CSci 5561
Computer Vision

Programming assignment 2 report

Spring Semester 2016

Xiao Lu

ID：5079620

Overview Structure and Code Files

1. Algorithms and code files

   1.1 Hough transform for Line detection

   MainLine.m

   RobertsEdge.m

   OTSUThreshold.m

   HoughTransfromLine.m

   HoughLinePeak.m

   1.2 Hough trandform for Ellipse detection

   MainEllipse.m

   RobertsEdge.m

   OTSUThreshold.m

   HoughTransfromEllipse.m

   DrawEllipse.m

2. Test images

   2.1 *line1.jpg*

   2.2 *ellipse1.jpg*

Part 1 Hough Transform for Detecting Lines

1.1 Basic concepts

The Hough transform is a technique which can be used to isolate features of a particular shape within an image [1]. The simplest case of Hough transform is detecting straight lines. For computational reasons, we represent straight line by Hesse normal form as follows [2],

$$r = xcos\theta + ysin\theta \qquad (1\text{-}1)$$

Where r is the distance from the origin to the closest point on the straight line, and θ is the angle between the x axis and the line connecting the origin with that closest point.

Hence, given a single point in the plane, then the set of all straight lines going through that point corresponds to a sinusoidal curve in the (r, θ) plane, which is unique to that point. A set of two or more points that form a straight line will produce sinusoids which cross at the (r, θ) for that line. Thus, the problem of detecting straight lines can be converted to the problem of finding concurrent curves.

1.2 Implementation

Before implement Hough transform, we should get the thinned edge picture, this is to say we should implement Edge detection and Edge thresholding ahead. We already implemented these algorithms in programming assignment 1, so I will use them directly from programming assignment 1.

Then, for linear Hough transform, we uses a two-dimensional matrix, called Accumulator, to detect the line. The dimension of Accumulator should be D_nums-by-A_nums. Here, D_nums = 2*Distance/Distance resolution, A_nums = 180°/Angle resolution. After we get Accumulator matrix, we can get the histogram of final Hough matrix by hist() function. The dimension of it should be P_nums-by-180, Where P_nums = Number of edge points. Each element of the matrix has a value equal to the sum of the points or pixels that are positioned on the line represented by quantized parameters (r, θ). So the element with the highest value indicates the straight line that is most represented in the input image.

1.3 Results

The plots of line detection is shown in figure 1, 2, 3 and 4.

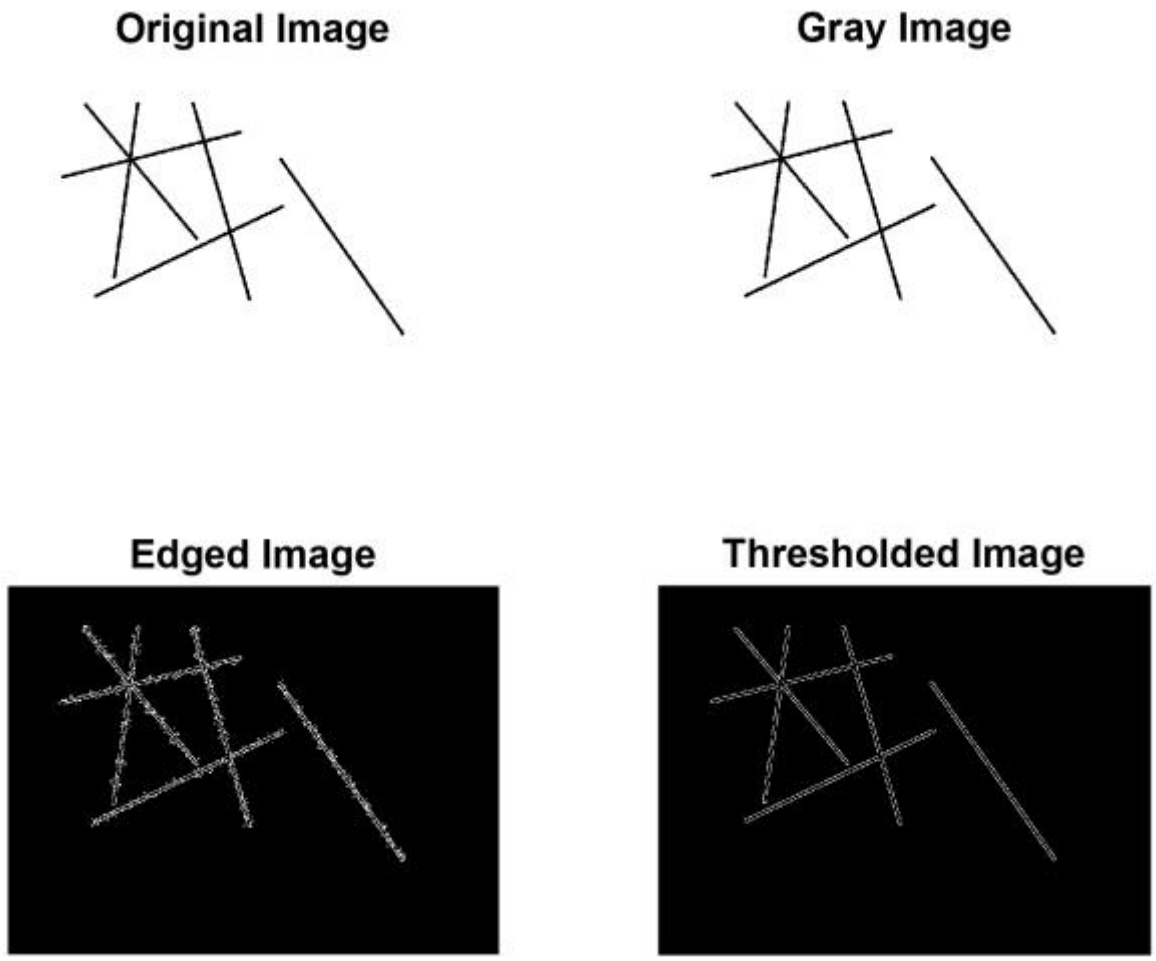

Figure 1. Output images before Hough transform

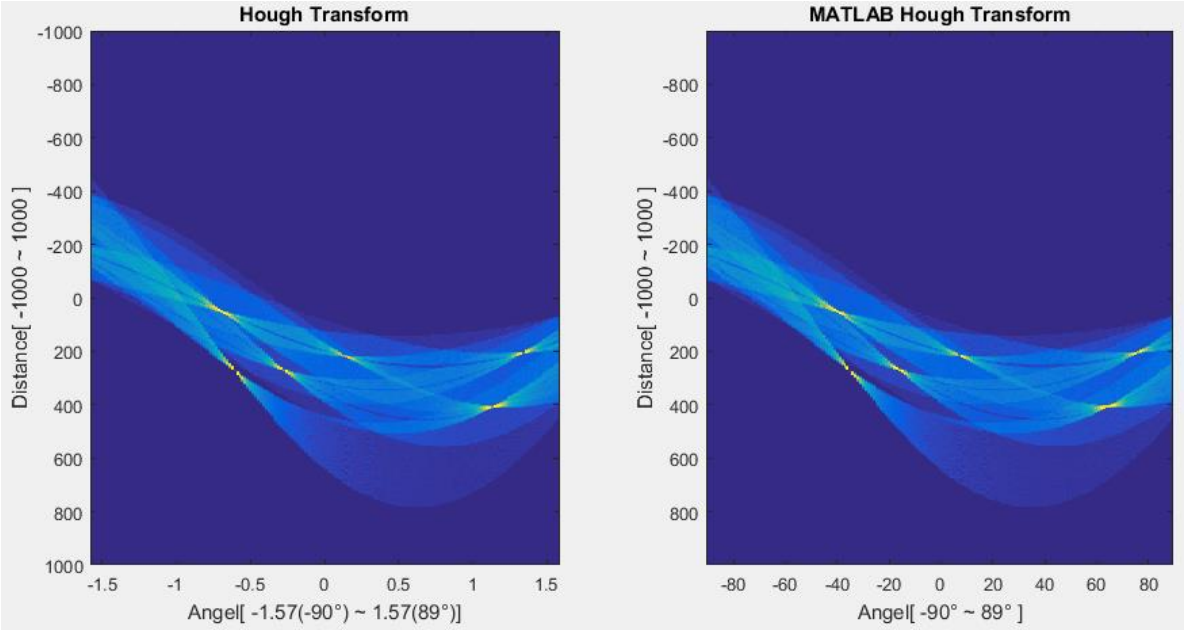

Figure 2. Plot of Hough matrices between my function and MATLAB function

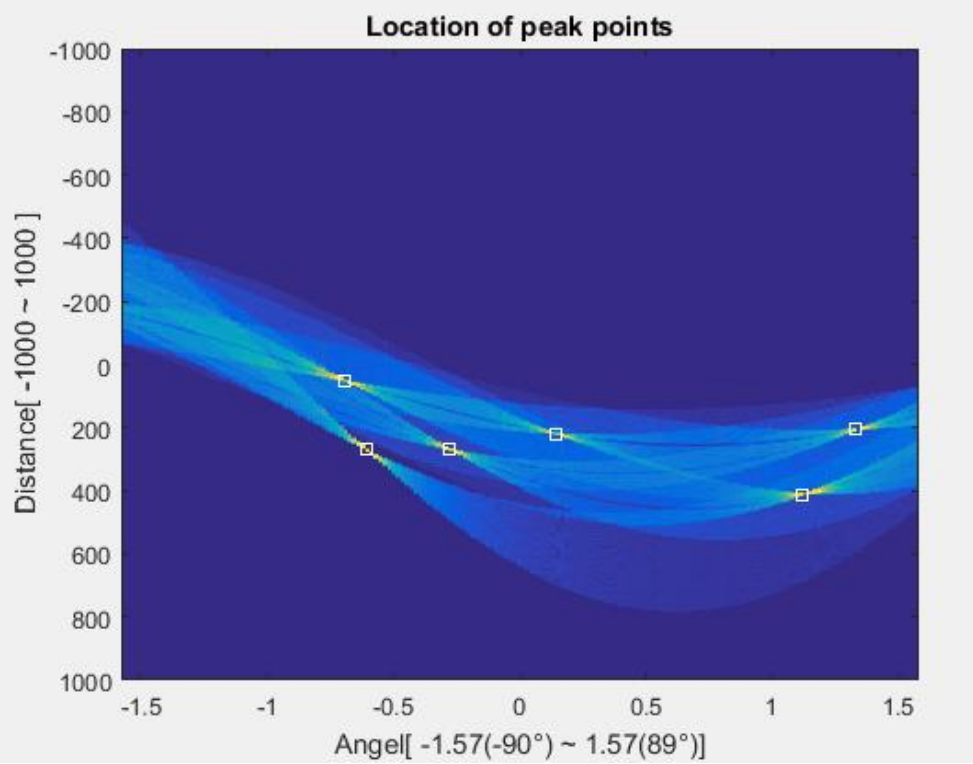

Figure 3. The peak points in Hough matrix

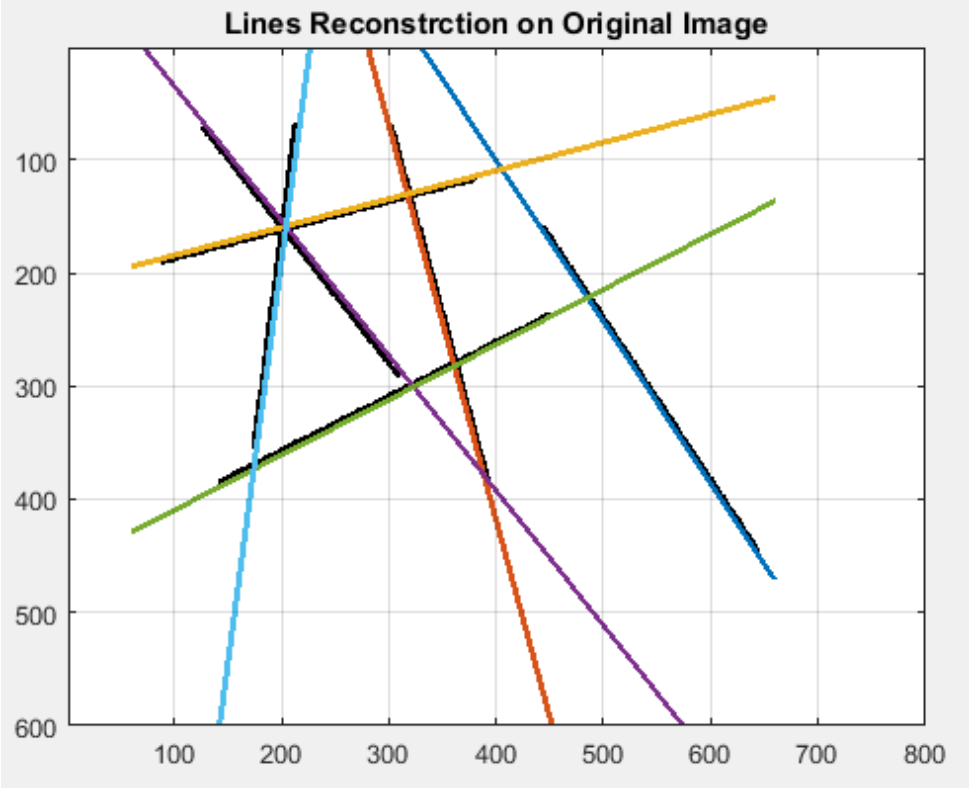

Figure 4. Line reconstruction on original image

The output parameters of line is shown in following table 1.1.

Angle(-90°-89°)	Distance(pixel)
-35°	270
-16°	269
76°	203
-40°	54
64°	412
8°	224

Table 1.1 Parameters of detected line

Part 2 Hough Transform for Detecting Ellipse

2.1 Basic concepts

The basic idea of Hough transform is to implement a voting procedure for all potential curves in the image, and at the termination of the algorithm, curves that do exist in the image will have relatively high voting scores.

However, this generalized Hough Transform needs a five dimensional parameter space to detect an ellipse. That is very memory and time consuming. Thus, I tried to deduce to computational complexity of Hough transform in three steps. Firstly, I adopted Randomized Hough Transform which is described by Wikipedia [3]. But since our dataset is not that big, I just wrote the code about it and did not utilize it in my final implementation. Secondly, I followed "A new efficient ellipse detection method" (see Yonghong, Qiang [4]) to implement Hough transform for detecting ellipse. Thirdly, based on the description of problem, I set sorts of boundary conditions to constrain my implementation.

The following paragraphs are mainly focus on the second step of complexity deduction.

For each pair of pixels (x_1, y_1) and (x_2, y_2), we assume they are two vertices on the major axis of an ellipse. We also can assume they are two vertices on the minor axis of an ellipse, but it will increase computational time because we need to check pixels in a larger range of the image. Then we can calculate four parameters for the assumed ellipse as following:

$$x_0 = (x_1 + x_2)/2 \quad (2.1)$$

$$y_0 = (y_1 + y_2)/2 \quad (2.2)$$

$$a = [(x_2 - x_1)^2 + (y_2 - y_1)^2]^{1/2}/2 \quad (2.3)$$

$$\alpha = atan[(y_2 - y_1)/(x_2 - x_1)] \quad (2.4)$$

Where (x_0, y_0) is the center of the assumed ellipse, a the half-length of the major axis and α the orientation of the ellipse.

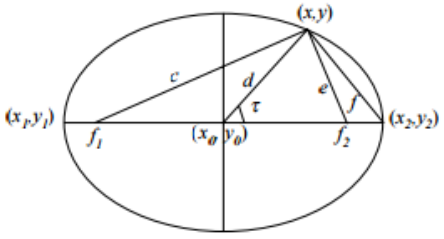

Figure 5. Ellipse geometry

Figure 5 shows ellipse geometry. f_1 and f_2 are foci of the ellipse and (x, y) is the third point used to calculate the fifth parameter. The distance between (x, y) and (x_0, y_0) should be less than the distance between (x_1, y_1) and (x_0, y_0) or between (x_2, y_2) and (x_0, y_0). So the half-length of the minor axis can be estimated by the following equation,

$$b^2 = (a^2 d^2 sin^2\tau)/(a^2 d^2 cos^2\tau) \qquad (2.5)$$

where $cos\tau$ is,

$$cos\tau = (a^2 + d^2 - f^2)/(2ad) \qquad (2.6)$$

and d is the distance between (x, y) and (x_0, y_0).

Consequently, by using equations (2.1) - (2.6) it is possible to calculate all five parameters of an ellipse.

2.2 Implementation

As I mentioned in part 1, we should implement Edge detection and Edge thresholding before Ellipse detection. For Ellipse detection part, I totally followed the step which is provided by previous paper.

2.3 Results

The construction of best-fitted ellipses is shown in figure 6.

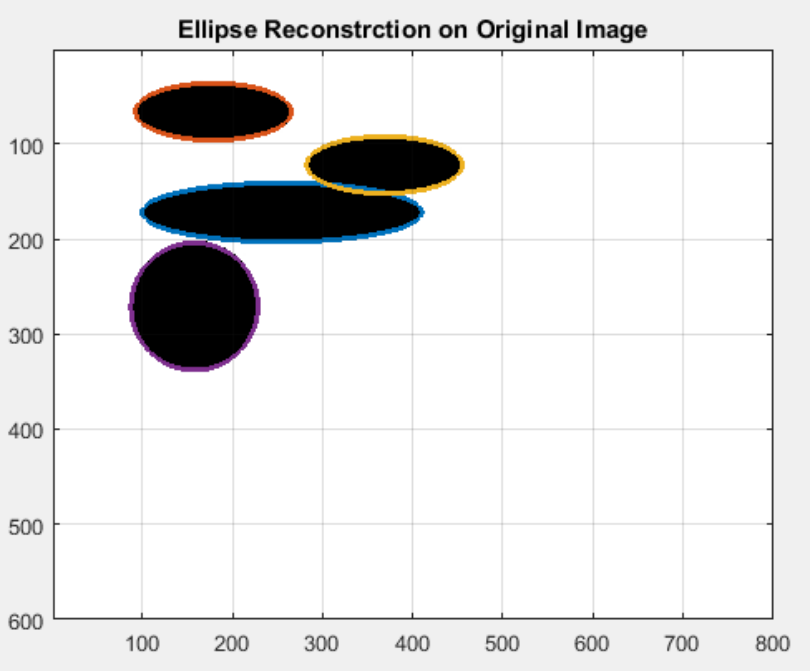

Figure 6. Ellipse reconstruction on original image

The parameters of best-fitted ellipses is shown in table 2.1.

X_0	Y_0	a	b	α
254.5	173	155.5	31	0
178.5	65	86.5	29	0
157.5	272	70.5	65	0
368.5	121	86.5	31	0

Table 2.1 Parameters of detected line

Part 3 Discussion and Conclusion

3.1 Hough Transform for Detecting Lines

Detecting lines is the basic application of Hough transform. Following the instruction of algorithm, I met the results as I expected. Thus, all I can say is that I obtained the basic understanding of Hough transform. Since I spent a lot of time in ellipses detection, I will discuss this part more in bellow paragraphs.

3.2 Hough Transform for Detecting Ellipses

As I mentioned in part 2.2, I followed step which is suggested by Yonghong, Qiang. During implementation, the dilemma I faced is not from the calculation part, but from the way to determine the best-fitted parameters of ellipses among a large number of potential parameters. At start of this assignment, I intuitionally selected the top 4 parameters votes without repetitive. But, I noticed that for the two separable ellipses, I can always get the good fit, but for the two overlapping ellipses, sometimes parameter b is either bigger or smaller, in worst situation, we just can detect one ellipse. To deal with this problem, I implemented another method which is inspired by k-means algorithm in machine learning area. In this method, we chose the best 60 scores at first, and clustered them into 4 clusters by k-means algorithm. Through k-means algorithm, I highly improved the stability of Hough transform for detecting ellipse. However, it is undeniable that the bad detection still happens in extremely rare case. Now all I can do just rerun my code when this happens.

3.3 Future work

There are two parts of my future work.

First, I should reduce the occurrence rate of bad detection.

Second, I should improve the efficiency of my program.

References:

[1] Nikos Papanikolopoulos. “CSci 5561Computer Vision” *Slide1-3.*

[2] Wikipedia. “Hough transform” *https://en.wikipedia.org/wiki/Hough_transform*

[3] Wikipedia. “Randomized Hough transform” *https://en.wikipedia.org/wiki/Randomized_Hough_transform*

[4] Xie, Yonghong, and Qiang Ji. "A new efficient ellipse detection method." *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. Vol. 2. IEEE, 2002.