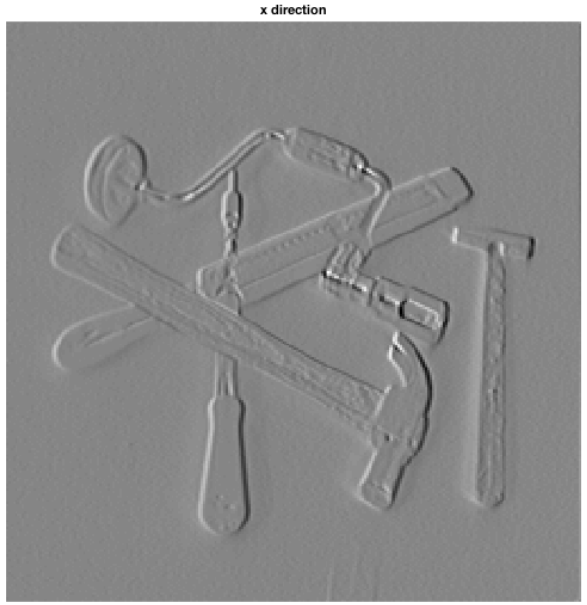
Lab 2: Edge Detection & Hough Transform

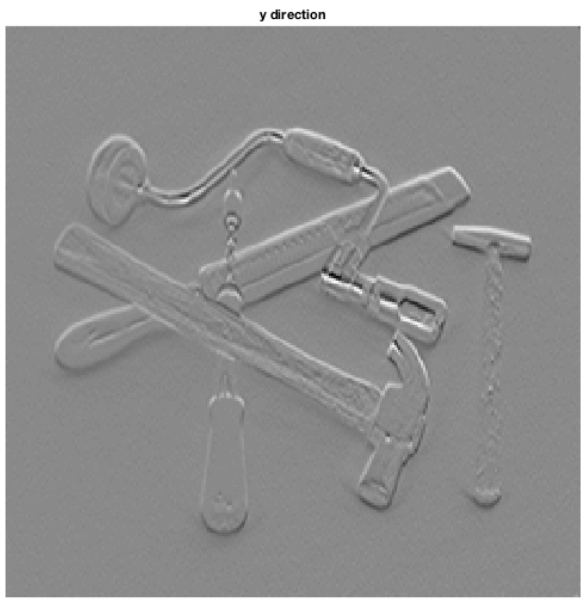
Tianxiao Zhao

The lab is about ...

- Derivative operators
- Thresholding for edge detection
- Differential geometry based edge detection
- Hough transform

Q1:





- Expected results:
  - grey level will be around zero for most part of dxtools and dytools
  - positive or negative ridges at edges (large gradient magnitude near edges)
  - dxtools only emphasizes differences in x direction, gives stronger responses for vertical lines

    similar for dytools

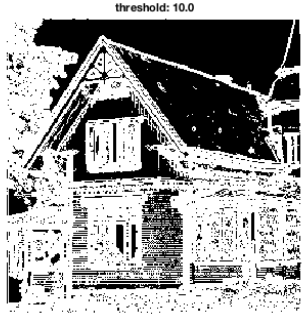
Q1:

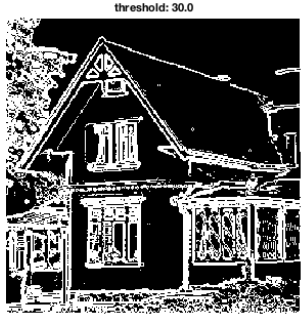
image	x-wise derivative		y-wise derivative	
	row	column	row	column
tools	256	256	256	256
SDO	256	254	254	256

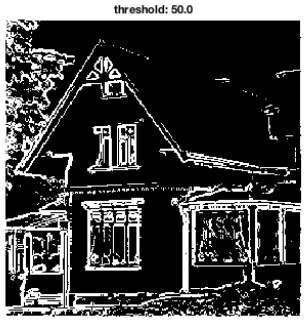
- dxtools smaller than tools
- conv2(… , ‘valid’) returns convolution without zero-padding at edges

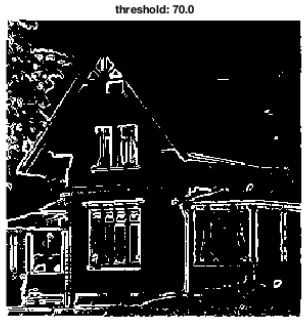
  (SDO is a 1*3 matrix)

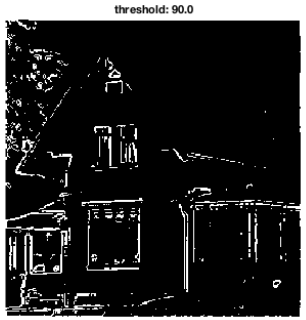
Q2:



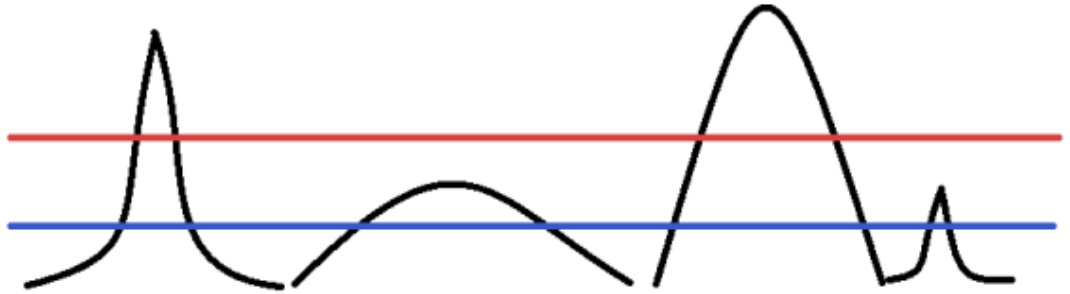








Discrete Gaussian filter, sigma = 0.2,
threshold = {10, 30, 50, 70, 90}



- easy to find an okay one, but hard for a perfect one
- **Reason:**
  - edges not consistently steep
  - lower threshold -> sharp edges become wide
    -> local maxima due to noise

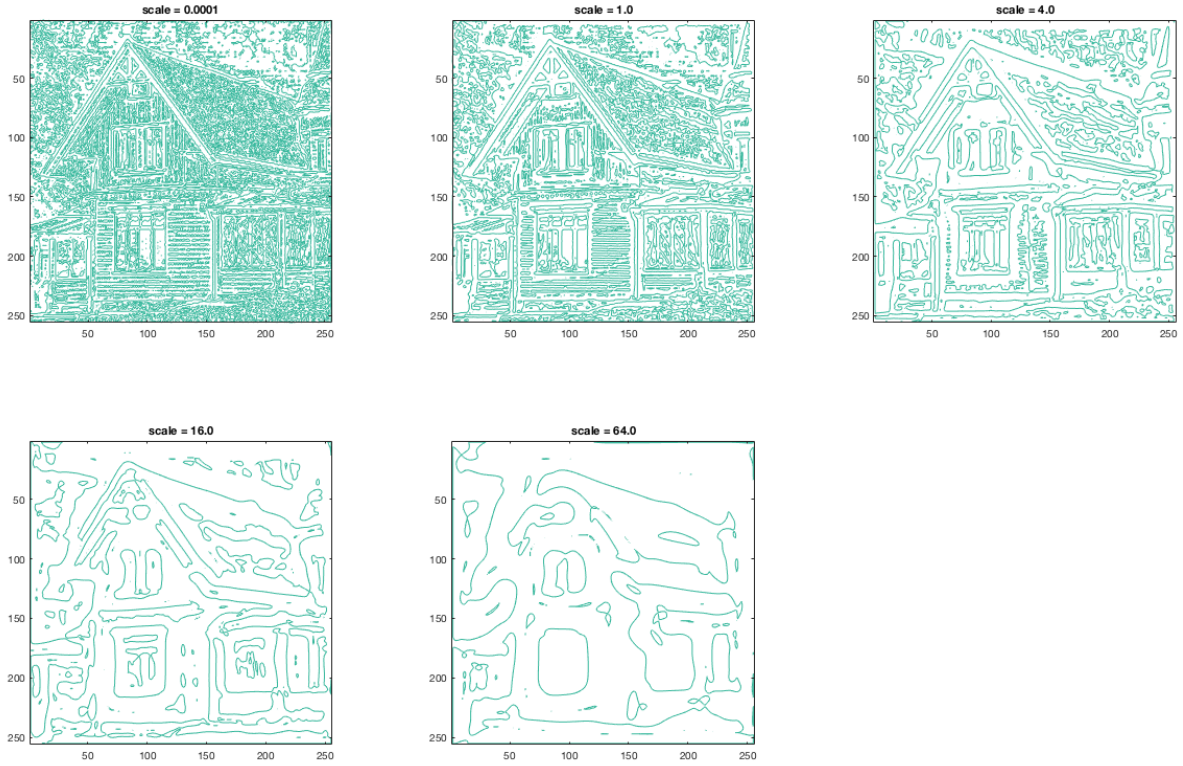
    higher threshold -> mild ones break or fade

Q3:

Discrete Gaussian filter, threshold = 30,
sigma = {0, 0.01, 0.1, 0.5, 1}

- Smooth helps to some degree
- **Reason:**
  - get rid of high frequency noises
  - even harder to choose threshold
  - lose edge information and distortion
  - a trade-off to the degree of smoothing

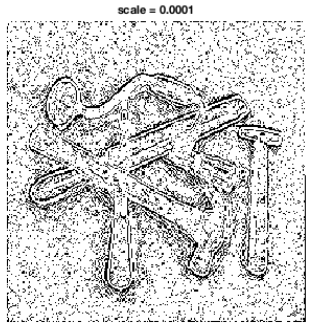
Q4:

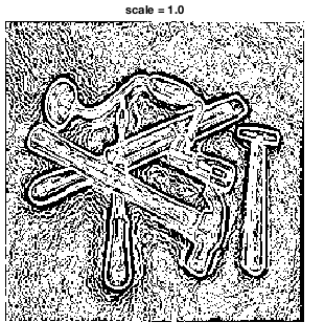


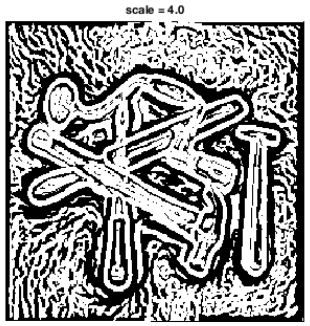
Discrete Gaussian filter, scale = {0.0001, 1.0, 4.0, 16.0, 64.0}

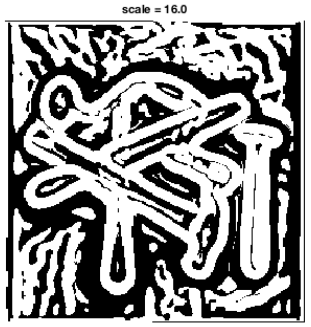
- small scale -> noisy image, every tiny details shown by many curves
- large scale -> edge curves more sparse and clearer
- too large scale -> some curves fade and distort
- **Reason:**
  - many zero-crossings due to minute waves, such as texture details and noise
  - smoothing -> fewer noises in 2nd-order zero-crossing contour
  - too much smoothing -> distort edges

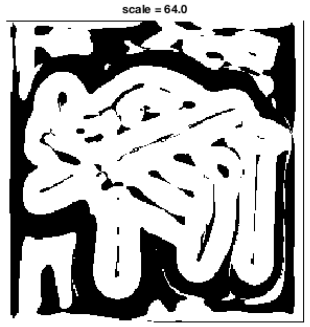
Q5:







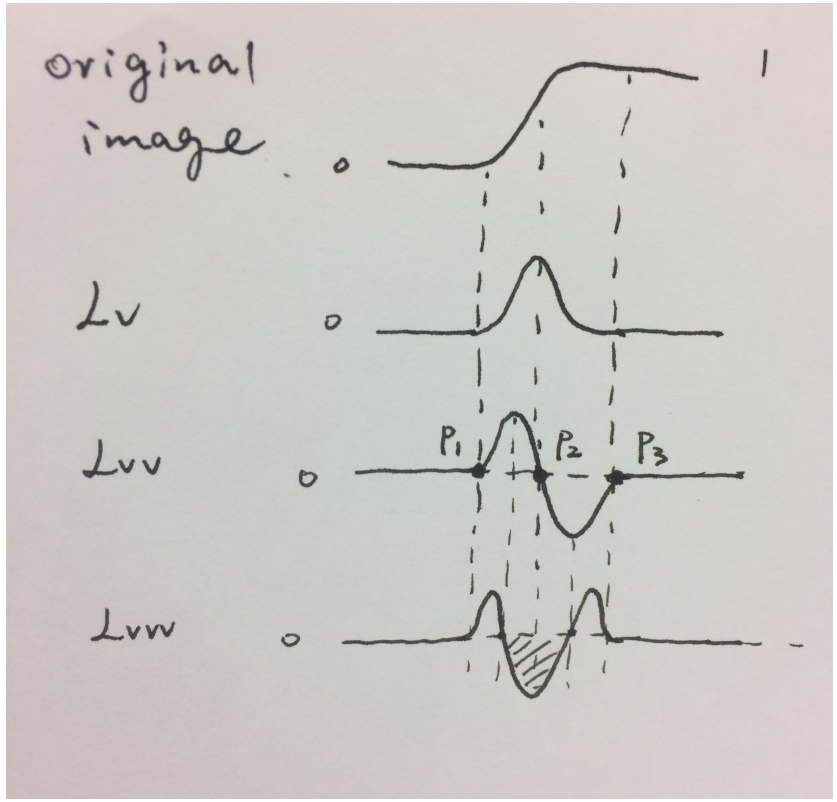


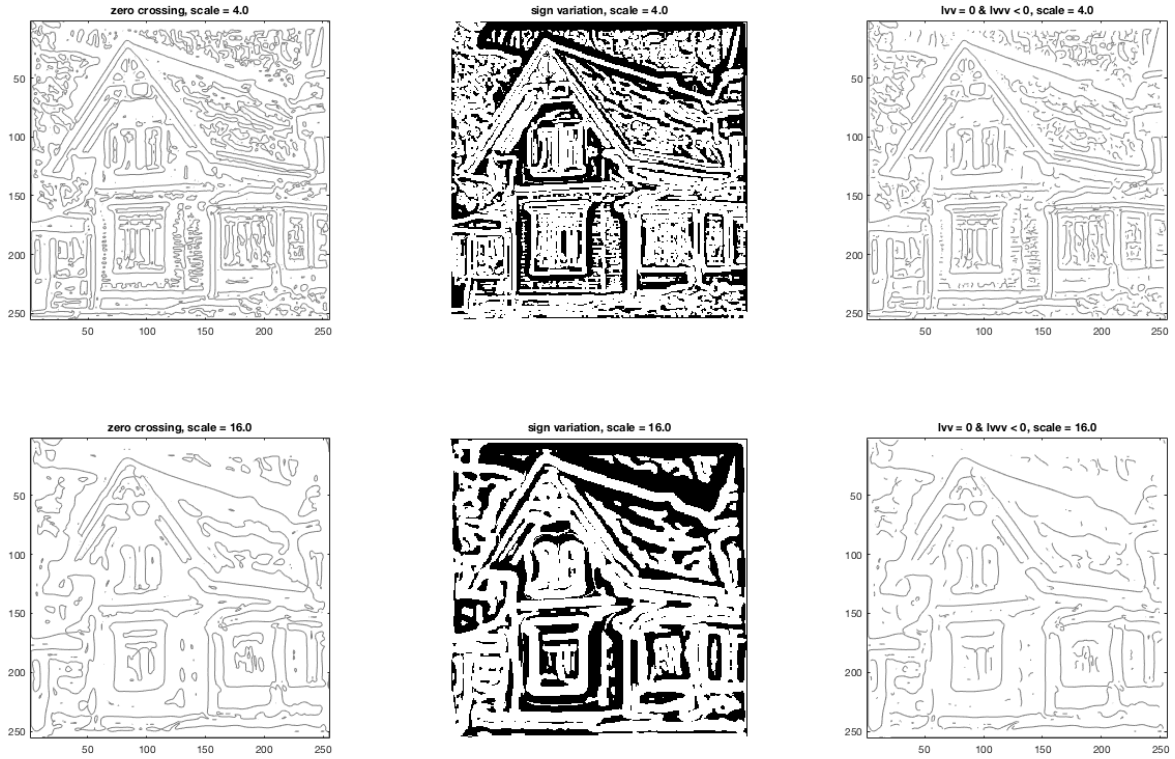


Discrete Gaussian filter, scale = {0.0001, 1.0, 4.0, 16.0, 64.0}

- lose a clear vision of edges
- white area -> edges + pixels with negative 3rd order derivatives
- Image smoother -> edges milder -> wider region
- Smoothing at certain level helps emphasize the edges; too much -> erode other parts of image

Q6:



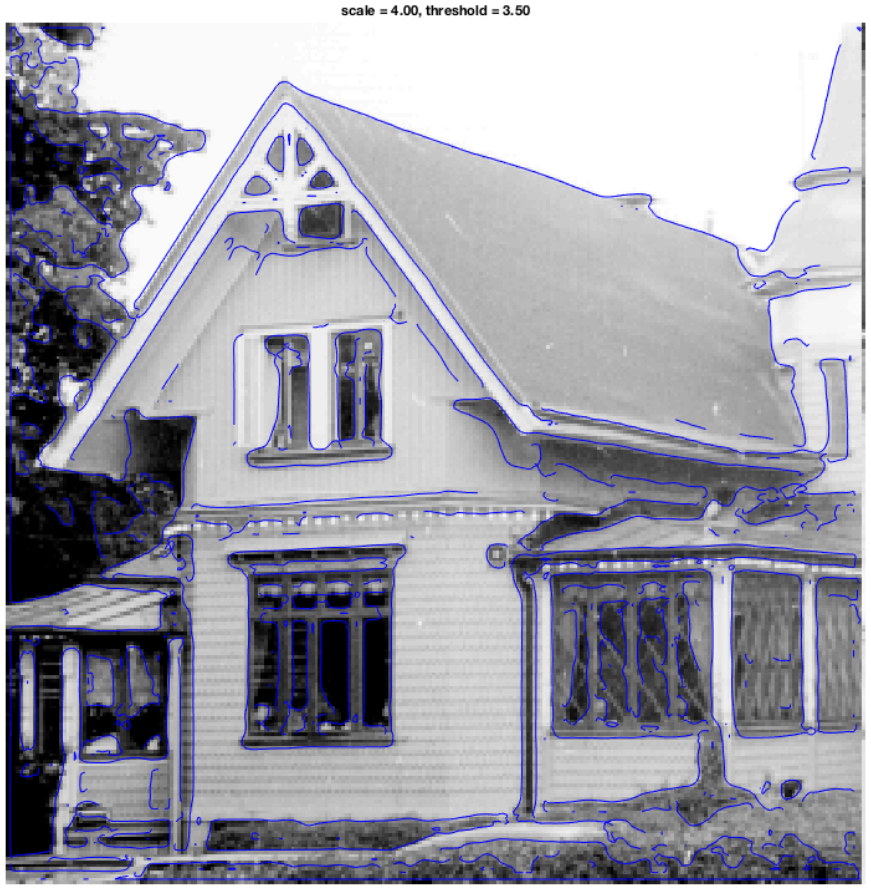


threshold for Lv to get rid of minute waves

- pick out points with Lvv = 0, link them into lines
- add constrain Lvvv < 0 to avoid incorrect detection
- trivial details and noise are main problem for complex images -> do not improve much

Q7:

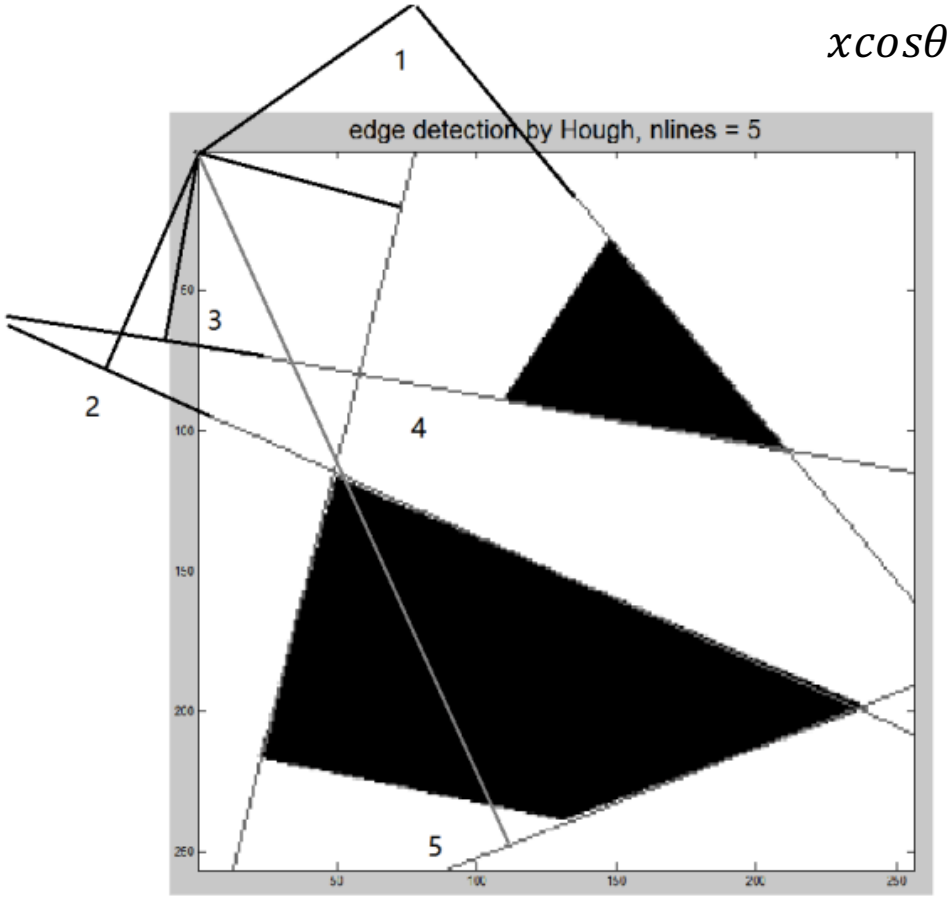
scale = 4, threshold = 8

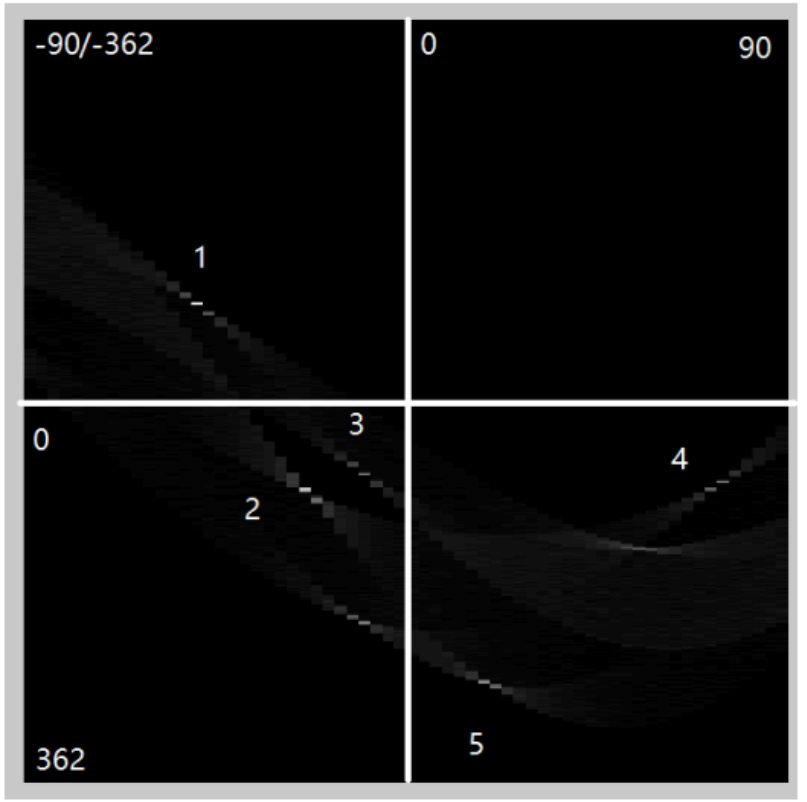


scale = 4, threshold = 3.5

Q8

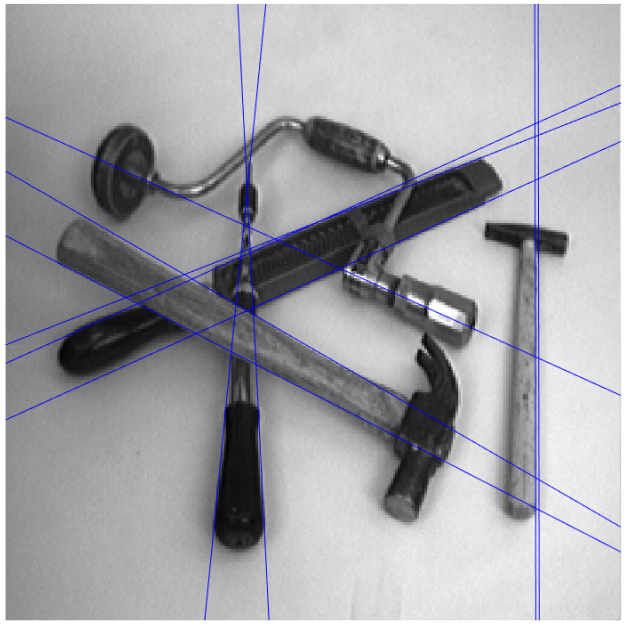
$$x\cos\theta + y\sin\theta = \rho$$



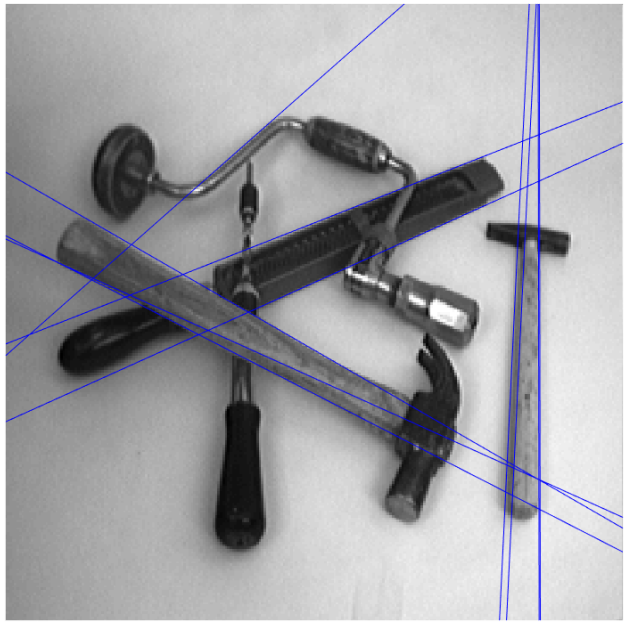


rho (pixel)	-92.812	84.310	68.724	75.808	271.352
theta (deg)	-46.34	-22.51	-9.27	70.17	19,86
votes ranking	1	2	3	4	5

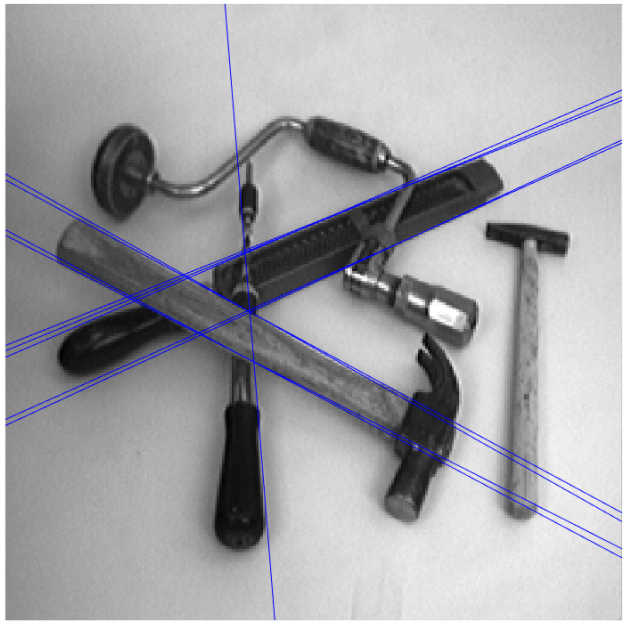
Q9:



nrho = 512, ntheta = 64



nrho = 1500, ntheta = 64

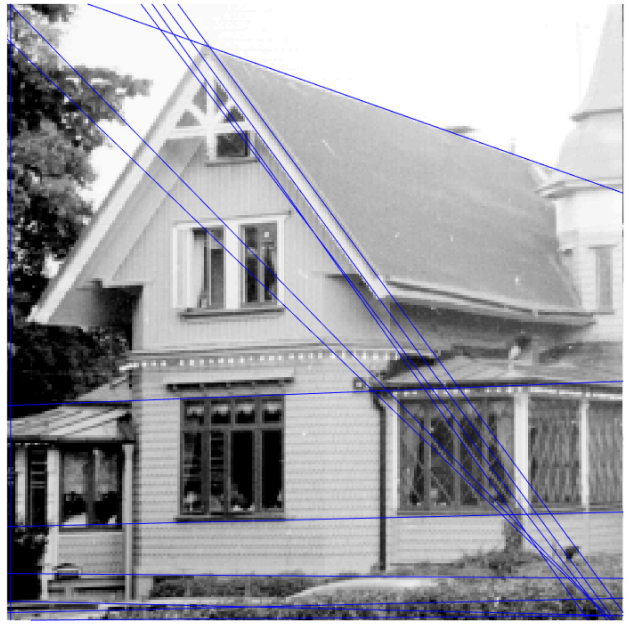


nrho = 512, ntheta = 512

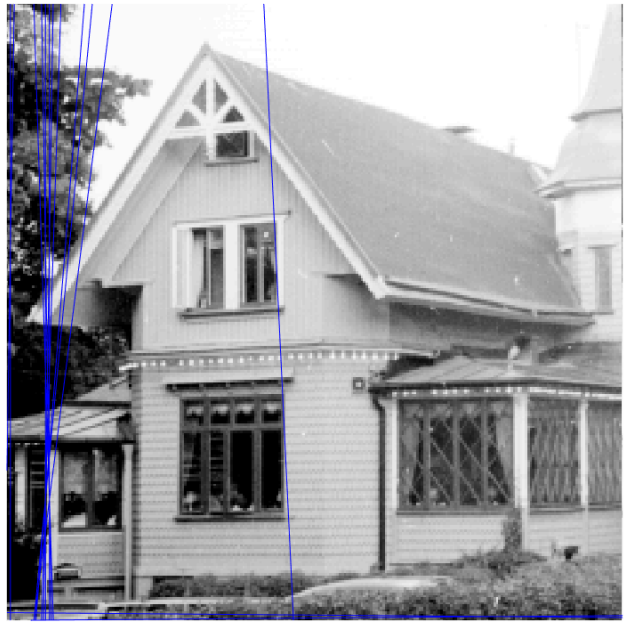
- Increase cells -> longer computation time; more lines gather at sharper edges, local maxima near edge line
- Decrease cells -> shorter computation time; cannot detect edges accurately

Q10:

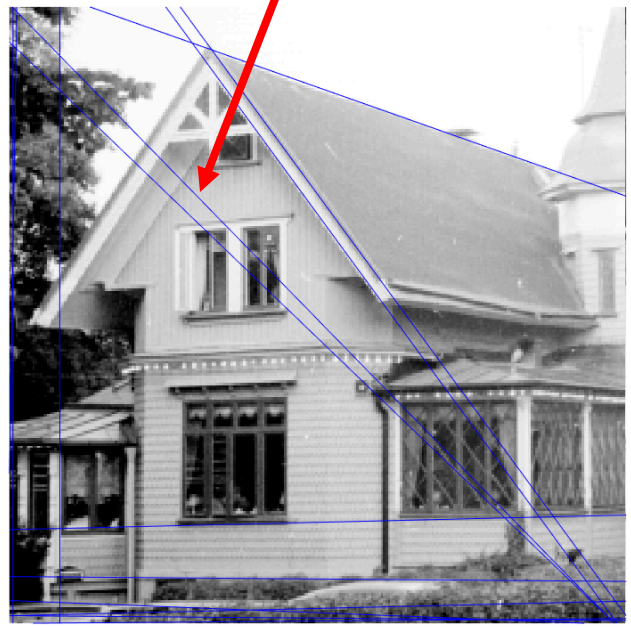
Points may not be from the same edge curves.
That is why these lines appear.



h( ) = 1



h( ) = mag^2



h( ) = log(mag)

- Incrementing with gradient magnitude reduces critical dependency on threshold
- Some parts of image have large noise -> h( ) = 1 -> treat noise in the same manner as other points
- h( ) = log(mag) -> increase the possibility of edges with lower values of gradient magnitudes