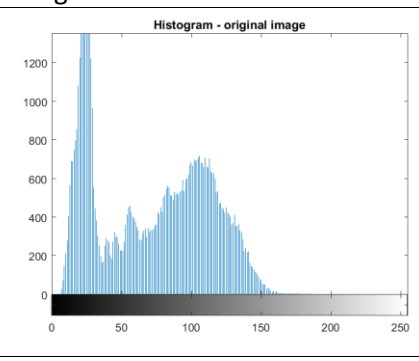
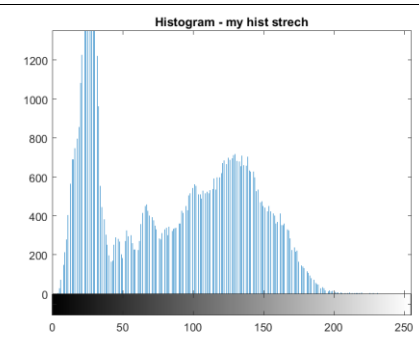
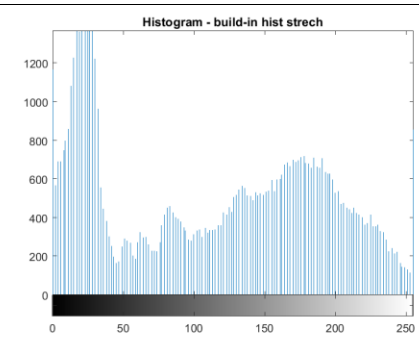
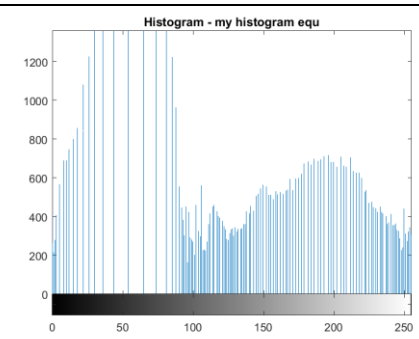
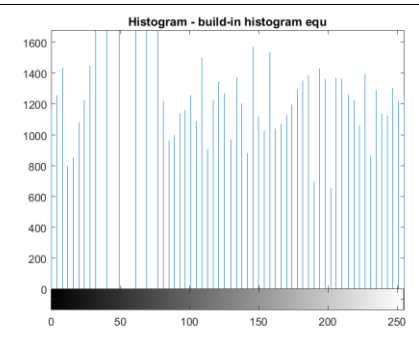
Q1.

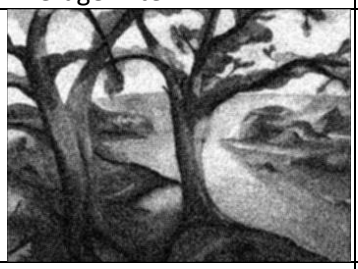
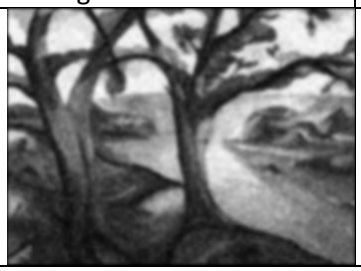
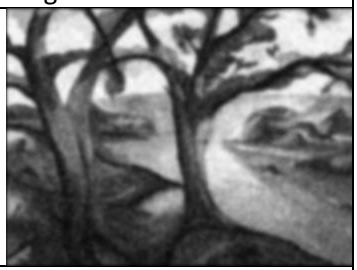
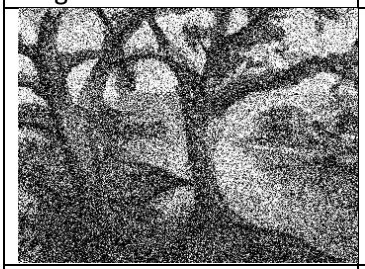
	Image	Histogram
Original		
Stretch (Own)		
Stretch(Build-in)		
Equalization(Own)		
Equalization(Build-in)		

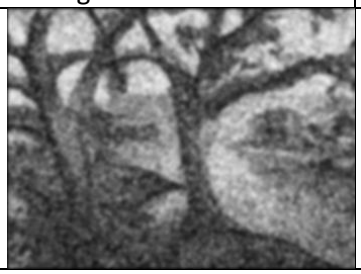
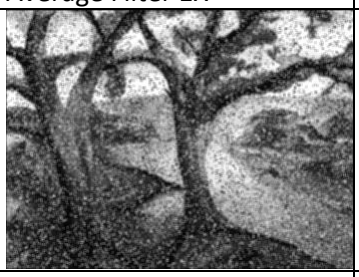
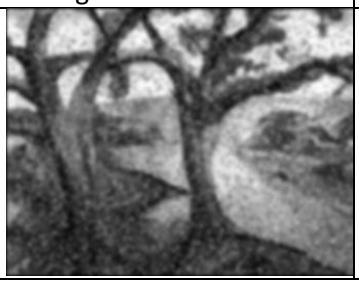
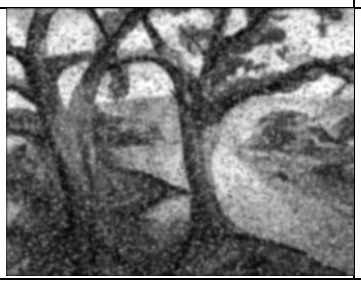
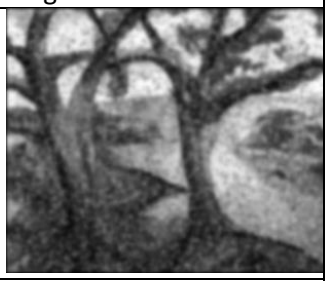
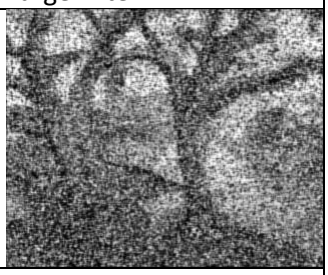
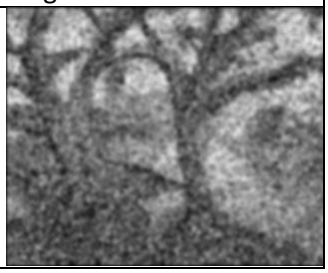
Q1 Conclusion

In this experiment, I found that by horizontal stretching the histogram, the contrast of the is enhanced since by stretching it means trying to make color intensity more extreme. I noticed there's a difference between my implementation of stretch compared to the Matlab build-in one. The build-in one seems to give a result image with higher contrast. I checked the implementation of the build-in one and found that besides stretching, it also makes 1% of the pixel is saturated at low and high intensities. Since there are some pixels that are already very high intensity (Perhaps the sun area) in the original image, stretching them to 255 won't be effective. This explains why my image looks darker to the build-in one. On the other hand, the histogram equalization looks very similar. By using more bins in my implementation, the resulting image histogram look more dense compared to the build in one.

Q2.

In this experiment, I found that the median filter works well on salt and pepper noised images. My guess is because it requires sorting the image density and taking the median. The outlier pixels created by the salt and pepper noise can be ignored. However, if the level of salt and pepper noise is increased, we found that the median filter no longer works so well since there are too many noisy pixels. Overall, I found that by applying multiple iterations of filters into the image, there will be more blur in the image. Increasing the filter size also make the image more blur.

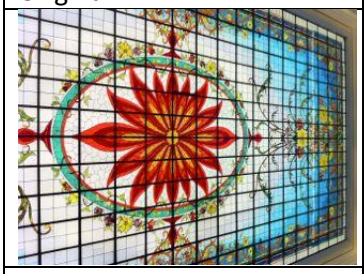
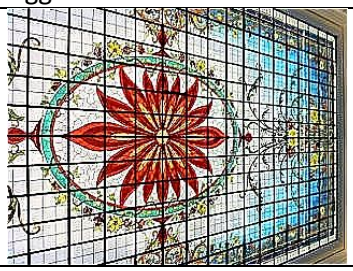
trees_var005.tif			
Original	Average Filter 1X	Small Filter 1X	Large Filter 1X
			
Midfilt2	Average Filter 5X	Small Filter 5X	Large Filer 5X
			
trees_var020.tif			
Original	Average Filter 1X	Small Filter 1X	Large Filter 1X
			

Midfilt2	Average Filter 5X	Small Filter 5X	Large Filer 5X
			
trees_salt020.tif			
Original	Average Filter 1X	Small Filter 1X	Large Filter 1X
			
Midfilt2	Average Filter 5X	Small Filter 5X	Large Filer 5X
			
trees_salt050.tif			
Original	Average Filter 1X	Small Filter 1X	Large Filter 1X
			
Midfilt2	Average Filter 5X	Small Filter 5X	Large Filer 5X
			

Q3.

In the sharpening experiment. I found that by applying the filter on the Luminance component, compared to applying the filter directly on the RGB channels, seems to create better overall result since it only manipulates the luminance density of the pixel. An example will be if I apply the (more aggressive) filter $\begin{matrix}-1 & -2 & -1\\ -2 & 14 & -2\\ -1 & -2 & -1\end{matrix}$ on RGB, there are some parts of the image showing color distortion, while apply on the luminance component does not seem to have this problem. (The edge between the red and green part.) Overall, a larger sharpening filter creates a subtler sharpening effect.

 VS 

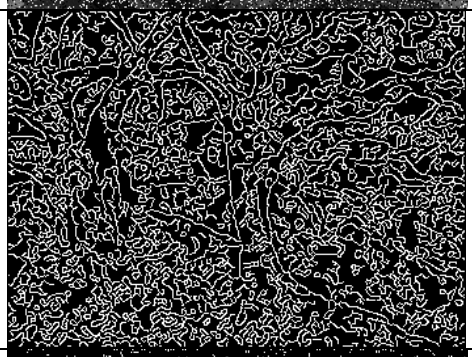
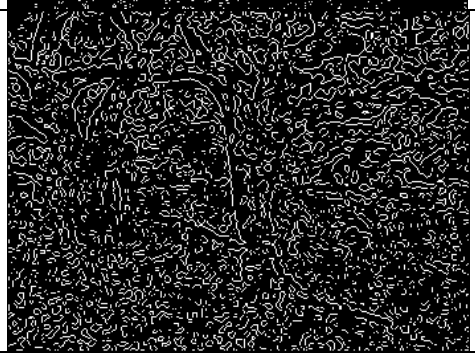
peppers.png			
Original	Small Filter RGB	Aggressive Filter RGB	Large Filter RGB
			
	Small Filter Luminance	Aggressive Filter Luminance	Large Filter Luminance
			
peppers.png			
Original	Small Filter RGB	Aggressive Filter RGB	Large Filter RGB
			
	Small Filter Luminance	Aggressive Filter Luminance	Large Filter Luminance
			

Q4

Q4 Conclusion

In this experiment, I found that by smoothing the image before extracting the edges, it really helps improve the accuracy of the edge being extracted. The image I used is the salt and pepper one and the one applied 5X larger filter.

Overall Canny detection has the best performance.

	Noisy	Smooth
Original		
Canny		
Prewitt		
Laplacian of Gaussian (LOG)		

Sobel