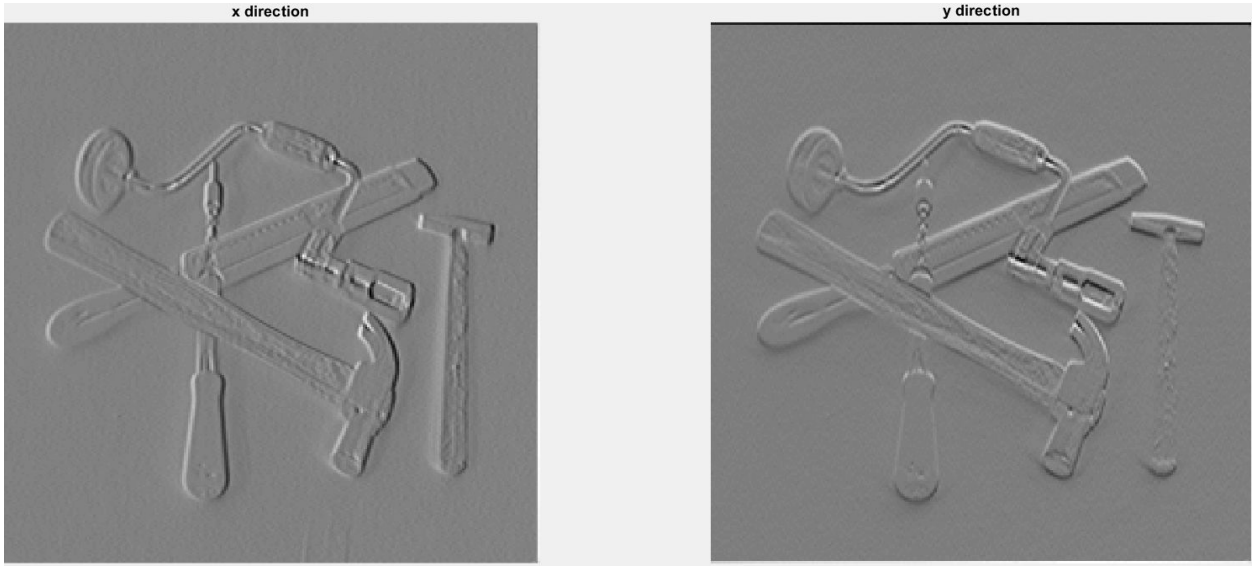
Question 1: What do you expect the results to look like and why? Compare the size of dxtools with the size of tools. Why are these sizes different?

Answer:

Expected results: Simple difference operator

- In the x-direction our discrete derivative approximation gives stronger response for vertical lines. The opposite for our discrete derivation approximation in the y-direction.
- Expect to show large changes in intensity near edges(magnitude of derivative is large)
- The magnitude of the resulting image will be around 0 for most part of the image.

This is because the difference operators emphasizes rapid changes in intensity in the image associated with borders.



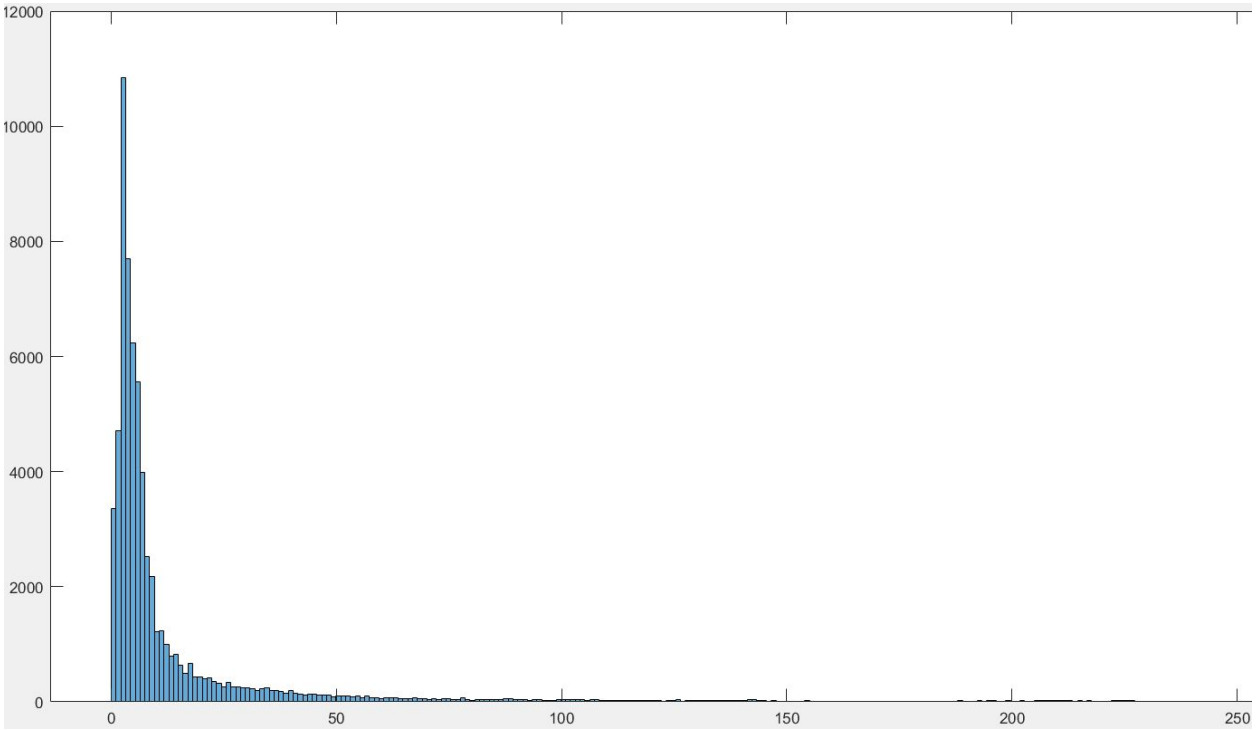
The size is different because of the convolution operation. In Matlab we used conv2(A, B). When A and B are matrices, then the convolution C = conv2(A, B) has size:

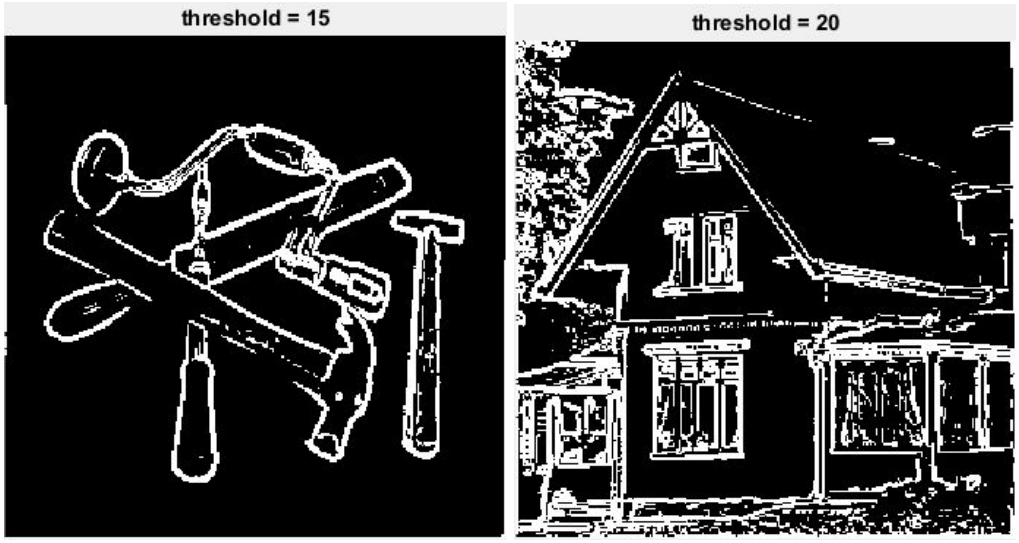
$$size = \#columns(A) + \#columns(B) - 1.$$

In our case the image is 256x256 and the kernel is in the simplest case is 1x2 in the x direction and 2x1 in the y direction. The full convolution then returns an image of size 256x257

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

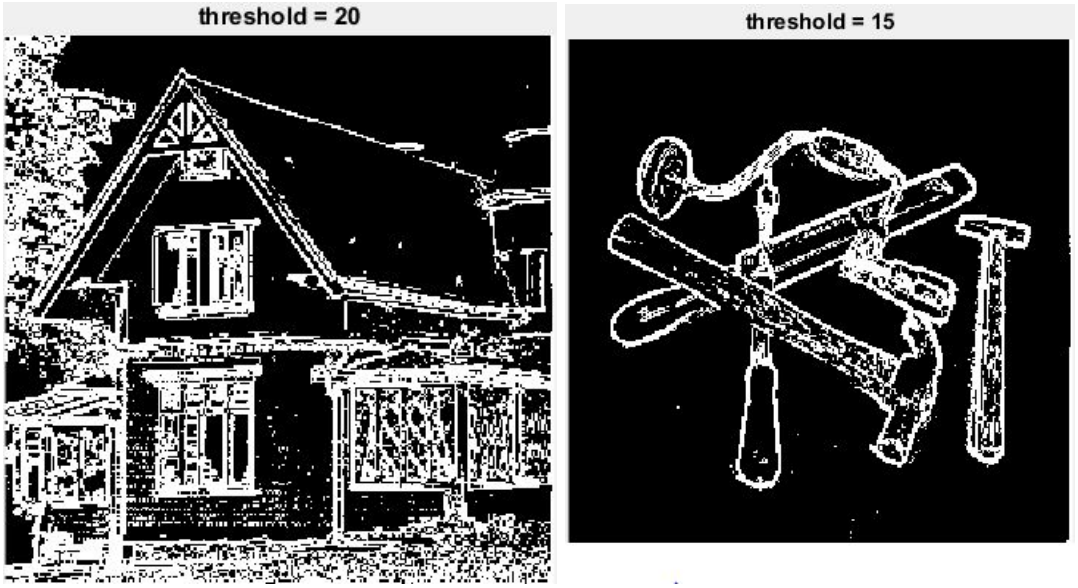
The histogram of the magnitude shows us that most pixel values are concentrated close to 0. This is an expected result because most part of the image is dark. A sensitivity threshold ignores all edges that are not stronger than the threshold.
For this reason it is easy to guess a threshold that yields in reasonably thin edges. Although it is hard to get perfectly defined and thin edges without losing some information. This is because edges contains both light and darker pixels. Lower threshold gives wider edges and is used for *high edge sensitivity*. Higher threshold gives thinner but edges bergin to disappear, *low edge sensitivity*.





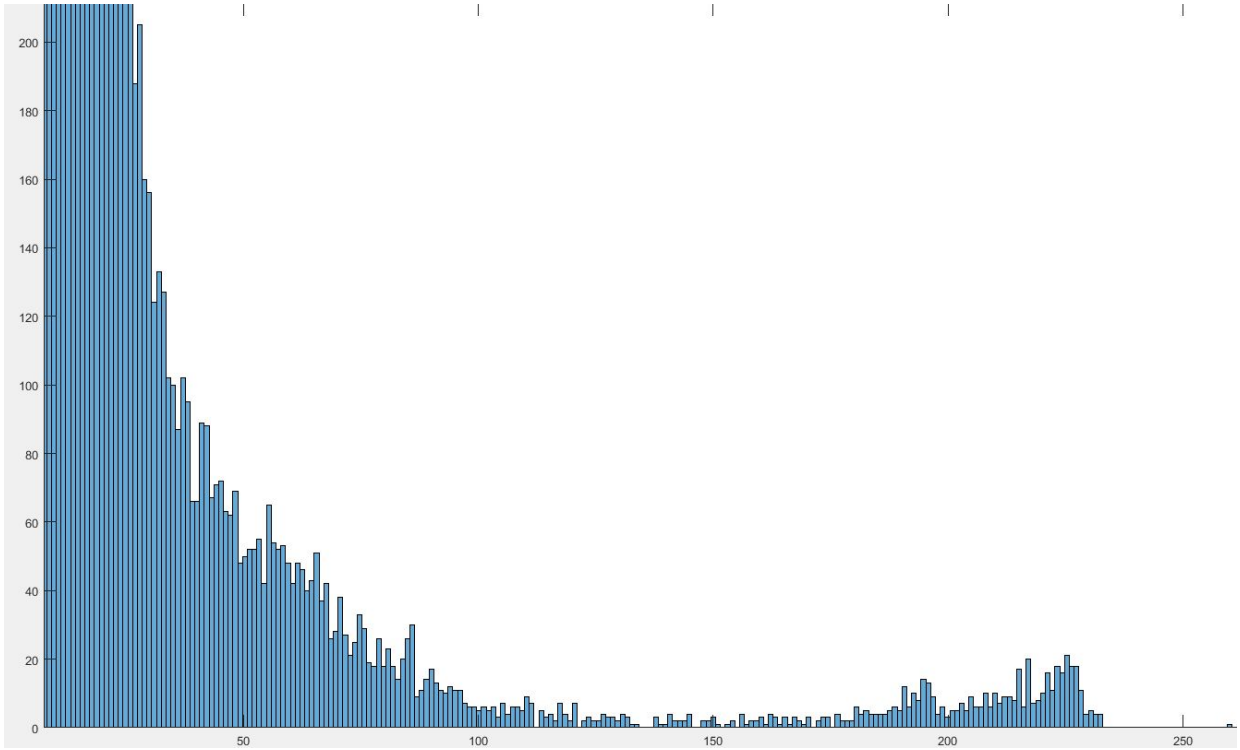
Question 3: Does smoothing the image help to find edges?

The difference operators allows us to detect edges but the approximated discrete derivative is sensitive to noise, especially for smaller kernels. The produced image is then affected by that noise. From the previous lab we concluded that the gaussian filter reduce noise in the image . It is therefore necessary to preprocess the image with a smoothing filter before to achieve a better result. Because we concluded it removes high-freq noise. Without pre-filtering we have distortion from noise. Notice that smoothing to much will blur the edges if using a gaussian-filter. Another effect is it is more easy to find a good threshold.
Same thresholds without smoothing, we have more noise.

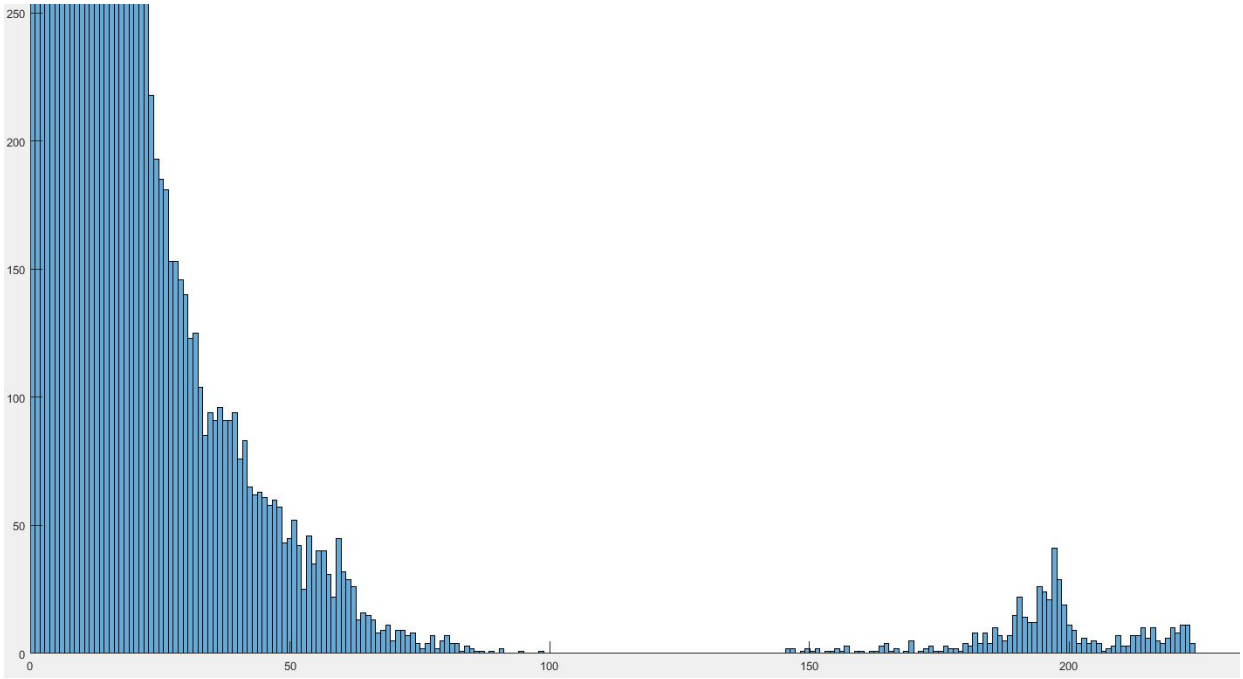


Histogram comparison:

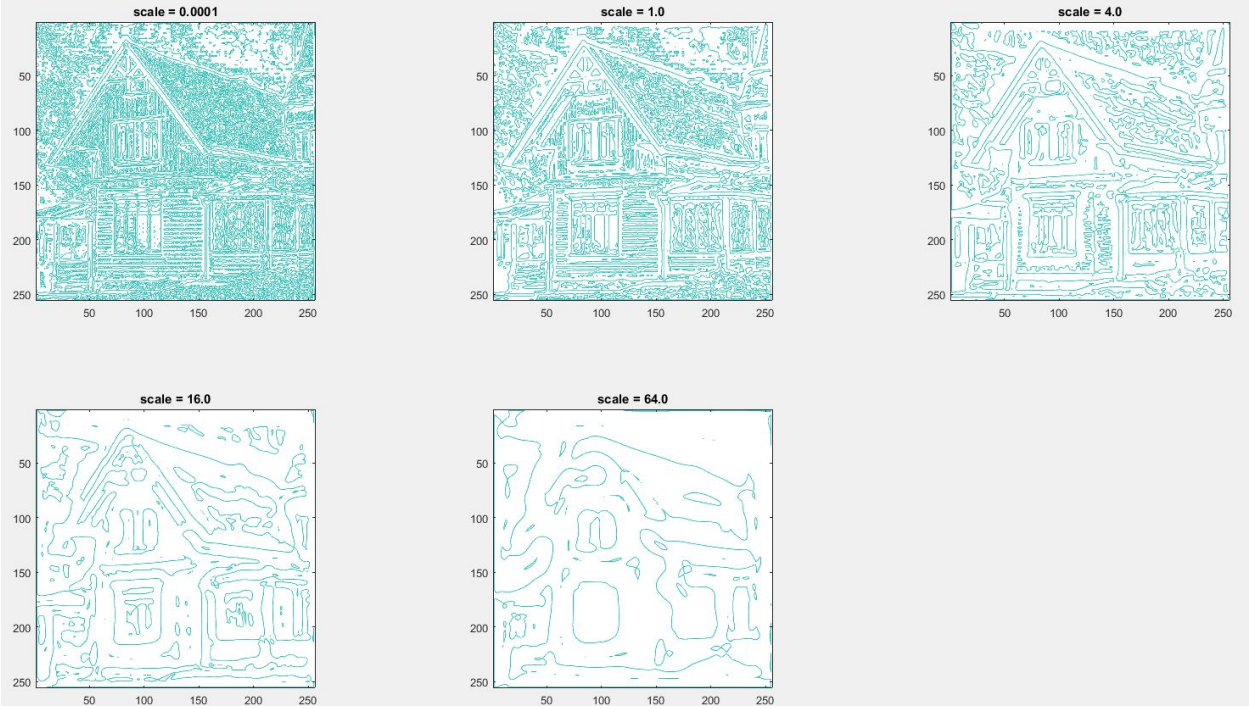
Tools: No smoothing



Tools: With smoothing



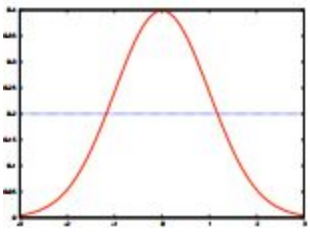
Question 4: What can you observe? Provide explanation based on the generated images.



Observations:

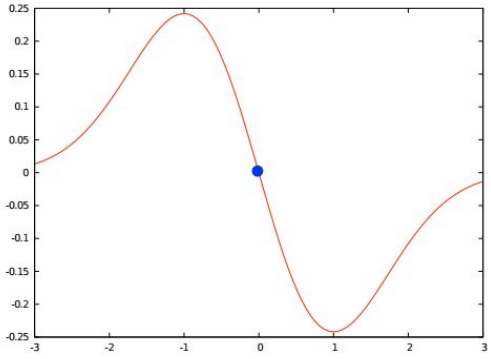
- Small scale returns a noisy image where every tiny detail(difference) is shown
- Overly large scale returns an image where borders are faded and distorted.
- Middle values of the scale return an image with edges more clear and less noise.
- All contours i.e zero-crossings are in the form of closed curves, if it not exceeds the edge of the image.

Explanation:



At an edge the derivative looks like a gaussian. Where the peak corresponds to the edge.

The second derivative of a gaussian has the form of a Wave. The peak of the first derivative corresponds to the 0 in the second derivative.



The function contour(Z,V) draw a contour line for each

level specified in vector V. In our case contour(IMG ,[0 0]) drew contours for every single level in v.

When we had a small scale we registered many peaks in the first derivative due to the sensitivity of the derivative to the noise.

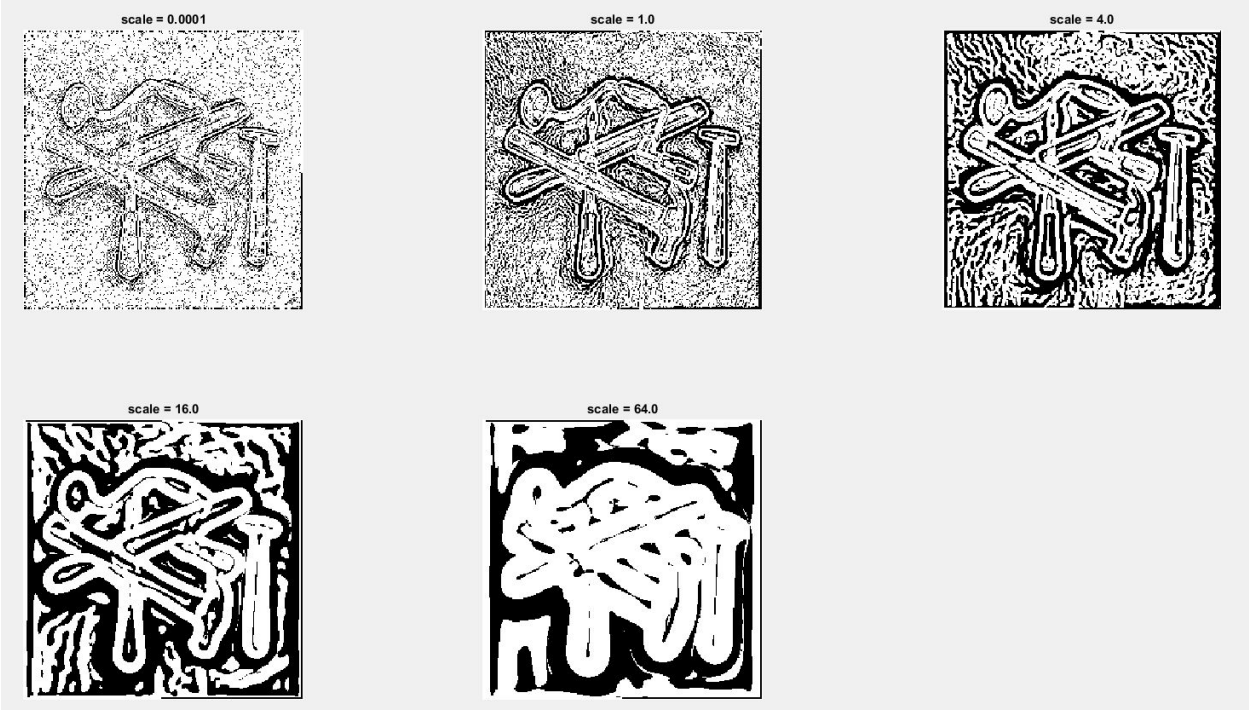
In the second derivative we then had many zero-crossings which then resulted in more lines in the contour plot of the image.

When we smooth we neglected some of the noise but with scale values too high we distorted the image and lost too much information.

Question 5: Study the sign of the third order derivative in the gradient direction by loading the image. Assemble the results of the experiment above into an illustrative collage with the subplot command. Which are your observations and conclusions?

Observations

- Noise for small scales
- By increasing the scale the borders become wider
- The white area = Lvvv < 0 = positive edges; gradients of the second derivative < 0
- More smoothing = more emphasized edges
- Too much smoothing and the image is unrecognizable

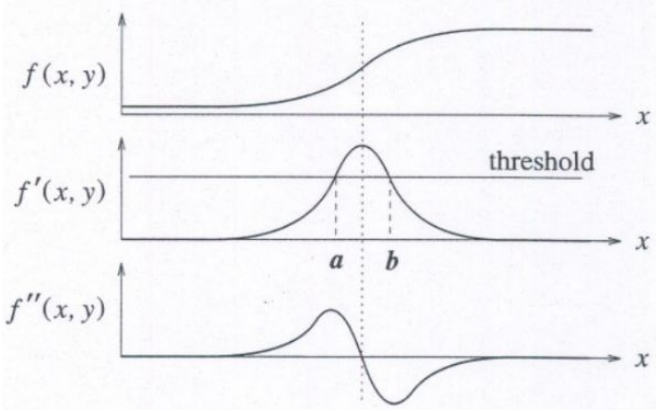


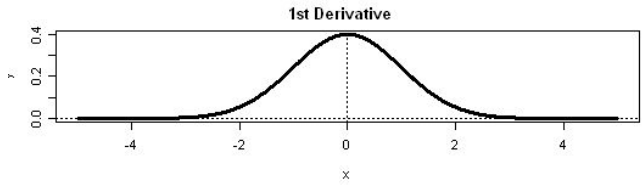
Question 6: How can you use the response from ~Lvv to detect edges, and how can you improve the result by using ~Lvvv?

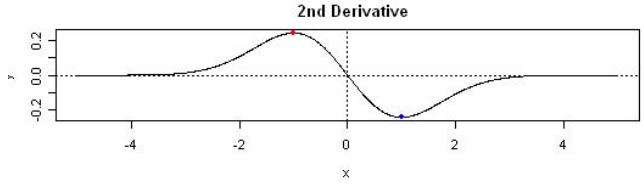
If the first derivative a relative maximum, the second derivative must be 0 and the third derivative must be negative if the edge is crossed from a lower to a higher value in the greyzone image.
That means positive-edges are from lower to higher greyzone values, and negative edges from higher to lower greyzone values.

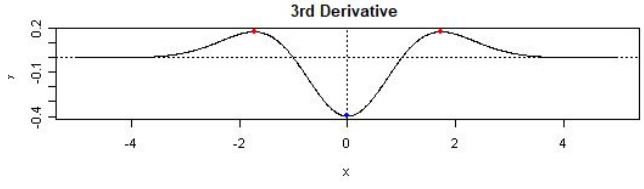
When we study edges, the third derivative is the gradient of the zero-crossing. And in order to consider the zero-crossings of the real edges we need to set a threshold on the third derivative so that only zero crossings of real edges are preserved. This is necessary because the second derivative is even more sensitive to noise in the original image than the first derivative, it contains more false positives on edges.

Figure: Local maximum in first derivative is a zero crossing in the second derivative.









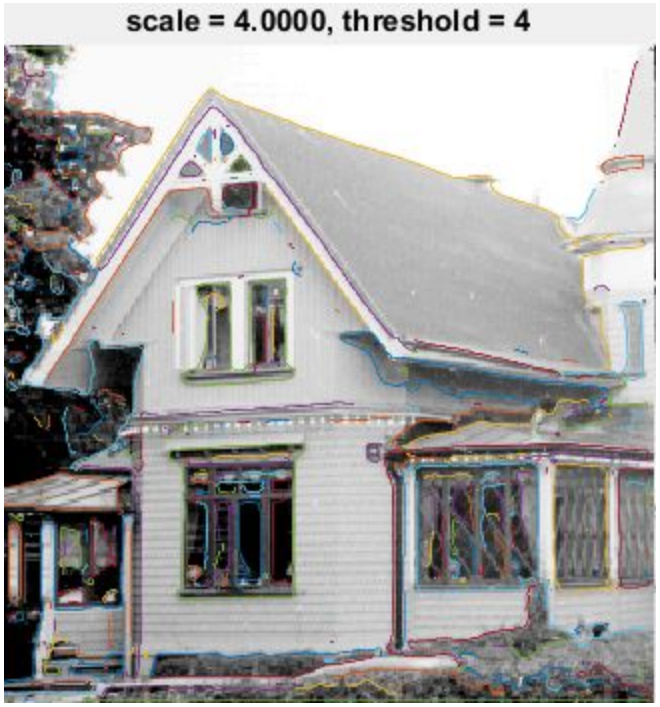
Question 7: Present your best results obtained with extractedge for house and tools.

In order to find the most satisfying result we tried all the previously defined scales with threshold values 1, 4 and 10.

House

- Scale = 0.001 → not a good result for any tested value of the threshold, due to noise
- Scale = 1 → best result for threshold = 10, clearly defined edges although sometimes discontinued
- Scale = 4 → best result for threshold = 4, well defined border with no almost no weak edges showing
- Scale = 16 → best results for threshold =1, not as good of a result as the previous.
- Scale = 64 → Not a good result for any tested value of threshold due to too much smoothing.

Most satisfying was smoothing with variance = 4 and applying a threshold value of 4.

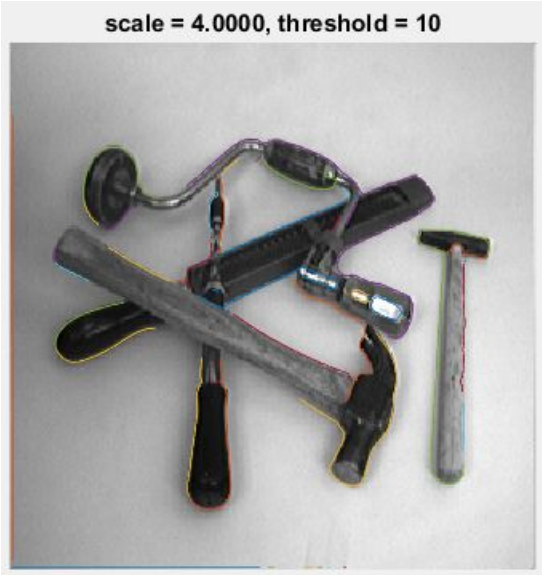


Tools

- Scale = 0.001 → not a good result for any tested value of the threshold, due to noise
- Scale = 1 → best result for threshold = 10, Good result
- Scale = 4 → best result for threshold = 10, Better than previous result, clearer edges.
- Scale = 16 → best results for threshold =4, not as good of a result as the previous. (Better to have a higher threshold instead of extra smoothing)

- Scale = 64 → Best result for threshold =1 but not a good enough result for any tested value of threshold due to too much smoothing that caused distortion of the borders.

Most satisfying was smoothing with variance = 4 and applying a threshold value of 10



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 in one or more figures.

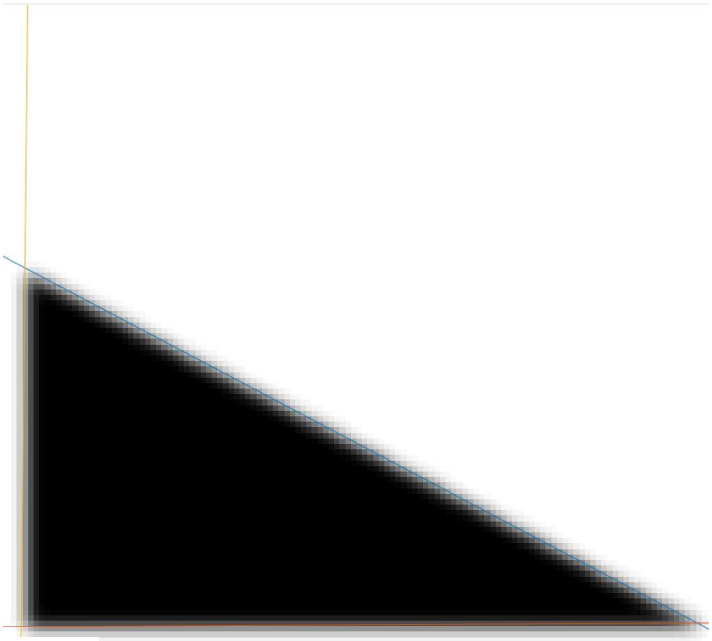
We extracted the edge segments from the image as in the previous question by using extractedge function, we then performed the Hough transform the result by parametrization of the type

$$x \cos\theta + y \sin\theta = \rho$$

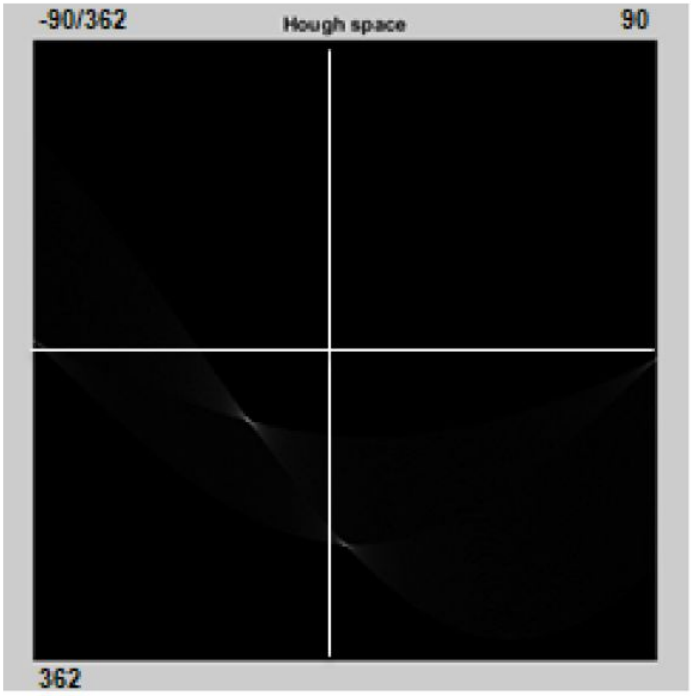
Where x and y are the position of the the edge points. The Hugh space represents points as (ρ, θ) where ρ is the distance from the origin to the closest point in the straight line. And theta is the angle between the origin and the connecting line.

What we did practically was loop through the matrix created by extracting the edges and from that calculating ρ for every possible value of theta. We then introduced a accumulator matrix that forms the Hough Space and updated the index value at the appropriate place. Local maximums in the Accumulator corresponds to the line segments in the output.

Best result:



pic = triangle128;
scale = 2;
nrho = 300;
ntheta = 300;
gradmagnthreshold = 20;
nlines = 3;
verbose = 2;



Yellow line: Vertical line in image is almost 90 degree angle and small distance, peak can be seen near edge but is blocked by coordinate system a bit.
Blue line: negative angle, peak can be seen in third quadrant in hough space.
Red line: angle almost 0 degrees, large distance peek can be seen just inside fourth quadrant.

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

We defined the accumulator as

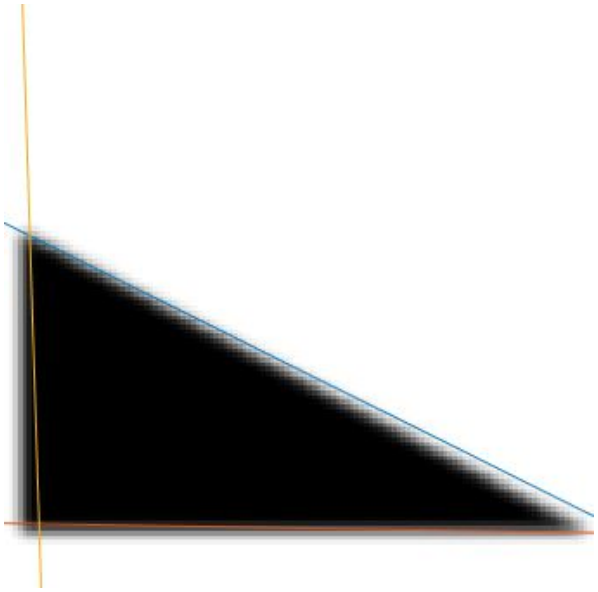
```
acc = zeros(nrho, ntheta);
```

For each coordinate point in the extracted edge matrix we perform loops and calculations that are dependent on nrho and ntheta, that also defines the cells in the accumulator.

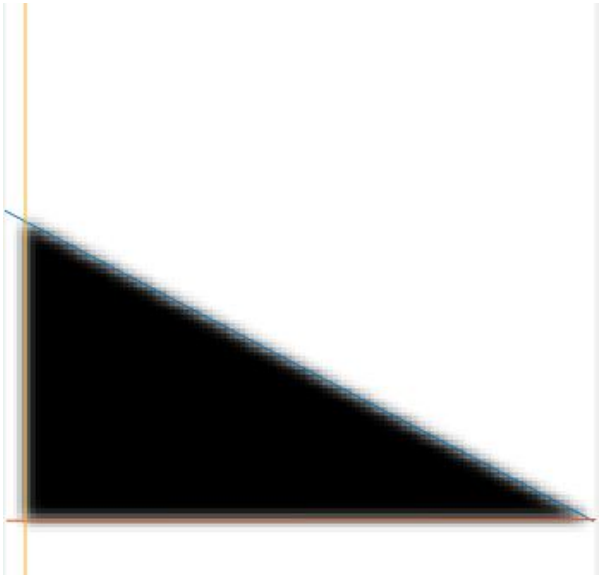
- By increasing the number of cells the we have longer computational times
- By decreasing the cells and keeping all other parameters static we achieve a worse result

Conclusion: Compromise between good result and short computational times

nrho = 100;
ntheta = 100;



nrho = 1000;
ntheta = 1000;

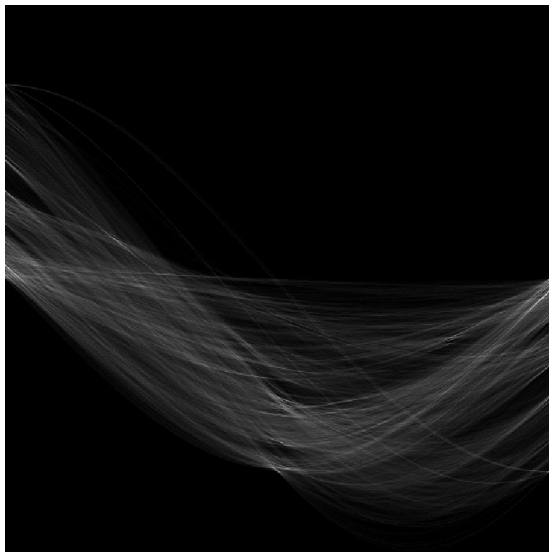


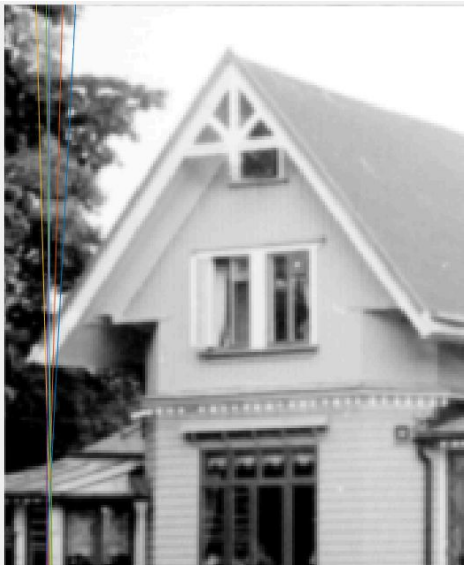
Question 10: How do you propose to do this? Try out a function that you would suggest and see if it improves the results. Does it?

Suggestions on monotonical functions on the magnitude could be the magnitude squared and the logarithm of the magnitude.

Function: gradient magnitude squared
Observations

- Noticed more white areas in the parameter space (values are higher in the accumulator)
- The peaks in the Hough domain does not actually correspond to edges in the image domain
- Emphasize the high values of the magnitude more since large values squared takes over
- Phantom lines due to a few very bright points





The gradient magnitude can also be used as a heuristic in the incrementing procedure. Instead of incrementing by unity, the accumulator array location may be incremented by a function of the gradient magnitude. This heuristic can balance the magnitude of brightness change across a boundary with the boundary length, but it can lead to detection of phantom lines indicated by a few bright points, or to missing dim but coherent boundaries.

Function: log(gradient magnitude)
Observations

- Emphasize the low values of the gradient magnitude more, since the difference between large and low values decreases.