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

Name: _____PIETRO ALOVISI_____Program: _____CDATE_____

**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:

I was expecting that the result in *dxtools* would enhance the change in intensity only on the horizontal axis, while *dytools* would do the same but in the vertical direction.
The different sizes are due to the fact that when we convolve we loose the the pixels on the edge because the convolution centered in those pixel would require to know the pixels outside the image.
In my case, with a filter `deltax=[-1/2,0,1/2]`, I loose a the full leftmost and rightmost columns, leading to a new image size of `256X254`.

---

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

Answers:
It is not easy using this method, because there is a trade off between the resulting thickness of the edge and the amount of edges I can detect. This is due to the fact that if I use a high threshold I get only the sharpest changes in the intensity in the image, which only take 1 or 2 pixels in the image, obtaining thin edges. But, since most of the real edges are blurred and their intensity grows as a ramp, choosing a high threshold leads to few edges detected, but if I choose a lower threshold I would get a thicker edge, because I will not filter the blurred part.

---

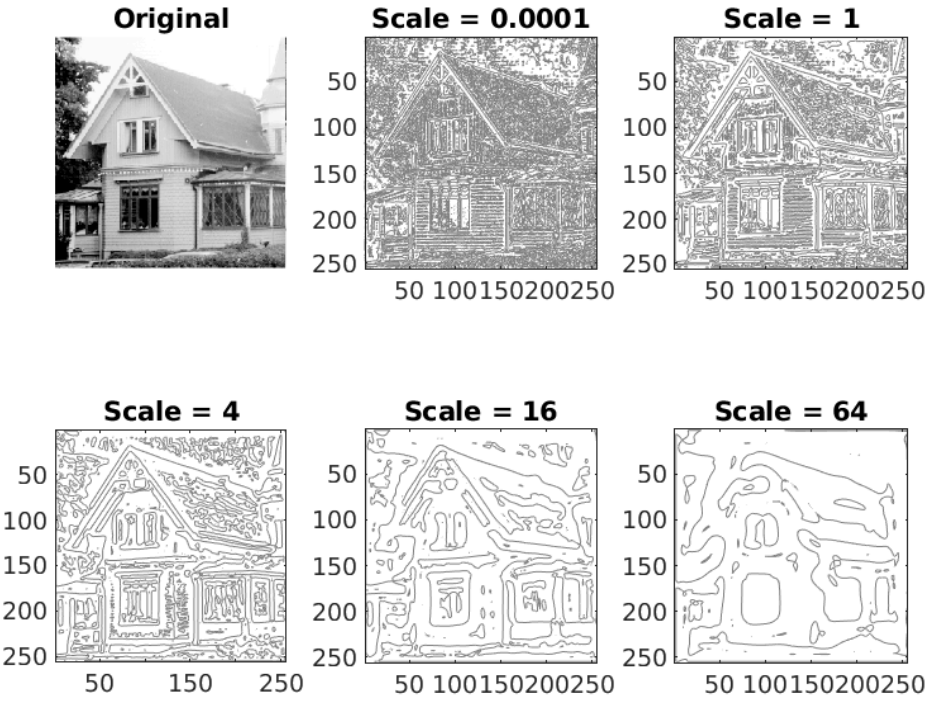
**Question 3**: Does smoothing the image help to find edges?

Answers:
Short answer: yes. It can help to reduce noise, and so to avoid some "fake" edges, but the most important effect is that it helps in selecting the "*scale*" of the edges.
For example, blurring out fine details of the house in `godthem256` leads to not recognize no more the wooden tiles of the house and the leaves in the tree and bushes.

---

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

Answers:
I can observe that if I choose a lower scale value I get almost all the edges in the image, even the small one like leaves and windows details. While for a higher value of the scale I get the coarser details, like the house and tree silhouette.
As I said before, the scale is very important because it lets us choose the size of the object we want to recognize. Basically the blur is a filter in the domain of the scale, it blurs out the smaller edges and leaves the image only with the coarser ones. So its value must be carefully selected.



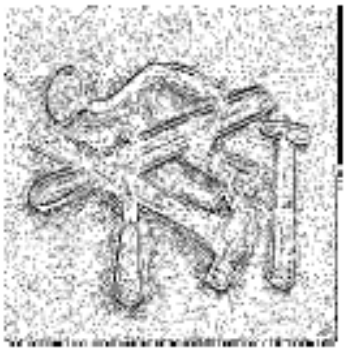

---

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

Answers:

Original

Scale = 0.0001



Scale = 1

Scale = 4

Scale = 16

Scale = 64

We observe that if we choose a greater scale we get wider white areas around the edges.

---

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

Answers:
I can detect edges by finding values where *Lvv* is 0, or better, interpolate those points from adjecent pixel having opposite sign values.
We can improve the result by looking at both *Lvv* and *Lvvv* at the same time. As highlighted in the theory part, we want *Lvv* to be 0 to have a single pixel in the border, while we choose *Lvvv* to be negative to make sure that the point is a maximum point in the magnitude of the magnitude of the first derivative (in the direction of the gradient).

---

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

Answers:
Here are the best results I got. It's quite hard to tune the house one because of the low contrast of some parts of the roof with the sky, and because of the small details on the image. For the roof we need a small scale, otherwise it will be blurred too much and not be detected, while the small details require a high scale to be ignored.

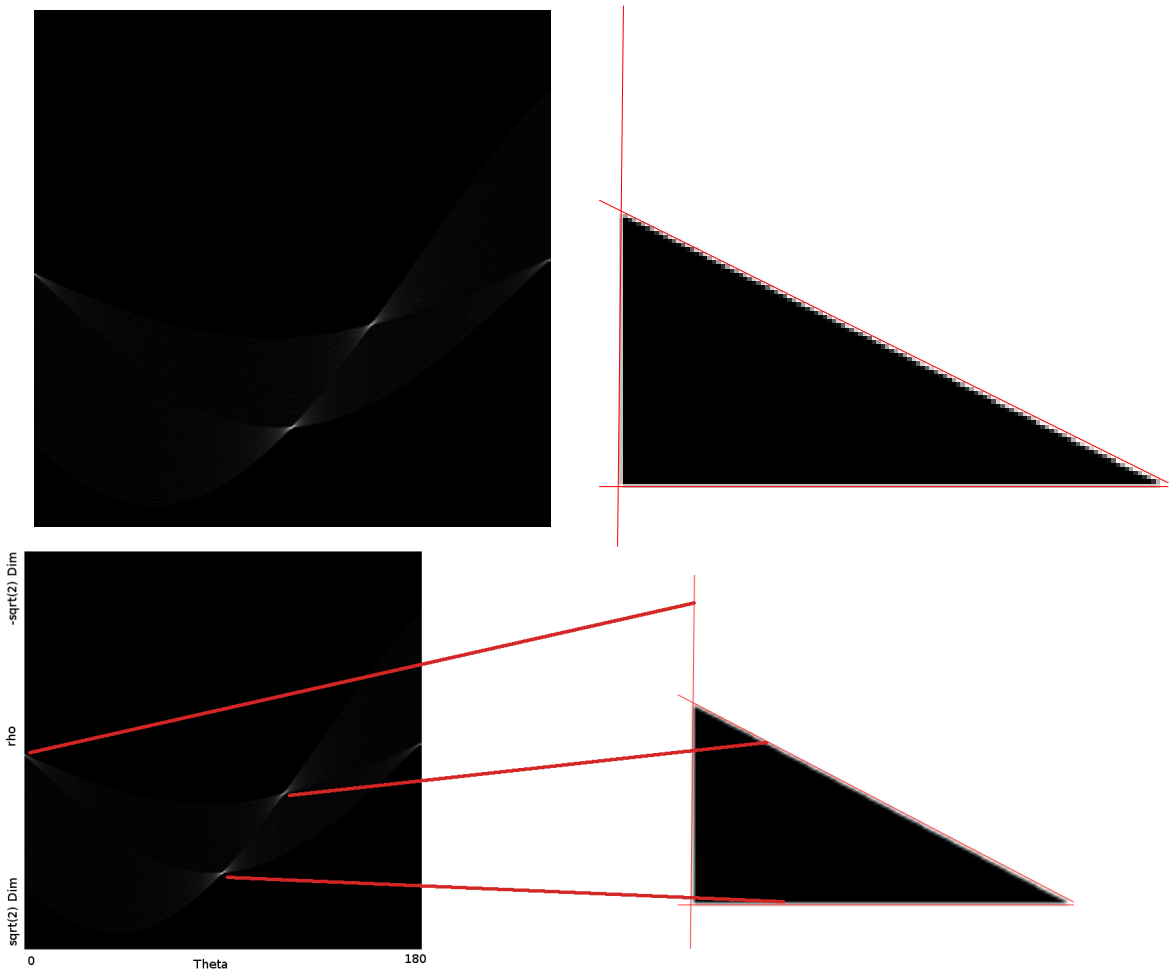
**scale 15, threshold 10**

**scale 8, threshold 15**

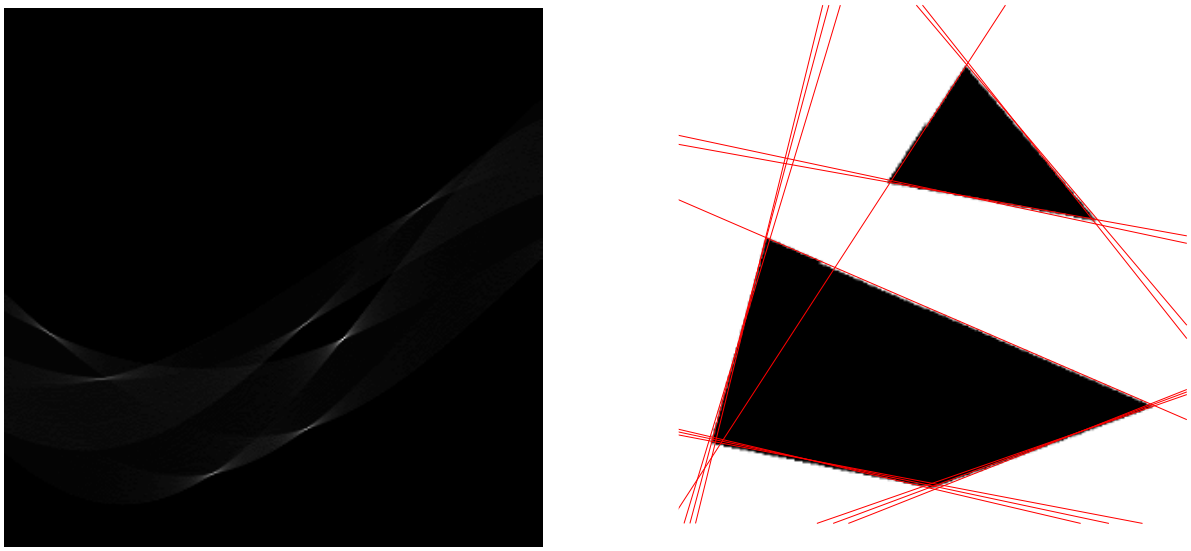
---

**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.

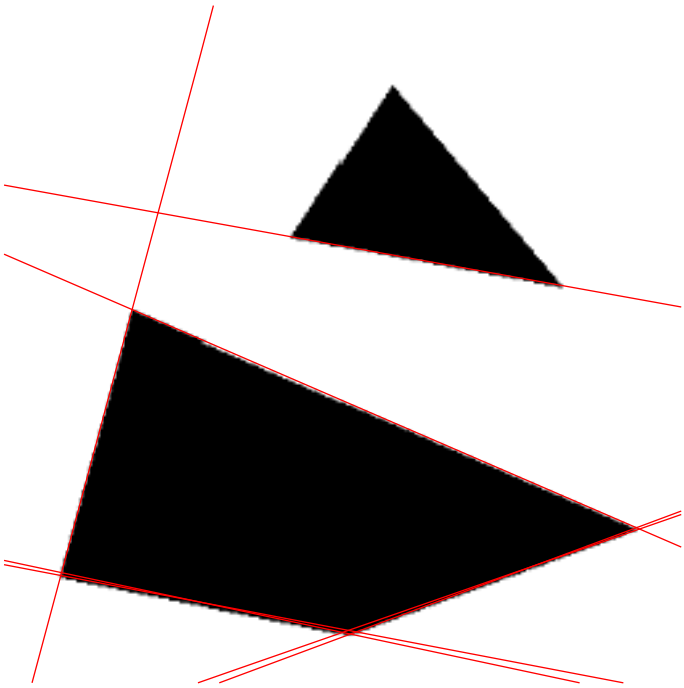
Answers:
This is the result applied to the triangle image. I plotted the 3 most likely lines, and we can find the correspondences easily. I left the origin in the top-left corner, and the axis using the normal convention for images. Then for the parameter $\theta \in [0°, 180°], \rho \in [-\sqrt{2}N, \sqrt{2}N]$ where $N$ is the image size(assuming square image).we can spot the 3 peaks as having approximately the coordinates $(\rho, \theta)$ : $(0,0), (N, 90), (N/3, 120)$ .
The correspondences of these tree points are highlighted by the red lines in the image below.




Here is the result on the algorithm for the first 15 best lines In the image `houghtest256`.



And here for the best 7 lines. As we can see the algorithm does not get all the lines, because some of them are counted twice, probably because the value in the neighborhood of their maxima is still greater than some peaks of the other maxima.



---

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

Answers:
For simplicity assume a square image of size $N$. Then we have to scan over the points belonging to an edge, that are maximum all pixes $N^2$. Then we have to compute for each of these points its value of $\rho$ for each of the $N_\theta$ parameter $\theta$. Computing $\rho$, and incrementing the accumulator takes $O(1)$. Then to find the $m$ maximums in the accumulator we need to scan all the cells and perform some comparison that ends up in $O(N_\theta N_\rho m)$ time complexity.
so the overall complexity of the algorithm is:

$$O(N^2 N_\theta + N_\theta N_\rho m)$$

---

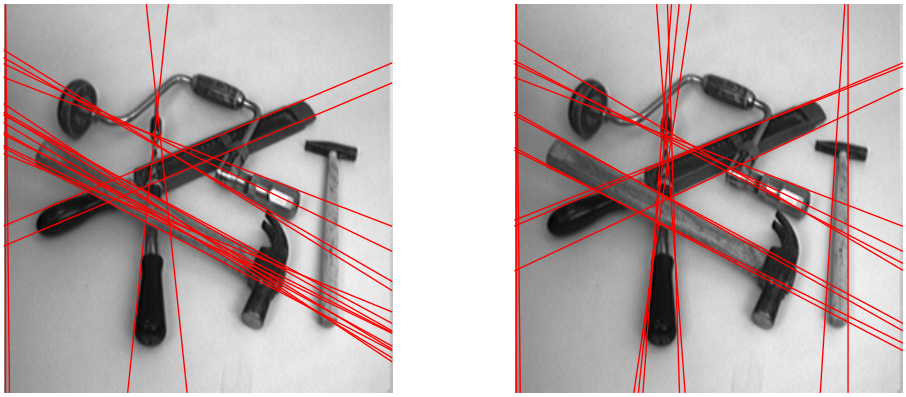
**Question 10**: How do you propose to do this? Try out a function that you would suggest and see if it improves the results. Does it?

Answers:
I implemented the new vote for each point as :

$$1 + k \cdot \log(\nabla L)$$

This function is the same as the original one plus an additive term in the gradient, that grows as the logarithm to avoid that high contrast borders would take very high vote compared to the ones at low resolution. If we apply this method to the tools image, choosing k=e, we get:

**Normal vote** **1 + k * log(magnitude)**



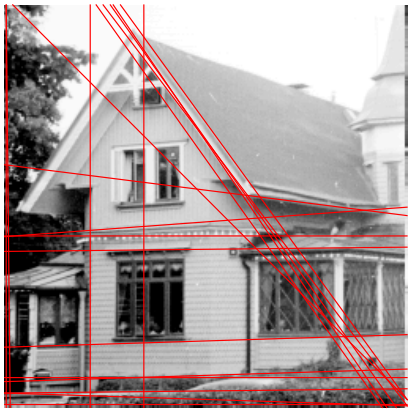
This helps, as we can see lines are spread evenly among the objects, and not only focused in the middle hammer as in the left one. Moreover, we removed some lines that were caused by noise or details.

Let's see the effect on the house image. Here we can see two effects:

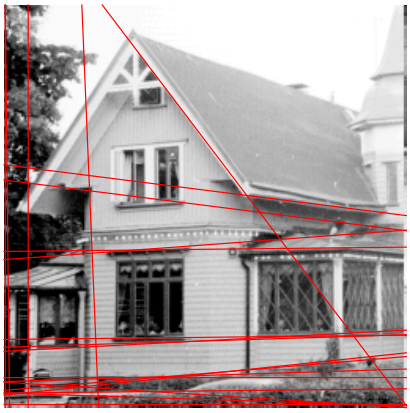
- The "messy" parallel lines on the right images tends to get closer, therefore the result is more clean
- On the other hand this new voting method doesn't highlight any other edge that was not present in the first image.

So this new function can be used primarily as a regularization term, that sometimes can spot new edges.

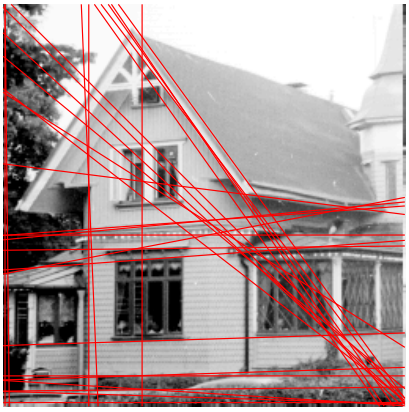
**Normal vote, 20 lines**



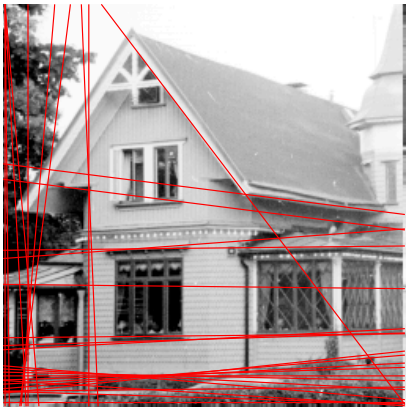
**1 + k * log(mag), 20 lines**



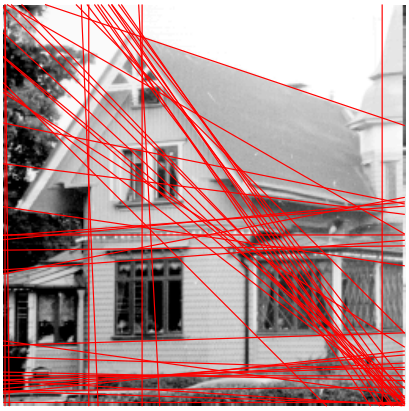
**Normal vote, 30 lines**



**1 + k * log(mag), 30 lines**



**Normal vote, 50 lines**



**1 + k * log(mag), 50 lines**