Answers to questions in

Lab 2: Edge detection & Hough transform

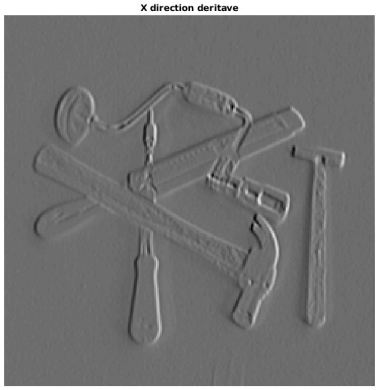
Name: Per Emil Hammarlund Program: TMAIM

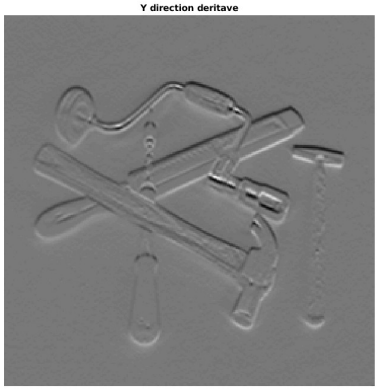
Instructions: Complete the lab according to the instructions in the notes and respond to the questions stated below. Keep the answers short and focus on what is essential. Illustrate with figures only when explicitly requested.

Good luck!

Question 1: What do you expect the results to look like and why? Compare the size of *dxtools* with the size of *tools*. Why are these sizes different?

Answers:
I expected to find high pixel magnitudes where the rate of change in the image in the given direction is large and low values where there is little change in the image. And, with little surprise that is exactly what was observed:



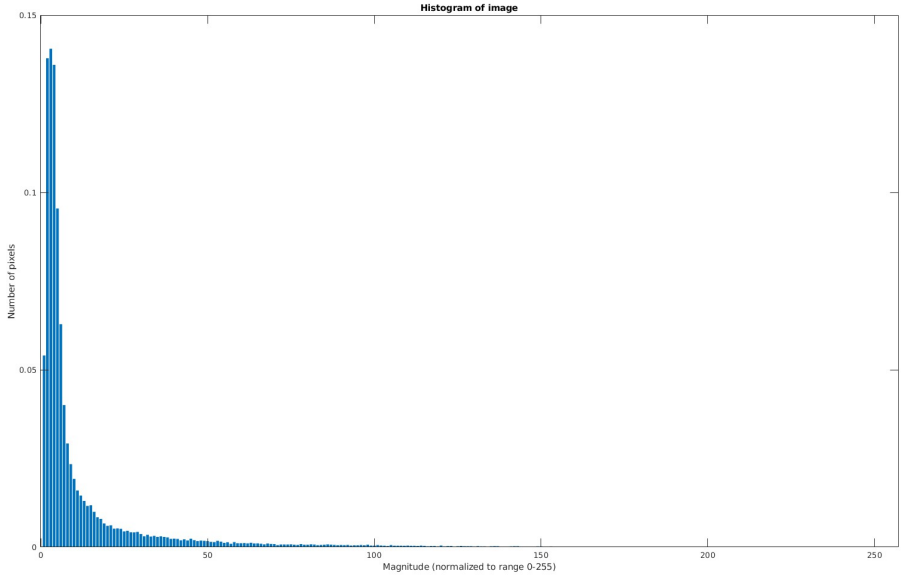


Which makes a lot of sense since derivatives are rates of change.

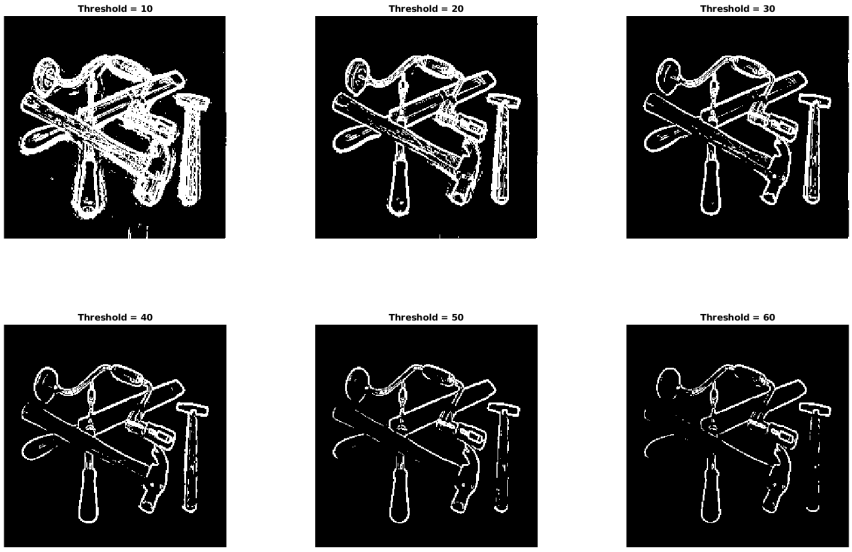
The size of dxtools is (254, 254), and the size of tools is (256, 256), this makes sense since we are convolving with a 3x3 mask and we are using 'valid'. This gives us the non-zeropadded version which cuts out 2 pixels in each of the dimensions.

Question 2: Is it easy to find a threshold that results in thin edges? Explain why or why not!

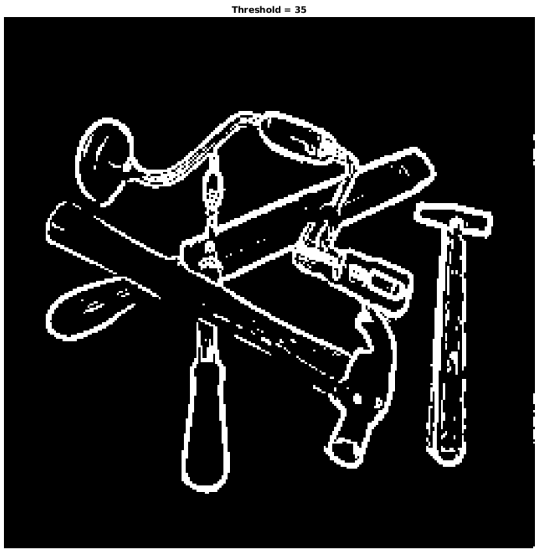
Answers:
Normalizing the gradient magnitude to the range 0-255 and computing the histogram of the image gave the following result:



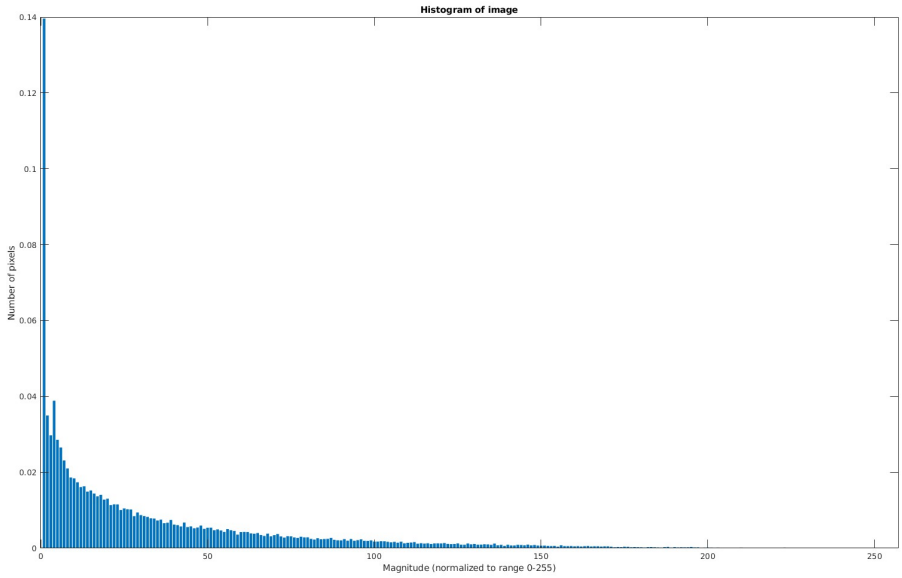
Judging from the figure above, thresholding between 20-50 seems to make the most sense. But it is not entirely obvious since we cannot deduce what intensities are unnecessary for being able to see the structure of the image. Testing thresholds in [10 20 30 40 50 60], we get:



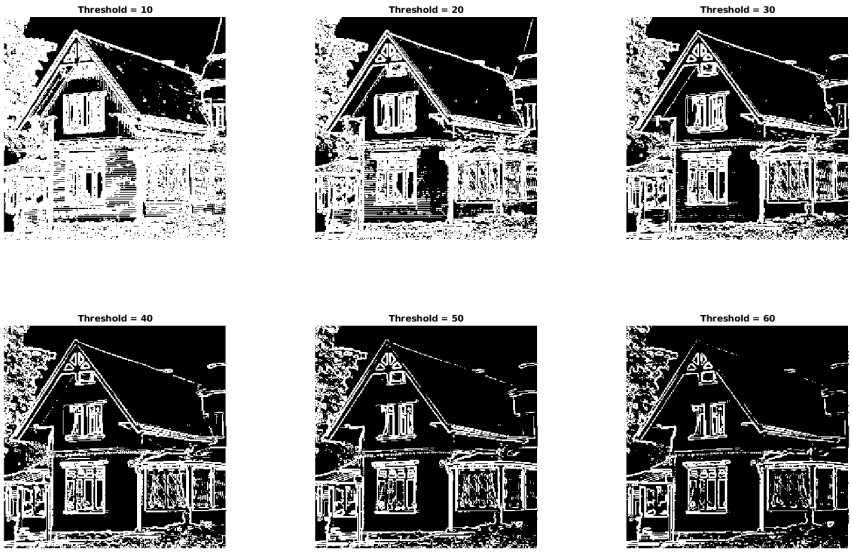
From which it can be seen that the optimal value lies between 30-40, thresholding for the value 35 we get:



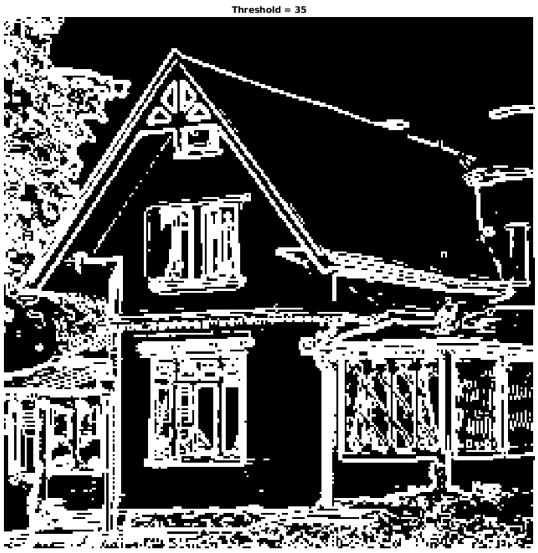
The histogram of the second image took the following form:



Here we can see that the image has a wider distribution of intensities, and deciding on an appropriate threshold has become abit harder, testing for the values [10 20 30 40 50 60] we get:

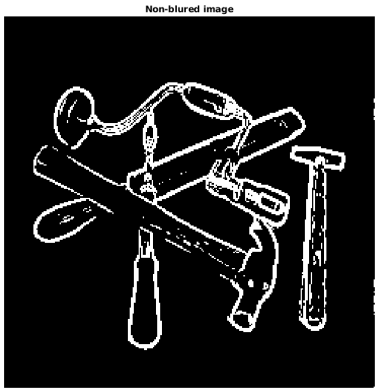


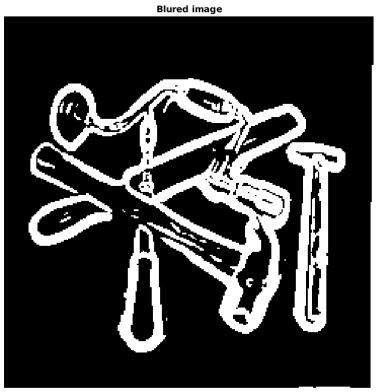
From which we can see that once again, the optimal value lies between 30-40, finally we get:

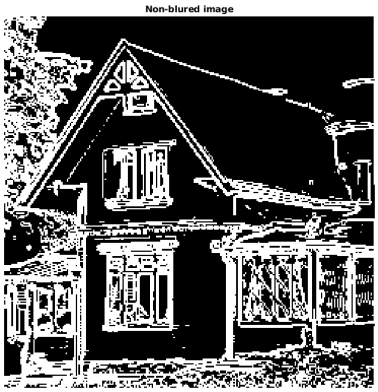


Question 3: Does smoothing the image help to find edges?

Answers:
Performing the thresholding with the "optimal" threshold for both of the images with and without smoothing gave the following result:



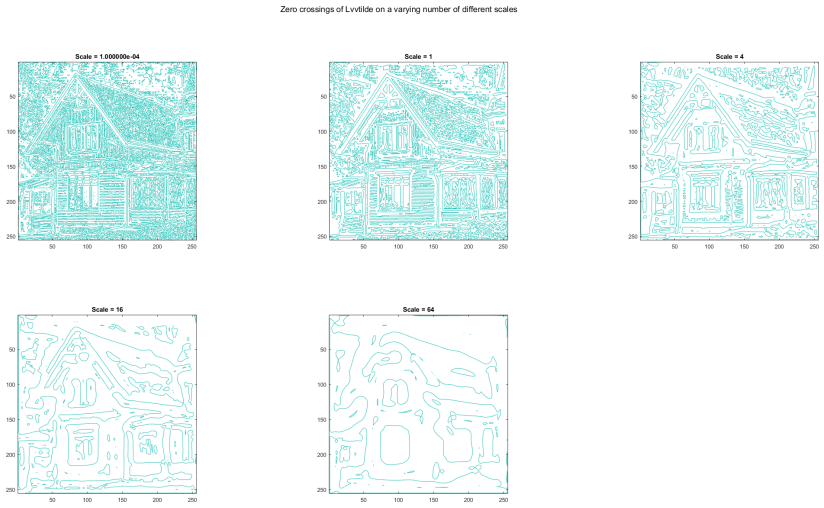




Here we can see that that not much is gained in terms of how good the search for edges is, but noise does get removed. The edges also get a bit distorted.

Question 4: What can you observe? Provide explanation based on the generated images.

Answers:
The generated images took the following form:

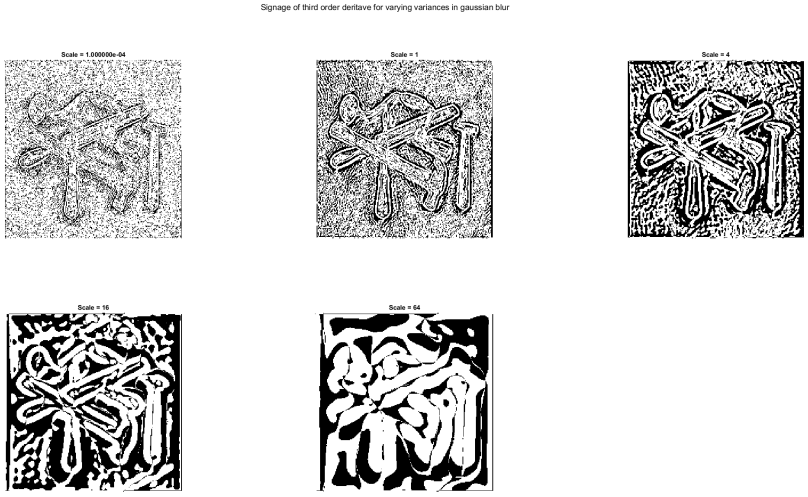


As be seen in the figure above. Lower scales are more noisy and contain finer structural information from the image and as the scale increases the finer details disappear and larger details emerge.

This is because the scale is simply the variance of the Gaussian blur that we use to smooth the image. As the scale increases, the small extreme points get smudged out and only the larger extremums survive giving us more general blobby structural information amd decreasing the noise in the image. And as the scale approaches zero, the smaller extremums survive giving us finer details in the image but we also get more noise.

Question 5: Assemble the results of the experiment above into an illustrative collage with the *subplot* command. Which are your observations and conclusions?

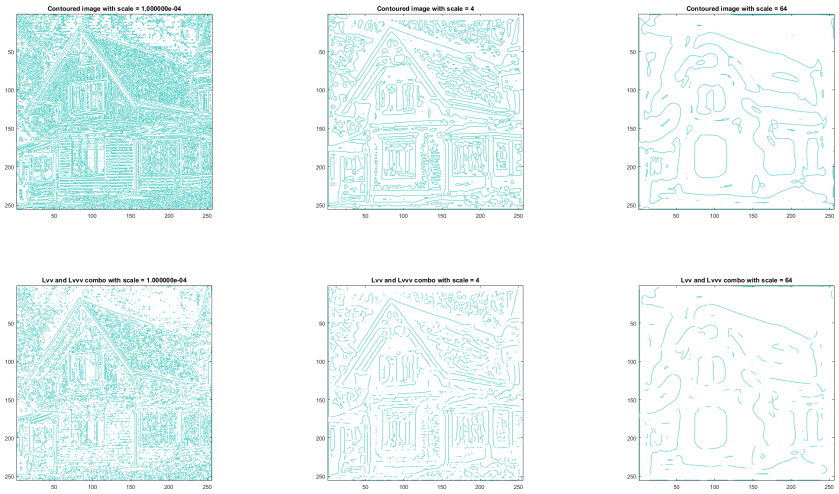
Answers:
The results took the following form:



From the figure above, we can see that the third order derivative is sensitive to noise, as we increase the scale the we can see that the positions with negative signage seems to give us some sort of notion of edges in the image, but it is not sufficient for truly knowing where edges are due to the third order derivative being as sensitive to noise as it is. As the scale grows yet larger, the “edges” become so large that we lose the notion of there being any edges there.

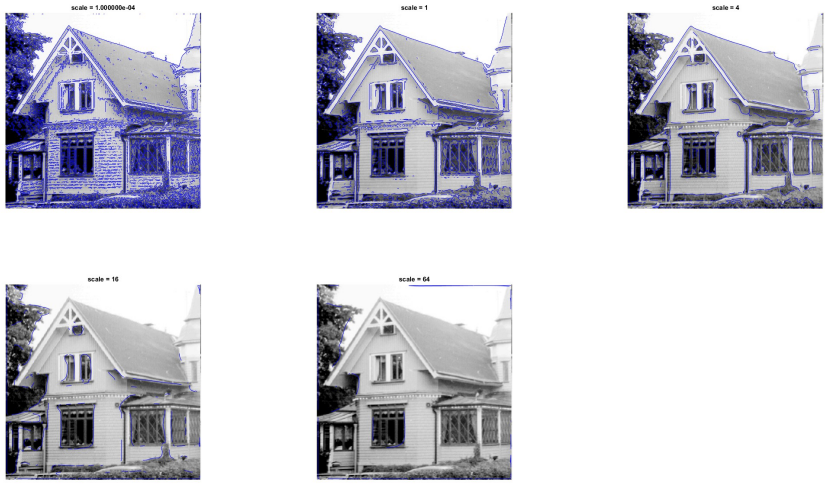
Question 6: How can you use the response from *Lvv* to detect edges, and how can you improve the result by using *Lvvv*?

Answers:
Edges can be detected by finding where there are zero crossings in Lvv and the results can be improved by checking for if Lvvv satisfies Lvvv < 0 at every point. There is a slight improvement, but in order to get rid of unwanted edges thresholding shold also be applied to the image.

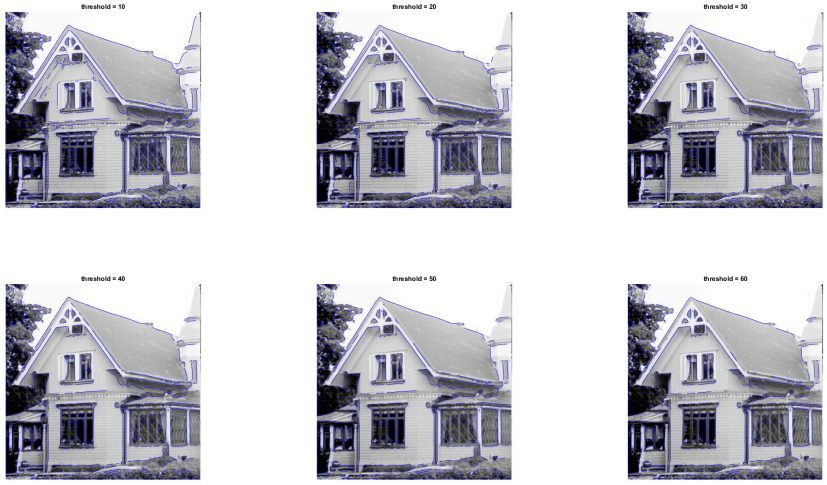


Question 7: Present your best results obtained with *extractedge* for *house* and *tools*.

Answers:
Varying the scale while maintaining the threshold the was found to be optimal in task 2 (threshold = 35) gave the following results:



From which it can be seen that the threshold 4 is optimal, varying the threshold with the values [10 20 30 40 50 60] gave the following results:



Judging the figure above, although subjective, threshold = 40 seems to be optimal. Setting the scale = 4 and threshold = 40 gave the following result:

Performing the same procedure for tools gave:

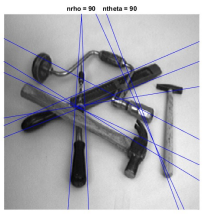
Question 8: Identify the correspondences between the strongest peaks in the accumulator and line segments in the output image. Doing so convince yourself that the implementation is correct. Summarize the results of in one or more figures.

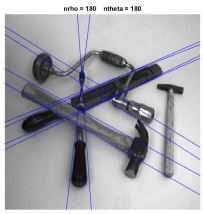
Answers:

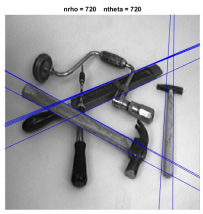
In the figure above we can see the triangle image and its representation in houghspace. Here we can clearly see that there are three peaks, the first and most obvious being in the left and right edges of the houghspace (theta close to pi/2 and pi/2) which corresponds to the leftmost edge of the triangle. The second one that is easy to spot can be found in in the middle of the theta axle of the image and below the brightest peak. Since it is in the middle of the theta axle, it corresponds to the bottom edge of the triangle. Finally, the last edge is the brightest peak in the houghspace. This peak corresponds to the hypotenuse of the triangle since theta is between -pi/2 and 0.

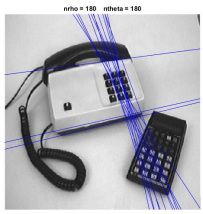
Question 9: How do the results and computational time depend on the number of cells in the accumulator?

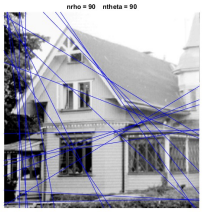
Answers:

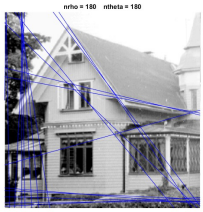


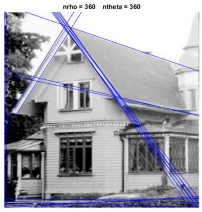


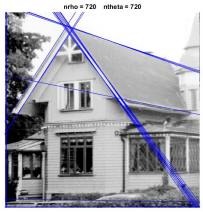












Unsurprisingly, the accuracy of the lines increases as the number of cells increases. If the resolution is too low, then our estimates of edges are wildly inaccurate and the higher the resolution, the more accurate the lines.

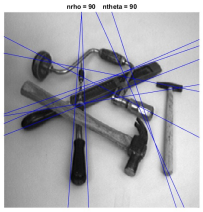
Since we loop over all theta values for each point that exceeds the threshold, and we only access rhos with a lookup in the coordinate system for rho. The computational complexity of the algorithm grows in polynomial time with nmakethresh * ntheta. (where nmakethresh is the number of points that make the threshold)

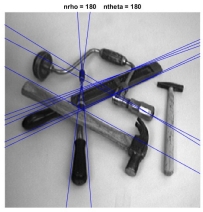
Question 10: How do you propose to do this? Try out a function that you would suggest and see if it improves the results. Does it?

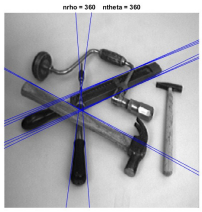
Answers:
I propose doing this by incrementing with the logarithm of the magnitude of the gradient for the current point x, y:

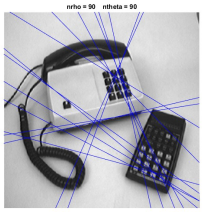
```
acc(rhoind, thetaind) = acc(rhoind, thetaind) + log(gradmag(x, y));
```

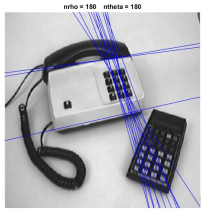
Doing this gave the following results:

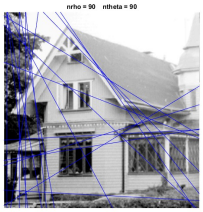


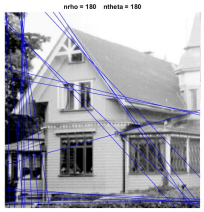


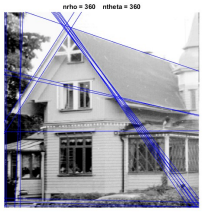


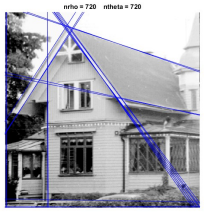












Looking at the images, we see that the results are similar, if not worse than just simply incrementing with 1.
