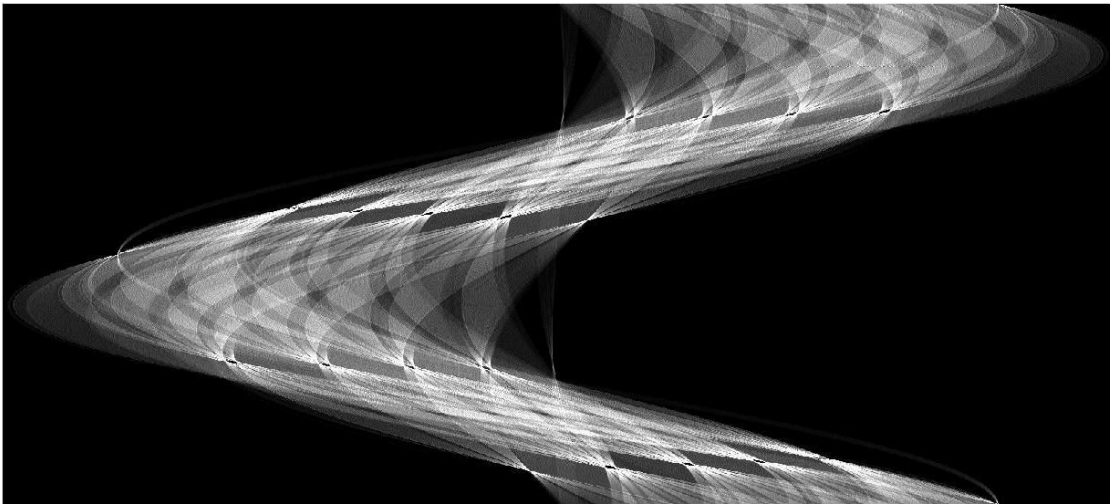# Project 1
Image Filtering & Hough Transform

2.1 Convolution

Figure 2.1.1 is an image which will be used to perform convolution on, using two different kernels. The first kernel is generated using fspecial('gaussian',hsize,sigma); where sigma=3, which will give a gaussian kernel for blurring. The results will be shown in Figure 2.1.2. The second kernel is a box filter using 9x9 matrix of all ones. This will result in a blurring effect but will not be as pleasing as the gaussian kernel due to artifacts introduced by the frequency response of the box kernel. The results will be shown in Figure 2.1.3.
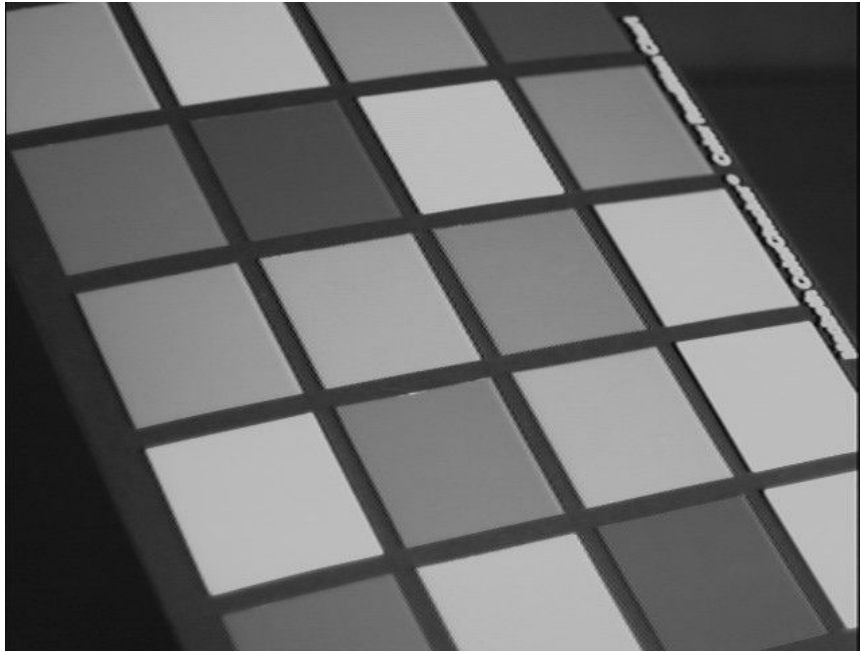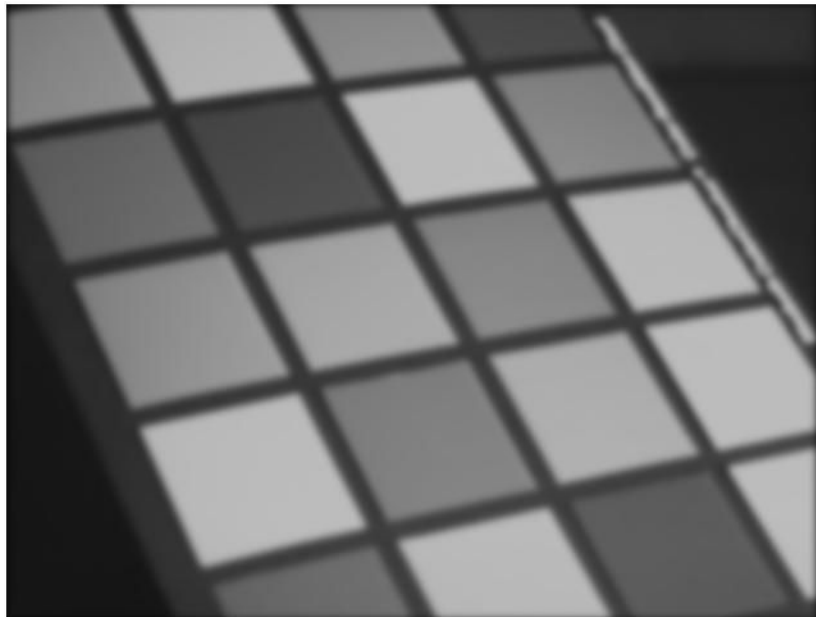
Figure 2.1.1 - Original Image

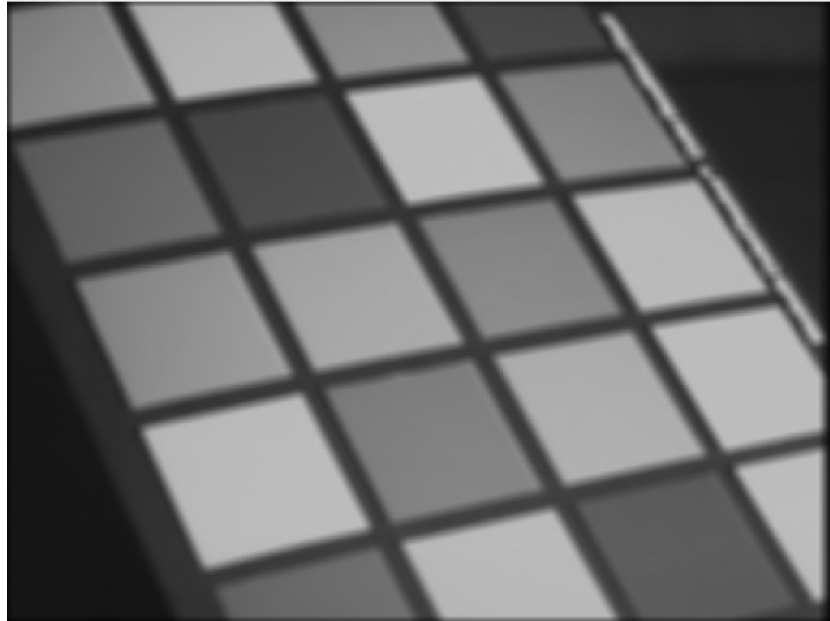Figure 2.1.2 - Convolved with Gaussian kernel (sigma=3)

Figure 2.1.3 - Convolved with 9x9 Box kernel

2.2 Edge Detection

In this part, we will perform edge detection and display imgx, imgy, the magnitude image, as well as the non-maximum suppressed output in Figures 2.2.1, 2.2.2, 2.2.3, and 2.2.4 respectively. All of these will be demonstrated on the original image in Figure 2.1.1.
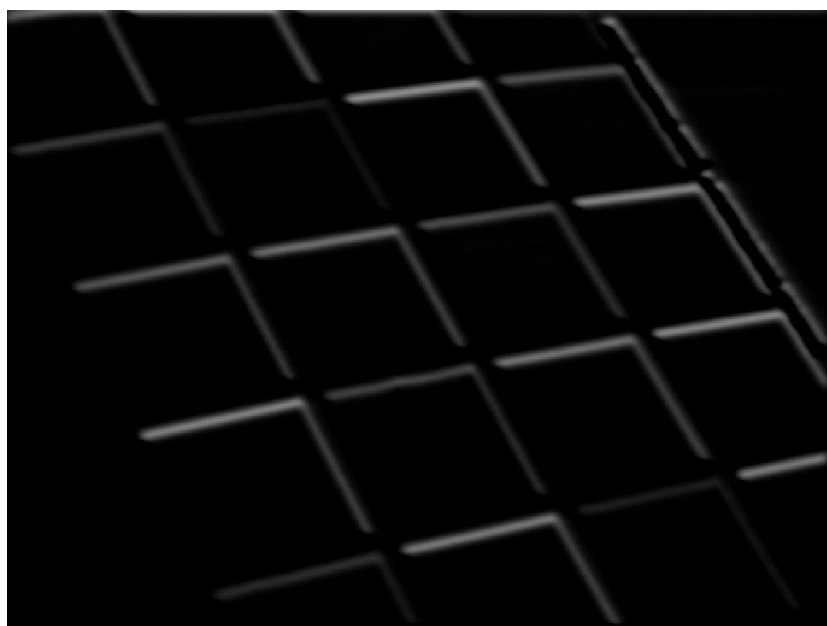
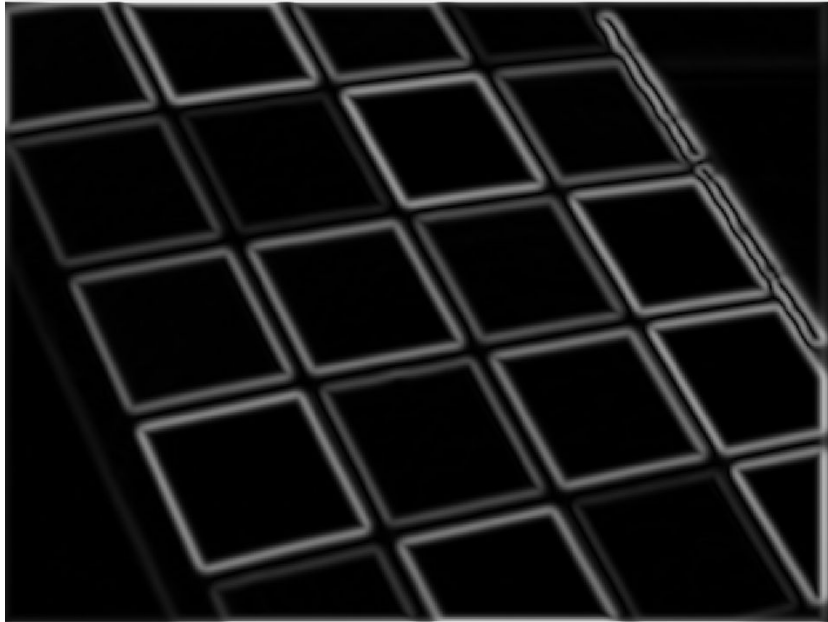Figure 2.2.1 - imgx

Figure 2.2.2 - imgy

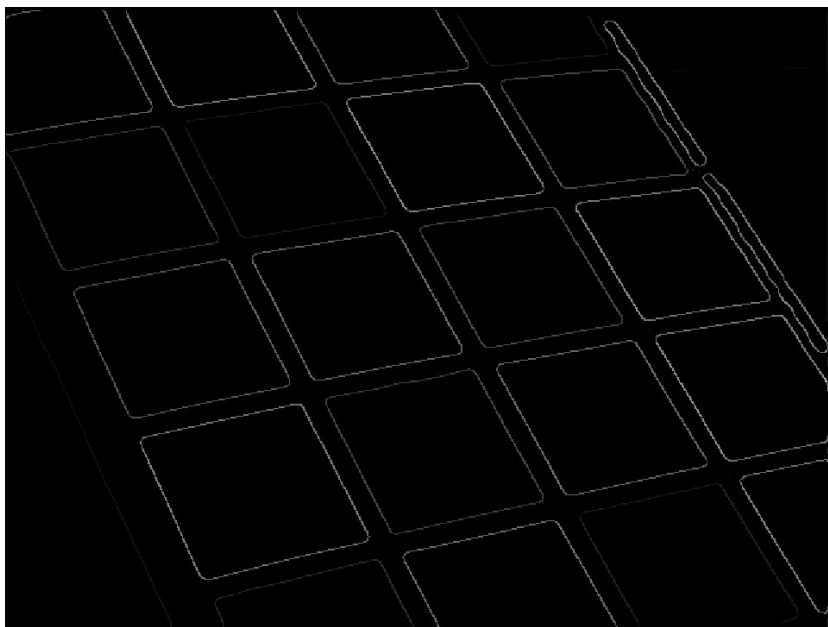Figure 2.2.3 - Edge Magnitude Image

Figure 2.2.4 - NMS Edge Magnitude

2.3 Hough Transform

In this section, we will be performing the Hough transform on the NMS edge magnitude image. The X-axis represents the theta values from 0 to Pi, the Y-axis represents the rho values from -M to M, where M is the maximum possible rho value which can be calculated using sqrt((rows-1)^2 + (cols-1)^2) where rows is the height of the image and cols is the width of the image. The resulting Hough plot is shown in Figure 2.3.1.
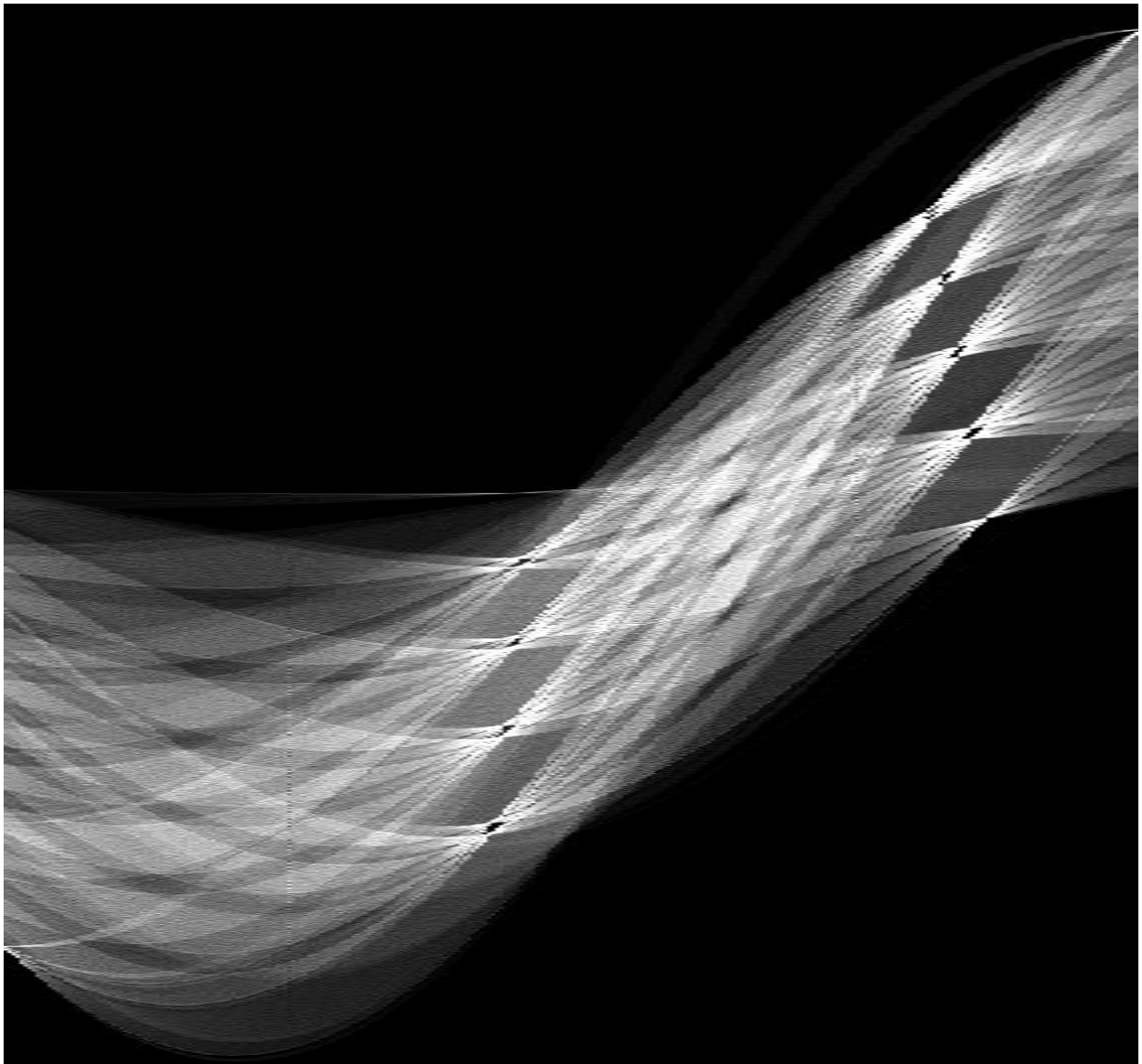

Figure 2.3.1 - Hough Transform

Figure 2.3.1 is achieved using imadjust(rescale(H)) for a more visually appealing Hough transform image.

2.4 Finding Lines

In this section, we will be displaying the top 75 highest peaks in the Hough transform image in Figure 2.3.1 after performing NMS. The nLines parameter will be set to 75. The resulting Hough transform image which includes the peaks in shown in Figure 2.4.1. The resulting plot is not modified like above (i.e. imadjust(rescale(H)) is not applied) in order to visually see the peaks. There are 75 peaks shown as 75 lines are set to be found.
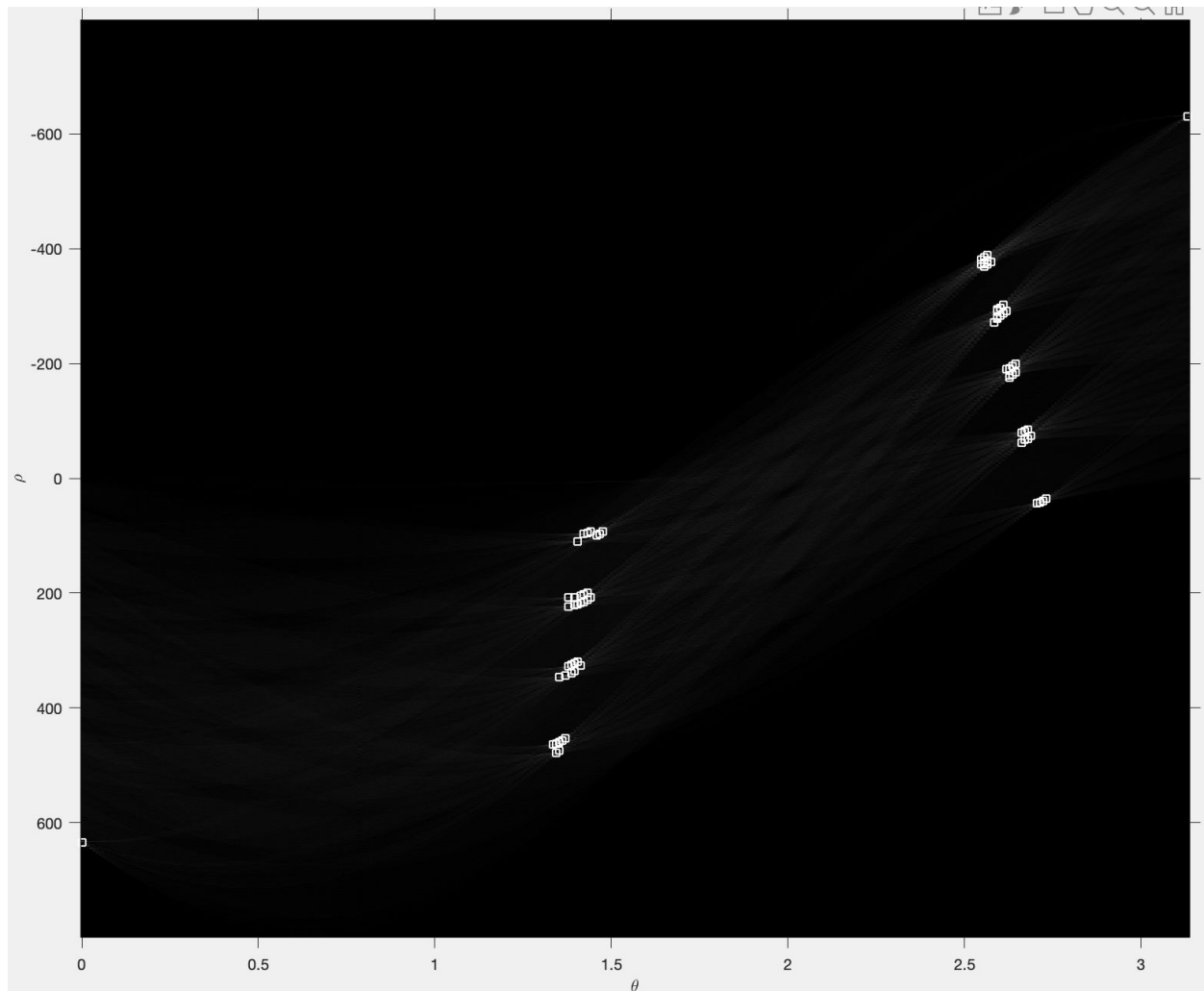

Figure 2.4.1 - Hough Peaks for 75 Lines

2.5 Fitting Line Segments for Visualization

In this part we will be using the 75 lines detected above and overlaying them on top of the original image. The following is the parameters used to achieve these results:

parameters

sigma   = 3;

threshold = 0.1;

rhoRes   = 1;

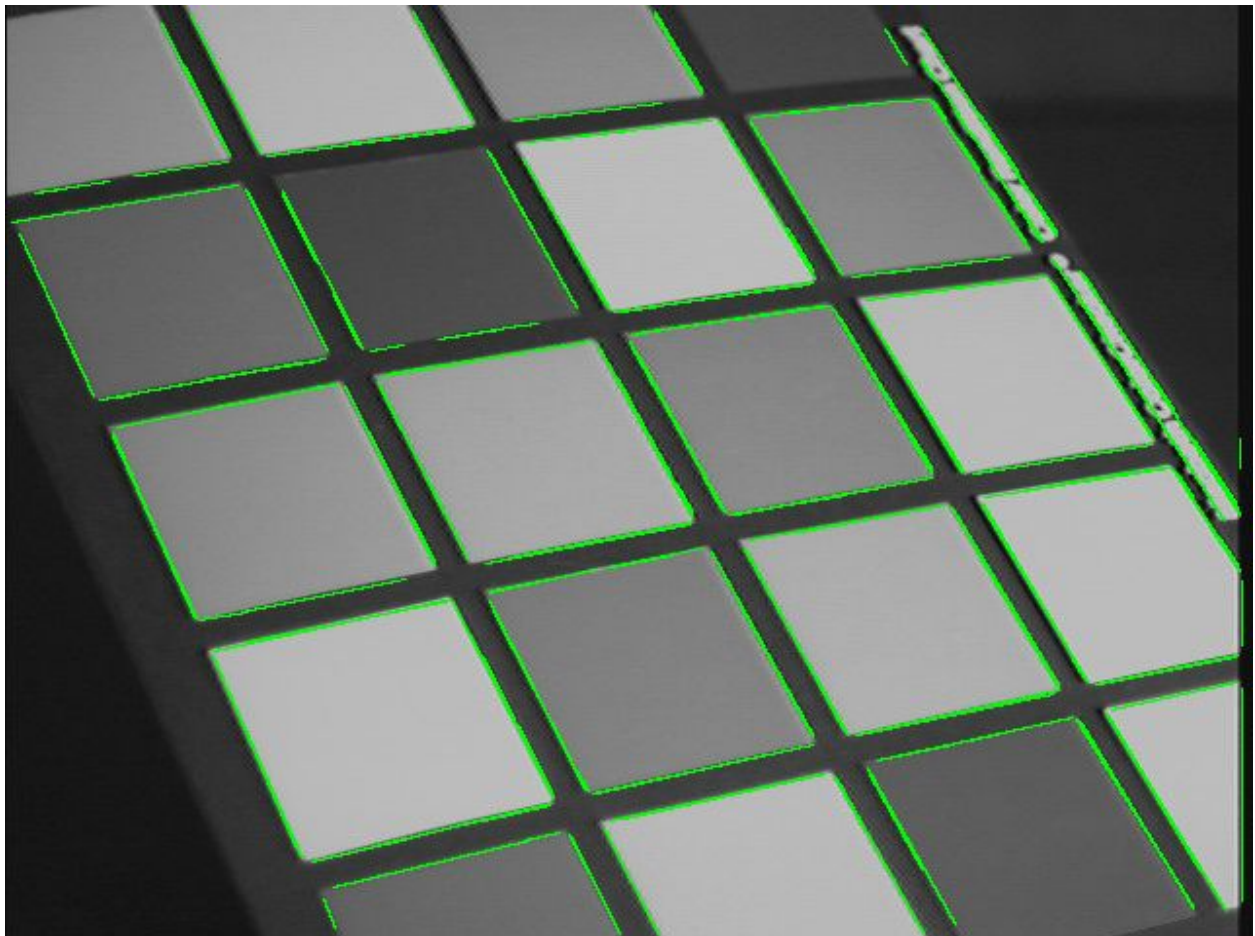thetaRes = pi/360;

nLines   = 75;

Figure 2.5.1 - Hough Lines for 75 Peaks

Below is a set of output images generated by running my code on a different original image. Figure 2.6.1 is the original image. The rest are intermediate images. The Figure labels are shown below each image. The set below is generated using the same set of parameters as above. The results are good enough with the same parameters. However, for different images, tuning the parameters certainly makes a difference. Some images which contain more noise, benefit from a larger sigma for smoothing. The threshold also makes a difference in images with a lot of artifacts or noise. rhoRes and thetaRes don't really need to be tuned for different images. nLines can be tuned to ignore certain lines which are not needed in the image. The hough transform part of the algorithm seems to cause the most problems as it requires most of the parameter tweaking to get right and also takes the longest to finish.

Figure 2.6.1 - Original Image
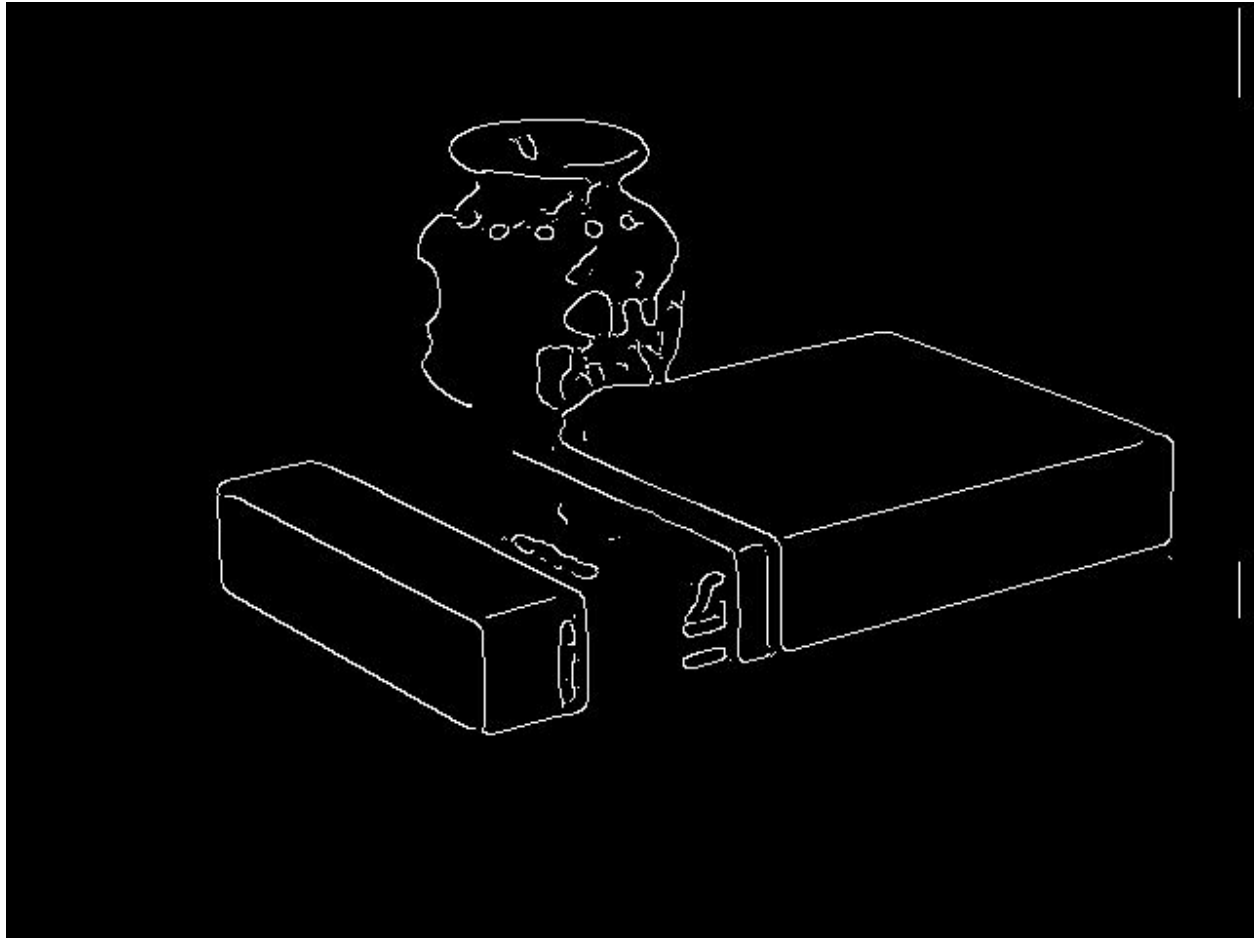
Figure 2.6.2 - Edge
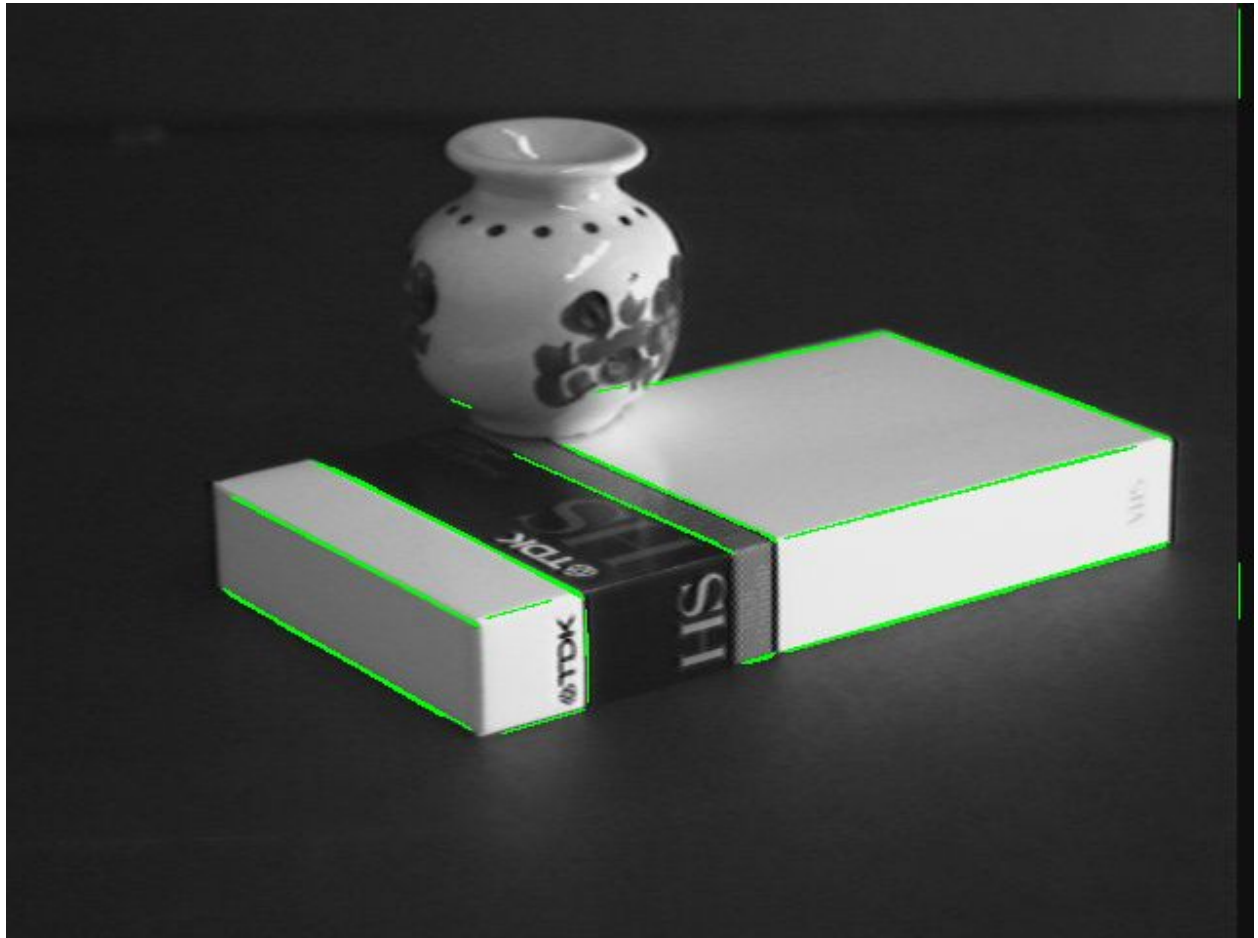
Figure 2.6.3 - Threshold

Figure 2.6.4 - Hough

Figure 2.6.5 - Lines

The following images are 2 images shot by me on an iPhone 6. Figure 2.6.6 through 2.6.10 are for the first image. Figures 2.6.11 through 2.6.15 are for the second image.
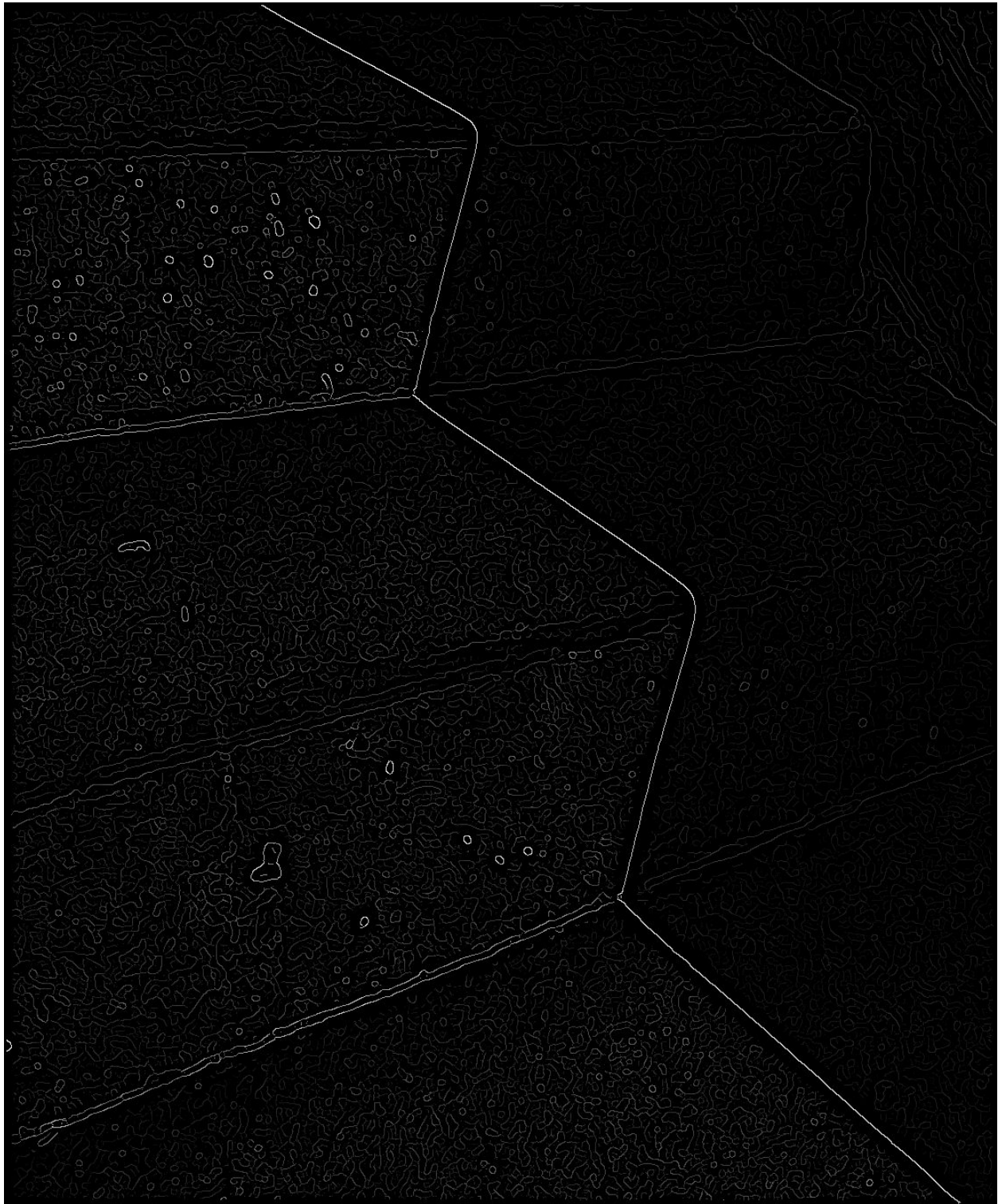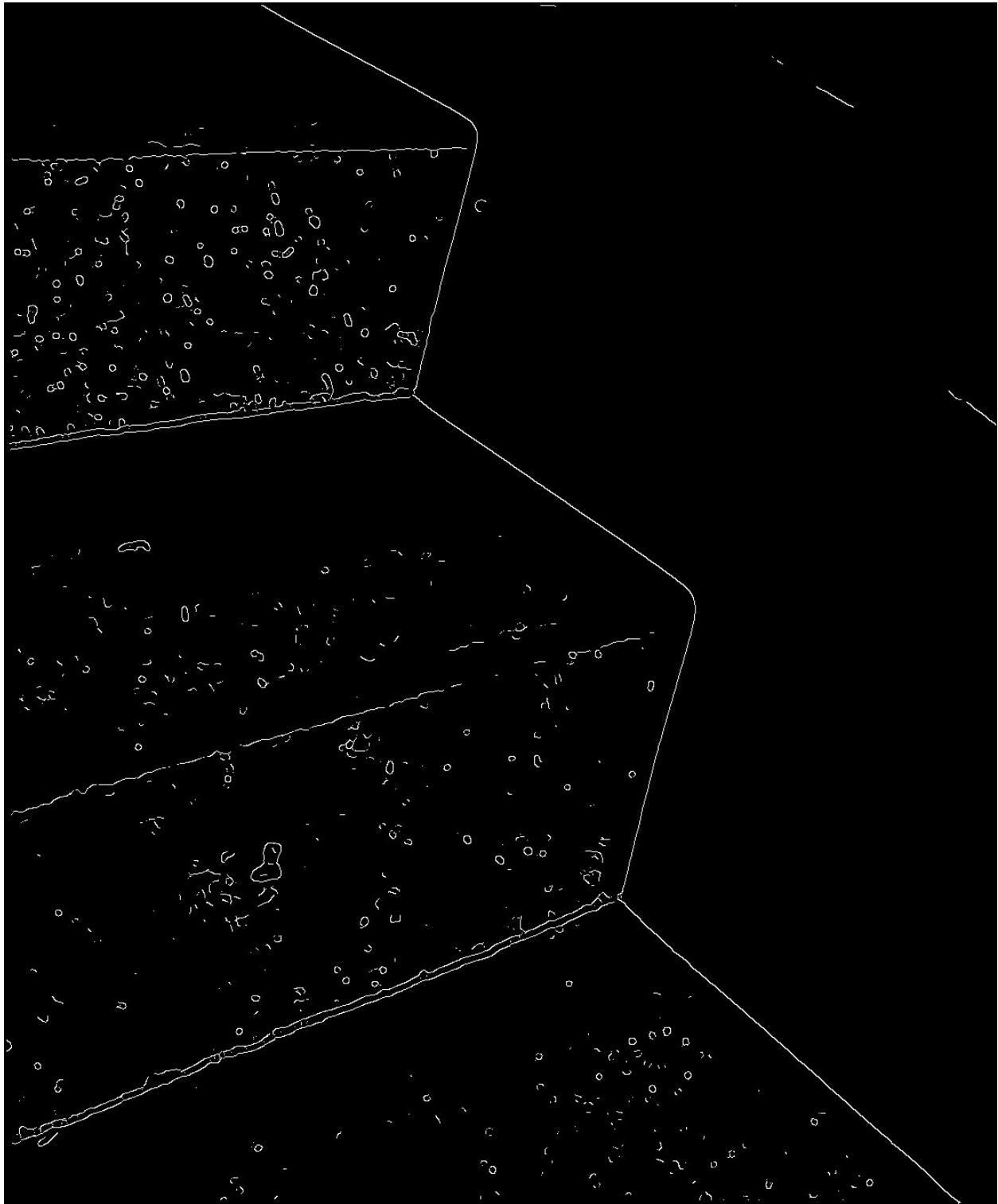
Figure 2.6.6 - Original Image

Figure 2.6.7 - Edge

Figure 2.6.8 - Threshold
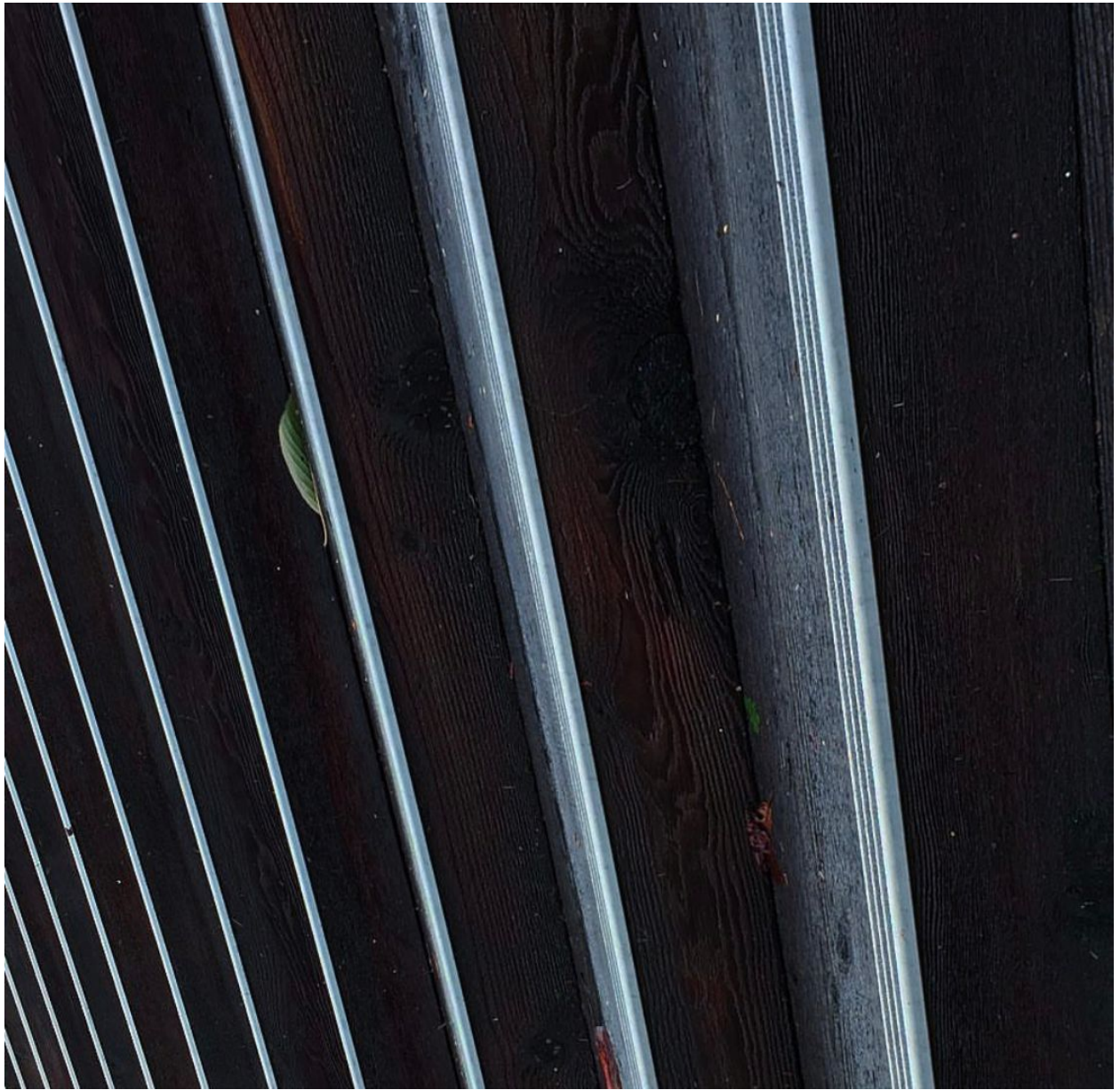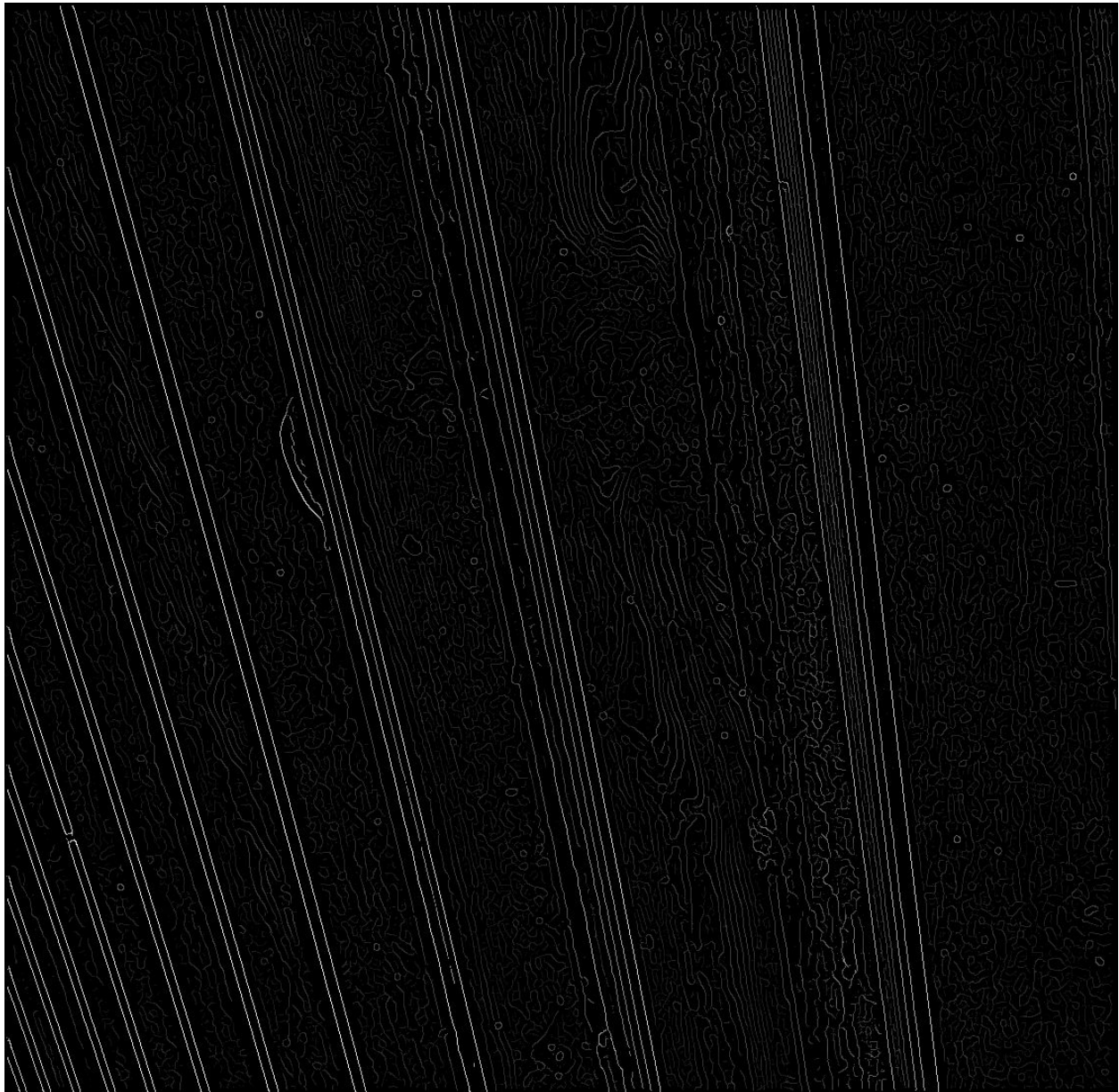
Figure 2.6.9 - Hough

Figure 2.6.10 - Lines
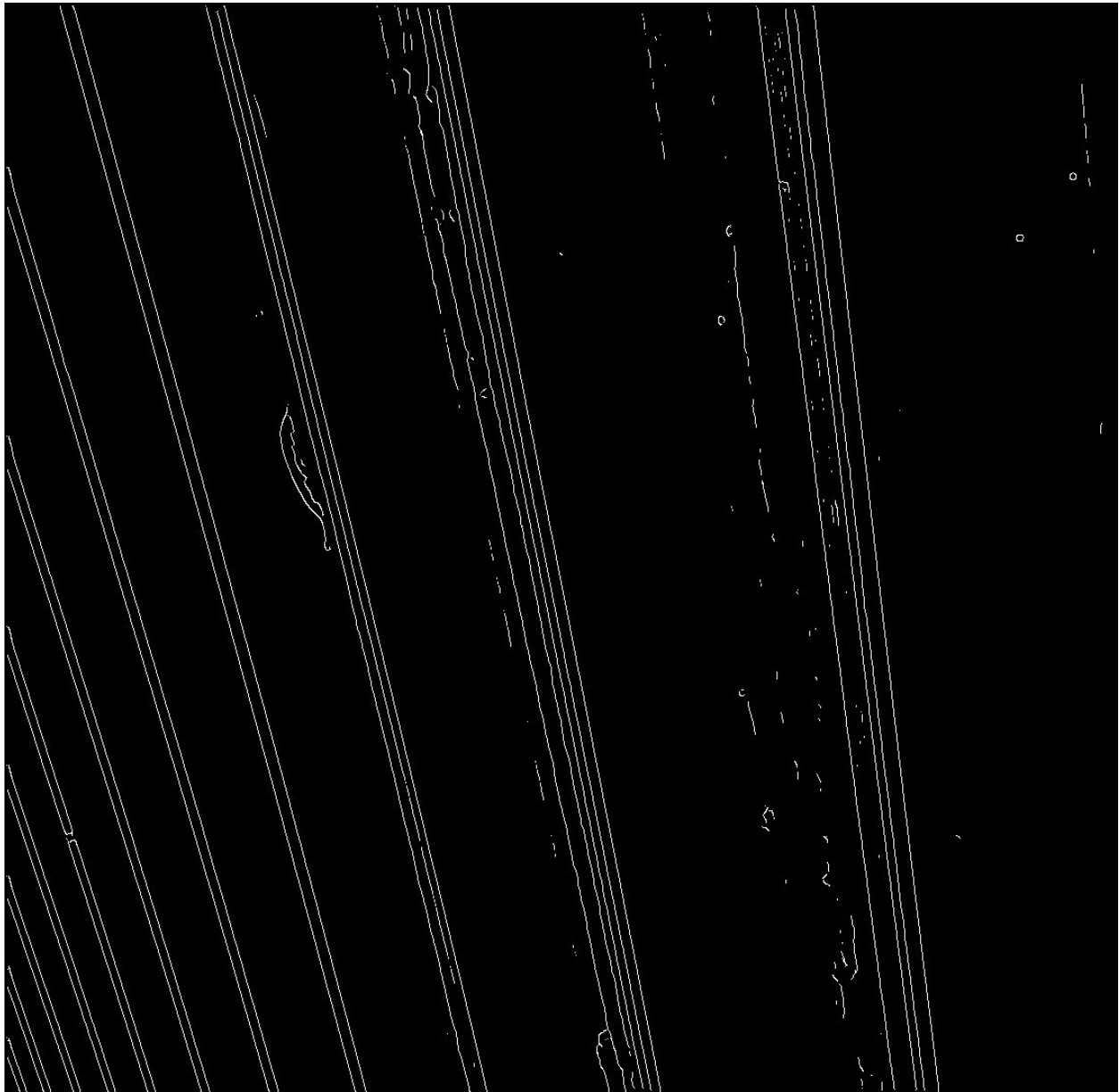
Figure 2.6.11 - Original Image

Figure 2.6.12 - Edge

Figure 2.6.13 - Threshold

Figure 2.6.14 - Hough

Figure 2.6.15 - Lines