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

Name: _____Chia-Hsuan Chou_____ Program: _____ SCR 2018_____

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?

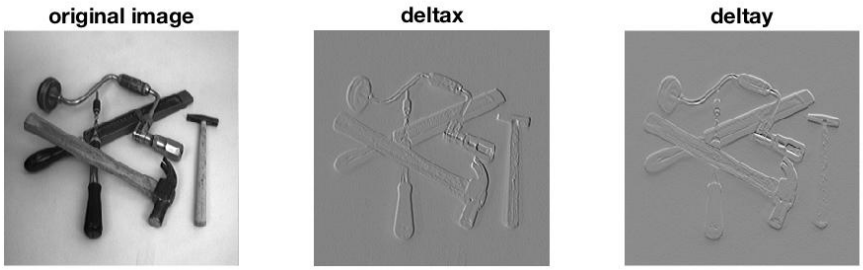
Answers:

My expectation on the results are that there are only edges showed in the figure. Because our image few256 has a larger contrast between black and white, therefore, the edge might be clear. On the other hand, when we apply simple difference and central difference, I expect to have a much clearer edge on the direction I differentiate.

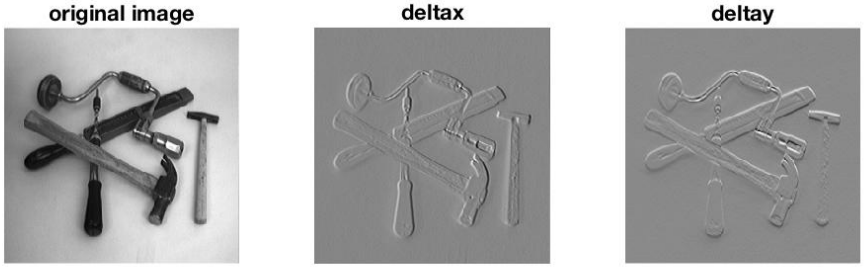
I use a 3x3 kernel to implement the differentiation and I found that my size of *dxtools* and *dytools* is less than my initial figure because of the function parameter. we use 'valid' as the shape parameter, which only returns those parts of the convolution that are computed without the zero-padded edges. Hence we don't calculate the corner because there is no next / previous element among them. In our case the size changes from (256,256) to (254,254).

Here are the results with different operators:

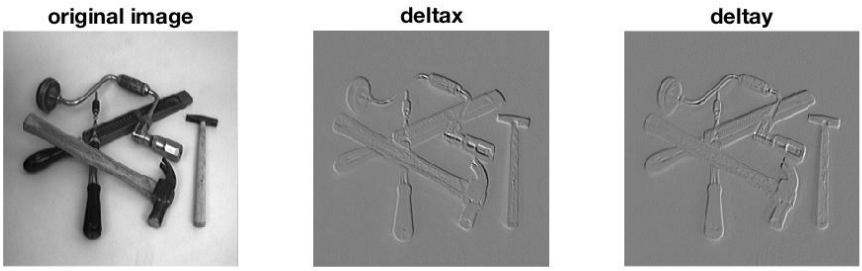
1. Simpler different operator



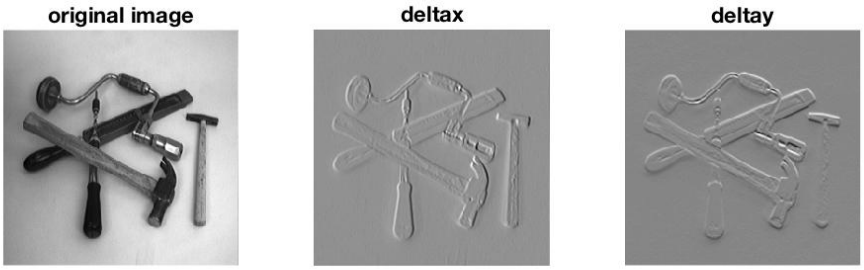
2. Central different operator



3. Robert's diagonal operator



4. Sobel operator



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

Answers:

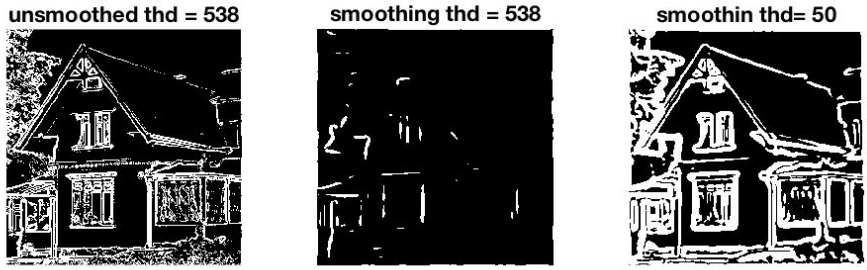
It is not easy to find a suitable threshold. This is because if the threshold is too low, there will be many maxima due to noise; if the threshold is too high, the weak edge might be disconnected as the figure here shows. So we have to find a threshold that can greatly filter out the noise while remain the weak edge.

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

Answers:

Smoothing the image does help to find edges because it removes the local maxima and we don't need such high threshold to find the edge. The figure below shows the comparison between smoothed and unsmoothed image.

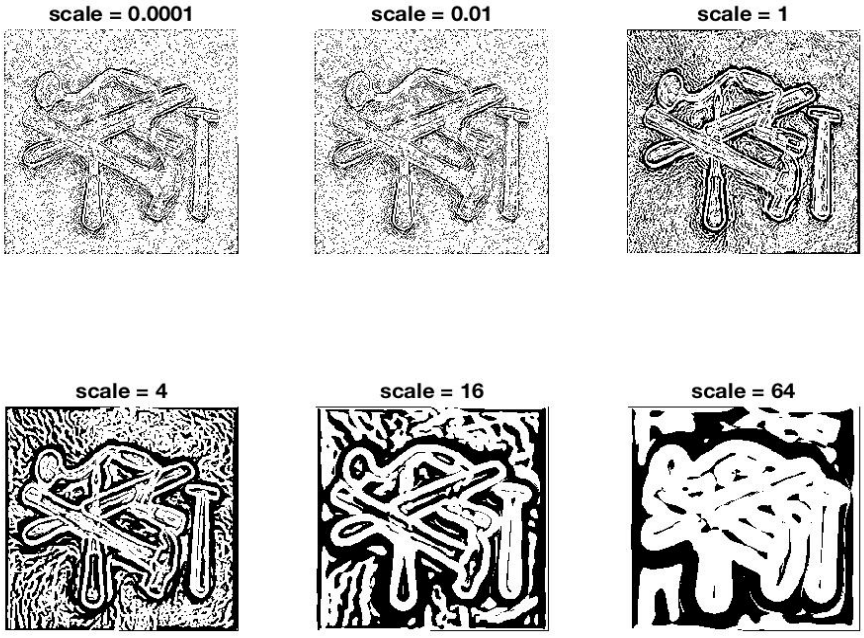
The left figure below is the unsmoothed image with the threshold equals to 538, and the image in the middle is the image smoothed by Gaussian filter with sigma equals 4 and the threshold is 538 as well. By comparing the left and middle image, we found that we don't need a threshold as high as we used in the unsmoothed image. The threshold = 50 is used in the right image and there is obviously more clear edge in right image than in middle image. Besides, the edge in the smoothed image is much thicker than the edge in the unsmoothed image. However, it is much difficult to find the threshold after smoothing the image.



Question 4: What can you observe? Provide explanation based on the generated images. What is the effect of the sign condition in this differential expression?

Answers:

The white lines in the figure below is the third derivative of the image few256. We can see that the edges are no longer clear. If we choose larger scale, we can recognize edges more clearly. However, if we choose too large scale, the white lines are much thicker, and some of the pixels near the edge is considered as a part of edge and we the edge is no longer accurate.

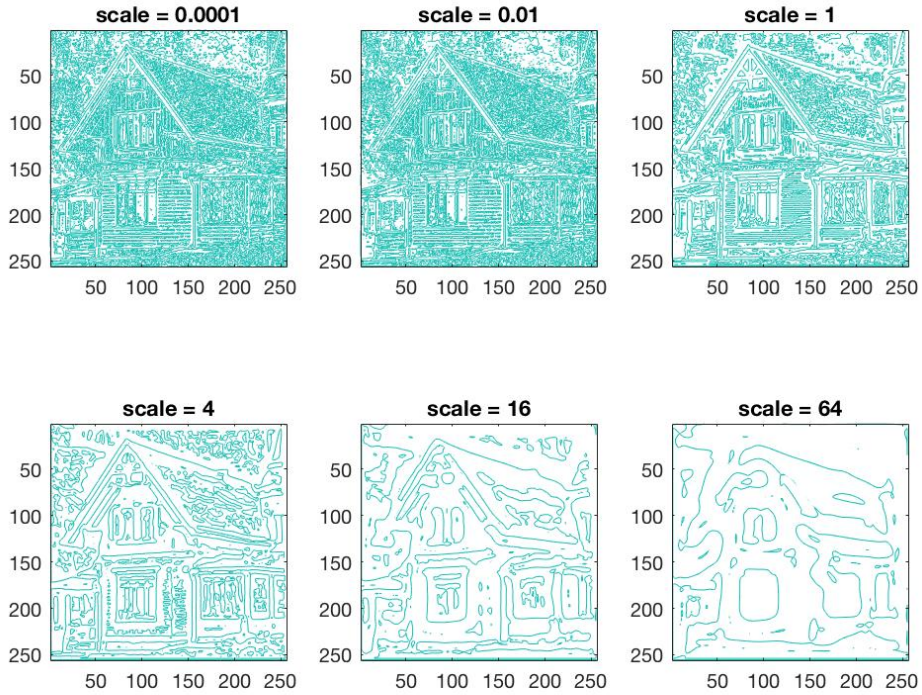


Question 5: Assemble the results of the experiment above into an illustrative collage with the *subplot* command. Which are your observations and conclusions?

Answers:

In the figure below, the green lines are the pixels whose second derivative is 0. we can find that when the scale is small in the Gaussian filter, it is hard to see any edge because there are still many local maxima left. The image will contain more details such as texture, noise and so on, so we can hardly find a clear edge. If we choose a larger value of scale, we

can see some edge emerge, and can recognize the edge. If we choose too larger value of scale, we can see that some edge is faded and distorted that is not exactly what we want.



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

Answers:

If we would like to use the response from *Lvv* to detect edges, we should first smooth the image with a proper scale. Then we take only the pixels with their second derivatives is 0 as edge.

If we want to improve the result by using *Lvvv*, we can pick the pixels which not only their second derivatives are 0 but also their third derivatives are smaller than 0 to remove more isolated points (noise) which we don't consider them as edges.

The edge can be defined as below:

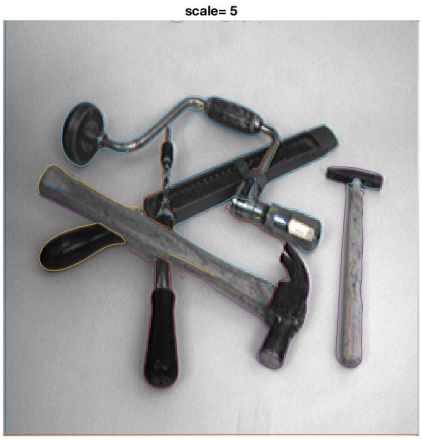
$$Lvv = 0$$

$$Lvvv < 0$$

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

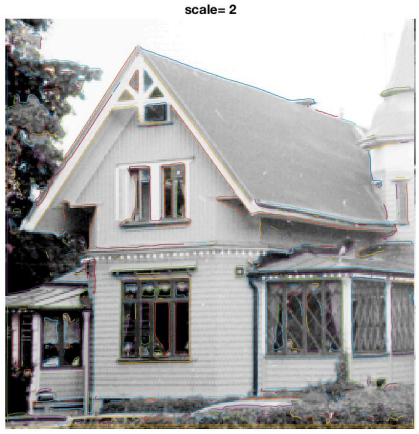
Answers:

1. The tools:



Scale = 5 & Threshold = 20

2. The house:



scale = 2 & threshold = 100

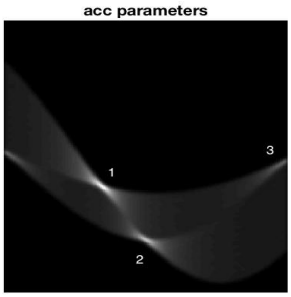
Question 8: Identify the correspondences between the strongest peaks in the accumulator and line segments in the output image. Doing so convince yourself that the implementation is correct. Summarize the results of in one or more figures.

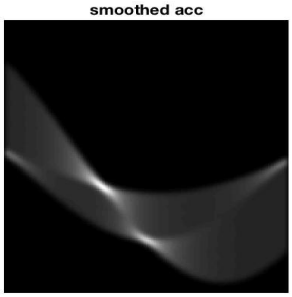
Answers:

In this question, we are going to use the figures ***triangle128, houghtest256*** to identify the correspondences between the strongest peaks in the accumulator and line segments in the output image and then use ***few256*** and ***phonecalc256*** to test my code.

1. ***triangle128***

The Accumulator pramameters with the ranking :

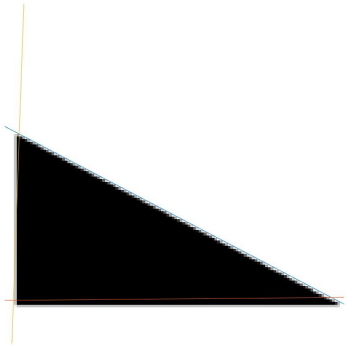




The ranking of (rho , theta):

Ranking	rho	theta
1	40.9340	-0.4826
2	111.8863	0.0088
3	8.1868	1.5357

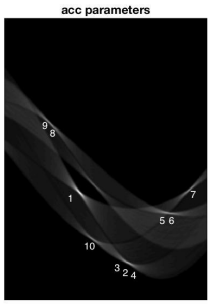
The result image:



Scale =2 & threshold = 5

2. ***houghtest256***

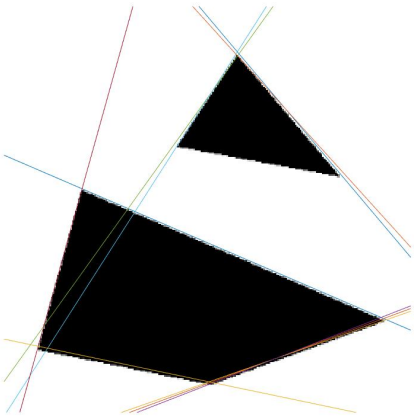
The Accumulator pramameters with the ranking :



Ranking	rho	theta
1	86.3379	-0.4069

2	267.72	0.3544
3	266.2689	0.3369
4	269.1710	0.3719
5	138.5759	0.9495
6	140.0270	1.0020
7	79.0826	1.299
8	-93.5932	-0.8707
9	-87.7889	-0.8357
10	205.3245	-0.2056

Since the lines of ranking 2,3,4 locate really close in the accumulator, we assume that there will be three lines really close in the result figures. Ranking 5,6 and Ranking 8,9 are really close to each other as well, so we assume that there will be two set of two lines really close in the result figures as well.

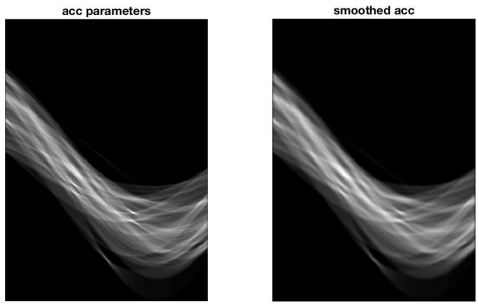


scale = 16 & threshold = 250

The figures above show that our assumption and implementation are correct.

3. ***few256***

The Accumulator pramameters:

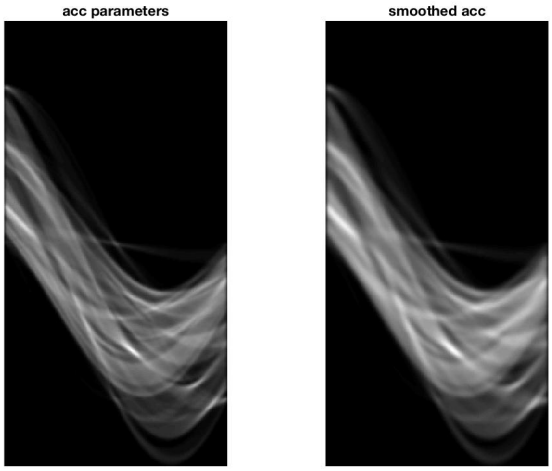


and the output image:

scale = 5 & threshold = 20

4. ***phonecalc256***

The Accumulator pramameters:



And the output image:

scale = 16 & threshold = 15

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

Answers:

If we have more cells in the accumulator, the computational time will increase because we have to iterate all the theta when looking for the lines. On the other hand, if we have more cells in the accumulator, the results will be more accurate because we can find the rho and theta in the accumulator which are closer to the actual rho and theta.

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:

In this question, instead of setting s=1, I also tried the other functions which are:

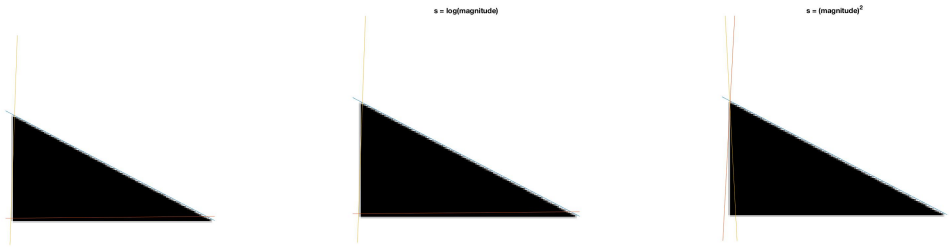
1. s = 1
2. s = log(magnitude)
3. s = magnitude^2

In first function s = 1, it treats all the magnitude, including the local maxima as equal. On the second function, it will smooth out the importance of large magnitude, which indirectly smooth out the noise. Therefore, the lines in which s = log(magnitude) will be closer to the actual edge. On the other hand, the third function s = magnitude^2 enlarge the importance of the magnitude, hence some lines might locate close not to the edge but to the local maxima. For example, in the image ***few256*** and ***phonecalc256***, when s = magnitude^2, unlike when s = log(magnitude) or s = 1, lines are really close to each other because there is a large magnitude on that line where the accumulator increment is large.

Among these three functions, the second function indeed improve the results. It removes the lines which are caused by local maxima instead of the edge.

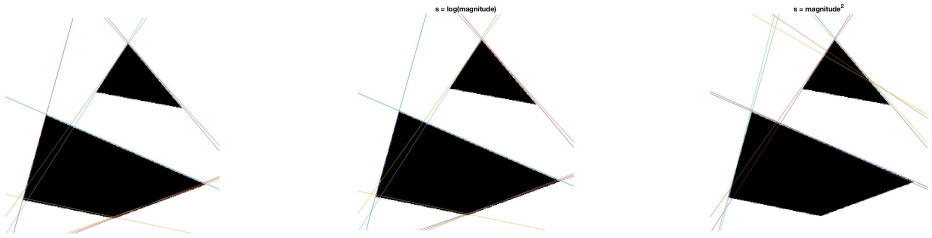
To compare these three functions, the images in Question 8 are used:

1. ***triangle128***



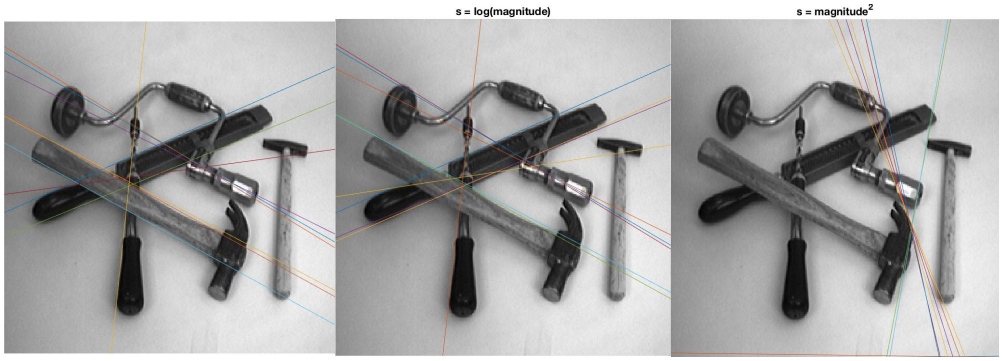
s =1 s = log(magnitude) s = (magnitude)^2

2. ***houghtest256***



s =1 s = log(magnitude) s = (magnitude)^2

3. ***few256***



s =1 s = log(magnitude) s = (magnitude)^2

4. ***phonecalc256***

s =1 s = log(magnitude) s = (magnitude)^2