Jonah Rubino, Yoni Carver | ECES 435 - Dr. Matthew Stamm
Assignment #1 | 1.24.2020

Part 1

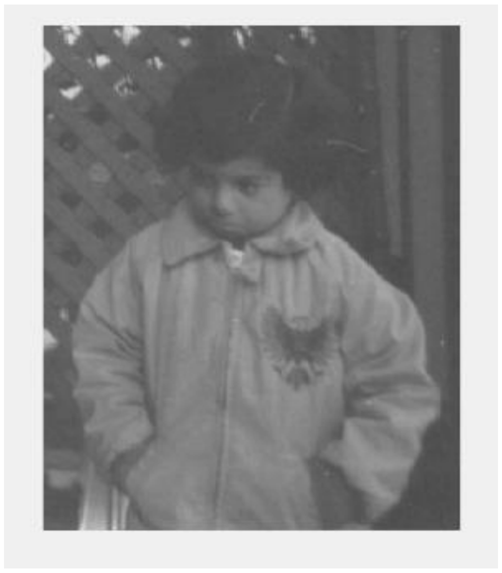
Figure 1: Image read only through imread command

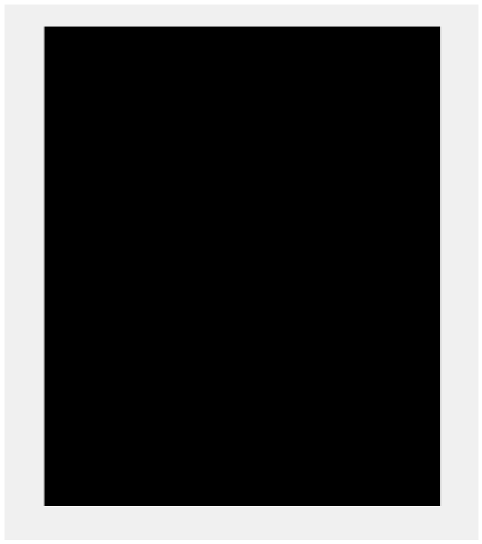
Figure 2: Gamma corrected image with γ = 2

When the image is subjected to a gamma correction where the gamma value is set to 2, all of the pixels increase in value which means they will get darker. For this image, a gamma value of two turns the image completely black.

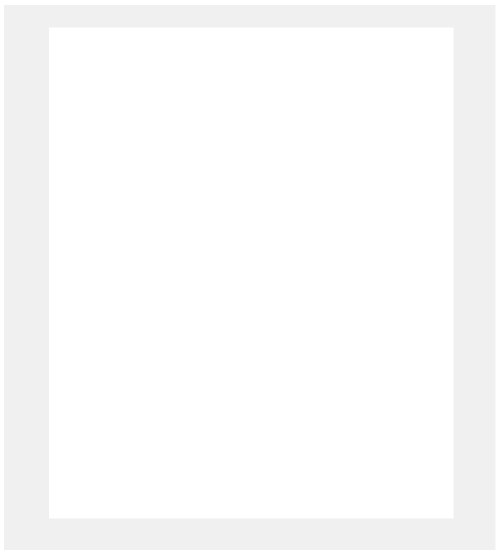
Figure 3: Gamma corrected image with γ = 2

When gamma is set to a number between 0 and 1, it causes the value of each pixel to decrease as they are being raised to a decimal exponent. When this image is subjected to a gamma value of .75, the values become so low they only show white.

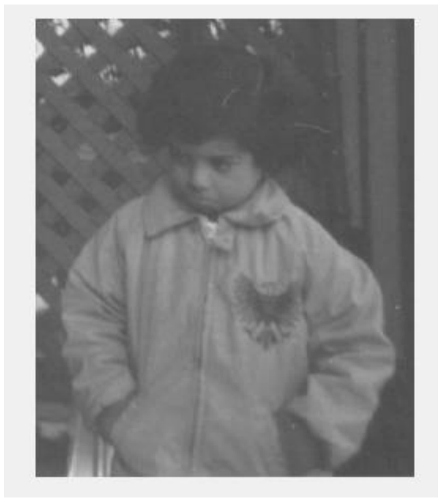
Figure 4: Gamma corrected image with γ = 1

When the image is subjected to a gamma equal to 1, it doesn't enhance the image in any way. The image seen in figure 4 is the same as the image seen in figure 1.

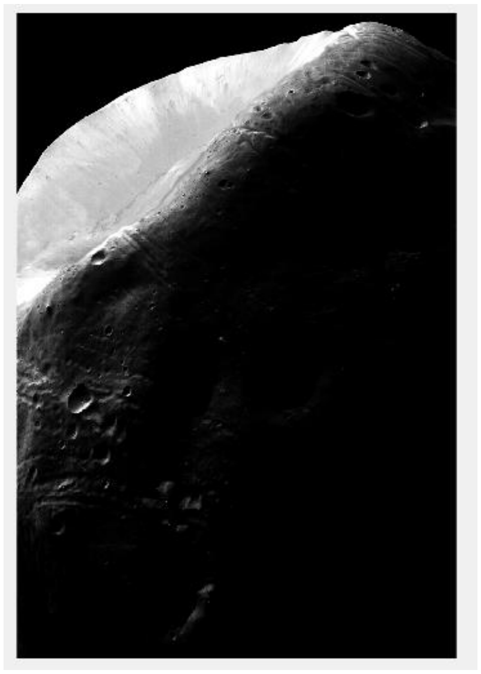

Figure 5: Image read through MATLAB imread command

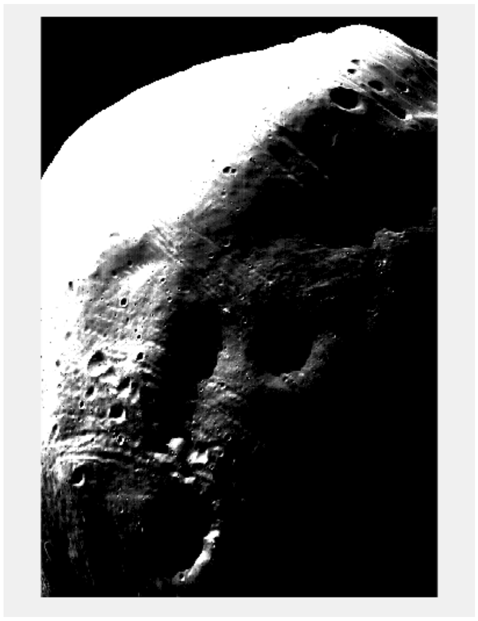

Figure 6: Gamma corrected image with γ = .72

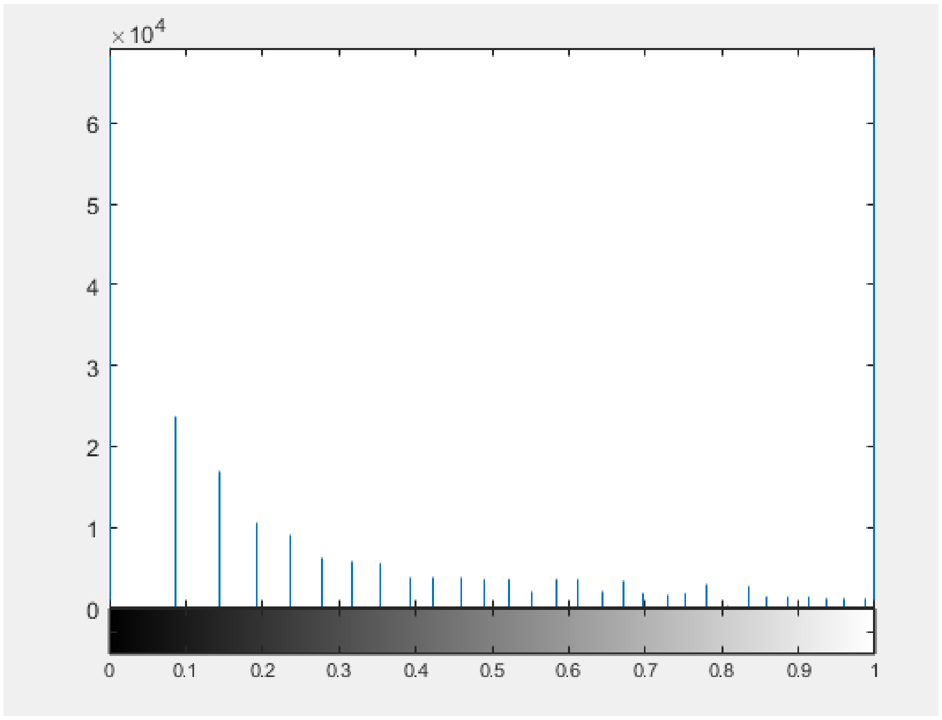

Figure 7: Pixel value histogram for gamma corrected image

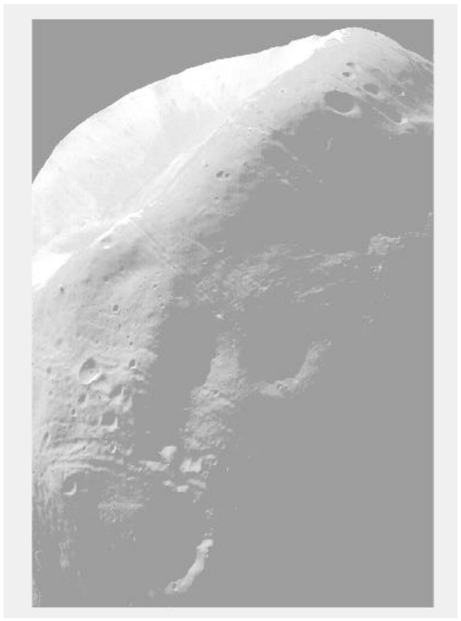

Figure 8: Image read through the MATLAB histeq command

When comparing figure 8 to figure 6, the image shown by the histeq command is better to enhance the image. More detail can be seen in this image in all areas. The image enhanced by gamma correction does show more detail in the lower part of the moon than the original image. However, the upper part has too light of pixels that become lighter with a less than one gamma. However, the bottom area of the moon is too dark to use any gamma greater than 1. Gamma being set to .72 shows the most detail it can, with showing a bit of the upper half of the image. Since this code is rudimentary, it is recommended to use the built in MATLAB command for a more detailed image.

Part 2

moon.tiff

After trying several different values for α, the value that appears to work best is when α is equal to 5 (shown above). As you increase α after 5, the image appears to become more grainy and pixelated.

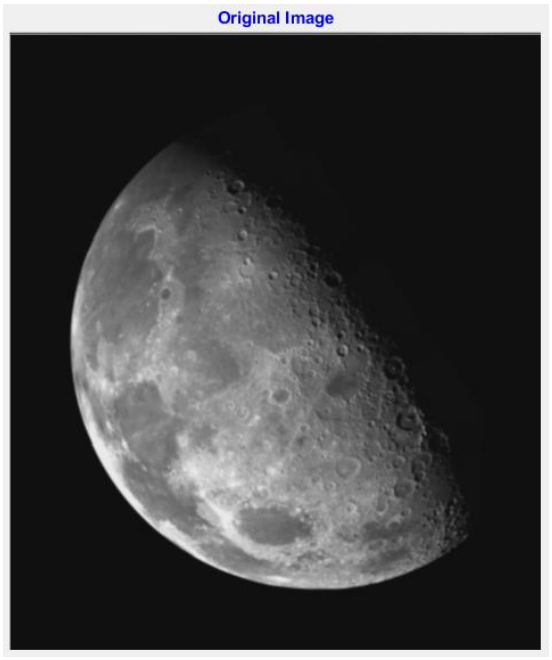

Figure 9: Original moon.tiff image

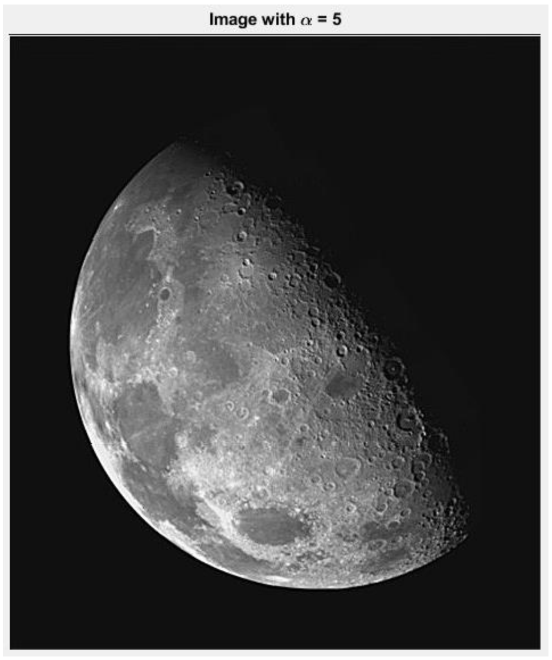

Figure 10: Sharpened image (α = 5)

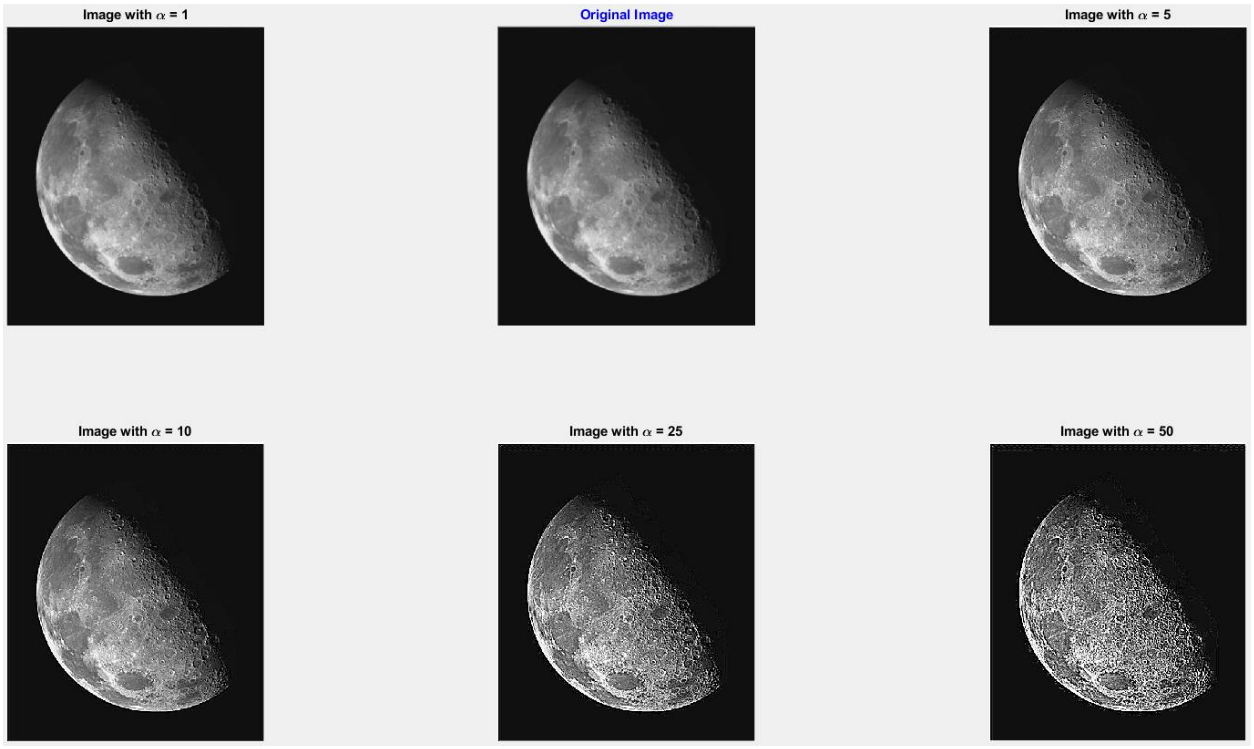

Figure 11: moon.tiff - Original image and α = 1, 5, 10, 25, 50

outoffocus.tiff

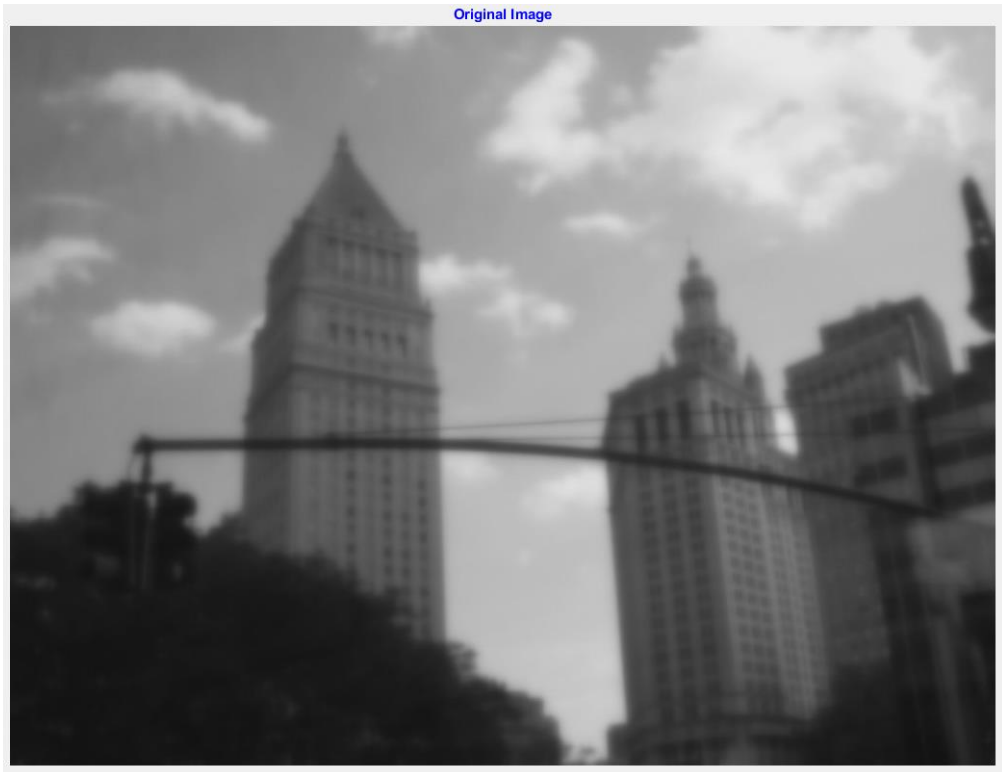

Figure 12: Original outoffocus.tiff image

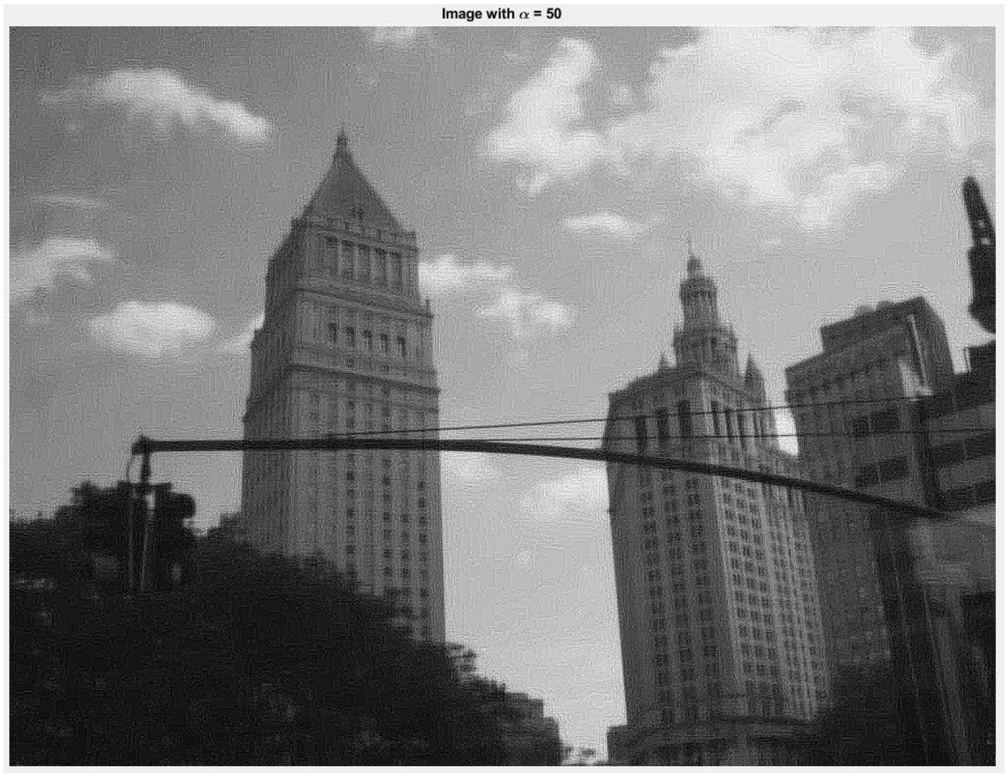

Figure 13: Sharpened image (α = 50)

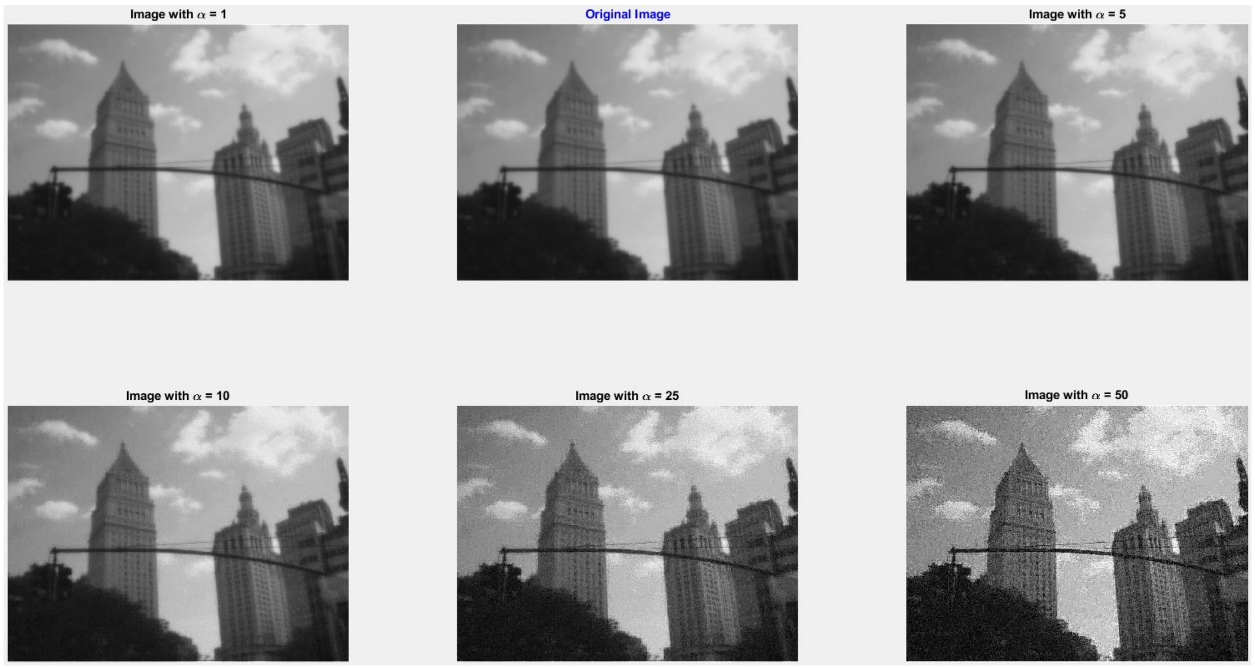

Figure 14: outoffocus.tiff - Original image and α = 1, 5, 10, 25, 50

It does not appear that you can recover the original in-focus image. You may appear to recover the image if you sharpen it and make the image smaller but the eyes are correcting for the blurriness.

There are unintended artifacts, such as contouring, that show up as you increase the value of α (shown below).

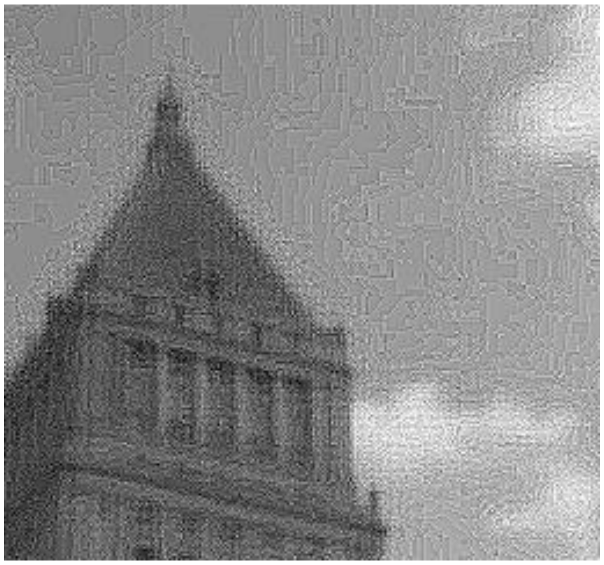

Figure 15: α = 50

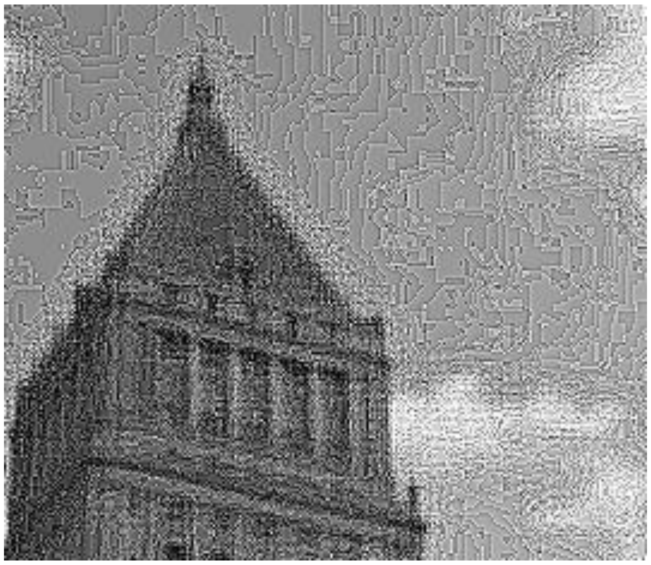

Figure 16: α = 100

Part 3

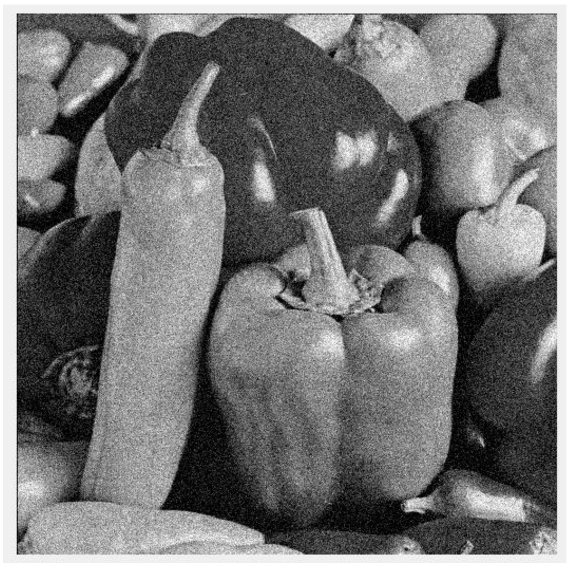

Figure 17: Scanned image of "peppersNoise1"

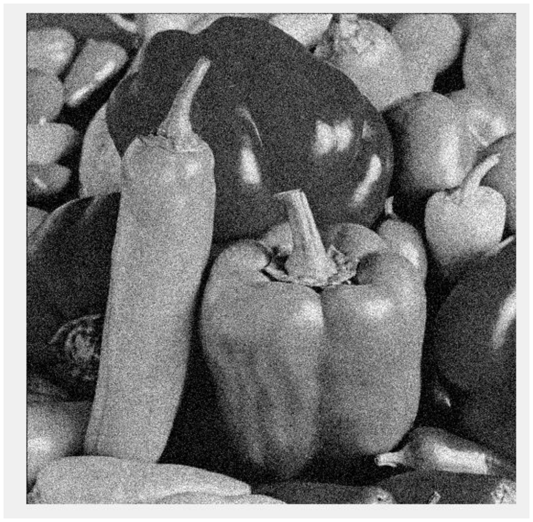

Figure 18: Scanned image of "peppersNoise2"

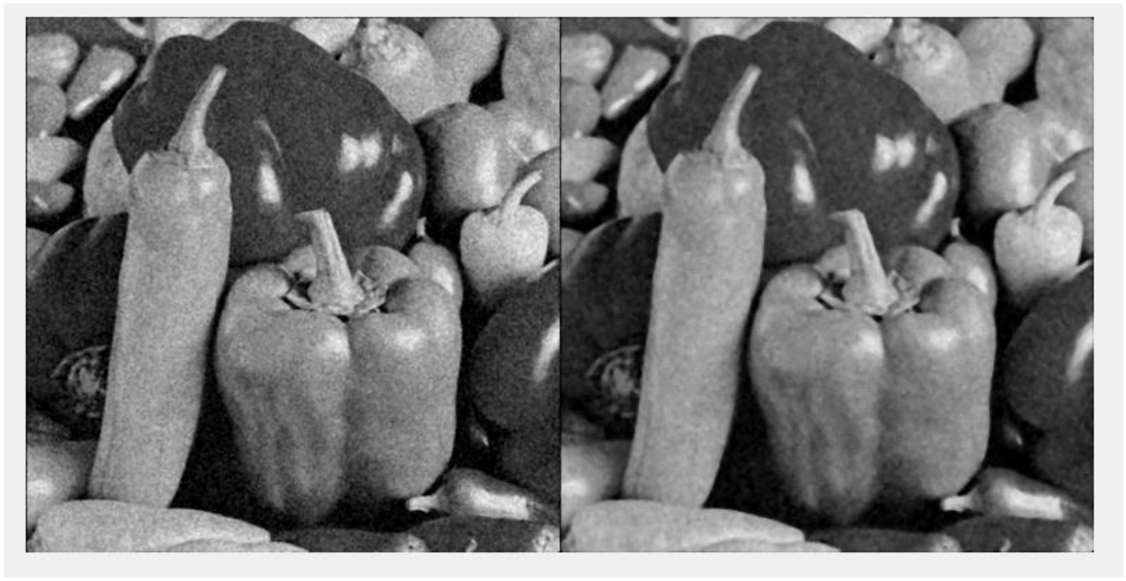

Figure 17: "peppersNoise1" run through median filter
Left: 3x3 Right: 5x5

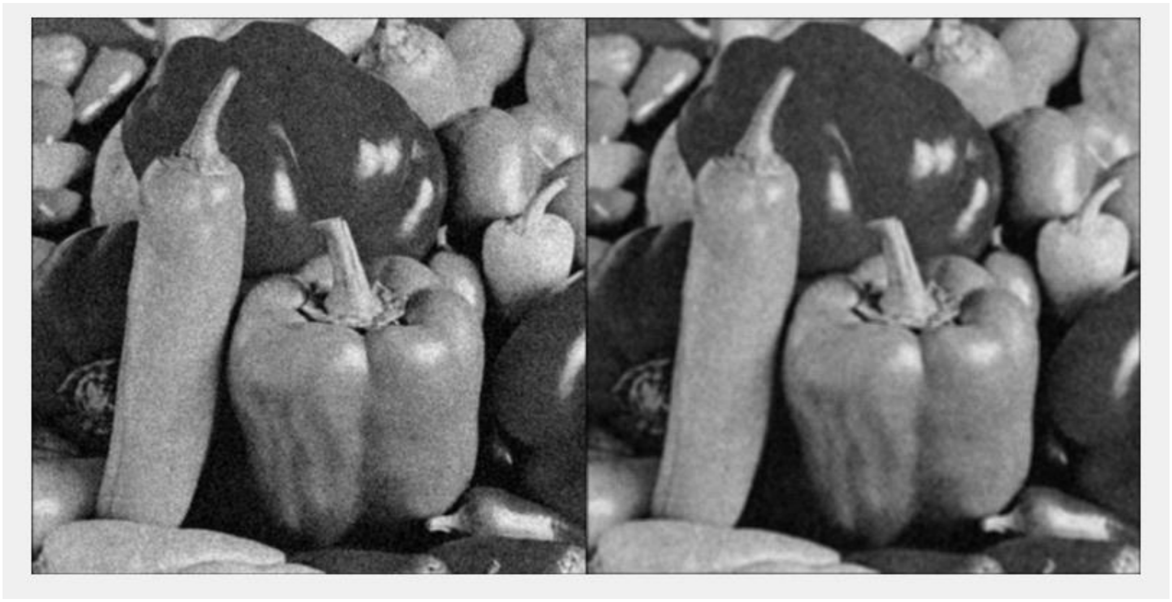

Figure 18: "peppersNoise1" run through average filter
Left: 3x3 Right: 5x5

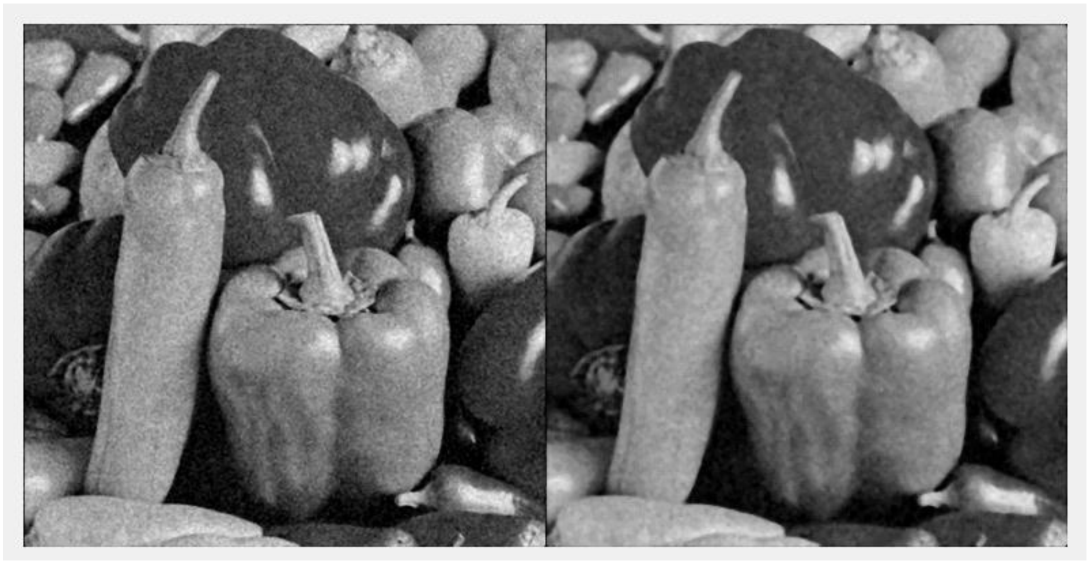

Figure 19: "peppersNoise2 " run through median filter
Left: 3x3 Right: 5x5

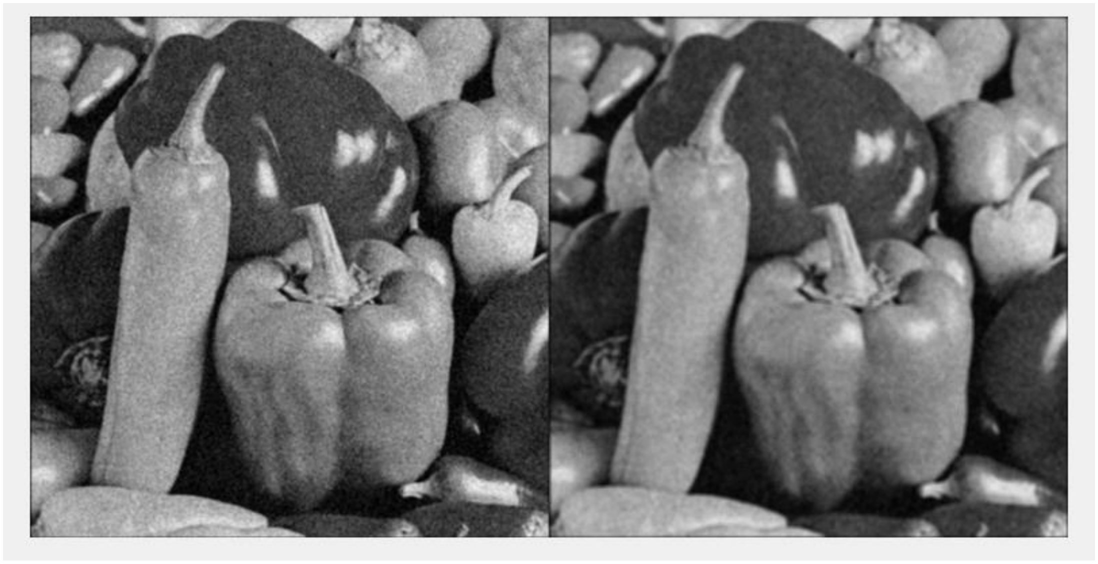

Figure 20: "peppersNoise2" run through average filter
Left: 3x3 Right: 5x5

The pros of using a median, is that is excellent at removing noise that is an extreme outlier like "salt and pepper noise". This is because a median filter works by ignoring the noise outright. When encountering "salt and pepper noise", a median filter can produce an image with little degradation. However, when it comes to more averaging noise, the median filter will not produce an image of the best quality. This is where an averaging filter succeeds in being better. Since an averaging filter folds the noise into itself, so it's sum average becomes 0., an averaging filter will do better with an image with more scattered noise. However, images with high peaking noise, will not have a good quality if an averaging filter is used on it

The advantage of changing filter size is changing the resolution of the image. The smaller the filter size the clearer the images appear. In both instances of using the larger sized filter, the image comes out blurrier. From these pictures, it seems a smaller filter size should be used for a clearer picture.

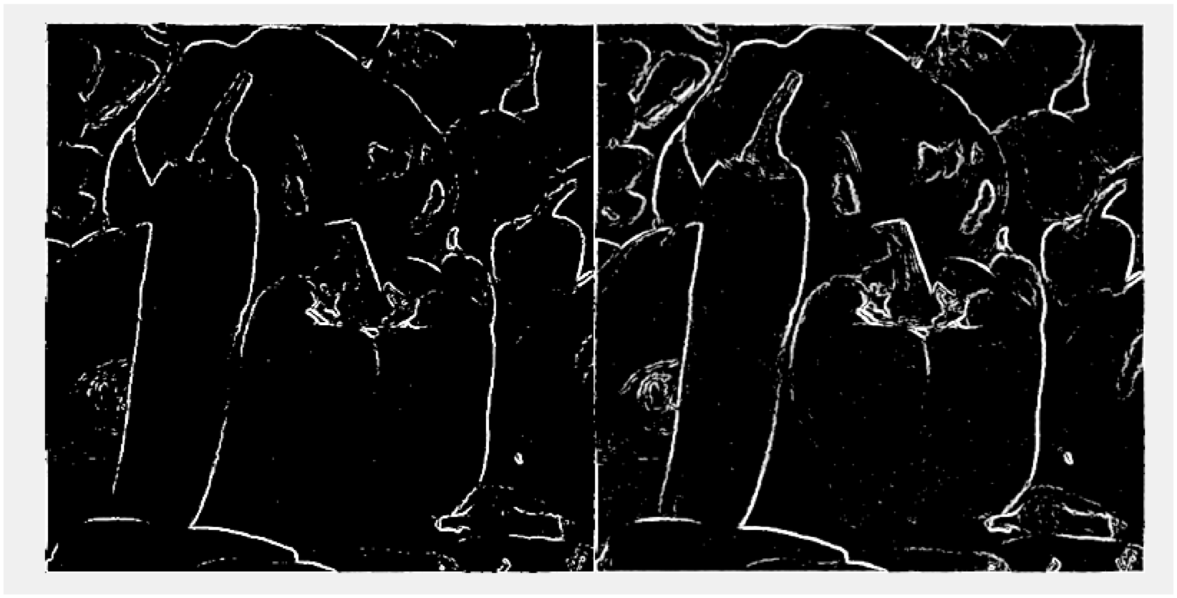

Figure 21: Edge map for "peppersNoise1" run through filters with a threshold of .8
Left: Median Filter Right: Averaging Filter

The median filter is better at showing the edges of the image, but it also shows more noise than the average filter. But, in terms of edge preserving properties, the median filter is better.

Jonah Rubino, Yoni Carver | ECES 435 - Dr. Matthew Stamm
Assignment #1 | 1.24.2020

Part 1 - Code

```
function scan = gamcor(x,gamma) %x is the image, and gamma is the parameter
y = im2double(x);
y =  y(:,:,1); %ensures the picture only has 2-D values
imcor = y; %

for i=1:size(y,1) %iterates over the rows
    for j = 1:size(y,2) %iterates over the columns
        pixel = y(i,j); %has the pixel value
        newpix = 255*(pixel/255).^(gamma); %performs the gamma correction
        imcor(i,j) = newpix; %sets pixel at the same position to the new value
    end
end
scan = imcor; %produces filtered image
```

Part 2 - Code

```
% load images
moon = imread('moon.tiff');
outoffocus = imread('outoffocus.tif');

alpha = 10;

% run the sharp_image function on both images
sharp_moon = sharpen(moon, alpha);
sharp_outoffocus = sharpen(outoffocus, alpha);

function sharp_image = sharpen(image, alpha)
    %{
    sharpened image equation: f_s(x,y) = f(x,y) + ?g(x,y)
        f(x,y) = original image
        ? = user-specified scaling constant
        g(x,y) = high frequency content (found using the Laplacian filter)
    %}

    % laplacian filter to obtain the high frequency data from the image
    laplacian_filter = [0 -0.25 0; -0.25 1 -0.25; 0 -0.25 0];

    % filter the original image with ?*g(x,y)
    image_high_freq = filter2(alpha * laplacian_filter, image);

    % sum the high frequency component with the original image to obtain the
    % sharpened image
    sharp_image = uint8(double(image) + double(image_high_freq));
end
```

Part 3 - Code

```
pn1 = imread('peppersNoise1.tiff'); %Reads in the image

pn1 = im2double(pn1); %Converts image to a double to filter

pn1med3 = medfilt2(pn1,[3 3]); %Performs a median filter

h1 = ones(3,3)/9; %Set's the matrix up for the averaging filter

pn1avg3 = imfilter(pn1,h1); %Runs the image through an averaging filter

sobX = [-1 0 1 ; -2 0 2; -1 0 1]; % X Axis matrix for the Sobel filter
sobY = [-1 -2 -1 ; 0 0 0 ; 1 2 1] ; %Y Axis matrix for the Sobel Filter

med3x = conv2(pn1med3,sobX,'same'); %convolves the image with the X Sobel Matrix
med3y = conv2(pn1med3,sobY,'same'); %convolves the image with the Y Sobel Matrix

medSobel = sqrt(med3x .^2 + med3y .^2); %Computes the magnitude of the gradient

thresh = medSobel <.8; %determines which of the pixels are below the threshold
medSobel(thresh) = 0; %set's all values of the pixels below threshold to 0, creates
the edges

avg3x = conv2(pn1avg3,sobX,'same'); %convolves the image with the X Sobel Matrix
avg3y = conv2(pn1avg3,sobY,'same'); %convolves the image with the Y Sobel Matrix

avgSobel = sqrt(avg3x .^2 + avg3y .^2); %Computes the magnitude of the gradient

thresh = avgSobel <.8; %determines which of the pixels are below the threshold
avgSobel(thresh) = 0; %set's all values of the pixels below threshold to 0, creates
the edges

montage({medSobel,avgSobel}) %Plots the images next to one another
```