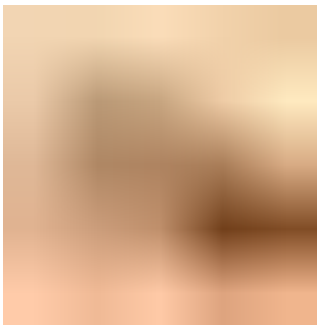
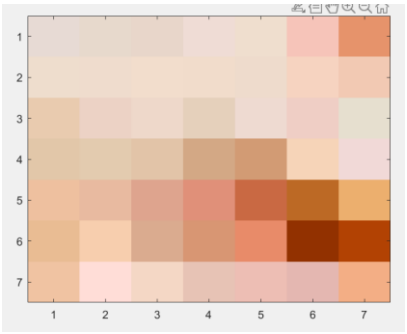
Question 1

(a)





The small neighborhood with 7*7 pixel chosen is like this. Matlab reads this image as: (The initial image is denoted as I, and this neighborhood is J)

J(:,:,1) =

231 231 232 239 239 246 230

238 239 242 241 238 246 242

233 236 238 229 238 239 230

226 227 226 211 210 246 241

238 232 222 224 201 188 236

233 247 218 216 232 146 178

240 255 244 231 237 228 243

J(:,:,2) =

218 217 214 220 222 196 147

221 220 221 220 219 211 201

203 210 216 208 218 206 223

199 203 196 168 155 212 217

192 186 164 144 105 105 175

188 206 171 150 139 49 66

195 221 215 195 190 183 174

J(:,:,3) =

212	204	202	213	206	185	107
205	205	204	203	204	192	179
175	197	202	187	209	197	207
169	175	168	133	116	184	215
159	160	142	121	67	37	110
147	174	143	115	105	0	3
162	215	197	181	180	177	133

The output(R,G,B) is:

R =

0	0	0	0	0	0	0
238	0	242	0	238	0	242
0	0	0	0	0	0	0
226	0	226	0	210	0	241
0	0	0	0	0	0	0
233	0	218	0	232	0	178
0	0	0	0	0	0	0

G =

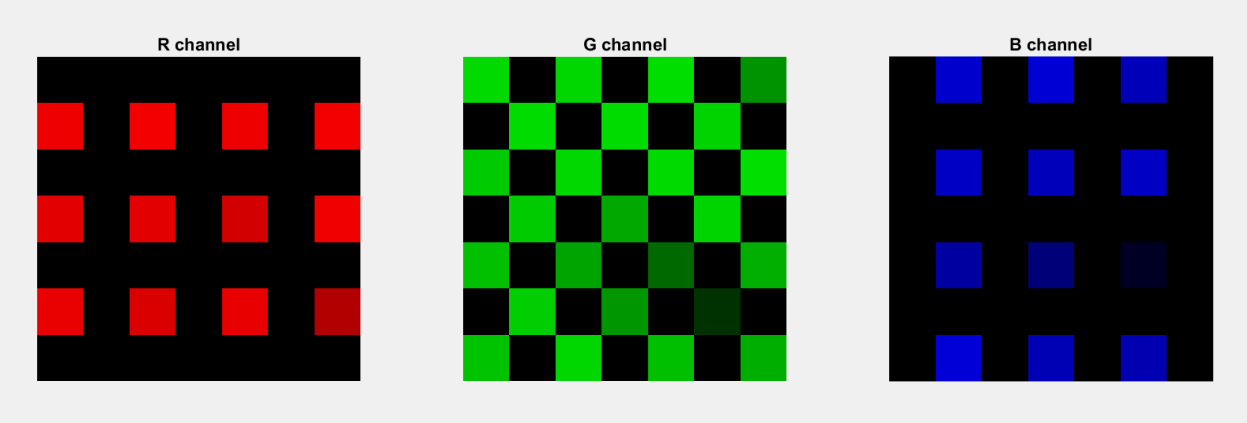
218	0	214	0	222	0	147
0	220	0	220	0	211	0
203	0	216	0	218	0	223
0	203	0	168	0	212	0
192	0	164	0	105	0	175
0	206	0	150	0	49	0
195	0	215	0	190	0	174

B =

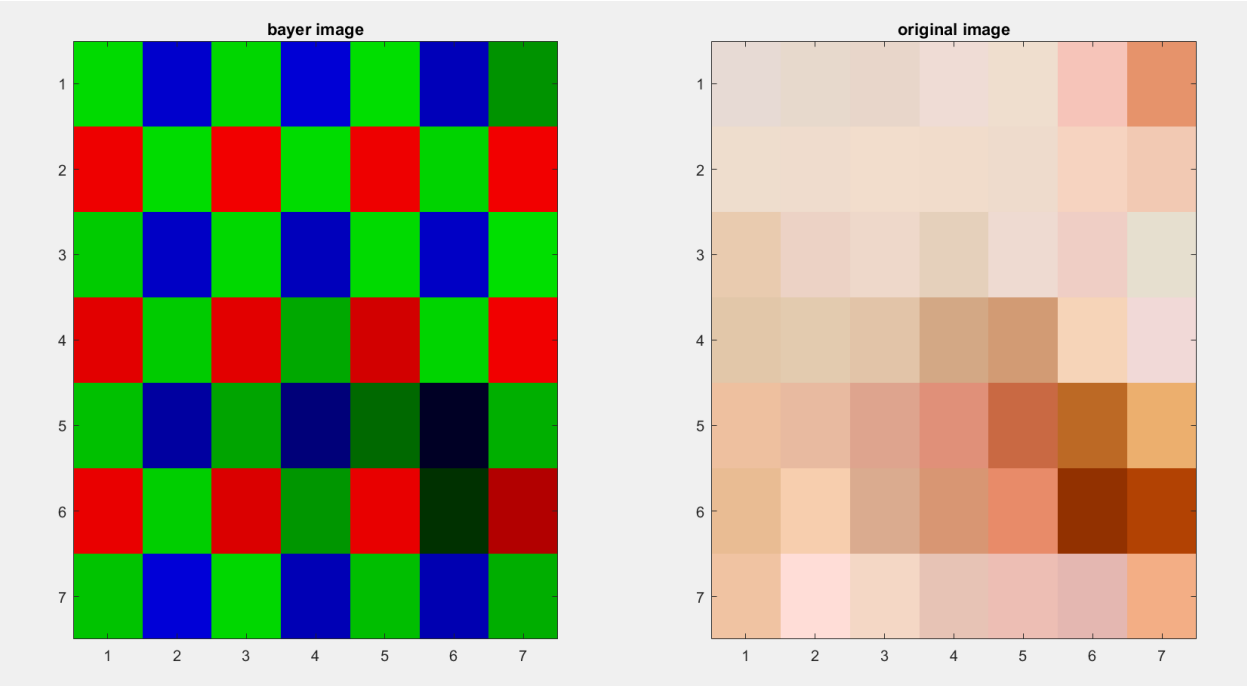
0	204	0	213	0	185	0
0	0	0	0	0	0	0
0	197	0	187	0	197	0
0	0	0	0	0	0	0
0	160	0	121	0	37	0
0	0	0	0	0	0	0
0	215	0	181	0	177	0

The value in specified positions are extracted out as shown, so it is sampled correctly.

Or, we can show the graph. These are the separate channels:



This is the comparison between the bayer pattern and the original image



So it suggests that we sampled correctly.

(b)

The filter used for R and B channels is: [0.25 0.5 0.25; 0.5 1 0.5; 0.25 0.5 0.25], or in matrix form,

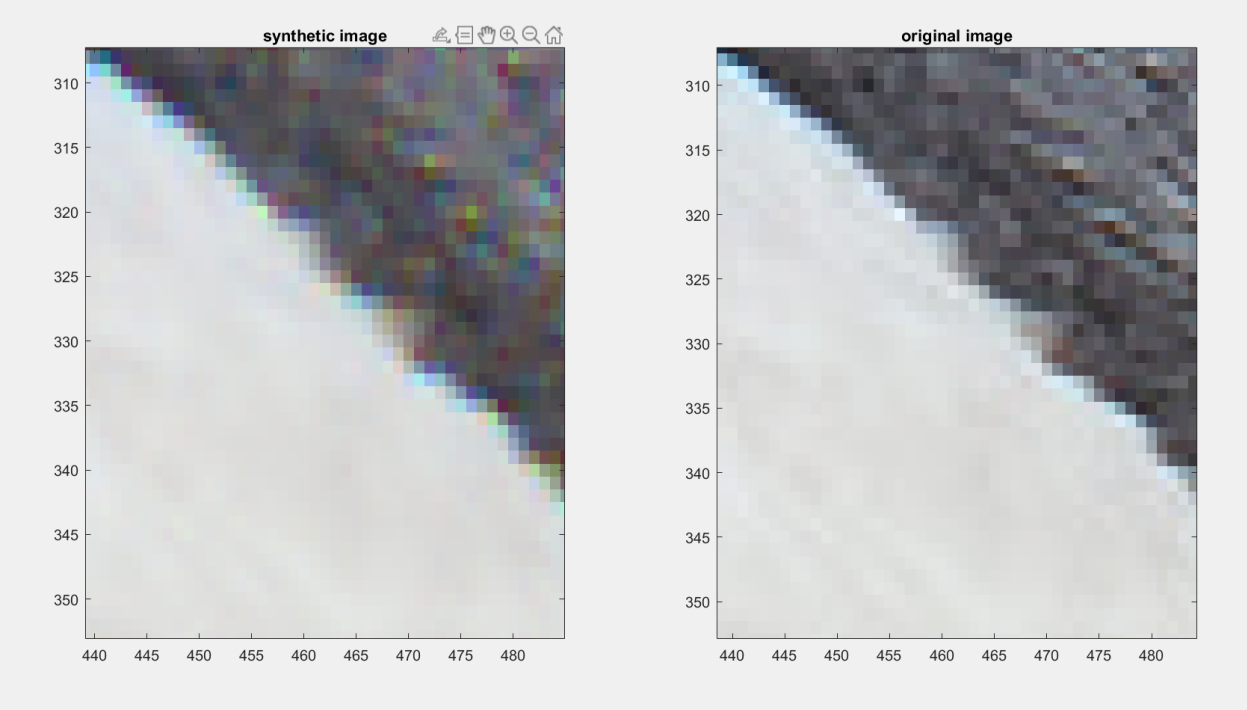
```
0.2500    0.5000    0.2500
0.5000    1.0000    0.5000
0.2500    0.5000    0.2500
```

The filter used for G channel is: [0 0.25 0; 0.25 1 0.25; 0 0.25 0], or in matrix form,

```
         0    0.2500         0
    0.2500    1.0000    0.2500
         0    0.2500         0
```

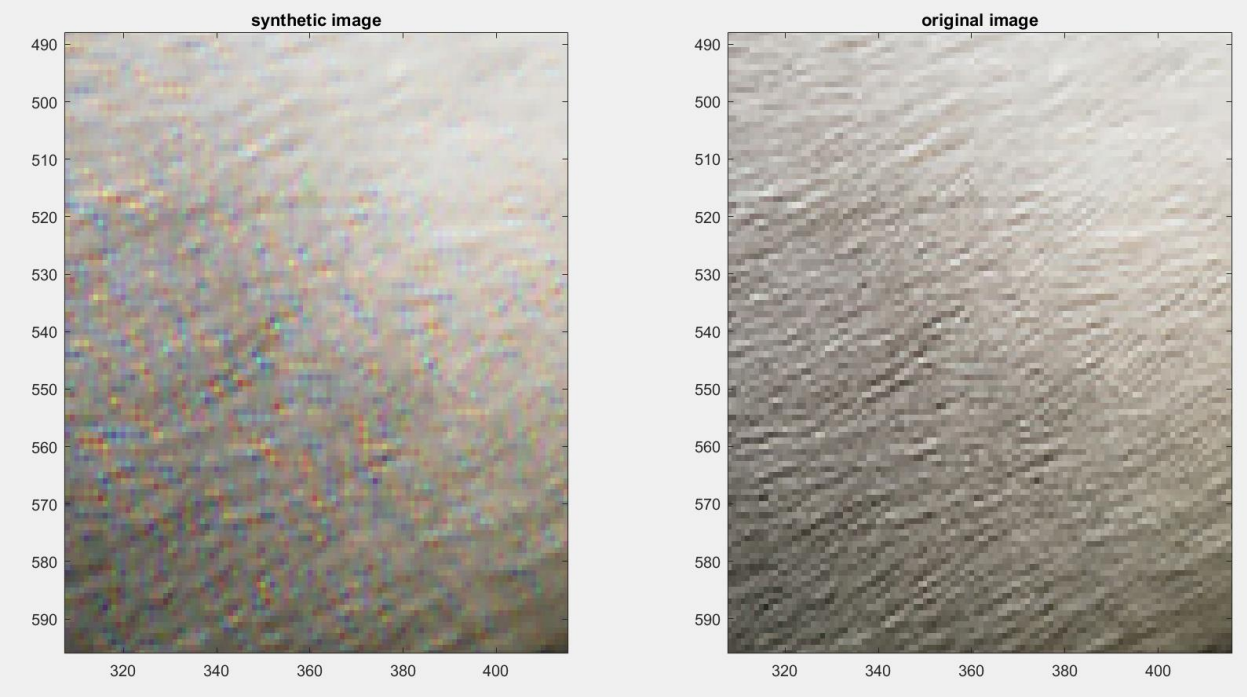
(c)

If we look at the picture in a whole, there is no difference for human eyes. But if we zoom in, there are some differences we can see as well. In this above picture, the left image is the synthetic image; the right one is the original image, and they are pretty much in the same place if not exactly.



We can see in the bottom left part, the slight strip in the original image is clearer than synthetized one: the strips in the synthetized one looks blurred. Similarly for top right part, There are lots of “grey-like” lines going to bottom right, but in the synthetized image it only looks like a block of greys. The main difference is at the sharp intensity change(from white to black). The boundary in the original image is clear, especially in bottom right and top left parts, with a width about 2 pixels. However, in the synthetized image, the boundary is hard to tell if we must specify particular pixels. The width of the boundary is about 3 to 4 pixels. As for the middle part, for some reason it lacks some black in areas should be “black area”, but the boundary in original image is much clearer than the synthetized one, which contains some blurred greys. As a result, since there are less information provided for the

synthetized image, we approximate using linear interpolation; it is okay in most areas, but in boundary areas, the boundary is less obvious and often blurred than the original boundary of sharp intensities.(The boundary is derived from estimation, not real observation)



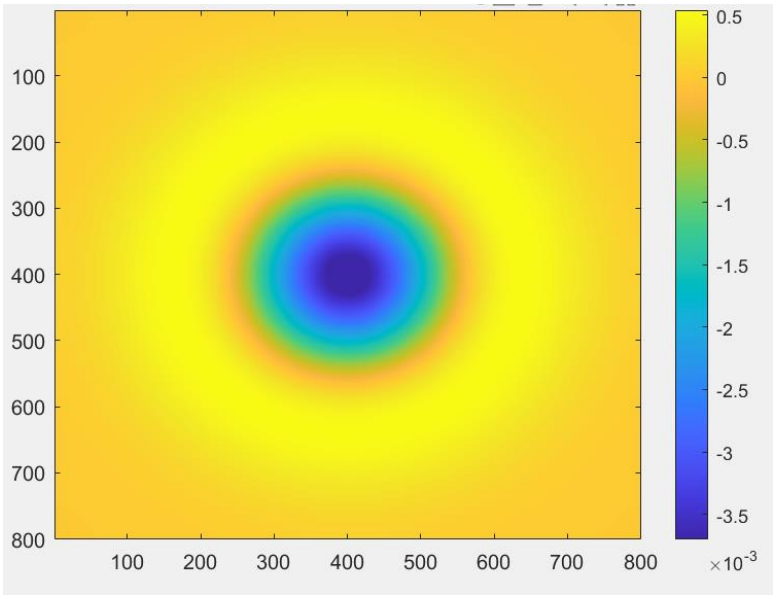
However, in another spot having a general top down change in intensity, we see not only the left image is much blurry (it is more obvious in a picture with lots of details), but there is a significant noise that eye can see. If we see closely in the synthetized image, in middle left part there is some cyan points and some parallel “from top left to bottom down” strips that do not belong to the image at all (if we compare in the original image, there are no such strips), and the original strip (positively sloped) is nearly undetectable in the synthetized image. As a result, in the synthetized image, the noise may be exaggerated in a full detail area with a general intensity change. In this case, there are white and black back and forth in a small area, so the interpolation of some pixels will be awkward, because interpolation is an estimation based on that the image is relatively smooth, but in this case it is not, so we may have some colors that do not appear in original image at all.

2.

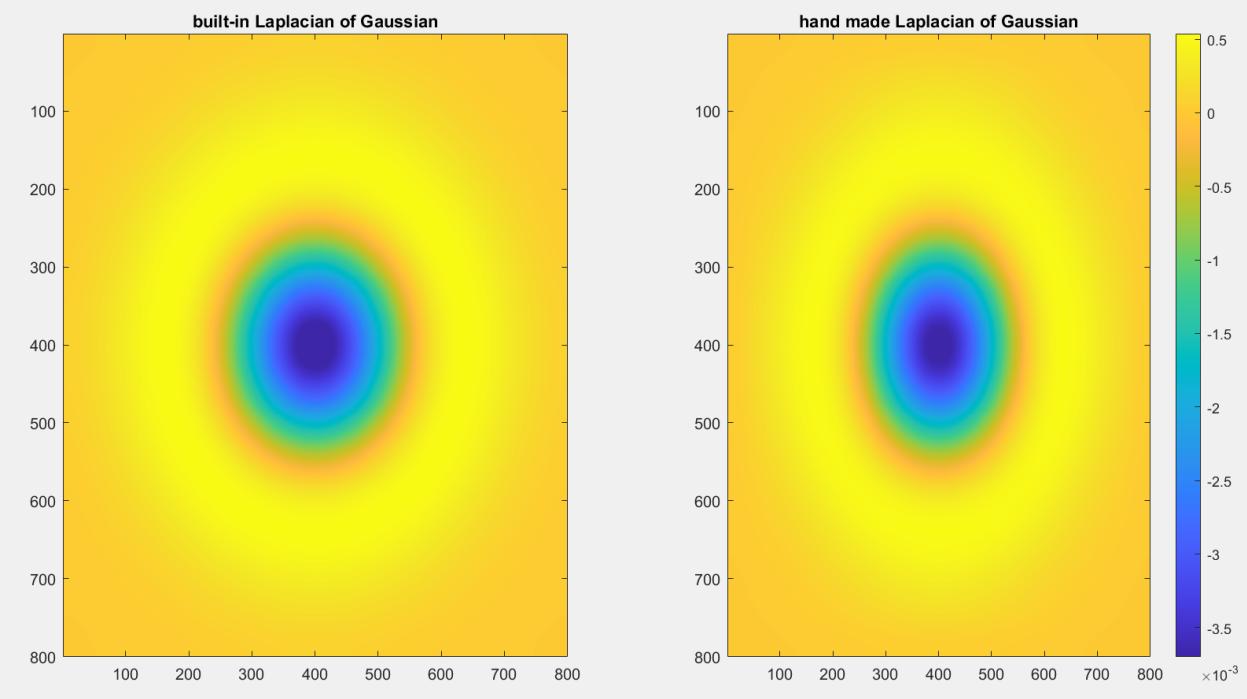
(a)

The image is:

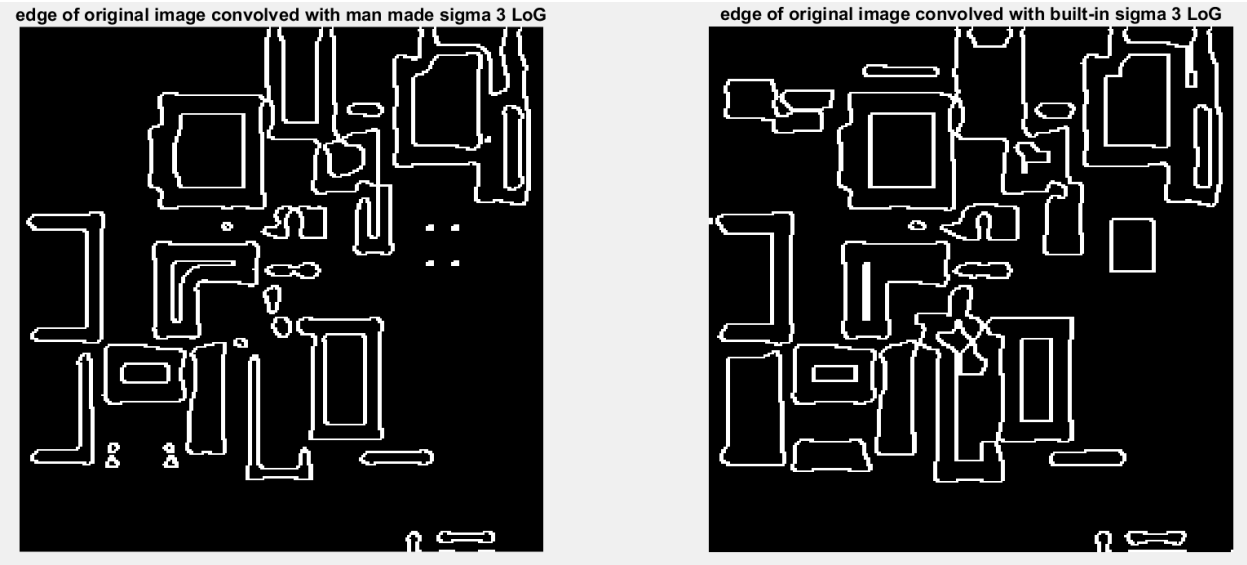
(b)



According to “Common Doubts”, we can try to man made a Laplacian of Gaussian. It is like below, and the comparison between edge detections by two LoGs is one picture below:

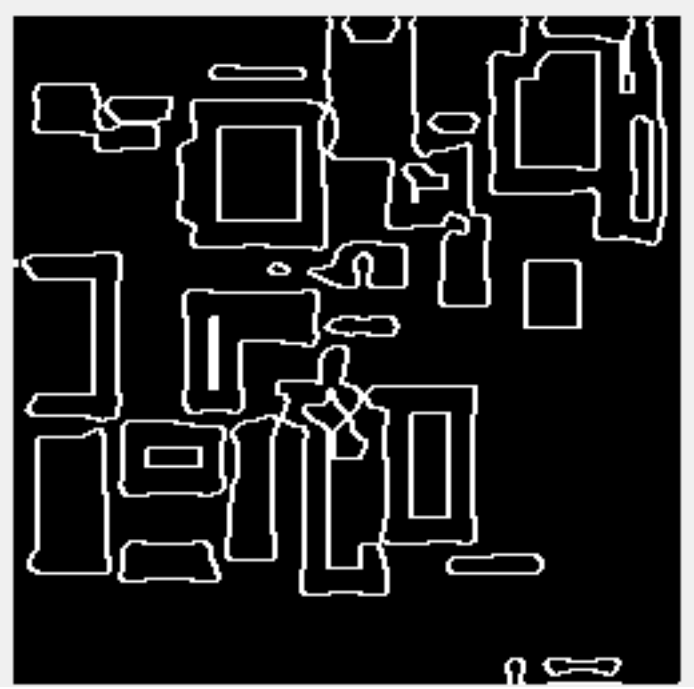


We see some minor difference in blue areas. The comparison of two results of edge detections:

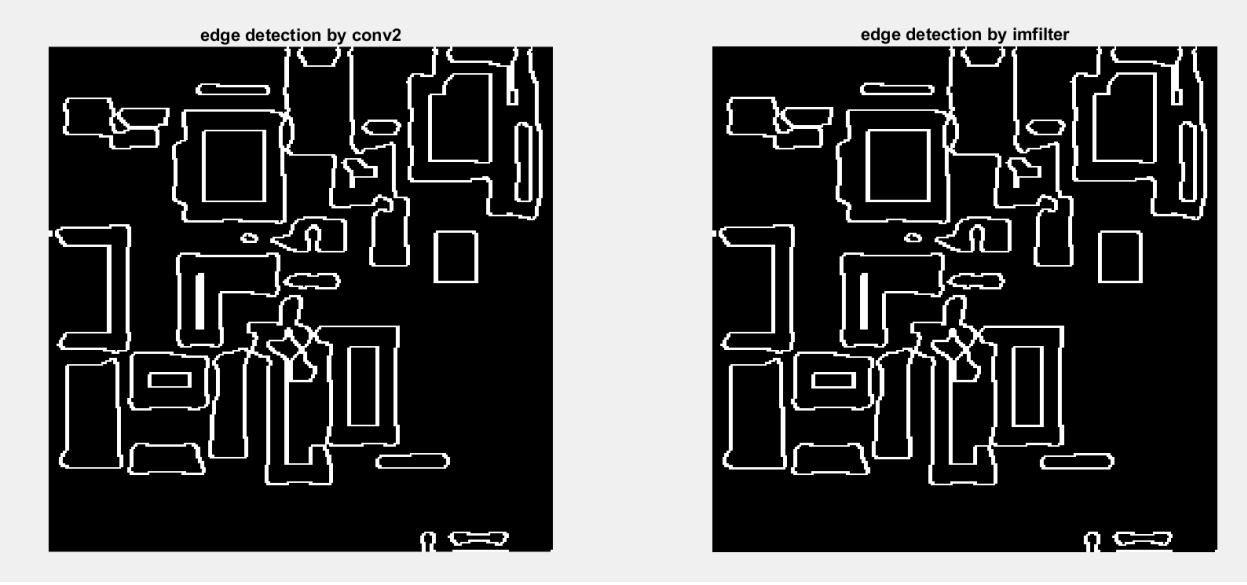


With the built-in method detecting more shapes(edges), the built-in LoG method wins. We will continue to use built-in LoG.

(c)



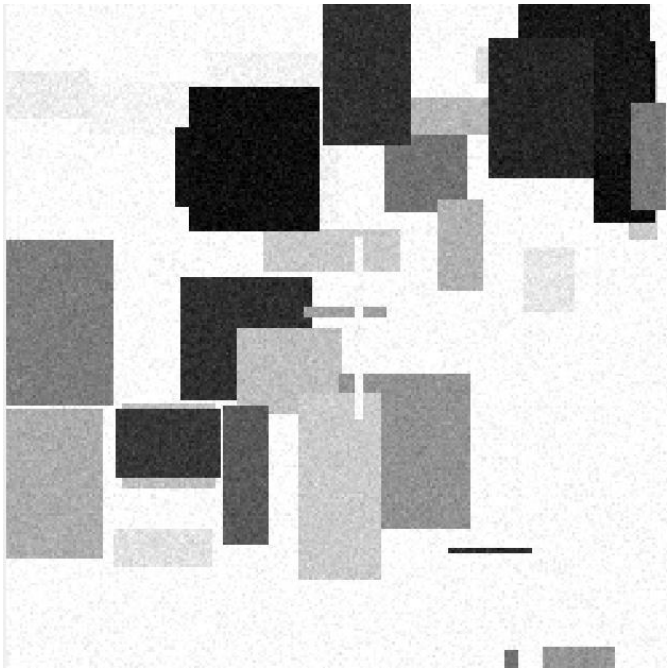
According to “Common Doubts”, we can try to use another method, imfilter. The comparison is like this:



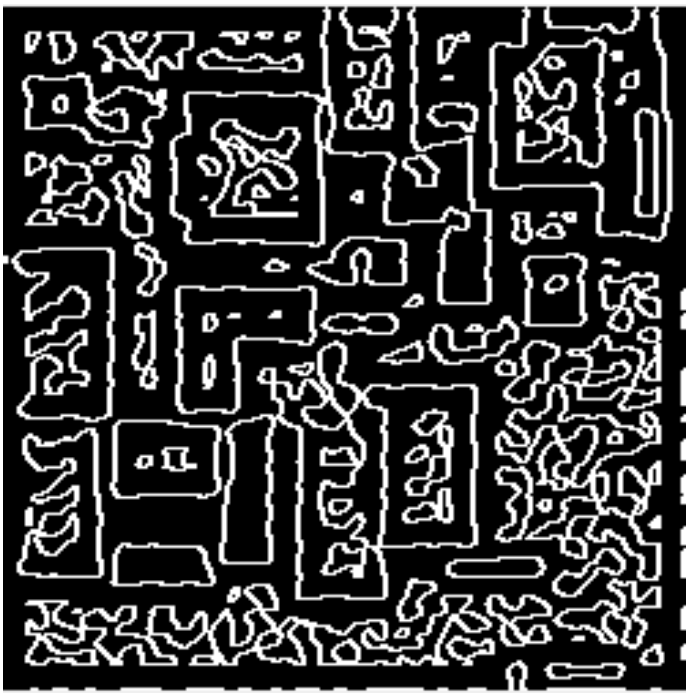
The difference is insignificant. We will continue using conv2.

(d)

Noisy image:

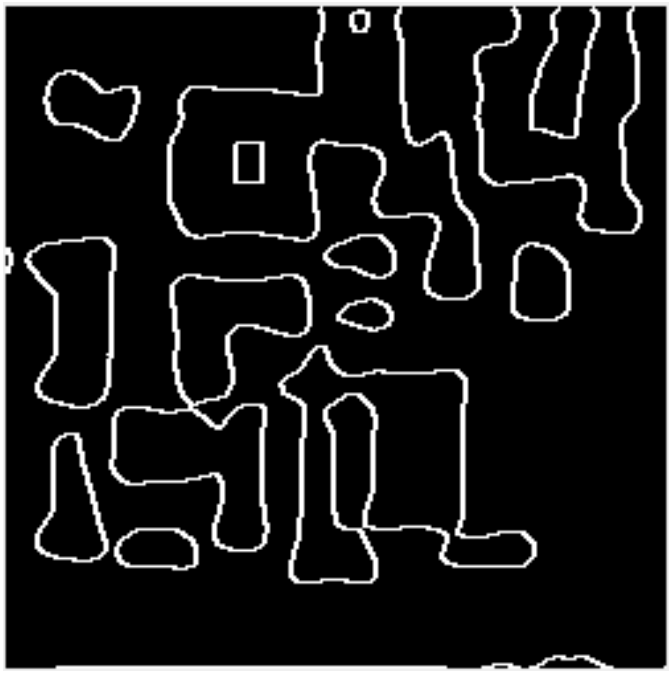


Zero crossings: (intensity is clipped)

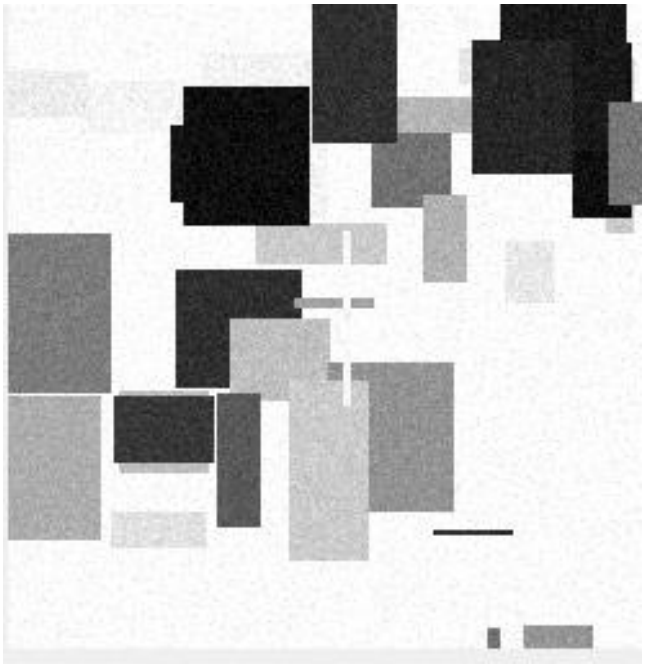


(e)

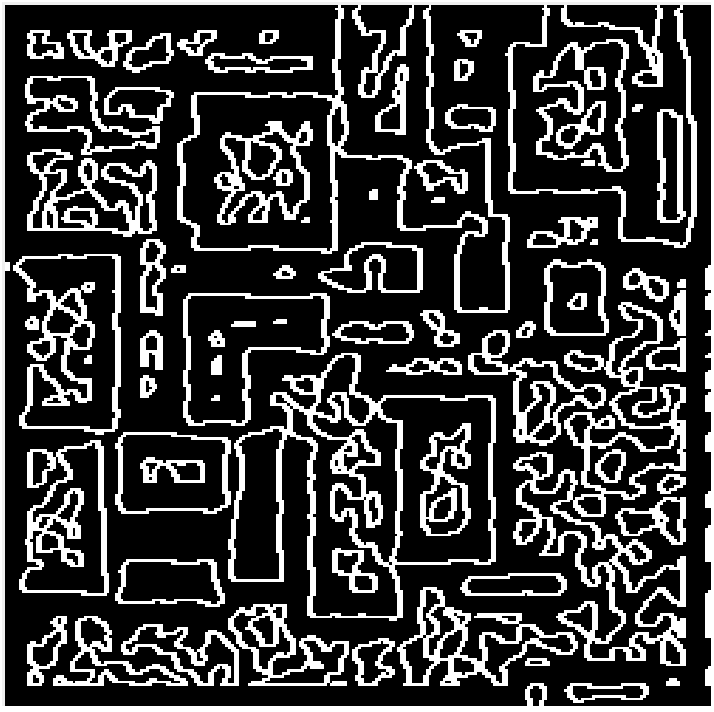
Zero crossing (by a new Gaussian) is:

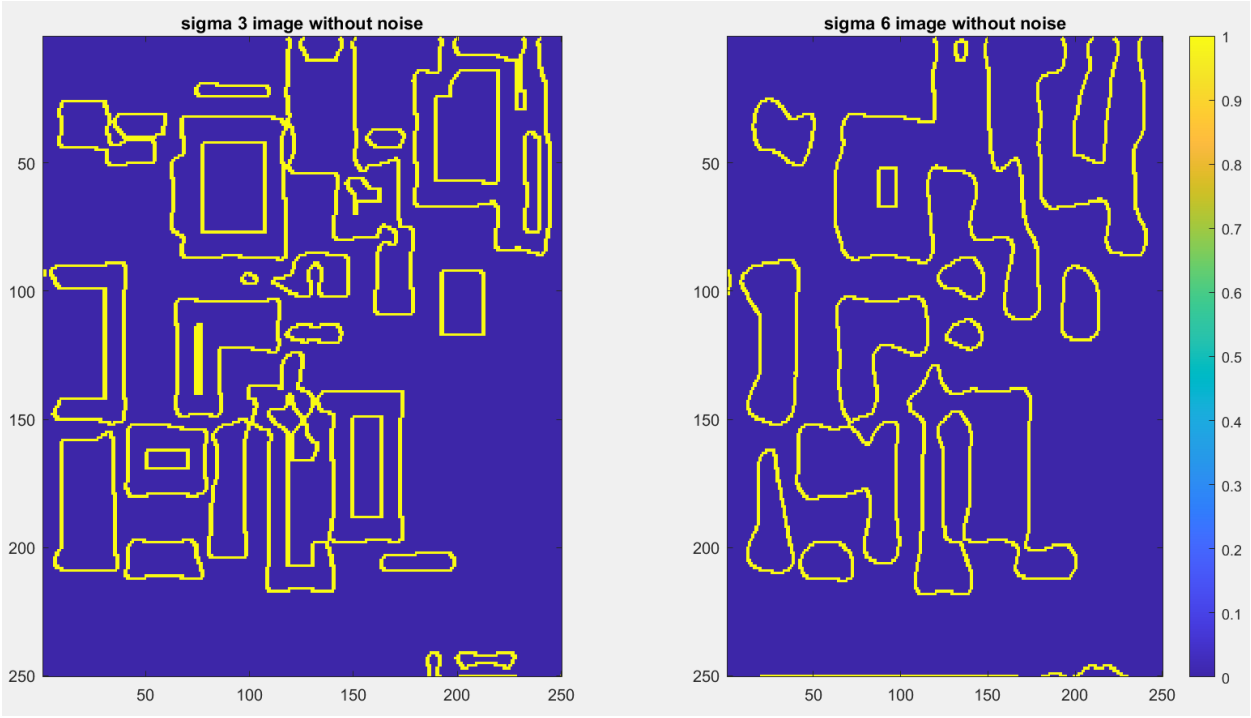


The noisy image by the new image is:



The zero crossings of this image is

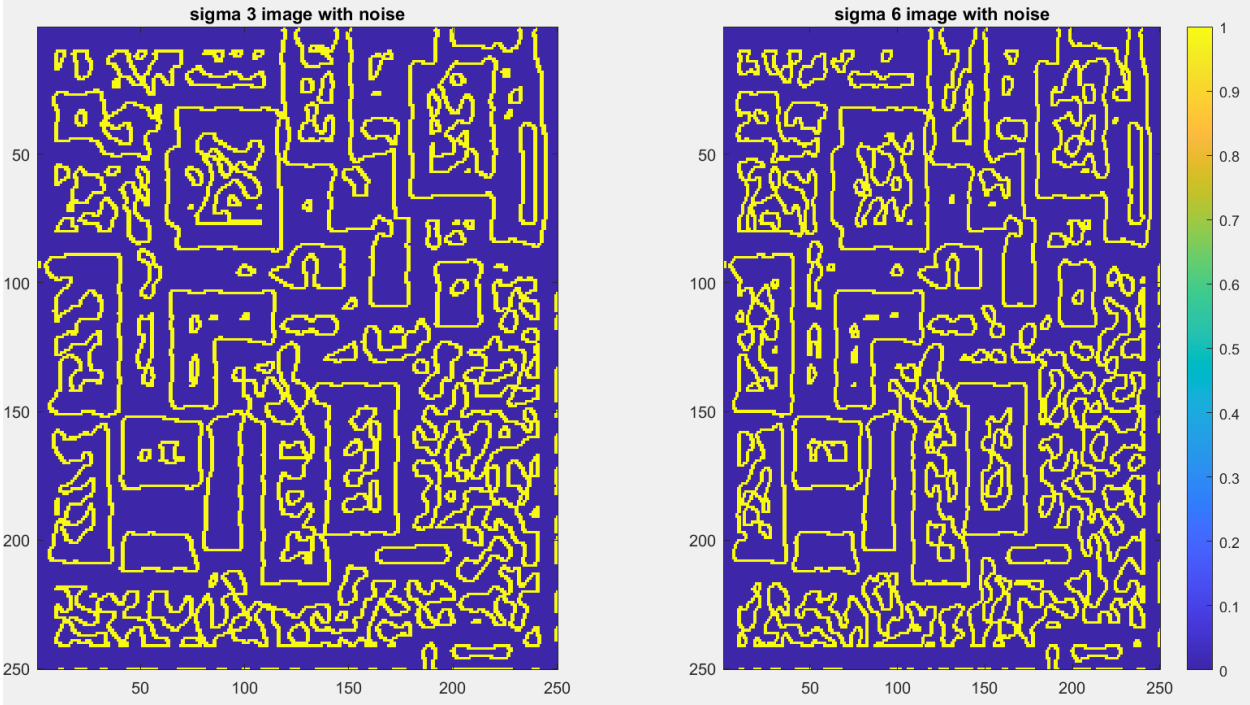




Discuss:

The above two pictures are edges without noise(convoluted with different Laplacian of Gaussian) (We show it in a colorful way if we want to look at it clearly, but it is essentially the same as shown in

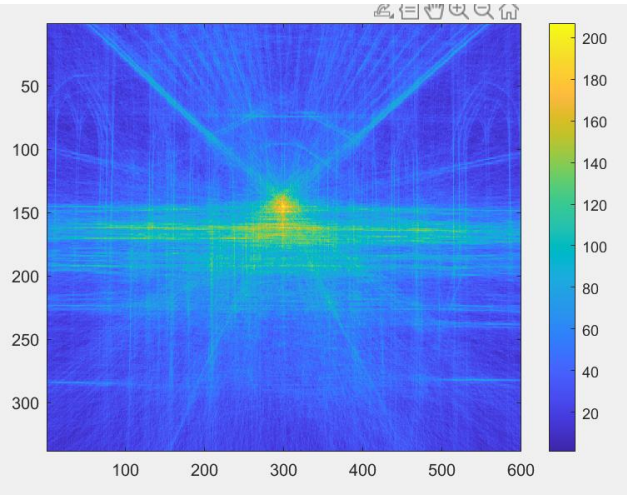
previous sections) (You can see the colorbar, it is not from 0 to 255, that's because matlab automatically make the range 0 to 1 stretched to 0 to 255, so the showings are the same) (every intensity in Q2, including the standard deviation, is done in range 0 to 1, so the noise is in proper scale). The left picture is convoluted with a Laplacian of Gaussian with a smaller std; you can see that the edges of rectangles are quite strange, but still much better than the one with a bigger std, on the right. At least many sides of "rectangles" on the left are straight, but much less of sides are straight in the right image. As a result, we say the increase in sigma and width makes lines to be curvier(the curvy begins from corner of rectangle, because Laplacian of Gaussian is radially symmetric, the corner becomes a curve, and more blurring makes the rectangle curvier). What's more, there are less rectangles detected; especially the smaller rectangles are ignored, like the two rectangles in the bottom right. As a result, in short, the shape zero crossings gets blurred from original image as sigma (and width) increases.



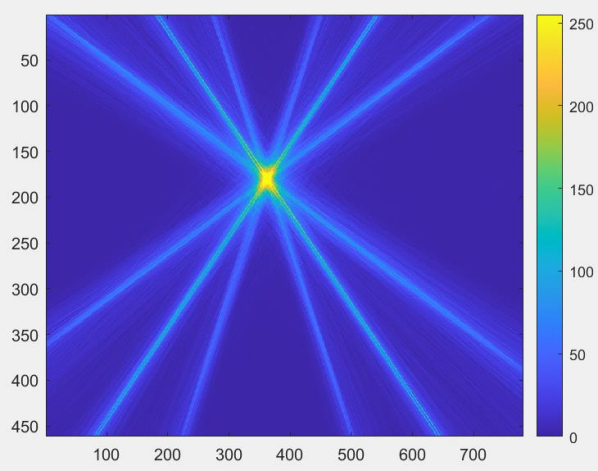
They are images with noise. We can see comparing to previous two pictures, it adds lots of random lines(meaning a zero crossing in its second derivative), but they are not appearing before, so they are made by noise. We can see that in the left picture, we can sort of tell that the picture is about 10 rectangles, similarly in the right image, but the number of these random lines are more for right image, in other words the density of zero crossing is larger(for example if we compare the top left corner). As a result, in practice, there are lots of noise with a standard deviation about 5 intensities, we in general do not use this method, because we will detect lots of "edges" that we do not want at all. As a result, as the sigma(and width) increases, the density of zero crossings increases. As a result, as the sigma (and width) increases, the density of zero crossings increases and the zero crossings are further from original image.

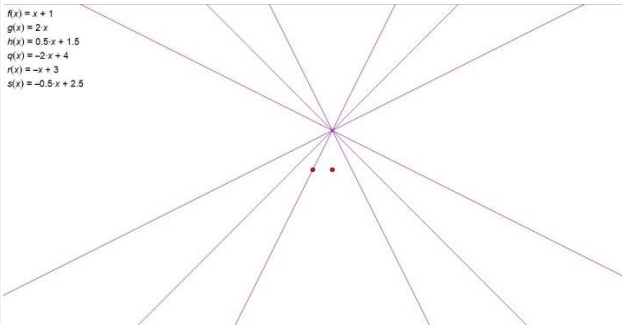
3.

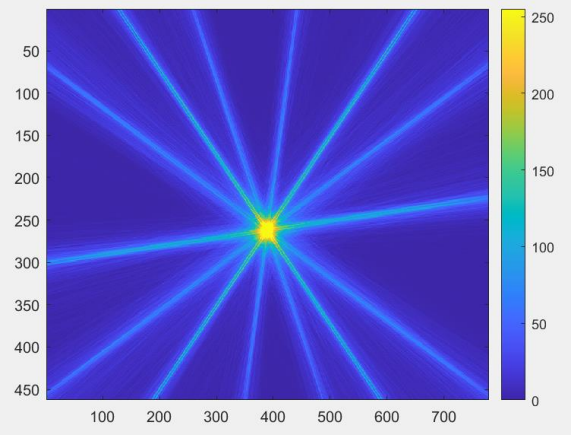
(a) This is a real-life picture (we sort of know the answer)

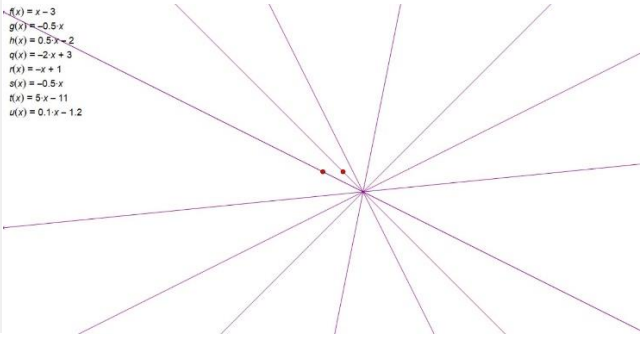


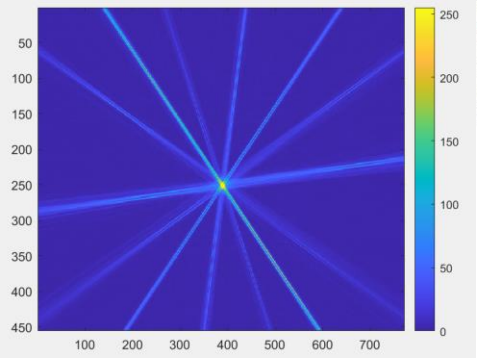
The question asks for examples that I know the answer. The followings are. (draw in mathematics software, and those lines are intersecting at the same point)

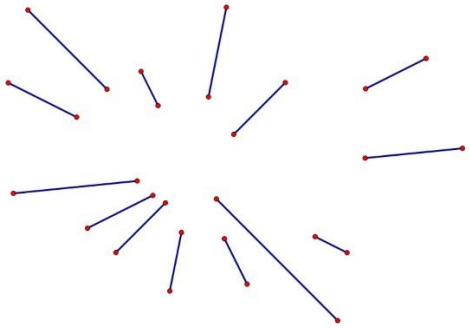












The last image is essentially the same as the second last: it draws some segments in the line in second last picture, so they still intersect at the same point. (Since there is less points, so the intensity at the center is less obvious than second last image, of course.)

(b)

For (a), we basically do this: (simplify that into: one algorithm to all kinds of edges)

For every edge: (E times)

For every N (each discrete point in x): (N times)

Vote one for a particular (x,y) (x is already known) (1 time)

For the new method, it will be

For every edge: (E times)

For every other edge: (E times) (E-1 times to be exact)

Vote one for a particular (x,y) (computed intersection) (1 time)

We assume cost of calculating for one vote is similar, meaning we count both statements in the inner most loop as 1. And the two edges will produce the same intersection, no matter the order, so we need E^2/2 operations. As a result, Hough method depends on number of edge E and size of image (or length) N, but the new method depends on number of edge E only. As for the Hough method, we need EN operations. In terms of big Oh notation, Hough method is O(NE), and the alternative method is O(E^2). As a result, for an image with a great number of edges, Hough method is better; for an image with a relatively small number of edges(E<2N), the new method is better.

However, in practice, the cost for the most inner loop is not the same. For the inner most statement of Hough transform, x is already known, and we have a direct solution for y, which costs about 3 calculations (line 52 of Q3Hough.m: y = round((r-(hedges(i,3)*j)) / hedges(i,4));)

As for the alternative method, we do not know x and y before hand: we only have two lines with indirect knowledge of slope. In order to get x, it needs 9 calculations: (two appearance of cot of same angle needs to calculate once only)

$$x = \frac{y_2 - y_1 + x_1 cot\theta_1 - x_2 cot\theta_2}{cot\theta_1 - cot\theta_2}$$

Similarly, so get y, we need 3 calculations $y = y_1 + cot\theta_1(x - x_1)$

As a result, since we need to calculate for x and y each time, the cost for each vote is 12 calculations, (quantitatively, by a factor of 4 than Hough). As a result, if we take into account of this, Hough needs 3NE calculations, but this alternative method needs 6E^2 calculations, so the equalizer is at N=2E. If N>2E, then the alternative method costs less; if N<2E, Hough transform costs less.