EEE 6512 Image Processing & Computer Vision

Assignment 05

Name: Chen Yang Username: cyang3 UF id: 52109967

7.3

From the book page 330, [1] we got that each reduction by a factor of two is known as an octave, therefore,

a) If the downsampling factor is $\sqrt[n]{2} = 2^{\frac{1}{n}}$, then there are n images per octave [1], in this problem, since n = 3 images per octave, so the downsampling factor is $2^{\frac{1}{3}}$ = 1.2599

b) Since repeated convolutions with a Gaussian are equivalent to a single convolution with a Gaussian whose variance is the sum of the individual variances, we define $\sigma'^2 = \frac{1}{n}\sigma^2$ to ensure that the overall smoothing btw octaves in the same as btw consecutive levels as equation 7.2 [1] (Page 329). So, the variance $\sigma'^2 = \frac{1}{3}\sigma^2 = \frac{1}{3} * 1.2 =$ 0.4

7.4

a) From the book page 332, we know that in order to ensure that each octave is convoluted with the same sequence of variances relative to the image size, the variance ratio should be set the $\rho = 2^{\frac{1}{n}}$ [1], So when n =5, $\rho = 2^{\frac{1}{5}}$ = 1.1487

b) A reasonable choice for the initial variance is $\sigma_0^2 = \frac{1}{n}(0.5)$, therefore,

let $\sigma_0^2 = \frac{0.5}{5}$ = 0.1, $\rho^2 = \left(2^{\frac{1}{5}}\right)^2$=1.3195. Therefore,

$\sigma_1^2 = \rho^2\sigma_0^2$ = 1.3195*0.1 = 0.13195, $\sigma_2^2 =$ 1.3195*0.13195=0.1741

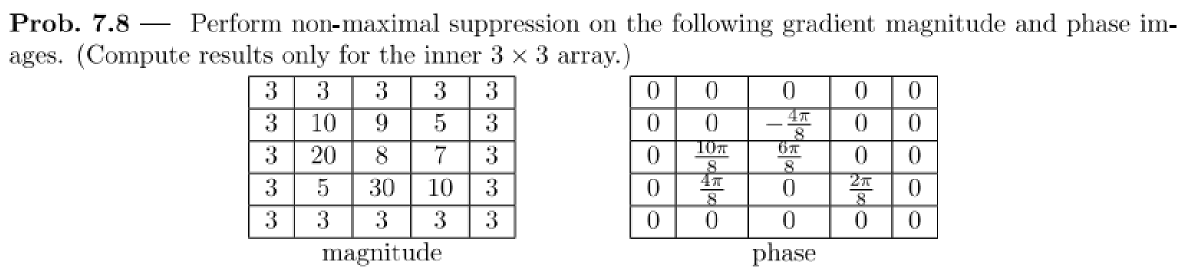
7.5

Causality criterion can ensure that the number of local extrema does not increase as we proceed to coarser levels of scale. That is to say, the maxima are flattened while the minima are raised.[1] So, all local extrema found by the scale space computation are due to the image itself.

7.7

That is because the Canny edge detector is based on an assumption that there is single line within the immediate neighborhood. Therefore, when there is a intersection of two lines, at the point of intersection there is more than one lines. So it fails the assumption. That is why Canny cannot perform well at the intersection of two lines.

7.8



Prob. 7.8 — Perform non-maximal suppression on the following gradient magnitude and phase images. (Compute results only for the inner 3 × 3 array.)

3	3	3	3	3
3	10	9	5	3
3	20	8	7	3
3	5	30	10	3
3	3	3	3	3

magnitude

0	0	0	0	0
0	0	$-\frac{4\pi}{8}$	0	0
0	$\frac{10\pi}{8}$	$\frac{6\pi}{8}$	0	0
0	$\frac{4\pi}{8}$	0	$\frac{2\pi}{8}$	0
0	0	0	0	0

phase

The nine values of the image can be analyzed as follows:
10: angle is 0, horizontal (red), [3 10 9], local maximum 10, is the local max
9: angle is -4pi/8 = -pi/2 → -pi/2 +2pi = 3pi/2, vertical (blue), [3 9 8], is the local max
5: angle is 0, horizontal (red), [9 5 3], NOT the local maximum
20: angle is 5pi/4, down-right (yellow), [3 20 30], NOT the local maximum
8: angle is 3pi/4, down-left (green), [5 8 5], is the local maximum
7: angle is 0, horizontal (red), [8 7 3], NOT the local maximum
5: angle is pi/2, vertical (blue), [20 5 3], NOT the local maximum
30: angle is 0, horizontal (red), [5 30 10], is the local maximum
10: angle is pi/4, down-right (yellow), [8 10 3], is the local maximum
Therefore, based on the conclusion above, we can draw the final result as follows:

$$\begin{bmatrix} 10 & 9 & 0 \\ 0 & 8 & 0 \\ 0 & 30 & 10 \end{bmatrix}$$

7.9
The location detection trade-off is a dilemma, which is about how to pick a good filter for computing the gradient or if the Gaussian derivative is used, what value to choose for the standard deviation. As it turns out, a large sigma value will lead to a better SNR, but a smaller sigma value will lead to a more accurate location for the edges. So, a large region of support will have a high detection rate but low localization accuracy. A small region, will have a good performance on detect the pixels near the edge, but low detection rate. That is the most important choice btw localization and detection accuracy we have to pick in the Canny edge detector.

7.10
The main disadvantage is that the Marr_Hildreth operator is isotropic, which means it will smooth the image across the edges and also along those edges. It will reduce the accuracy of locating the edges.

7.14
That is because Harris corner detector is isotropic, but Moravec is unisotropic.

7.15
Non-maximum suppression only preserves those pixels with a large cornerness measure compared with their neighbors. Pixels with small cornerness are generally not important in this process. There are two method to modify this algorithm: 1) Ignoring the direction, because direction information is not available 2) Increasing the window size within those pixels that are compared, this method is optional. But it is really useful to reduce the number of pixels retained by the computation.

7.18

Harris:

$$Z = \Sigma_{x\in R} w(x) \begin{bmatrix} I_x^2(x) & Ix(x)I_y(x) \\ Ix(x)I_y(y) & I_y^2(x) \end{bmatrix} = \begin{bmatrix} z_x & z_{xy} \\ z_{xy} & z_y \end{bmatrix}$$

Z_x = 25+81+25+49+9+64+36+81+9=379

Z_y = 4+49+36+1+64+81+25+4+9=273

Z_{xy} = -5*2 +(-7)*(-9)+5*(-6)+7*(-1)+3*8+(-8)*9+(-6)*(-5)+9*2+3*3=25

So, $Z = \begin{bmatrix} 379 & 25 \\ 25 & 273 \end{bmatrix}$

Cornerness = det(z) – R(trace(z))^2, where R = 0.04[1]

= 379*273-25^2 – 0.04*(379+273)^2

= 85837.84 = 85838

Tomasi-Komade:

Cornerness = min({λ_1, λ_2}) = λ_2 （equation7.33）

$\lambda_2 = \frac{1}{2}((z_x + z_y) - \sqrt{(z_x - z_y)^2 + 4z_{xy}^2}\ )$ （equation7.34）

$= \frac{1}{2}$ (652-117.2)

= 267.4

Required Images:

myCannyEdgeDetector.m

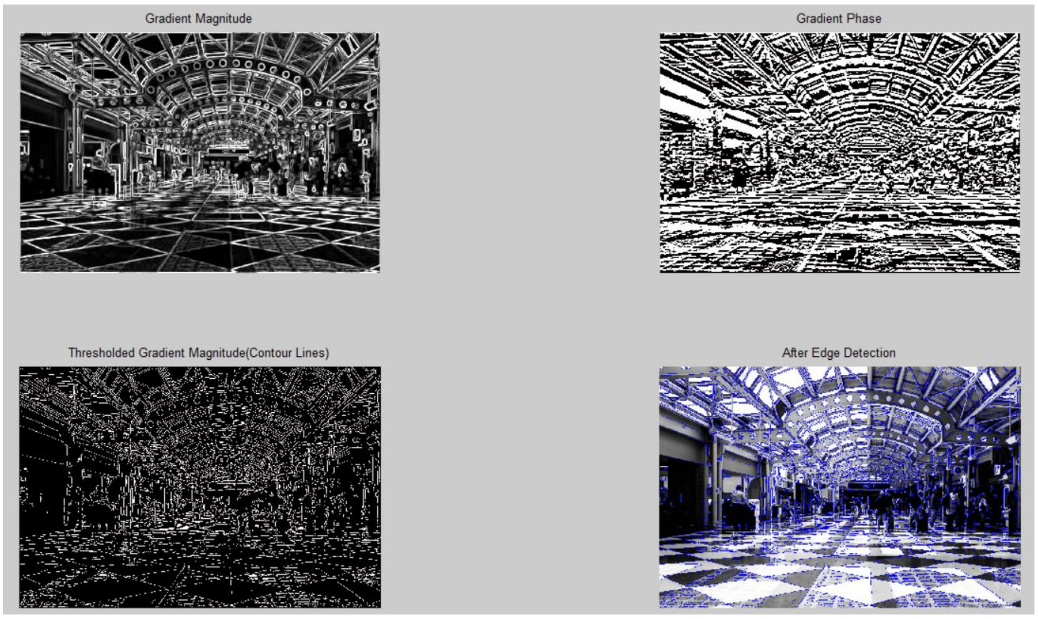


Figure. Four intermediate images for img1.pgm

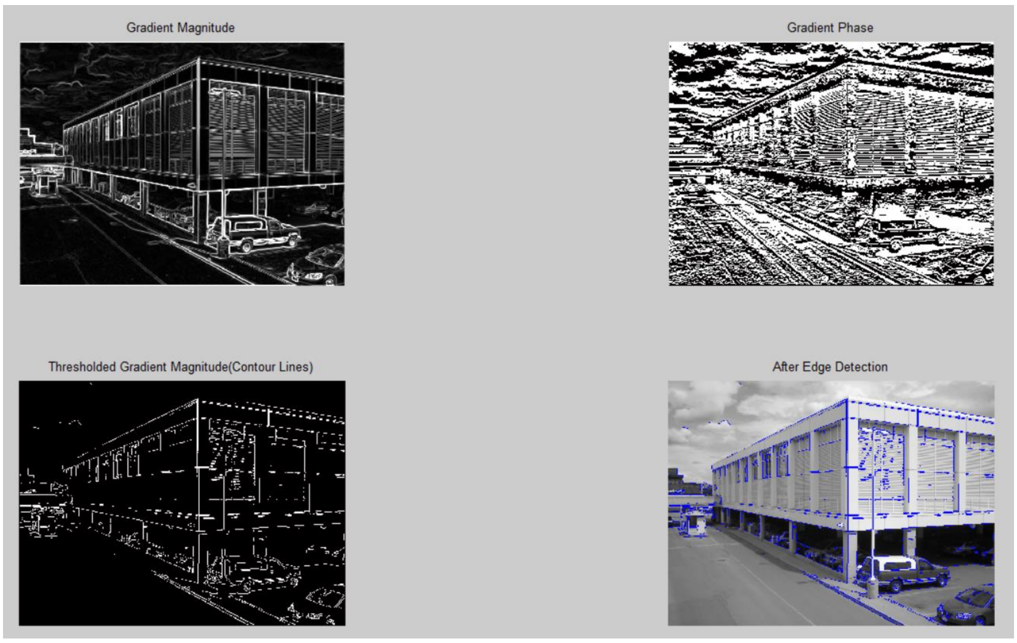


Figure. Four intermediate images for img2.pgm

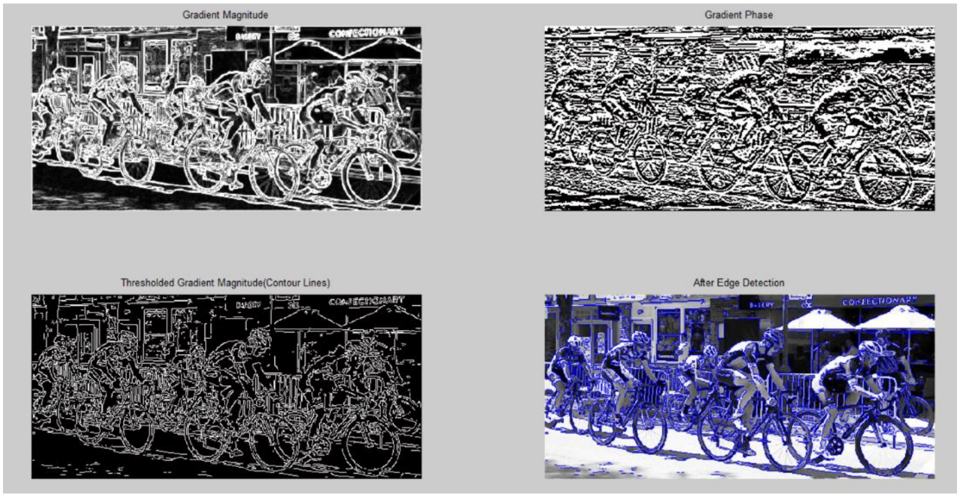


Figure. Four intermediate images for img3.pgm

myHarrisCornerDetector.m

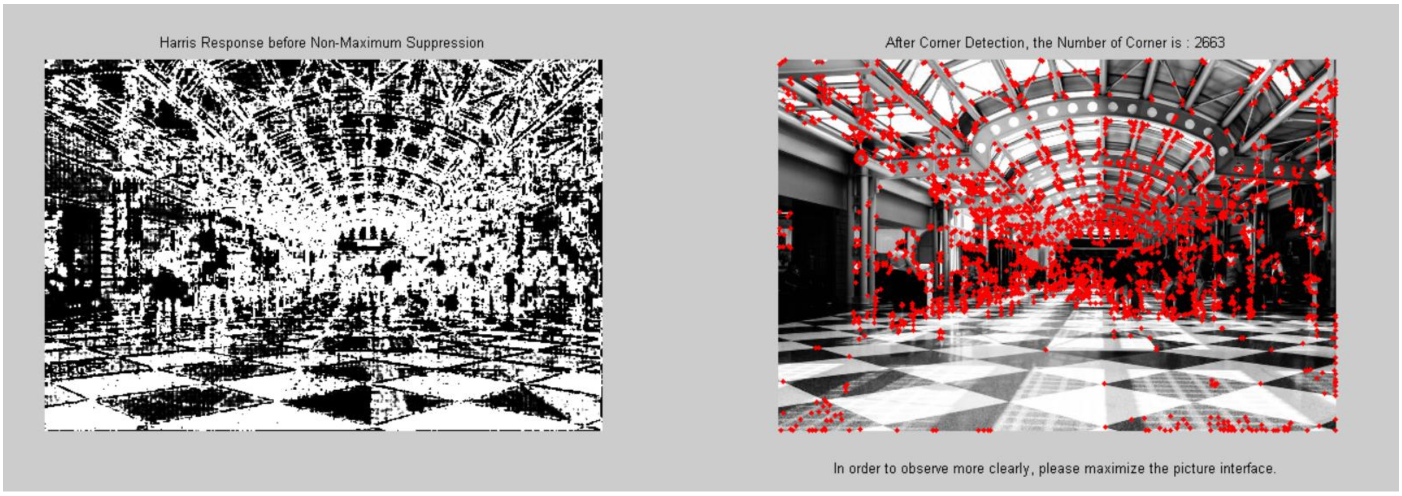

Harris Response before Non-Maximum Suppression
After Corner Detection, the Number of Corner is : 2663
In order to observe more clearly, please maximize the picture interface.

Figure. Two intermediate images for img1.pgm

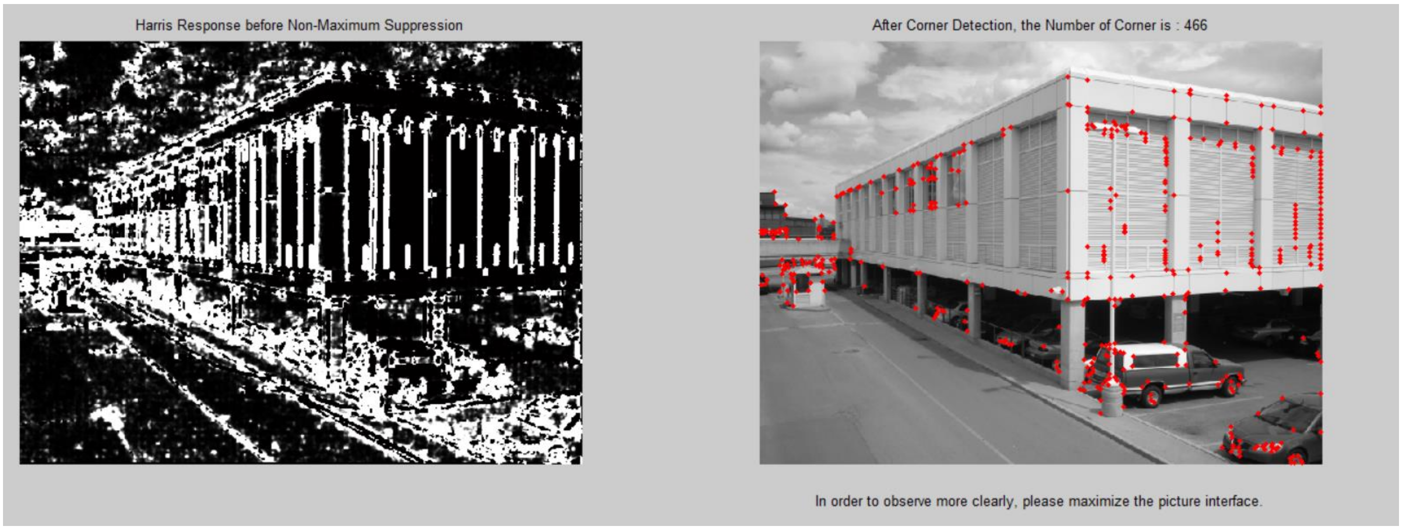

Harris Response before Non-Maximum Suppression
After Corner Detection, the Number of Corner is : 466
In order to observe more clearly, please maximize the picture interface.

Figure. Two intermediate images for img2.pgm

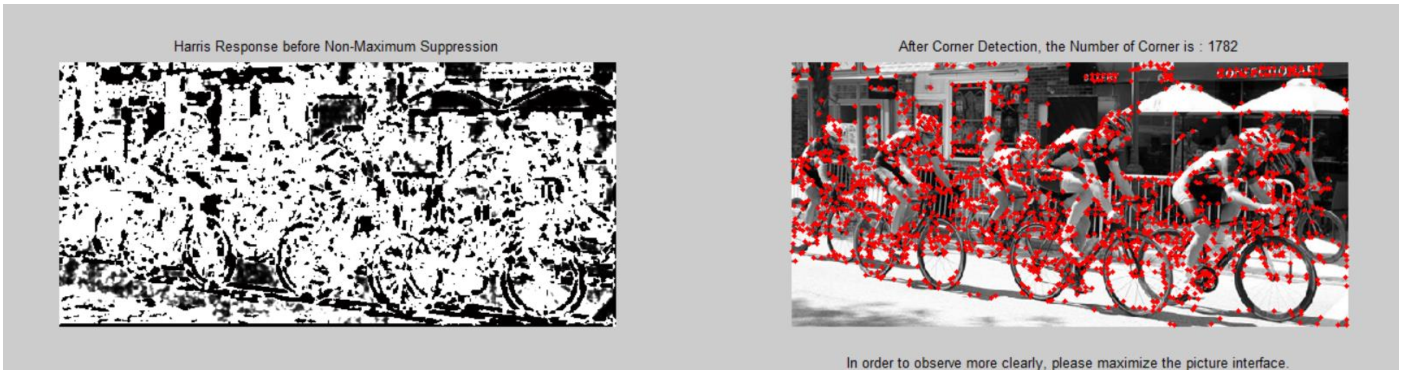

Harris Response before Non-Maximum Suppression
After Corner Detection, the Number of Corner is : 1782
In order to observe more clearly, please maximize the picture interface.

Figure. Two intermediate images for img3.pgm

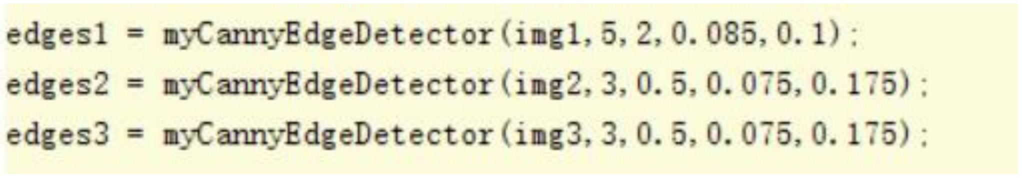
Note: In order to make the output pseudocolor image more clear to view, I use matlab tool to mark those corner points with red points

Questions：

myCannyEdgeDetector.m

What input parameters does the program accept (besides the input image) and what purpose does each serve in the algorithm?
My function will accept input as follows:
img: a grayscale image, do edge detection on this img
k: the size of Gaussian kernel to smooth the img
sigma: the parameter of Gaussian kernel to smooth the img
T_low: after non max suppression, if the points' gradient value is lower than this value, just ignore them
T_high: after non max suppression, if the points' gradient value is higher than this value, regard those points as important edges. If the value is in the range of there two threshold value, use 8 neighbor edge linking method, to decide if the edges will be ignored or picked
Which method did you use to compute the gradient and why did you choose this method over others?
Before computing the gradient of the image, I use a Gaussian kernel to smooth the original image, (to throw out those noises), then I use the smoothed image to convolve with sobel kernel to compute the img's vertical and horizontal gradient respectively. I pick sobel because its performance is better than prewitt, and it is also one of the gradient computing masks. Sobel is more robust.

Looking at the results of the given images, where does your algorithm perform well and why? Where does your algorithm perform poorly and why?
After try different threshold values, finally I decide to use those following parameters for this algorithm. See the following figure. As we can see , in the img1.pgm, nearly all the edges are founded by the algorithm, which proves the ability of the method. However, it also has drawbacks, especially for the second and the third images, if we try to find all the edges, some of the points will not be linked well. If we try to adjust the lower bound up and adjust the upper bound up, we can have a smooth contour of the images, but some of the edges are not marked in the result image. The reason is that in different images, we usually focus on different target, we cannot rely on algorithms can only find the important information that we need. We should adjust the input parameters according to different images and conditions. For example, in the img1.pgm, let's assume the lattice pattern of floor is not a important clue, we wanna find the edges of the ceil and pillar, and adjust the parameters, the edges of floor will show up inevitably.



```
edges1 = myCannyEdgeDetector(img1,5,2,0.085,0.1);
edges2 = myCannyEdgeDetector(img2,3,0.5,0.075,0.175);
edges3 = myCannyEdgeDetector(img3,3,0.5,0.075,0.175);
```

How could your algorithm be improved?
We can use this algorithm after image preprocessing, and add some adaptive parameters into this function, which can detect the input images' information, such as the whole contrast, the important area (that we want to focus on). Or we can just use another better Gaussian filter to make the features of images more clear and viewable.

myHarrisCornerDetector.m

What input parameters does the program accept (besides the input image) and what purpose does each serve in the algorithm?

In this function, the input parameters are listed as follows:

img: a grayscale image, do corner detection on this img

k: the size of Gaussian kernel to smooth the img

sigma: the parameter of Gaussian kernel to smooth the img

a: the constant k in the textbook's harris cornerness computation equation, which is recommended to be set as 0.04

Looking at the results of the given images, where does your algorithm perform well and why? Where does your algorithm perform poorly and why?

The advantages of the algorithm is that it will find a great number of corner points of images. Take the img1.pgm as an example, it can clearly mark the corner points of the McDon's logo, the pillar, the floor pattern and even the strong light shadow. However, some of the edges in the detail cannot be found, which is the main drawback. For example, in the second image, the left side of the building, there are many corners. But the algorithm can only mark a part of them. That is because when the gaussian filter smoothing the images, it reduce the noises of the image and it also eliminates some details. Or we can also blame the drawback on the choosing of parameters.

How could your algorithm be improved?

In this algorithm, I found that if the luminioius condition of the images is way too contrasted. The algorithm cannt catch all the corners properly. For example, in the second image, the first floor of building (parking lot) is very dark and the second and other floors are way too bright, in this situation, the choice of paremeters is very difficult. If we want to catch all the corners, we have to pick a small gaussian filter to keep details as much as possible, even if we do so, the details of dark regions will still be throwed. So, if possible , we can split the input images into different section according to different lumination. And then process the image with different parameters. This method will greatly improve the algorithm's performance.

Reference

[1] B. H. Brown, R. H. Smallwood, D. C. Barber, P. V Lawford, and D. R. Hose, *Image processing and analysis*. 2004.