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

Name: ___Alex Gunnarsson_____ Program: ___TCSCM_____

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!

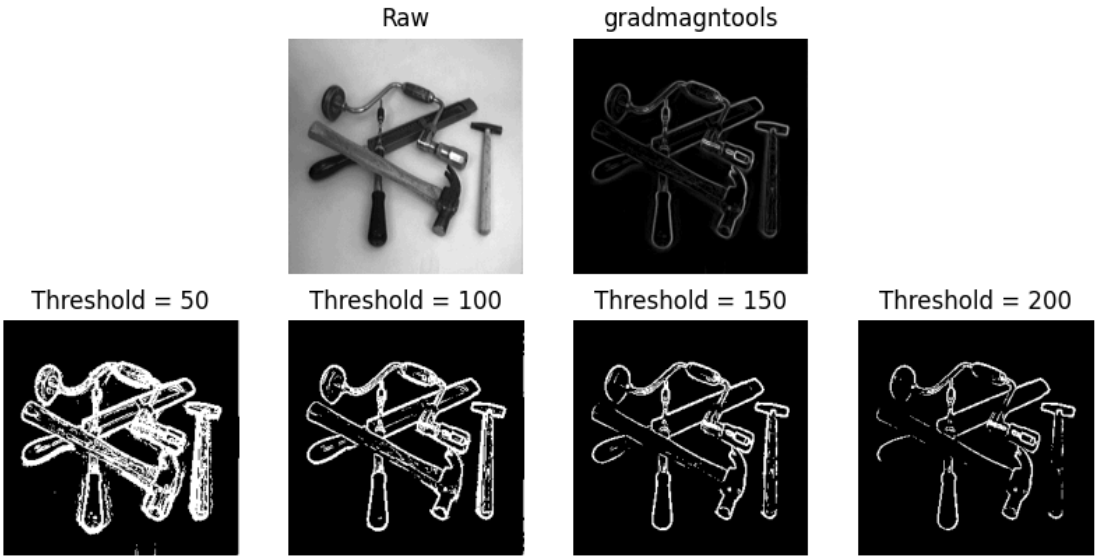
Question 1: What do you expect the results to look like and why? Compare the size of *dxtools* with the size of *tools*. Why are these sizes different?

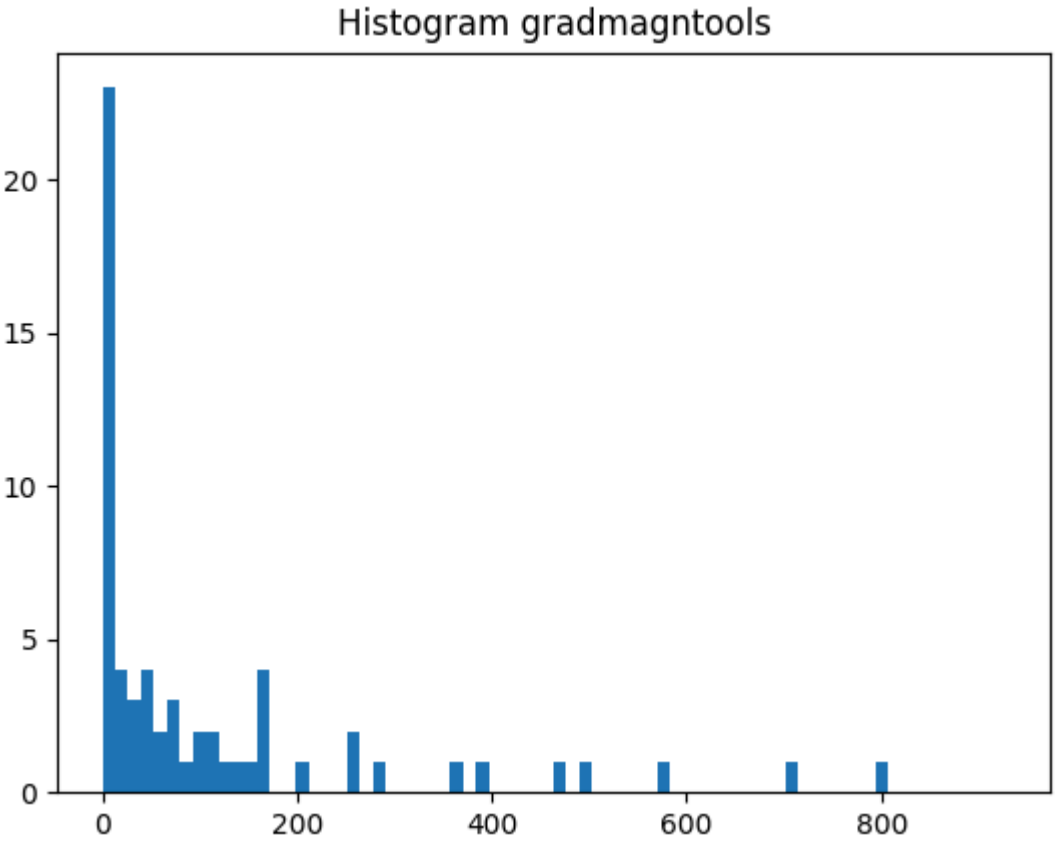
Answers: Since the Sobel operator approximates the first order derivative in the x and y directions, we get highlights of where the image changes rapidly in either orthogonal direction.

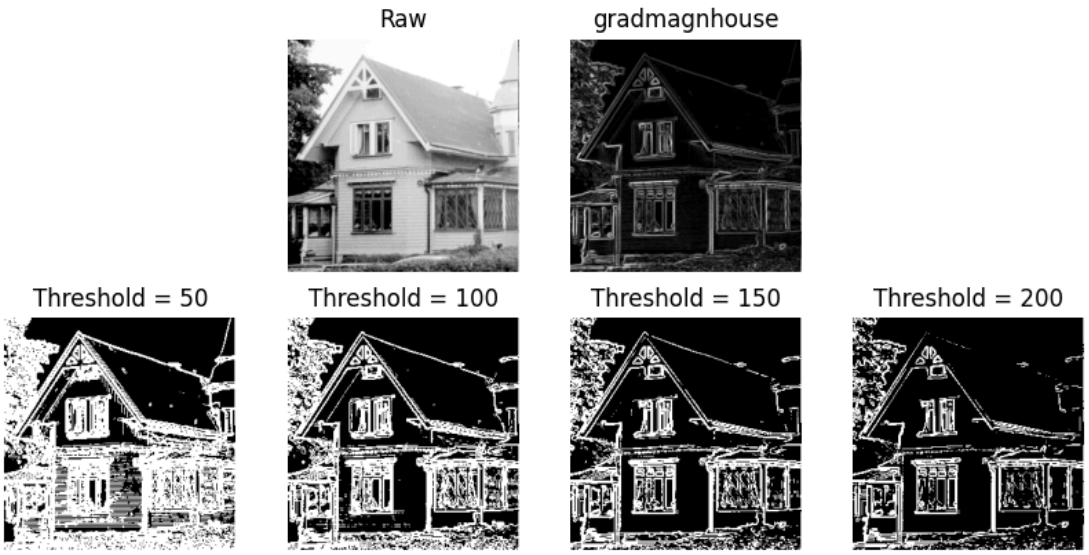
The size of the original image is 256x256 while both dxtools and dytools are 254x254. This is because we cannot calculate the values along the edges so we lose 1 pixel on all sides since the matrix is 3x3. The "valid" parameter ensures this and does not add padding to help in calculating the borders.

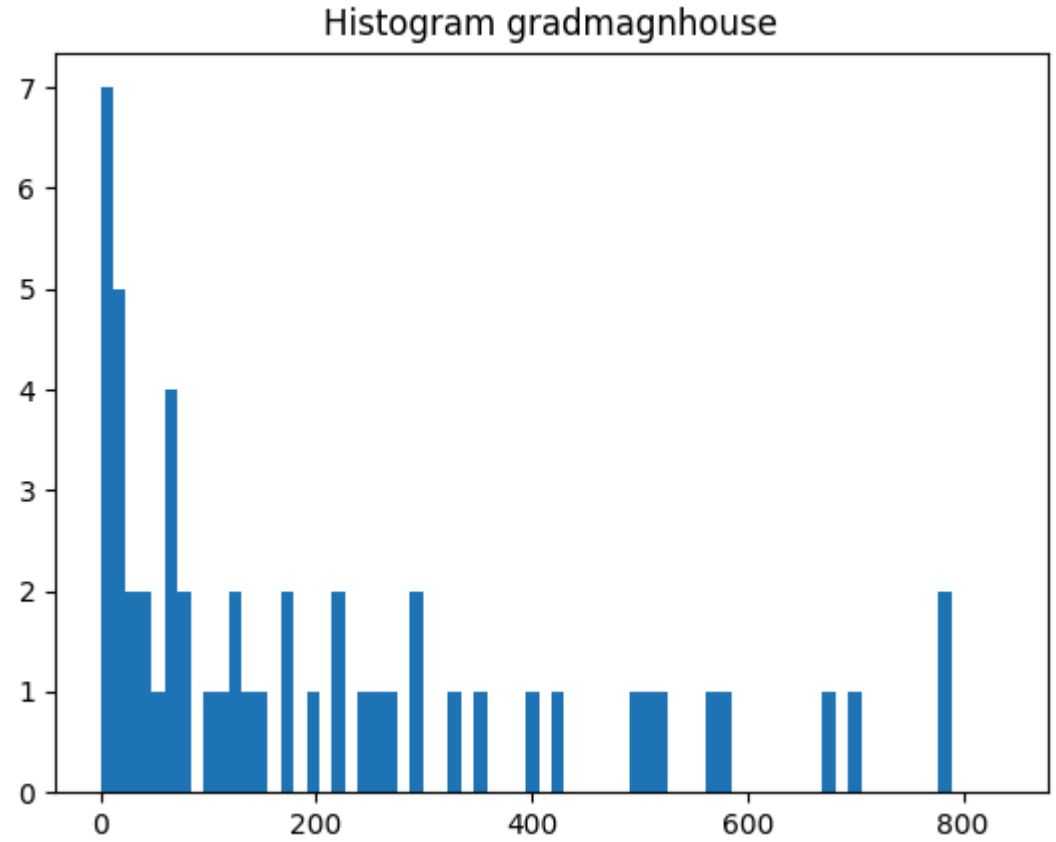
Question 2: Is it easy to find a threshold that results in thin edges? Explain why or why not!

Answers: Not really, there is a bit of noise in both images that is either detected as an edge, or the real edge is not fully detected. This is also true for the higher complexity image.


Raw
gradmagntools
Threshold = 50
Threshold = 100
Threshold = 150
Threshold = 200

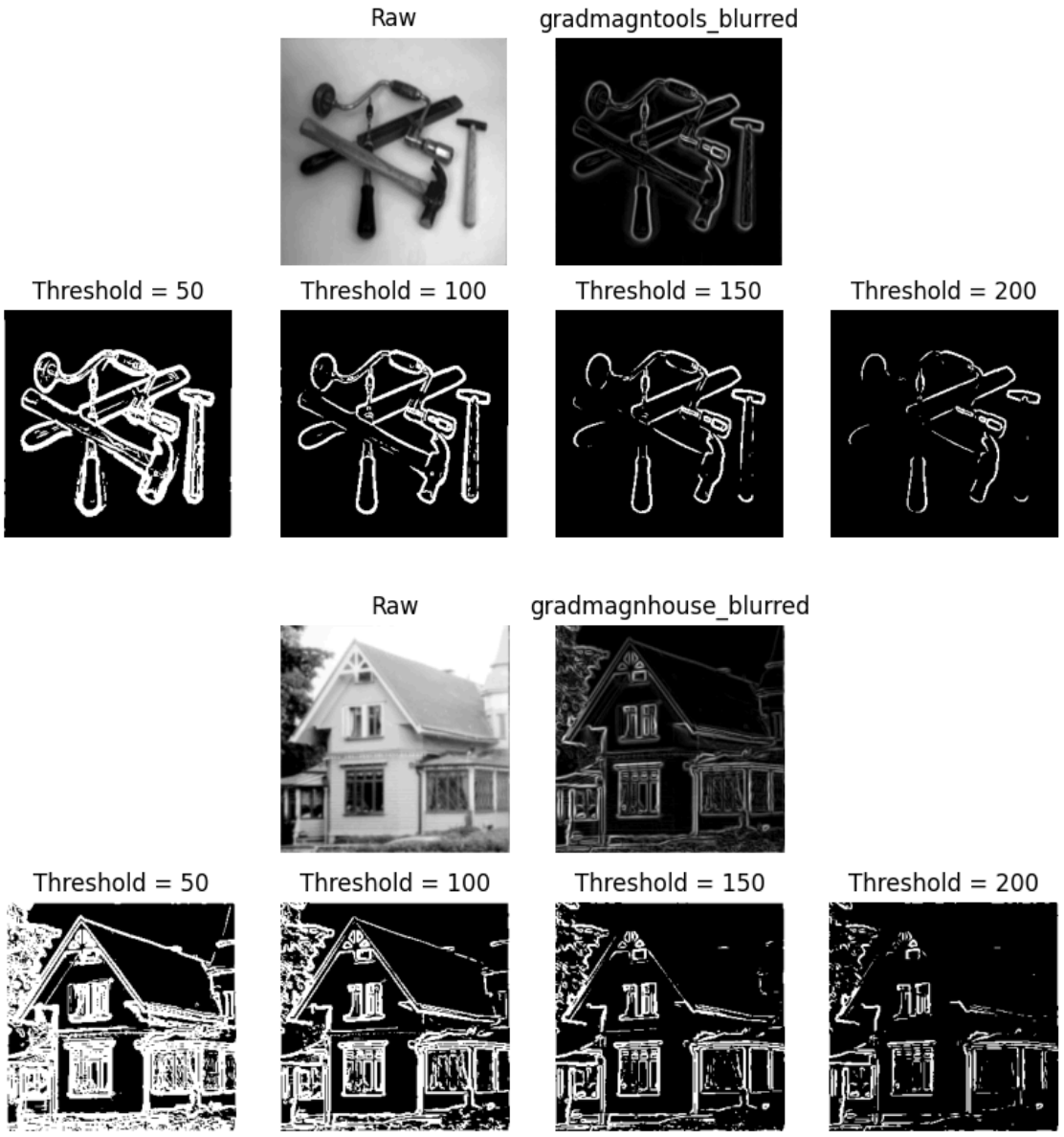

Histogram gradmagntools
20
15
10
5
0
0
200
400
600
800





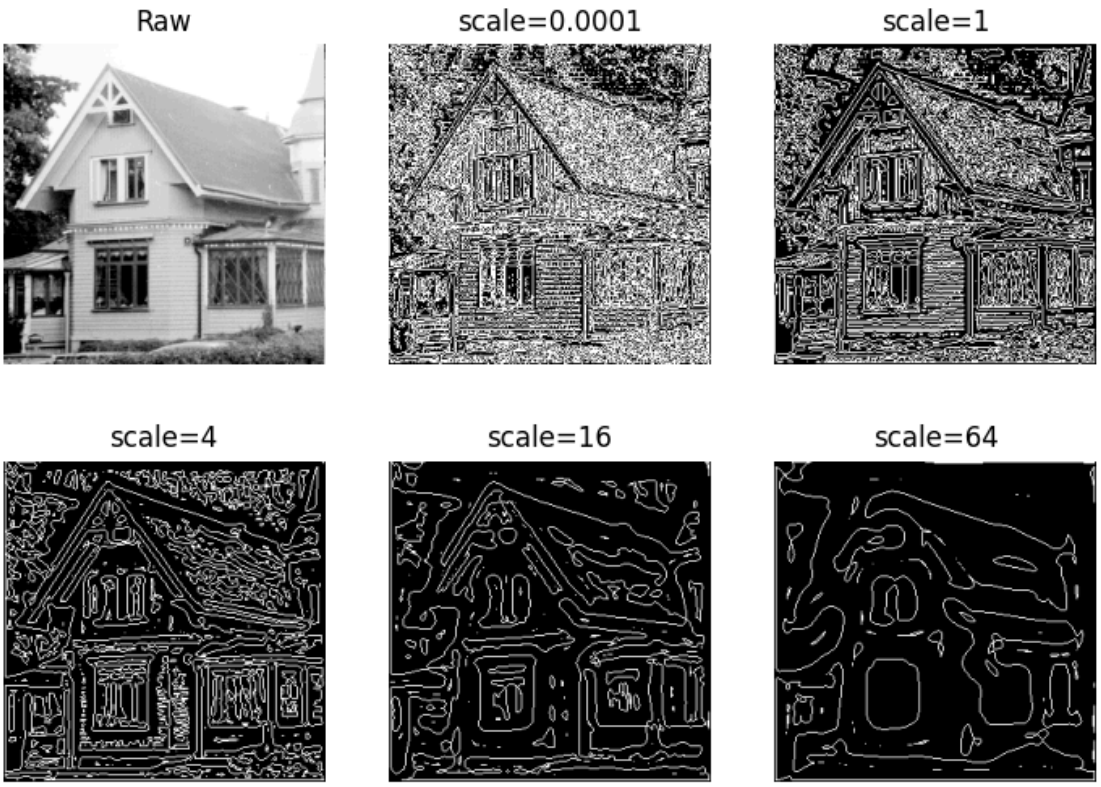
Question 3: Does smoothing the image help to find edges?

Answers: Perhaps slightly, although it mostly seems to shift the optimal threshold value, while the optimal threshold image remains pretty much the same. The images below were blurred with a Gaussian kernel with sigma=1.



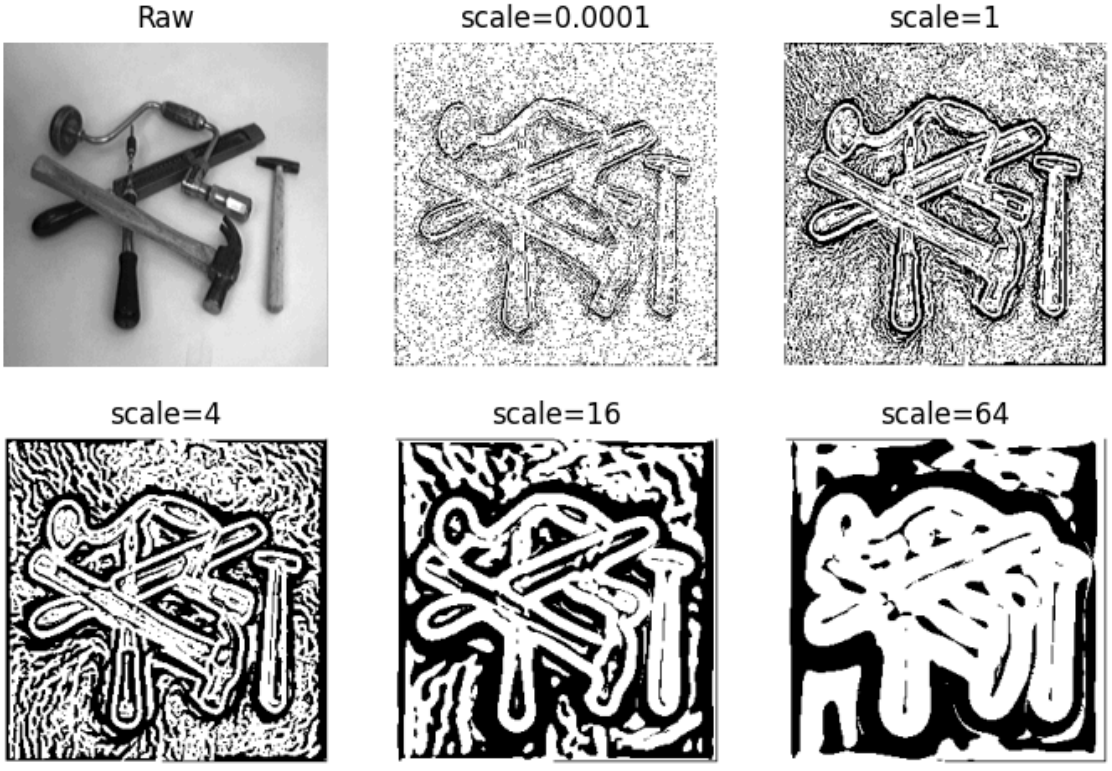
Question 4: What can you observe? Provide explanation based on the generated images.

Answers: The goal is to find where an edge is by identifying where there are drastic changes in the image, L. Quite naturally, this should be the one point where the image is changing the most, so it should be where Lv (the first derivative) is the largest, which is given by Lvv=0, and to make sure that Lv is the largest, not the smallest, which is given by Lvvv<0.



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

Answers: We can see that we get where more general regions of where relevant edges may be.



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

Answers: By combining the two, such that an edge is where Lvv=0 and Lvvv<0.

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

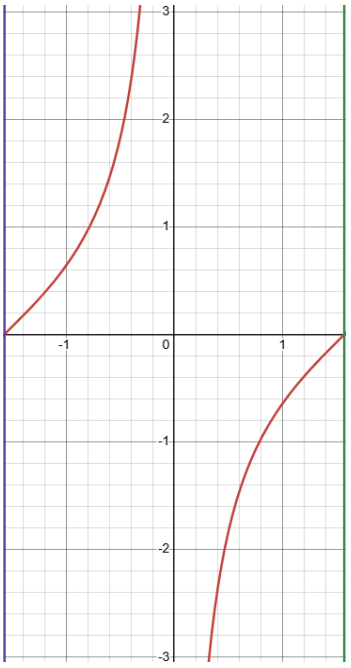
Answers:

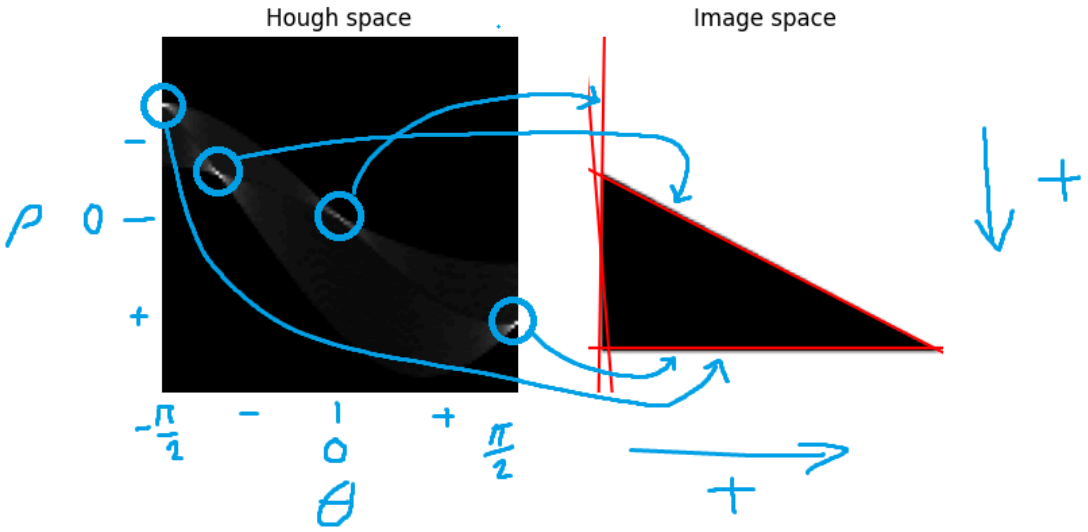
scale=5, threshold=35

scale=8, threshold=35

Question 8: Identify the correspondences between the strongest peaks in the accu-mulator 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: The parametrization yields $y = \frac{\rho - x \cdot cos(\theta)}{sin(\theta)} = \frac{\rho}{sin(\theta)} - x \cdot \frac{cos(\theta)}{sin(\theta)}$ so the line will have intersection at x = 0 and $y = \frac{\rho}{sin(\theta)}$ with a slope of $-\frac{cos(\theta)}{sin(\theta)}$, which will look like:



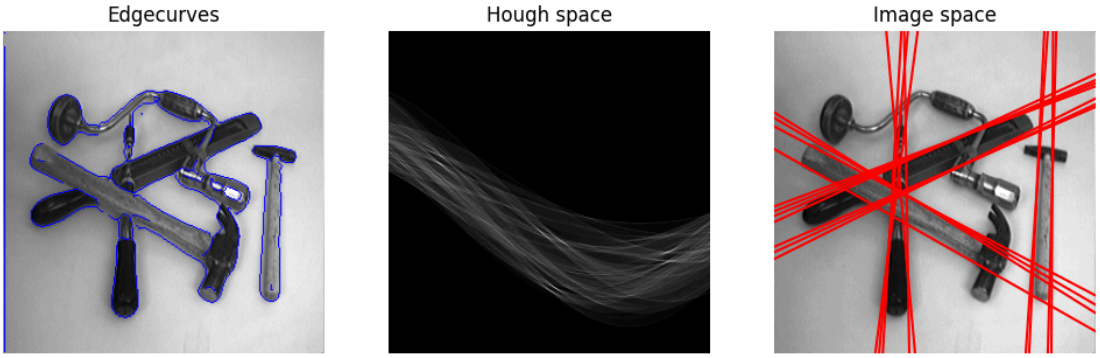


Question 9: How do the results and computational time depend on the number of cells in the accumulator?

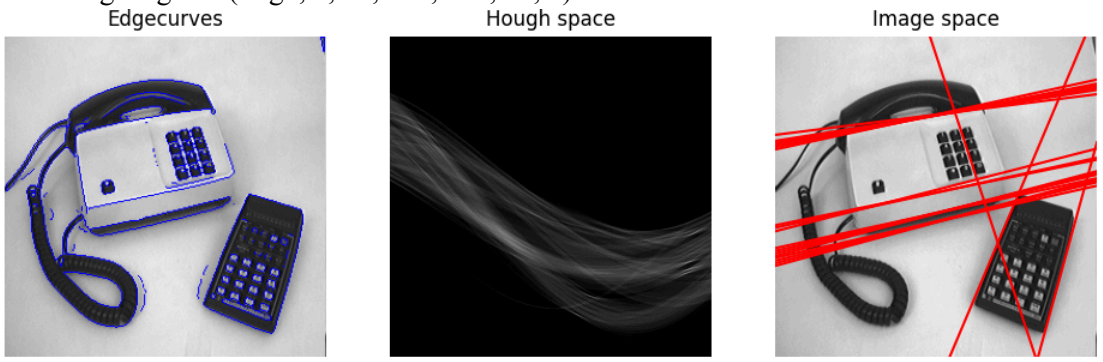
Answers: The computational time increases, but is more dependent on the number of theta values rather than the number of rho values, since thetas are iterated through, while rhos are just searched for and found.

Below are the results from the given images.

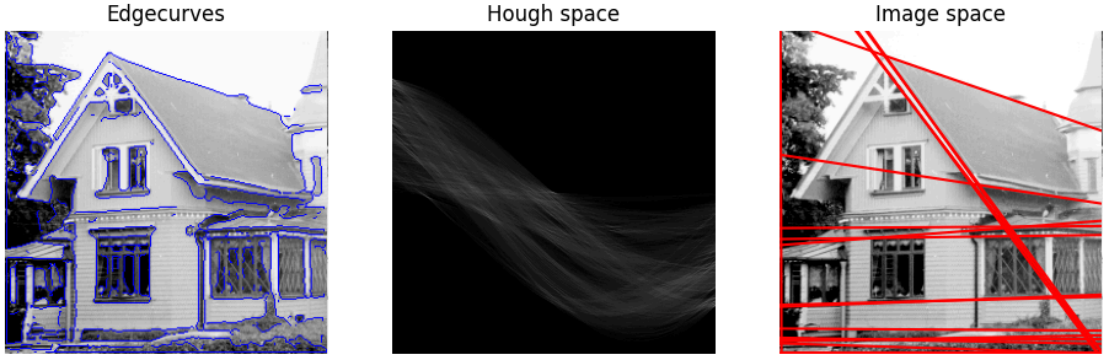
With houghedgeline(img1, 8, 35, 300, 300, 20, 2):



With houghedgeline(img2, 2, 45, 500, 500, 20, 2):

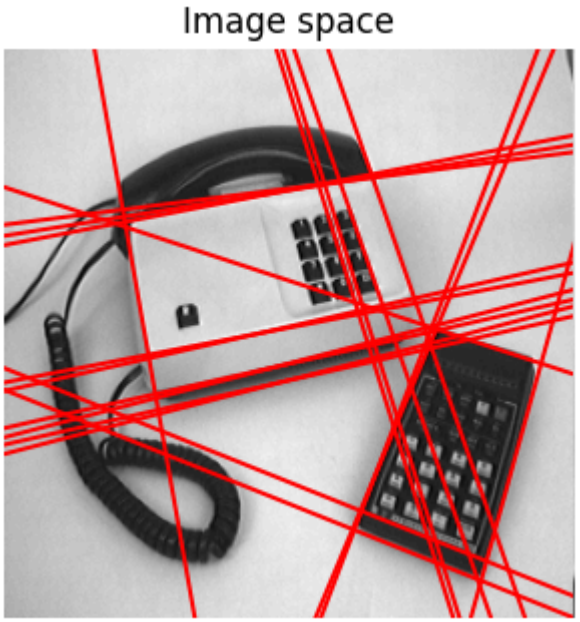


With houghedgeline(img3, 5, 35, 500, 500, 20, 2):



And some variations:

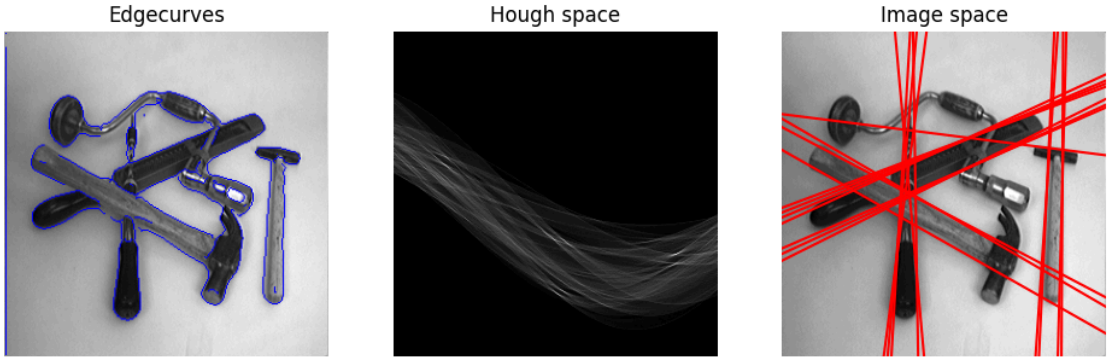
houghedgeline(img2, 2, 30, 800, 100, 20, 2):

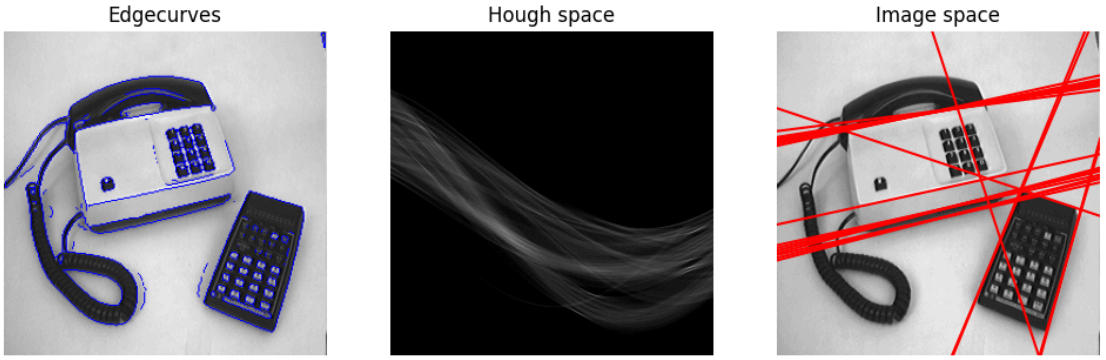


houghedgeline(img3, 1, 70, 1000, 200, 40, 2):

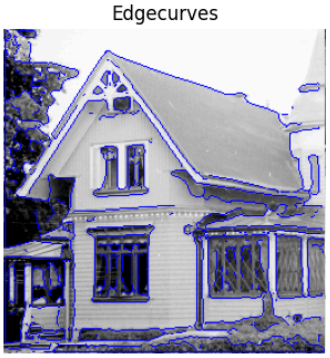
Question 10: How do you propose to do this? Try out a function that you would suggest and see if it improves the results. Does it?

Answers: We can weight them according to the magnitude of the gradient by prioritising larger magnitudes and counting their vote as more valuable. Magnitudes closer to the threshold should be counted less, while magnitudes farther away from the threshold should be counted equally much. Therefore, the logarithm of the difference between the magnitude and the threshold makes sense, but as we see below, it didn't make much of an actual difference. The only visible difference is that it found an extra edge on the top of the hammer in "tools".





Edgecurves



Hough space

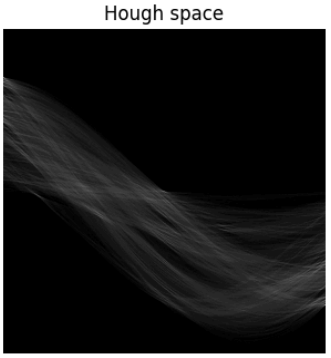


Image space

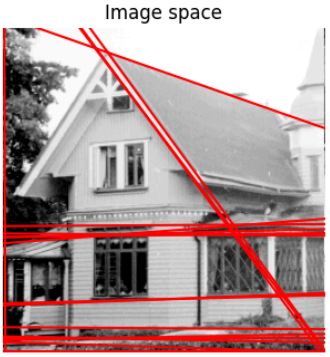


Image space

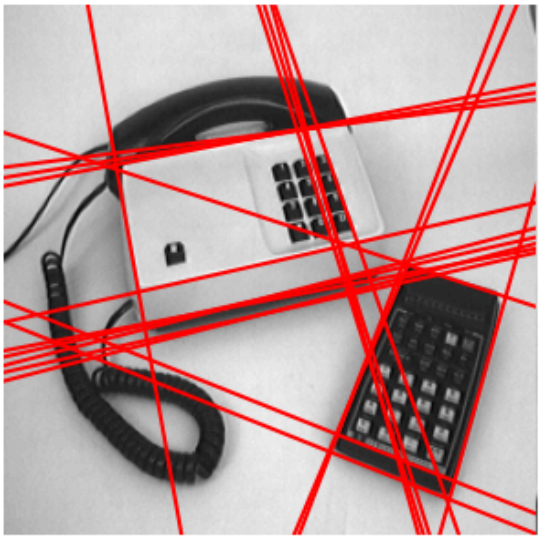


Image space