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

Name: Rakin Ali Program: TCSM222, CINTE 19

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.

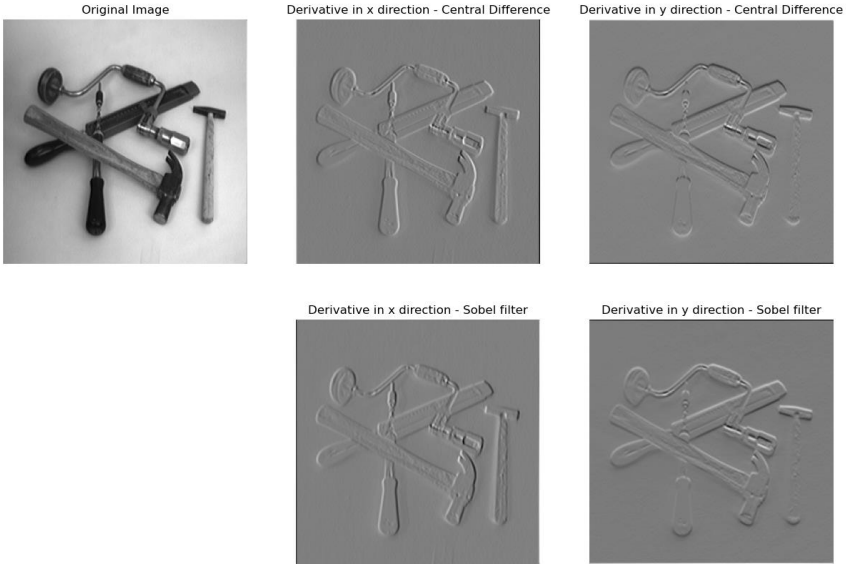
Good luck!

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

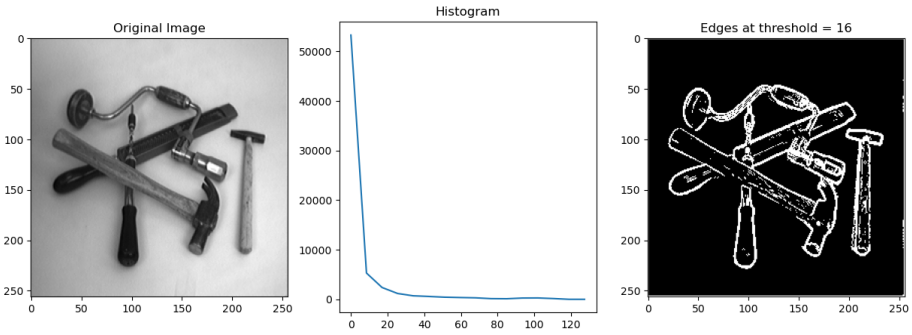
The image below shows the edges in the X and Y direction depending on the derivation direction of the Sobel filter and the central difference. The sobel seems to extract the edges slightly better than the central difference.

Regarding why images look smaller: You cannot slide the "kernel filter" past the edges of the image which leads to a size reduction. There are ways to avoid this using padding, zero padding etcetera however each have their pros and cons. Below is the image.

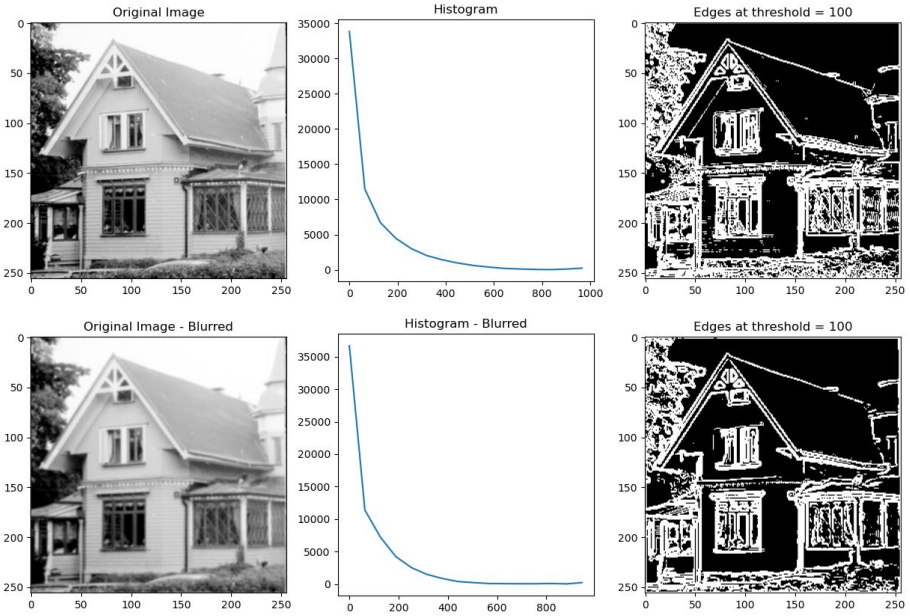


Question 2: Is it easy to find a threshold that results in thin edges? Explain why or why not!

Answers:
In my opinion, no it is not easy to find a threshold that results in thin edges. For the first image I used a 16 bin histogram to determine the best threshold. Eventually I picked a threshold of 16 and got the image on the right. See below



For the second one a Gaussian blur was applied with different variance, all of which were arbitrary numbers. Eventually I used the variance 0.5 and the threshold for the image was 100. See below



I'd argue that it is difficult to find a good threshold by only looking at the histogram. Picking a value that is near where "all values drop" which in this was near 20 in the first image and 200 on the other image is a good starting point.

Filtering helps reduce noise and eventually becomes filtered out by the threshold which is why blurring was useful here. You need to find a good balance between filtering out the noise and a good value to find the edges, there is a risk of filtering the entire image.

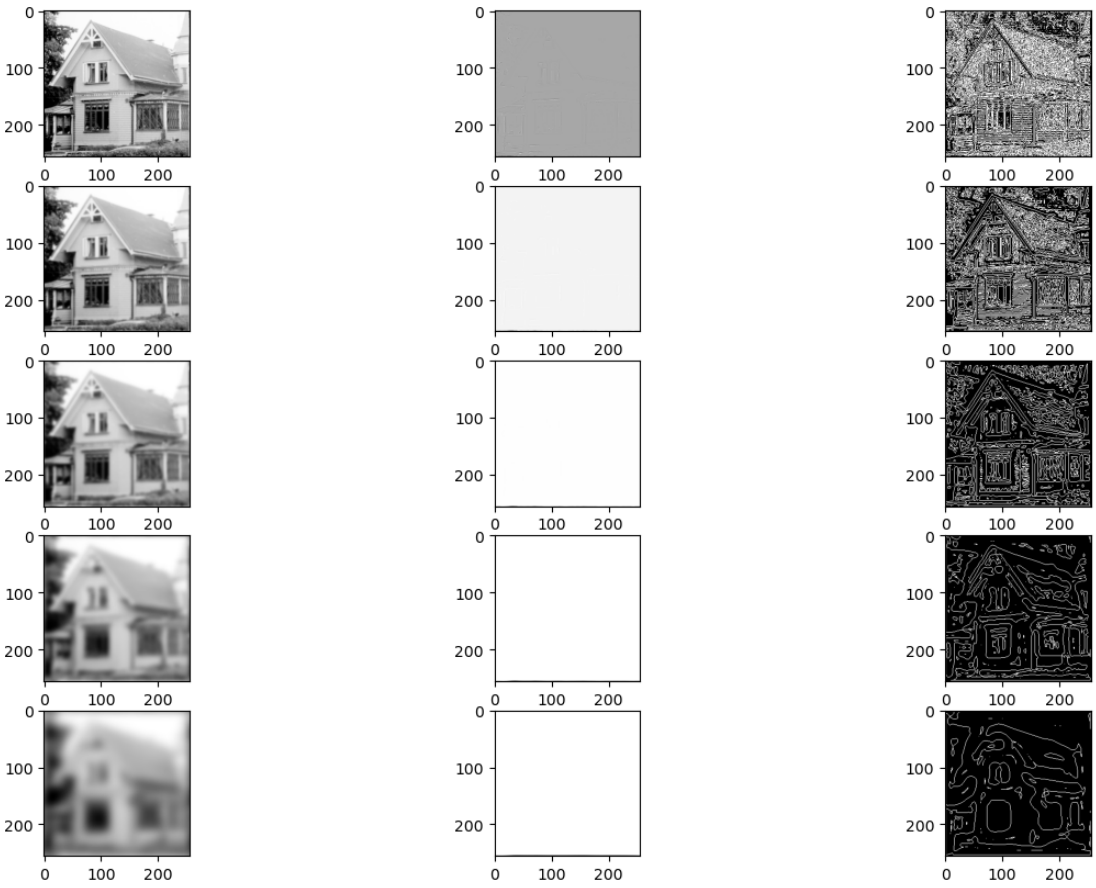
Question 3: Does smoothing the image help to find edges?

Answers:
Blurring the images made a difference. Blurring removes noise, this makes the output detect edges that are from 'real edges' and not from the noise.

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

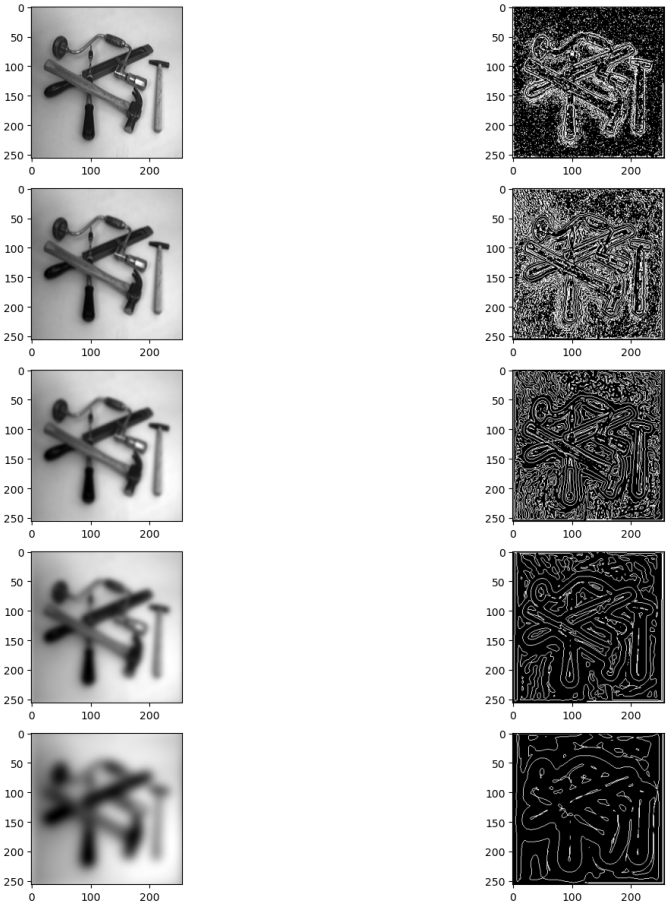
Answers:

The edges that become detected depends on the scale of the gaussian blur applied. Less blur means higher frequency edges are detected. More blur means less edges remain and thereby less stuff detected. As there are many zero-crossings in the image, the second derivates are shifting between negative and positives values. Smoothing the images causes less of the zero crossings. Smoothing too much however causes the edges to disappear and the images becomes too blurry. Check the graphs below. The first column represents the house blurred at different values, middle column shows LVV and the last one is the contour of 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:



Third derivative L_{vvv} allows only the point which are the local maxima to get passed. For lower scales, the details are visible however diminishes as the scale increases.

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

Answers:
There are three types of gradients that needs to be calculated

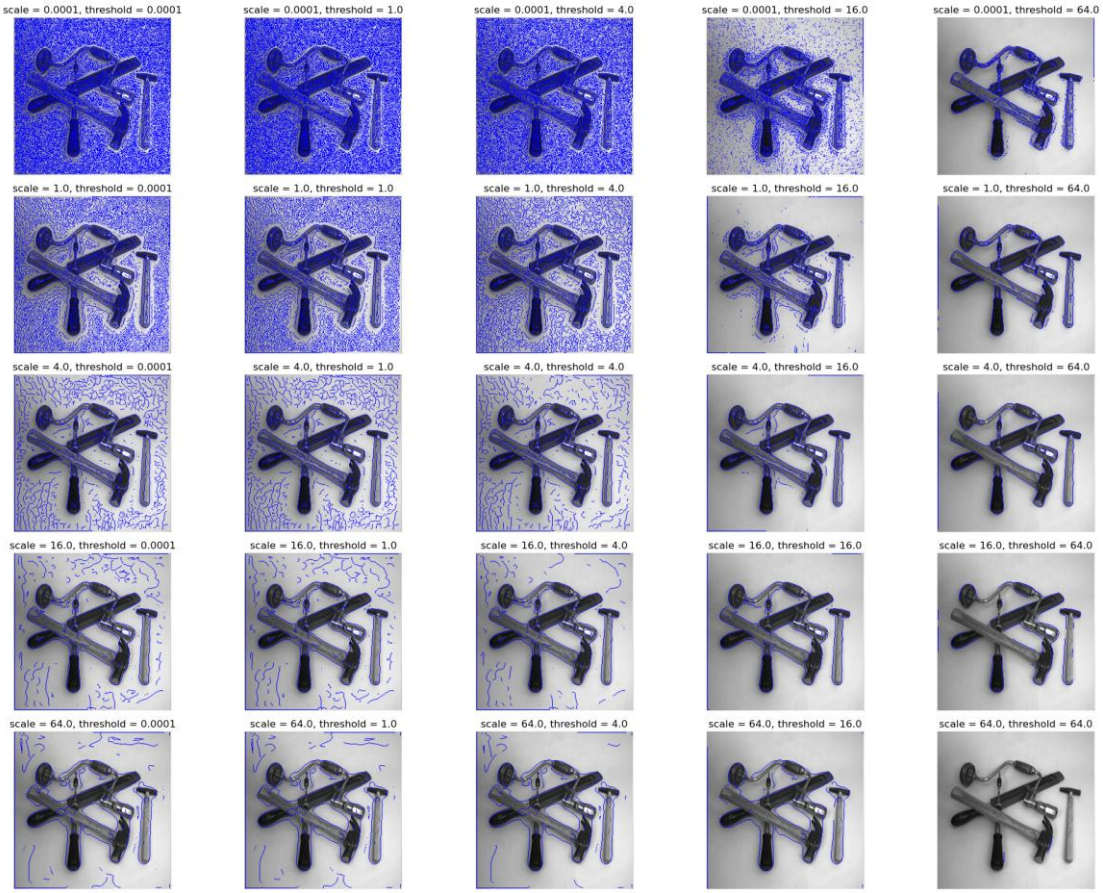
- **LV**: Represents the gradient of the image. A local maxima in LB represents an edge in the image. However to find the edges you need to detect maxima on the first derivate of the image. This is where the second derivative comes in

- **LVV**: Here you check if you found a local maxima or minima. Therefore you only check if this is equal to zero. However it's not sufficient to know that a point is maxima or minima. You want to be certain that this point is a maxima. This is where the third derivative comes in

- **LVVV**: Here you can determine if the point is a maxima or minima. You do this by checking the sign of the third derivative and checking if it is positive or negative. If negative then it's a maxima and positive then it's a minima. **If all three conditions are met then you have found a maxima point = Edge**

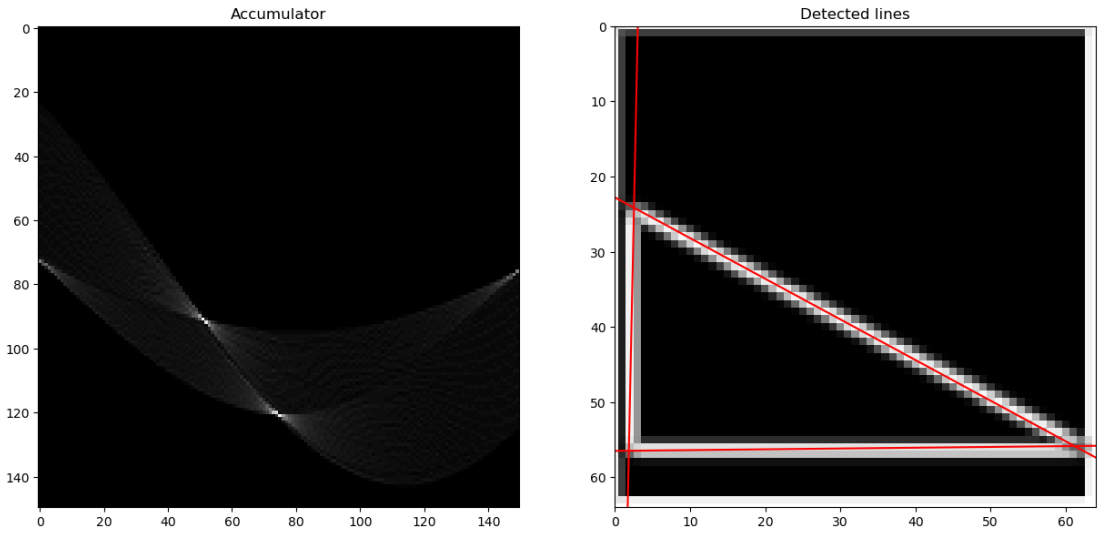
Question 7: Present your best results obtained with *extractedge* for *house* and *tools.*

Answers:



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:
Here is a picture of my implementation. The lines are quite good so I am conviced that my implementation is correct. The strongest peaks are the “light” dots in the accumulator where a lot of lines meets. In this image there were 3 dots.



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

Answers:
The size of each cell in the grid represents the resolution of the accumulator. A higher number of smaller cells allows for more precise detection of lines but requires a lot of computational recourses. If we increase the cell size, that is have a less larger grid, then we will have a less accurate direction of lines/edges however increase the computational power.

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:
There are a few ways to improve Hough transofmrs

1. Increment with gradient magnitude or some other weighting. This reduces critical dependency on thresholds however rarely improves

2. We use information about gradient direction from edge detection. By this we increment the accumulation by only for those r- values that seem reasonable. This according to the theory is more effective.

3. Lastly we increment the accumulator not only point wise but also with some windowing function. This is equivalent to smoothing after voting is done.
