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

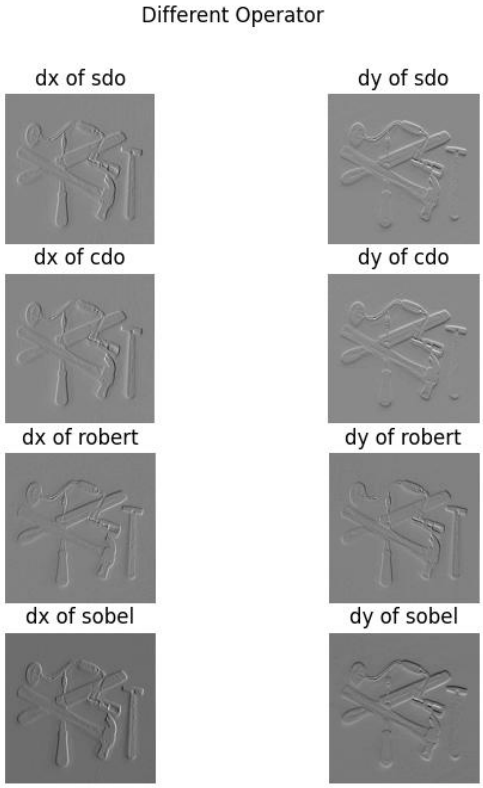
Name: Hanqi Yang Program: TINNM

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 edge in the x-direction will be clear if the image convolves with dxtools, and the edge in the y-direction will be clear if convolve with dytools.
The dxtools computes the gradient in x-direction and emphasizes the corresponding edges. Similar for dytools.
The results of different operator are shown below.



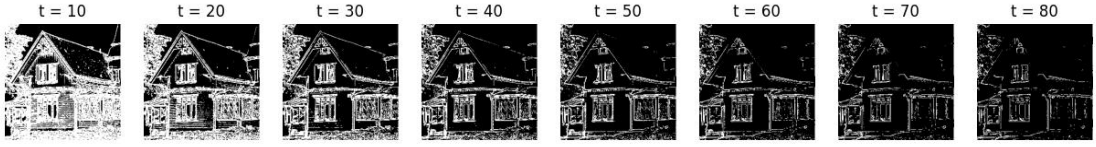
The comparison between tools and dxtools are shown below. It can be seen that dxtools is smaller than tools. The output of function convolve2d(..., 'valid') consists only of those elements that do not rely on the zero-padding. If we set the size of original image is $x \times y$, the

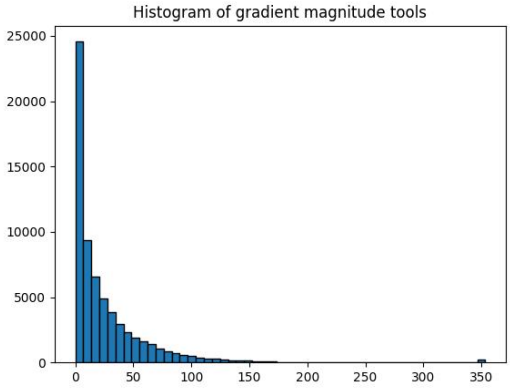
size of operator is *fx*×*fy*, the row of dxtools will be *x*-*fx*+1 and the column of dxtools will be *y*-*fy*+1.

	row	column
tools	256	256
sdo-dxtools	256	254
cdo-dxtools	256	254
robert-dxtools	255	255
sobel-dxtools	254	254

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

Answers:
The results using different thresholds are shown below. It can be seen that the best result is threshold=30.

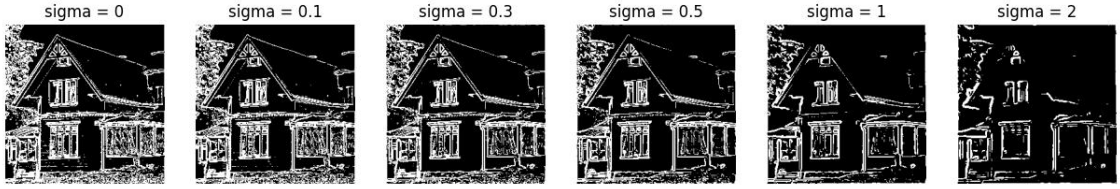




It is not easy to find an appropriate threshold. A lower threshold will result in wider edges and more edges would be found. But the result will also be affected by noise. A higher threshold will lose the mild part of edges.

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

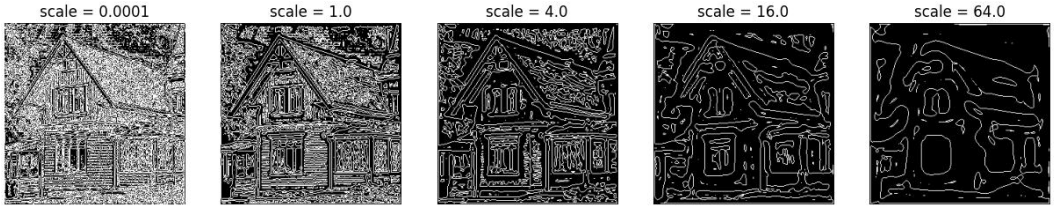
Answers:
Fixing the threshold to 30, the results after smoothing of different sigma are shown below.



The smoothing helps to some degree. The sharp edges contain too many details which may be detected as noise. So the high frequency components which may affect the edge detection will be eliminated after Gaussian smoothing.

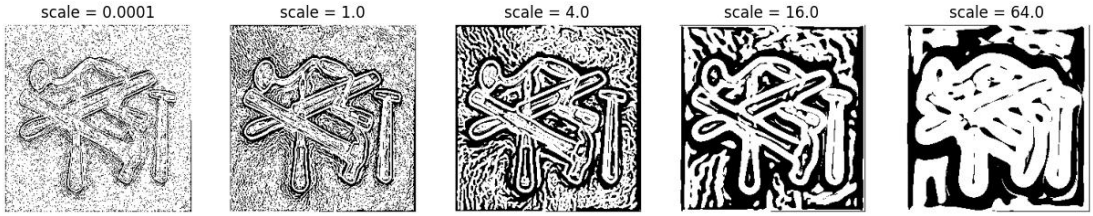
Question 4: What can you observe? Provide explanation based on the generated images.

Answers:
Smaller scale will show more edges but also introduce noise, while larger scale opposite. When the scale increases, the image is more blurry, which result in the loss of information and the distortion of edges.
The function *Lvvtilde()* returns the approximation value of the second order derivative, and the function *contour()* returns the number of zero-crossings. The texture details and noise will generate too many unwanted zero-crossings.

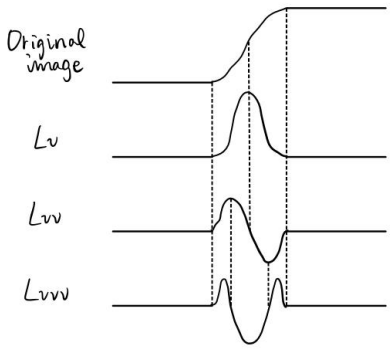


Question 5: Assemble the results of the experiment above into an illustrative collage with the *subplot* command. Which are your observations and conclusions?

Answers:
When the scale increases, the detected edges of the tools image will become thicker, since a smoother image has milder edges. Before blurred, the image with sharp edges has rapidly changed derivatives, which will cause thin edges.
$\widetilde{L}_{vvv} < 0$ is represented as white, and $\widetilde{L}_{vvv} > 0$ is represented as black.



Each order of derivatives are shown below.



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

Answers:
According to the second figure in Question 5, *Lvv*=0 can detect the extreme value (both maximum and minimum) in *Lv* and *Lvvv*<0 can help to avoid incorrect detection. The exact point belonging to the edge can be obtained by combining *Lvv*=0 and *Lvv*<0.

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

Answers:
The best results are shown below.

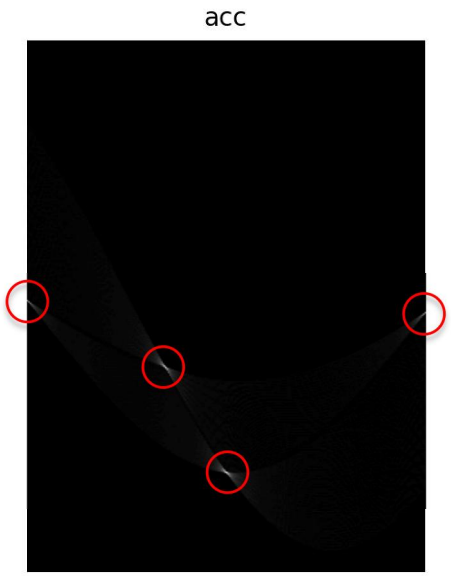
scale = 4.0, threshold = 8

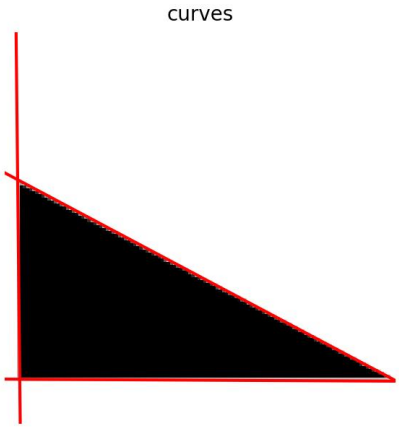
scale = 8.0, threshold = 6

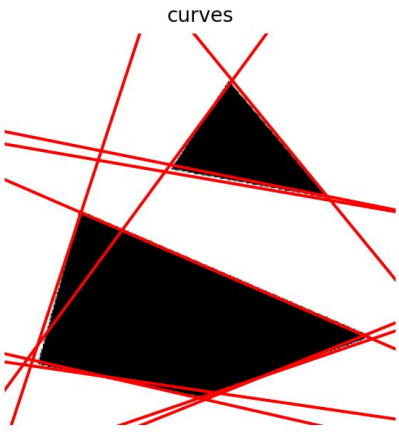
Question 8: Identify the correspondences between the strongest peaks in the accumulator and line segments in the output image. Doing so convince yourself that the implementation is correct. Summarize the results of in one or more figures.

Answers:

Since θ is defined between $\left[-\frac{\pi}{2},\frac{\pi}{2}\right]$, the leftmost and rightmost bright points in the accumulator represent the vertical line. The horizontal line is represented by the center bright point since it has a zero slope. The line of the hypotenuse of the triangle has a negative slope, so it is represented by the bright point to the left of the center, which lies between $\left[-\frac{\pi}{2},0\right]$.



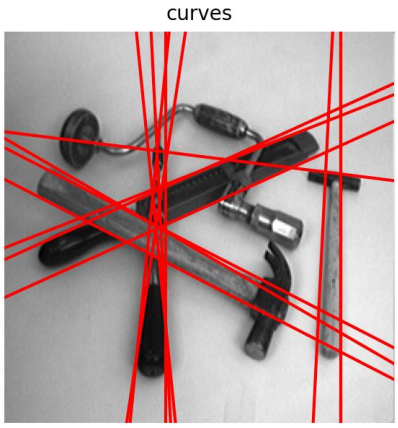




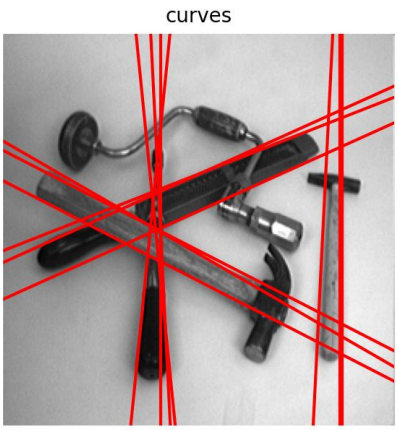
Question 9: How do the results and computational time depend on the number of cells in the accumulator?

Answers:
Increasing the number of cells will increase the computational time. The values of nrho and ntheta that are too low will result in an inaccurate direction of Hough lines, while the too high

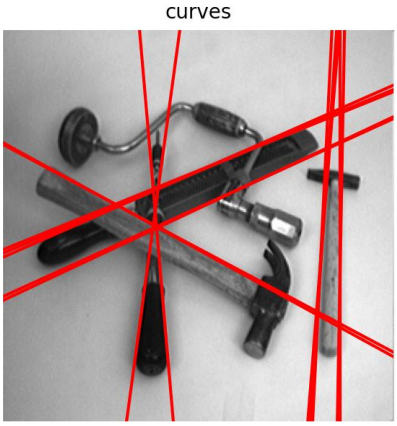
values will result in multiple responses for the same line and loss of lines that are not significant enough.



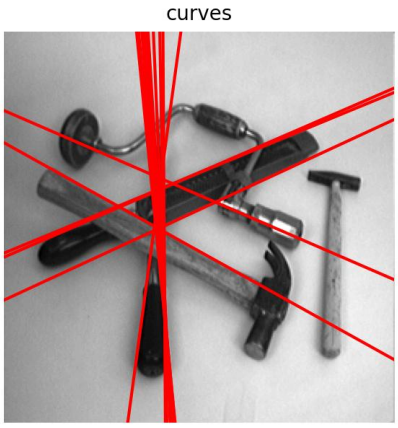
nrho=512, ntheta=64



nrho=1024, ntheta=64



nrho=512, ntheta=512

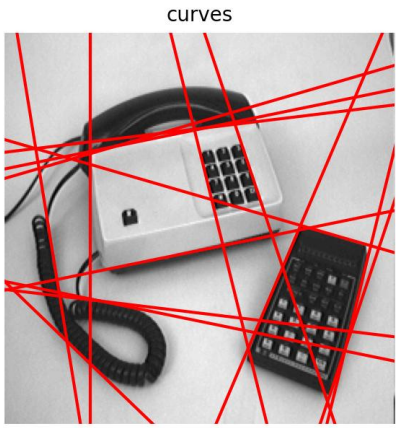


nrho=1024, ntheta=512

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?

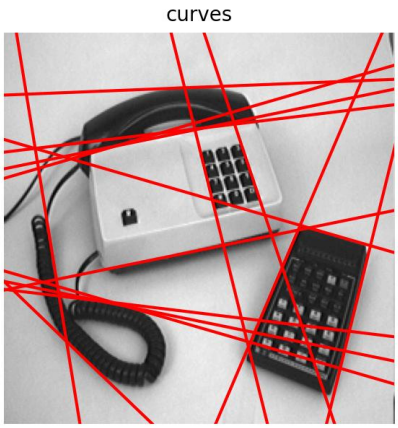
Answers:
We can change the updating procedure of the voting system. The update function can be set as *h*. In the normal case, *h*=1, which can be seen in the first figure below. If we change the update function to a form which is depending on the magnitude of the first order derivatives of the image, the results will decrease the critical dependency of threshold. The three forms of *h* are shown in the other three figures below.

When h=magnitude[x, y], the edges with stronger contrast will be picked up first, such as the lines between black and white. When h=np.log(magnitude[x, y]), the weight of the edges with lower magnitude values will also be found. So voting system with logarithm will be better than the original solution.



h=1

h=magnitude[x, y]



h=np.log(magnitude[x, y])

This is the result of applying houghedgeline() on godthem256.npy using scale=3, threshold=5, nrho=800, ntheta=100 and nlines=15.