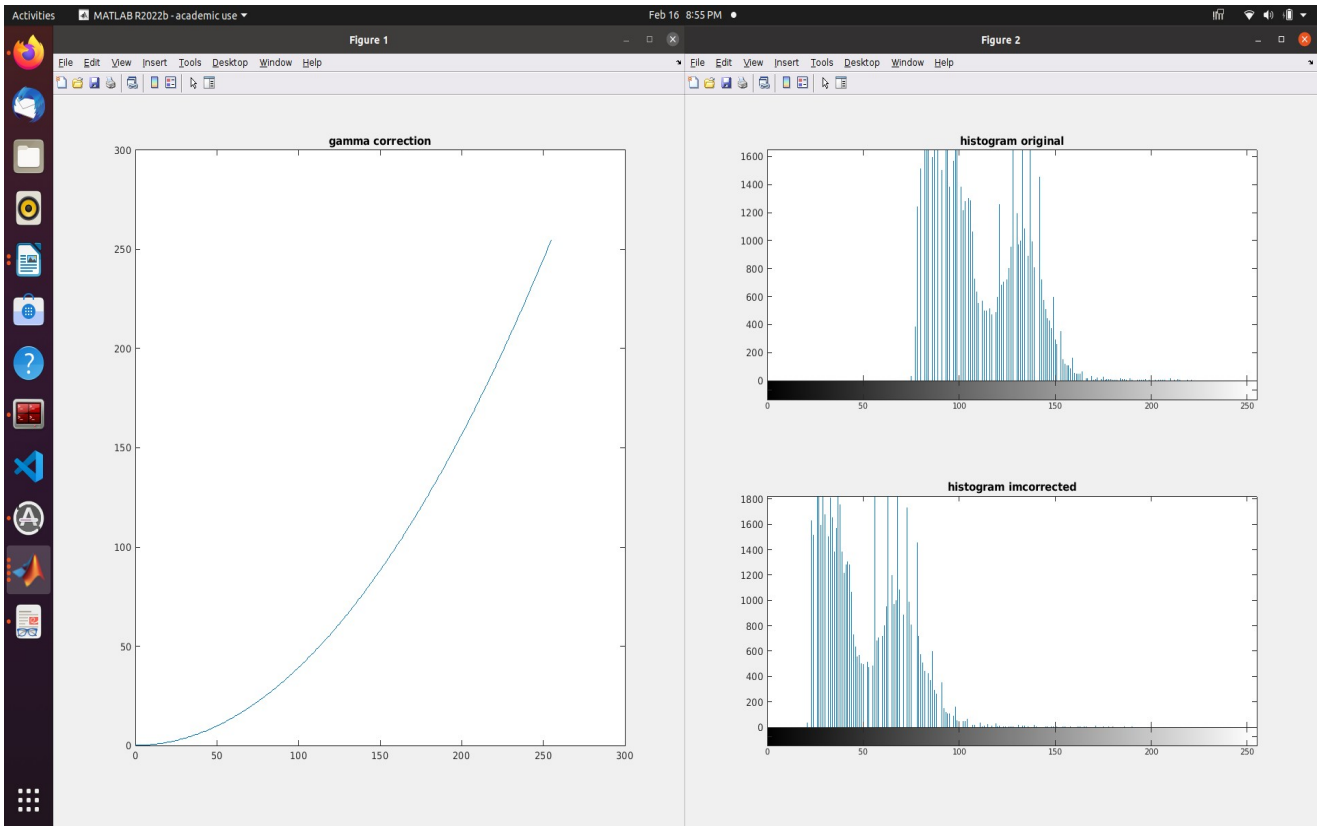
Assignment 1
ECE 620
Vikram Shahapur
14597876

Answer 1)

Code:

```
clc
clear all
close all

%Read the image
pout = imread("pout.tif");

%Get the input for the Gamma value
prompt = ("Enter Gamma value ");
gamma = input(prompt);

%Transformation function
u = 0:255;
v = 255*(u/255).^gamma;

%Plotting the graph
figure; plot(u,v); title('gamma correction');

%Implementing the transform function
imcorrected = v(pout);

%Convert the image to 8-bit integer format
imcorrected = uint8(imcorrected);

%Plot the pixel histogram to see the difference
figure;
subplot(2,1,1); imhist(pout); title('histogram original');
subplot(2,1,2); imhist(imcorrected); title('histogram imcorrected');

%Compare the images before and after contrast enhancement
figure;
montage({pout,imcorrected},"Size",[1 2]);
```

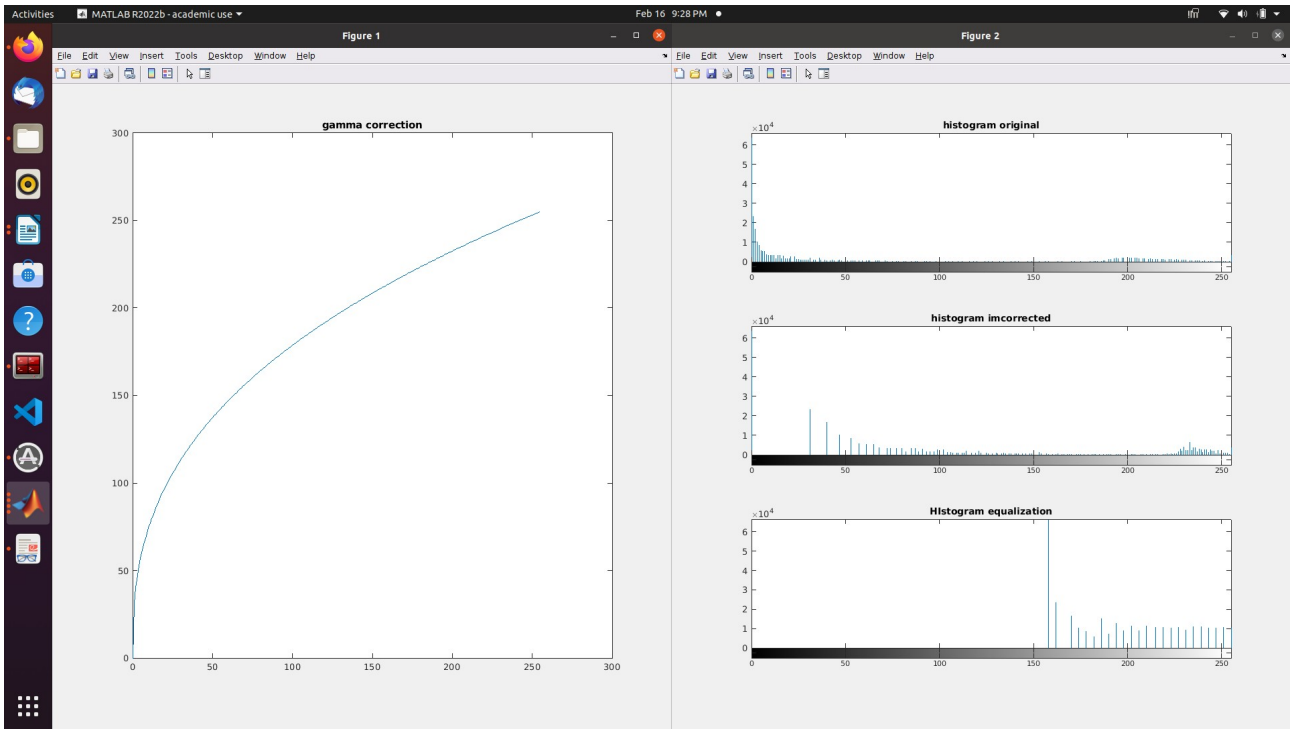
Gamma value: 1.9

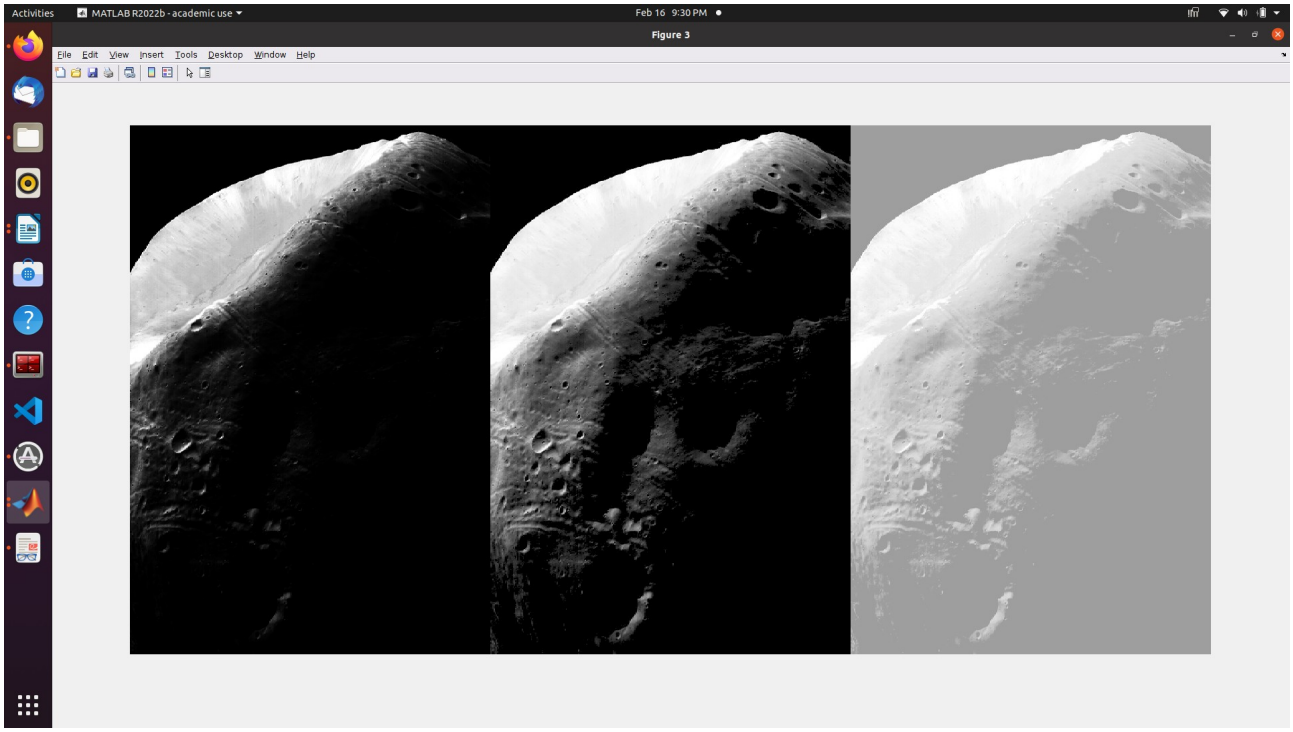


If gamma value is greater than 1, the image become darker and darker. The pixel histogram shifts toward the left most side (0 pixel side).
If the gamma value is equal to 1, there is no change in the image. The pixel histogram is same as original image.
If the gamma value is less than 1, the image becomes lighter and lighter. The pixel histogram values shift towards the right side. (255 pixel side).

Gamma value: 0.4

```
clc
clear all
close all

%Read the image
MP = imread("MoonPhobos.tif");

%Get the input for the Gamma value
prompt = ("Enter Gamma value ");
gamma = input(prompt);

%Transformation function
u = 0:255;
v = 255*(u/255).^gamma;

%Plotting the graph
figure; plot(u,v); title('gamma correction');

%Implementing the transform function
imcorrected = v(MP+1);

%Convert the image to 8-bit integer format
imcorrected = uint8(imcorrected);

%Histogram equalization
HE = histeq(MP);

%Plot the pixel histogram to see the difference
figure;
subplot(3,1,1); imhist(MP); title('histogram original');
subplot(3,1,2); imhist(imcorrected); title('histogram imcorrected');
subplot(3,1,3); imhist(HE); title("HIstogram equalization");

%Compare the images before and after contrast enhancement
figure;
montage({MP,imcorrected,HE},"Size",[1 3]);
```



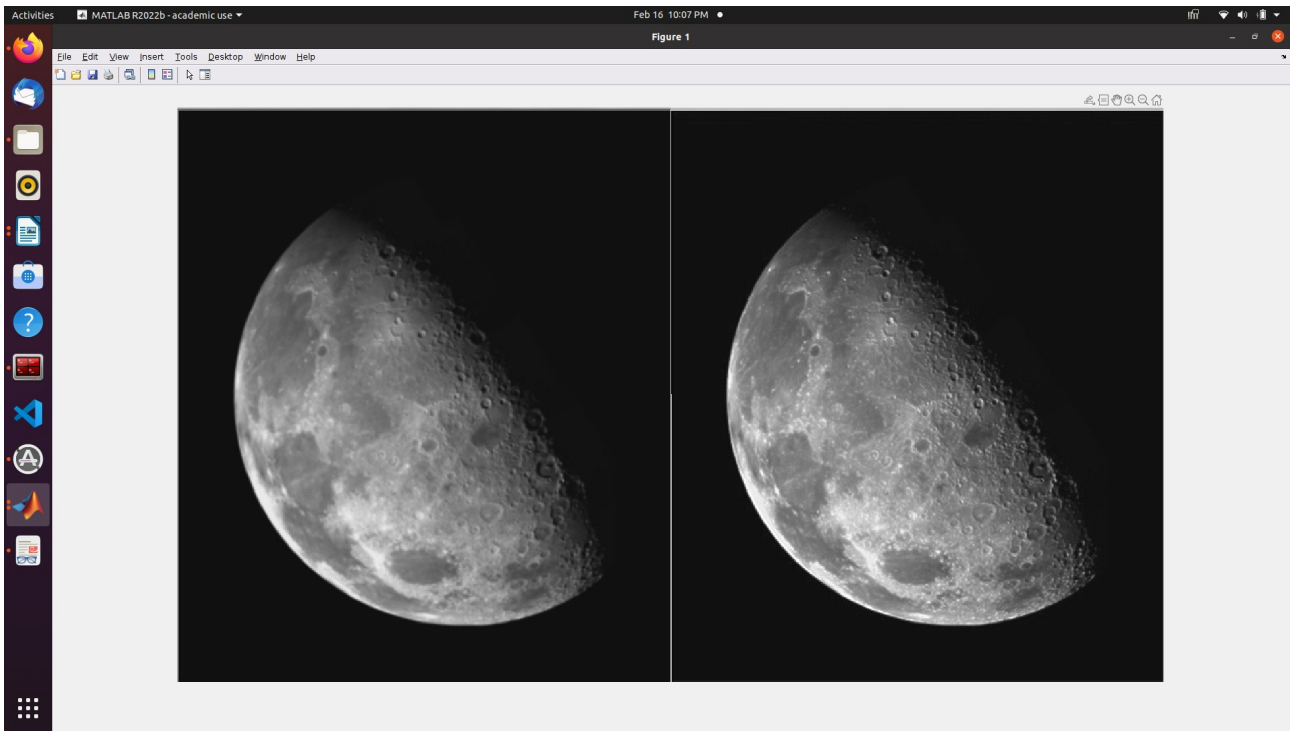


Answer 2)

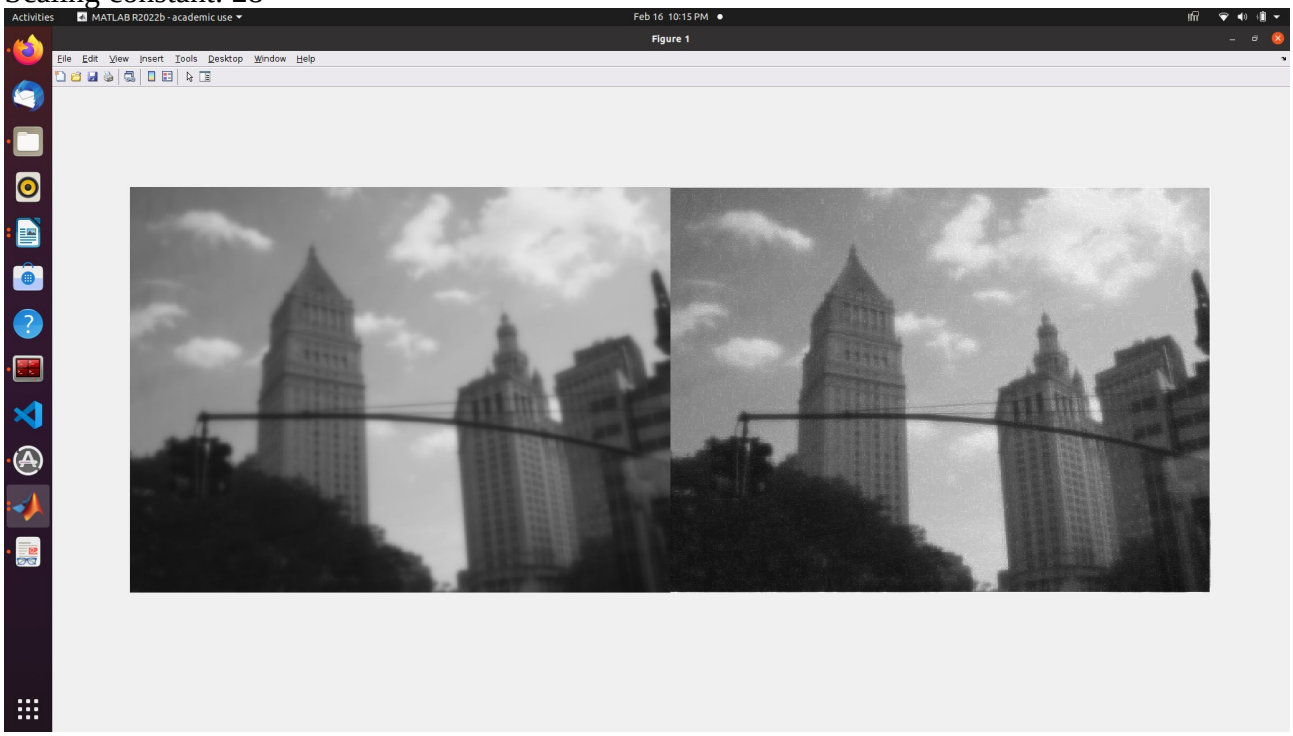
Code:

```
clc
close all

%Reading the image
shar = imread("moon.tiff");

%Defining the laplace filter
lap = [0 -0.25 0; -0.25 1 -0.25; 0 -0.25 0];

%Getting the value of scaling constant
prompt = ("Enter the scaling constant ");
const = input(prompt);

%Convolve the image "shar" using laplacian mask
c = conv2(shar,lap,"same");

%Scaling
scale = (const*c);

%Converting the image to 8-bit integer format
n1 = uint8(scale);

%Adding back to the original image to get the sharpened image
a = shar+n1 ;

%Compare the images
figure;
montage({shar,abs(a)},"Size",[1 2]);
```

Scaling constant: 3.8



Scaling constant: 28



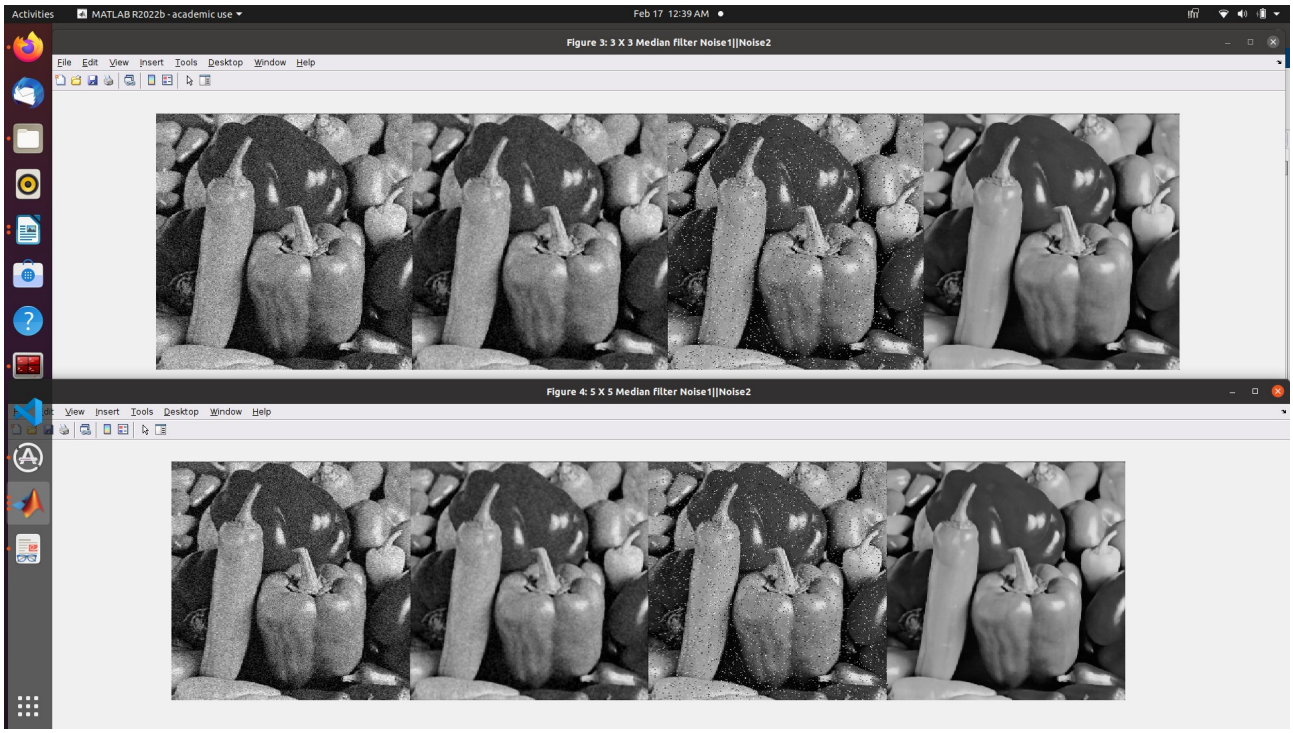
No, it is not possible to recover the original in-focus image through this method. As we increase the scale, the noise is being added to the image. So the image becomes noisy. This is the downside of the sharpening of the image.

Answer 3)

Code:

```
clc
close all

%Read the image
In = imread("peppersNoise1.tiff");
In2 = imread("peppersNoise2.tiff");

%Denoising images using Averaged filter with 3 x 3 pixel window size
avIn3 = filter2(fspecial('average',3),In)/255;
avIn23 = filter2(fspecial("average",3),In2)/255;

%Denoising images using Averaged filter with 5 x 5 pixel window size
avIn5 = filter2(fspecial('average',5),In)/255;
avIn25 = filter2(fspecial("average",5),In2)/255;

%Denoising images using Median filter with 3 x 3 pixel window size
MIn3 = medfilt2(In,[3 3]);
MIn23 = medfilt2(In2, [3 3]);

%Denoising images using Median filter with 5 x 5 pixel window size
MIn5 = medfilt2(In,[5 5]);
MIn25 = medfilt2(In2, [5 5]);

%Comparing the images
figure('Name','3 X 3 Averaged filter Noise1||Noise2');
montage({In,avIn3,In2,avIn23},"Size",[1 4]);

figure('Name','5 X 5 Averaged filter Noise1||Noise2');
montage({In,avIn5,In2,avIn25},"Size",[1 4]);

figure('Name','3 X 3 Median filter Noise1||Noise2');
montage({In,MIn3,In2,MIn23},"Size",[1 4]);

figure('Name','5 X 5 Median filter Noise1||Noise2');
montage({In,MIn5,In2,MIn25},"Size",[1 4]);
```

We can see from the image that denoising by averaged filter is working well when the window size is 5 by 5 pixel. But you can also see the disadvantage that image is being blurred after the noise being removed. However, in window size 3 by 3 pixel, the image is not blurred but the noise is still there.



In this image we can se that Median filter works so muc better than Averaged filter, especially in window size of 5 by 5 pixel, the image has been denoised almost perfectly. We dont see the effect of blurring in Median filter like that of averaged. Median filter with window size 3 by 3 pixel worked much better than averaged filter of 5 by 5 pixel.

Code:
Threshold value: 0.5 for averaged filter.
Threshold value: 2.5 for Median filter.

```
clc
clear all
close all

%Read the image
I = imread("peppersNoise1.tiff");

%Getting the input for threshold
prompt=("Enter threshold ");
threshold = input(prompt);

%Applying the averaged filter and median filter first
AVI = filter2(fspecial('average',3),I)/255;
MEI = medfilt2(I,[3 3])/255;
MEId = medfilt2(I,[3 3]);

%Sobel filter for x-direction and y-direction
Sx = [-1 0 1; -2 0 2; -1 0 1];
Sy = [-1 -2 -1; 0 0 0; 1 2 1];

%Covolve the image using Sobel filters along x-direction and y-direction
X = conv2(AVI,Sx,'same');
Y = conv2(AVI,Sy,'same');
A = conv2(MEI,Sx,'same');
B = conv2(MEI,Sy,'same');

%Magnitude of the gradient
M1 = sqrt(X.*X + Y.*Y);
M2 = sqrt(A.*A + B.*B);

%Comparing the magnitude of the gradient with threshold
EdgeI = uint8((M1 > threshold)*255);
EdgeI2 = uint8((M2 > threshold)*255);

I1 = repmat(EdgeI, [1 1 3]);
I2 = repmat(EdgeI2, [1 1 3]);

%Result
figure;
montage({I,AVI,I1},"Size",[1 3]);
figure;
montage({I,MEId,I2},"Size",[1 3]);
```

Averaged filter

Median filter

Median filter has better edge preserving properties.
The difference is, in Median filter the edgemaps are well preserved and defined compared to that of Averaged filter.

Thank you !!