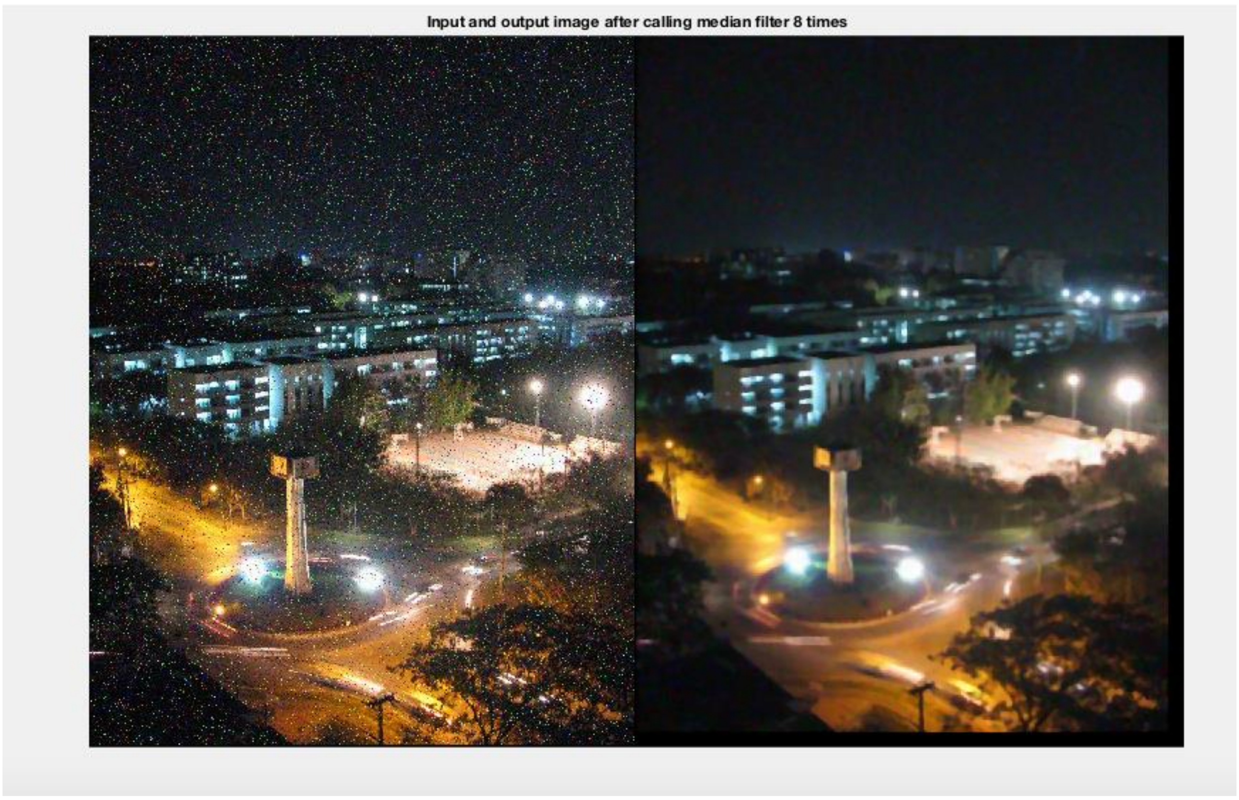


Part_2: It totally make sense because consider some situation where we just want the edges like when we do inverse bilateral filtering then the edges got highlighted as we can see in the below image. Application: Salient feature detection.



Question 8: Median filter can be done by using for loops like firstly we have to do padding and then collect all the elements of size equal to window size and then sort them and then take the medium of that and then slide to the next window.

I’m running my median filter with the input image for 8 times because its going better and better after every filtration. This is the output when after I run median filter for 8 times:


Input and output image after calling median filter 8 times