Efficient algorithm for detecting concentric circles in binary images

ICIC Express Letters ICIC International ©2010 ISSN 1881-803X
Volume **4**, Number **6(B)**, December **2010** pp. **2495-2500**

EFFICIENT ALGORITHM FOR DETECTING CONCENTRIC CIRCLES IN BINARY IMAGES

Zhenjun Tang and Xianquan Zhang

Department of Computer Science
Guangxi Normal University
No. 15, Yucai Road, Guilin 541004, P. R. China
{ tangzj230; zxq6622 }@163.com

Received March 2010; accepted June 2010

Abstract. *To reduce time complexity and space complexity of detecting concentric circles, we propose an efficient algorithm which doesn't need an accumulator used in standard Hough transform (SHT). It extracts edge pixels from binary image and computes their gravity center. To find circle number, these points are divided into different subsets by using the distances between points and gravity center. The subset with enough points is considered as a circle. By searching around the gravity center, the real circle center is determined and its radiuses are then obtained. The proposed algorithm can correctly extract circles with $O(n \log n)$ time at the worst case. Theoretical analysis and experimental results show that it is better than SHT in time complexity and space complexity. It has a fast speed, indicating the usefulness of the technique in real-time applications.*
Keywords: Circle detection, Gravity center, Hough transform (HT)

1. **Introduction.** Circle detection in digital images is an important problem in various industrial applications, such as object recognition, automatic inspection [1, 2] and assembly. Many researchers have devoted themselves to circle detection. Their works are mainly classified into two kinds. One is Hough transform (HT) based algorithms and the other is non-HT-based schemes.

HT is initially introduced by Hough and Paul [3] in computer vision to detect straight line. Duda and Hart [4] first propose the original standard Hough transform (SHT) method to detect circles and Kimme et al. [5] use it to detect circle in real images. The basic idea of SHT is to detect parameterized circles by mapping edge pixels into a three-dimensional parameter space. Circles are then obtained by finding out the peak values in the parameter space. SHT is a robust technique, but it needs large computation and space. In order to overcome these drawbacks, some modified versions of SHT have been proposed. Davies [6] observes that it is applicable to use just one instead of a large number of planes in the parameter space. Kierkegaard [7] proposes a HT-based scheme for detecting circular arcs. Its computational complexity is $O(nm)$, where n and m are the maximal values of x-coordinate and y-coordinate, respectively. Li et al. [8] present a fast Hough transform. It recursively divides the parameter space into hypercubes from low to high resolution and only performs HT on the hypercubes with votes exceeding a selected threshold. In another work, Xu et al. [9] give a randomized Hough transform (RHT). RHT has high performance for images with slight noises, but the performance will slow down as the noises increase [10]. Ioannou et al. [11] present a two-step algorithm based on 2D HT, which uses the radius histogramming to validate the existence of circle center. Considering that conventional voting method of SHT affects the detection accuracy, Chiu and Liaw [12] propose a new voting method to improve the efficiency of circle detection by letting each pixel belong to one candidate of circle parameters.

Several non-HT-based schemes are also applied to detect circles. In [13], Ho et al. use global geometric symmetry to find circle center and radius. Yin [14] proposes a scheme to detect circle and ellipse, which consists of a genetic algorithm phase and a local search phase. In another study, Kim et al. [15] propose a method based on the intersection of pairs of chords. Ayala-Ramirez et al. [16] apply genetic algorithms to extract circles.

Concentric circle detection is a special problem of circle detection, where all circles have the same center. Conventional detection methods, such as SHT, can also extract concentric circles, but have high complexities of time and space. To reduce the complexities, we propose a method for concentric circle detection. Our method doesn't need an accumulator and can find circles with $O(n \log n)$ time at the worst case. It is better than SHT in time complexity and space complexity. The rest of this paper is organized as follows. The detailed algorithm is described in Section 2. The complexity analysis and experimental results are presented in Section 3 and Section 4, respectively. Finally, conclusions are drawn in Section 5.

2. **Proposed Algorithm.**

The proposed algorithm extracts circles from a binary image. It consists of four steps, i.e., image scanning, gravity center computation, point set division, and circle center determination. In the first step, we scan the binary image and record the coordinates of edge pixels. Then, we estimate reference circle center by calculating the gravity center of point set. In the third step, we divide these points into different subsets by using the distances between points and gravity center, and then determine the circle number. Finally, the concentric circle center and radiuses are computed. Suppose that the binary image has n edge pixels, which are recorded in an array A.

2.1. **Compute the gravity center.**

Obviously, the circle center is the circle's gravity center. Thus, it can be determined by computing the gravity center. Let the point set $\mathbf{P} = \{p_1, p_2, \ldots, p_m\}$ denote all points on the circle, where p_i's coordinates are (x_i, y_i). Then the circle center is obtained by computing the gravity center (x_g, y_g) as follows.

$$\begin{aligned} x_{\mathrm{g}} &= \tfrac{1}{m}\textstyle\sum_{i=1}^{m} x_i \\ y_{\mathrm{g}} &= \tfrac{1}{m}\textstyle\sum_{i=1}^{m} y_i \end{aligned} \tag{1}$$

In practice, noises often occur in the image and some points on the circle are probably missing. So the extracted points are not always on the circles. This means that if $\mathbf{P}$ represents the extracted points, the coordinates (x_g, y_g) are not always the real circle center. Nonetheless, the gravity center is around the real circle center since the points on the circles are dominant. So the gravity center can be considered as the reference circle center.

2.2. **Circle extraction by point set division.**

Circle properties indicate that the distance between the circle center and arbitrary points on the circle is a constant, called radius. If there are several concentric circles, there are several corresponding radiuses. Obviously, points on different circles have different radiuses. Thus, they can be divided into different subsets by using these distances. A subset with enough points is expected to construct a circle. Otherwise, these points forming the subset are considered as noises. So the circle number is obtained by extracting subsets. For each subset, the gravity center is considered as the reference circle center for circle detection.

Let (x_c, y_c) be the coordinates of reference circle center. The point set division is done as follows. Use Equation (2) to calculate the distance d between each point and the reference circle center.

$$d = \sqrt{(x - x_{\mathrm{c}})^2 + (y - y_{\mathrm{c}})^2} \tag{2}$$

Take the distance as key and sort the array A recording the edge pixels to make an ascending array B. Thus, for B, elements with the similar distances are close in position.

Scan B and divide the elements into different subsets. During division, elements satisfying the following two conditions are divided into the same subset. The first condition is that the absolute difference between the mean distance and point's distance is not bigger than a threshold Δd. The other condition is that the absolute difference between the distances of current point and previous points is not bigger than Δd. For each subset, we use Equation (3) to calculate the R value:

$$R = \frac{s}{2\pi r} \tag{3}$$

where s is the point number of subset, and r is the mean of those points' distances. In fact, R is a ratio between the number of points on the circle and the theoretical circle perimeter. As the coordinates of point are discrete, the point number is usually smaller than the theoretical perimeter. This means that the R value is generally smaller than 1. The bigger the R value, the bigger the number of points on the circle. Thus, we can set a threshold ΔT to discard those subsets having few points as they couldn't form a circle. So the subset is expected to construct a circle if its R value is bigger than ΔT.

Let D be an array recording the points' distance, where $D[i]$ represents the distance of $B[i]$. The detailed steps are as follows.

STEP 1: Calculate the distance between each point and the reference circle center by Equation (2).

STEP 2: Take distance as key, and sort A to make an ascending array B by using the heapsort algorithm.

STEP 3: Let $i = 1$, and $j = 2$.

STEP 4: Compute the mean distance r using the points from $B[i]$ to $B[j]$.

STEP 5: If $(|r - D[j]| > \Delta d$ or $|D[j] - D[j-1]| > \Delta d)$ goto **STEP 7**.

STEP 6: $j \leftarrow j + 1$. If $(j \leq n)$ goto **STEP 4**.

STEP 7: $s \leftarrow j - i$, and compute R by Equation (3).

STEP 8: If $(R \leq \Delta T)$ goto **STEP 10**.

STEP 9: Calculate the gravity center of subset forming by the points from $B[i]$ to $B[j-1]$, and record the gravity center and mean distance r for circle drawing.

STEP 10: If $(j == n)$ the division is done. Otherwise, $i \leftarrow j, j \leftarrow j + 1$, and goto **STEP 4**.

2.3. **Determine the concentric circle center.** The above analysis show that the point set's gravity center is not always the real circle center. But it is around the circle center. This means that the circle center can be determined by searching around the gravity center. Using circle properties, we can derive the following theorem.

Theorem 2.1. *The variance of distances between the circle center and points on the circle is 0.*

Proof: Let $\{d_1, d_2, \ldots, d_m\}$ be all distances between circle center and points on the circle. Circle properties show that $d_i = d_j (1 \leq i, j \leq m)$. Use d_0 to represent $d_i (1 \leq i \leq m)$ and compute the mean of distances μ by Equation (4). Then, $\mu = d_0$. Substitute d_i and μ into Equation (5). Hence, the distance variance v is 0.

$$\mu = \frac{1}{m} \sum_{i=1}^{m} d_i \tag{4}$$

$$v = \frac{1}{m-1} \sum_{i=1}^{m} (d_i - \mu)^2 \tag{5}$$

Consider that the coordinates of points are discrete. The distance between point and circle center is not equal to d_0 all the time. Hence, the distance variance is not accurately equal to 0. But it should be very small and close to 0. In other words, the distance variance on other position is larger than that on the real circle center. Thus, the circle center can be determined by the following steps. Take the gravity center as the center of window sized $(2L + 1) \times (2L + 1)$. Calculate the distance variance on each point of the window. The point with minimum variance is considered as the circle center, and then the corresponding mean of distances is the radius. For each circle, compute its circle center. Thus, the coordinates of concentric circle center is obtained by calculating the mean coordinates of these circle centers. Generally, the bigger the value of L, the bigger the possibility of finding the real circle center. Note that as the L value increases, the computational cost will linearly increase. A tradeoff is thus needed in choosing the L value. In experiments, we find that $L = 20$ can achieve a balance between the computation and the accuracy of coordinates.

3. **Complexity Analysis.** Let n be the number of edge pixels in the binary image. Assume that n edge pixels make t concentric circles. Since the proposed algorithm only requires storing the information of edge pixels and circles, its space complexity is $(n+t)$. Most computation of the proposed algorithm is done in sorting the points. In our scheme, the heapsort algorithm is exploited to achieve this goal, whose time complexity is $O(n\text{log}n)$ at the worst case. So the proposed algorithm can detect concentric circles with $O(n \log n)$ time. The time complexity and space complexity of SHT are $O(nN^{k-1})$ and $O(N^k)$, where N is the maximal number of quantization levels for various parameters and k is the number of parameter. Both complexities grow exponentially with the value of k (For circle detection, $k = 3$.). Comparisons show that the proposed algorithm has lower complexities of time and space than those of SHT.

4. **Experimental Results.** The proposed algorithm is implemented in C language and executed on a PC with Pentium D 2.8 GHz CPU and 512 MB RAM. To validate the performance, we compare it with SHT. In experiments, $\Delta T = 0.75$, $\Delta d = 6$, and $L = 20$. Many experiments are conducted to validate the proposed algorithm. In here, three typical results are listed, as shown in Figures 1, 2 and 3. The image sizes of Figure 1(a), Figure 2(a) and Figure 3(a) are 300×290, 201×203 and 204×198, respectively.

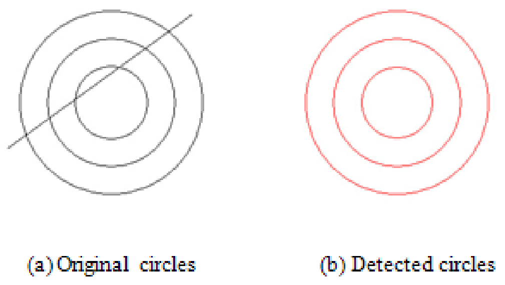

FIGURE 1. A straight line runs through concentric circles

Figure 1(a) is a synthetic image with three concentric circles, where the circles center is $(150, 150)$ and the radiuses from big to small are $100, 70, 40$ respectively, and a straight line runs through these circles. Figure 1(b) is the detected circles by the proposed algorithm. Table 1 shows the comparisons of detecting Figure 1(a) between the proposed scheme and SHT. Both schemes can detect the real center correctly and compute two right radiuses. Figure 2(a) shows another image with two broken concentric circles. To test the robustness of the proposed algorithm, salt and pepper noise is added to the image, where the noise density is 0.02. Figure 2(b) is the detected results of our scheme. Since the image is contaminated by noises, both schemes can not correctly detect the center. But there is

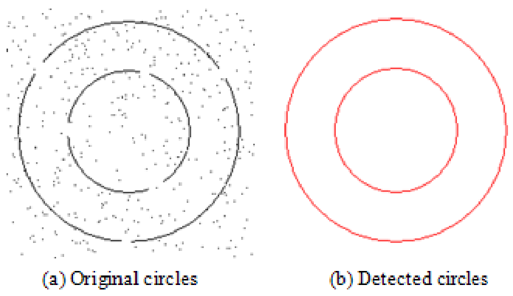

FIGURE 2. Broken concentric circles detection

only 1 pixel translation. The proposed algorithm correctly detects two radiuses, while SHT only correctly extracts one. Table 2 shows the detailed comparisons.

Figure 3(a) presents an image with a single circle whose edge is slightly distorted. The proposed scheme exactly finds the circle, as shown in Figure 3(b). Matching between the original and detected circles is shown in Figure 3(c). Comparisons about Figure 3 between the two schemes are listed in Table 3. The above results show that the detection accuracies of the proposed algorithm and SHT are almost equal. But our scheme has a fast speed. As the image size or the edge pixels' number increases, both schemes need more time. However, SHT requires much more time than our algorithm.

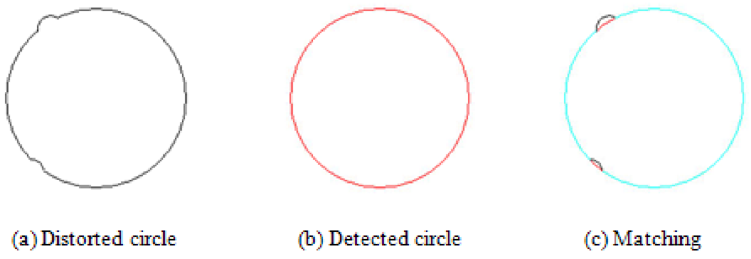

FIGURE 3. Distorted circle detection

TABLE 1. Comparison of detecting Figure 1(a) between our algorithm and SHT

	circle center	big radius	middle radius	small radius	CPU time(second)
Real circles	(150,150)	100	70	40	–
Our algorithm	(150,150)	100	70	39	0.093
SHT	(150,150)	100	69	40	22.16

TABLE 2. Comparison of detecting Figure 2(a) between our algorithm and SHT

	circle center	big radius	small radius	CPU time(second)
Real circles	(100,100)	90	50	–
Our algorithm	(99,100)	90	50	0.078
SHT	(99,99)	89	50	7.610

Table 3. Comparison of detecting Figure 3(a) between our algorithm and SHT

	circle center	radius	CPU time(second)
Real circle	(100,100)	80	–
Our algorithm	(100,100)	80	0.047
SHT	(101,101)	79	3.016

5. Conclusions.

In this paper, we have proposed an efficient algorithm for detecting concentric circles. It only needs to store edge pixels and circle information. Hence, it reduces the space complexity. Since the computation of circle detection mainly depends on the heapsort algorithm, the proposed algorithm can extract concentric circles with $O(n \log n)$ time at the worst case. Theoretical analysis and experimental results show that the propose algorithm is better than SHT in space complexity and time complexity. The fast speed and low storage of our algorithm satisfy the requirements of real-time applications and then make itself practical.

Acknowledgment. This work was partially supported by the Natural Science Foundation of China (60963008, 60763011), the Natural Science Foundation of Guangxi (0832104, 0447035), the project of the education administration of Guangxi (200607MS135), the Scientific and Technological Research Projects of Chongqing's Education Commission (KJ081309), and the Scientific Research Foundation of Guangxi Normal University for Doctors.

REFERENCES

[1] Z. Tian, Z. Ying and K. Yuan, Automatic detection of abnormal regions using Guassian mixture model, *ICIC Express Letters*, vol.3, no.4(A), pp.921-926, 2009.
[2] T. Azim, M. A. Jaffar and A. M. Mirza, Automatic fatigue detection of drivers through real time eye tracking, *ICIC Express Letters*, vol.4, no.2, pp.341-346, 2010.
[3] V. Hough and C. Paul, *Methods and Means for Recognizing Complex Patterns*, US patent 3069654, 1962.
[4] R. O. Duda and P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures, *Communications of the ACM*, vol.15, no.1, pp.11-15, 1972.
[5] C. Kimme, D. Ballard and J. Sklansky, Finding circles by an array of accumulators, *Communications of the ACM*, vol.18, no.2, pp.120-122, 1975.
[6] E. R. Davies, A modified Hough scheme for general circle detection, *Pattern Recognition Letters*, vol.7, no.1, pp.37-43, 1988.
[7] P. Kierkegaard, A method for detection of circular arcs based on the Hough transform, *Machine Vision Applications*, vol.5, no.4, pp.249-263, 1992.
[8] H. Li, M. A. Lavin and R. J. L. Master, Fast Hough transform: A hierarchical approach, *Computer Vision, Graphics, and Image Processing*, vol.36, no.2/3, pp.139-161, 1986.
[9] L. Xu, E. Oja and P. Kultanan, A new curve detection method: Randomized Hough transform, *Pattern Recognition Letters*, vol.11, no.5, pp.331-338, 1990.
[10] N. Kiryati, H. Kälviäinen and S. Alaoutinen, Randomized or probabilistic Hough transform: Unified performance evaluation, *Pattern Recognition Letters*, vol.21, no.13/14, pp.1157-1164, 2000.
[11] D. Ioannou, W. Huda and A. F. Laine, Circle recognition through a 2D Hough transform and radius histogramming, *Image and Vision Computing*, vol.17, no.1, pp.15-26, 1999.
[12] S. Chiu and J. Liaw, An effective voting method for circle detection, *Pattern Recognition Letters*, vol.26, no.2, pp.121-133, 2005.
[13] C. Ho and L. Chen, A fast ellipse/circle detector using geometry symmetry, *Pattern Recognition*, vol.28, no.1, pp.117-124, 1995.
[14] P. Yin, A new circle/ellipse detector using genetic algorithms, *Pattern Recognition Letters*, vol.20, no.7, pp.731-740, 1999.
[15] H. S. Kim and J. H. Kim, A two-step circle detection algorithm from the intersecting chords, *Pattern Recognition Letters*, vol.22, no.6/7, pp.787-798, 2001.
[16] V. Ayala-Ramirez, C. H. Garcia-Capulin, A. Perez-Garcia and R. E. Sanchez-Yanez, Circle detection on images using genetic algorithms, *Pattern Recognition Letters*, vol.27, no.6, pp.652-657, 2006.