Answers to questions in

Lab 2: Edge detection & Hough transform

Name: Jacob Malmberg, Marcus Nystad Öhman Program: CINEK/MAIG/TIEMM1

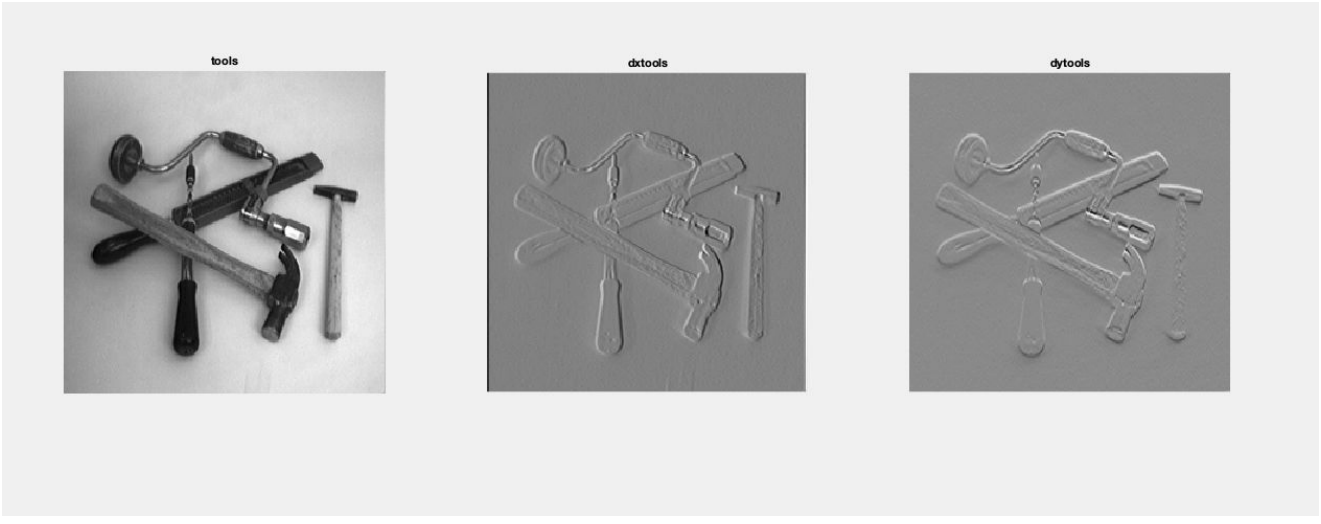
Instructions: Complete the lab according to the instructions in the notes and respond to the questions stated below. Keep the answers short and focus on what is essential. Illustrate with figures only when explicitly requested.

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:
We used the central difference operator.

If we use it x-wise - we get the difference x-wise between pixels. That means that we see changes in the x direction better.
If we use it y-wise - we get the difference y-wise between pixels. We see changes in the y direction better.

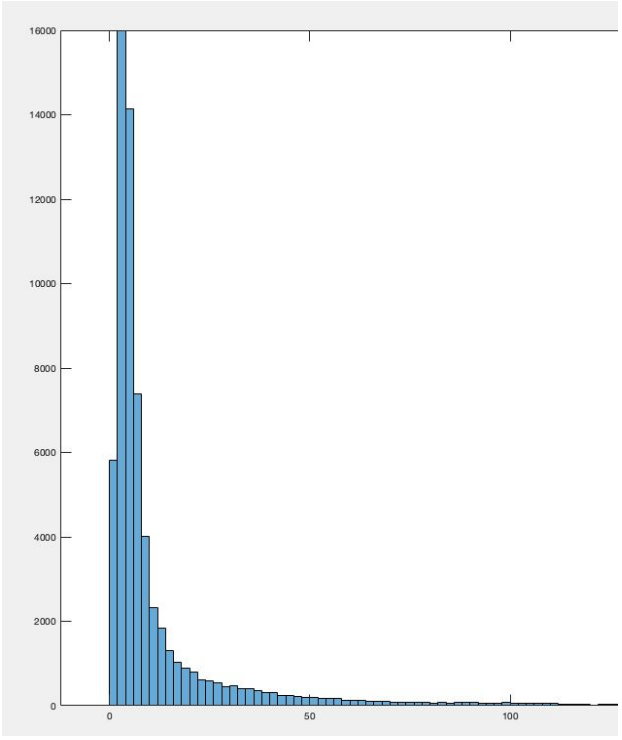


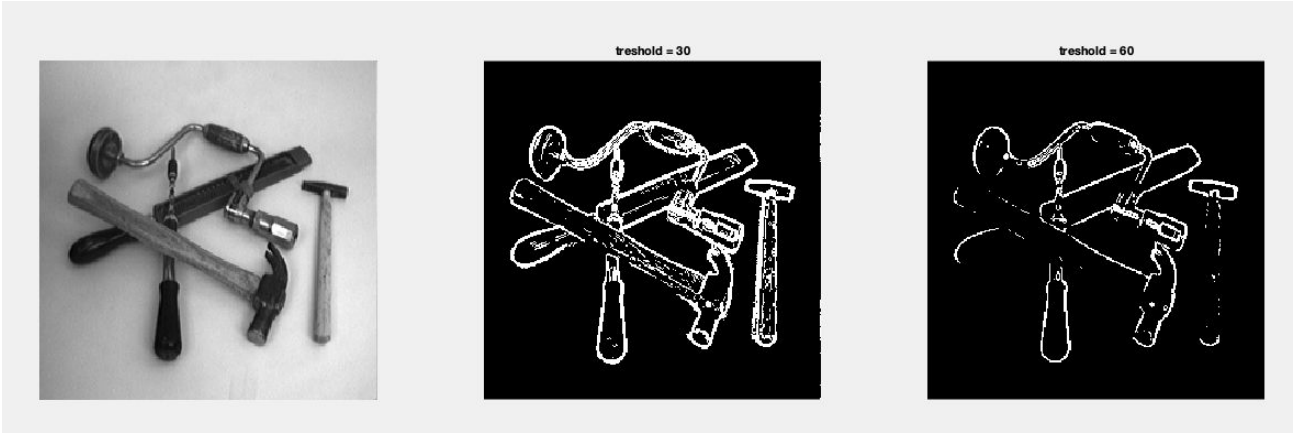
The sizes are difference because of the 'valid' parameter. What happens is that we convolve but only return the values that are computed without the zero-padded edges.

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

Answers:

The histogram:
On the x-axis we have the magnitude, higher to the right. On the y-axis, the count for different bins of magnitudes.





using central difference operator

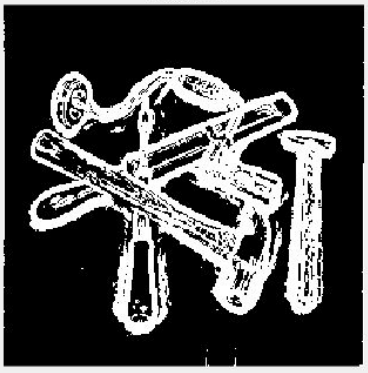
Is it easy to find a threshold that results in thin edges?

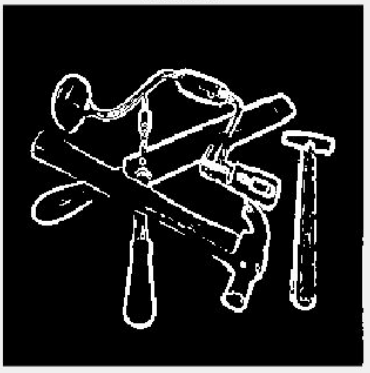
Sort of. If the threshold is too low, we get thick lines. If it is too high, it doesnt detect all the edges, just the edges with highest contrast. Such as the hammerhead to the right.
The reason the diagonals disappear with higher threshold is because there arent any clear edges, they bleed into the background. The problem persists even with Roberts diagonal difference operator.

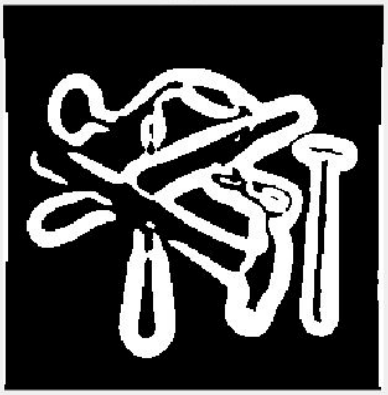
Question 3: Does smoothing the image help to find edges?

Answers:

Smoothing smoothes the edges, but also removes the noise. There may be some noise very close to the edges that we can remove, which if we smooth just the right amount amounts in smaller edges. Smoothing mostly affect noise. However, if we smooth too much we lose the edges altogether.

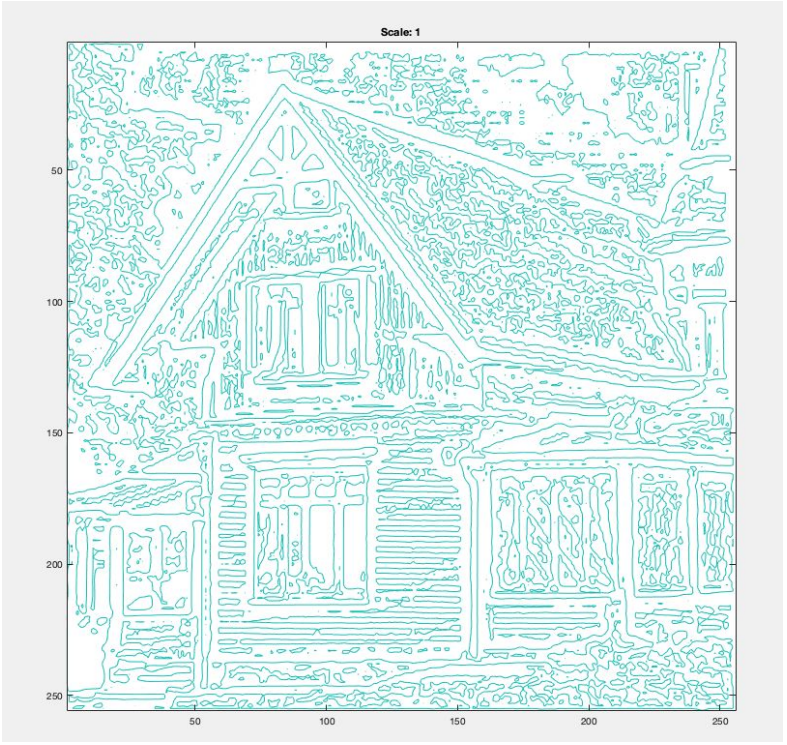


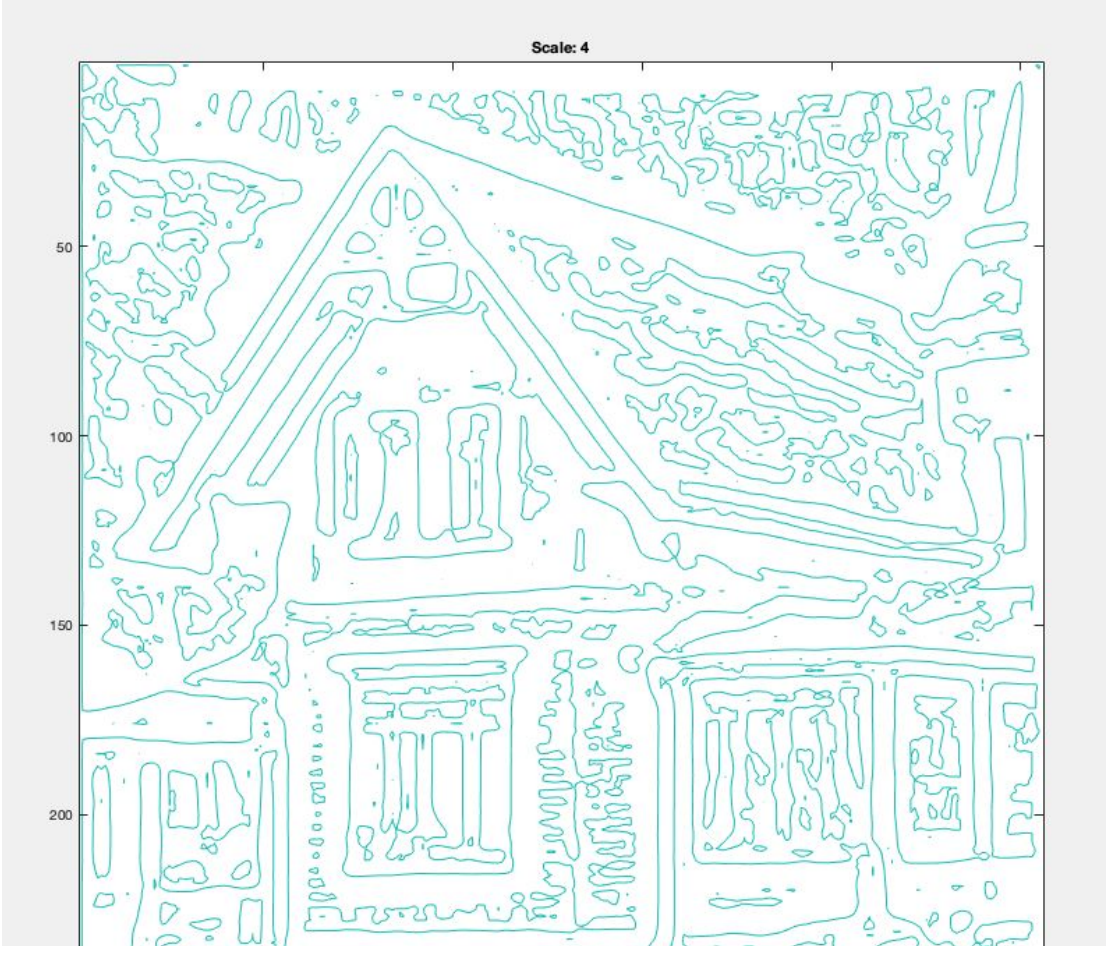




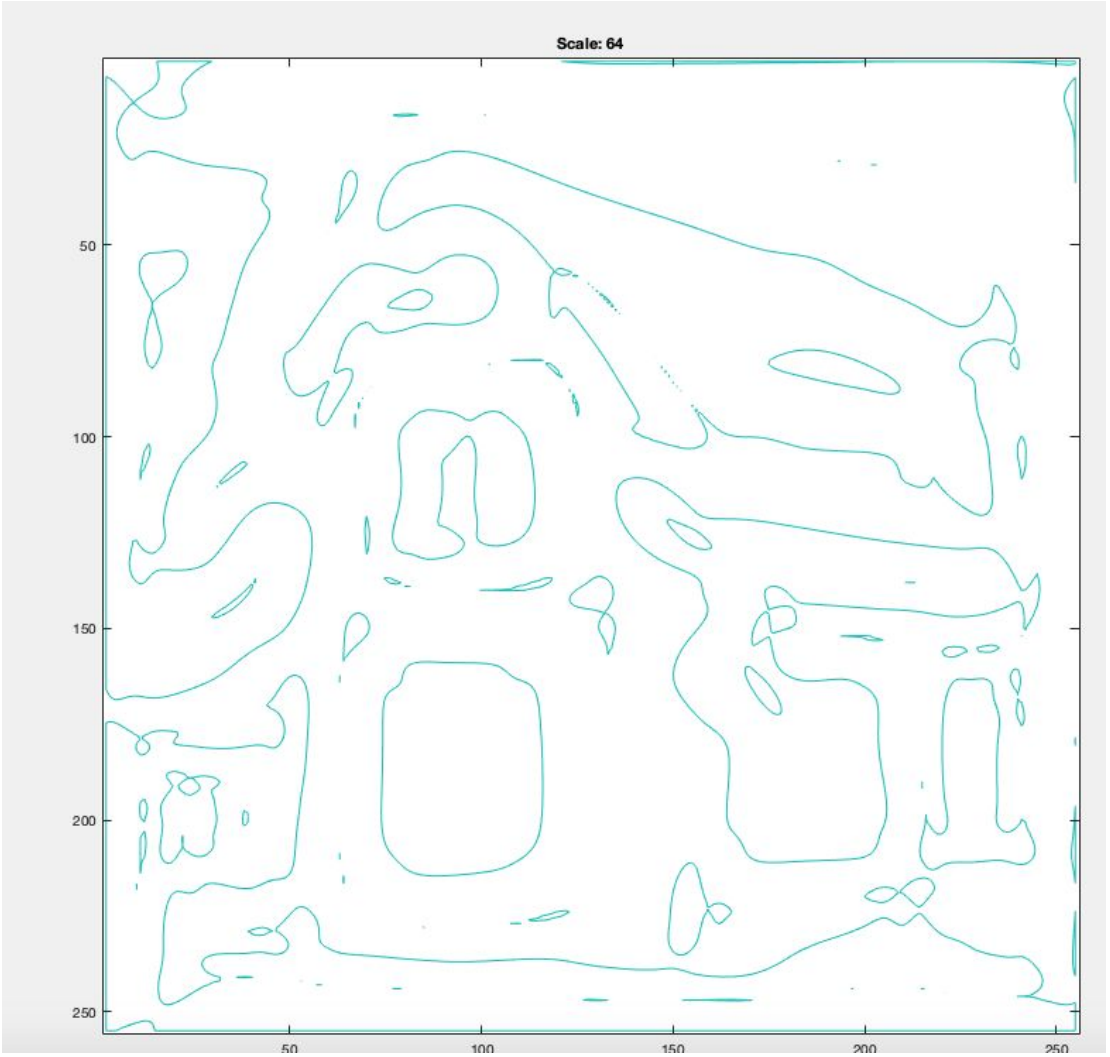
Experiments:


Scale: 0.0001
50
100
150
200
250
50
100
150
200
250


Scale: 1
50
100
150
200
250
50
100
150
200
250


Scale: 4
50
100
150
200


Scale: 16
50
100
150
200
250
50
100
150
200
250



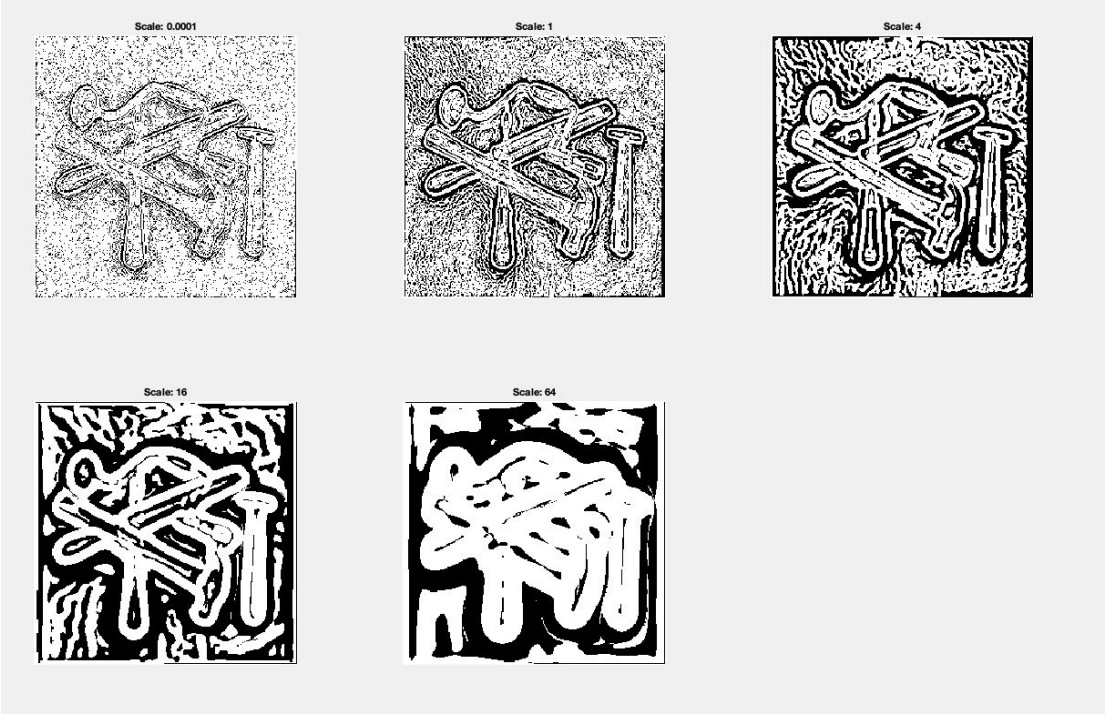
Question 4: What can you observe? Provide explanation based on the generated images.

Answers:
Lower scale means less smoothing -> more noise.

As we smooth more, more noise gets removed but we also loss less significant edges. At scale = 4 we lose all the “planks” on the house, but the noise above the house is still in the picture mostly.

Lots of zerocrossings. However, since we do not do the third derivative test, we also get minpoints!

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



White areas are where third derivative is < 0 and contains the max point (that is, edges). Black is where the third derivative >= 0, and contains the minpoints. You roughly have the same amount of maximum and minimum points due to the fact that you can't have two following maximas!

Answers:

The sign says that the third derivative should be less than zero. This means that the second derivative is decreasing.

By smoothing more, we get thicker edges!

When we smooth, the changes in the picture get less rapid / slower. Therefore, the first drivative is lower and wider. The second derivative and the third derivative is effected the same way.

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

Answers:

Lvv gives us the zero crossings. However, we get the zero crossing from both max and min crossings! By using the third derivative, we get only the local maximas, not the local minimas.

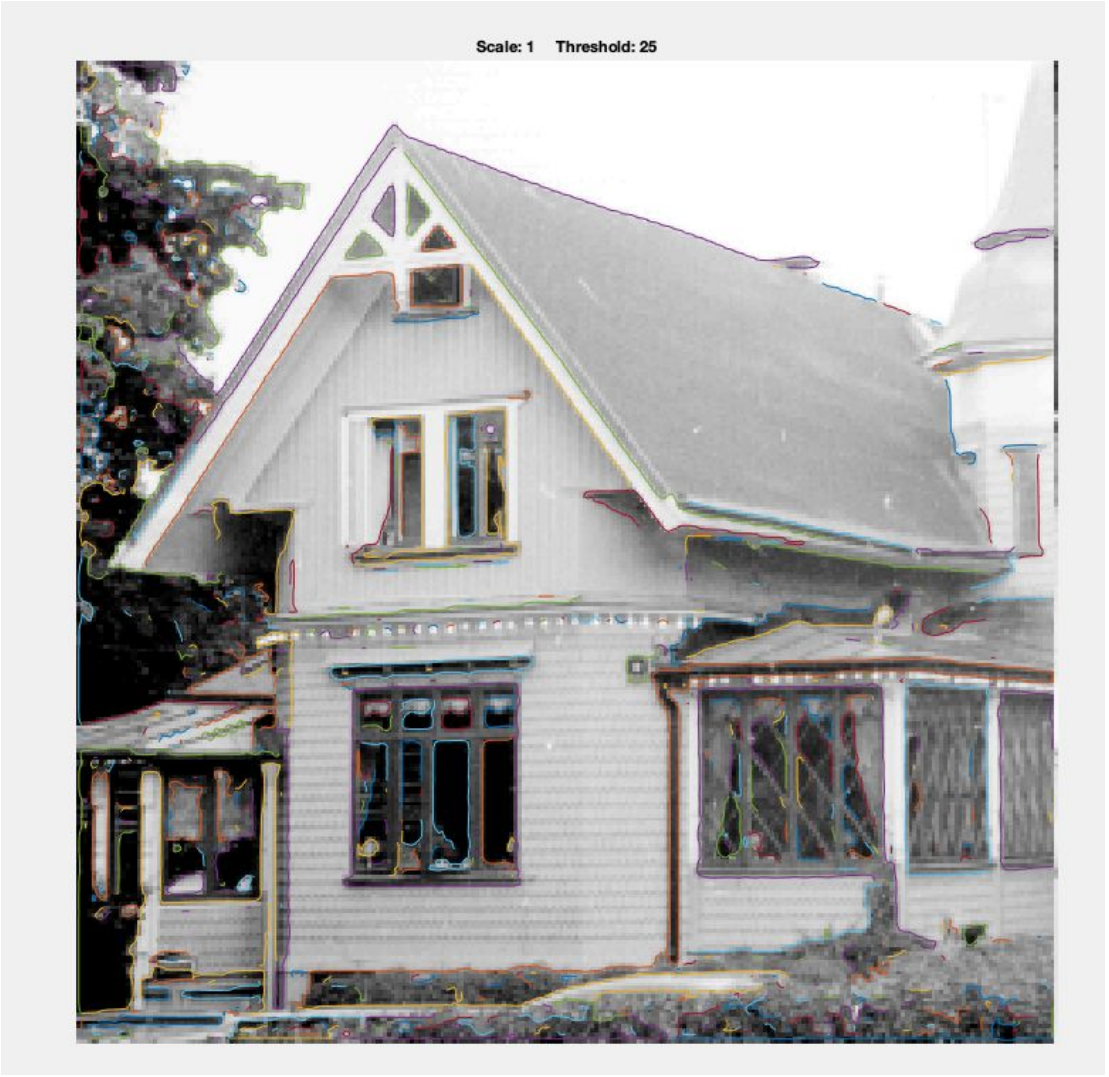
Question 7: Present your best results obtained with *extractedge* for *house* and *tools*.

Answers:

If we smoothe just the right amount, we should get less noise and better/thinner edges. Depending on how we set the threshold, we get thinner edges, or disregard edgepts that do not have large enough Lv.

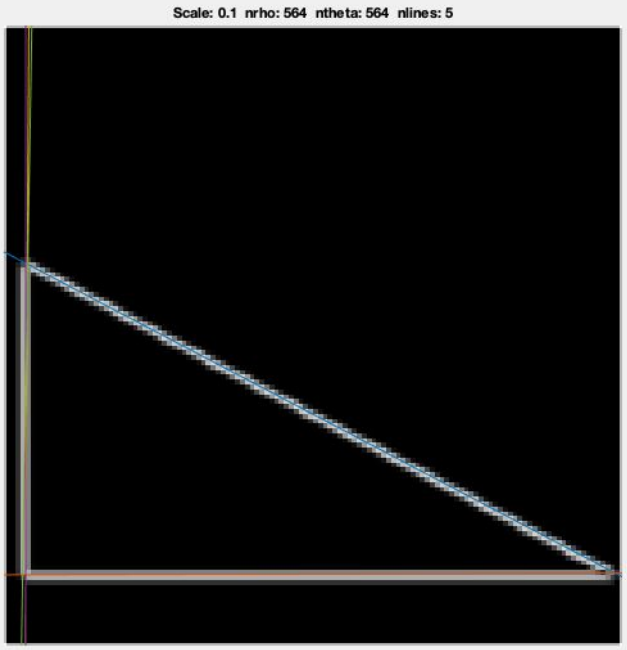
If threshold is too low, we recognize noise as edges. Say threshold = 10. If we scale too much, the threshold doesnt matter since we dont have any edges left…

Thresholding works well when the noise has lower gradient magnitude than the edges.



Question 8: Identify the correspondences between the strongest peaks in the accu-mulator 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.

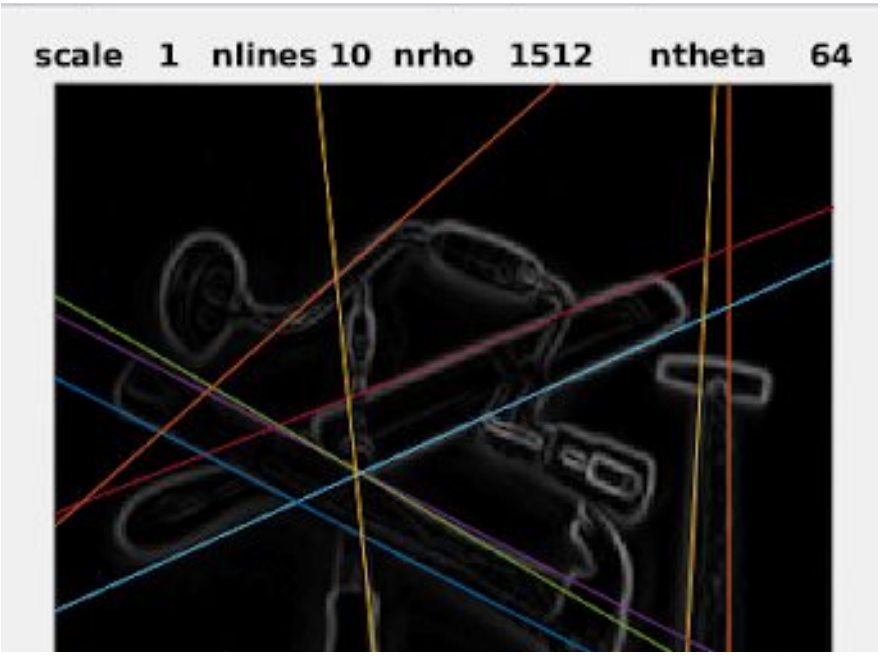
Answers:

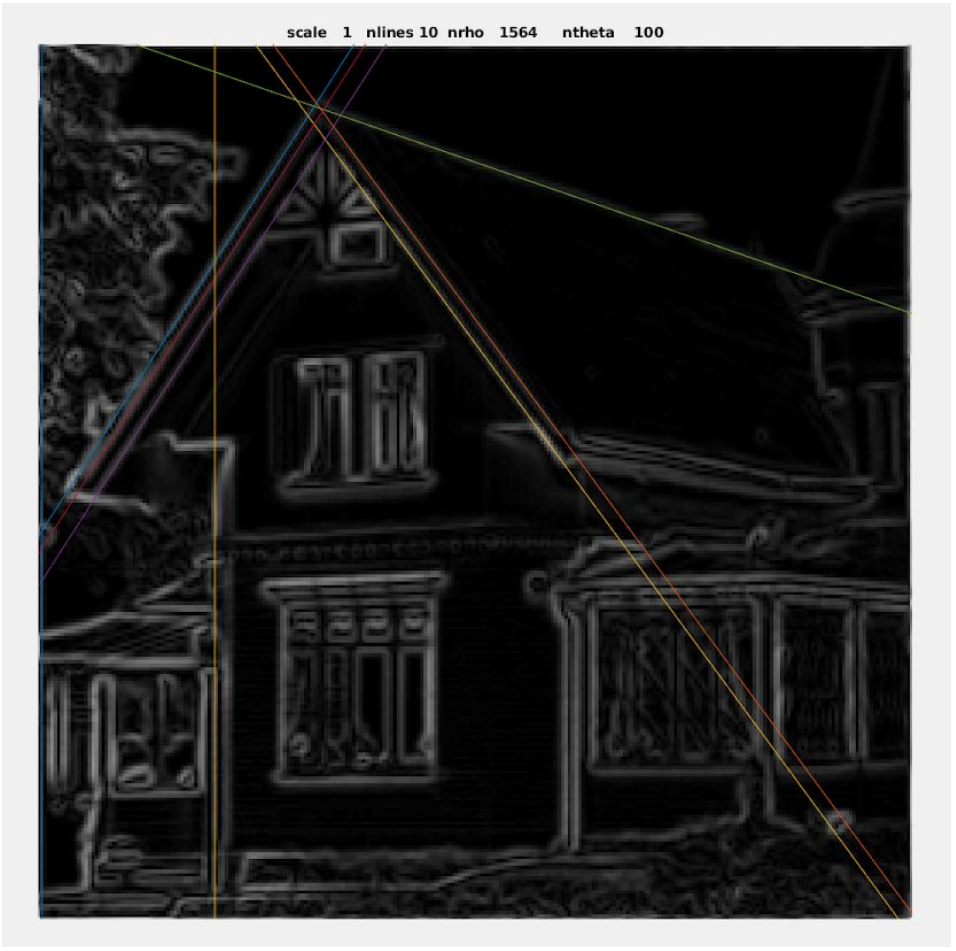


vertical axis is rho, horizontal is theta.

The angle is measured from the vertical axis in anticlockwise direction to the horizontal-axis. Therefore the left straight line corresponds to an angle of 90 degrees and a very small rho. Which leads to that it corresponds to the rightmost point (having about 0 rho and 90 degrees. The horizontal line corresponds to an angle of 0 degrees and a big rho, that is corresponding to the point in the middle (that has 0 angle and the biggest rho of the three points).

The angled line corresponds to a negative angle and has a rho in between the others => it's the point left to the one in the middle.




scale 1 nlines 10 nrho 1564 ntheta 100

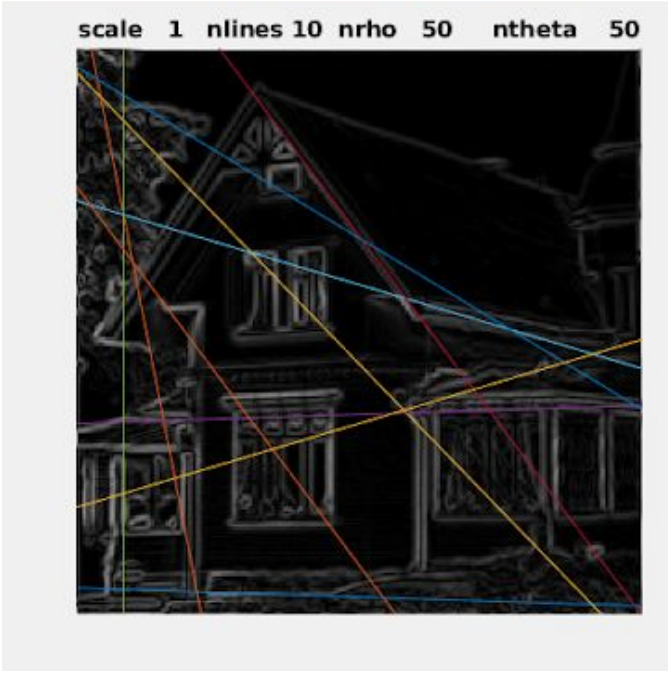
Question 9: How do the results and computational time depend on the number of cells in the accumulator?

Answers:

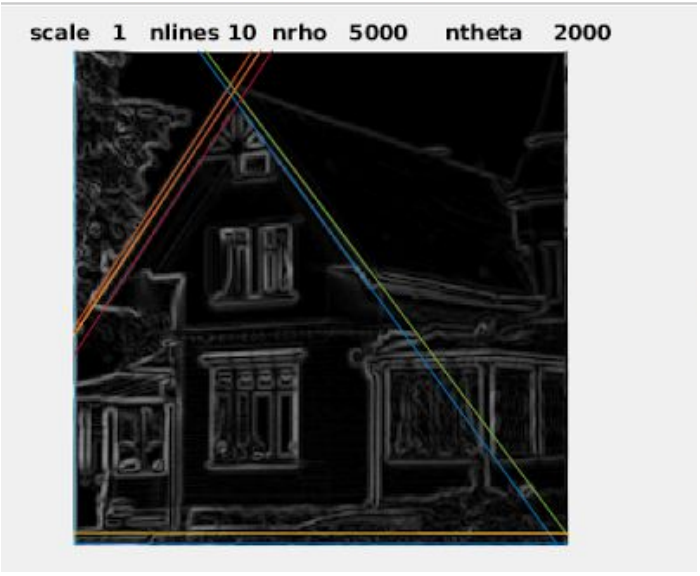
pic	**nrho**	**ntheta**	**time (s)**
godthem256	1500	500	20.95
godthem256	1000	500	14.92
godthem256	1500	1000	40.45

In **houghline,** we do "for index_theta = 1 : ntheta". So, if we increase ntheta the computational time should increase. Furthermore, in that loop we do "[d, rho_index] = min(abs(rho-rho1))" so if rho is larger there are more matrix operations, which should increase computational time.

The best results are achieved with a ntheta = 100. If we set ntheta or nrho to a very small value, say 50, we get very low accuracy.

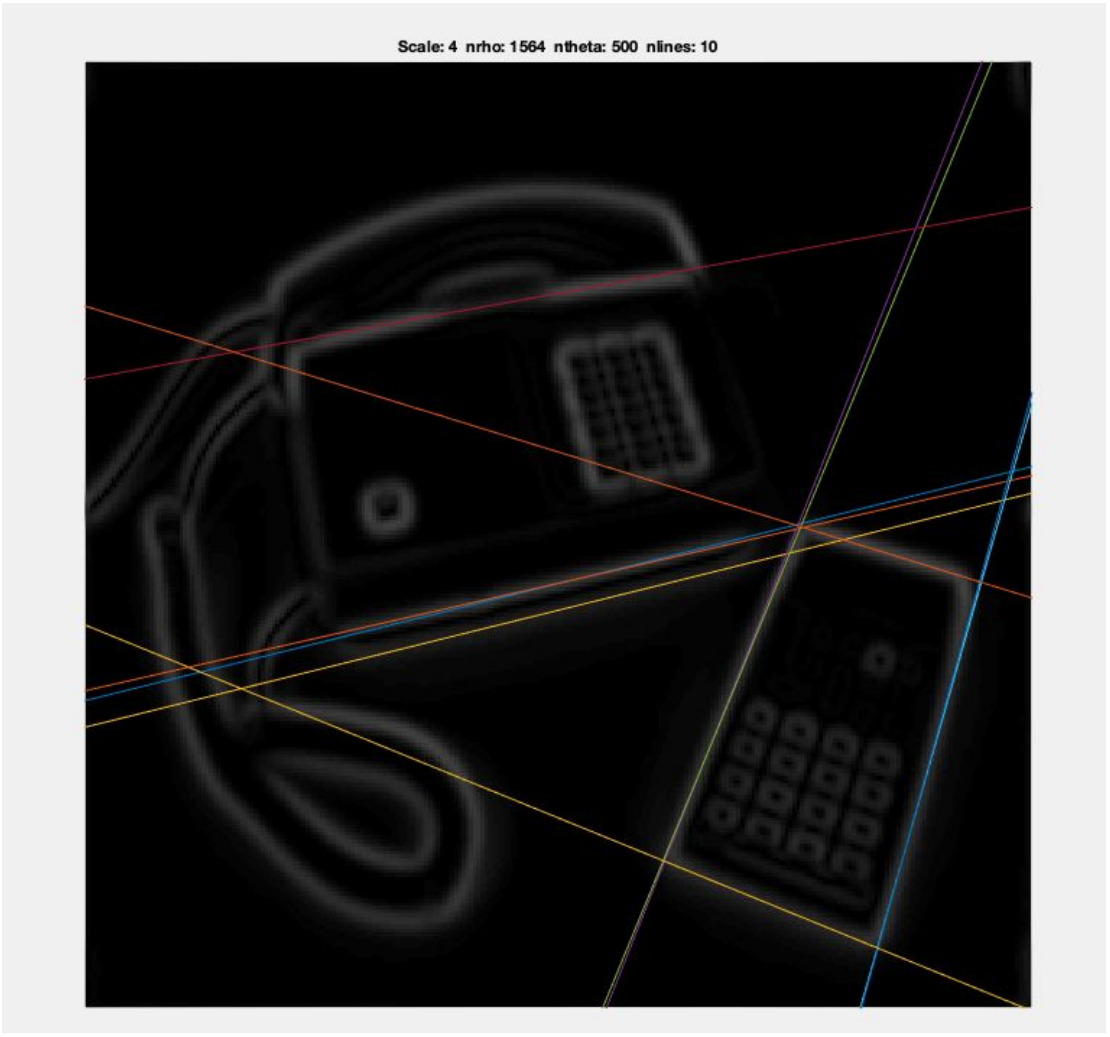


If we set ntheta or nrho to a very high value, we get higher accuracy but there are multiple responses for the same line



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:


Scale: 4 nrho: 1564 ntheta: 500 nlines: 10

By incrementing the accumulator with the gradient magnitude, we reduce critical dependency on thresholds. If we compare the results with the results in Q8, we can see that the results are slightly better now. We did this by increasing the vote with log(magnitude). This is because the log decreases the dynamic range. Intuitively this makes sense, because a (real) edge point should have a high magnitude and thus its' vote should have higher weight.
