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

Name: Timo Haubner Program: DD2423

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 application of derivative filters to the original image reveals variations in pixel intensity. For deltax(), horizontal changes become visible: if the original image transitions from dark to bright, the derivative is positive (bright); if it transitions from bright to dark, the derivative is negative (dark); and if there is no change, the derivative is zero (grey). The same applies analogously to deltay() in the vertical direction.

The original size of (256, 256) is reduced to (254, 254) by the derivative operations. This is because no padding was implemented and the filter has a size of (3, 3). For pixels at the image borders, derivatives cannot be computed since the required neighbouring values are missing. The filters could also be implemented in a non-square form (3, 1) for deltay() and (1, 3) for deltax() which would result in image sizes of (254, 256) and (256, 254) respectively.

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

Answers:

Finding a threshold that produces ideally thin edges (ideally one pixel wide) is very difficult, if not impossible. Since edges vary in strength, it is impossible to determine a single global threshold that preserves every edge at exactly one pixel width. At the same time, textures and noise can also generate strong gradient values. As a result, some edges or edge segments will always disappear, while others remain too thick. However, by roughly testing several threshold values, it is still possible to achieve results that, although not perfect, are acceptable. This is demonstrated by applying different thresholds to the test images.

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

Answers:

Smoothing reduces noise in an image and makes edge detection more stable. In practice, edges are never ideal; they are always affected by noise. Due to this noise and the discretization of the image, edges are not continuous but appear as “jumps” in intensity values. Smoothing therefore improves the reliability of detecting real edges. However, for large smoothing scales, fine structures in the image can become blurred and may no longer be detected.

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

Answers:

Scale 0.0001:
A very small scale leads to a noisy directional derivative with many zero crossings. Since textures and noise are barely smoothed, Lvv detects local maxima in most parts of the image, not only at edges.

Scale 1.0:
The contour of the directional derivative shows fewer zero crossings. It becomes evident that noise and some small textures have been successfully removed by Gaussian filtering. However, the edges still cannot be clearly distinguished.

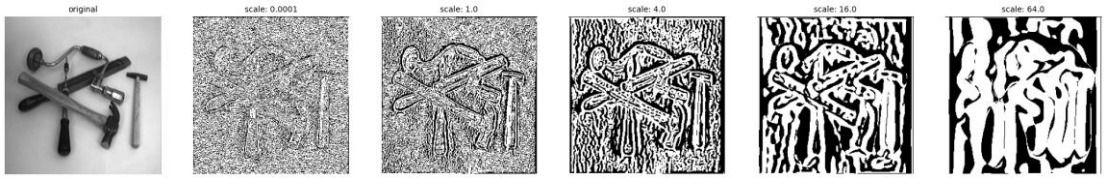
Scale 4.0:
This scale produces the most balanced result. Additional textures are smoothed out while all relevant edges remain clearly visible and undistorted.

Scale 16:
The high scale results in fewer zero crossings, and only coarse structures remain visible. Although textures are almost completely removed, some relevant edges are lost as well.

Scale 64.0:
This very large scale renders the results nearly unusable. Many relevant edges are no longer detected by Lvv, and the edges that are still present appear distorted and incorrectly rounded due to excessive smoothing.

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 third derivative is visually much more difficult to interpret than derivatives of lower order. From the code it is known that the white regions in the image correspond to areas with a negative third derivative. If the second derivative is zero at a point in these regions, this indicates a maximum of the first derivative, which is typically interpreted as the center of an edge.
It becomes clear that without smoothing (scale = 0.0001) the third derivative is highly unstable, while for large scale values (e.g., 64.0) it becomes very coarse and therefore only of limited use. This behaviour mirrors that of the second derivative across different scale levels.

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

Answers:

First, the image is smoothed using a specific scale. The importance of this step becomes evident in the preceding exercises.
Then, the condition Lvv = 0 is used to identify maxima of the directional derivative in the gradient direction for individual pixels. These points are typically interpreted as the centers of edges. Optionally, they are filtered using a threshold on the gradient magnitude to exclude weak edges.
Next, Lvvv is used to filter the detected zero-crossings of Lvv. Logically, only maxima of the directional derivative, not minima, should be interpreted as edges. Therefore, only points for which Lvvv < 0 holds are considered as valid edge points.

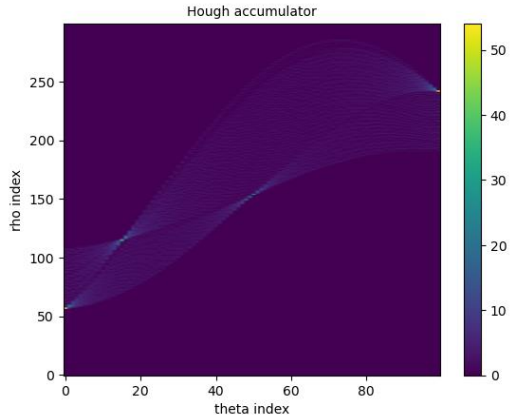
Question 7: Present your best results obtained with *extractedge* for *house* and *tools.*

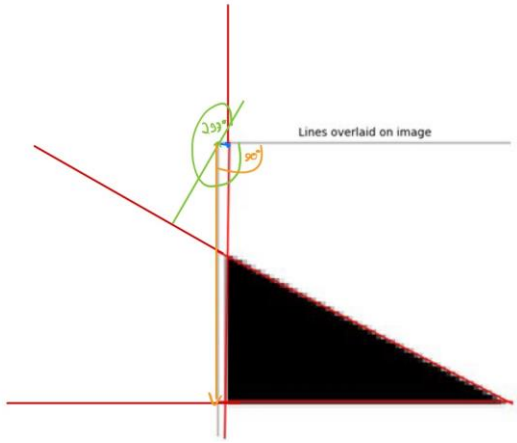
Answers:

Tools: Best result obtained for Scale = 4.0/Threshold = 8.0
House: Best result obtained for Scale = 4.0/ Threshold = 7.0

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:





Point indices	[57, 0]	[242, 99]	[115, 15]	[154, 50]
θ-value	$\pi/2 = 90°$	$-\pi/2 = -90°$	$-1.094 \approx -67°$	$2.72 \approx 1°$
ρ-value	56	-56	-21	3

Every point in Hough space corresponds to an infinitely long line in the spatial domain, but only the lines corresponding to the strongest peaks (local maxima) are used in the image. The figures above show the Hough accumulator, the detected maxima of this accumulator, and the resulting lines. The maximum at [154, 50] corresponds to a line with an almost horizontal normal vector and a small distance from the origin, which clearly represents the single vertical line in the image. The maximum at [115, 14] can also be clearly assigned to the triangle's hypotenuse. Note that its normal vector formally points upward to the right, but the distance given trough ρ is negative. It is noticeable that only three lines appear in the image, even though nlines was set to 4 and four maxima were indeed detected. This is because [57, 0] and [242, 99] represent the same line. In the first case, the normal vector is drawn at 90°, which is equivalent to a normal vector at –90° with a negated distance. From this, we can conclude that lines repeat periodically for increasing θ values. Therefore, the range for θ was restricted to $[-\pi/2, \pi/2]$. Since both interval boundaries are included in my implementation, the maximum for the boundary appears twice.

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

Answers:

nTheta:
A high value allows for a more accurate estimation of the parameter values, resulting in better alignment of the detected lines in the image. However, if nTheta is too large, the maxima in the Hough space become less distinct, which can lead to multiple detected lines for a single edge. Since every edge point must be evaluated for all θ values to determine the corresponding ρ, the computational cost scales linearly with nTheta.

nRho:
A high value again enables more precise parameter estimation, similar to nTheta. This results in more accurate positioning in the image. However, the peaks in the Hough space also become less distinct. In contrast to nTheta, increasing nRho does not inherently increase the runtime directly.

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 most obvious approach is to use a linear function. However, when applied to a sample image, it became clear that with a linear function, no meaningful relationship between the detected lines and the actual edges remains. The gradient influence is presumably too strong, causing points with high gradient values to be overweighted and thereby distorting the overall result.
A logarithmic function appears to be a reasonable solution to mitigate this overweighting. In fact, compared to the linear function, the results are improved, and some edges are correctly detected. Nevertheless, the outcome is still noticeably worse than when using the original uniform weighting of the edge points.
