Answers to questions in Lab 2: Edge detection & Hough transform

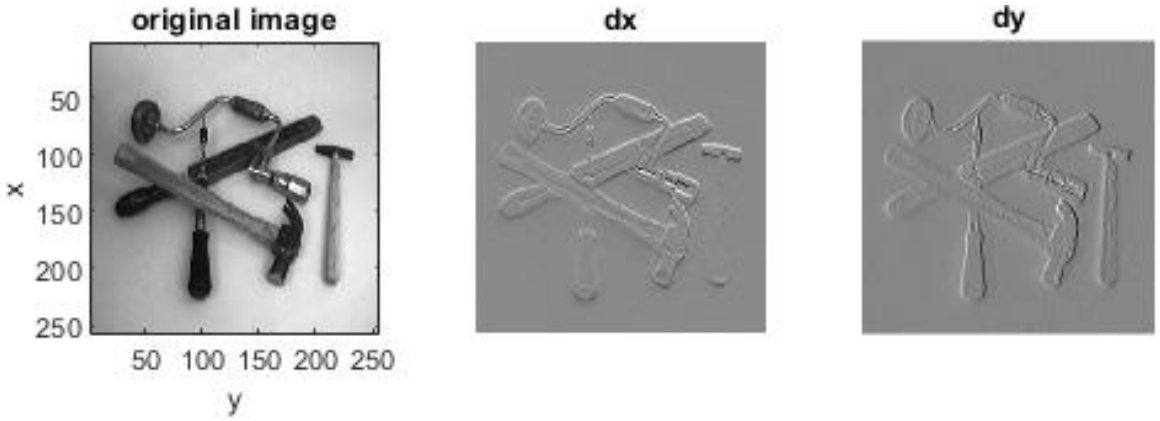
Name: Jonathan Pernow Program: TIPUM

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?

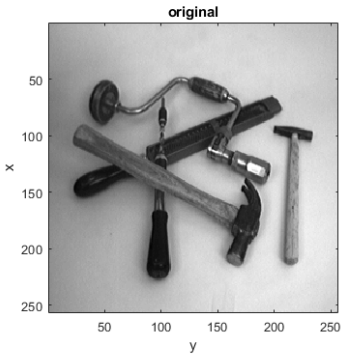
Answers:

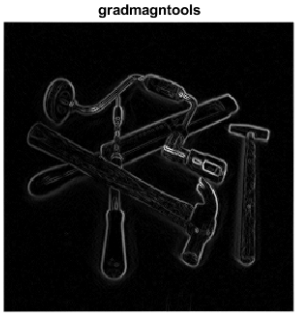


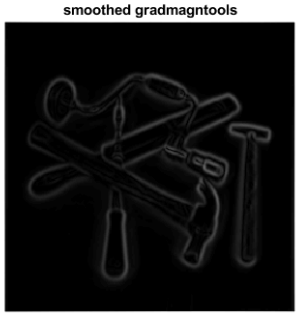
The image convolved with the x-mask will result in an image with clear vertical lines, while convolving with the y-mask will result in an image with clear horizontal lines. This is because the x-mask computes the derivative in x-direction while the y-mask computes the derivative in y-direction.
The central difference-masks were used in images shown.
Dxtools is of size 254 x 254 and tools is of size 256 x 256. We lose all the border lines of the image since the kernel isn't applied to the outermost pixels of the image (due to the third parameter of conv2 being 'valid'). From "help conv2" in MATLAB we get a clear description of the 'valid' parameter:
'valid' - returns only those parts of the convolution that are computed without the zero-padded edges.
If this wasn't the case part of the kernel would overlap with non-existing pixels.

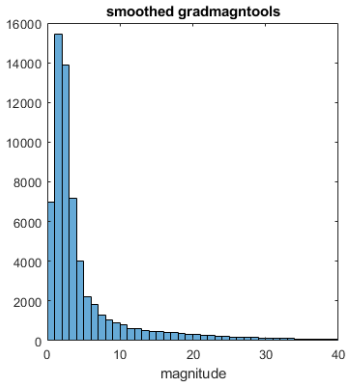
Question 2: Is it easy to find a threshold that results in thin edges? Explain why or why not!

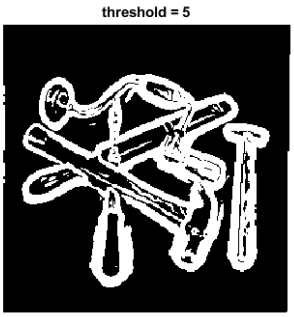
Answers:

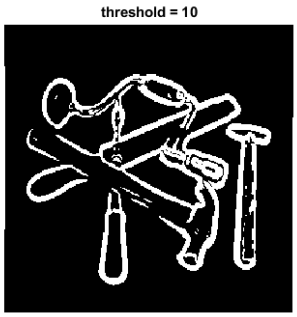

original
x
y
50
100
150
200
250

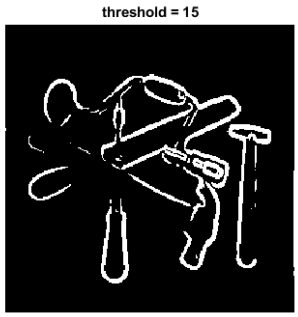

gradmagntools

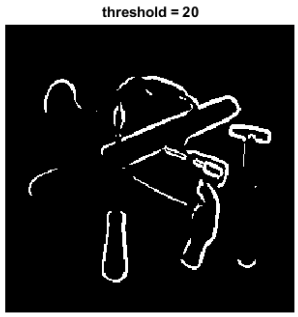

smoothed gradmagntools

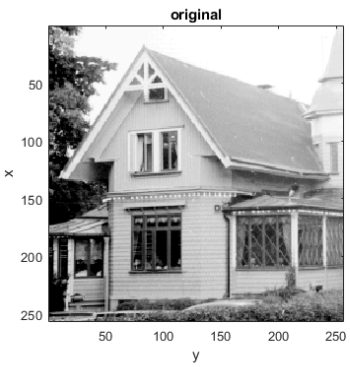

smoothed gradmagntools
16000
14000
12000
10000
8000
6000
4000
2000
0
0
10
20
30
40
magnitude

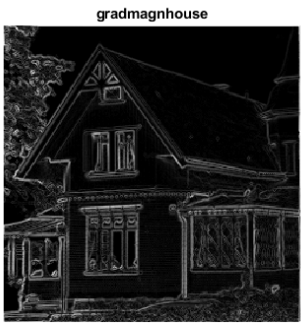

threshold = 5


threshold = 10

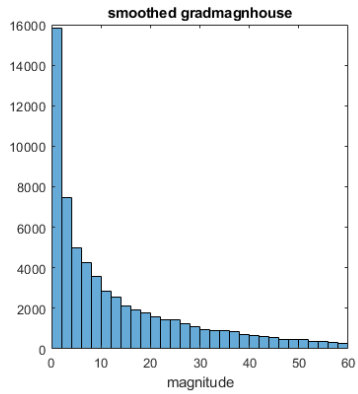

threshold = 15


threshold = 20

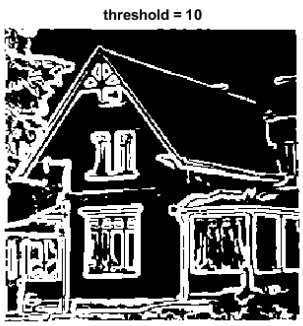

original
x
y
50
100
150
200
250

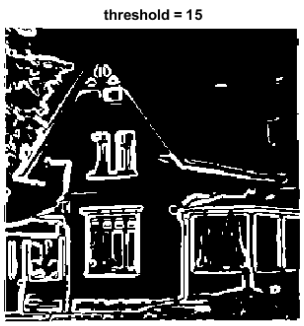

gradmagnhouse

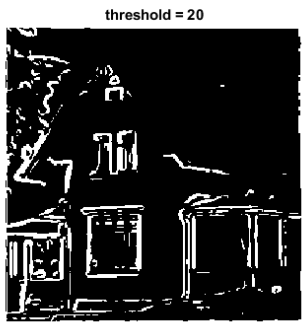

smoothed gradmagnhouse


smoothed gradmagnhouse
16000
14000
12000
10000
8000
6000
4000
2000
0
0
10
20
30
40
50
60
magnitude


threshold = 5


threshold = 10

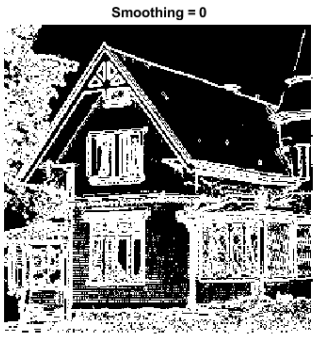

threshold = 15

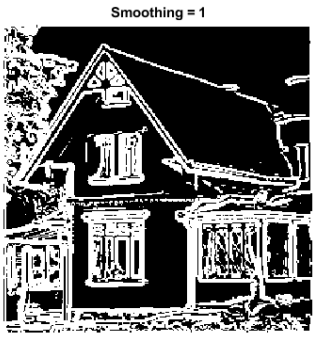

threshold = 20

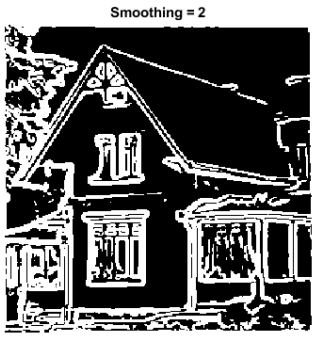
No, because each image is different and you have to guess the threshold for each image. Noise in the image will disappear, so will edges that are already thin. A smaller value for the threshold will result in wider lines and therefore more edges to be found. A specific threshold for an image has to come from trial and error in order to get a good value for that image.

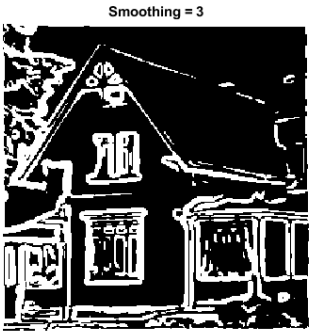
Question 3: Does smoothing the image help to find edges?

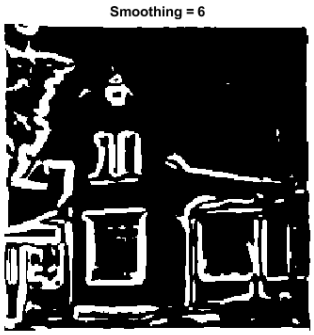
Answers:

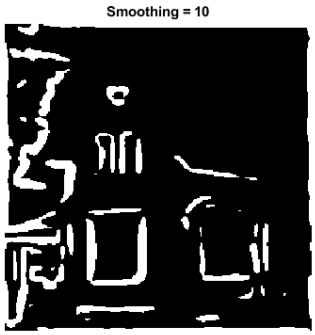












The image is of high complexity with a lot of details. This level of fine detail is often undesirable in edge detection because it tends to act as noise, which is enhanced by derivative computations and thus complicates detection of the principal edges in an image. Smoothing is often used as a pre-processing step for edge detection for noise reduction.
Smoothing could also be a good idea when we only want to detect the most prominent lines in an image, since the smoothing usually doesn't affect these lines too much but at the same time reduces noise.

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

Answers:
The image we are looking at represents the second order derivative of the smoothed intensity function,

$$L_{vv} = \begin{bmatrix} \frac{\partial^2 L}{\partial x^2} \\ \frac{\partial^2 L}{\partial y^2} \end{bmatrix}$$

This combined with the definition of the gradient,

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial x} \\ \frac{\partial L}{\partial y} \end{bmatrix}$$

allows us to see the image as a representation of the derivative of the gradient magnitude. The derivative of the gradient vector is zero when we encounter a minimum or maximum point i.e. an edge. It is these points (contours) that are plotted in green in the image.

With increased scale (blur) less edges are detected. This will also reduce the noise that is brought out by the lines shown in the image. To a certain point an increased scale will lead to clearer lines, where the desired lines are more highlighted. However, if the image is blurred too much too much information will be lost and the contouring will not be able to show even formerly sharp lines in the image.

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

Answers:
An increased scale will increase the amount of blurring that is done to the image. This will reduce the amount of lines by only having the most distinguishable lines show on screen. The lines also become thicker, this is due to the edges being smoothed and therefore result in a derivative that doesn't change as rapidly. For the first figure a scale of between 1 and 4 would be the most reasonable for this image.
The contouring is done with $\tilde{L}_{vv}$, which is the second order derivative

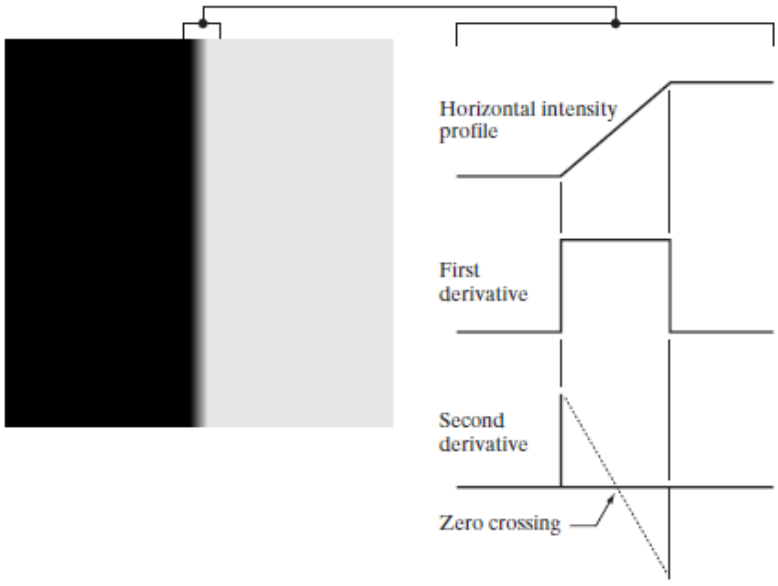
$$\tilde{L}_{vv} = L_x^2 L_{xx} + 2L_x L_x L_{xy} + L_y^2 L_{yy} = 0$$

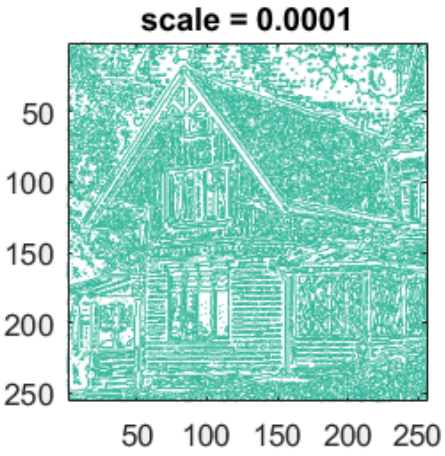
The other two figures are done with $\tilde{L}_{vvv}$, which is the third order derivative

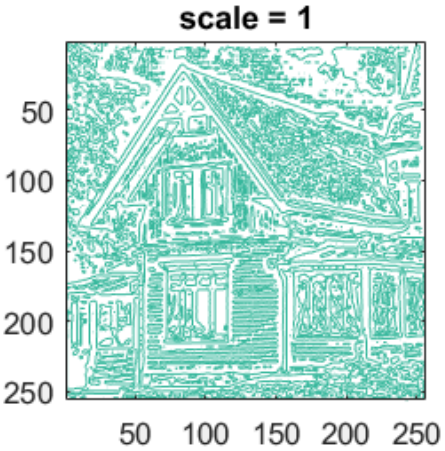
$$\tilde{L}_{vvv} = L_x^3 L_{xxx} + 3L_x^2 L_y L_{xxy} + 3L_x L_y^2 L_{xyy} + L_y^3 L_{yyy} < 0$$

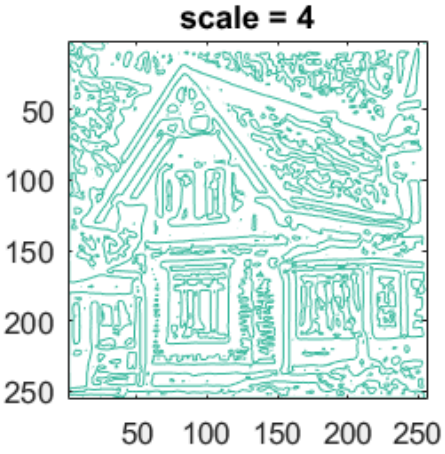
Only the signs of the derivatives are of interest for the edge definition, that is why edges can be defined this way.

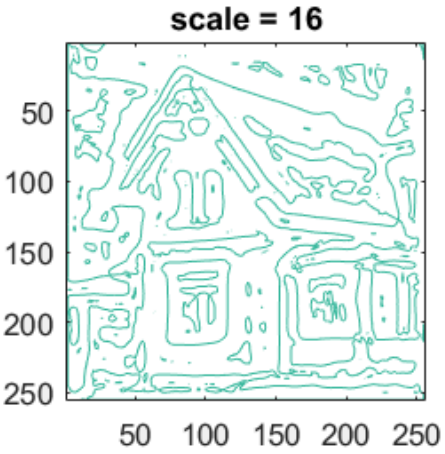
In the tools image, $\tilde{L}_{vvv} < 0$ is represented as white, while $\tilde{L}_{vvv} > 0$ is represented as black.

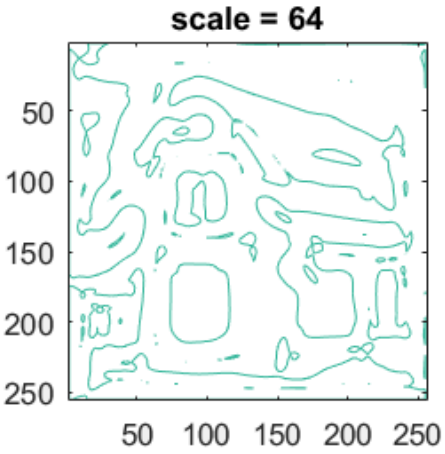



scale = 0.0001
50
100
150
200
250
50 100 150 200 250

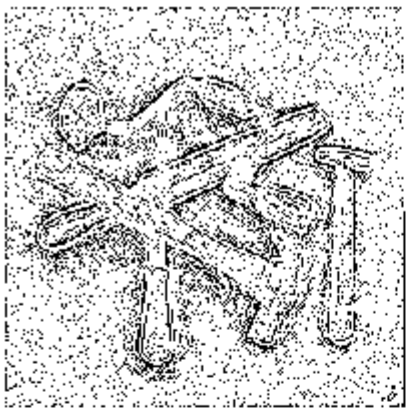

scale = 1
50
100
150
200
250
50 100 150 200 250


scale = 4
50
100
150
200
250
50 100 150 200 250

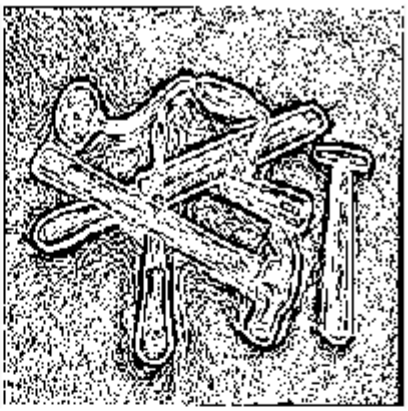

scale = 16
50
100
150
200
250
50 100 150 200 250


scale = 64
50
100
150
200
250
50 100 150 200 250

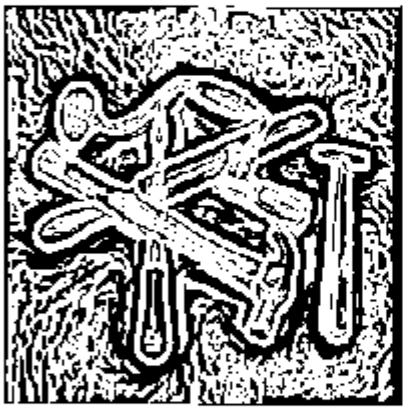
scale = 0.0001



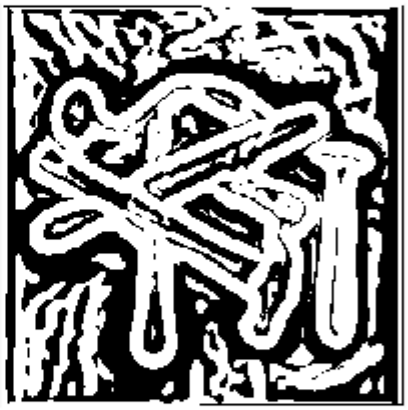
scale = 1



scale = 4



scale = 16



scale = 64

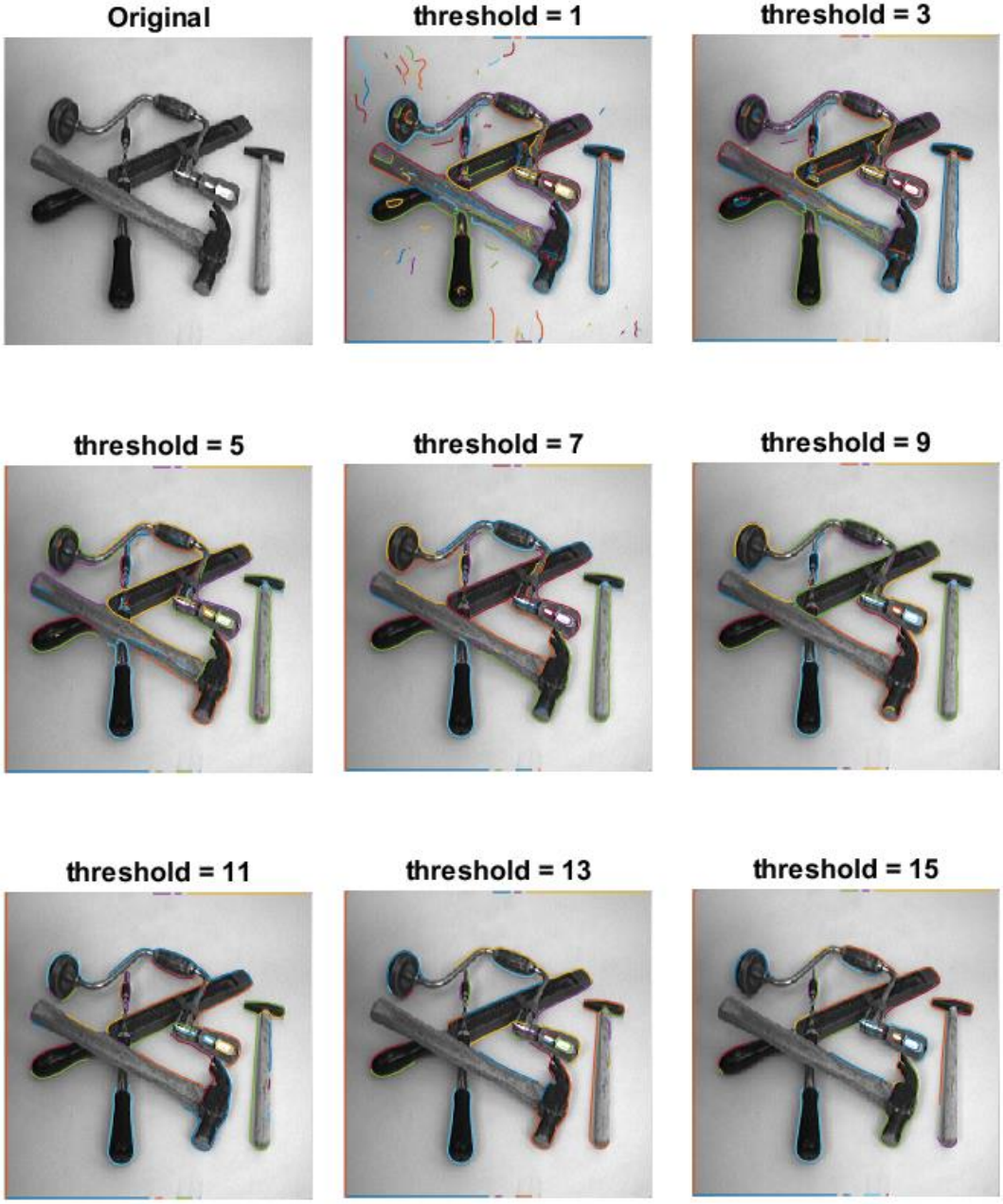
Question 6: How can you use the response from *Lvv* to detect edges, and how can you improve the result by using *Lvvv*?

Answers:

Lvv = 0 will find the points were the gradient magnitude Lv has an extreme value (zero-crossing). Lvvv < 0 will show the maximum values. A combination of Lvv = 0 and Lvvv < 0 will find the maximum points of the gradient magnitude Lv, these points are defined to be edge points (Lecture 6). As commented on before, an increased blurring will show more distinguishable lines, those are usually the larger ones in an original image.

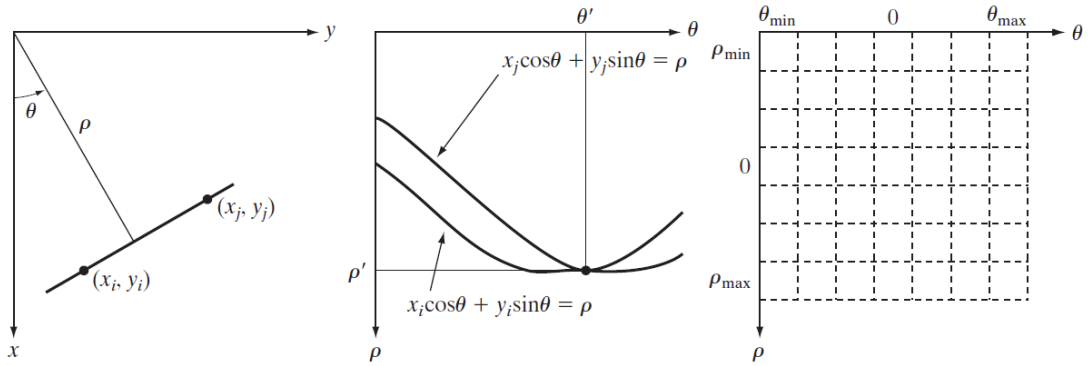
Question 7: Present your best results obtained with *extractedge* for *house* and *tools.*

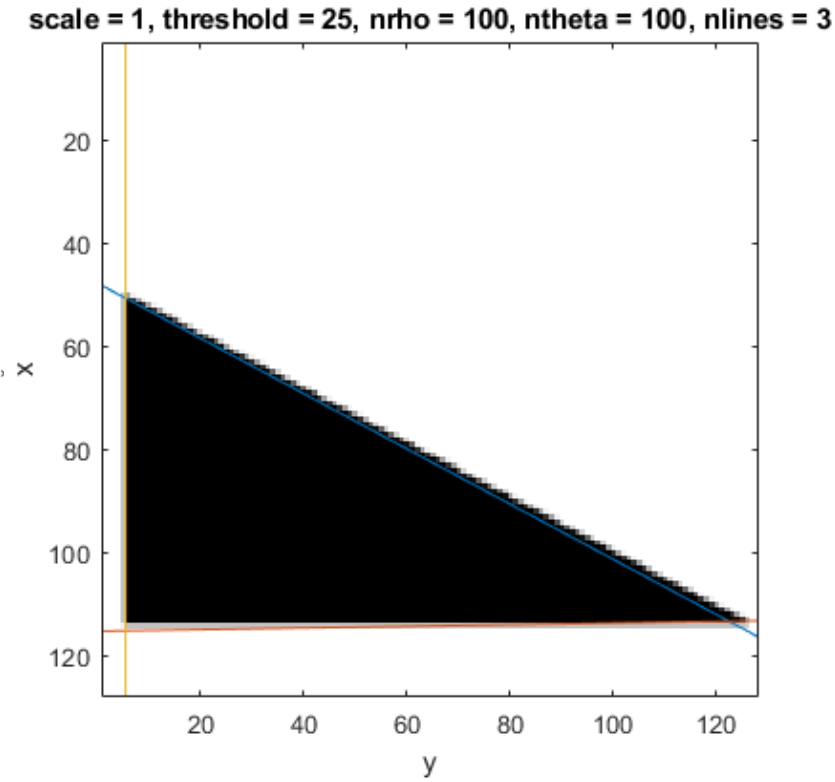
Answers: The scale is 4. The function only shows edges that have a Lv larger than a given threshold. By doing this some false positives can be removed, for example by getting rid of the confetti in the image for threshold 1. However, a threshold value too large will result in fewer edges being shown, since they do not fulfil the given threshold.

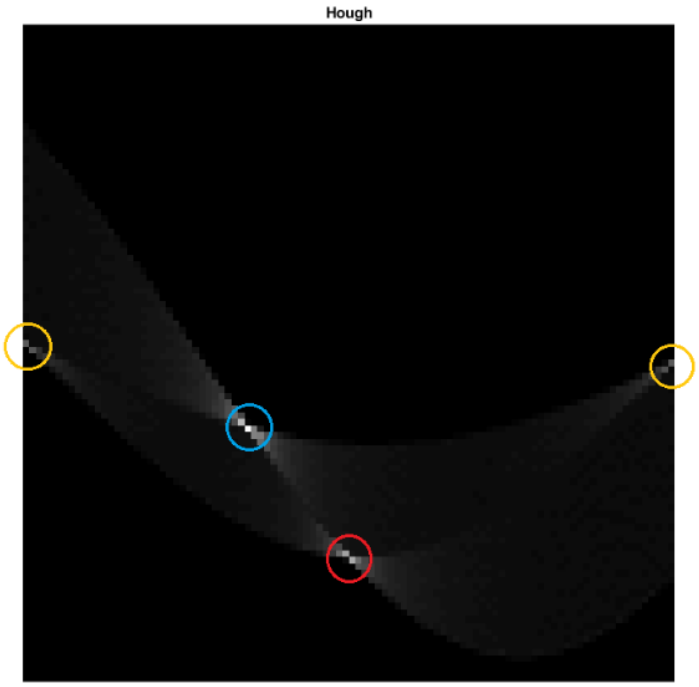


Question 8: Identify the correspondences between the strongest peaks in the accumulator and line segments in the output image. Doing so convince yourself that the implementation is correct. Summarize the results of in one or more figures.

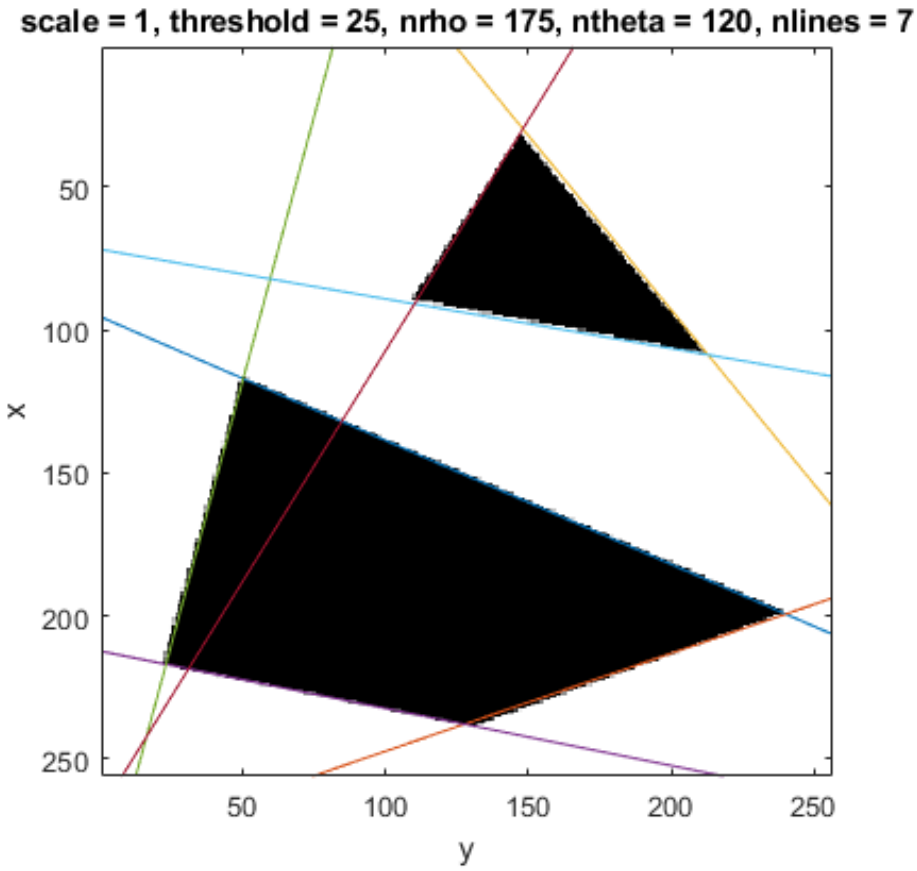
Answers:

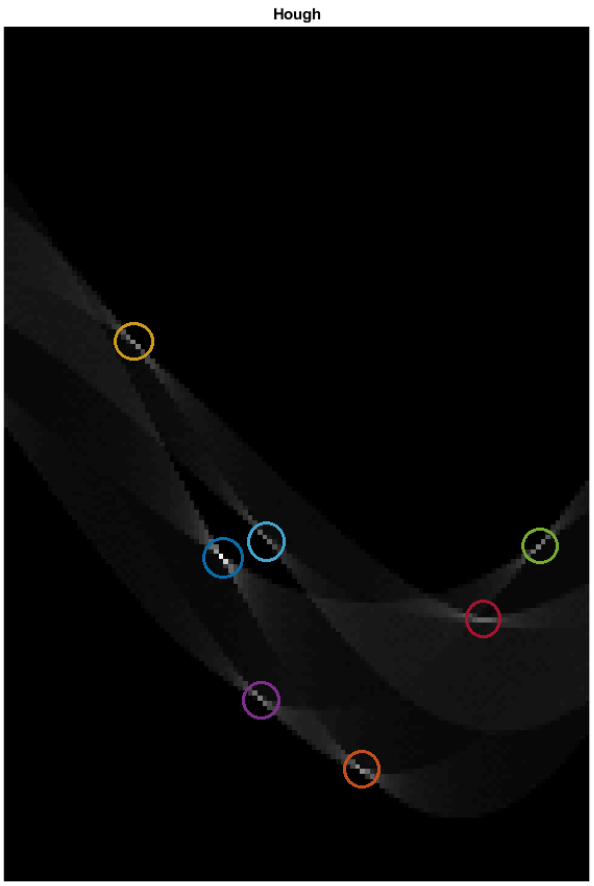




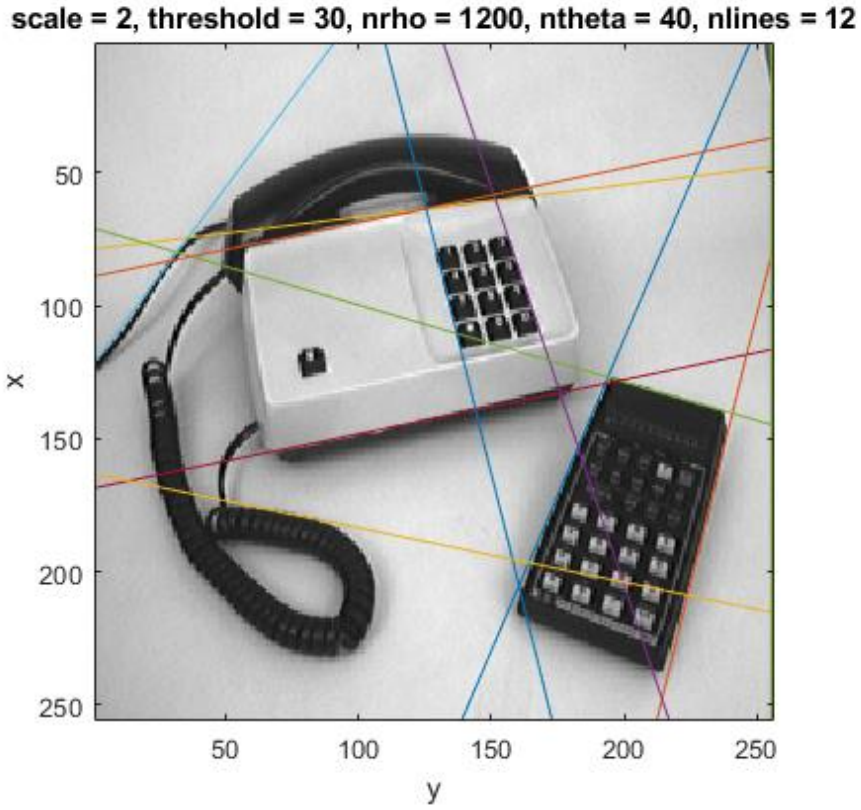


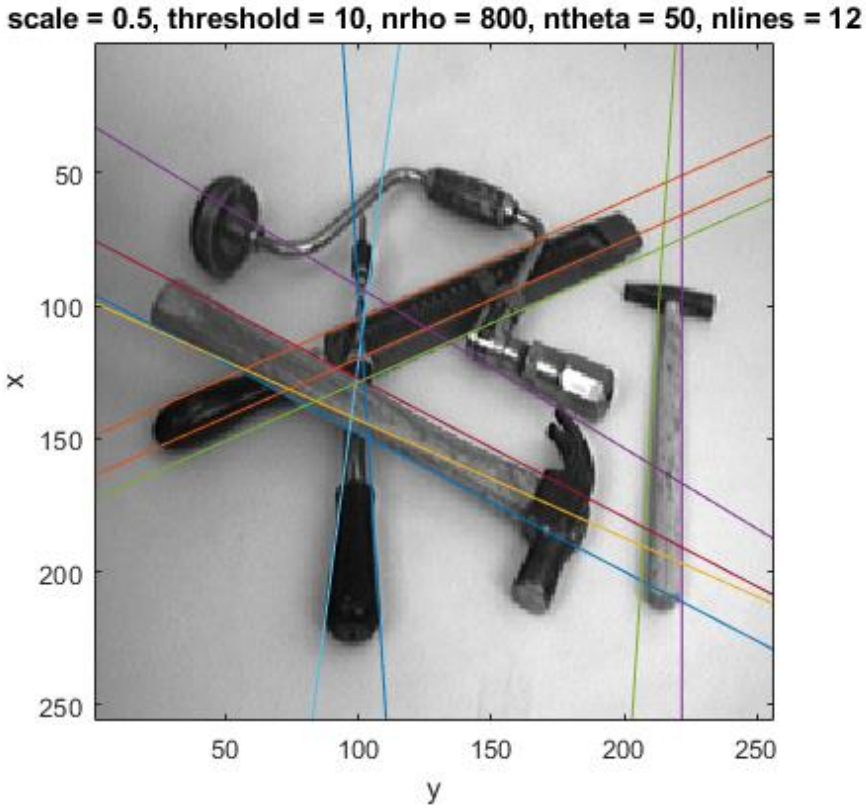
Theta is defined as -pi/2 to pi/2 on the horizontal axis, which is why the most vertical line is represented by the two outermost bright spots in the Hough diagram. The horizontal line has a zero-degree slope meaning it is in the middle in the Hough diagram. The sloping line has a negative slope, also seen by being in the middle of 0 and -pi/2. The same way of the understanding the lines from the Hough diagram in the second picture can be done.

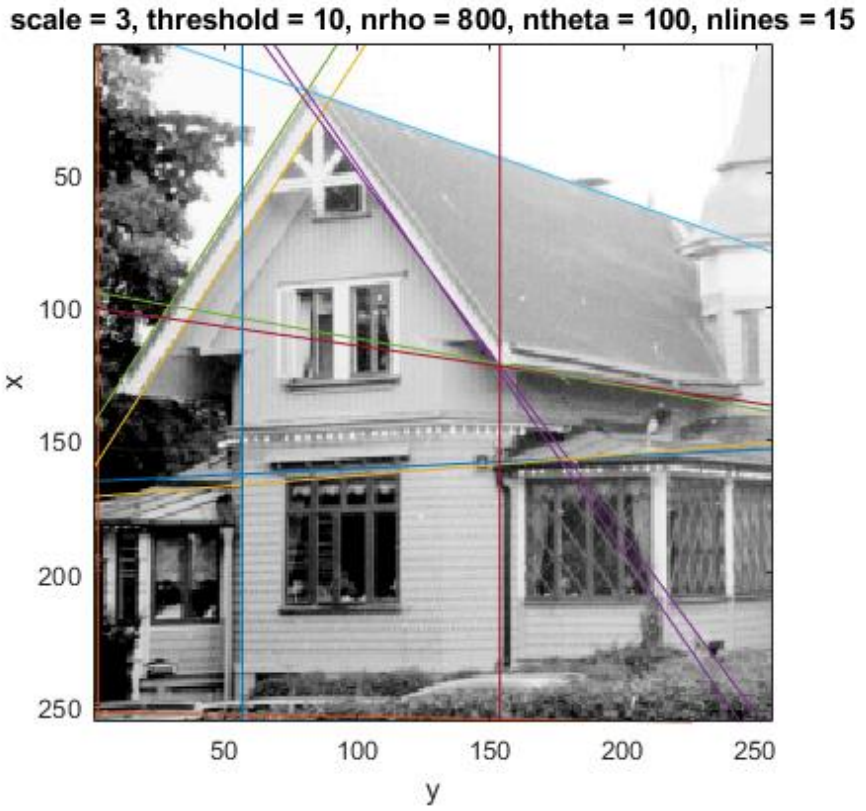




A conclusion that can be made from both images is that longer edges correspond to brighter spots in the Hough diagram. This is because there have been more votes on that specific point (line).


scale = 2, threshold = 30, nrho = 1200, ntheta = 40, nlines = 12
50
100
150
200
250
x
50
100
150
200
250
y

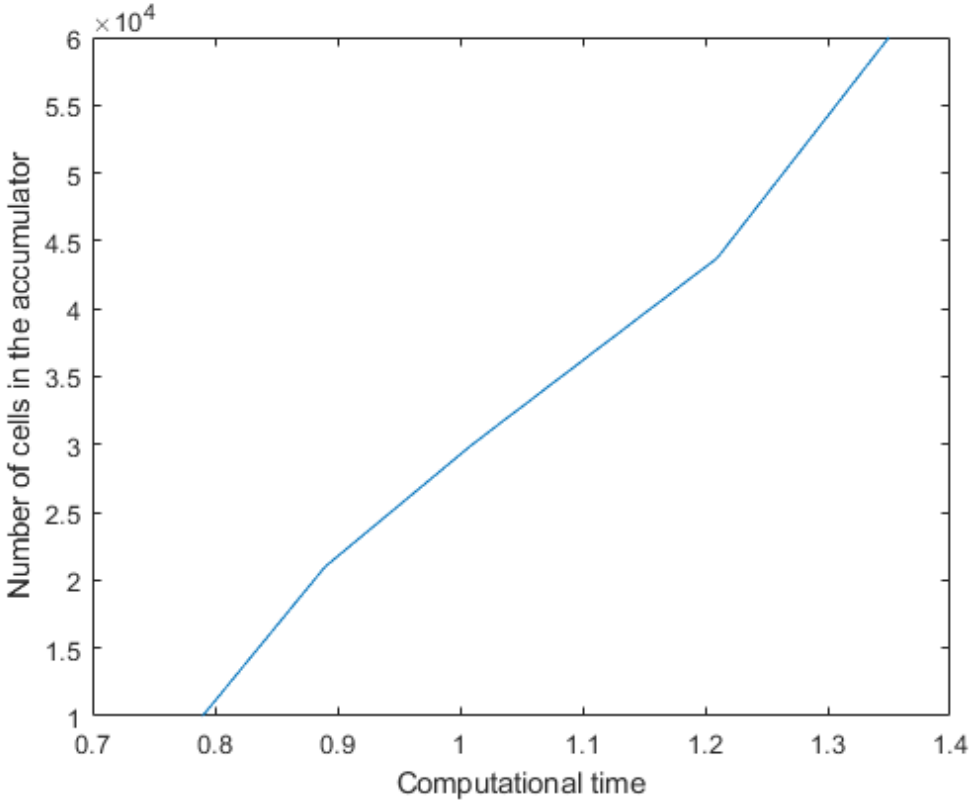




A coarser discretization of rho and theta in the accumulator matrix results in lines of lower accuracy. On the other hand, fine discretization may lead to multiple responses for the same edge.

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

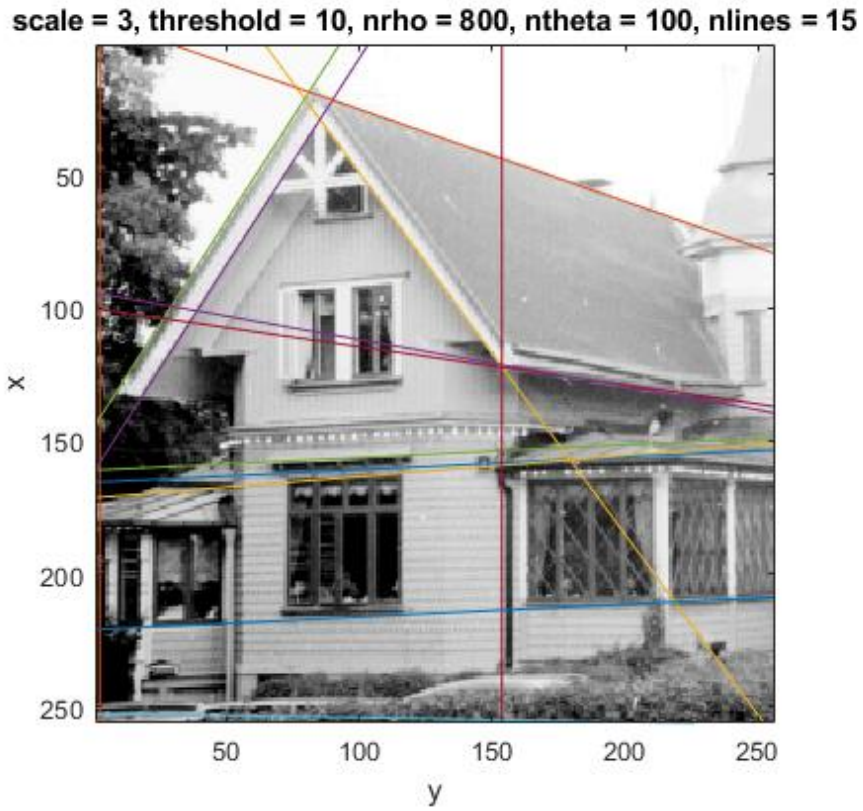
Answers:



The computational time depends more or less linearly on the amount of accumulator cells. A too small value of nrho and ntheta will lead to the resolution of the line being too rough and the line direction will not be sufficient. A too large nrho and ntheta will lead to for example multiple responses for the same line in the image. A specific nrho and ntheta has to be found for each image manually.

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:



The idea is to have a weighted voting system depending on the magnitude of the gradient in that point.
Voting with a logarithm will increase the chance of finding edges with lower values since the characteristics of log will reduce the difference between high and low intensities. This is done by voting with log(gradmagn) instead of the usual +1.
Increment with gradient magnitude or some other weighting would reduce the critical dependency of thresholds.
An increment only for those rho values that seem reasonable would use information about gradient direction from edge detection.
An increment accumulator that is not only point wise but also has some windowing function, like a gaussian blur, would be the same as smoothing after the voting is done which is more effective.
