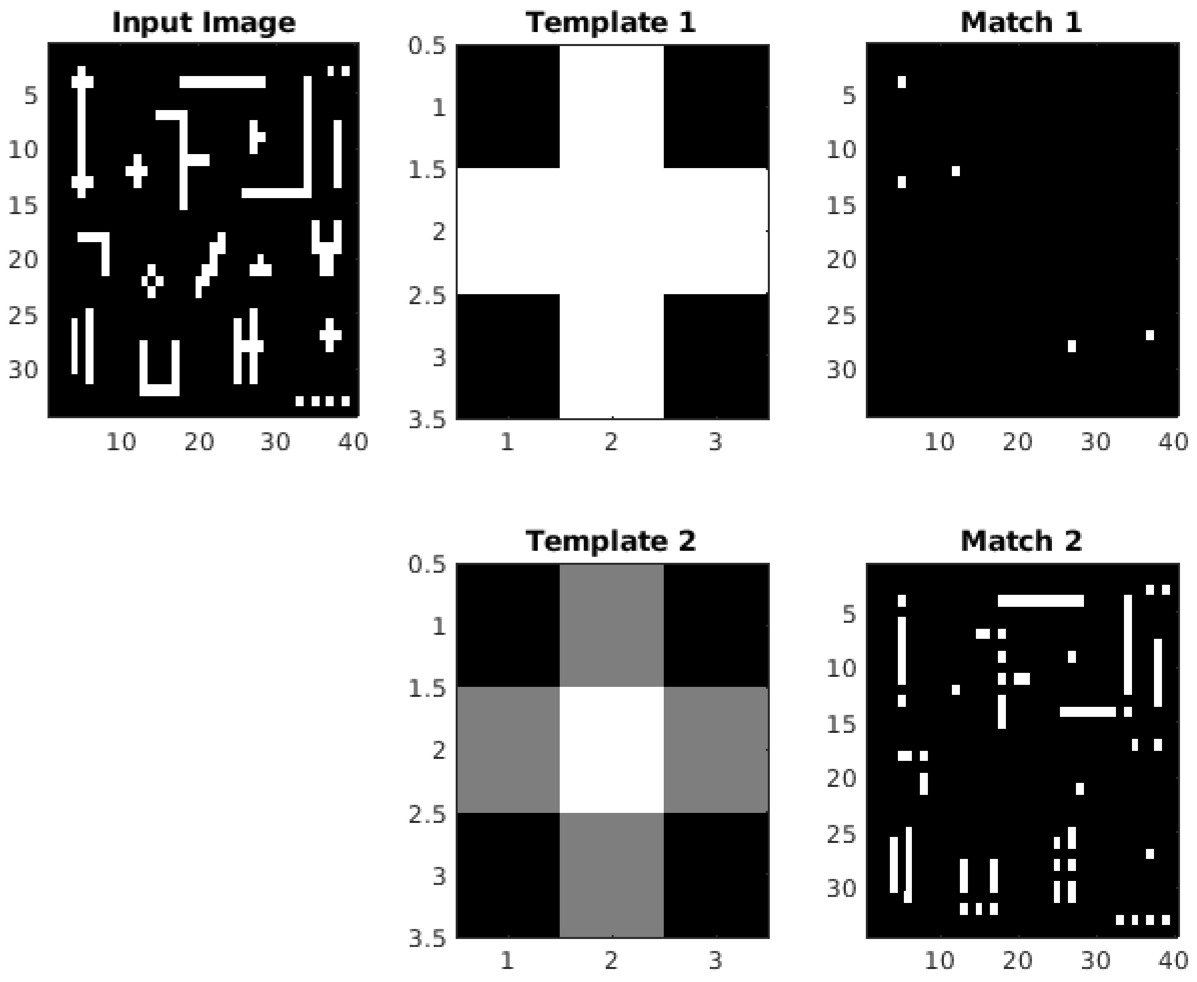
Justin Andre E. Po	Oscar Vian L. Valles

CMSC 178 - A

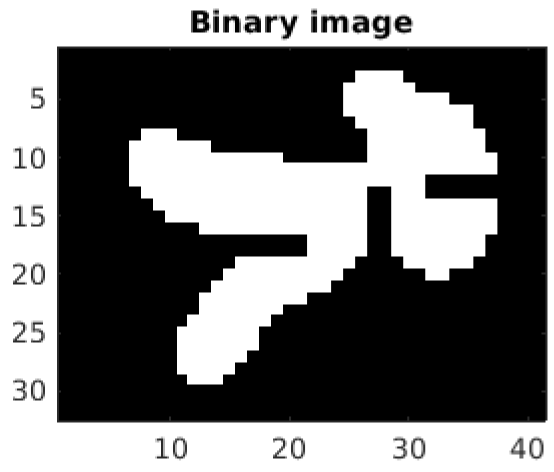
# Assignment 3

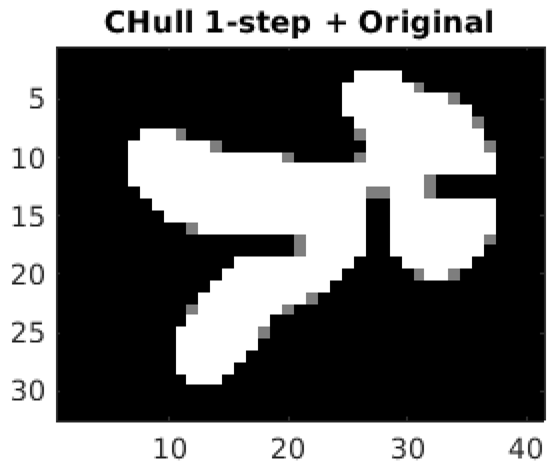
## 3A - Morphological HitorMiss

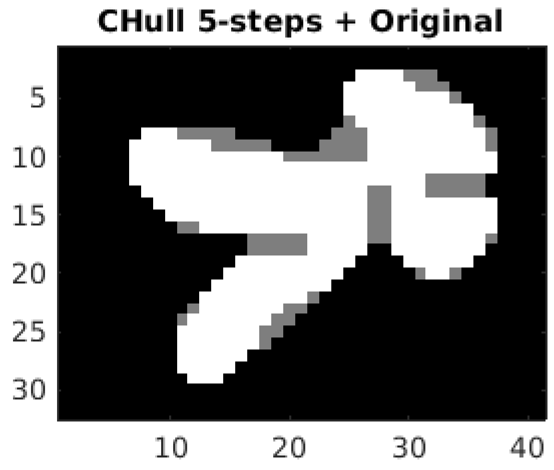
### Results (hitormiss_test.m)

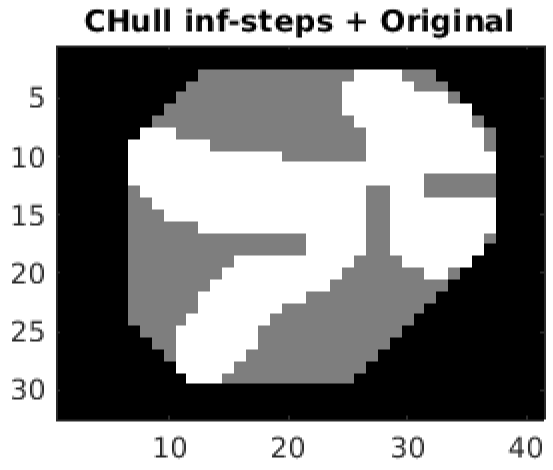


Results (chull_test.m)

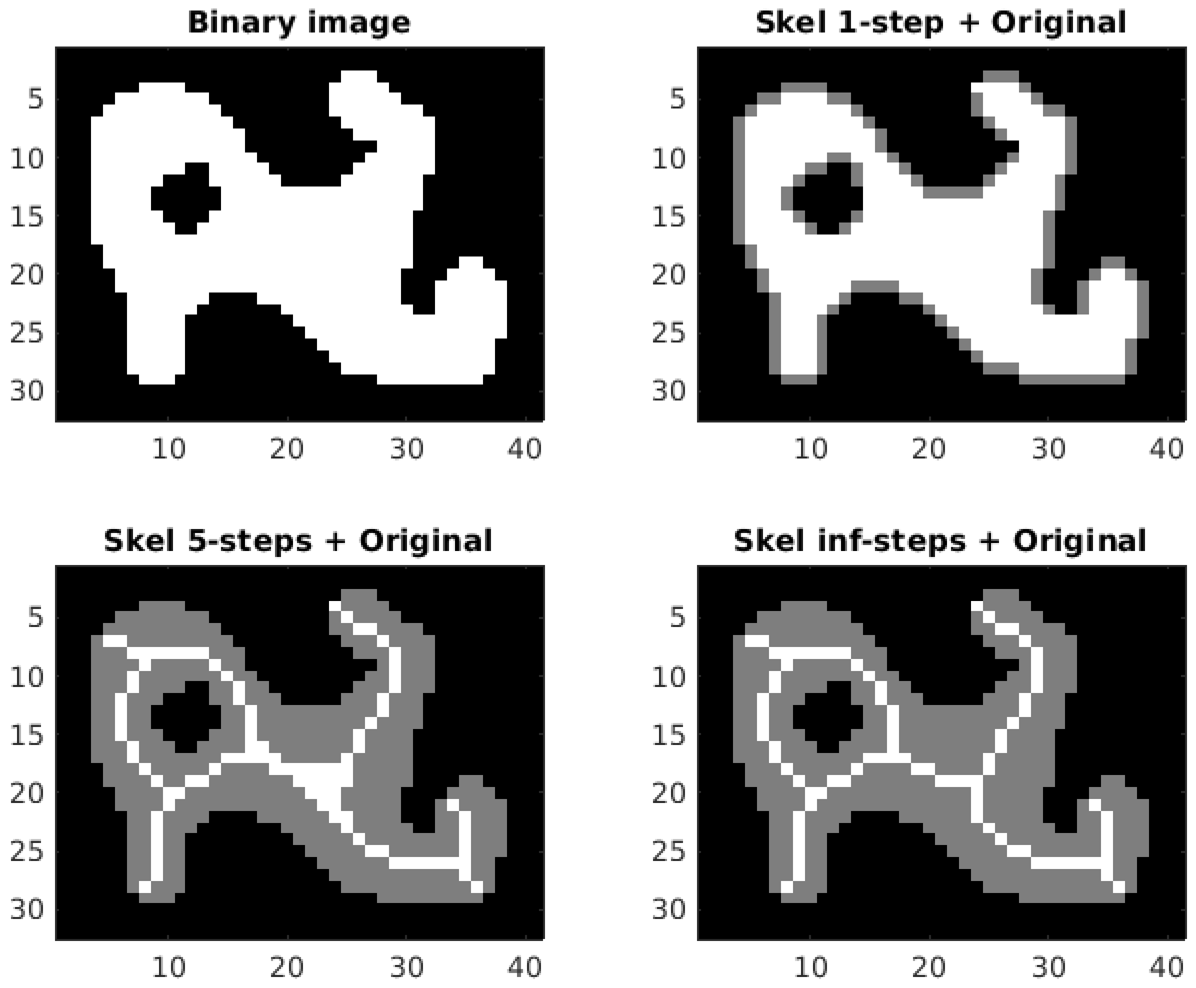








Results (skel_test.m)



## Comments

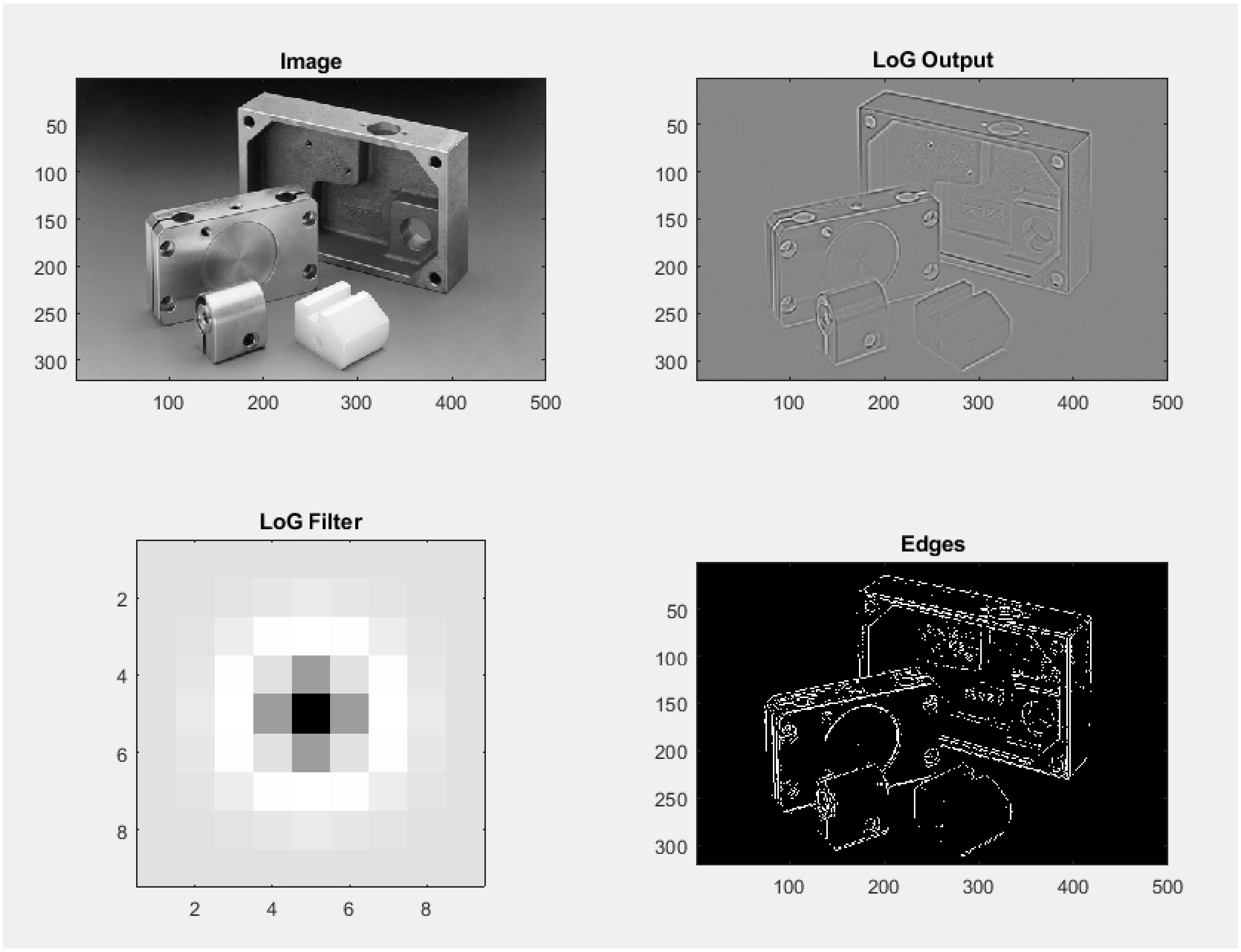
The `hitormiss_3x3` function, when given a binary image and a 3x3 matching template, was able to accomplish its tasks of returning points in the image which match the template. It was also able to handle the two special test cases which implement simple forms of the morphological convex hull and skeletonisation processes. This is shown in the outputs of `chull_test.m` and `skel_test.m`, whose results are identical to the ones provided.

The line that took the most time to run was the one with two `erode_3x3` function calls, which took 66.8% of the `hitormiss_3x3` function's total running time (0.015s). This is because the algorithm erodes each pixel twice, once for the set template and another for the unset template to be able to determine if that pixel should be part of the final image.

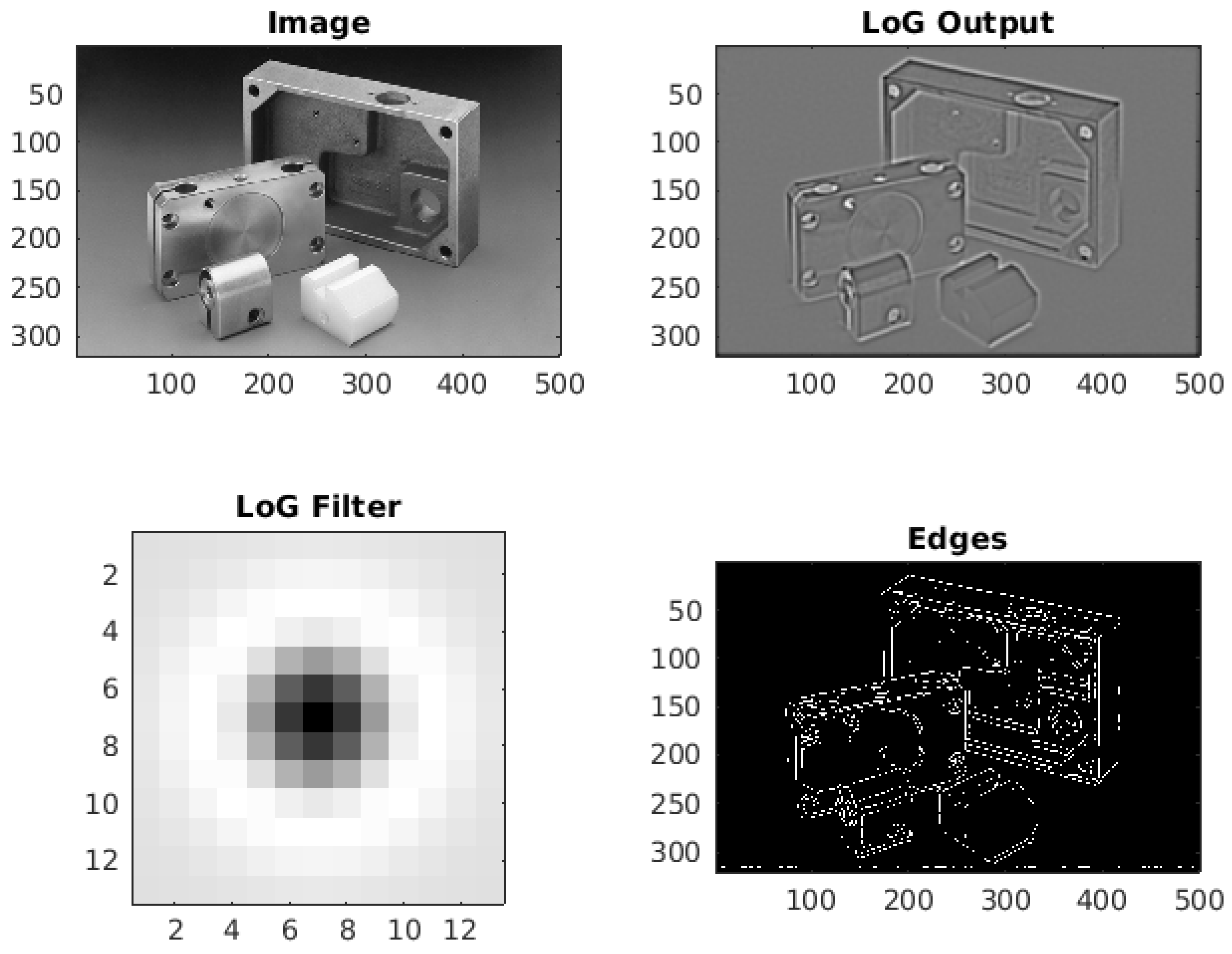
When running the `erode_3x3` function, on the other hand, the line that took the most time was the conditional statement `if S(i) == 1 && I(i) ~= 1` that checks if there are any pixels that are not set in the array when it should be set according to the template. This took the longest because this line runs for each pixel in the given sample in the worst case.

## 3B - Laplacian of Gaussians Edge Detection

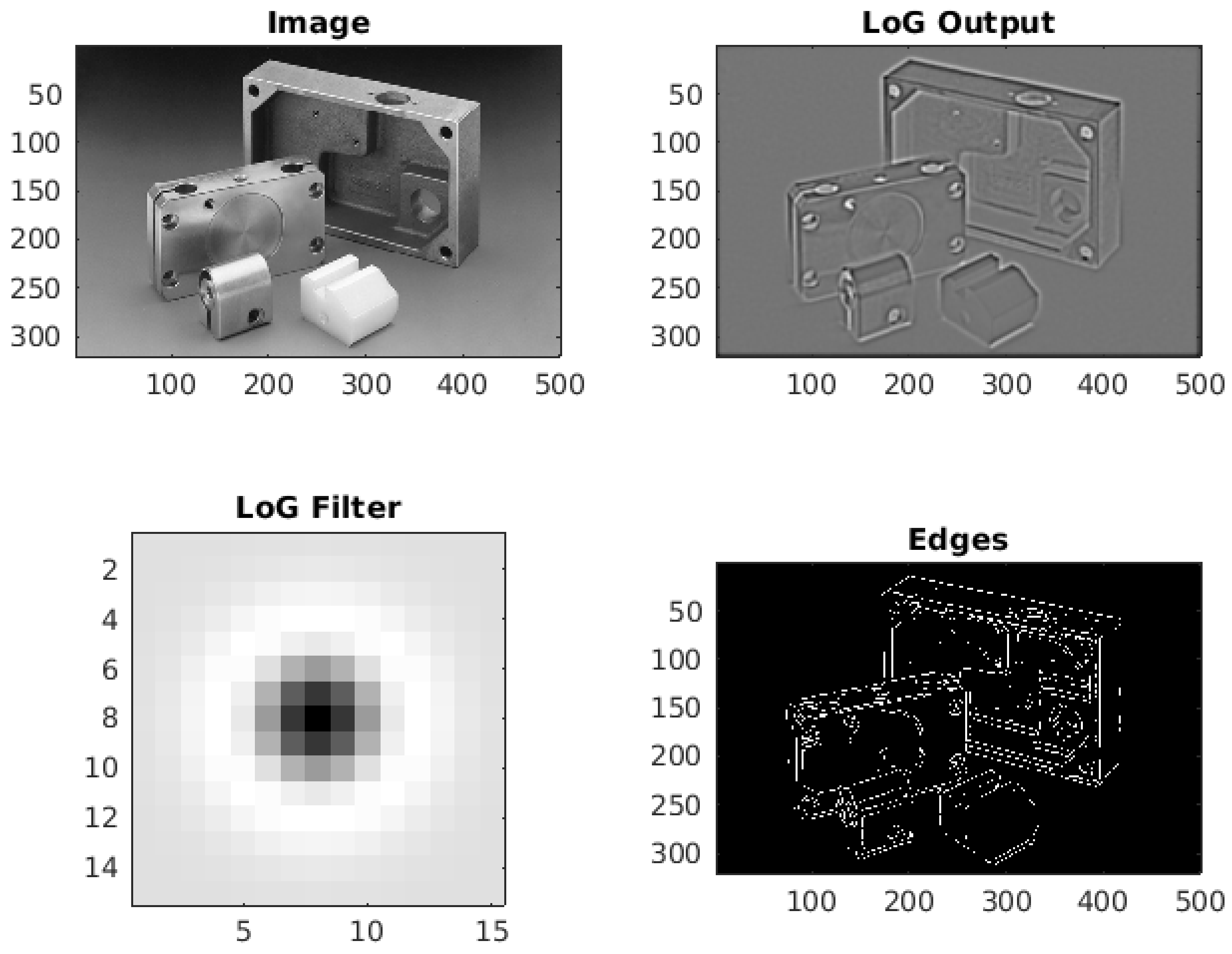
### Results



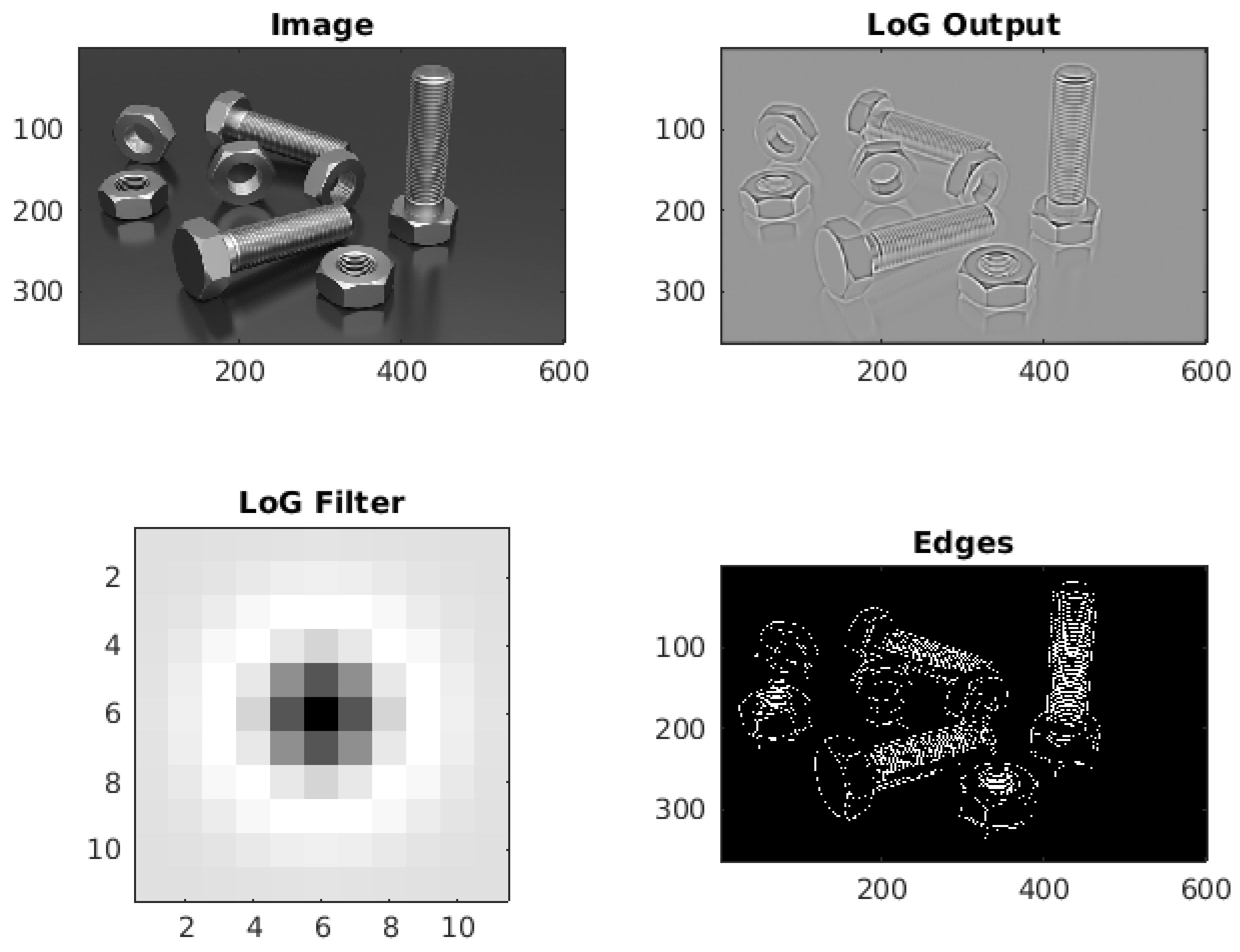
*components.tif with a 9x9 filter*



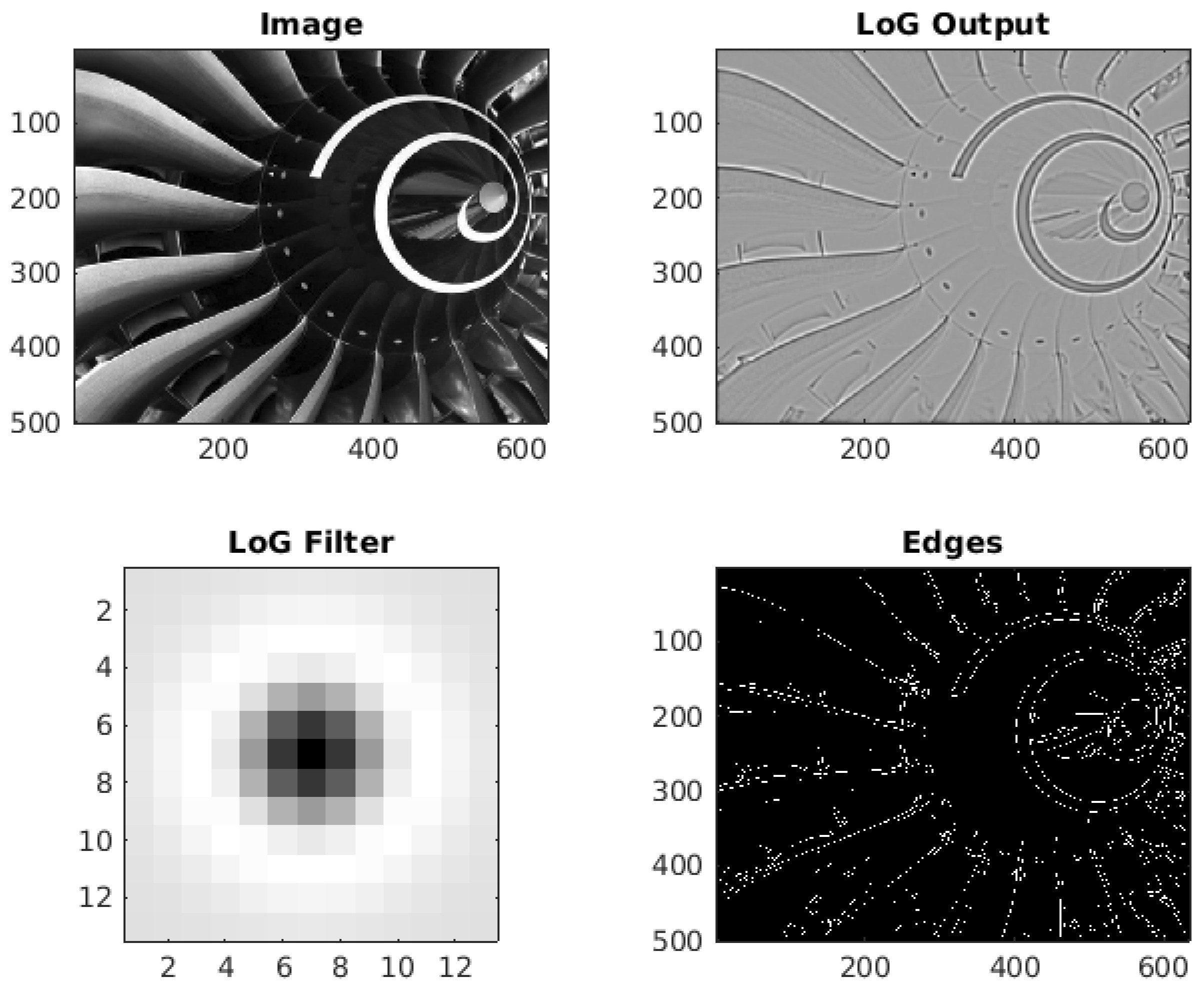
*components.tif with a 13x13 filter*



*components.tif with a 15x15 filter*



*bolts.tif with an 11x11 filter*



*Engine.tif with a 13x13 filter*

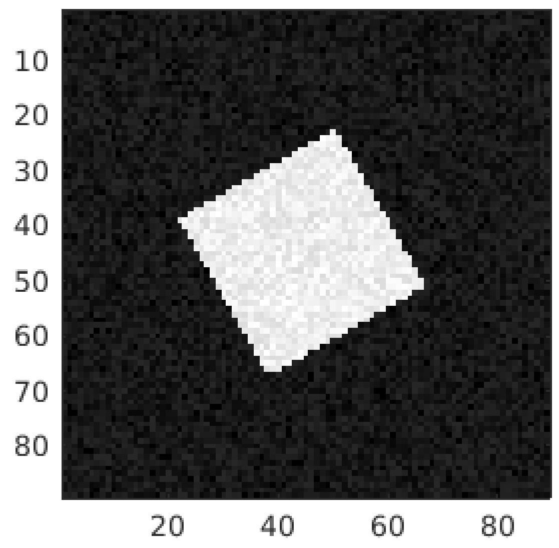
## Comments

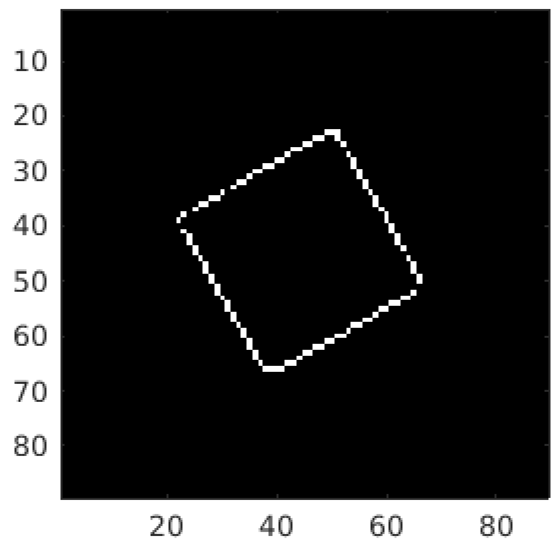
The `log_edge` function was able to successfully obtain the edges of the image based on the Laplacian of Gaussians filter. Increasing the filter size results in less zero crossings present in the final edge. Adjusting the filter for each image is a requirement to be able to get better accuracy. A filter that is too small may introduce noise in the edge image, and a filter too large may result in edges that are incomplete.

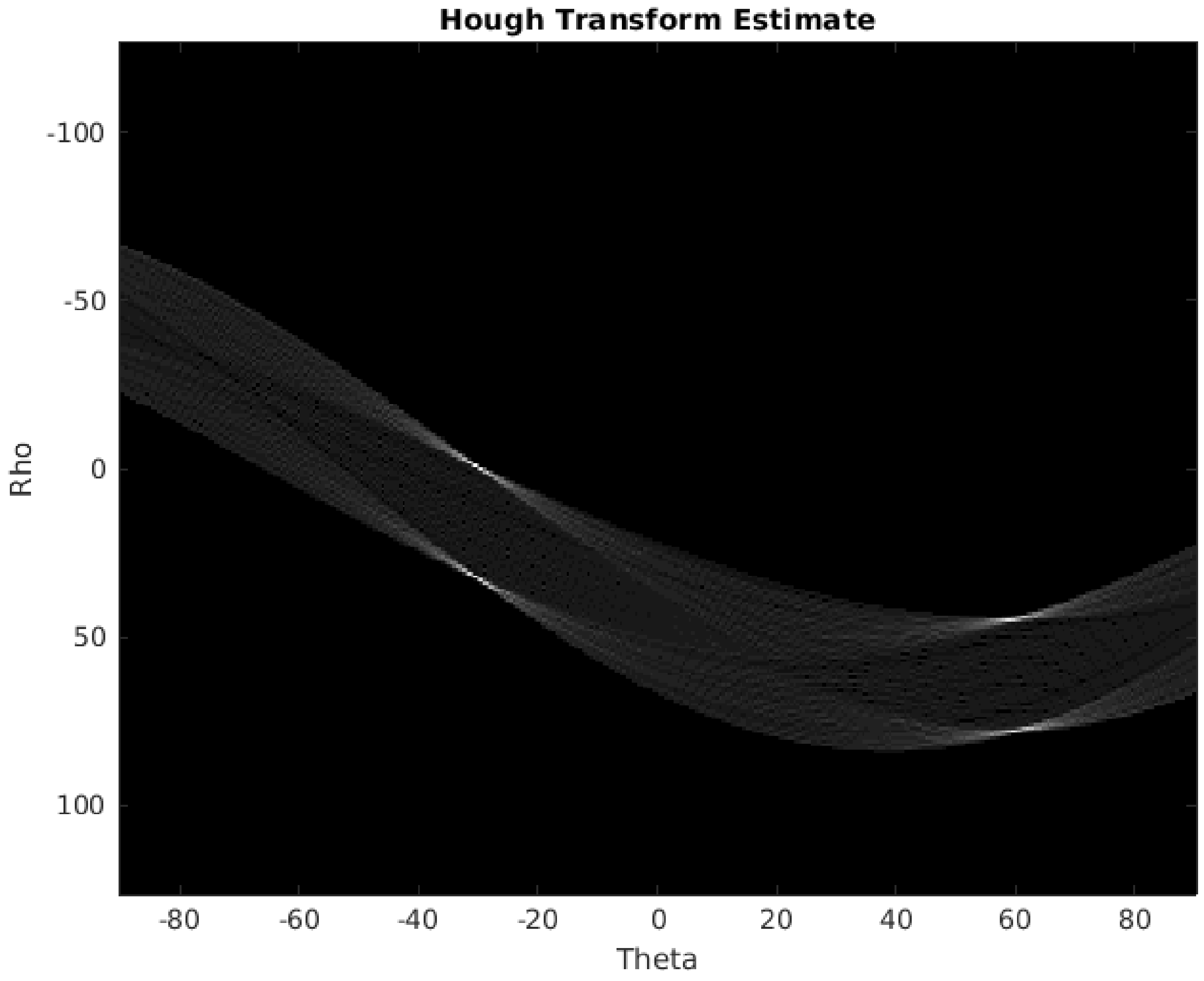
When running the `log_test` function, `log_edge` took 27.1% of the total running time of the whole function. Looking deeper into `log_edge`, identifying if the current pixel is a zero crossing took the most time, accounting for 15.1% of the total running time of `log_edge`. This was because a comparison of the pixel and its four sides was required in the worst case. In addition, this means that each pixel needs to be iterated through. The next two lines that took the most time were the generation of the LoG filter and the convolution of the LoG filter with the image.

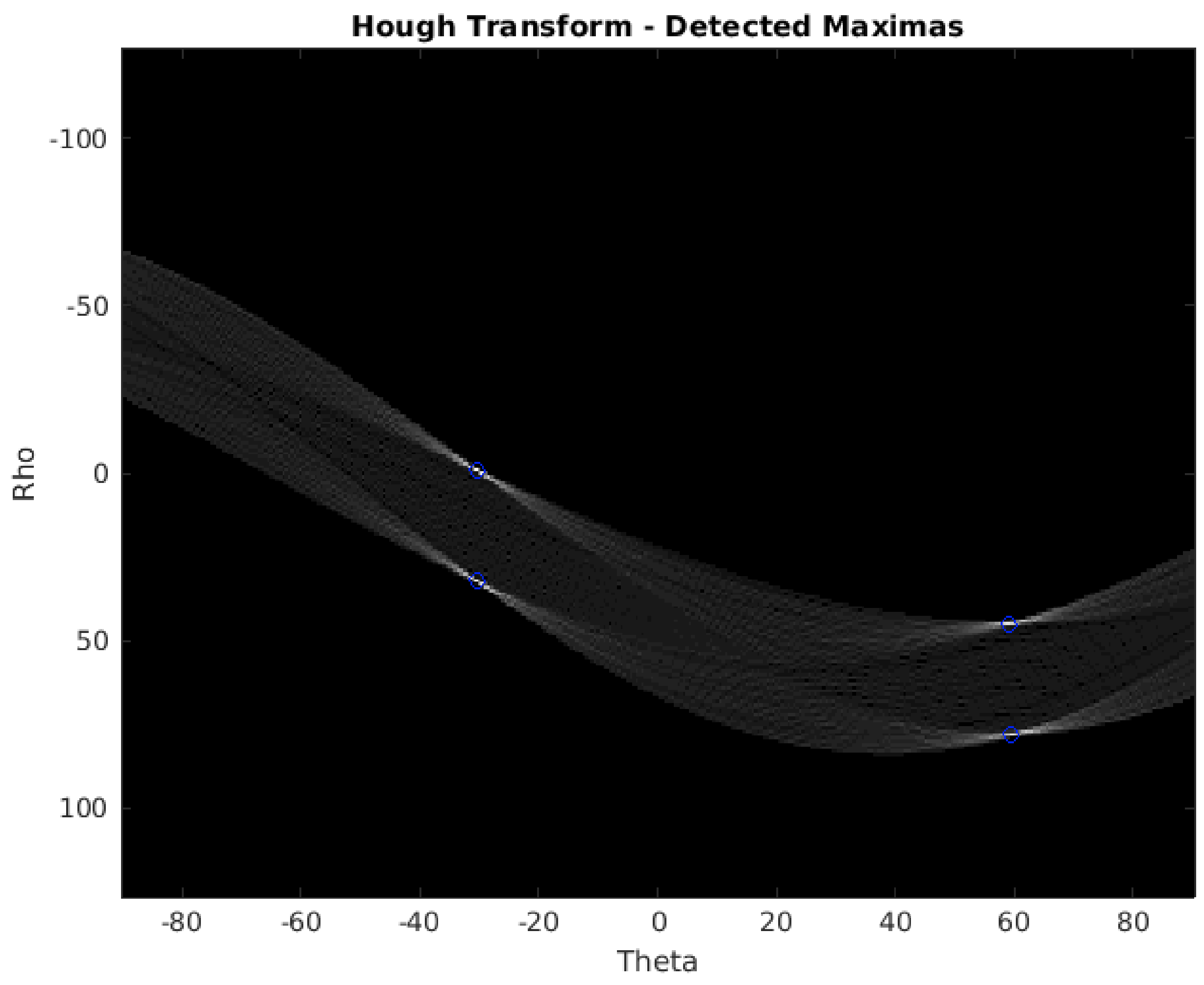
## 3C - Hough Transform Line Estimates

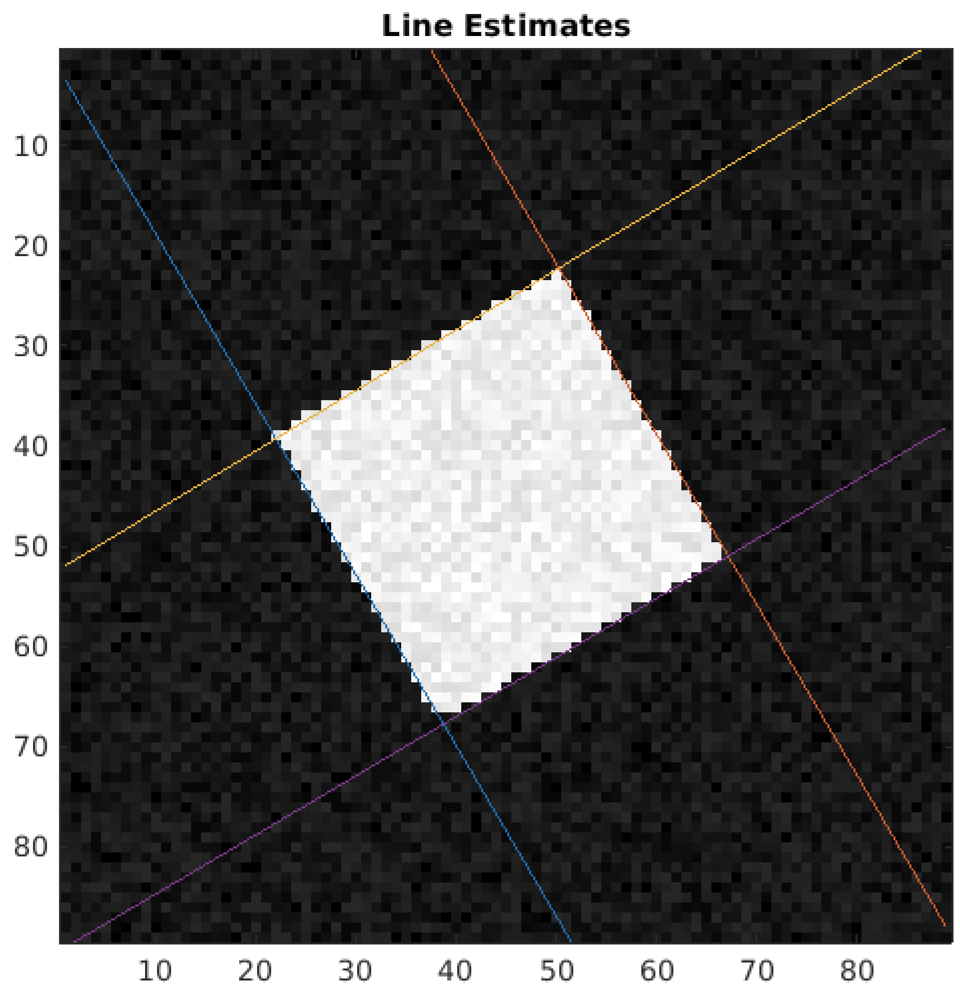
### Box Image (4 lines)



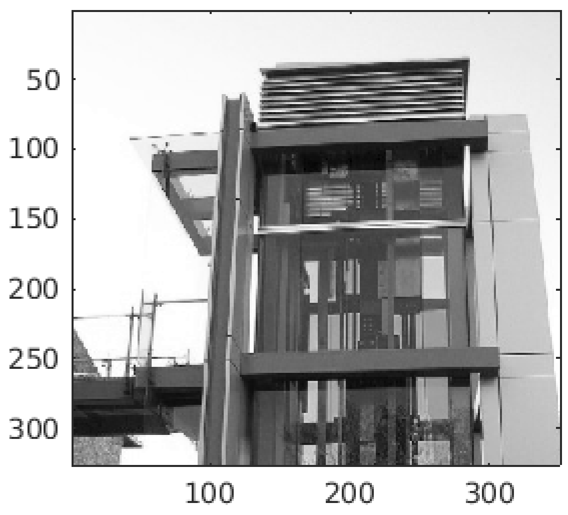


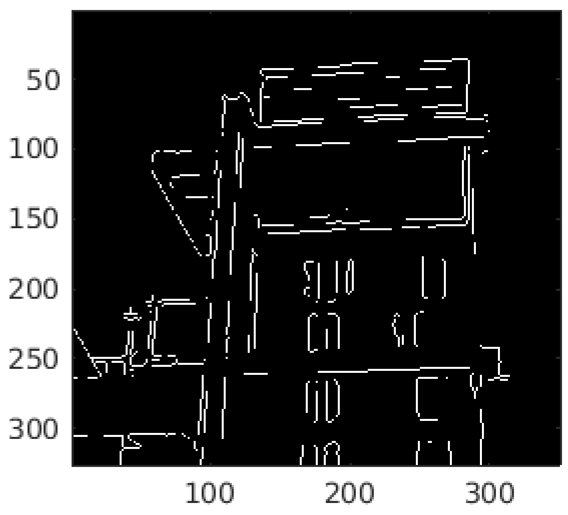

Hough Transform Estimate
Rho
Theta
-100
-50
0
50
100
-80
-60
-40
-20
0
20
40
60
80

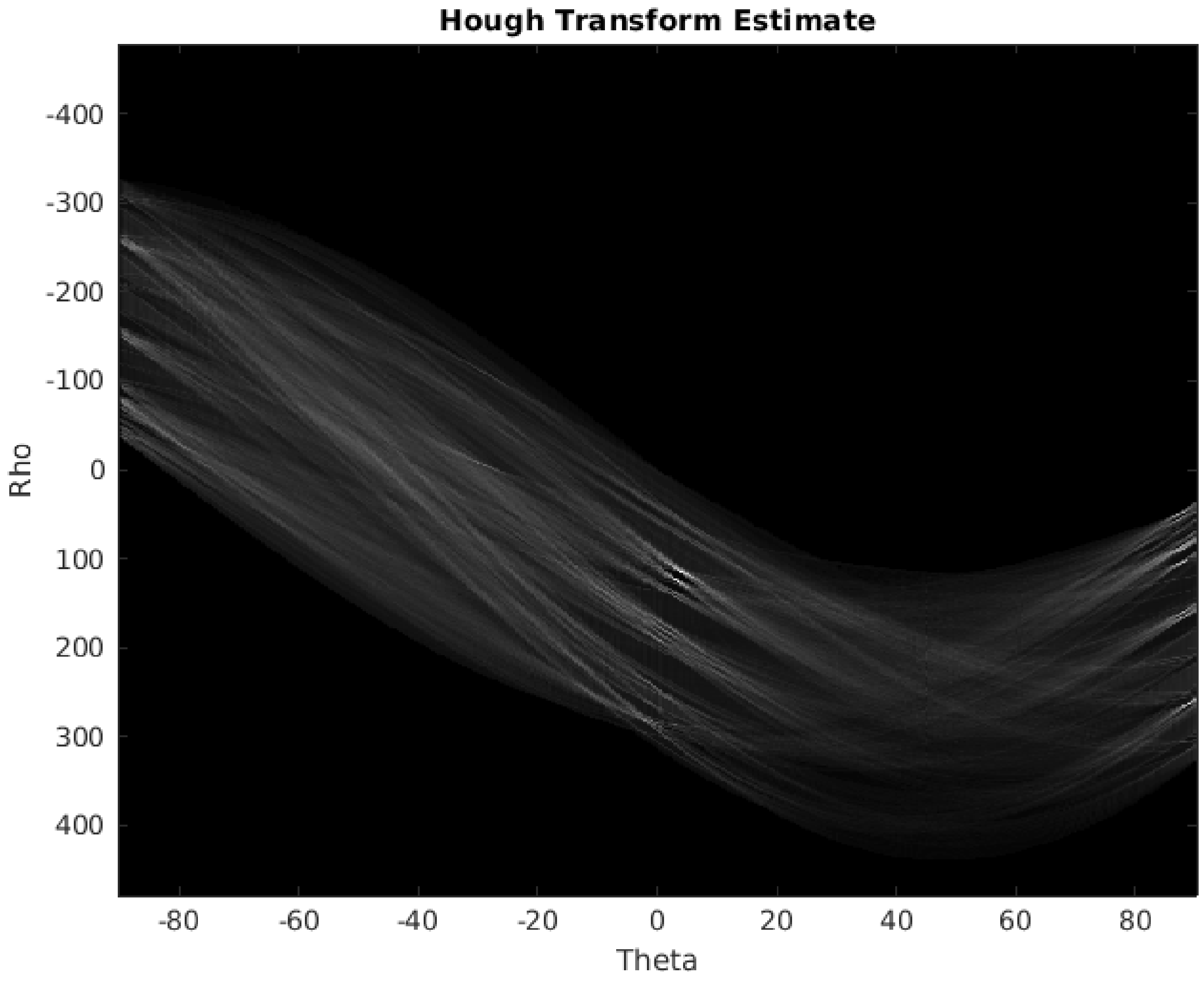

Hough Transform - Detected Maximas
Rho
-100
-50
0
50
100
Theta
-80
-60
-40
-20
0
20
40
60
80

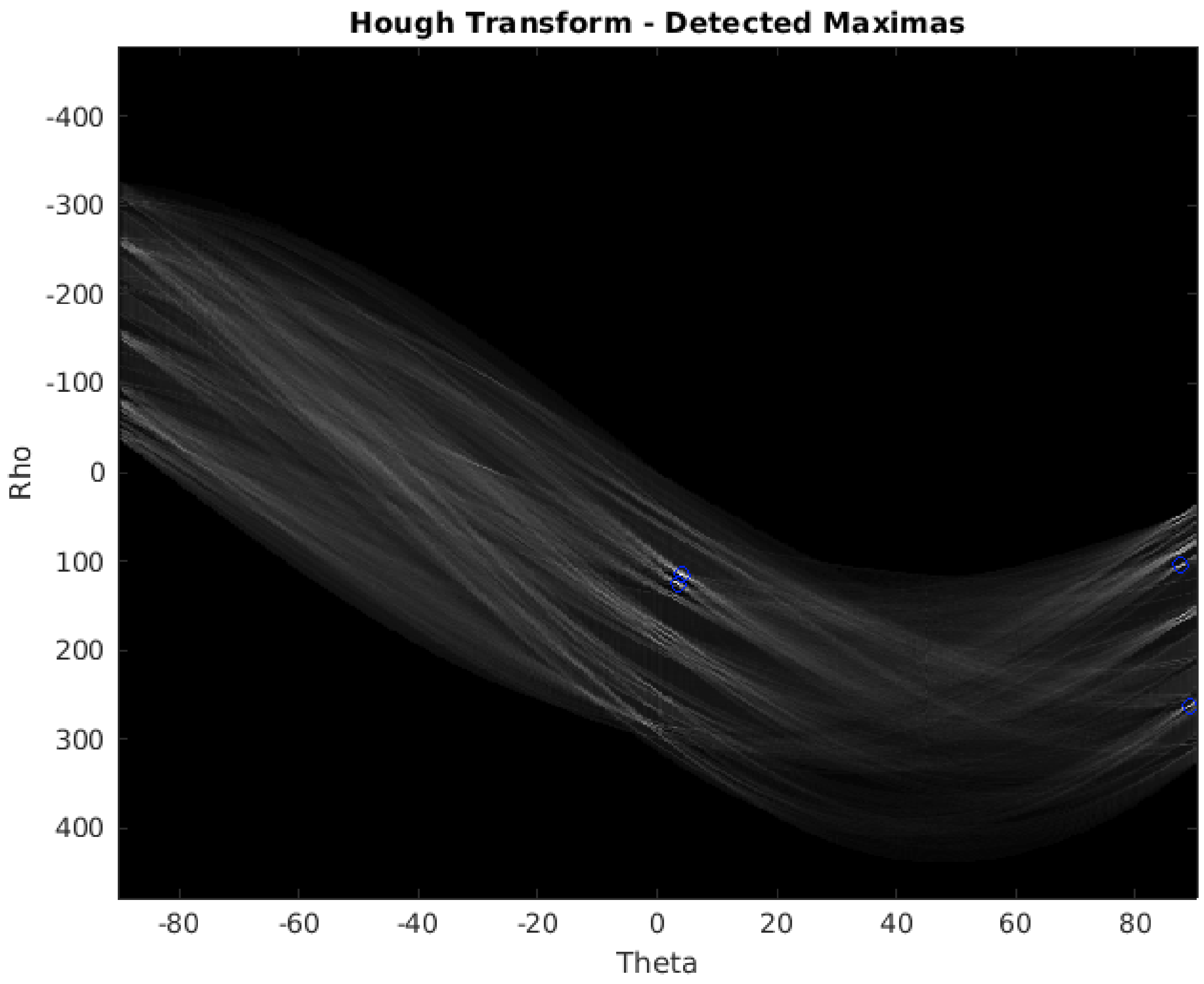

Line Estimates
10
20
30
40
50
60
70
80
10
20
30
40
50
60
70
80

Lift Edge Image (4 lines)

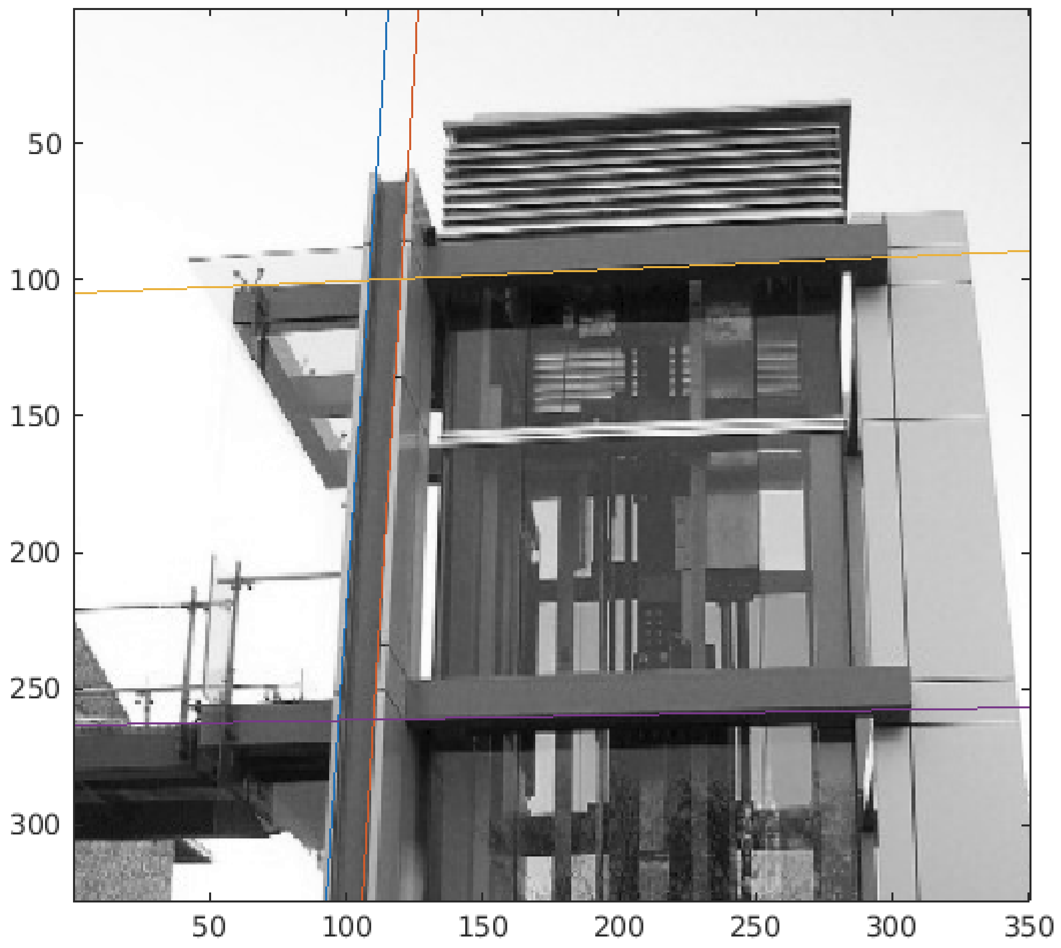




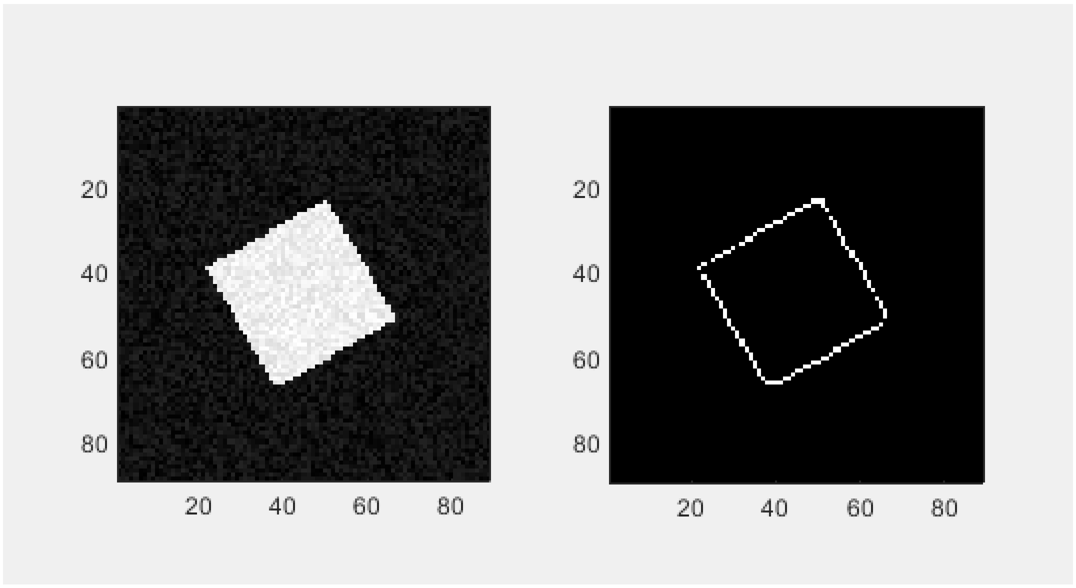


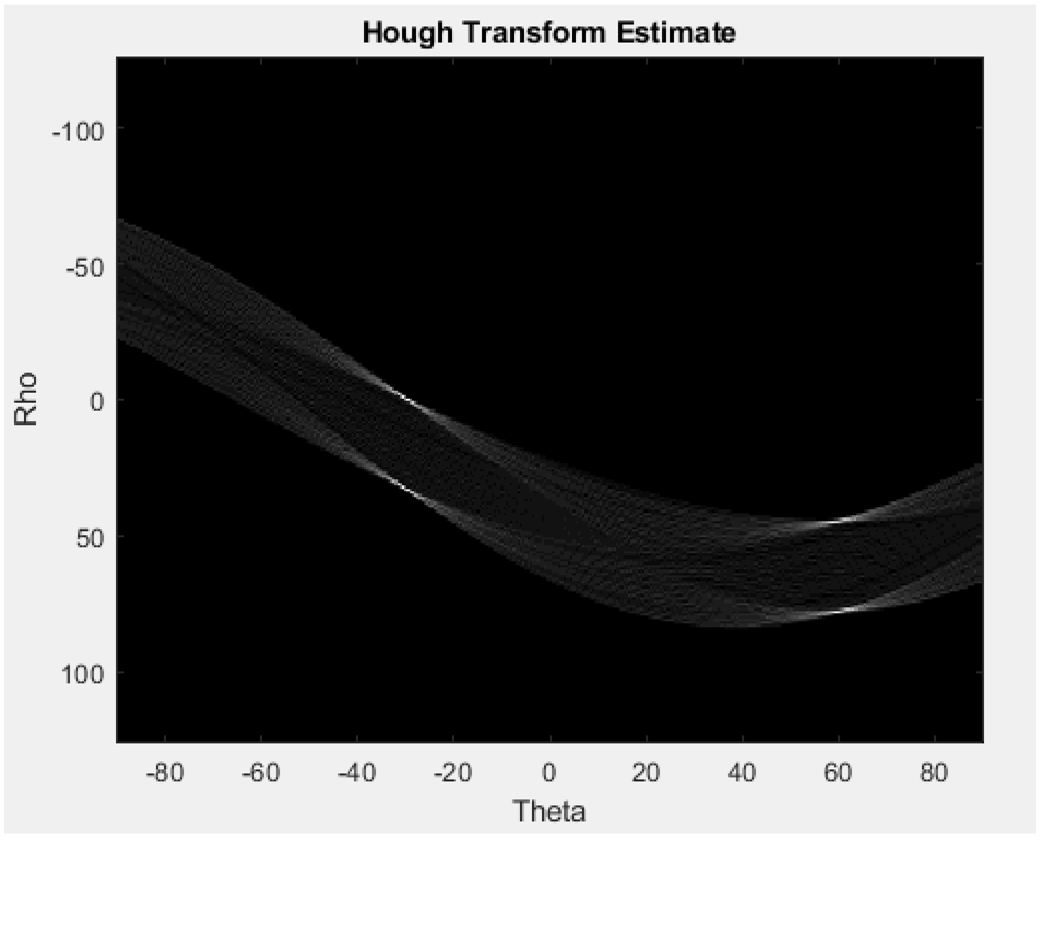

Hough Transform - Detected Maximas
Rho
Theta
-400
-300
-200
-100
0
100
200
300
400
-80
-60
-40
-20
0
20
40
60
80

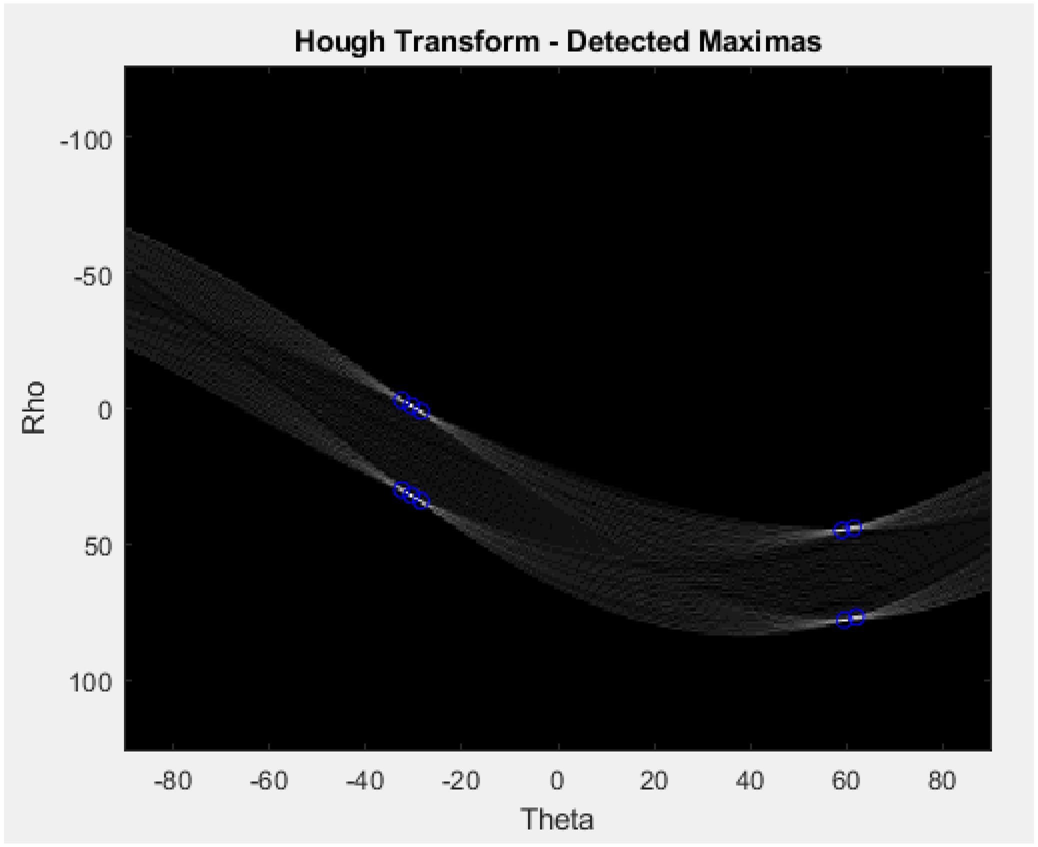
**Line Estimates**

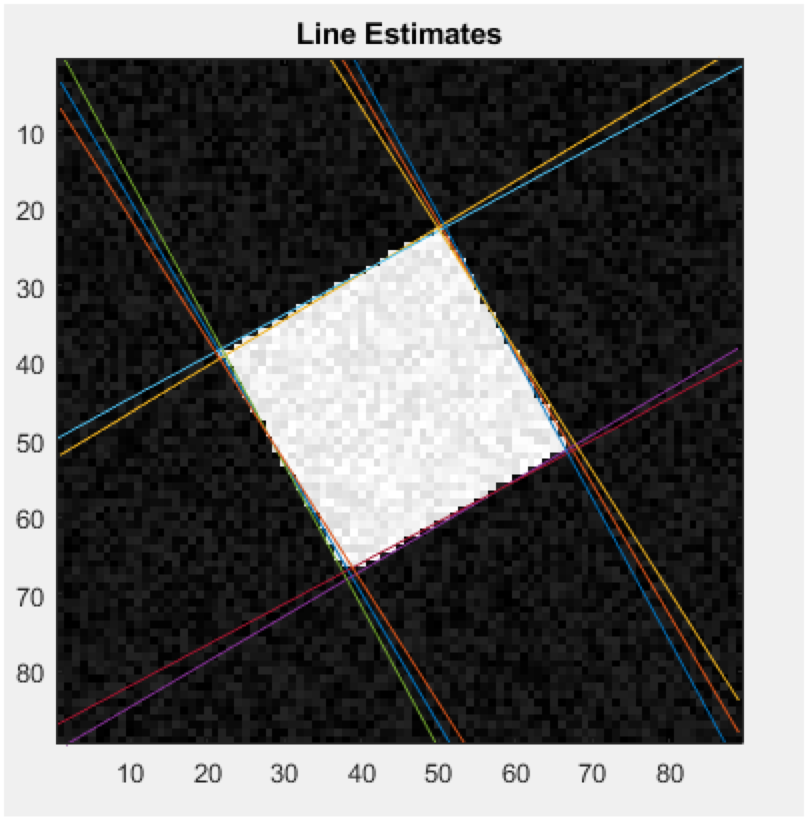


Box Image (10 lines)

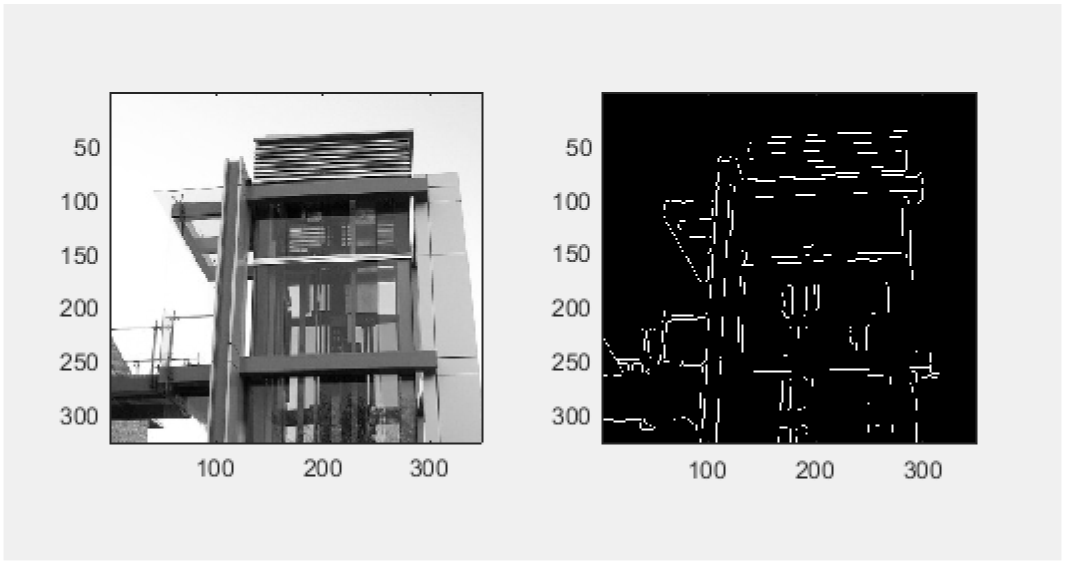


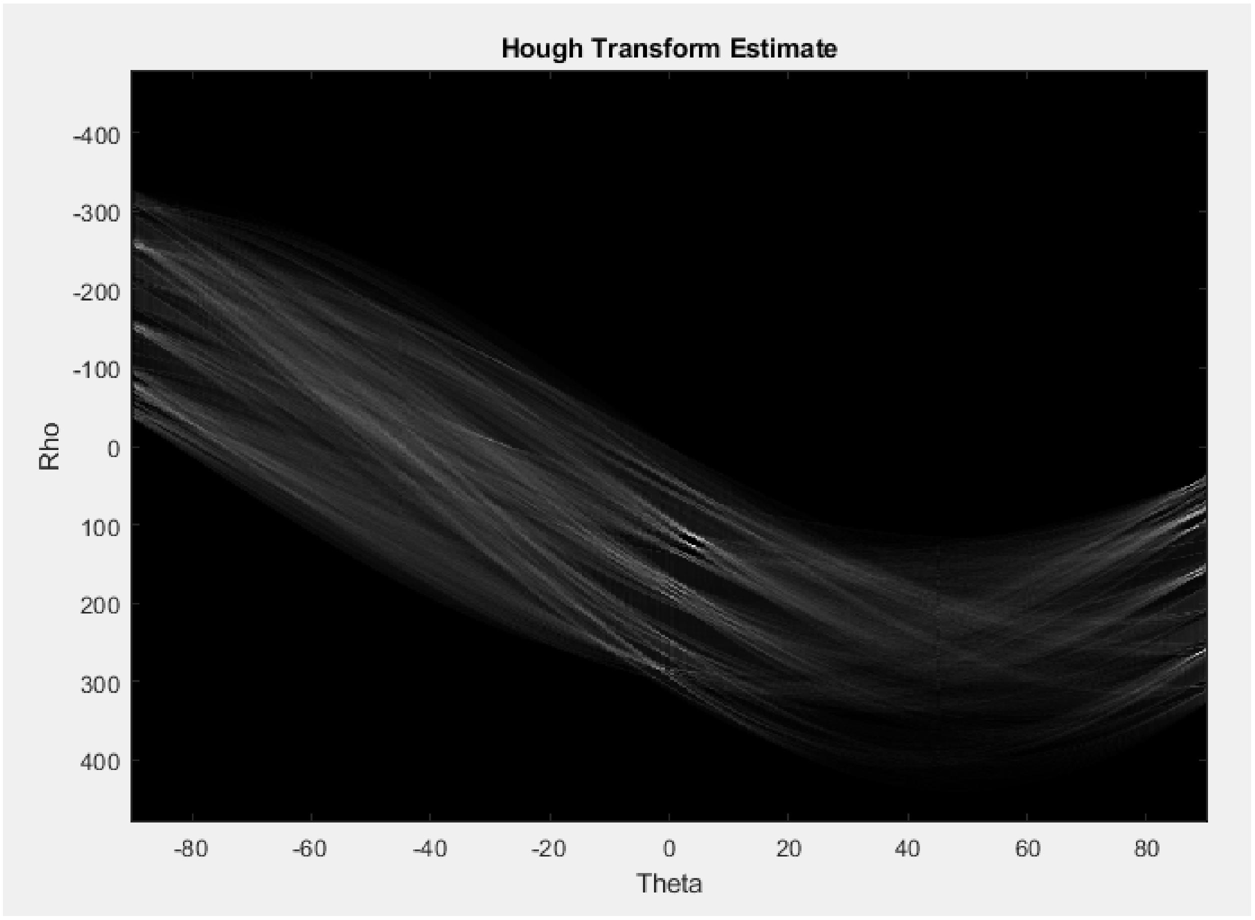


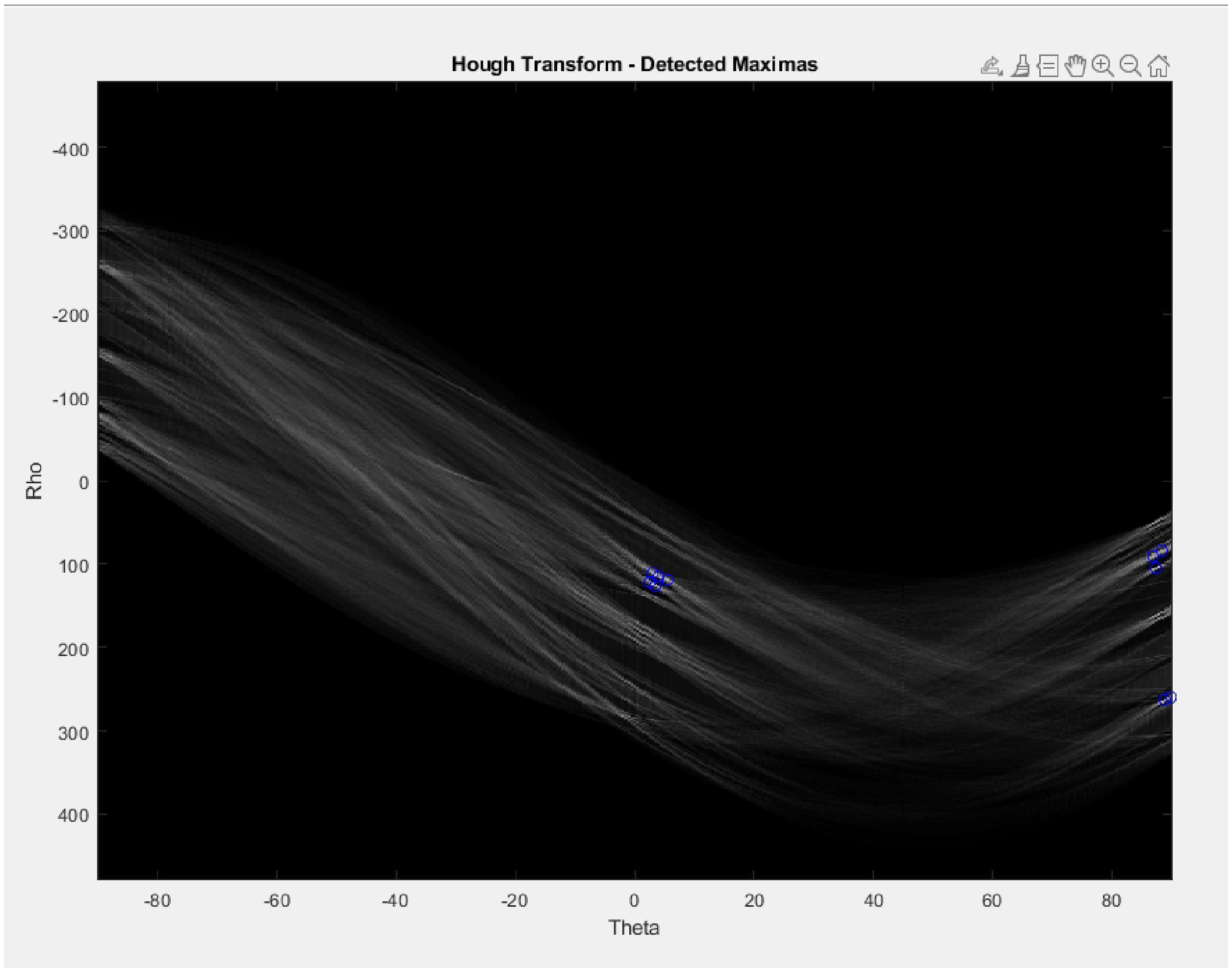

Hough Transform - Detected Maximas
Rho
-100
-50
0
50
100
-80
-60
-40
-20
0
20
40
60
80
Theta


Line Estimates
10
20
30
40
50
60
70
80
10
20
30
40
50
60
70
80

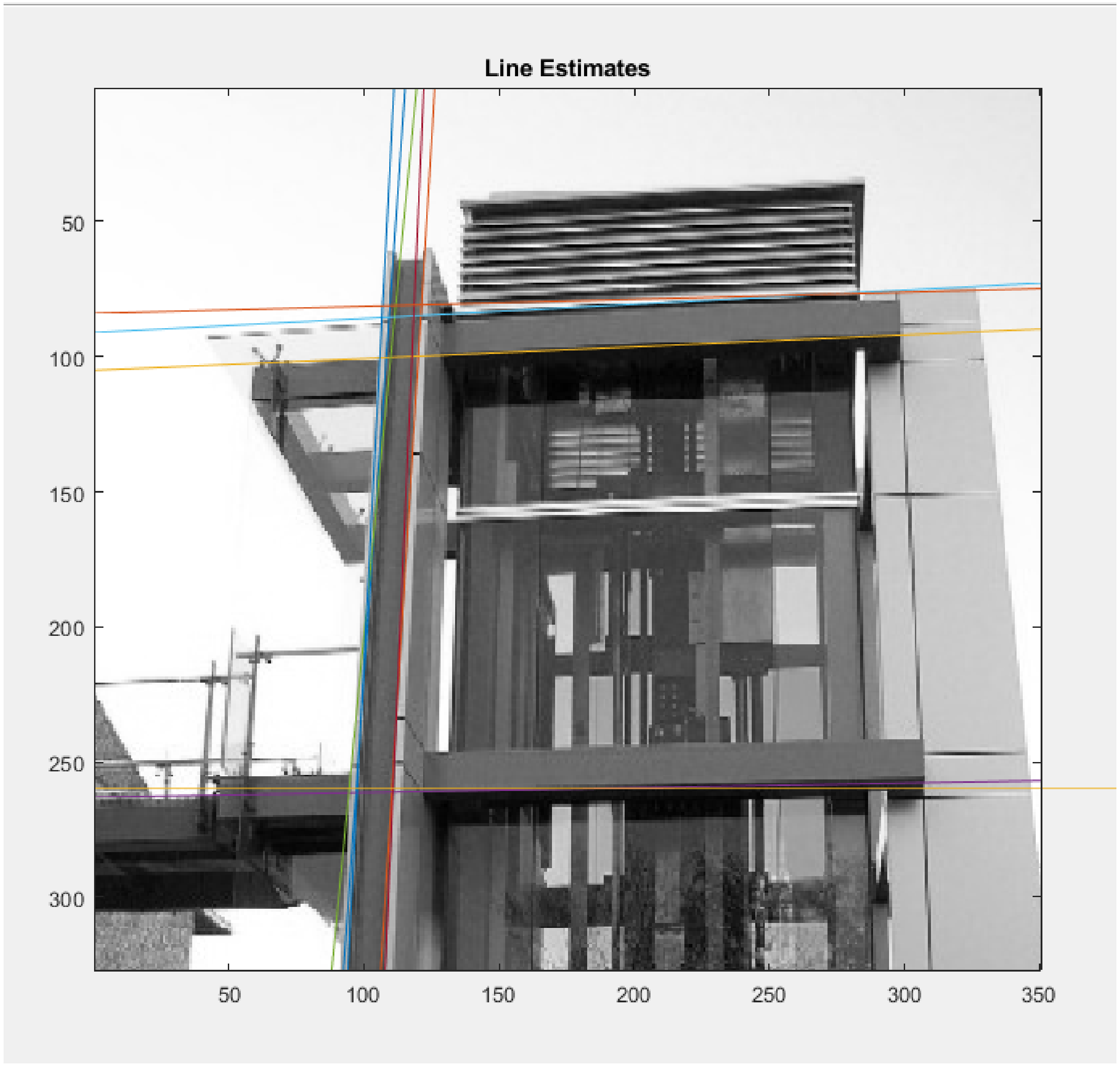
Lift Edge Image (10 lines)



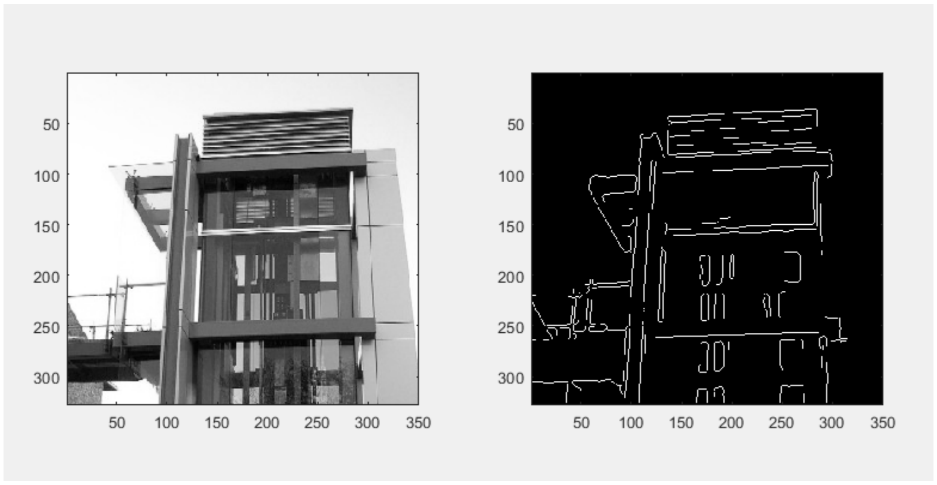


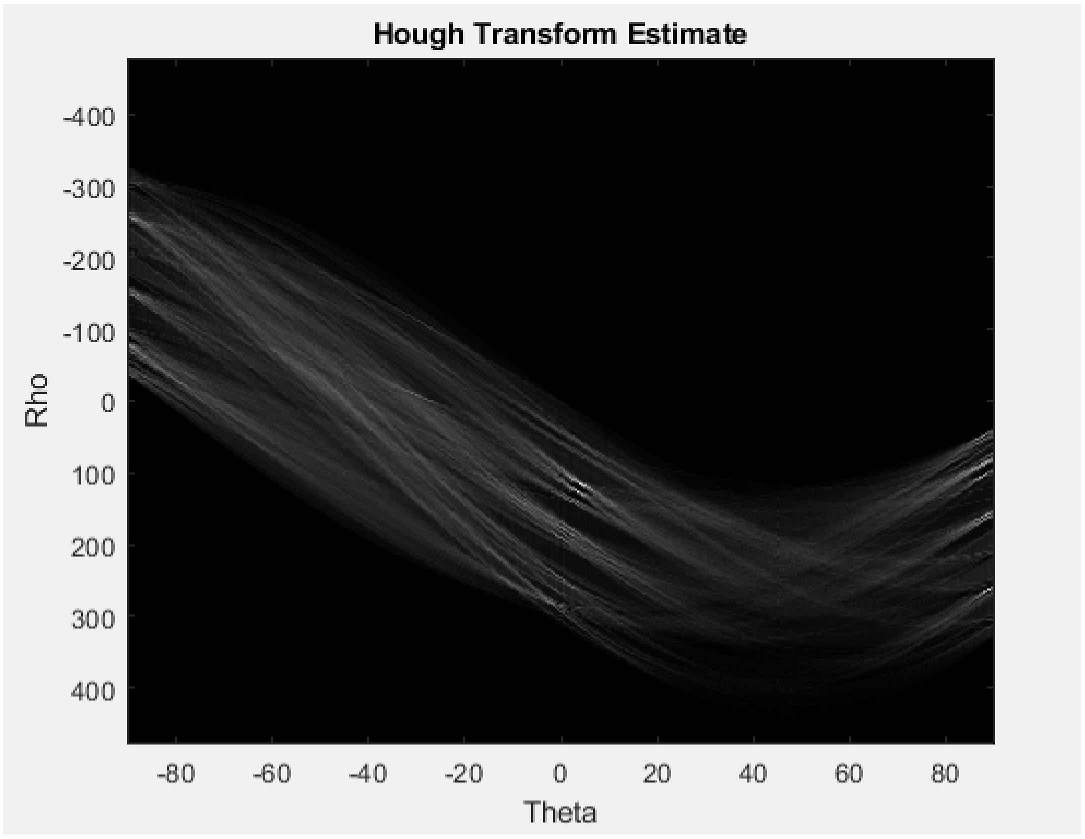


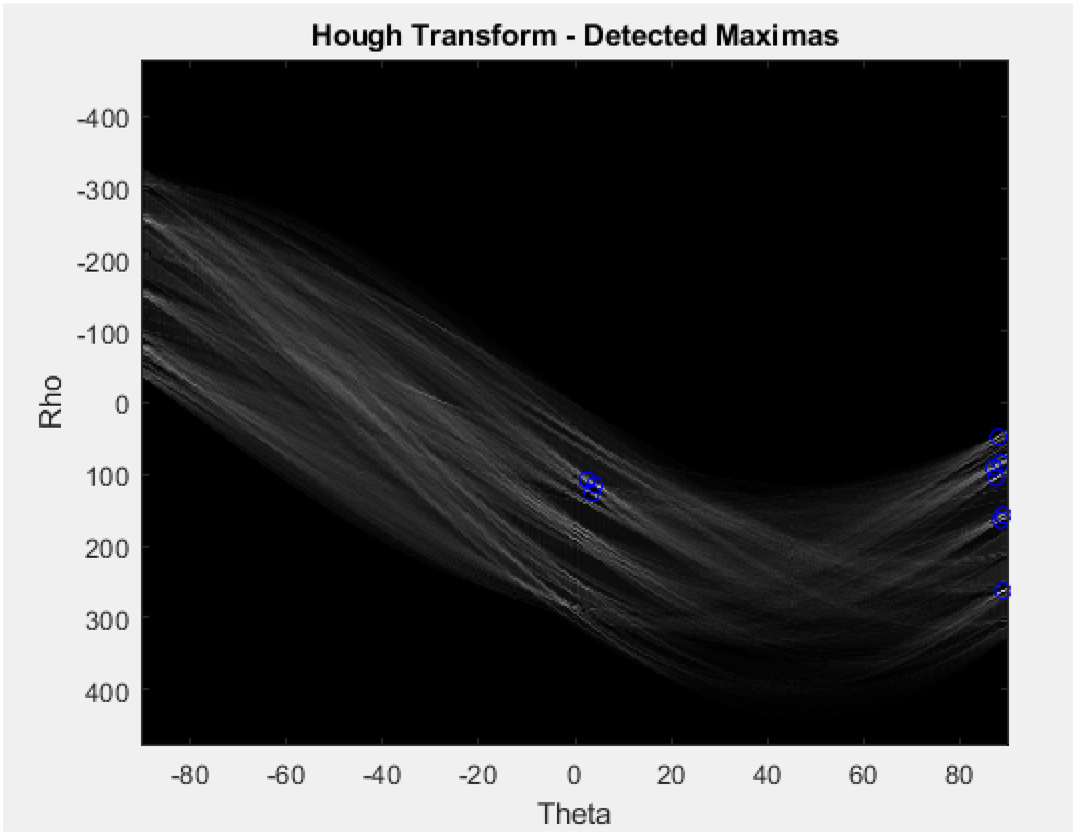
Hough Transform - Detected Maximas
Rho
Theta
-400
-300
-200
-100
0
100
200
300
400
-80
-60
-40
-20
0
20
40
60
80

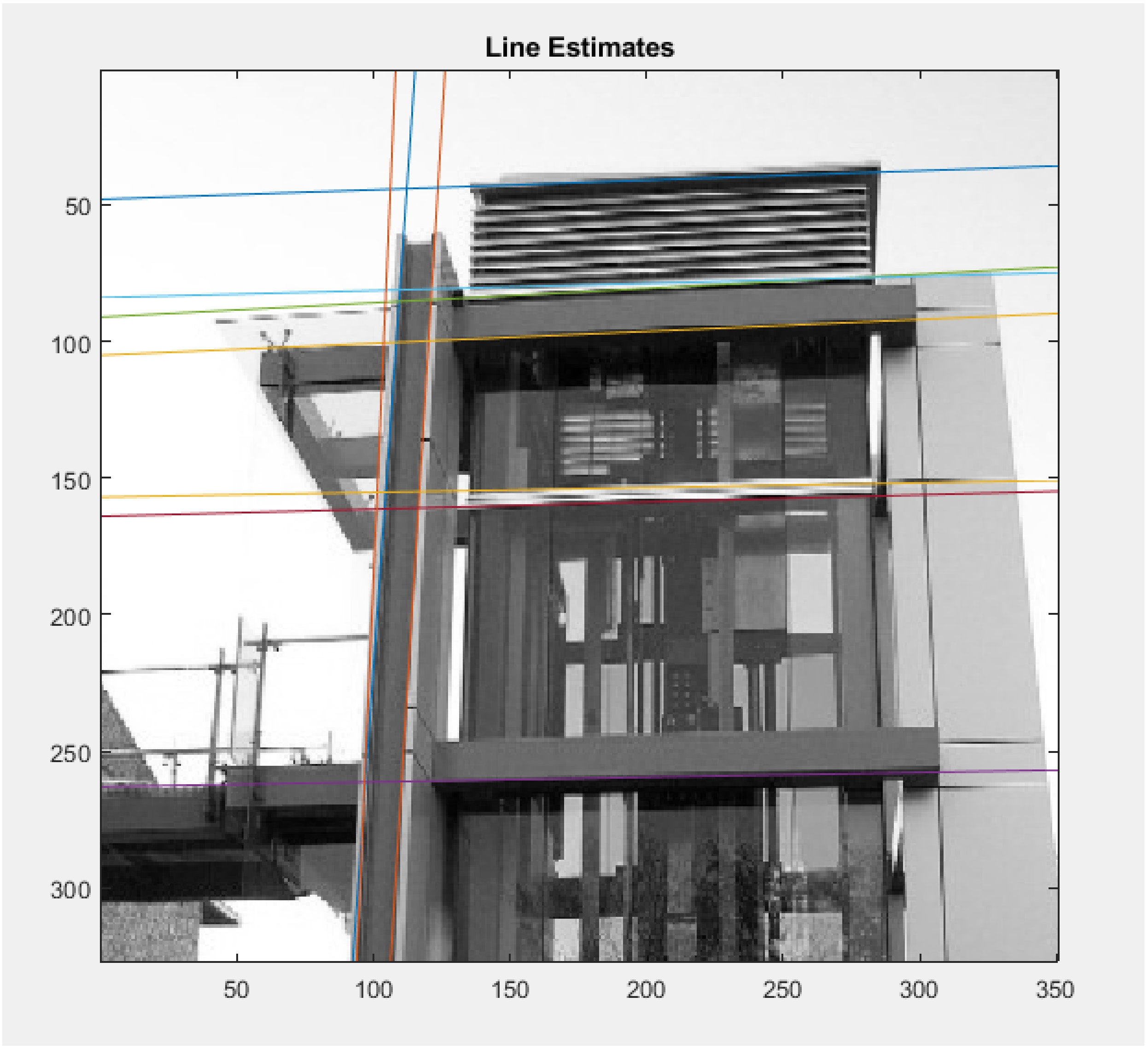

Line Estimates
50
100
150
200
250
300
50
100
150
200
250
300
350

Lift Edge Image (10 lines, Neighboring 11 pixels are zeroed)






Hough Transform - Detected Maximas
Rho
Theta
-400
-300
-200
-100
0
100
200
300
400
-80
-60
-40
-20
0
20
40
60
80


Line Estimates
50
100
150
200
250
300
50
100
150
200
250
300
350

## Comments

The `extract_lines` function was able to successfully extract the peaks of the hough transforms. However, an issue exists when generating 10 line estimates in the lift image. Rather than finding 10 distinct lines, there are line estimates in the image that refer to the same line. This is because neighboring maxima around the local maximum of the current iteration are not completely zeroed out. There are remnants of the local maximum remaining and these remnants become the global maximum on the next iteration. By increasing the region to 11x11 from 5x5, these remnants, and consequently, duplicate line estimates were reduced.

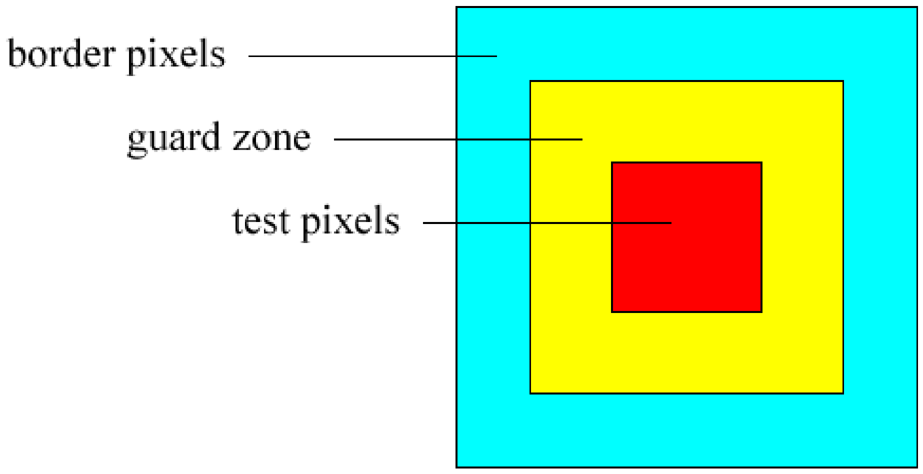
This means that there is a need to adjust the line finder depending on the image being analyzed. Following the suggestion of a 5x5 region may not be enough for complex images with multiple lines, like the lift image, but it may be enough for simpler images like the box image.

Running the `extractlines_test` function without inputs, Generating the hough data took the longest time with it taking 67.2% of the total running time. In addition, `extract_lines` took 19.1% of the total running time. A huge chunk of the running time was taken by the loop that finds the global maximum of the current iteration. This loop was consuming 65.5% of the total running time. The time complexity of the loop is O(N * size(H)) since the algorithm loops through the entire array to find the peak in the hough data N times to get N number of line estimates.

## 3D - Written Questions

1. Explain how the CFAR detector works (use a diagram etc if required) and explain under what conditions does it make sense to use a local thresholding scheme such as this as opposed to a global one (such as iso-thresh).

- CFAR detection is an adaptive thresholding technique that compares local noise statistics in a region around the point of interest to the test point. This technique is used for target detection by identifying small regions of light/dark from a varying background. The test area consists of border pixels (B), a guard zone (G), and central test pixels (P).



The points inside the guard zone are ignored in the simplest version of CFAR, however alternative forms also text pixels within the guard zone. Assuming Gaussian distribution, the test condition is defined as:

$$\frac{\left|P-\mu_B\right|}{\sigma_B} > k$$

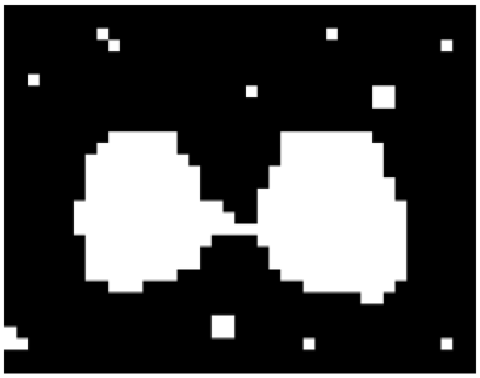
Where $k$ equals the number of standard deviations away from the mean $\mu$.

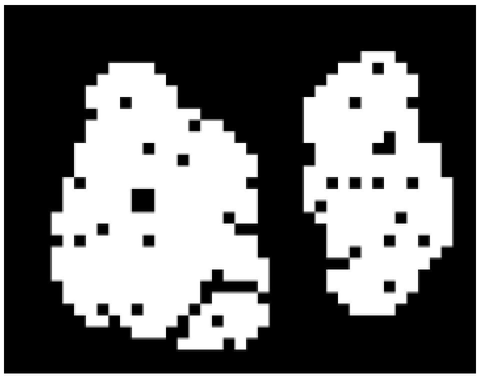
The size/shape of the guard zone would determine the size/shape of detectable objects and its sensitivity decreases as local noise/clutter increases. CFAR detection can readily handle gradual variations in the background, however, it is computationally intensive for larger guard zones.

Although global thresholding is simpler and has less computational cost, local thresholding schemes such as CFAR detection can handle more complex situations, such as images with varying contrast levels or images whose global intensity histogram doesn't contain distinctive peaks. This is due to the fact that they can choose different threshold values for every pixel in the image based on an analysis of its neighboring pixels.

2. A binary image contains a number of thresholded objects including several which have been accidentally fragmented during segmentation (See below). Some smaller regions are also present.

a. Using the morphological filters discussed in class, suggest how you would clean up the following two binary images containing 2 shapes each:

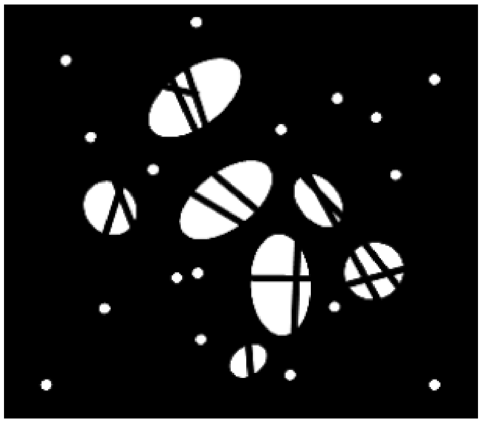




- The first image can be cleaned up through erosion since it would be enough to remove the isolated points and weak connections. However, if shape preservation is very important, then opening would be the best option, since the dilation after the erosion would help preserve the shape of the two regions.

  The second image, on the other hand, can be fixed through dilation since it would merge the fragmented regions. But its shape can still be better preserved through closing, since the succeeding erosion would help restore it to its original size.

b. Suggest how the small regions can be removed without adversely affecting the line-like structures of the larger regions using a combination of morphological filters and some simple logic.



- A combination of an erosion, followed by a dilation would be appropriate for this use case. This combination of morphological filters is called opening. This type of morphological filter removes isolated regions and weak connections while retaining the majority of the shapes of the larger region.

  This is because the first filter, erosion removes the smaller objects and some parts of the bigger image. Afterwards, the second filter, dilation, approximately restores the shape and size of the bigger objects by filling back the pixels removed from the remaining shapes.

c. Conversely, what would we do if we wanted to remove the fragmentation of the large regions?

- A combination of dilation, followed by erosion would be appropriate for this use case. This combination of morphological filters is called closing. This type of filter fills in holes in an image and connects fragmented regions while retaining the majority of the shapes of the larger region.

  This is because the first filter, dilation adds additional pixels around each region to connect the different regions. In this case, dilation closes up the holes in the larger image. Afterwards, the second filter, erosion, approximately restores the shape and size of the shapes by removing the additional pixels.

## References

(2009) Local Adaptive Thresholding. In: Li S.Z., Jain A. (eds) Encyclopedia of Biometrics. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-73003-5_506