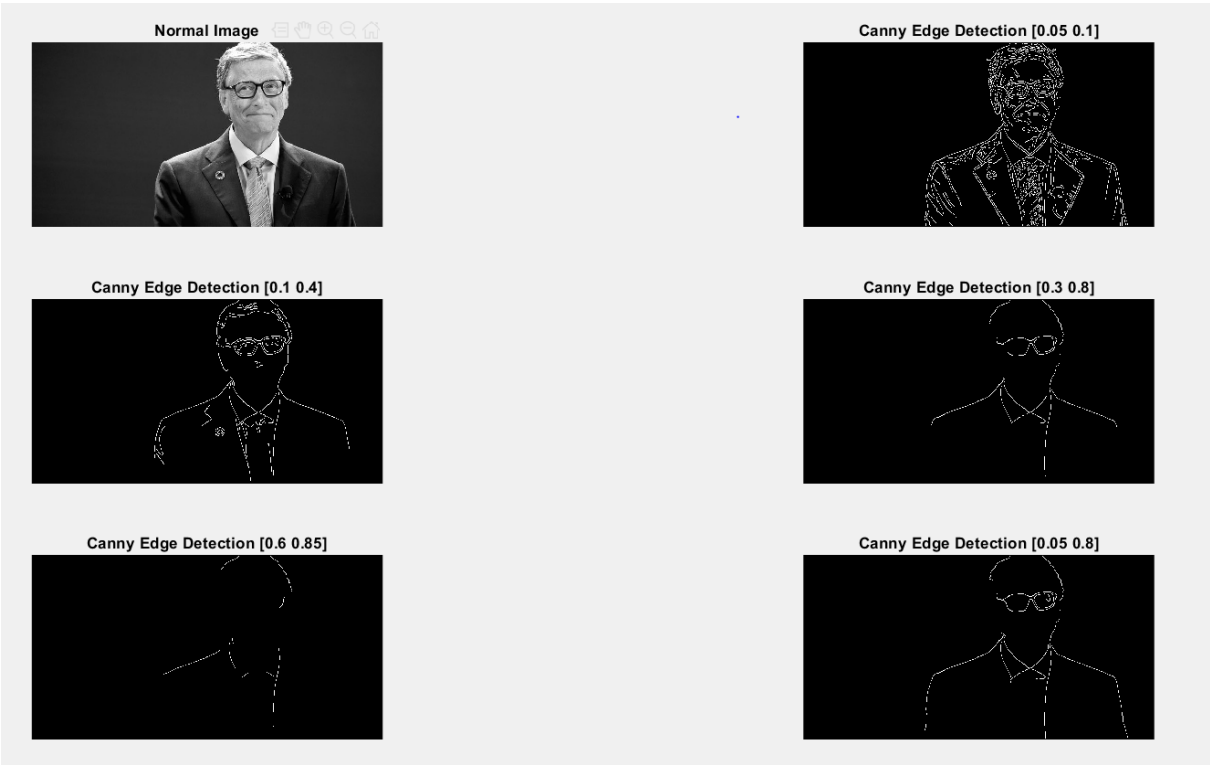
EE417

COMPUTATIONAL ASSIGNMENT #1

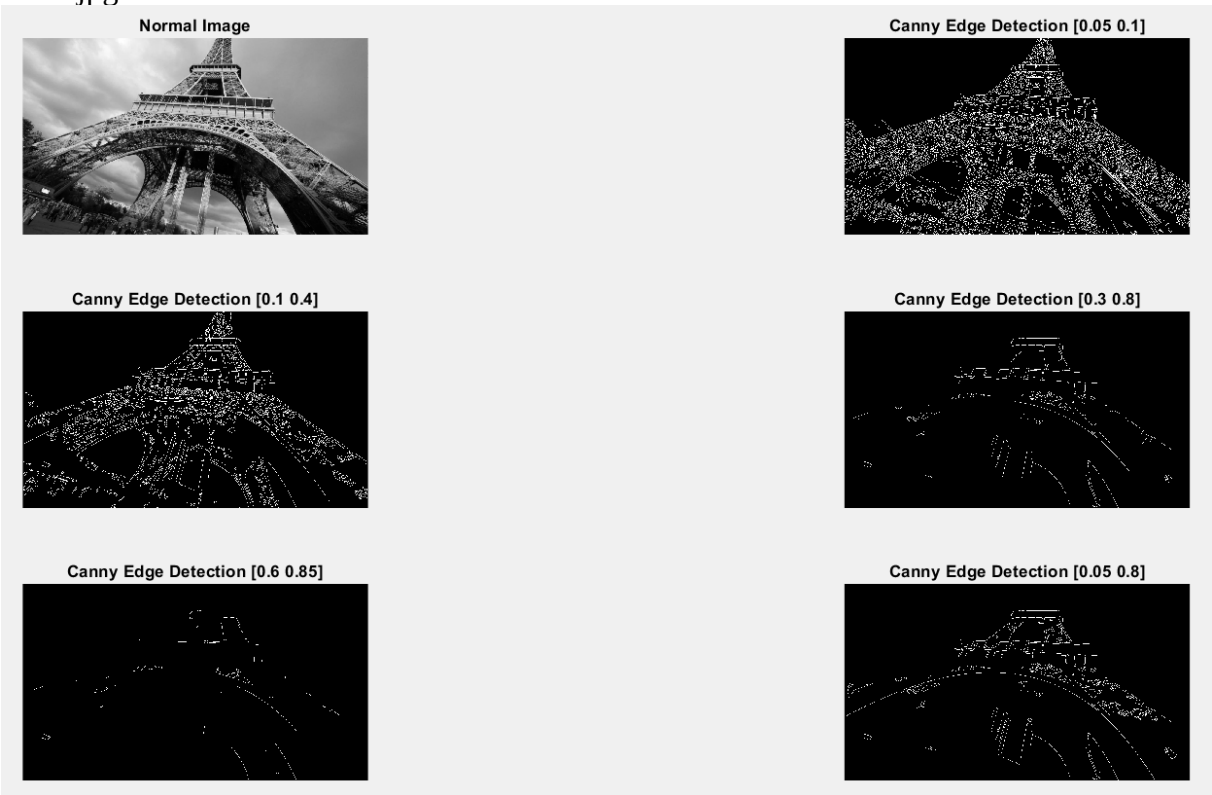
1. Canny Edge Detection

Canny method looks for local maxima of the gradient of image. The edge function calculates the gradient using the derivative of a Gaussian filter. This method uses two thresholds to detect strong and weak edges, including weak edges in the output if they are connected to strong edges. By using two thresholds, the Canny method is less likely than the other methods to be fooled by noise, and more likely to detect true weak edges. If the threshold is too high it causes more misdetections and many gap. If the threshold is too low too many (all pixels) edges

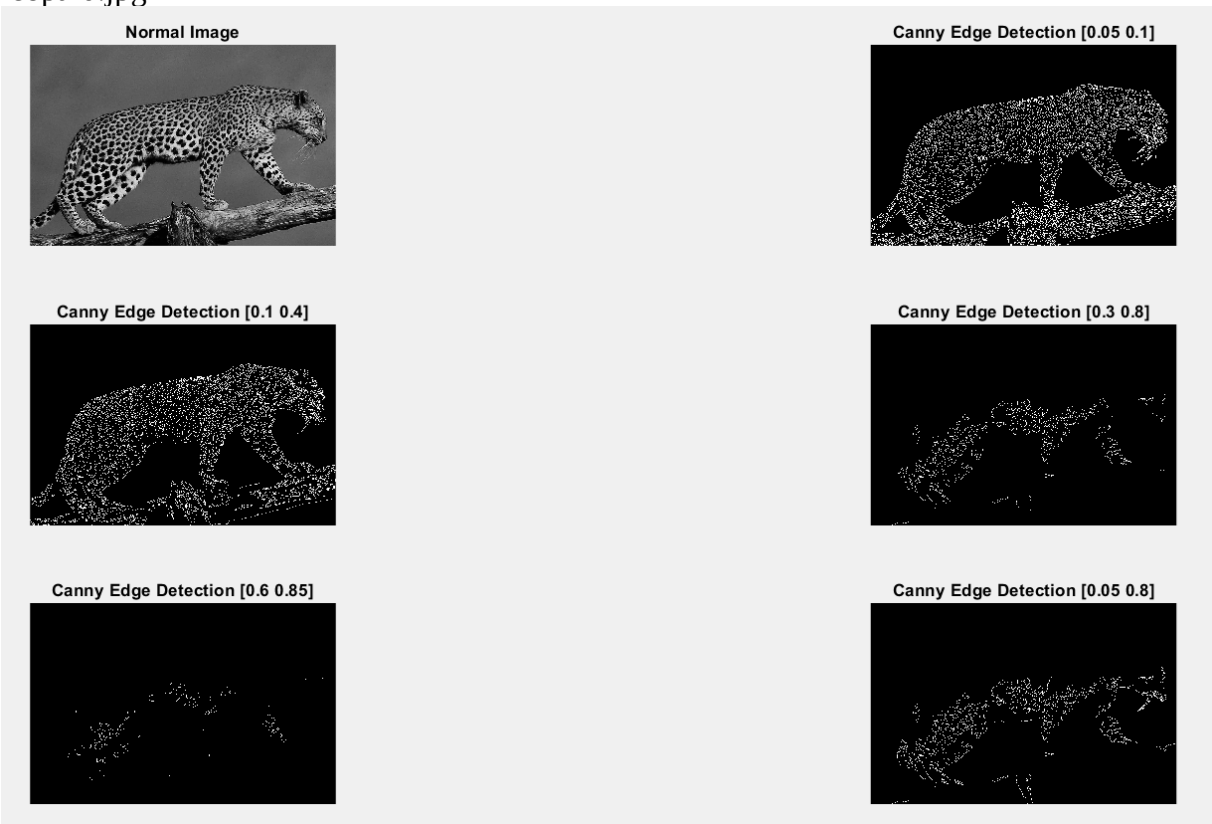


We applied Canny edge detection on Bill Gates' photo. We determined two thresholds which are as lower and upper. As can be seen above, 0.05 and 0.1 thresholds are too low that causes to show more weak edges because it disregards all edges with edge strength below the lower threshold, and preserves all edges with edge strength above the higher threshold. 0.1-0.4 values are seem to give more reliable results. It actually gives necessary edges to be considered. Other ones gives more sparse results. 0.05-0.8 can also be considered as a good one.

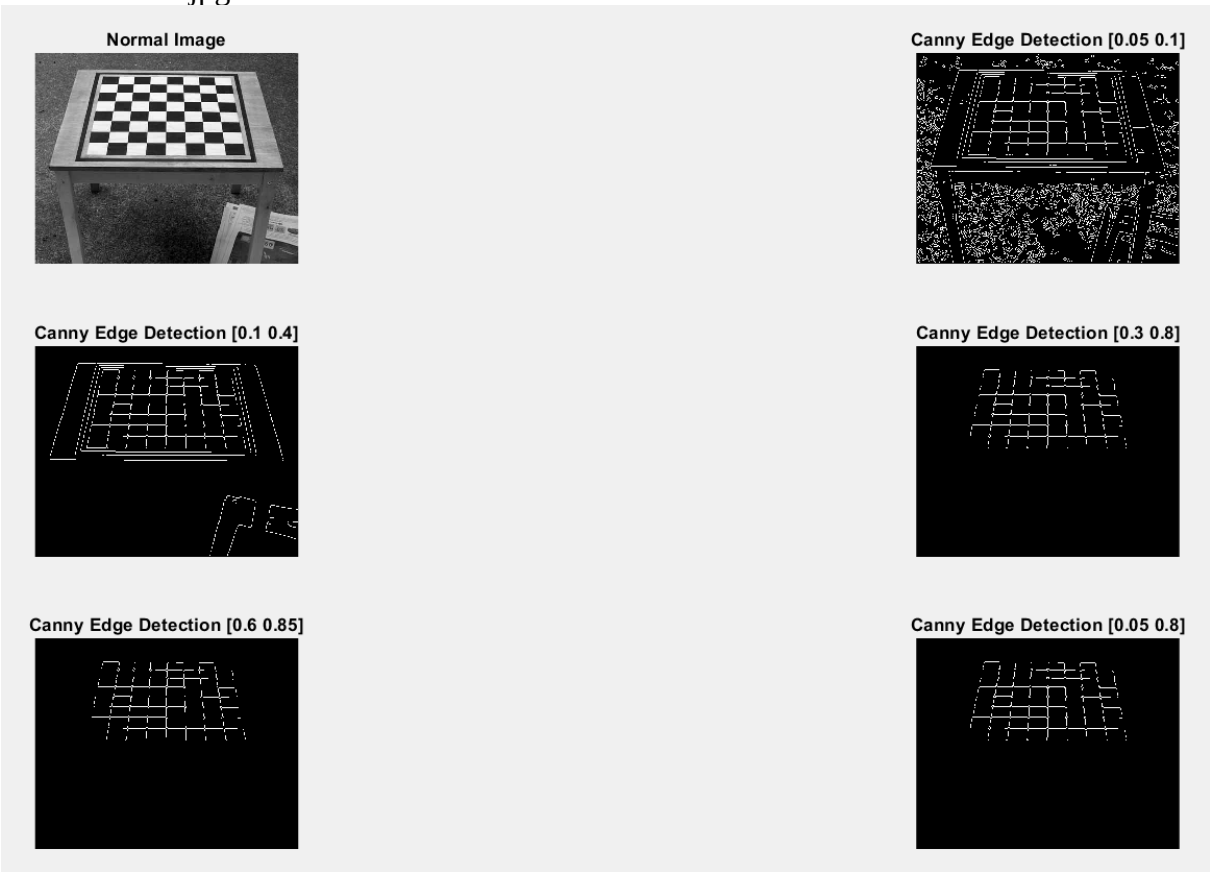
eiffel.jpg



leopard.jpg



Checkerboard.jpg



Again, we applied Canny method to get the edges of Eiffel, leopard, checkerboard images. We can see that 0.05-0.1 and 0.1-0.4 gives the best results because the image contains more weak edges. Other ones are pruning the weak edges more than other first two images edge detected images.

2. **Roberts Edge Detection**

Finds edges at those points where the gradient of the image is maximum, using the Roberts approximation to the derivative.

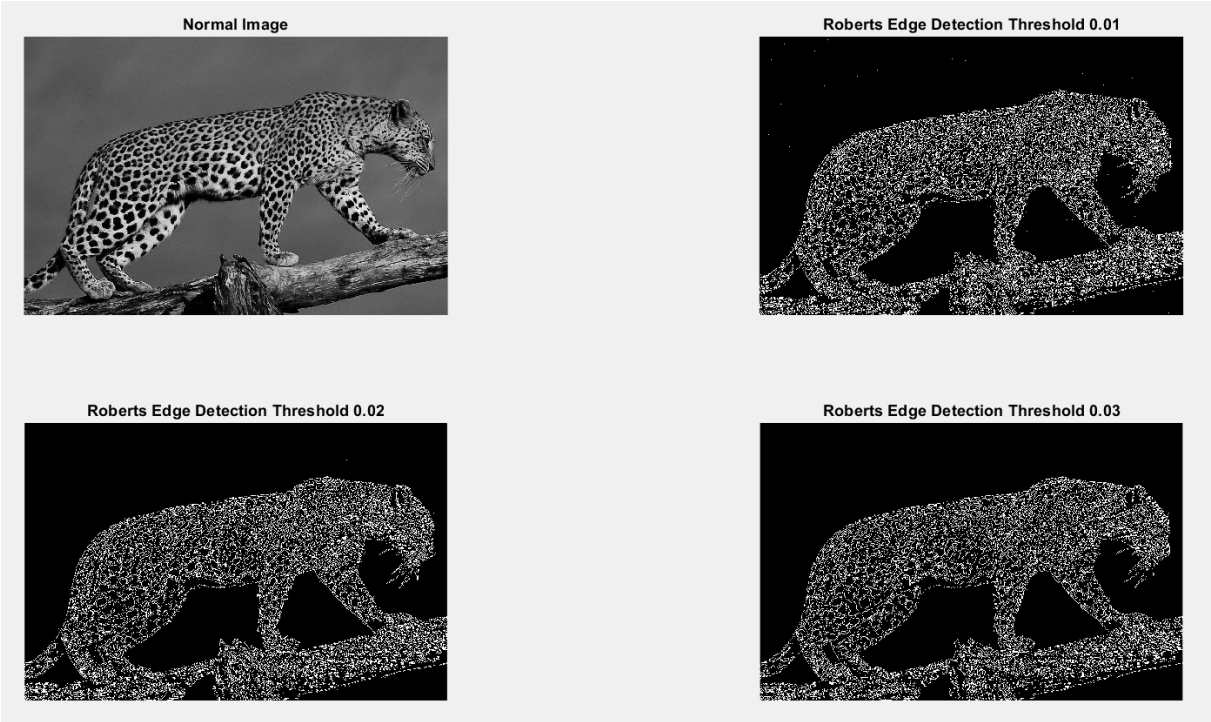
+1	0
0	-1

Gx

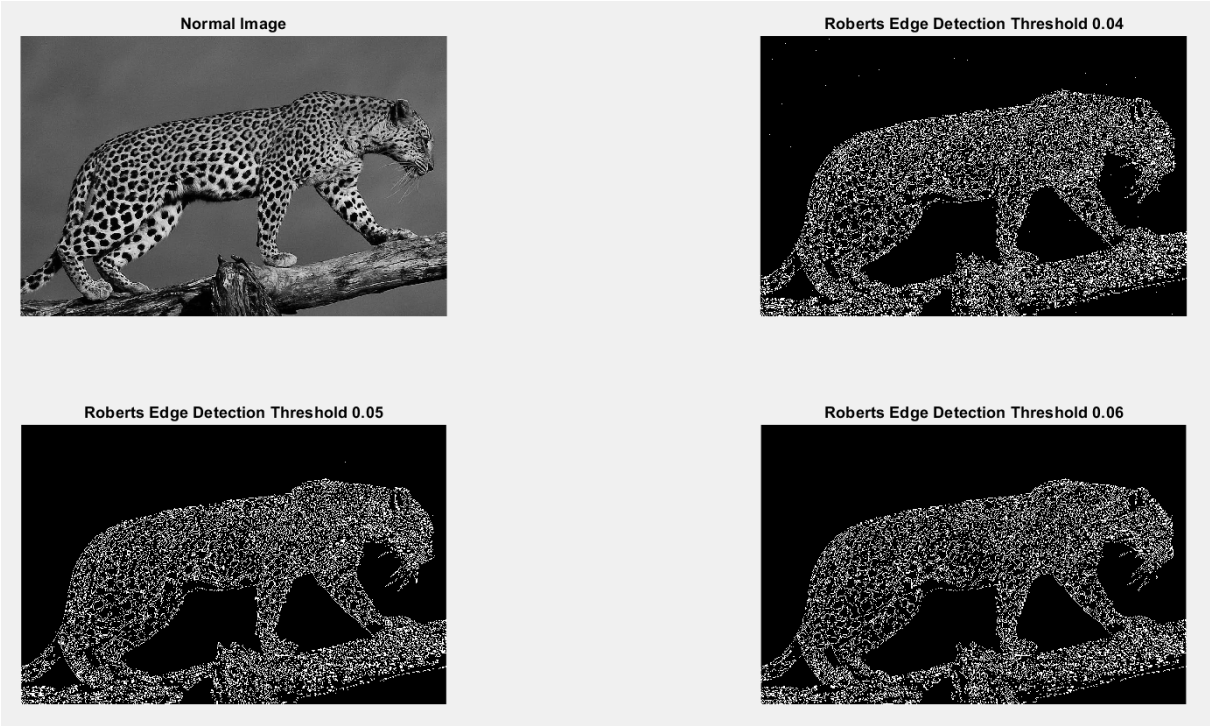
0	+1
-1	0

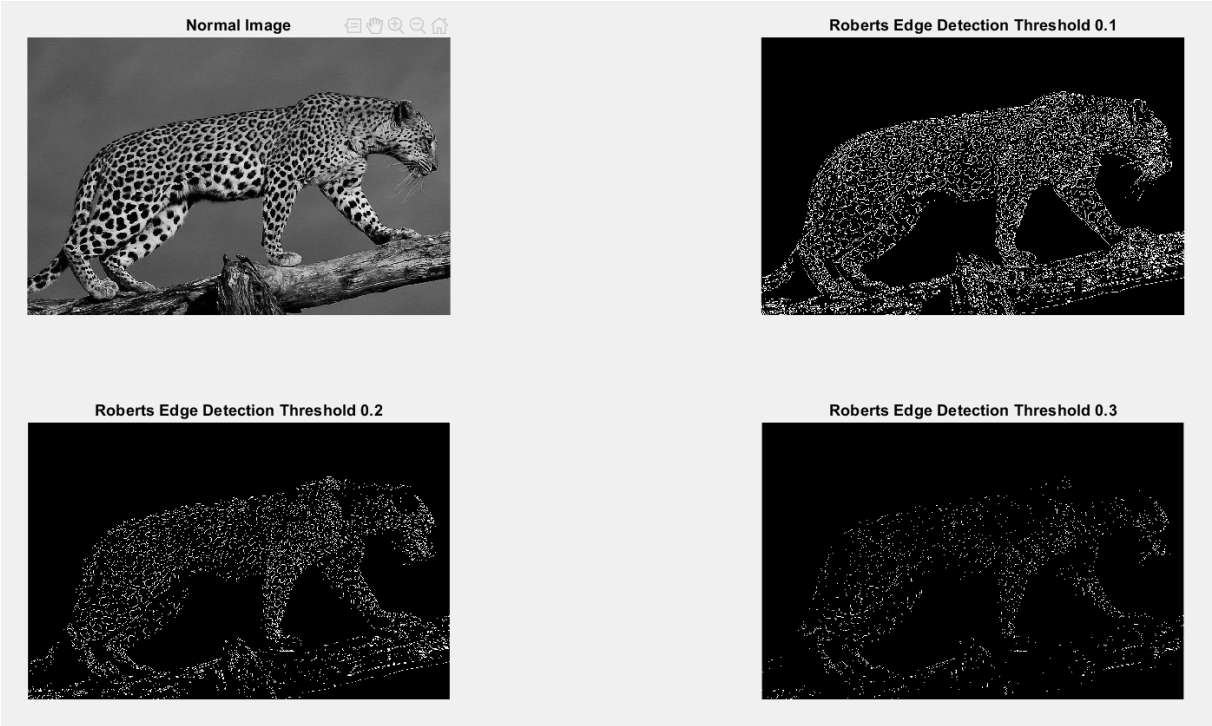
Gy

These convolution kernels are used to achieve a Roberts edge detection.



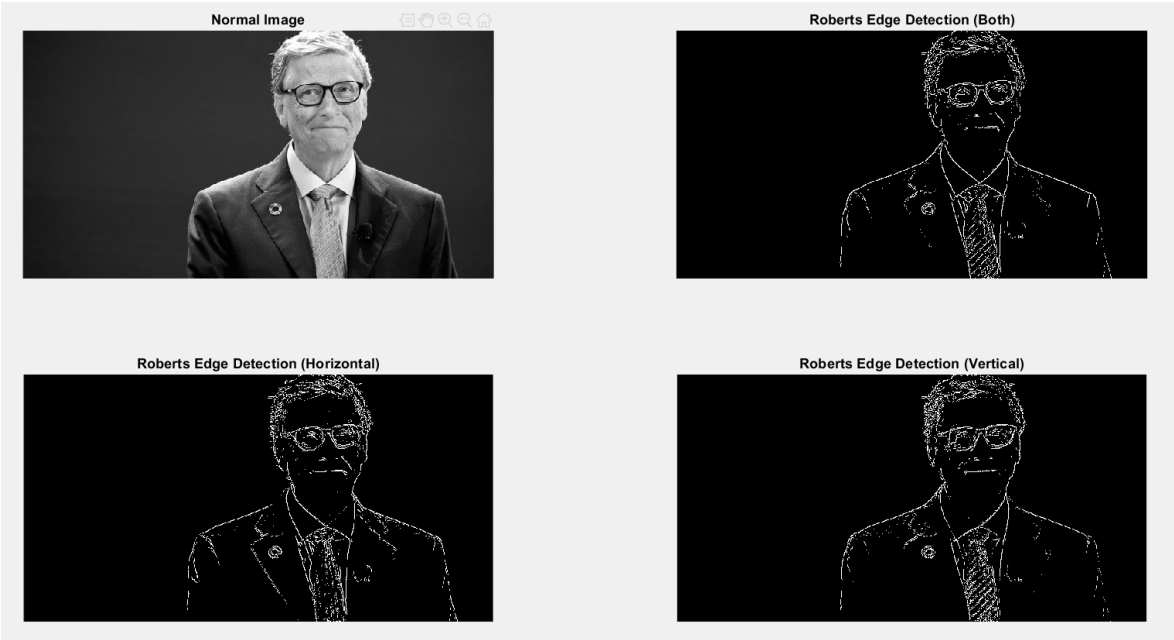
We examined threshold values to apply Roberts edge detection method and we can say that still have a understanding as too little threshold gives more pixels and weak edges. As we can see on the leopard image with a threshold of 0.01, there is some white spots on the background that is caused of a low size of threshold.

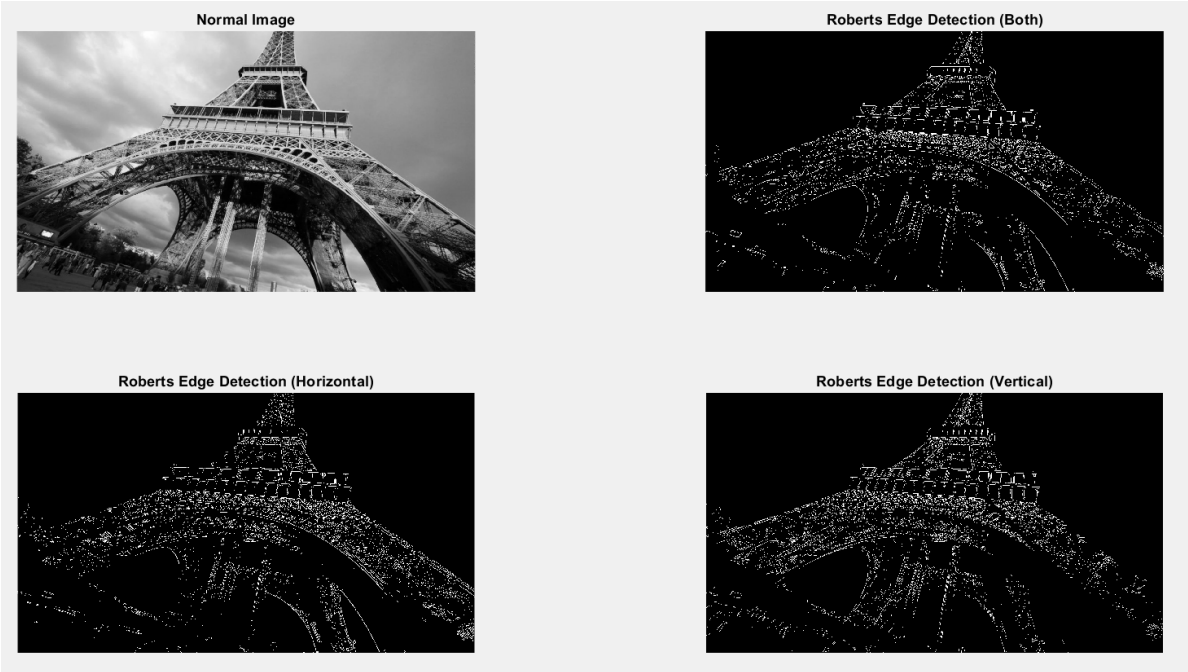




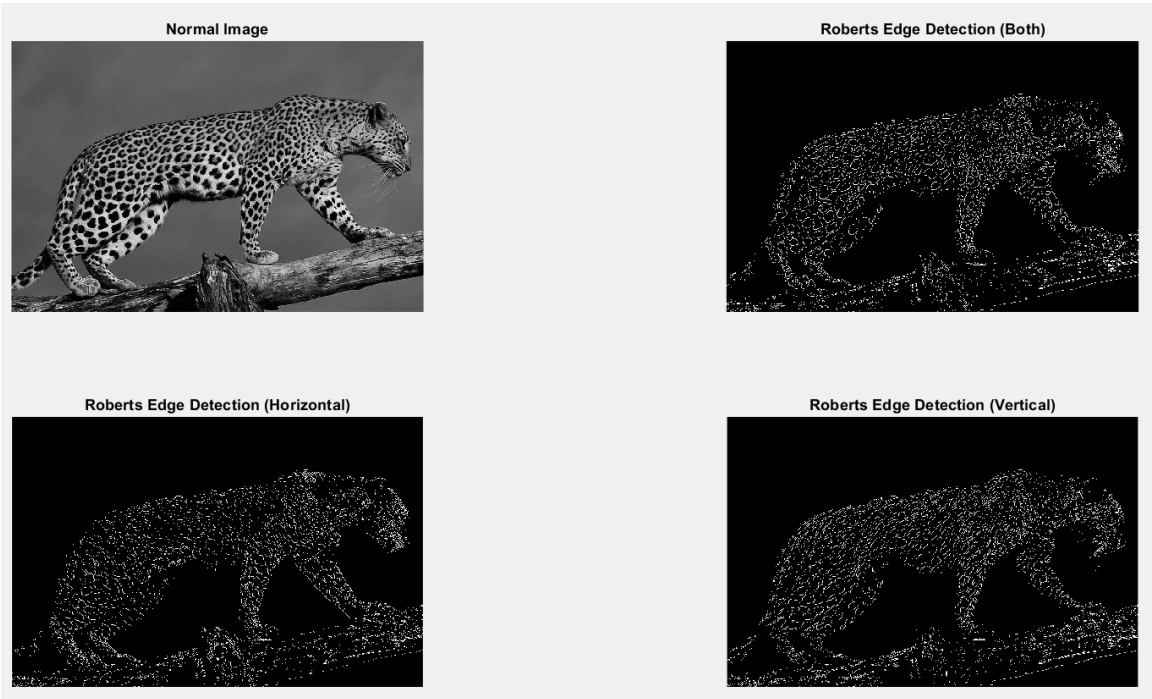
As we increase the threshold values, we obtain more strong edges in comparison with the other results.

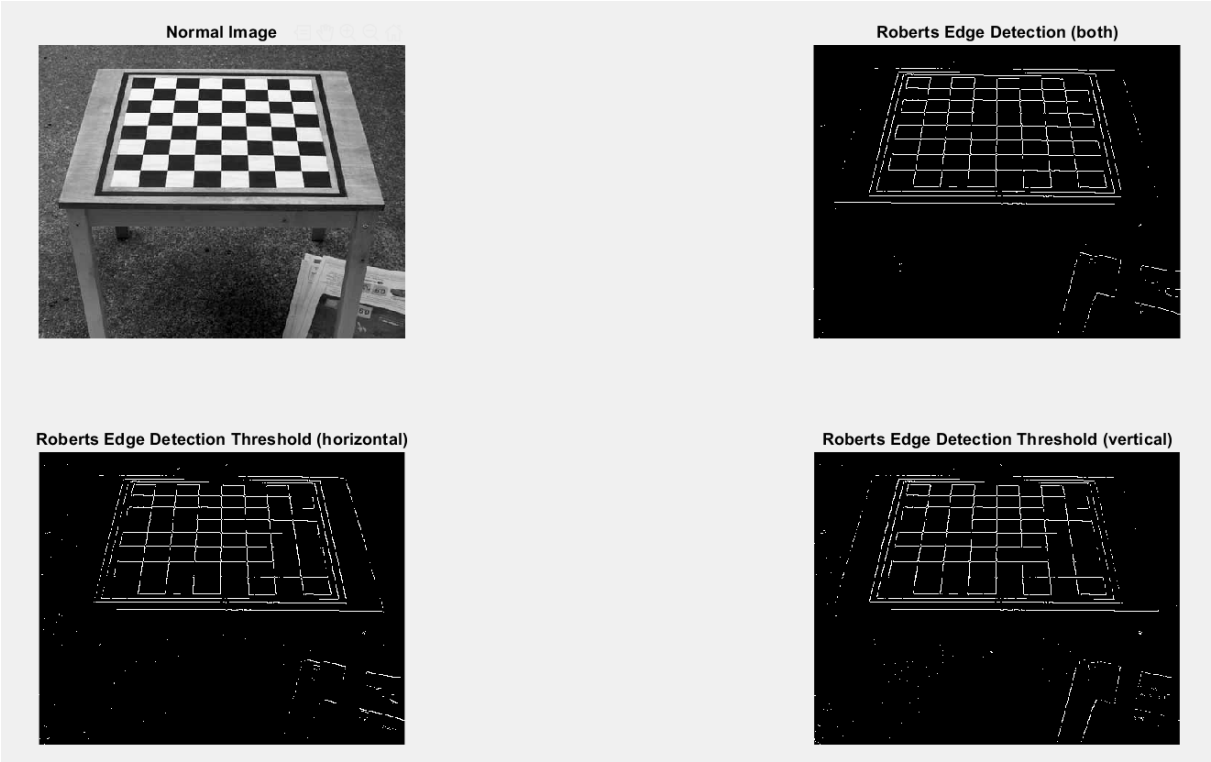
Gates.jpg





On gates.jpg and Eiffel.jpg image Roberts edge detection is applied as horizontal, vertical and both together. On the horizontal one first kernel is used and can be seen in the image it shows the edges on horizontal axes, on the other hand in the last image it is applied as vertically that shows vertical edges. MATLAB default of this Roberts detection function is showing the edges on the both directions.





On leopard.jpg, checkerboard.jpg images actually we did not obtain very different results. I think it is because of the feature of the texture of the leopard fur that show the similar results on both vertical and horizontal directions.

On the checkerboard we can only see the newspaper part on the ground as vertical and horizontal difference.

3. Prewitt Edge Detection

In Prewitt Edge detection we use find the edges at those points where the gradient of the image is maximum, using the Prewitt approximation to the derivative.

-1	0	1
-1	0	1
-1	0	1

Vertical Edge Detection

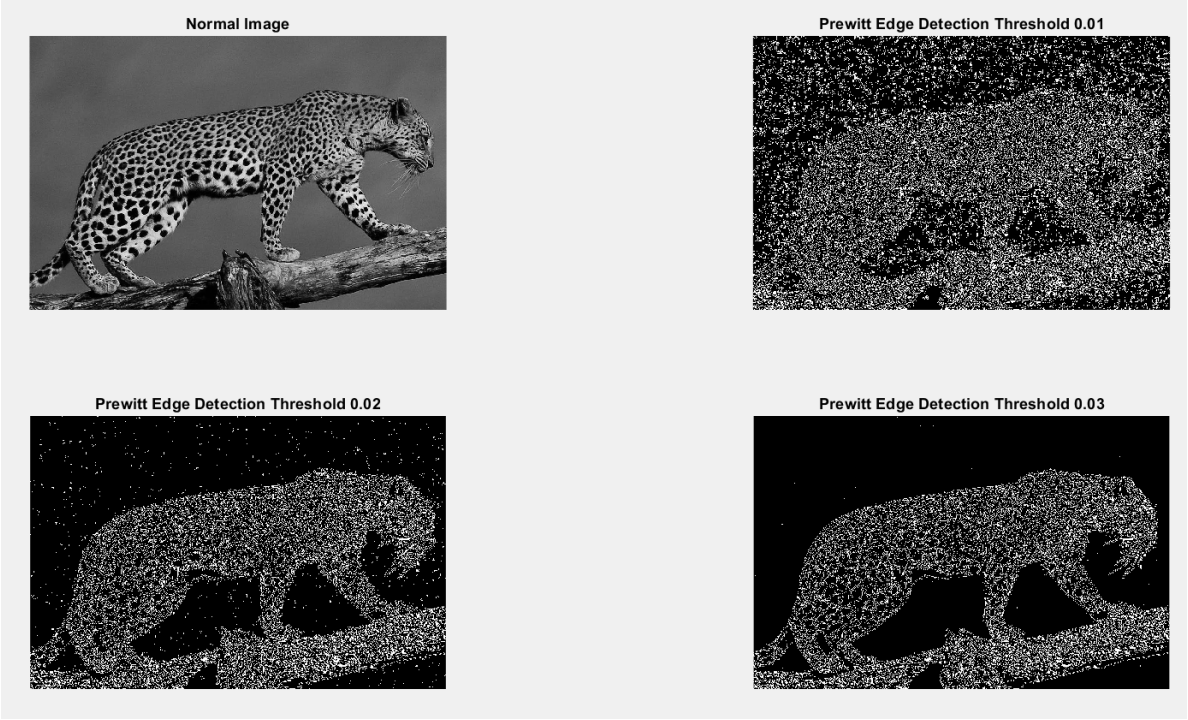
Noise Smoothing

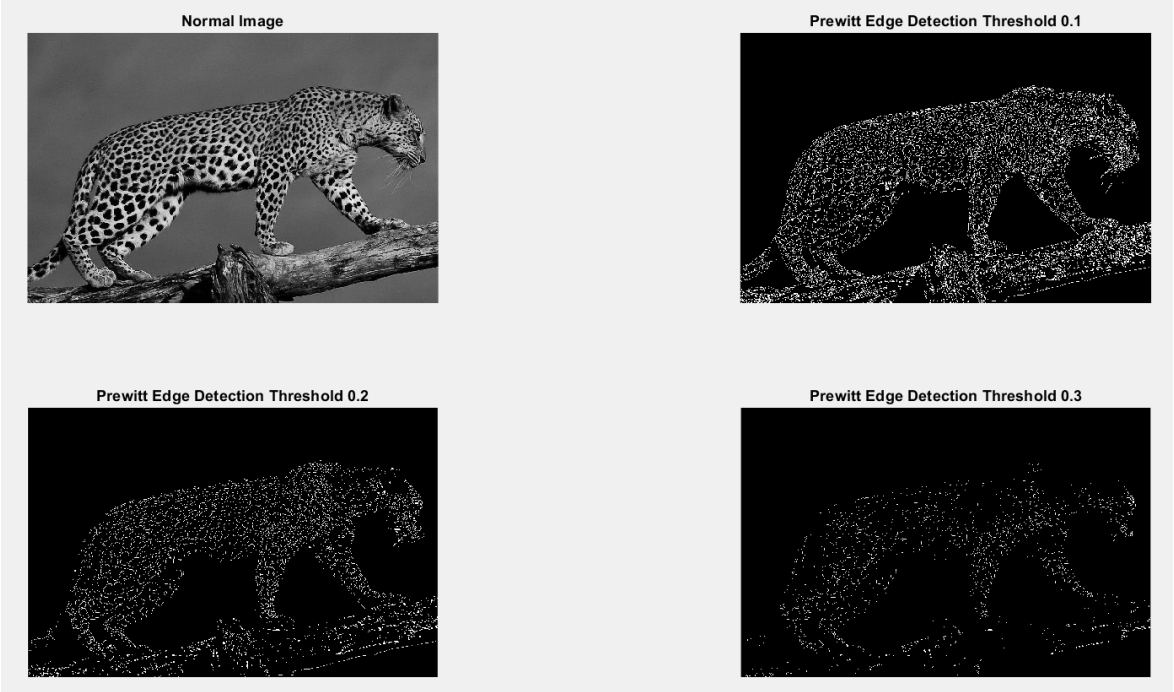
-1	-1	-1
0	0	0
1	1	1

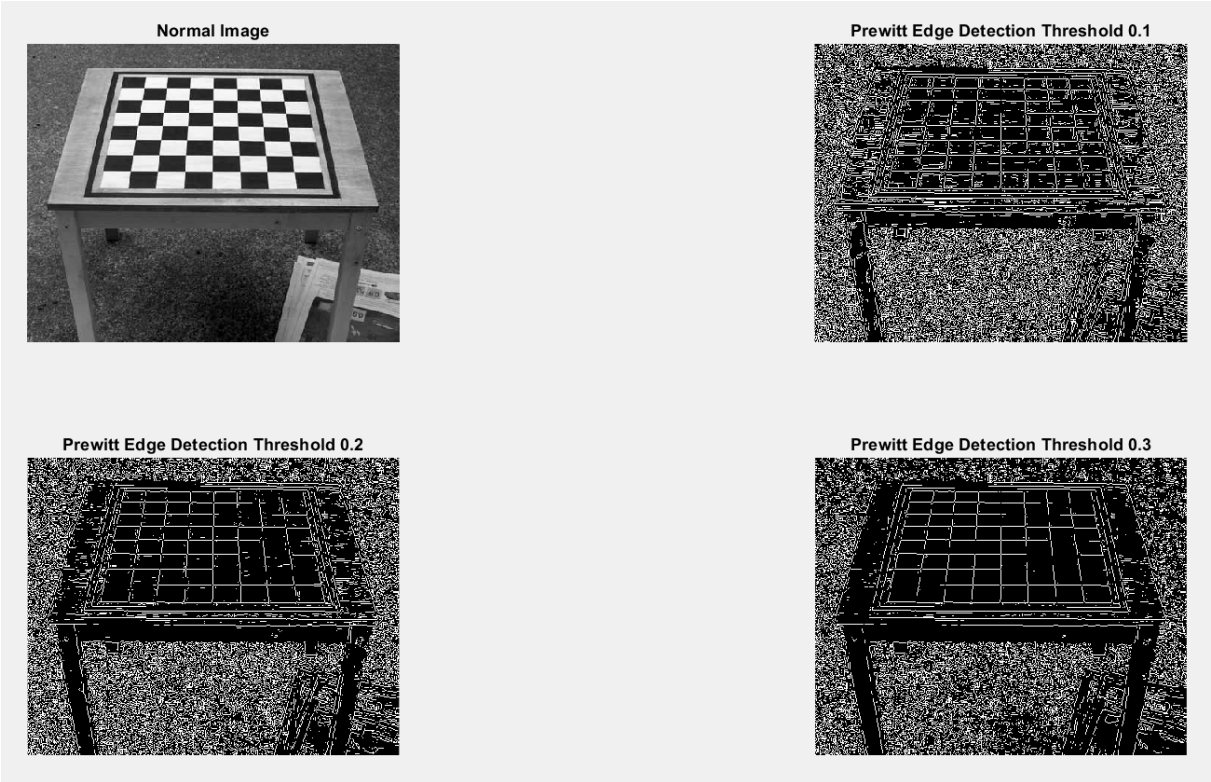
Noise Smoothing

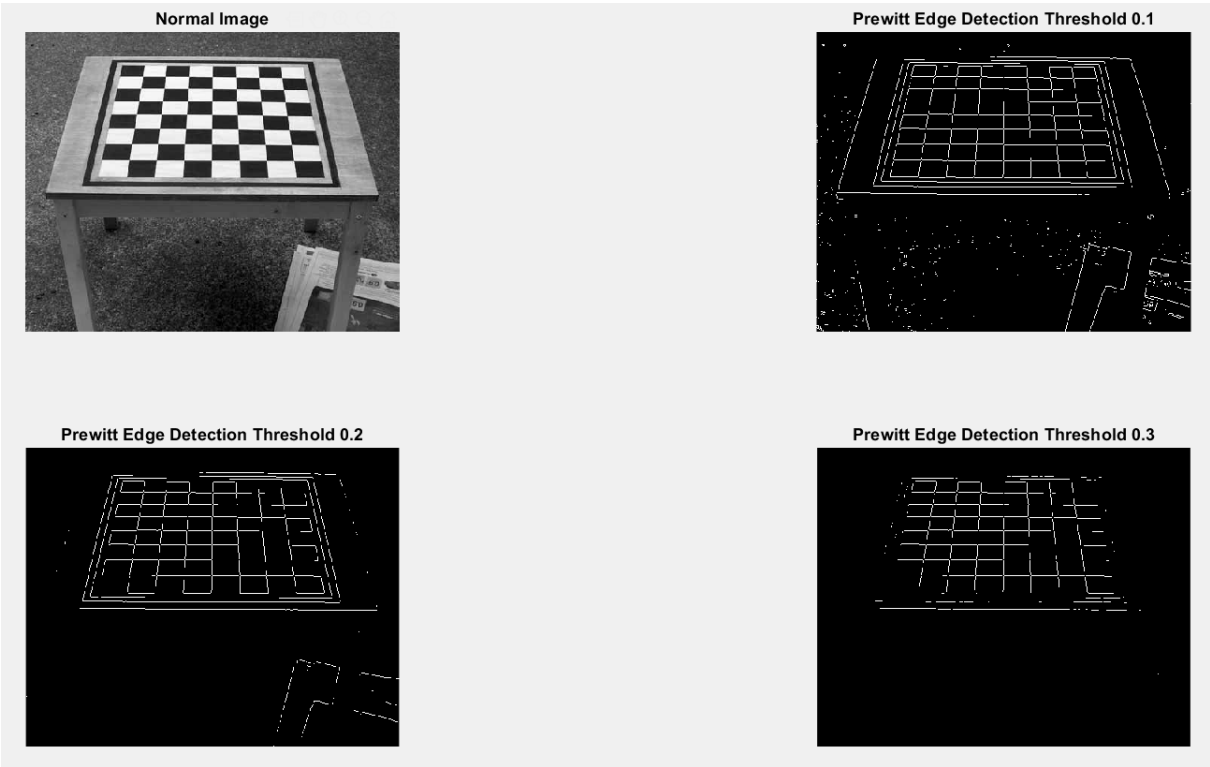
Horizontal Edge Det

By convolving these kernels with the image, we actually filtering the noise and detect the vertical/horizontal edges.

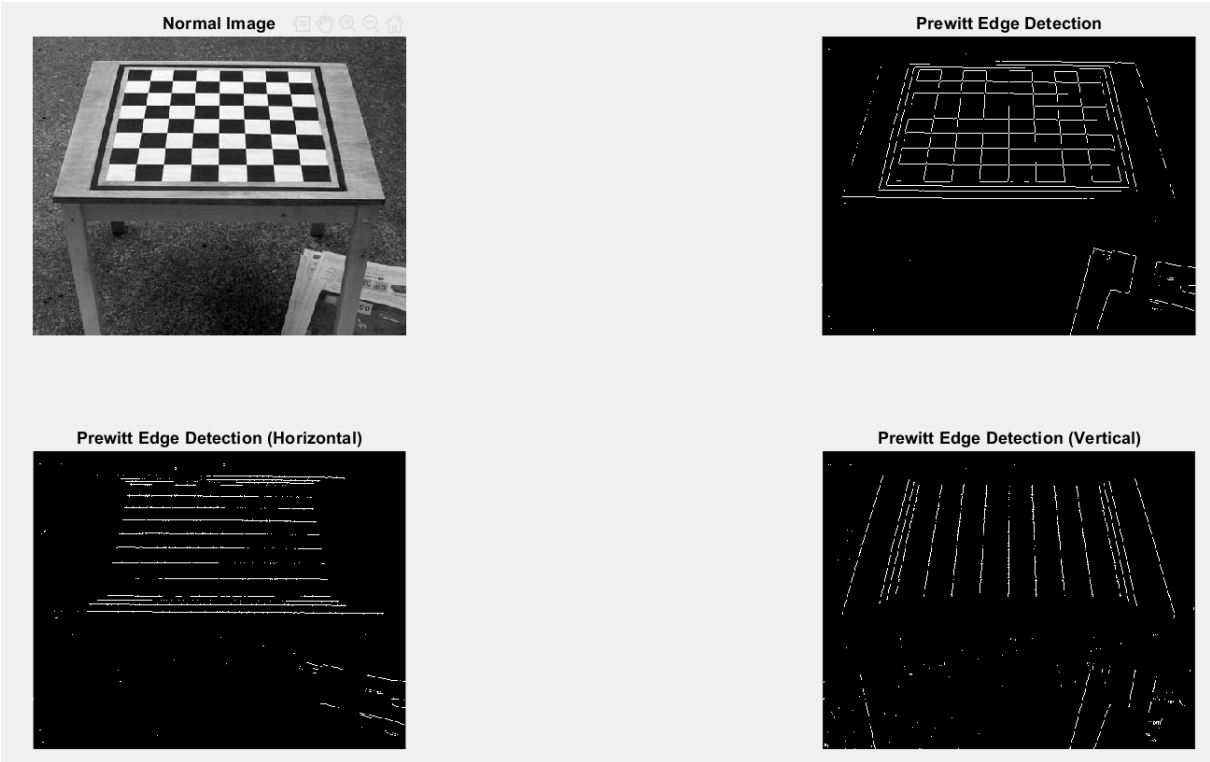

Normal Image
Prewitt Edge Detection Threshold 0.01
Prewitt Edge Detection Threshold 0.02
Prewitt Edge Detection Threshold 0.03


Normal Image
Prewitt Edge Detection Threshold 0.1
Prewitt Edge Detection Threshold 0.2
Prewitt Edge Detection Threshold 0.3





Again as Roberts edge detection, we examined threshold values to apply Prewitt edge detection method and we can say that still have a understanding as too little threshold gives more pixels and weak edges. As we can see on the leopard image with a threshold of 0.01, there are noise that is caused of a low size of threshold.



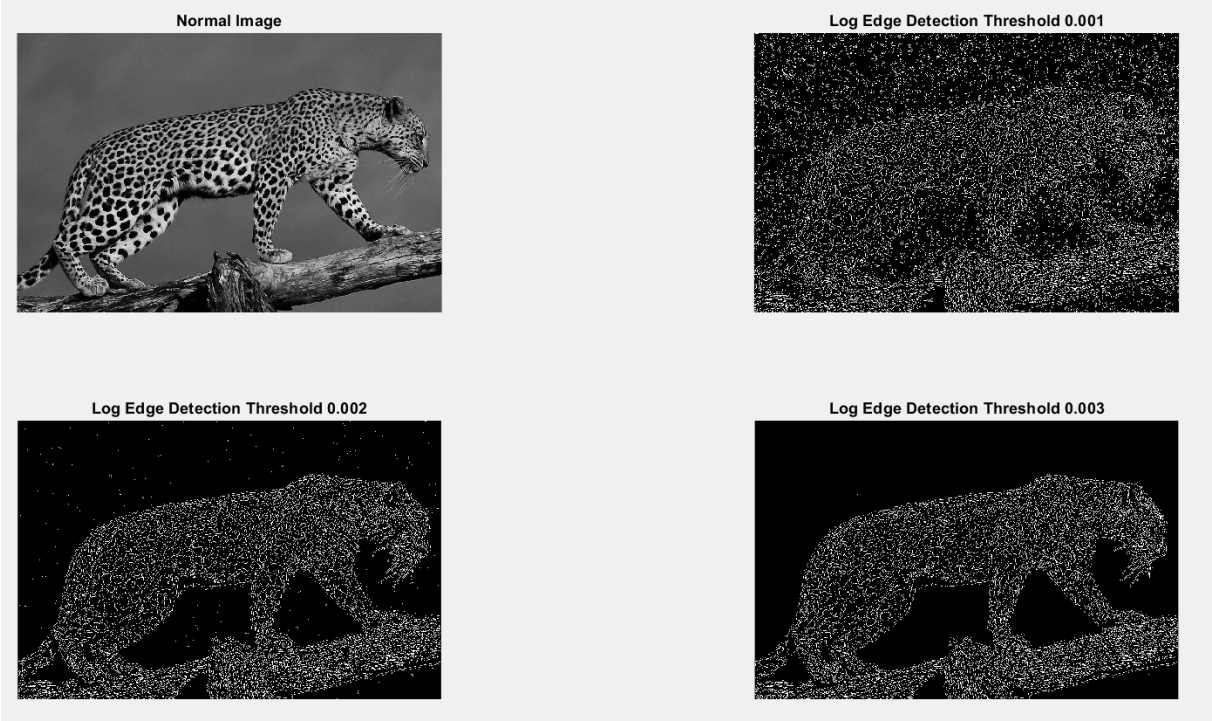
In comparison with the Roberts edge detection we obtained better results for the edges in horizontal and vertical directions.

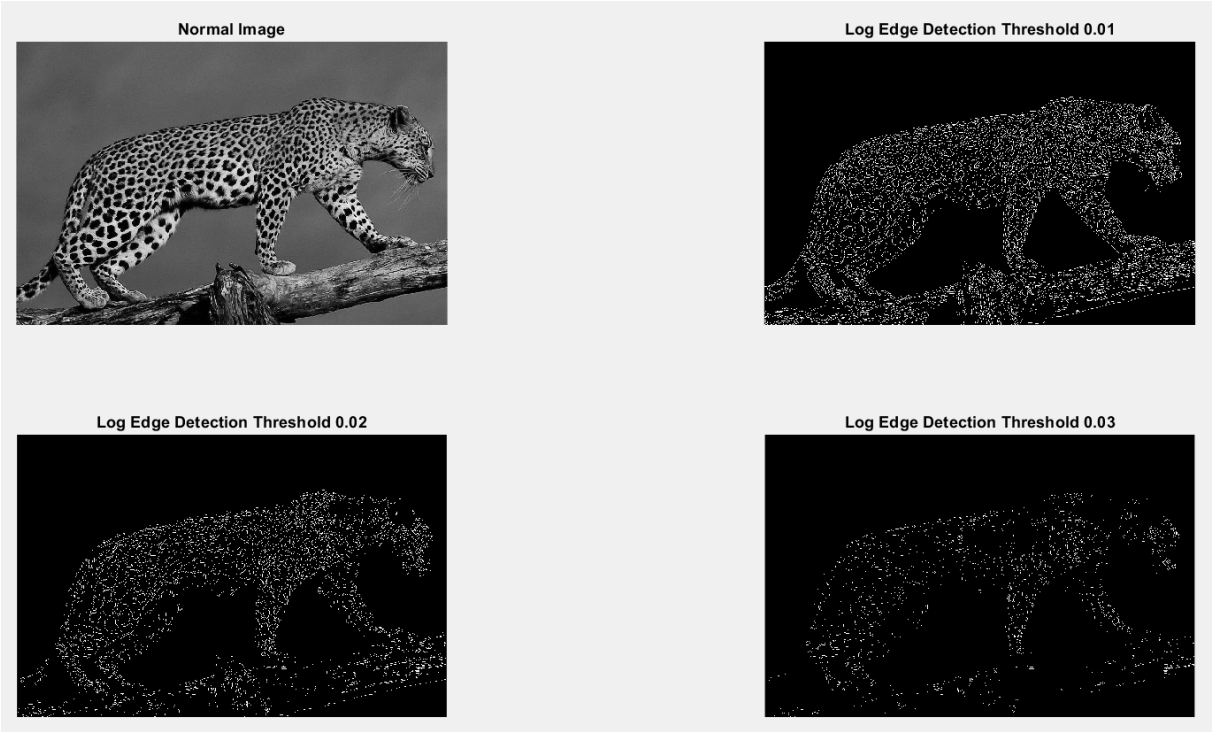
4. **Log Edge Detection**

In that edge detection method the image is firstly smoothed by Gaussian filter then, find zero-crossings (Laplacian filter). It can be approximated as second derivative.

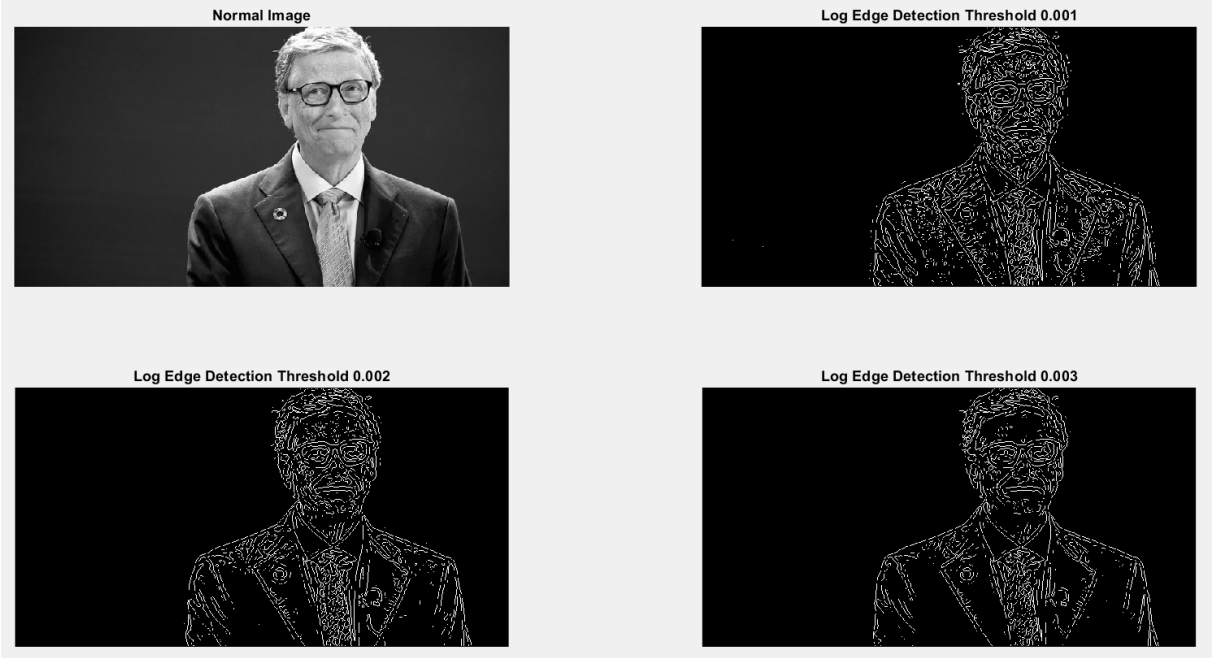
$$O(x,y) = \nabla^2(I(x,y) * G(x,y)) = \nabla^2(I(x,y) * G(x,y))$$

Leopar.jpg

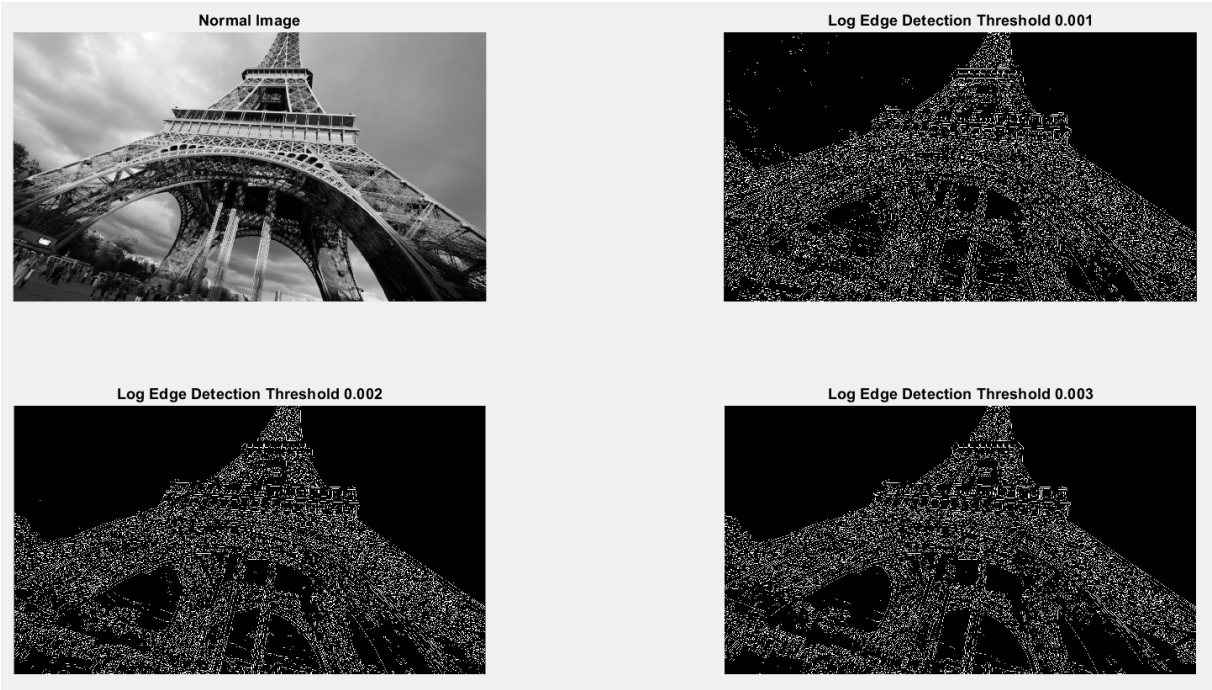

Normal Image
Log Edge Detection Threshold 0.001
Log Edge Detection Threshold 0.002
Log Edge Detection Threshold 0.003


Normal Image
Log Edge Detection Threshold 0.01
Log Edge Detection Threshold 0.02
Log Edge Detection Threshold 0.03

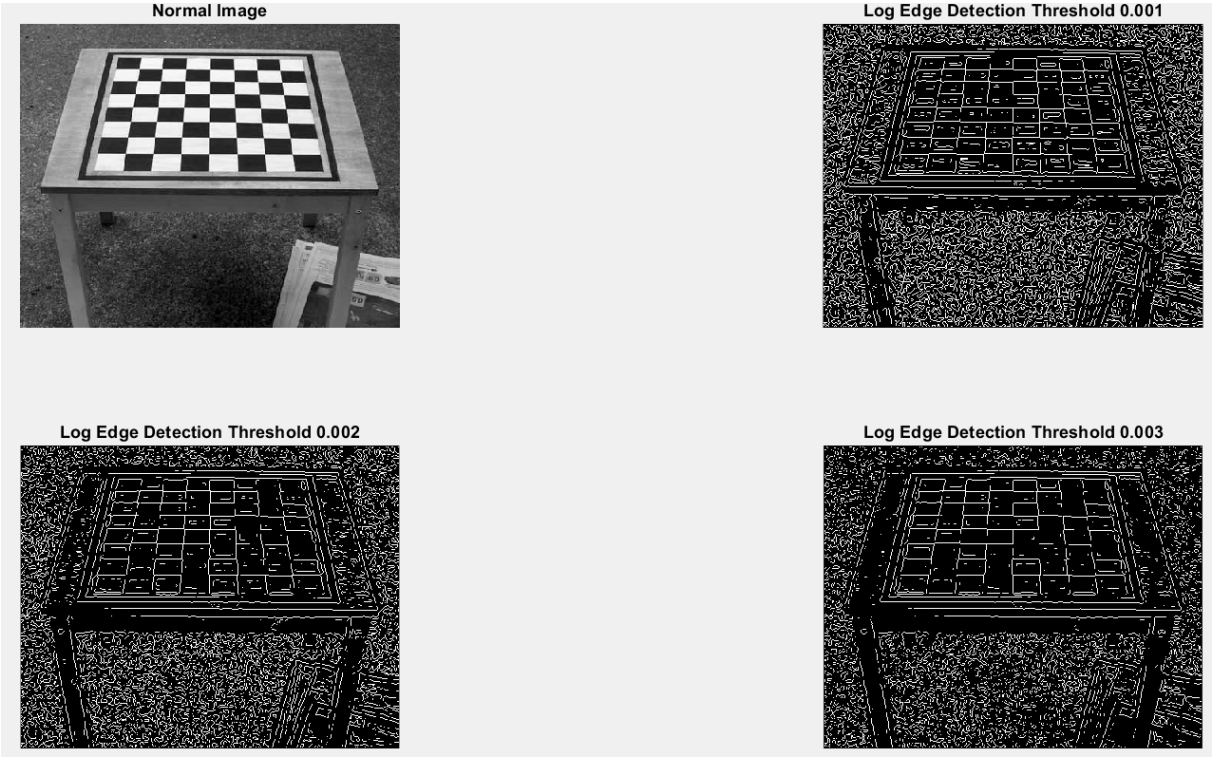
Gates.jpg



Eiffel.jpg



Checkerboard.jpg



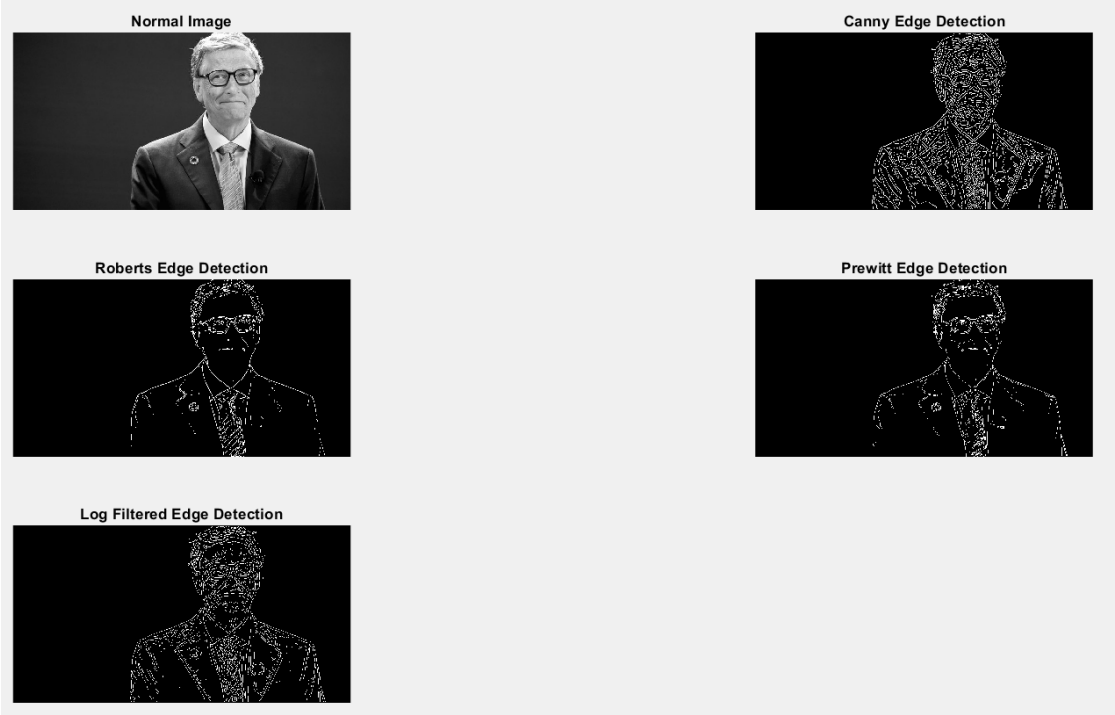
We examined threshold values to apply LoG edge detection method and we can say that still have a understanding as too little threshold gives more pixels and weak edges. As we can see on the images with a threshold of 0.001 (it also have lower threshold values than first derivative edge detectors), there are some white spots on the background that is caused of a low size of threshold.

On the other hand, on checkerboard.jpg image there is so much noise and it could not give a sufficent result.

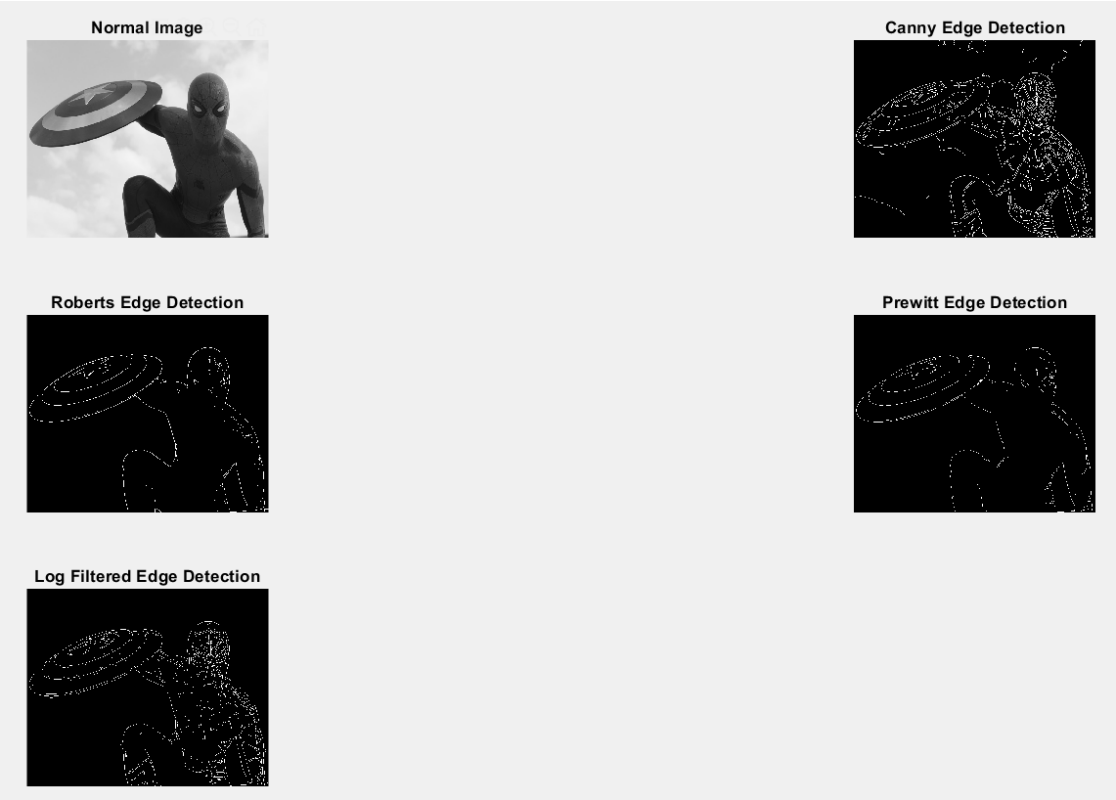
5. **Edge Detector Comparisons**

We applied several edge detection algorithms on the images.

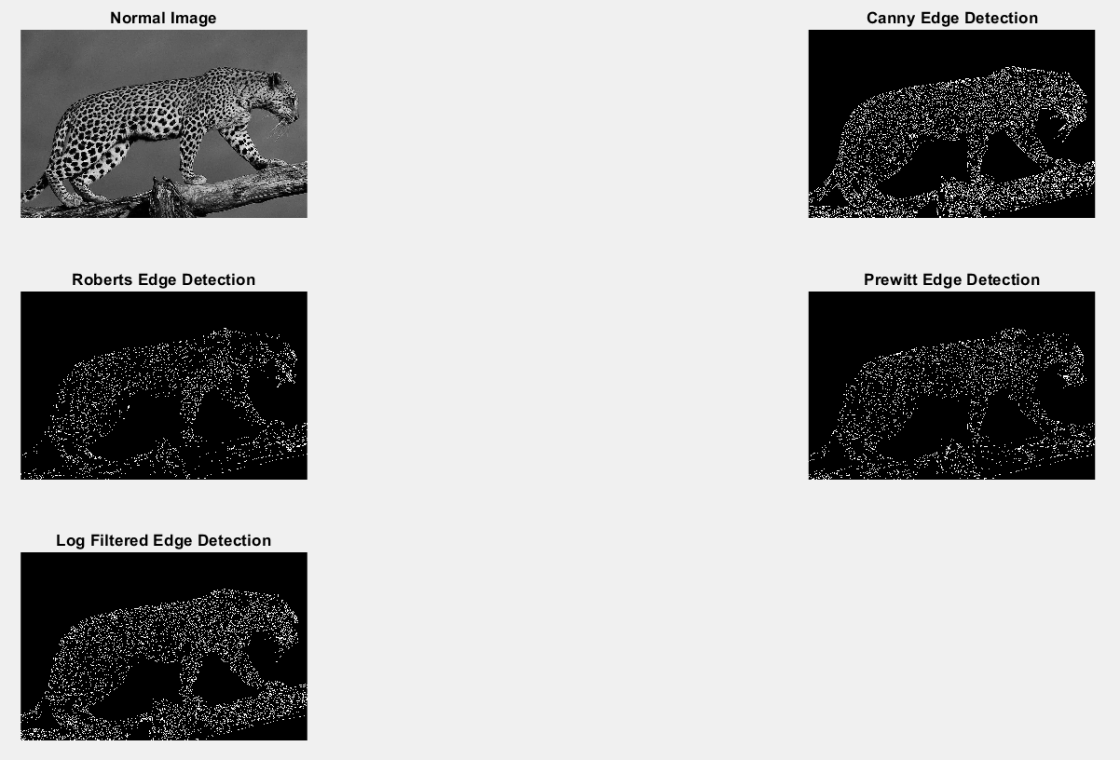
Gates.jpg



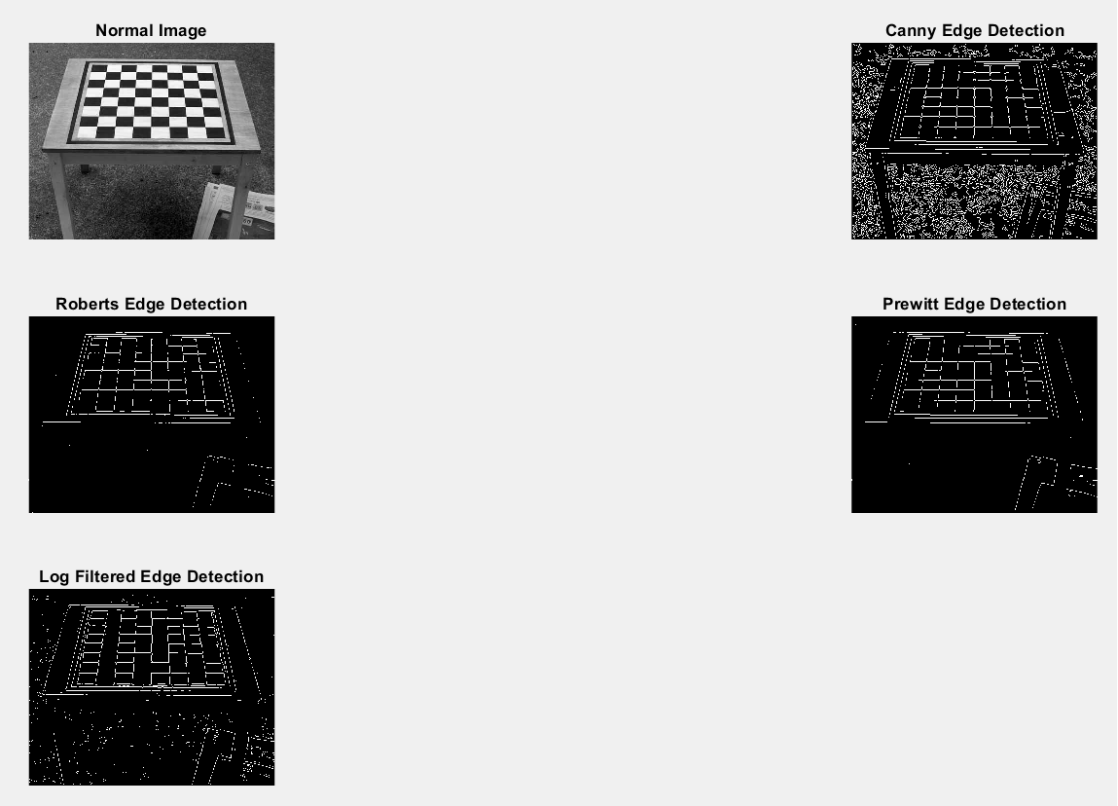
Spiderman.jpg



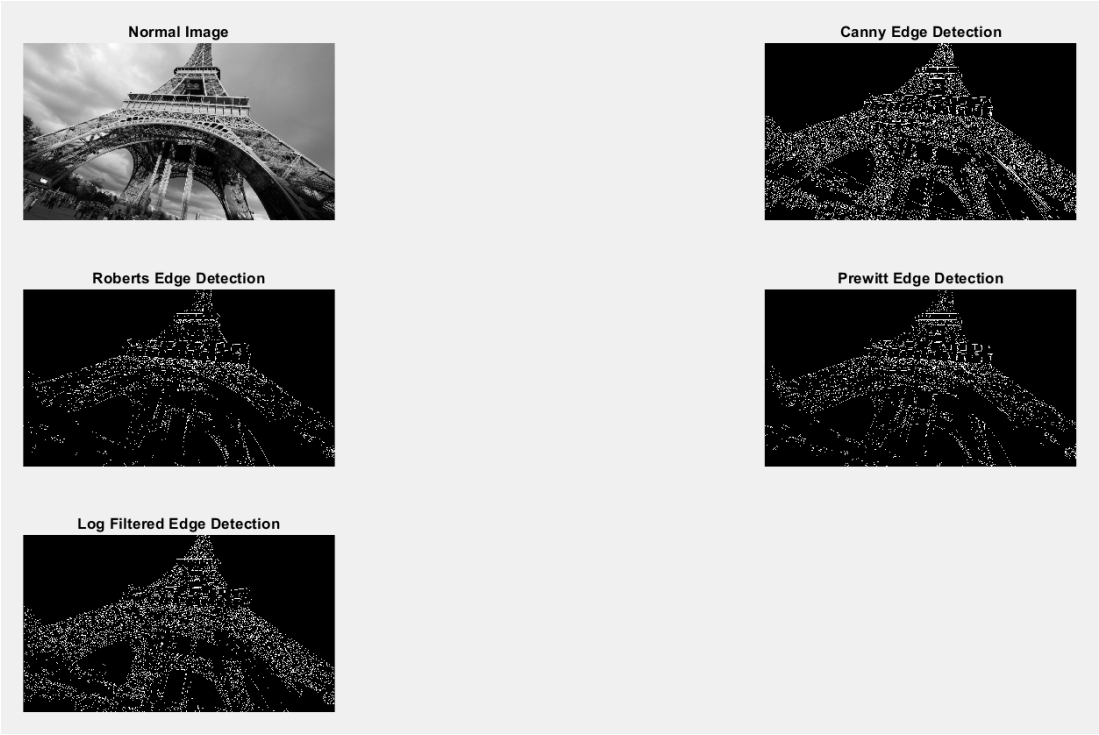
Leopard.jpg



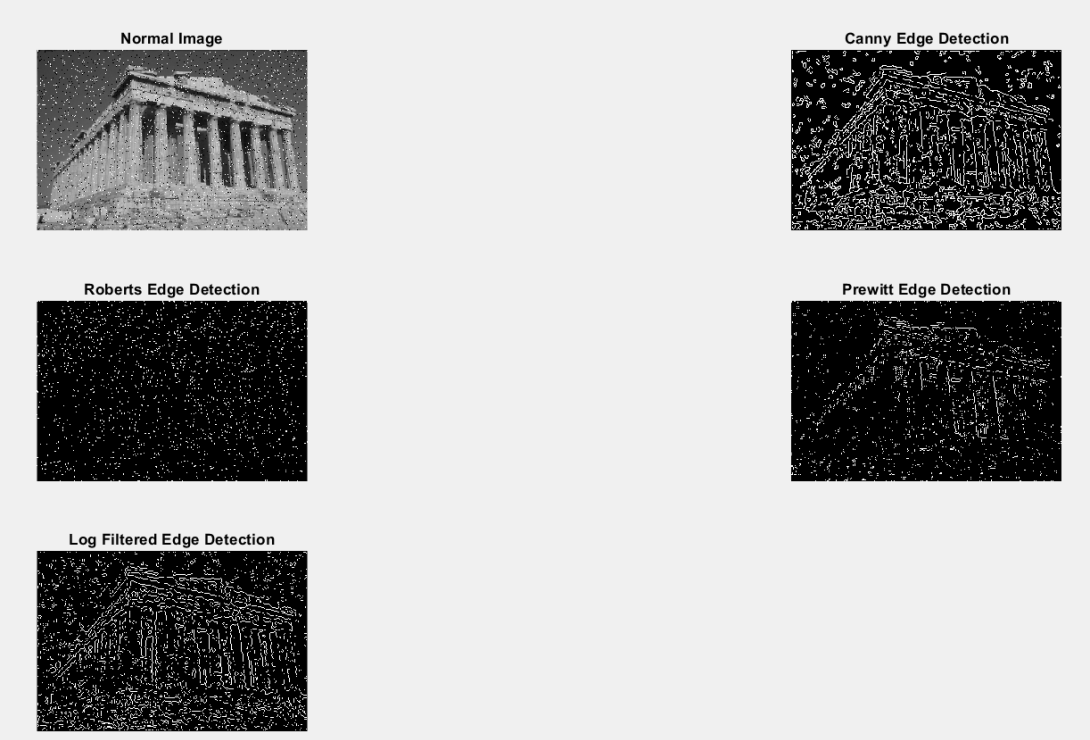
Checkerboard.jpg



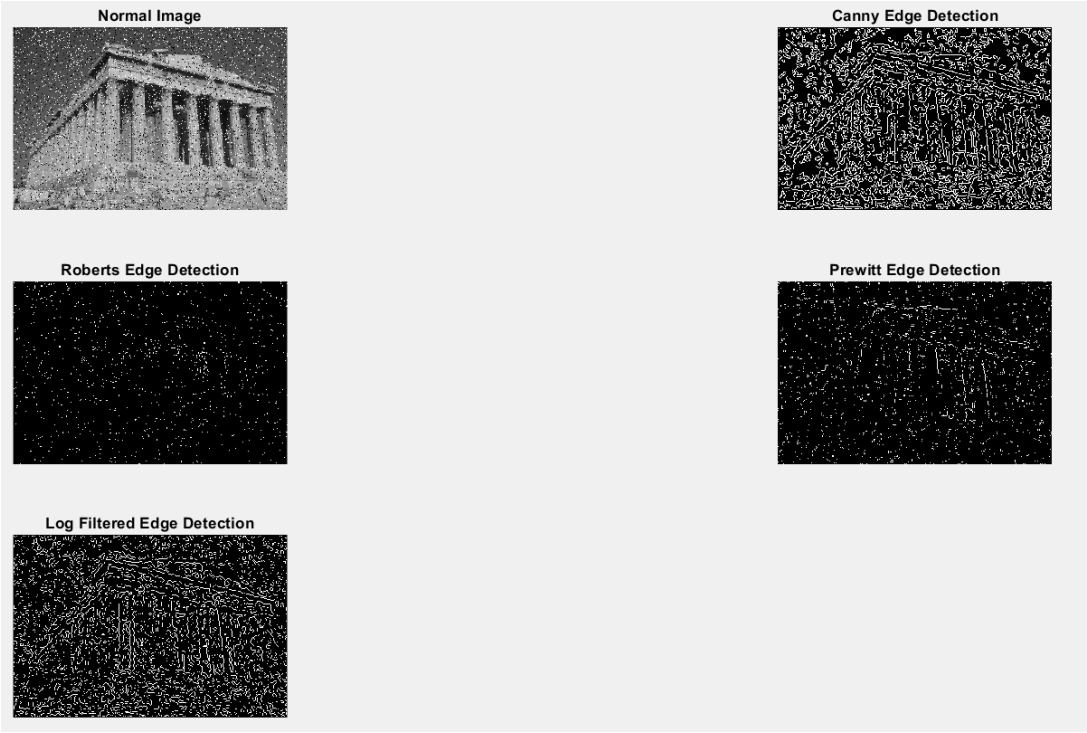
Eiffel.jpg



Rome.png %5 Noisy



Rome.png %10 Noisy



DISCUSSION

We can say that Canny edge detector is an optimized approach to edge detection. As we mentioned earlier it has a understanding of strong edges reinforces weak edges. As we can see on gates.jpg, spiderman.jpg, checkerboard.jpg, eiffel,jpg, leopard.jpg images it covers most of the edges (especially weak edges). On checkerboard.jpg it gives a poor solution because there are many points that is considered as weak edges by the canny edge detector.

If we look at Roberts and Prewitt Edge detectors they actually cover mostly strong edges as can be seen clearly gates.jpg, spiderman.jpg and checkerboard.jpg images, strong edges are shown and weak edges are disregarded. Their results are better at checkerboard.jpg because this image contains weak edges that needs to be filter out. Strong edges are important in that image.

Edges are so similar in leopard.jpg, eiffel.jpg images and because of that reason all edge detectors performed at similar performance. There are not strict weak and strong edge differences.

Apart from all these, if we look at rome.png which is a noisy image, Canny edge detector and Laplacian of gaussian edge detector gives better results than Roberts and Prewitt because it Canny is less vulnerable to noise also Gaussian filter of log is also beneficial for edge detection in the noisy images. Roberts and Prewitt looked for strong edges but because of the noise, they are degenerated, so they had poor results.

To sum up, if there are so many weak edges that needs to be considered, Canny Edge detector is better, if we need to consider strong edges more importantly we can prefer Roberts and Prewitt Edge detectors.

CODES

hw1.m

```
img = imread('checkecboard.jpg');
%img = imread('leopard.jpg');
%img = imread('gates.jpg');
%img = imread('eiffel.jpg');
%img = imread('rome.png');

img = rgb2gray(img);
cannyedge = edge(img,'Canny');
prewittedge = edge(img,'Prewitt');
robertsedge = edge(img,'Roberts');
logedge = edge(img,'log');

% cannyedge2 = edge(img,'Canny',[0.1 0.4]);
% robertsedge2 = edge(img,'Roberts',0.01); % Thresholding
% robertsedge3 = edge(img,'Roberts', 'horizontal'); %
Horizontal Edge Detection
% robertsedge4 = edge(img,'Roberts', 'vertical'); %
Vertical Edge Detection
% prewittedge2 = edge(img,'Prewitt',0.01); % Thresholding
% prewittedge3 = edge(img,'Prewitt', 'horizontal'); %
Horizontal Edge Detection
% prewittedge4 = edge(img,'Prewitt', 'vertical'); %
Vertical Edge Detection
% logedge2 = edge(img,'log',0.01); % Thresholding
figure;
subplot(3,2,1);
imshow(img);
title('Normal Image');

subplot(3,2,2);
imshow(cannyedge);
title('Canny Edge Detection');

subplot(3,2,3);
imshow(robertsedge);
title('Roberts Edge Detection');

subplot(3,2,4);
imshow(prewittedge);
title('Prewitt Edge Detection');

subplot(3,2,5);
imshow(logedge);
title('Log Filtered Edge Detection');
```