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

Name: Alice Karnsund Program: TMAIM-17

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.

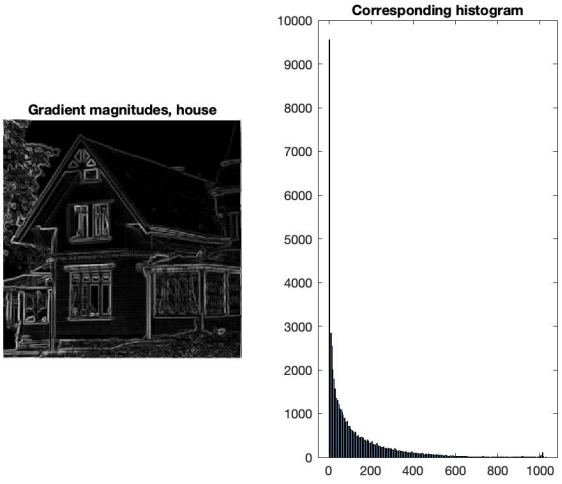
Good luck!

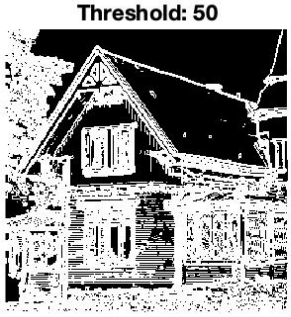
Question 1: What do you expect the results to look like and why? Compare the size of *dxtools* with the size of *tools*. Why are these sizes different?

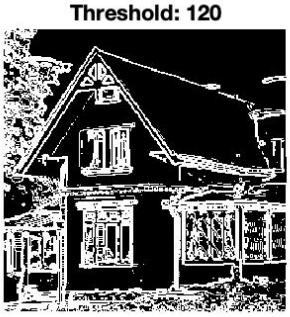
Answers:
Below are the horizontal and vertical derivatives of the 'few256' image, for both central differences and the Sobel operator. We expect the results to enhance only the edges in the image. The difference operators calculates the gradient in the vertical and horizontal direction. A change in magnitude will occur at the edges and thus these points will have increased or decreased gray values (depending on if it is an increase or decrease in the magnitude), as can be seen in the figure below.
Comparing the size of *tools* with either *dxtools* or *dytools* we find that *tools* has a size of 256×256 and *dxtools* and *dytools* has a size of 254×254. The reduction in size has to do with the fact that we are using a convolution where we only save the parts that are computed without the zero-padded edges. This is specified by the 'valid' argument to conv2. In our case the difference operators are of size 3×3, meaning that the output of the convolution will shrink by 2 in both dimensions. Thus one can imagine this as, when the difference operator slides over the image, it will only perform calculations where the whole operator fits in the image. And the output for each position of the operator will be at the center of the mask.

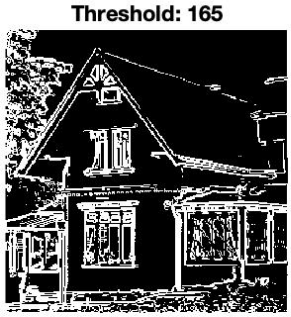
Question 2: Is it easy to find a threshold that results in thin edges? Explain why or why not!

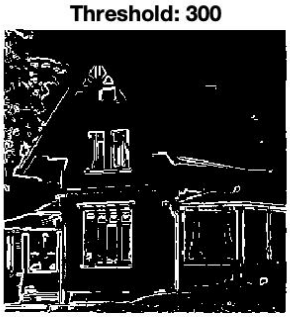
Answers:
The edges in the gradient magnitude image are of different width, intensity and shape (see first figure). Thus a given threshold might result in thinner edges for some and may make others disappear or get fragmented. Due to this it is not easy to find a threshold that all edges benefit from. Most of them will get thinner, while others disappear. Below, in the second figure, is an example of the 'godthem256' image for different thresholds. A lower threshold will allow wider edges, and therefore more edges. Thus the choice of the threshold will depend on the task and how much details one wish to have.





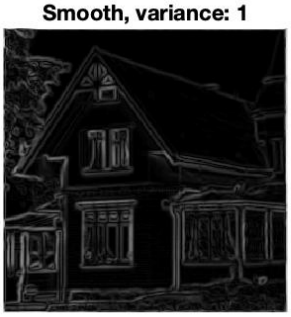


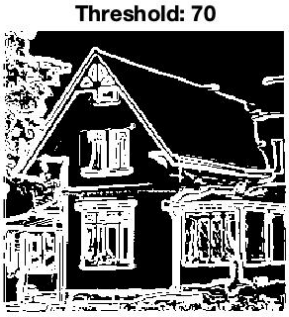


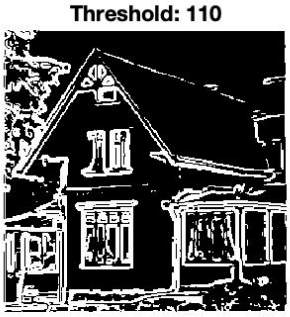


Question 3: Does smoothing the image help to find edges?

Answers:
The Gaussian filter is a low pass filter, that suppresses noise and also details that lives in the high frequencies, but relatively well preserves boundaries and edges. Preceding the differentiation by smoothing will help reduce noise and "false positives". But there is a trade-off problem, increasing the amount of smoothing will suppress noise but also give rise to higher distortions of "true" structures. With a decreased amount of smoothing, more accurate feature detection will be possible, but a higher number of "false positives" may be present. Smoothing can thus suppress details and one must adjust the amount of smoothing depending on what one wish to achieve. To find edges, smoothing may help to promote wider and more intense edges, but thin edges might fade away. Therefor smoothing is a good thing if one only wish to highlight the main structures in the image. When pre-smoothing the image, the threshold for thinning out the edges should be lower, since smoothing will decrease the intensity and sometimes also the width of features and therefore a lower threshold will be needed.

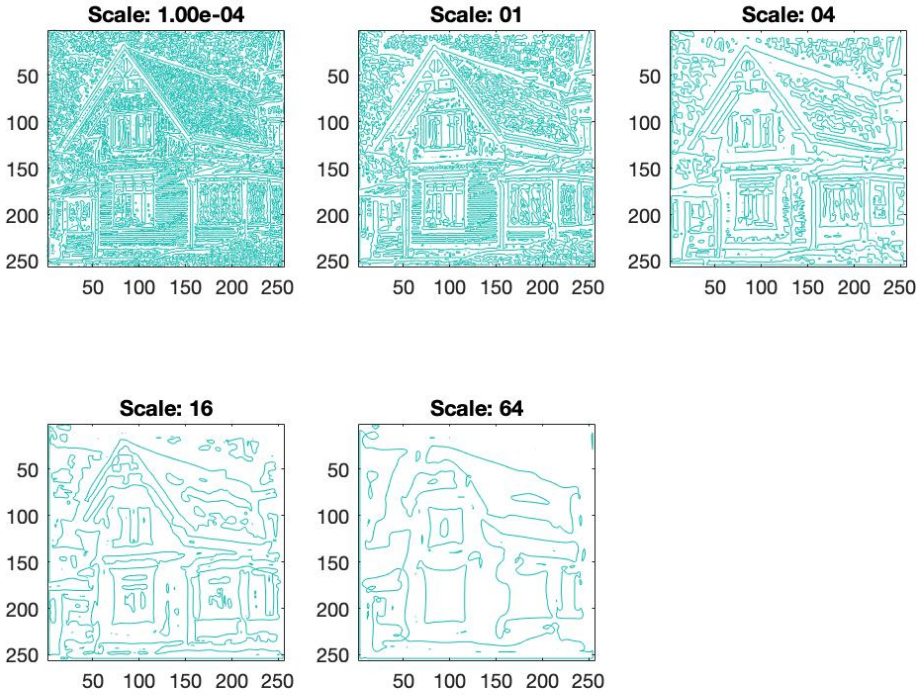






Question 4: What can you observe? Provide explanation based on the generated images.

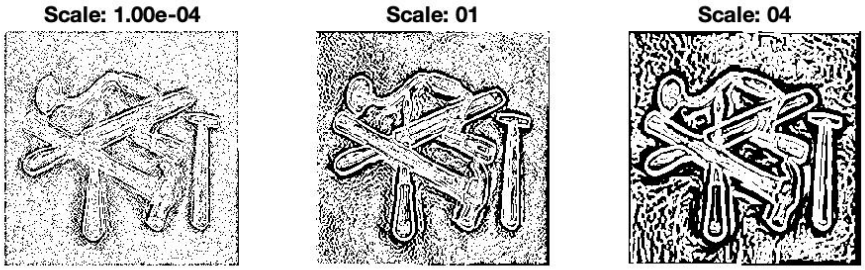
Answers:
Here we are looking at the second derivative of L, the smoothed intensity function, in the v direction. Where v is parallel to the gradient direction in each point. This can also be seen as the first derivative of the gradient magnitude L_v. Thus looking at points where $\tilde{L}_{vv} = 0$, means that we are looking at points where the gradient magnitude is not changing, i.e. we have reached a local min or maximum point, in the gradient direction.
Below we have plotted the contours for $\tilde{L}_{vv} = 0$ for four different scales of the pre-smoothing. Since we are only concerned with points where $\tilde{L}_{vv} = 0$, and not the value L_v in these points, all points will be plotted with the same intensity. And since, no matter the width of the edge in L_v , all will have one extremum in each cut (imagine a slice normal to the curve/edge). Due to this all edges will have the same intensity and width, which can be seen from the images below. In this way we will for sure find all the edges, but this also means noise and smaller details will be enhanced. Which is why, especially the first two images below are pretty messy. Increasing the pre-blurring may help to suppress noise, details and "false positives", and therefore the number of extremums will decrease in $\tilde{L}_{vv}$, which can also be seen in the images below. But too much pre-blurring will itself introduce higher distortions of "true" structures, which is the case where the scale is 16 and 64, below.

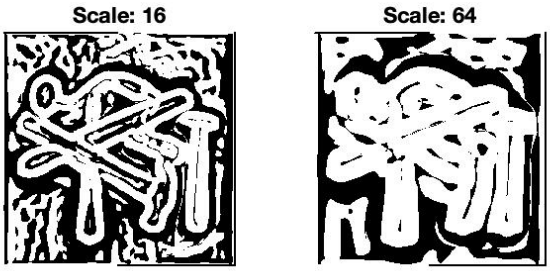


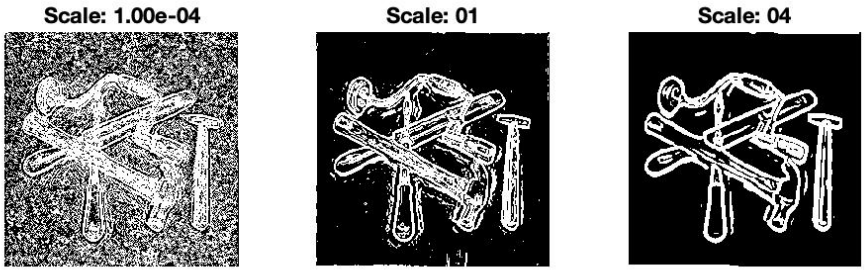
Question 5: Assemble the results of the experiment above into an illustrative collage with the *subplot* command. Which are your observations and conclusions?

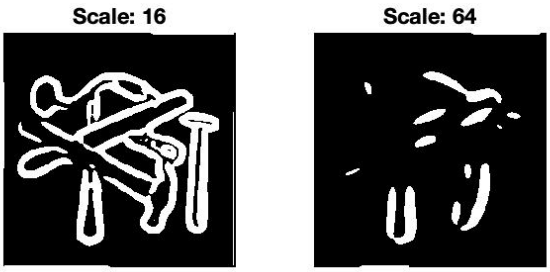
Answers:

The operation "Lvvvtilde(discgaussfft(tools, scale), 'same') $<$ 0)" returns a matrix of the same size as Lvvvtilde(discgaussfft(tools, scale), 'same') but with entries either 1 or 0 depending if the elements fulfill the criteria < 0. Therefor the resulting image will only hold black and white values, where white=1 implies maximum point in L_v. Below, in the first set, are five images showing $\tilde{L}_{vvv} < 0$ for the "tools" image. As can be seen, the borders of the tools are pretty clear, but in the rest of the image there is a lot of noise and "false positives". As the pre-smoothing scale increases, so does the width of the gaussian, then the noisy parts get more smoothed together and in general the areas of black and white increases. Thus with a too big scale, as for example 64, there is no longer an easy interpretation of what the image is showing. One way to get rid of the amount of noise is to decrease the threshold for $\tilde{L}_{vvv}$, meaning that we only keep points that considerably greater peaks than the noise and smaller details. This is shown in the second set of figures, where $\tilde{L}_{vvv} < -10$. Here it is clear that the noisy parts can be suppressed very well by just decreasing the threshold slightly.









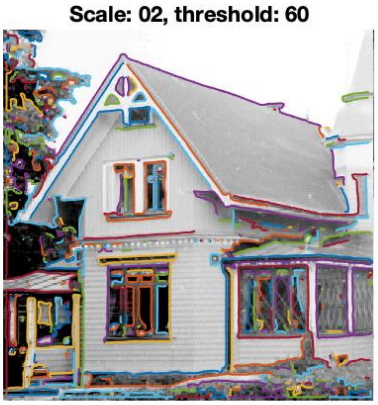
Question 6: How can you use the response from *Lvv* to detect edges, and how can you improve the result by using *Lvvv*?

Answers:

As has already been brought up in the two previous questions. $\tilde{L}_{vv} = 0$ will help us find the points where the gradient magnitude L_v assumes a minimum or maximum value and $\tilde{L}_{vvv} < 0$ will help us pick out the maximum values. Thus combining the results of $\tilde{L}_{vv} = 0$ with $\tilde{L}_{vvv} < 0$ we can find the maximum points of the gradient magnitude L_v in the gradient direction, and these points are defined to be edge points (Lecture 6). If we are interested in the larger structures of the image, one may increase the pre-smoothing slightly and also consider lowering the threshold for $\tilde{L}_{vvv}$.

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

Answers:
The function "extractedges" picks out the points in the smoothened intensity function L, where $\tilde{L}_{vv} = 0$ are max points. Thereafter it only keeps those edge points that have L_v values above a given threshold. In this way we can sort out the most representing edges in the image and in an efficient way suppress noise and "false positives".

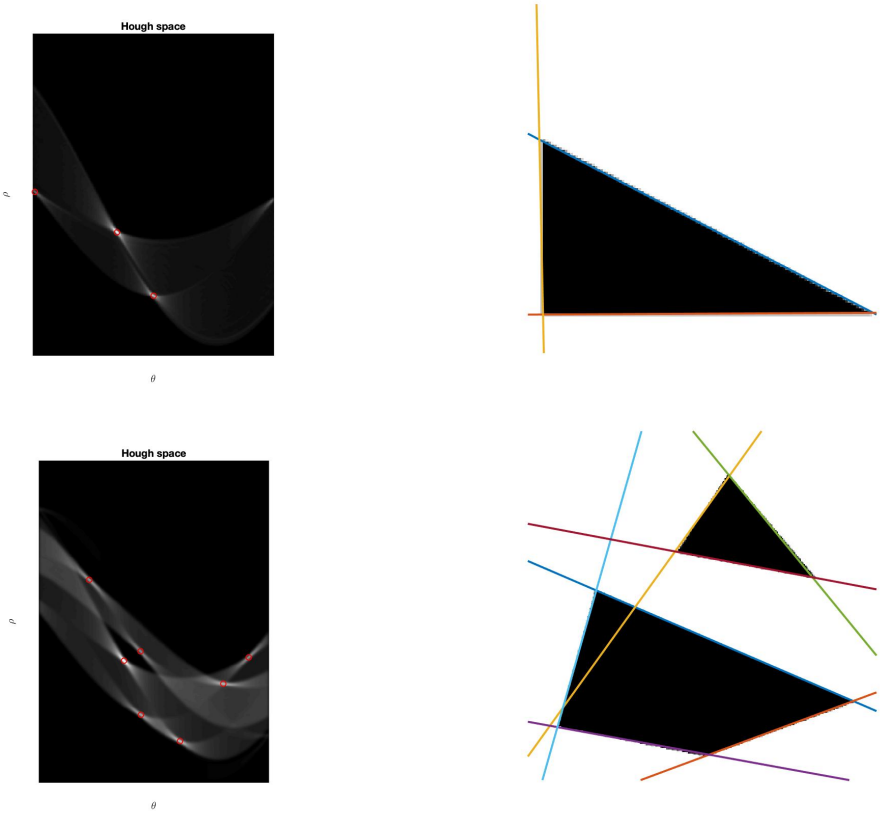


Question 8: Identify the correspondences between the strongest peaks in the accumulator and line segments in the output image. Doing so convince yourself that the implementation is correct. Summarize the results of in one or more figures.

Answers:
In Matlab the x-axis is pointing downwards and the y-axis is pointing to the right. The θ angle represents the orientation of a line and the angle is defined from the x-axis, having $\theta = \pi/2$ on the positive y-axis and $\theta = -\pi/2$ on the negative y-axis. Furthermore, ρ is the perpendicular distance from the origin to the line, and covers $\rho \in [-\sqrt{M^2 + N^2}, \sqrt{M^2 + N^2}]$ where M and N are the image resolution. Below are results of the function "houghedgeline" applied to the images "triangle128" and "houghtest256". The left images shows the Hough

space and the right one shows the resulting lines. In this case we used the following parameter settings,

Image	Scale	Threshold	nrho	ntheta	nlines	bin_smooth	K_bin
Triangle	20	20	400	300	3	0.1	10
Houghtest	1	40	700	500	7	0.5	18



For the "triangle128" case, the three lines corresponds to the following points in the Hough space,

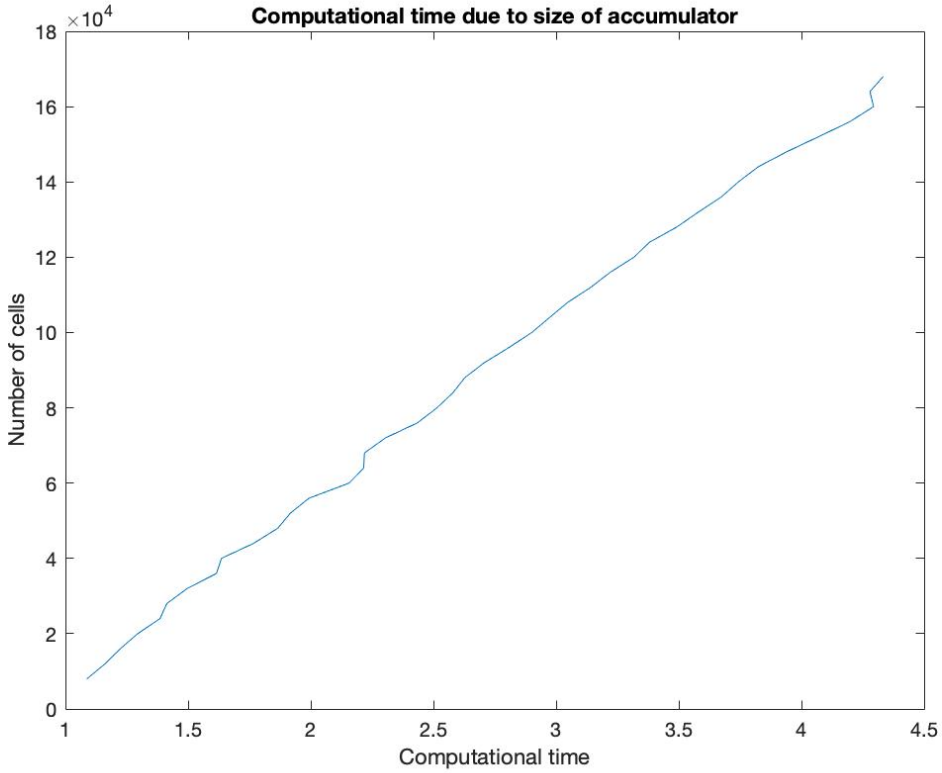
Line:	Blue	Red	Yellow
ρ	42.1925	113.8743	-4.0831
θ	-0.4781	0.0053	-1.5603

http://me.umn.edu/courses/me5286/vision/Notes/2015/ME5286-Lecture9.pdf

---

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

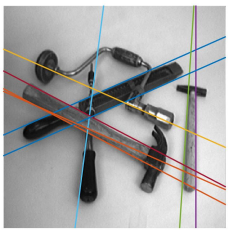
Answers:
Below is a plot of the computational time, in seconds, for the function "houghedgeline" used on the image "few256". In this case n_θ is constant over the iterations and set to $n_\theta = 80$. n_ρ on the other hand is $n_\rho = linspace(100,2100,41)$, thus we are evaluating the time for 41 different sizes of the accumulator. The size of the accumulator is simply $n_\rho \times n_\theta$, and will thus increase with a factor of 2 in each iteration. The results in the figure below are averaged over 10 runs.

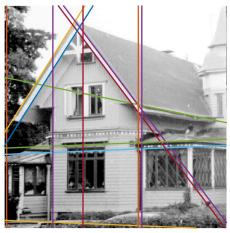


We see that the size of the accumulator as a function of time is approximately linear, which is a nice behavior. A greater number of cells doesn't necessarily imply better results. If θ and ρ are too coarsely quantized the resolution in line directions will be poor. But if they are quantized too fine, the accumulator won't be given enough samples. Therefore one must investigate what values of θ and ρ are suitable for each individual case. More detailed images usually needs a greater amount of cells to find good solutions, and in that case, the computational time increases, as can be seen from the figure above.

Furthermore, below are the results of applying "houghedgeline" to the images "few256", "phonecalc256" and "godthem256", with the following settings,

Image	**Scale**	**Threshold**	n_ρ	n_θ	n_{lines}
Few256	0.5	80	800	50	10
phonecalc256	10	30	1200	40	10
godthem256	3	60	900	40	18





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:
We let the accumulator increment be proportional to a function of the gradient magnitude, that looks like this,

$$\Delta S = (L_v)^{power}$$

Where "power" is chosen depending on the situation. It is convenient to increment the accumulator proportional to a function of the gradient magnitude. Because then we include more information when selecting lines. If we simply just increment by one, leftover noise and other details may get votes and concur against "real" edges. But taking the gradient magnitude into account, will get higher values in the accumulator and thus noise and false positives won't be competing at the same level anymore. But, if we use a "power" value that is too great, edges that has a greater change in magnitude will be much more voted for than "real" edges that has a slightly smaller change in magnitude. Therefore, in most cases, $power = 1$ seem to be a good choice.
