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?

```
deltax = [1 0 -1; 2 0 -2; 1 0 -1];
deltay = [1 2 1; 0 0 0; -1 -2 -1];
tools = few256;
dxtools = conv2(tools, deltax, 'valid');
dytools = conv2(tools, deltay, 'valid');
figure()
subplot(1,3,1);
showgrey(tools);
title("Original image")
subplot(1,3,2);
showgrey(dxtools);
title("Deltax")
subplot(1,3,3);
showgrey(dytools);
title("Deltay")
```

We expect to see contrast when the derivative is high in x for the dxtools and similarly for y. This can easily be seen in the screwdriver handle. For y, it has a higher "activation" than in y.

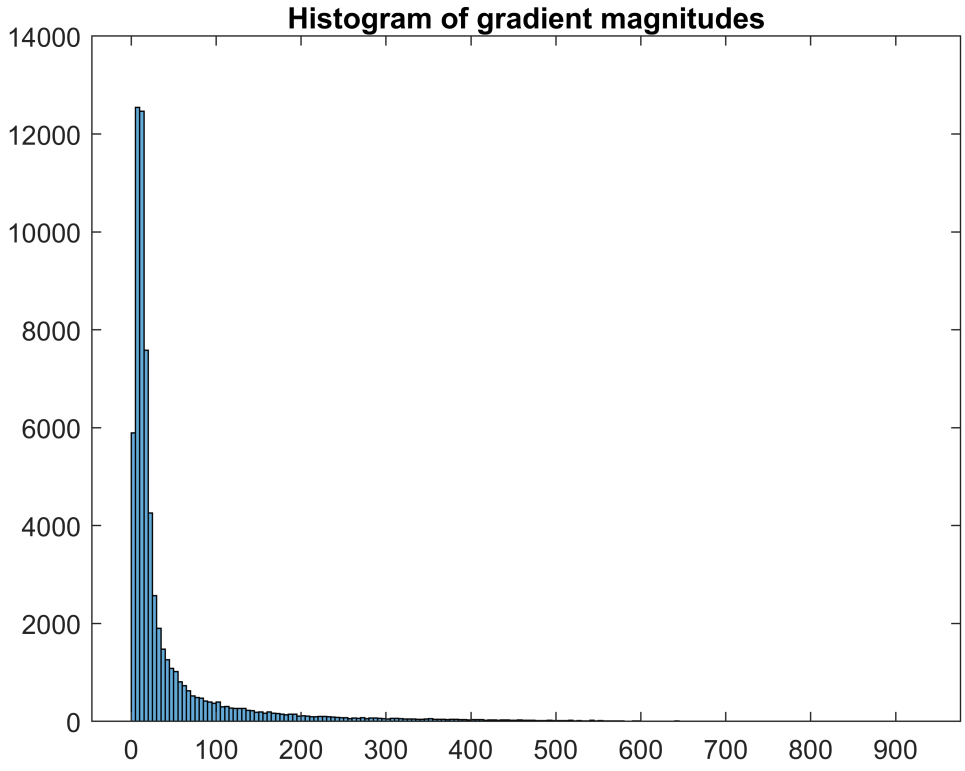
The sizes are different because of the 3x3 filter. conv2 with valid returns only pixels where the whole filter fits, which for a 3x3 filter causes 2 rows and columns to be removed. If we use 'same' instead of 'valid' this doesn't happen.

Question 2 & 3. Is it easy to find a threshold that results in thin edges? Explain why or why not!

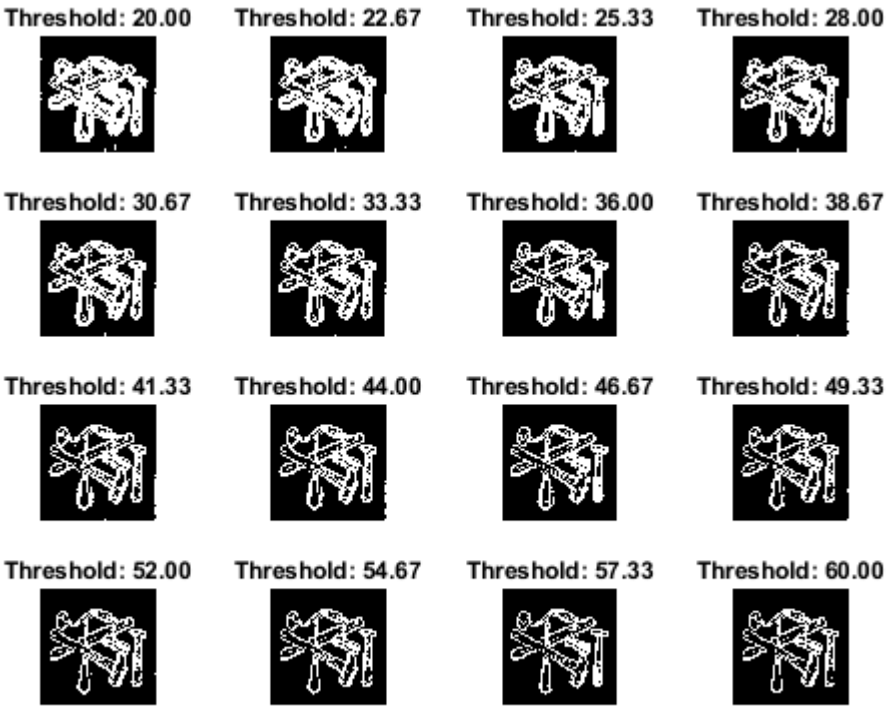
Does smoothing the image help to find edges?

```
figure()
thresholds = linspace(20,60,16);
gradmagntools = sqrt(dxtools .^2 + dytools .^2);
showgrey(gradmagntools);
title("No threshold gradient magnitudes");
```


```
histogram(gradmagntools);
title("Histogram of gradient magnitudes");
```



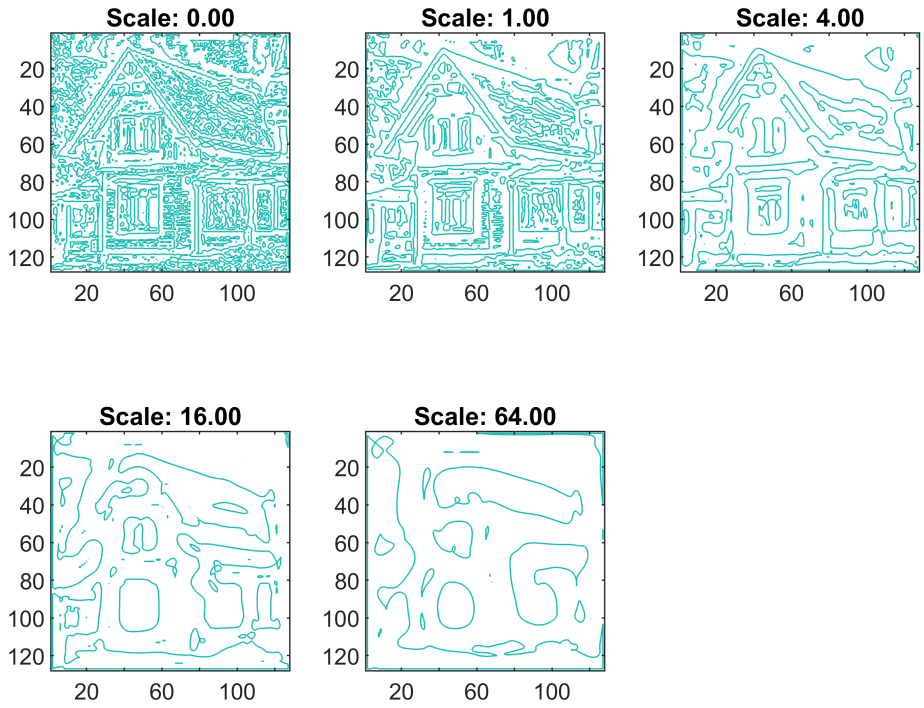

```
thresholdcomp(tools, thresholds);
```



It is not easy to find a threshold that results in thin edges. This could be because of smoothing, which results in wider edges. With a higher threshold we get a thinner edge.. Smoothing helps with finding edges because it removes a lot of noise and therefore the false edges.

Question 4-6: What can you observe? Provide explanation based on the generated images.

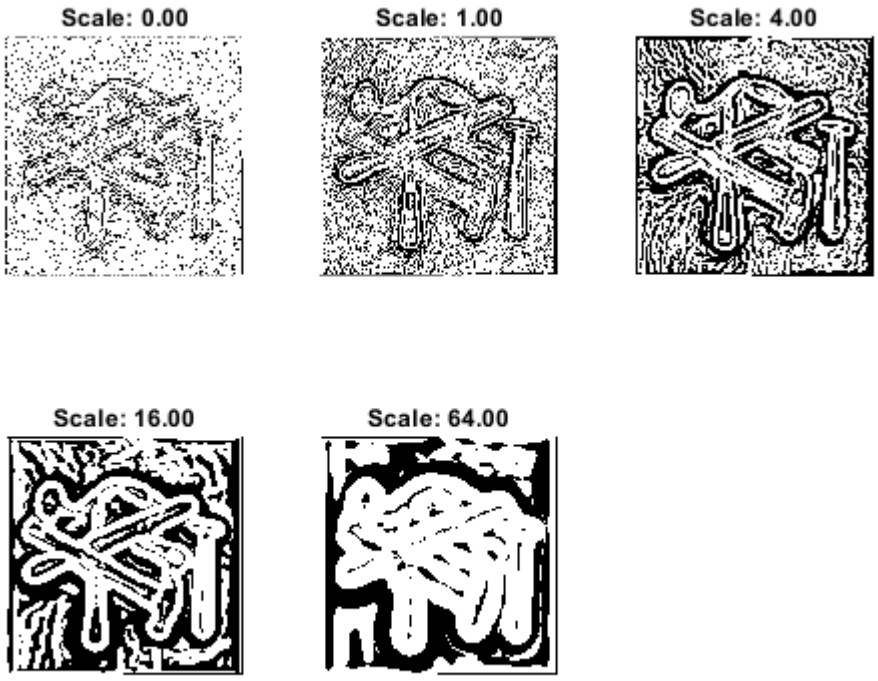
```
figure()
house = godthem128;
scales = [0.0001, 1.0, 4.0, 16.0, 64.0];
for sc = 1:length(scales)
    subplot(2,3,sc)
    contour(Lvvtilde(discgaussfft(house, scales(sc))), [0 0]);
    axis('image')
    axis('ij')
    title(sprintf("Scale: %0.2f",scales(sc)));
end
```



When smoothing we blur out edges that correspond to noise. The gradient magnitude reaches a maximum quite often on noisy images.

What is the effect of the sign condition in this differential expression?

```
figure()
tools = few256;
for sc = 1:length(scales)
    subplot(2,3,sc)
    showgrey(Lvvvtilde(discgaussfft(tools, scales(sc)))<0);
    axis('image')
    axis('ij')
    title(sprintf("Scale: %0.2f",scales(sc)));
end
```



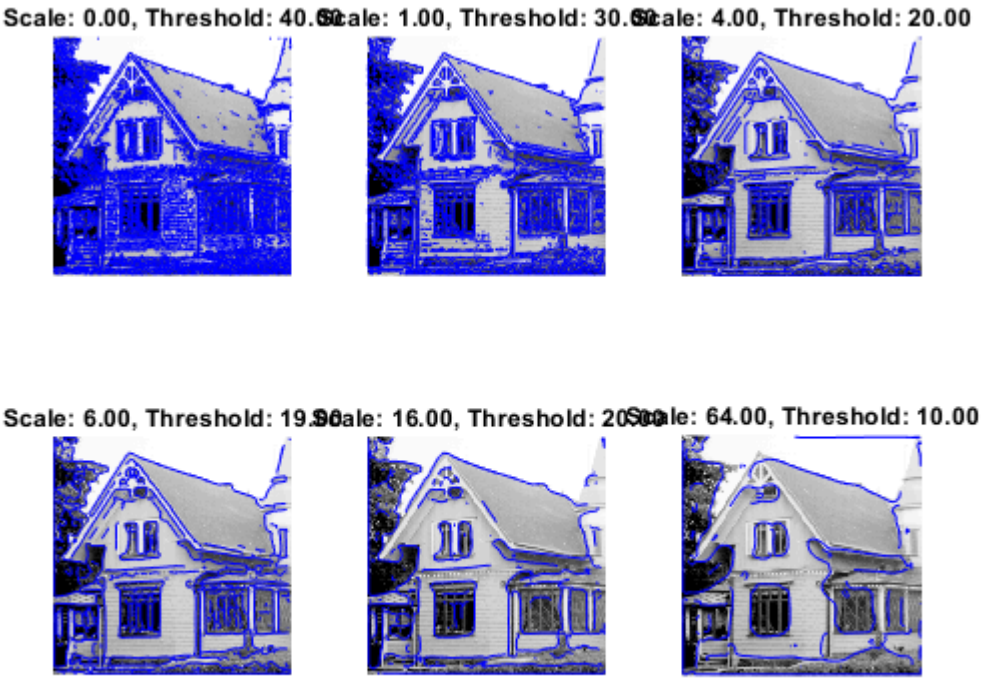
The sign condition masks the image into only black and white pixels. White pixels are where the sign is negative, which is where the gradient magnitude can be maximum and edges can be found.

We can conclude that smoothing the image results in removal of finer structures in the image and those edges. If we smooth more, we see that we have a much less noisy structure of possible edge locations.

We can combine the result of Lvv and Lvvv, by considering edges as where Lvv is 0 and Lvvv is negative.

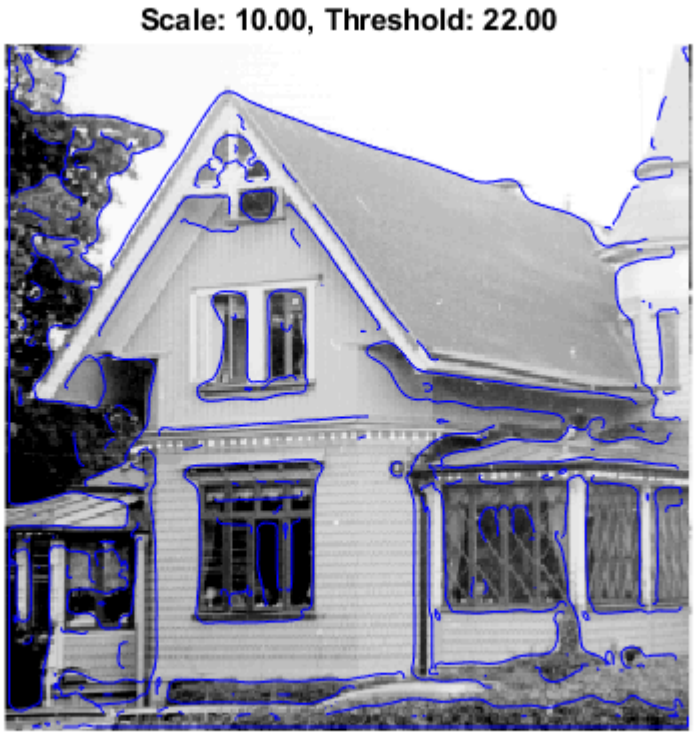
Question 7

```
warning('off','all')
figure()
scales = [0.0001, 1.0, 4.0, 6.0,16.0, 64.0];
thresholds = [40, 30, 20, 19, 20, 10];
house = godthem256;
for sc = 1:length(scales)
    subplot(2,3,sc)
    curves = extractedge(house,scales(sc),thresholds(sc));
    overlaycurves(house,curves);
    title(sprintf("Scale: %0.2f, Threshold: %0.2f",scales(sc), thresholds(sc)));
end
```



House:

```
warning('off','all')
house=godthem256;
figure()
scale = 10;
threshold = 22;
curves = extractedge(house,scale,threshold);
overlaycurves(house,curves);
title(sprintf("Scale: %0.2f, Threshold: %0.2f",scale, threshold));
```



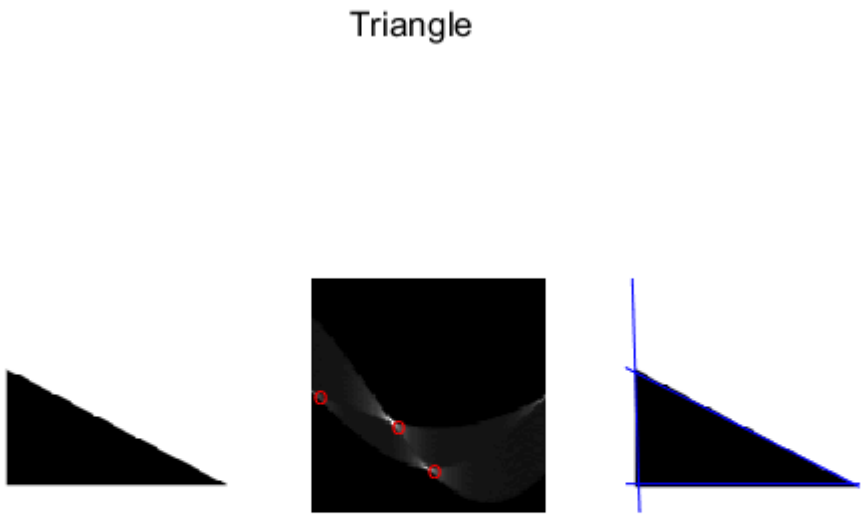
Tools:

```
warning('off','all')
figure()
tools = few256;
scale = 8;
threshold = 30;
curves = extractedge(tools,scale,threshold);
overlaycurves(tools,curves);
title(sprintf("Scale: %0.2f, Threshold: %0.2f",scale, threshold));
```

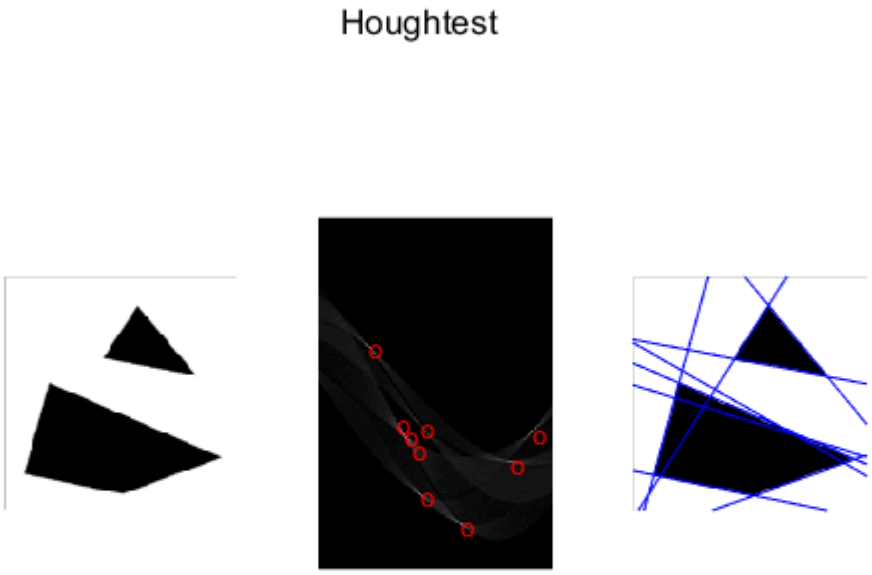
Question 8:

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 in one or more figures

```
figure()
triangle = triangle128;
houghedgeline(triangle,2,10,100,100,3,1);
sgtitle("Triangle")
```

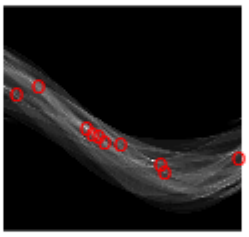



```
figure()
house = binsubsample(houghtest256);
houghedgeline(house,5,300,300,200,9,1);
sgtitle("Houghtest")
```



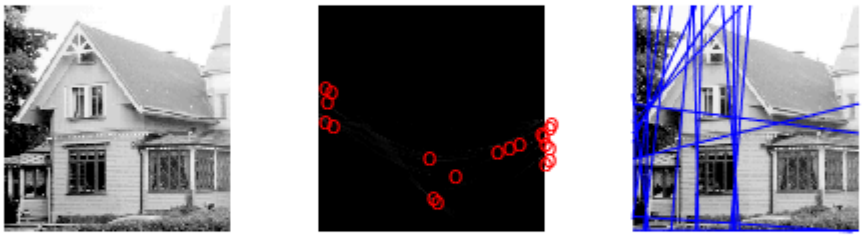

```
figure()
few = few256;
houghedgeline(few,1,20,340,360,10,1);
sgtitle("Tools")
```

Tools

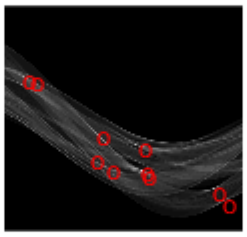

```
figure()
house = godthem256;
houghedgeline(house,20,100,1000,1000,20,1);
sgtitle("House")
```

House



```
figure()
phone = phonecalc256;
houghedgeline(phone,5,100,340,360,10,1);
sgtitle("Phone")
```

Phone


The strongest peak in the accumulator becomes a line in cartesian space with the corresponding theta and rho. Matlab origin is in the top left corner, negative angles to the left.

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

O(ntheta*pixels + ntheta*nrho*LOG (ntheta*nrho)) Calculating rho for each pixel and angle and sorting the found local maximas which in worst case should be the size of the accumulator space divided by 4.

Try out incrementing with a increasing function of the gradient magnitude

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

Voting with the gradient acts sort of like a magnitude threshold. Noise has a low gradient magnitude and is taken less into account.