J. Cent. South Univ. (2013) 20: 2663−2675
DOI: 10.1007/s11771-013-1782-6

Springer

Iterative circle fitting based on circular attracting factor

WANG Heng-sheng(王恒升)[1, 2], ZHANG Qiang(张强)[1], WANG Fu-liang(王福亮)[1, 2]

1. College of Mechanical and Electrical Engineering, Central South University, Changsha 410083, China;
2. State Key Laboratory for High Performance Complex Manufacturing (Central South University), Changsha 410083, China

Abstract: An intuitive method for circle fitting is proposed. Assuming an approximate circle ($C_{A,n}$) for the fitting of some scattered points, it can be imagined that every point would apply a force to $C_{A,n}$, which all together form an overall effect that "draws" $C_{A,n}$ towards best fitting to the group of points. The basic element of the force is called circular attracting factor (CAF) which is defined as a real scalar in a radial direction of $C_{A,n}$. An iterative algorithm based on this idea is proposed, and the convergence and accuracy are analyzed. The algorithm converges uniformly which is proved by the analysis of Lyapunov function, and the accuracy of the algorithm is in accord with that of geometric least squares of circle fitting. The algorithm is adopted to circle detection in grayscale images, in which the transferring to binary images is not required, and thus the algorithm is less sensitive to lightening and background noise. The main point for the adaption is the calculation of CAF which is extended in radial directions of $C_{A,n}$ for the whole image. All pixels would apply forces to $C_{A,n}$, and the overall effect of forces would be equivalent to a force from the centroid of pixels to $C_{A,n}$. The forces from would-be edge pixels would overweigh that from noisy pixels, so the following approximate circle would be of better fitting. To reduce the amount of calculation, pixels are only used in an annular area including the boundary of $C_{A,n}$ just in between for the calculation of CAF. Examples are given, showing the process of circle fitting of scattered points around a circle from an initial assuming circle, comparing the fitting results for scattered points from some related literature, applying the method proposed for circular edge detection in grayscale images with noise, and/or with only partial arc of a circle, and for circle detection in BGA inspection.

Key words: circle detection; circle fitting; grayscale image; iterative algorithm; least squares fitting (LSF); circular attracting factor (CAF); BGA inspection

1 Introduction

Circle detection in grayscale images is required in many applications, which has been studied for decades. Part of the applications of circle detection are automatic inspection in manufacturing process, medical diagnosis based on images [1], target tracking and pupil contour [2−3] detection.

Conventionally, the basic two stages for circle detection in a grayscale image are: 1) Edge extracting: outputting a binary image preserving edge information while reducing a large amount of data to be processed; 2) Circle fitting: estimating parameters of circle (center coordinates and radius of circle) from the edge information. For the first stage, there have been algorithms like CANNY [4], Robert and Prewitt [5−6] and Driche [7] as edge detectors. For the second stage, most widely used algorithms for circle detection are based on Hough transform (HT) [8] like the probabilistic HT [9−10], the randomized HT [11] and the fuzzy HT [12−13]. There are some other circle detection algorithms based on least squares fitting (LSF) [14−17], or based on the direction information of edges [18] or geometrical properties of circle [19−20]. A very good manual for the topic of fitting circles and circular arcs to observed points can be seen in Ref. [21].

The common drawback is that the threshold for making a binary image from grayscale image would be affected by the lighting, and thus the fitting result in the second stage. In this work, an iterative algorithm for circle detection in grayscale image is presented, which has no need for the image to be transferred to binary.

2 Intuitive idea of method

2.1 Intuitive idea

We use a set of three ordered elements, say (x, y, r),

Foundation item: Project(2013CB035504) supported by the National Basic Research Program of China; Project(2012zzts078) supported by the Fundamental Research Funds for the Central Universities of Central South University, China; Project(2009ZX02038) supported by the National Science and Technology Major Project of the Ministry of Science and Technology of China
Received date: 2012−07−18; **Accepted date:** 2012−10−20
Corresponding author: WANG Heng-sheng, Professor, PhD; Tel: +86−731−88877381; E-mail: whsheng@csu.edu.cn

to denote the parameters of a circle with the center at coordinates (x, y) and the radius being r. Figure 1(a) shows a set of scattered points, $(x_i, y_i), i \in [1, m]$, to be fitted by a circle. $C_{A,n}(x_n, y_n, r_n)$ is an approximate circle for the fitting of these points. Traditional geometric least squares fitting [22] is to minimize the following objective function:

$$S(x_n, y_n, r_n) = \sum_{i=1}^{m} (\sqrt{(x_i - x_n)^2 + (y_i - y_n)^2} - r_n)^2 \quad (1)$$

where the expression inside the summation is the square of the distance of the i-th point to the approximate circle $C_{A,n}$ (as the line segments indicated in Fig. 1(a)), and the summation is the overall distance squared of these points to $C_{A,n}$. The objective is to find the best circle $C_{A,n}(x_n, y_n, r_n)$ to minimize $S(x_n, y_n, r_n)$. There have been algorithms solving this problem.

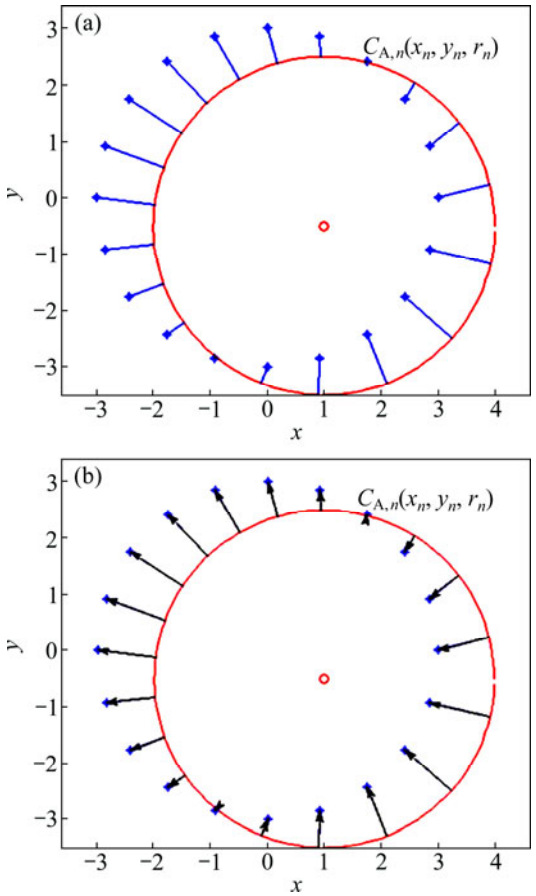

Fig. 1 Scattered points and approximate circles: (a) Distances between points and approximate circle; (b) Points applying forces to approximate circle

Our idea is that we think of the distances from circle $C_{A,n}$ to scattered points as some kind of forces as the arrows shown in Fig. 1(b). These imaginary forces apply to $C_{A,n}$ to make a fit better, which would pull $C_{A,n}$ to left-top direction in the case shown in Fig. 1(b). As for the radius, instinctively, we imagine that the forces (or arrows) pointing to the center (or inward) of $C_{A,n}$ would make $C_{A,n}$ shrink while the forces (or arrows) pointing to the outward of $C_{A,n}$ would make $C_{A,n}$ expand.

2.2 Formulating idea

Ideally, we think that there is an unknown target circle to be detected, which is denoted as C_T in Fig. 2. We are going to find the parameters (x_o, y_o, r_o) of C_T with an iterative method. Suppose we are now at the n-th step of the iteration with an approximate circle (x_n, y_n, r_n) which is denoted as $C_{A,n}$ in Fig. 2. We are going to see how C_T would apply forces to $C_{A,n}$ to draw it closer to C_T. To this end, we define circular attracting factor $\alpha_n(\theta)$ as follows.

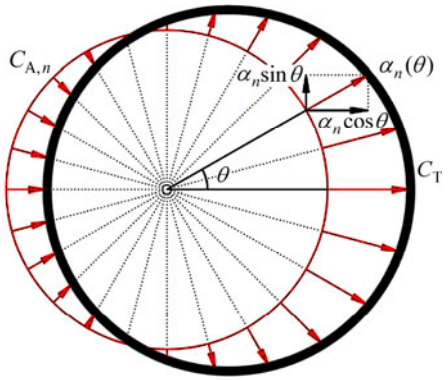

Fig. 2 Illustration of definition of CAF $\alpha_n(\theta)$

Definition 1: Circular attracting factor (CAF) $\alpha_n(\theta)$ is defined as a scalar of real value, such that the absolute value of $\alpha_n(\theta)$ equals the length of the arrow line in the radial direction of θ starting from $C_{A,n}$ and ending at C_T (Fig. 2). The sign of $\alpha_n(\theta)$ is positive with the outward arrow (when the arrow goes far away from the center of $C_{A,n}$) and negative with the inward arrow (when the arrow goes towards to the center of $C_{A,n}$).

With **Definition 1**, we further define virtual drawing forces for radius (F_{rn}) and for center coordinates (F_{xn}, F_{yn}) of $C_{A,n}$ which would make the radius and center of $C_{A,n}$ more closer to C_T in the following iteration.

Definition 2:

$$F_{rn} = \int_0^{2\pi} \alpha_n(\theta) \mathrm{d}\theta \quad (2)$$

$$F_{xn} = \int_0^{2\pi} \alpha_n(\theta) \cos\theta \mathrm{d}\theta \quad (3)$$

$$F_{yn} = \int_0^{2\pi} \alpha_n(\theta) \sin\theta \mathrm{d}\theta \quad (4)$$

The forces defined above would make increments of radius and center coordinates of the approximate circle for the following step as follows:

$$\Delta r_n = \lambda_r F_{rn} = \lambda_r \int_0^{2\pi} \alpha_n(\theta) \mathrm{d}\theta \quad (5)$$

$$\Delta x_n = \lambda_x F_{xn} = \lambda_x \int_0^{2\pi} \alpha_n(\theta) \cos\theta \mathrm{d}\theta \quad (6)$$

$$\Delta y_n = \lambda_y F_{yn} = \lambda_y \int_0^{2\pi} \alpha_n(\theta)\sin\theta \mathrm{d}\theta \tag{7}$$

which would have

$$\begin{cases} x_{n+1} = x_n + \Delta x_n \\ y_{n+1} = y_n + \Delta y_n \\ r_{n+1} = r_n + \Delta r_n \end{cases} \tag{8}$$

or the following approximate circle $C_{A,n+1}(x_{n+1}, y_{n+1}, r_{n+1})$ intentionally being closer to C_T than the current approximate circle $C_{A,n}(x_n, y_n, r_n)$. We will see how to determine coefficients λ_r, λ_x and λ_y in Eqs. (5)−(7) to this end.

To decide the coefficient λ_r in Eq. (5), we consider a situation as shown in Fig. 3 which shows that the two centers of the circles coincide but the radius of $C_{A,n}$ has little deviation from that of C_T. From the definition of $\alpha_n(\theta)$, it is easy to write

$$\alpha_n(\theta) = r_o - r_n \tag{9}$$

and to get $F_{rn} = (r_o - r_n)\cdot 2\pi$ from Eq. (2).

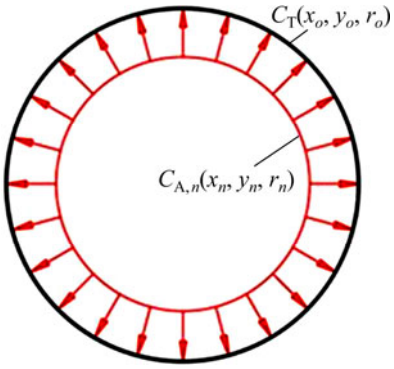

Fig. 3 Case of approximate circle with only little deviation from target circle in radius

To make a one-step convergence which means we want $r_{n+1}=r_o$, i.e. $\Delta r_n = r_o - r_n$, from Eq. (5), we have

$$\lambda_r = \frac{1}{2\pi} \tag{10}$$

Similarly, as for the determination of coefficient λ_T, we consider a situation that, as shown in Fig. 4, the radii of C_T and $C_{A,n}$ are the same, but the center of $C_{A,n}$ has little deviation (Δx_n) from that of C_T in horizontal direction.

The polar equation of C_T in Fig. 4 with the origin of polar coordinate system at the center of $C_{A,n}$ is

$$r(\theta) = \sqrt{r_o^2 - \Delta x_n^2 \cdot \sin^2\theta} + \Delta x_n \cdot \cos\theta \tag{11}$$

And CAF is

$$\alpha_n(\theta) = r(\theta) - r_n = \sqrt{r_o^2 - \Delta x_n^2 \cdot \sin^2\theta} + \Delta x_n \cdot \cos\theta - r_n \tag{12}$$

Putting Eq. (12) into Eq. (3), we have

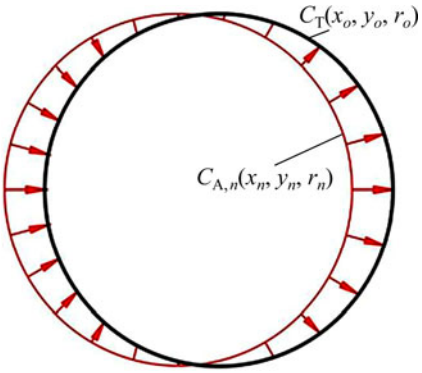

Fig. 4 Case of approximate circle with only little deviation from target circle in position of x direction

$$F_{xn} = \int_0^{2\pi} (\sqrt{r_o^2 - \Delta x_n^2 \cdot \sin^2\theta} + \Delta x_n \cdot \cos\theta - r_n)\cos\theta \mathrm{d}\theta = \int_0^{2\pi} (\sqrt{r_o^2 - \Delta x_n^2 \cdot \sin^2\theta}\cos\theta + \Delta x_n \cdot \cos\theta \cdot \cos\theta - r_n\cos\theta)\mathrm{d}\theta \tag{13}$$

The integration of the first and third term in Eq. (13) turns out to be 0, and we have

$$F_{xn} = \Delta x_n \cdot \pi \tag{14}$$

Putting Eq. (14) into Eq. (6), we have

$$\lambda_x = \frac{1}{\pi} \tag{15}$$

After a similar procedure for dealing with the attracting force in vertical direction as we do for λ_x, we have

$$F_{yn} = \Delta x_n \cdot \pi \tag{16}$$

All together, we have the following iterative formula:

$$\begin{cases} x_{n+1} = x_n + \Delta x_n \\ y_{n+1} = y_n + \Delta y_n \\ r_{n+1} = r_n + \Delta r_n \end{cases} \tag{17}$$

and

$$\begin{cases} \Delta x_n = \dfrac{1}{\pi}\displaystyle\int_0^{2\pi} \alpha_n(\theta)\cos\theta \mathrm{d}\theta \\ \Delta y_n = \dfrac{1}{\pi}\displaystyle\int_0^{2\pi} \alpha_n(\theta)\sin\theta \mathrm{d}\theta \\ \Delta r_n = \dfrac{1}{2\pi}\displaystyle\int_0^{2\pi} \alpha_n(\theta)\mathrm{d}\theta \end{cases} \tag{18}$$

where $\alpha_n(\theta)$ is defined in **Definition 1**.

Let's see a general situation shown in Fig. 5, which indicates that C_T and $C_{A,n}$ have differences in radius and in position of centers. We decouple the effects of these differences (or deviations) and have the following conditions.

1) The intermediate circle with dashed line is the

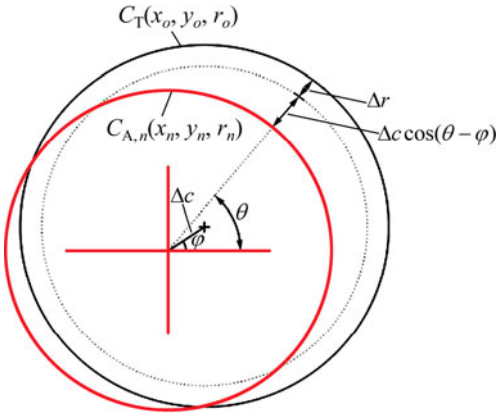

Fig. 5 One step convergence illustration

circle with the same center coordinates as C_T, and the same radius as $C_{A,n}$;

2) $\Delta r=r_o-r_n$, is the difference of the radii between C_T and $C_{A,n}$.

3) Δc is the distance between the centers of C_T and $C_{A,n}$ in the direction of φ.

The CAF defined in **Definition 1** consists of two parts caused by position deviation and radius deviation, and can be written as

$$\alpha_n(\theta)=\Delta c\cos(\theta-\varphi)+\Delta r \tag{19}$$

Putting Eq. (19) into Eq. (18), we know that in this situation,

$$\begin{cases}\Delta x_n=\dfrac{1}{\pi}\displaystyle\int_0^{2\pi}(\Delta c\cos(\theta-\varphi)+\Delta r)\cos\theta\mathrm{d}\theta=\Delta c\cos\varphi\\ \Delta y_n=\dfrac{1}{\pi}\displaystyle\int_0^{2\pi}(\Delta c\cos(\theta-\varphi)+\Delta r)\sin\theta\mathrm{d}\theta=\Delta c\sin\varphi\\ \Delta r_n=\dfrac{1}{2\pi}\displaystyle\int_0^{2\pi}(\Delta c\cos(\theta-\varphi)+\Delta r)\mathrm{d}\theta=\Delta r\end{cases} \tag{20}$$

From Eqs. (17) and (20) and Fig. 5, we can see that circle $C_{A,n+1}(x_{n+1}, y_{n+1}, r_{n+1})$ comes to the exact target circle C_T after only one step of iteration from $C_{A,n}(x_n, y_n, r_n)$.

3 Use of method in scattered data fitting

3.1 Algorithm

Suppose that we have a set of circular scatted points as shown in Fig. 6 marked with stars ("*"), and the assumed initial approximate circle $C_{A,n}(x_n, y_n, r_n)$. The CAF can be written according to **Definition 1** as

$$\alpha_n(i)=r_{ni}-r_n \tag{21}$$

where $r_{ni}(i\in[1, m])$ is the polar radius of the i-th point (x_i, y_i) respect to the polar coordinate origin at the center of $C_{A,n}$, and

$$r_{ni}=\sqrt{(x_i-x_n)^2+(y_i-y_n)^2} \tag{22}$$

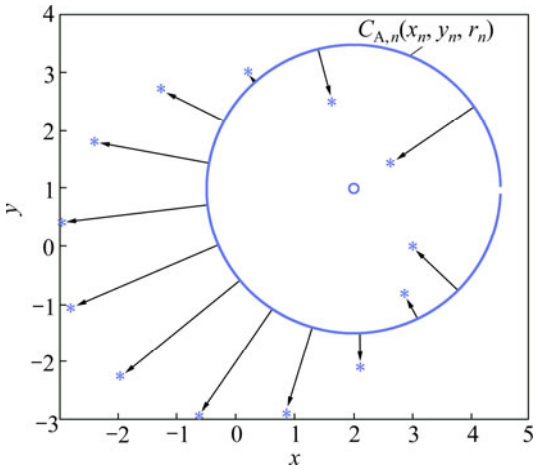

Fig. 6 Illustration of scattered points applying forces to approximate circle

Different from **Definition 1**, we have CAF with the form $\alpha_n(i)$ other than $\alpha_n(\theta)$, because we do not have continuous points to be fitted with a circle, but only numbered points. For the attracting force for radius, we have

$$F_{rn}=\sum_{i=1}^{m}(r_{ni}-r_n) \tag{23}$$

which is a summation of m terms (m scattered points) other than a continuous integration. We simply have the following increment of radius:

$$\Delta r_n=\lambda_r F_{rn}=\frac{1}{m}\sum_{i=1}^{m}(r_{ni}-r_n) \tag{24}$$

We let coefficient $\lambda_r=1/m$ in the sense of simple average other than $\lambda_r=1/(2\pi)$ in Eq. (18) after strict calculation. This is because the scattered points are arbitrarily distributed, and we do not really know the weight of every force (or the force from every point applying to $C_{A,n}$) in the overall drawing force to an approximate circle $C_{A,n}$. And this will lead to the disadvantage for the process of iteration, that is to say, we will need more iterations (instead of only one step like the case in Fig. 5) to get to accurate enough approximation circle $C_{A,n}$ of parameters (x_n, y_n, r_n).

Similarly, we have

$$F_{xn}=\sum_{i=1}^{m}(r_{ni}-r_n)\cos\theta_i \tag{25}$$

$$F_{yn}=\sum_{i=1}^{m}(r_{ni}-r_n)\sin\theta_i \tag{26}$$

where $\theta_i=\arctan[2(y_i-y_n, x_i-x_n)]$, is the polar angle of the i-th point (x_i, y_i) corresponding to r_{ni} in Eq. (22). The increments of center coordinates in horizontal and vertical directions are

$$\Delta x_n = \lambda_x F_{xn} = \frac{1}{m}\sum_{i=1}^{m}(r_{ni} - r_n)\cos\theta_i \tag{27}$$

$$\Delta y_n = \lambda_y F_{yn} = \frac{1}{m}\sum_{i=1}^{m}(r_{ni} - r_n)\sin\theta_i \tag{28}$$

We rewrite the iteration formula of circle fitting for scattered points as follows:

$$\begin{cases} x_{n+1} = x_n + \Delta x_n \\ y_{n+1} = y_n + \Delta y_n \\ r_{n+1} = r_n + \Delta r_n \end{cases} \tag{29}$$

and

$$\begin{cases} \Delta x_n = \dfrac{1}{m}\sum_{i=1}^{m}\alpha_n(i)\cos\theta_i \\ \Delta y_n = \dfrac{1}{m}\sum_{i=1}^{m}\alpha_n(i)\sin\theta_i \\ \Delta x_n = \dfrac{1}{m}\sum_{i=1}^{m}\alpha_n(i) \end{cases} \tag{30}$$

where $\alpha_n(i)$ is defined in Eqs. (21) and (22).

Definition 3: We define the following Lyapunov function to show the convergent rate of the iteration:

$$V_n = \Delta x_n{}^2 + \Delta y_n{}^2 + \Delta r_n{}^2 \tag{31}$$

V_n is nonnegative, and upper unbounded.

According to the iteration Eqs. (29), (30), we have the following algorithm. The algorithm will stop when V_n is small enough (say, less than prescribed ε_0) or the iterative number n is big enough (say, greater than the prescribed integer n_0).

Algorithm 1:

1) Set iterative variable $n \leftarrow 0$, and assume an initial approximate circle (x_n, y_n, r_n);

2) Calculate increments Δx_n, Δy_n, Δr_n with Eq. (30);

3) Update the approximate circle: $x_n \leftarrow x_n + \Delta x_n$, $y_n \leftarrow y_n + \Delta y_n$, $r_n \leftarrow r_n + \Delta r_n$;

4) Calculate the convergent rate: $V_n = \Delta x_n^2 + \Delta y_n^2 + \Delta r_n^2$ and undate $n \leftarrow n+1$;

5) if $V_n < \varepsilon_0$ or $n > n_0$ then

End the iteration and output the estimated circle (x_n, y_n, r_n)

else goto 2)

end if

3.2 Convergence and accuracy

Lemma 1: Algorithm 1 is convergent to some result monotonically.

To prove this lemma, observing the Lyapunov function (Eq. (31)), we are going to show that $\delta(V_n)$, the change of V_n, is non-positive under the process of **Algorithm 1**, which is the point of **Lemma 1**.

From Eq. (31), we know

$$\delta(V_n) = 2\cdot\Delta x_n\cdot\delta(x_n) + 2\cdot\Delta y_n\cdot\delta(y_n) + 2\cdot\Delta r_n\cdot\delta(r_n) \tag{32}$$

Because

$$\alpha_n(i) = r_{ni} - r_n = \sqrt{(x_i - x_n)^2 + (y_i - y_n)^2} - r_n \tag{33}$$

We have the change of $\alpha_n(i)$:

$$\begin{aligned}\delta(\alpha_n(i)) = {} & \frac{-(x_i - x_n)}{\sqrt{(x_i - x_n)^2 + (y_i - y_n)^2}}\Delta x_n - \\ & \frac{(y_i - y_n)}{\sqrt{(x_i - x_n)^2 + (y_i - y_n)^2}}\Delta y_n - \Delta r_n = \\ & -\cos\theta_i\cdot\Delta x_n - \sin\theta_i\cdot\Delta y_n - \Delta r_n\end{aligned} \tag{34}$$

Here, we use the definition of θ_i of Eqs. (25) and (26). So we have the changes of Δx_n, Δy_n and Δr_n as follows:

$$\begin{aligned}\delta(\Delta x_n) = {} & \frac{1}{m}\sum_{i=1}^{m}\delta(\alpha_n(i))\cos\theta_i = \\ & \frac{1}{m}\sum_{i=1}^{m}(-\cos^2\theta_i\Delta x_n - \sin\theta_i\cos\theta_i\Delta y_n - \\ & \Delta r_n\cos\theta_i)\end{aligned} \tag{35}$$

$$\begin{aligned}\delta(\Delta y_n) = {} & \frac{1}{m}\sum_{i=1}^{m}\delta(\alpha_n(i))\sin\theta_i = \\ & \frac{1}{m}\sum_{i=1}^{m}(-\cos\theta_i\sin\theta_i\Delta x_n - \sin^2\theta_i\Delta y_n - \Delta r_n\sin\theta_i)\end{aligned} \tag{36}$$

$$\begin{aligned}\delta(\Delta r_n) = {} & \frac{1}{m}\sum_{i=1}^{m}\delta(\alpha_n(i)) = \\ & \frac{1}{m}\sum_{i=1}^{m}(-\cos\theta_i\Delta x_n - \sin\theta_i\Delta y_n - \Delta r_n)\end{aligned} \tag{37}$$

And substitute the changes of Δx_n, Δy_n, Δr_n above into Eq. (32), we have

$$\delta(V_n) = -\frac{2}{m}\sum_{i=1}^{m}(\Delta x_n\cos\theta_i - \Delta y_n\sin\theta_i + \Delta r_n)^2 \le 0 \tag{38}$$

So, **Lemma 1** gets proved.

Remark 1: The changes of the parameters of approximate circle (x_n, y_n, r_n) drive the dynamic system to go ahead, and the Lyapunov function V_n in the trajectory of system process goes not-up because $\delta(V_n)\le 0$. The probability of $\delta(V_n)=0$ without $\Delta x\to 0$, $\Delta y\to 0$, $\Delta r\to 0$, is zero in practical sense, so the system will certainly get rest with $\Delta x_n\to 0$, $\Delta y_n\to 0$, $\Delta r_n\to 0$ and $V_n=0$.

Remark 2: The accuracy of approximation of the result of **Algorithm 1** in its theoretical rest is the same as the result of Lest squares fitting. This is because the objective function in $S(x_n, y_n, r_n)$ Eq. (1) can be rewritten

as

$$S(x_n, y_n.r_n) = \sum_{i=1}^{m}(\sqrt{(x_i - x_n)^2 + (y_i - y_n)^2} - r_n)^2 = \sum_{i=1}^{m}(\alpha_n(i))^2 \quad (39)$$

and

$$\delta(S) = \sum_{i=1}^{m}(2\alpha_n(i)\cdot\delta(\alpha_n(i)) = -2\sum_{i=1}^{m}\alpha_n(i)(\Delta x_n\cos\theta_i + \Delta y_n\sin\theta_i + \Delta r_n) \quad (40)$$

So, if $\Delta x_n \to 0$, $\Delta y_n \to 0$, $\Delta r_n \to 0$ and $V_n=0$, the system comes to its rest, then $\delta(S)=0$, which means that the objective function $S(x_n, y_n, r_n)$ gets its minimum.

Remark 3: Theoretically, the problem of least squares fitting with objective function (1) has not unique solution according to Ref. [17], so we say "convergent to some result" in **Lemma 1**. We use **Algorithm 1** to the example data points given by Ref. [17] to show this claim. See Example 4 in next subsection. It is also said in Ref. [17] that "if the data points are generated randomly with a continuous probability distribution, then the probability that the objective function has multiple minima is zero". So practically, the algorithm will approach to the only minimum of objective function in the meaning of least squares, which means the best fitting of the given scattered points.

3.3 Examples

Example 1: The scattered points are actually points on a circle (0, 0, 3) which means a circle centered at origin (0, 0) with radius of 3. The initial circle is chosen to be (2, 1, 2.5), and Fig. 7 and Fig. 8 show the process of iteration. Figure 7 shows how the center coordinates (x_n, y_n), radius r_n and Lyapunov function V_n approach to the destination. We can see in the process V_n goes down monotonically as **Lemma 1** predicts. The approaching process of approximate circle is visually shown in Fig. 8 where the small circles indicate the trajectory of centers $C_{A,n}$ in every step of iteration.

We slightly change the scattered points such that the points are not share a common circle. We do it simply with Matlab commands

t_1=0:1/2:2p_i; x=3cos(t_1+0.1); y=3sin(t_1)

by adding a small value of angle in cosine of the formula for the calculation of x coordinate. The visual process of

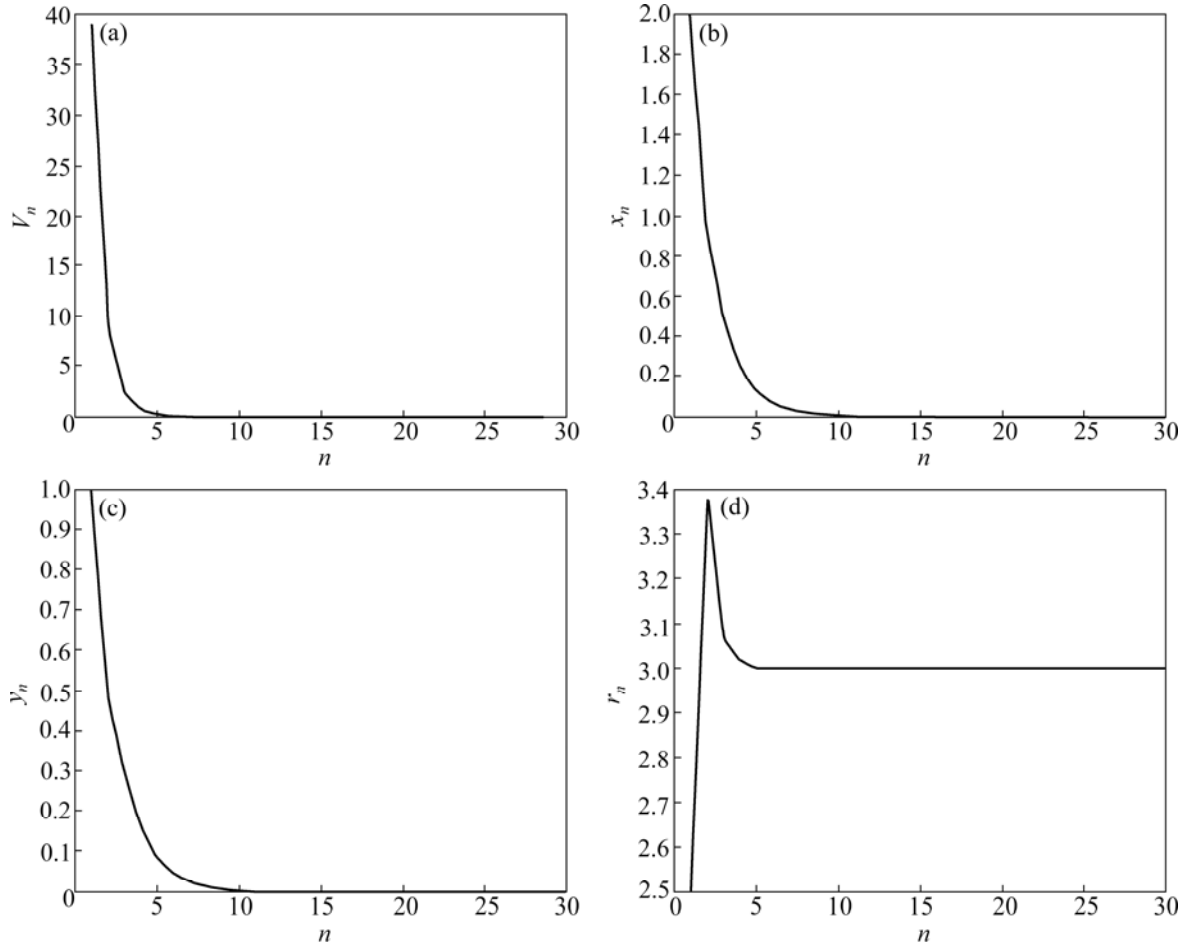

Fig. 7 Changes of parameters of approximate circles in process of iteration for Example 1: (a) V_n; (b) x_n; (c) y_n; (d) r_n

iteration is shown in Fig. 9. V_n also goes down consistently.

Example 2: We take the scattered points from Ref. [14] (x=[1 2 5 7 9 3]; y=[7 6 8 7 5 7]). The fitting result given by Ref. [14] through minimizing the geometric distance as shown in Eq. (1) is the circle with center coordinates (4.739 8, 2.983 5) and radius 4.714 2. And the minimized geometric distance S_{min}= 1.227 6. With the iterative algorithm given above, starting from the initial circle (1, −0.5, 3), the system approaches to the rest with the approximate circle parameters (4.739 6, 2.982 3, 4.715 2), or the center coordinates (4.739 6, 2.982 3) and radius 4.715 2. The Lyapunov function comes to the value of V_n=1.227 6. The process of iteration is very slow which can be seen in Fig. 10, and Fig. 11.

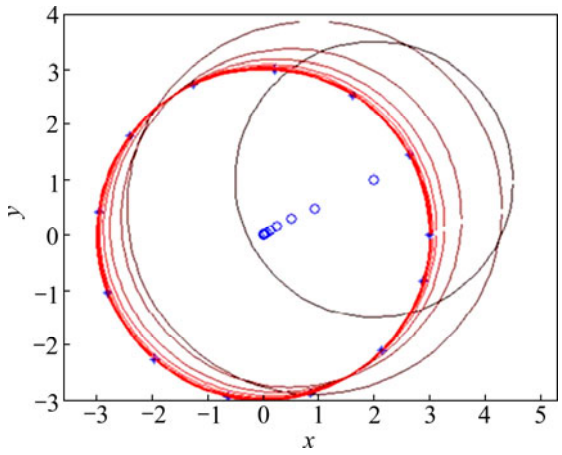

Fig. 8 Visual process of iteration for Example 1

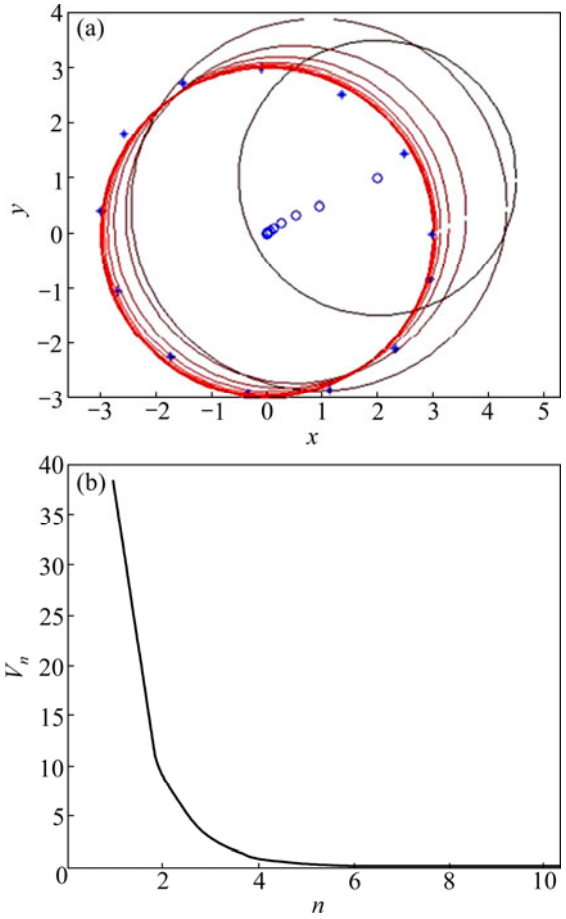

Fig. 9 Process of iteration for Example 1 with slightly changed points

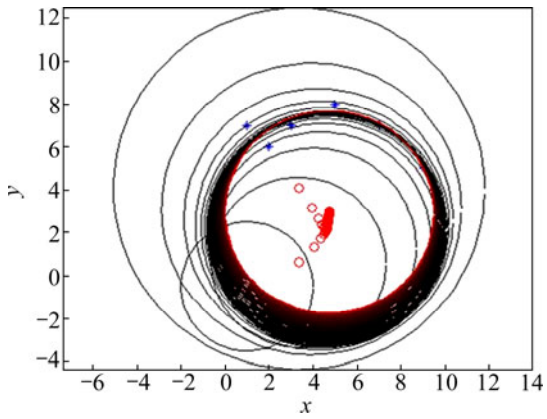

Fig. 10 Process of iteration for Example 2 with points from Ref. [14]

Example 3: We take the scattered points from Ref. [23], which are eight points from the parabola $y=x^2$ (x=0:0.5:3). The result given by Ref. [23] through a closed formula derived from minimizing the geometric distance as shown in Eq. (1) is the circle with center coordinates (−11.839, 8.446 4) and radius 14.686. And the value of objective function is S_{min}=84.949 4. With the iterative algorithm given above, starting from the initial circle (1.0, 1.5, 3), the iteration comes out with an approximate circle with parameters (−14.082 0, 9.144 6, 16.978 6), and the objective function S_{min}=0.116 5. From Figs. 12 and 13, we can see that the convergence rate is really slow, and the system does not really get to the rest because the parameters have a tendency of going further (in Fig. 13).

Example 4: We take five scattered points from Ref. [17] which are (±1, 0), (0, ±1) and (0, 0) shown in Fig. 14 with stars. These points are constructed to show the possibility of multiple minima of the objective function in 1. The fitting process and results are shown in Fig. 14 with four different initial circle resulting in four different fitting with the same value of minimum objective function $S_{min}(x_n, y_n, r_n)$. The processes of approximate circles are shown with darker to redder line from initial to final, and the small circles show the traces of centers of circles in the process.

4 Use of method for circle detection in grayscale images

In practical use of circle detection, the images are

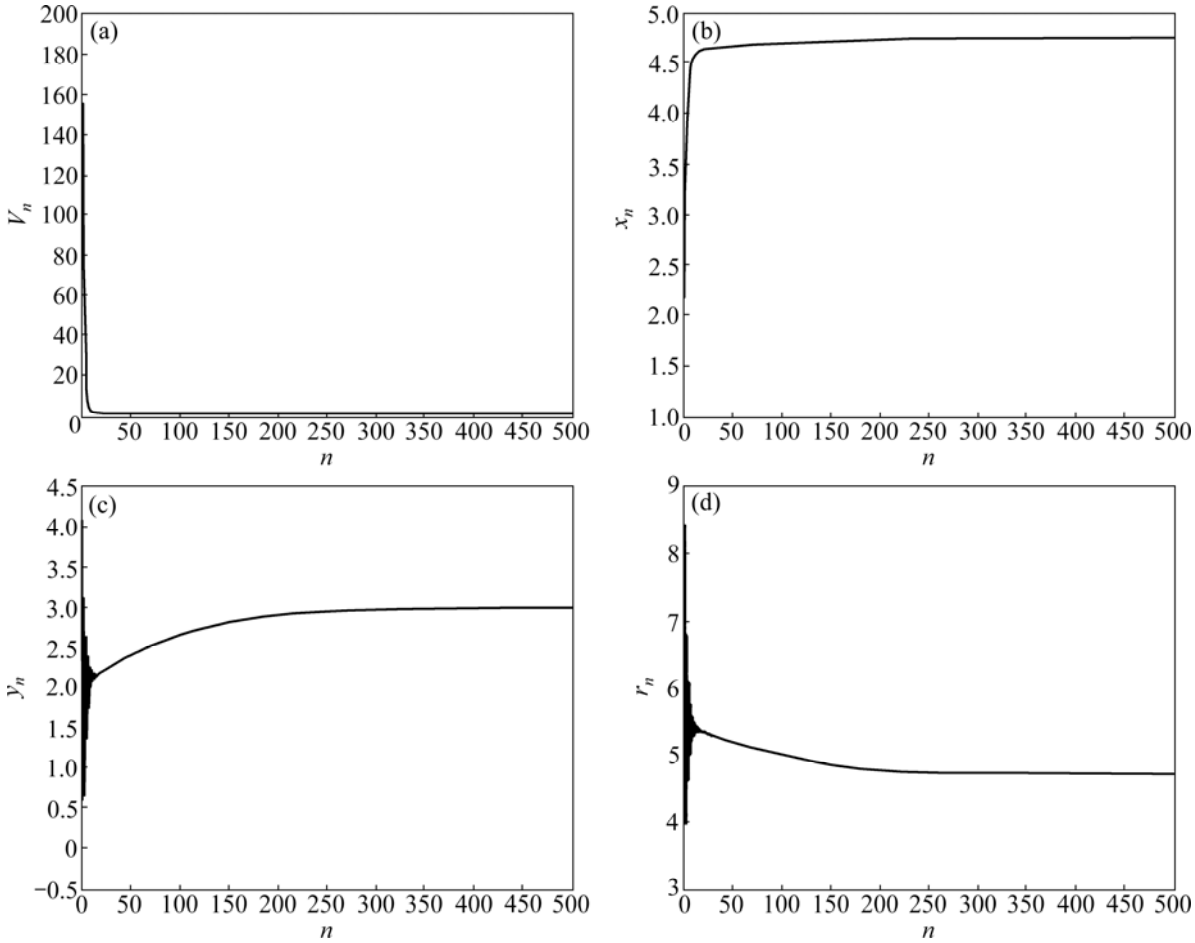

Fig. 11 Changes of parameters of approximate circles in process of iteration for Example 2: (a) V_n; (b) x_n; (c) y_n; (d) r_n

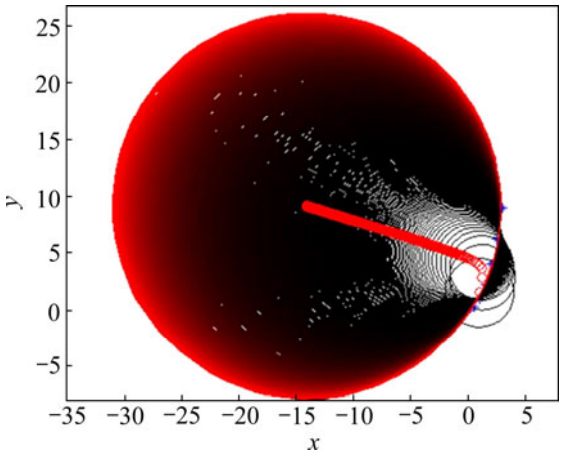

Fig. 12 Process of iteration for Example 3 with points from Ref. [23]

usually from photographs taken with cameras, and more likely after preprocessing as edge detection. We will see how to use the algorithm proposed above in this situation.

4.1 Calculating AF $\alpha_n(\theta)$ in a grayscale image

Figure 15(a) shows a grayscale image with a blurred edge of a big circle and many noisy parts. We are going to use the algorithm proposed above to detect the target circle C_T, which is a virtual circle best fitting the big blurred circular edge in the image, for its center coordinates and radius. Suppose somehow we have got an approximate circle $C_{A,n}$ shown in Fig. 15(b). We would like to think that every image pixel would apply a force to $C_{A,n}$ and the forces by candidate edge pixels of C_T would overweigh the forces by other pixels which are supposed to be noise. To decrease the amount of calculation, we only consider the forces by the pixels inside an annular area which is formed with two concentric circles by putting $C_{A,n}$ just in between. Figure 15(b) shows an annular area of calculation with the width of $2w$ from $-w$ to w based on the perimeter of $C_{A,n}$. For the overall force in the radial direction of θ (shown in Fig. 15(b)), the attracting factor is defined as

$$\alpha_n(\theta)=\frac{\int_{-w}^{+w}P(r_n+t,\theta)t\mathrm{d}t}{\int_{-w}^{+w}P(r_n+t,\theta)\mathrm{d}t} \tag{41}$$

where $\alpha_n(\theta)$ can be explained as the drawing effect from the centroid of pixels in the line segment cutting the annular area in the radial direction of θ, if we think that

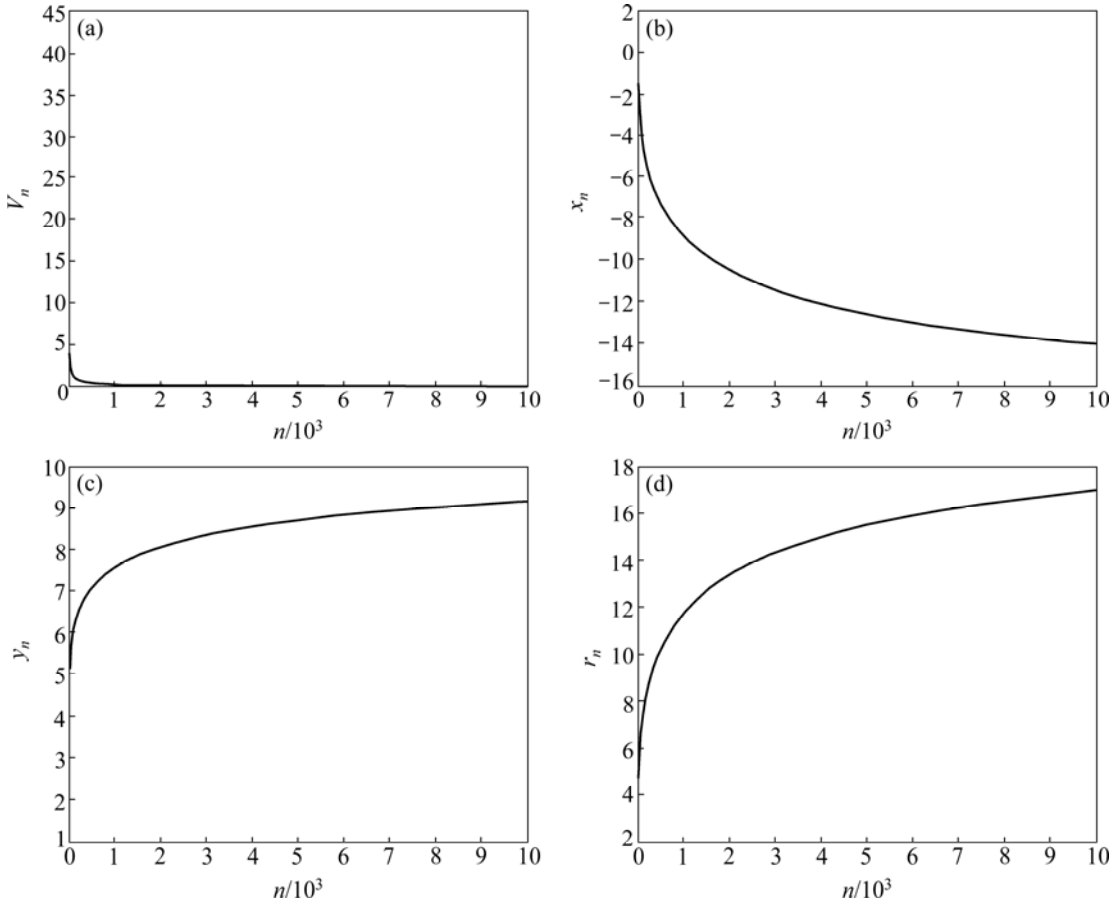

Fig. 13 Changes of parameters of approximate circles in process of iteration for Example 3: (a) V_n; (b) x_n; (c) y_n; (d) r_n

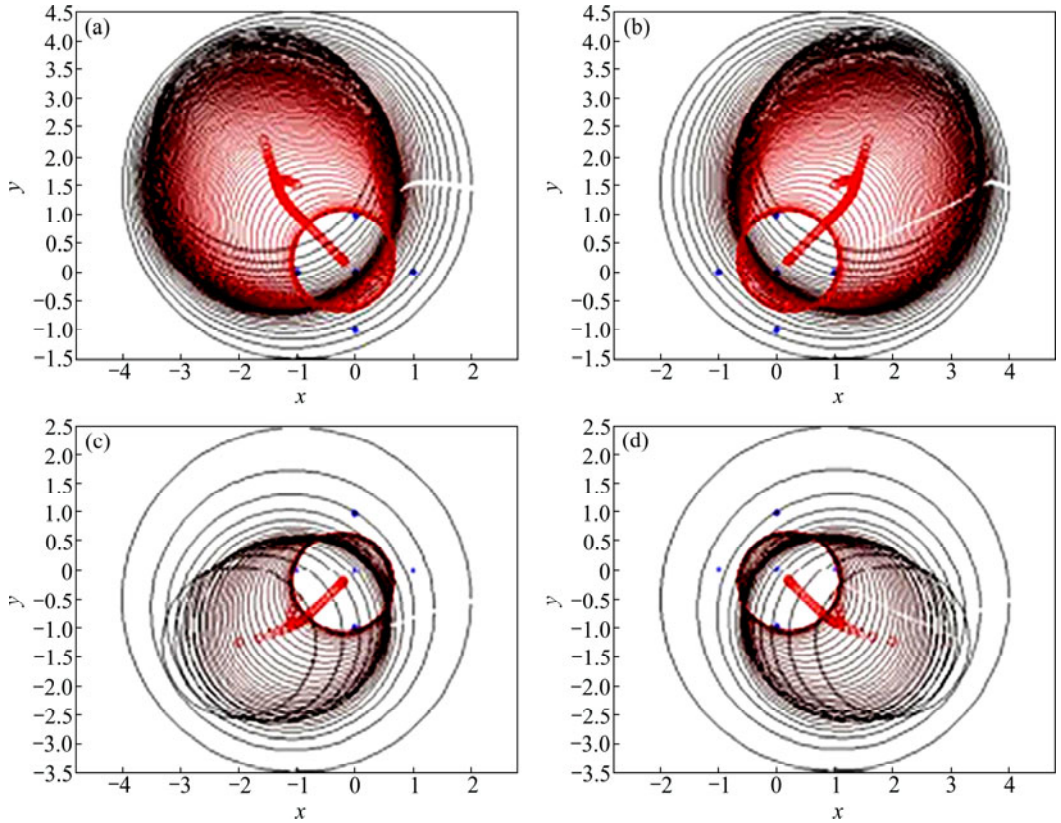

Fig. 14 Example of multiple minima: (a) Initial circle (−1, 1.5, 3), S_{min}=0.588 9; (b) Initial circle (1, 1.5, 3), S_{min}=0.588 9; (c) Initial circle (−1, −0.5, 3), S_{min}=0.588 9; (d) Initial circle (1, −0.5, 3), S_{min}=0.588 9

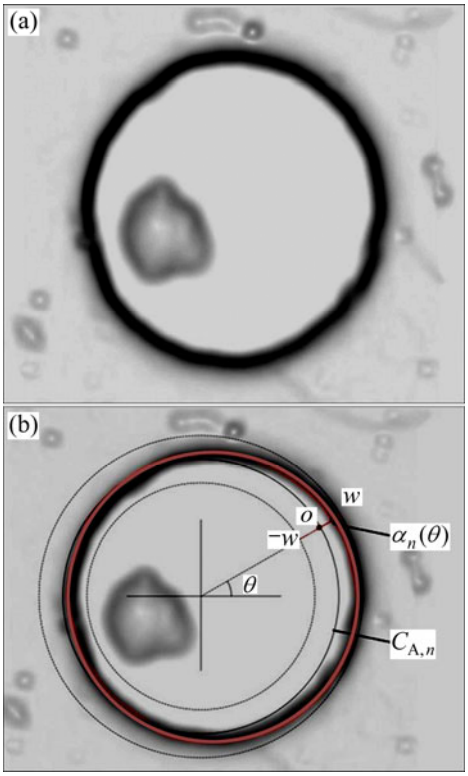

Fig. 15 Calculating attracting factor $\alpha_n(\theta)$ in a grayscale image: (a) A grayscale image with a blurred edge; (b) Illustration of calculating $\alpha_n(\theta)$

$P(r_n+t, \theta)$ is the "weight" of the pixel at polar coordinates (r_n+t, θ), and that the darker of the pixel, the heavier, and the larger of the value of $P(r_n+t, \theta)$, which means the stronger force of the pixel would be applied to $C_{A,n}$. Equation (41) is also consistent with **Definition 1** for $\alpha_n(\theta)$. This would be clear if we think of the image we used in Section 2 as black and white (or binary) and there was only one black pixel in a radial direction of $C_{A,n}$.

The red circular curve in Fig. 15(b) is the centroid curve calculated with Eq. (41) along the perimeter of $C_{A,n}$. If we drew arrows like Fig. 1 from $C_{A,n}$ to the centroid curve, we would see that the forces (arrows) would attract $C_{A,n}$ to the right, which is in the correct direction simply by viewing.

With CAF calculated with Eq. (41), the iterative formulas will be just like Eqs. (17) and (18). We omit the repeat writing here.

4.2 Algorithm for method

For digital grayscale images, Eq. (41) can be calculated by

$$\alpha_n(i)=\frac{\sum_{t=-w}^{w}P(r_n+t,\theta)\cdot t}{\sum_{t=-w}^{w}P(r_n+t,\theta)} \tag{42}$$

which is straight forward. The summation is from $-w$ to w in a radial direction of θ (Fig. 15(b)) represented by i in Eq. (42) with step of 1, which means that we do the summation with pixels one by one in a direction. The counted width is $2w$ around a point on the perimeter of $C_{A,n}$. The counter i in Eq. (42) is from 0 to the nearest integer to $2\pi r_n\cdot\alpha_n(i)$ for each i represents the CAF in the direction of $\theta=1/r_n$, so the angular step for the calculation of CAF along the perimeter of an approximate circle with radius r_n is $1/r_n$, which is a pixel by pixel scanning along a circular bound.

Correspondingly, the increment formulas can be written as

$$\begin{cases}\Delta x_n=\dfrac{1}{\pi}\sum_{i=0}^{2\pi r_n}\alpha_n(i)\cos\dfrac{i}{r_n}\\ \Delta y_n=\dfrac{1}{\pi}\sum_{i=0}^{2\pi r_n}\alpha_n(i)\sin\dfrac{i}{r_n}\\ \Delta r_n=\dfrac{1}{\pi}\sum_{i=0}^{2\pi r_n}\alpha_n(i)\end{cases} \tag{43}$$

The algorithm design is just like **Algorithm 1** in Section 2 except using Eqs. (42) and (43) instead of Eqs. (21), (22) and (30) for the calculation of $\alpha_n(i)$ and increments of parameters. Also we need a inner-loop for the calculation of $\alpha_n(i)$ for each i.

4.3 Examples

Here are some experimental results to show the effectiveness of the algorithm for circle detection in several situation of practical use of the algorithm. The images we use are digital grayscale images with the resolution of 500×500 pixels. $P(t,i)$ in Eq. (42) is plugged with the grayscale value at the corresponding pixel because the images are all black in background, and that the whiter, the bigger of the force the pixel would apply to an approximate circle. The width $2w$ in Eq. (42) is set to 160 pixels, which is the width of annular area in Fig. 15(b). The summation steps for Eqs. (42) and (43) is set to 3 instead of 1 (pixel by pixel) to deduce the amount of calculation. The algorithm is implemented with LabVIEW running on a laptop with Inter Core 2 Duo T6600 2.20 GHz CPU.

Example 1 (Noise sensitivity): The circle to be detected is shown in Fig. 16(a) with a thick white solid line in the pure black background. The green circle in Fig. 16(a) is the initial circle assumed arbitrarily, and only 5 steps of iteration with 29 ms of time consuming come to the end of implementation of algorithm. The red circle in Fig. 16(a) shows the last approximate circle. In Fig. 16(b), we add Gaussian noise with variance of 0.01 to the background, and the circles with thin solid white

line show the process of iteration from initial green circle to final red circle. More Gaussian noise with variance of 0.1 and 1 in Fig. 16(c) and Fig. 16(d) results in more iterative steps and more time consuming to get to the final approximate circle which shows good approximation to the target circle.

This example shows that the algorithm is robust to noise, although the speed of convergence is slowed down by more intensive noise on image.

Example 2 (Circle detection with only partial arc of circle): The thick white solid curve in Fig. 17(a) is only part of a circle, one sixth actually. From the initial green circle to the final red circle only takes 6 steps of iteration. An arbitrary initial circle cutting only part of a 120° arc of a target circle in Fig. 17(b) also converges to the target circle after 22 steps of iteration.

Example 3 (Circle detection for BGA inspection): Figure 18(a) is a cropped image photographed a PCB

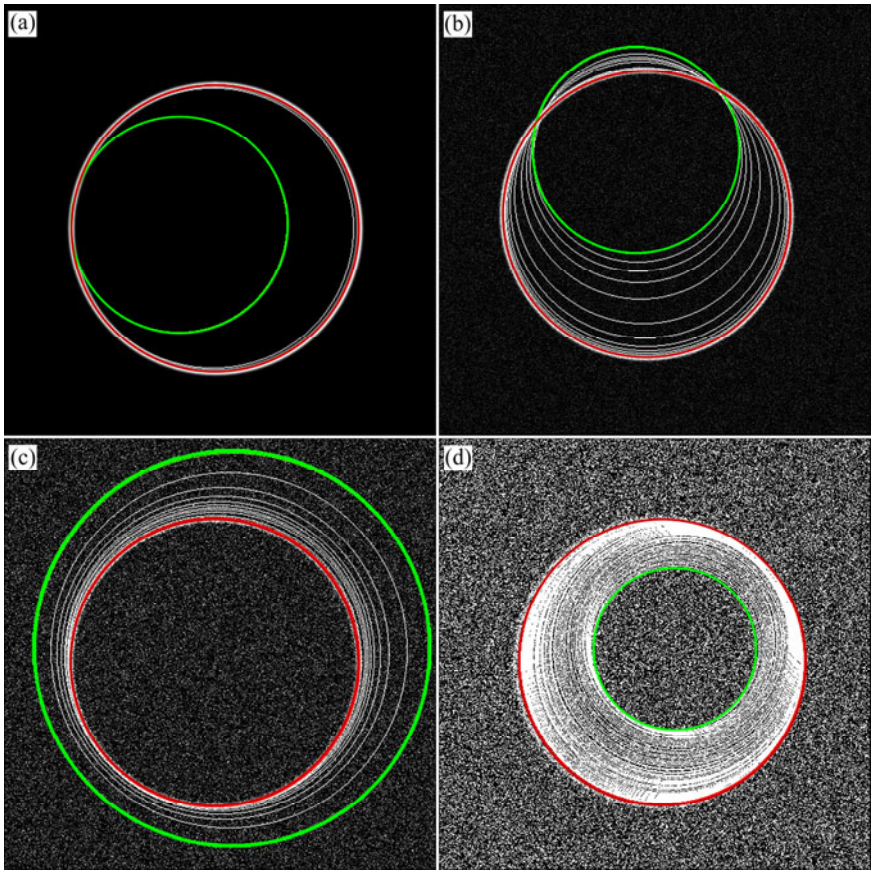

Fig. 16 Performance of algorithm to image with Gaussian noise: (a) Background without noise (Iterative step: 5, time: 29 ms); (b) With noise variance of 0.01 (Iterative step: 12, Time: 99 ms); (c) With noise variance of 0.1 (Iterative step: 11, Time: 118 ms); (d) With noise variance of 1 (Iterative step: 78, Time: 806 ms)

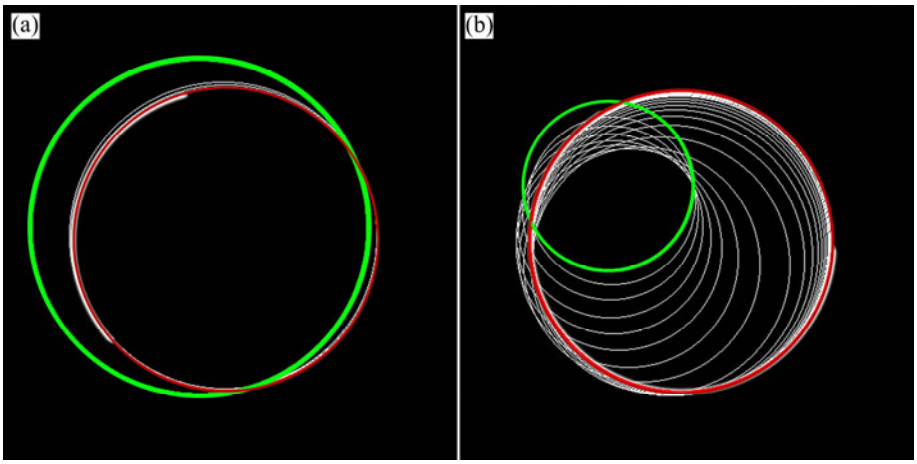

Fig. 17 Circle detection with only part of arc: (a) 60° of arc (Iterative step: 6, Time: 45 ms); (b) 120° of arc (Iterative step: 22, Time: 196 ms)

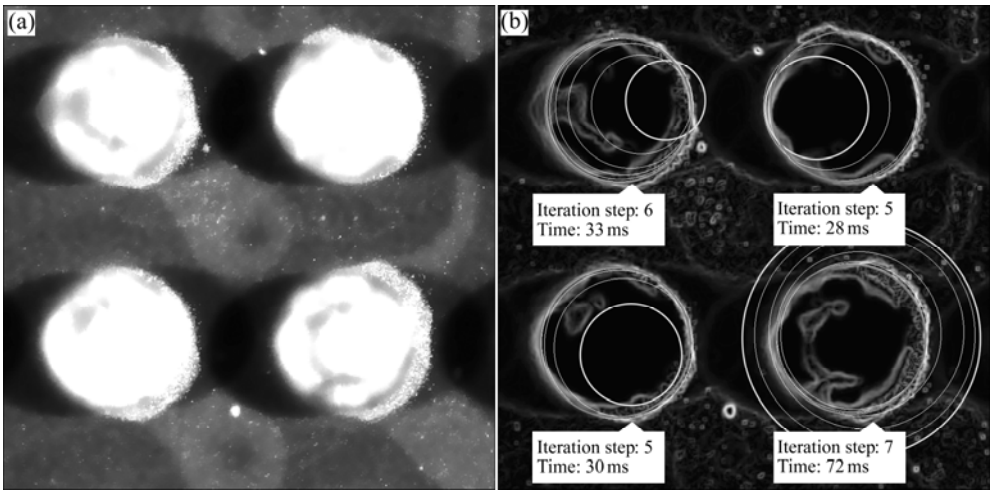

Fig. 18 Circle detection for BGA inspection: (a) Cropped image of PCB with BGA; (b) Circle detected after preprocessing of image

with ball grid array (BGA) for the inspection of BGA balls. We are going to measure the radii of balls. To this end, we process the image with a method of edge detection through calculating the gradient of pixel values. The circle detection algorithm proposed is used for each ball separately, and the results are shown in Fig. 18(b). We intentionally choose the initial green circles in relatively different positions, and the final approximate red circles match the circular edges very well.

5 Conclusions

1) The idea of the iterative method proposed is intuitive, coming from an imaginary drawing force applied to an assumed approximate circle ($C_{A,n}$) from the imaginary target circle (C_T). The concept of circular attracting factor (CAF) $\alpha_n(\theta)$ is defined and the overall forces for radius (F_{rn}) which would make the radius in next step better fit the radius of C_T, and for center position (F_{xn} and F_{yn} which would make the center coordinates in next step better fit the center of C_T) are also defined. The formulas to describe the idea are given.

2) The algorithm for circle fitting to scattered point based on the idea above is presented. The convergence of the algorithm is discussed, which shows that the algorithm converges monotonically to a point which is in accord with the equilibrium with minimum objective function in the sense of geometric least squares fitting. Theoretically, there are more than one equilibrium in the sense of geometric least squares fitting, but the possibility of multiple minima is zero if the data are generated randomly. Examples are given to show the effectiveness of the algorithm, including an example with designed points to show the possibility of multiple minima which can be approached with the algorithm under different initial circles.

3) The algorithm is also adopted to circle detection in grayscale images which often arises in machine vision. The point is the calculation of CAF $\alpha_n(\theta)$ in a grayscale image with noise, based on the idea that all pixels would apply a force to an approximate circle but the forces from potential useful edge pixels would overweigh the forces from other pixels. To deduce the burden of calculation, measures are taken in the algorithm, for example, pixels in an annular area around an approximate circle are used to calculate $\alpha_n(\theta)$ other than all the pixels in the image, and even further, not all but samples of pixels in the annular area are considered in the calculation of $\alpha_n(\theta)$. Examples are given to show that circle detection with this algorithm in grayscale images is robust to noise. Because there is no need to transfer grayscale image into binary image, the algorithm is less sensitive to lighting and background noise.

References

[1] RIHA K, BENES R. Detection in pulsative medical video sequence [C]// Signal Processing (ICSP2010), 2010 IEEE 10th International Conference on. Beijing, China: IEEE, 2010: 674−677.

[2] JARJES A A. Iris localization: Detecting accurate pupil contour and localizing limbus boundary [C]// Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on. Wuhan, China: IEEE, 2010: 349−352.

[3] JARJES A A. Improved greedy snake model for detecting accurate pupil contour [C]// Advanced Computer Control (ICACC), 2011 3rd International Conference on. Harbin, China: IEEE, 2011: 515−519.

[4] CANNY J. A computational approach to edge detection [J]. Analysis and Machine Intelligence, 1986, 8(6): 679−714.

[5] DAVIS L S. A survey of edge detection techniques [J]. Computer Graphics and Image Processing, 1975, 4(3): 248−260.

[6] DONG W, SHISHENG Z, Color image recognition method based on the Prewitt operator [C]// Computer Science and Software Engineering, 2008 International Conference on. Wuhan, China: IEEE, 2008: 170−173.

[7] LANSER S, ECKSTEIN W. A modification of deriche's approach to edge detection [C]. Pattern Recognition, 1992. Vol. III. Conference C:

Image, Speech and Signal Analysis, 11th IAPR International Conference on. The Hague, Netherlands: IEEE, 1992: 633−637.

[8] BALLARD D H. Generalizing the hough transform to detect arbitrary shapes [J]. Pattern Recognition, 1981, 13(2): 111−122.

[9] KIRYATI N. A probabilistic hough transform [J]. Pattern Recognition, 1991, 24(4): 303−316.

[10] SHAKED D. Deriving stopping rules for the probabilistic hough transform by sequential analysis [J]. Computer Vision and Image Understanding, 1996, 63(4): 512−526.

[11] XU L. A new curve detection method-randomized hough transform (rht) [J]. Pattern Recognition Letters 1990, 11(5): 331−338.

[12] CHATZIS V, PITAS I. Fuzzy cell hough transform for curve detection [J]. Pattern Recognition, 1997, 30(12): 2031−2042.

[13] HAN J H. Fuzzy hough transform [J]. Pattern Recognition Letters, 1994, 15(7): 649−658.

[14] GANDER W, GOLUB G H, STREBEL R. Least-squares fitting of circles and ellipses [J]. BIT Numerical Mathematics, 1994, 34(4): 558−578.

[15] SAMPSON P D. Fitting conic sections to very scattered data—An iterative refinement of the bookstein algorithm [J]. Computer Graphics and Image Processing 1982, 18(1): 97−108.

[16] FITZGIBBON A. Direct least square fitting of ellipses [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999, 21(5): 239−252.

[17] CHERNOV N, LESORT C. Least squares fitting of circles [J]. Journal of Mathematical Imaging and Vision, 2005, 23(3): 239−252.

[18] JIA L, PENG C, LIU H. A fast randomized circle detection algorithm [C]// Image and Signal Processing (CISP), 2011 4th International Congress on. Shanghai, China: IEEE, 2011: 820−823.

[19] SHANG F, LIU J. An improved circle detection method based on right triangles inscribed in a circle [C]// Computer Science and Information Engineering, 2009 WRI World Congress on. Los Angeles, USA: IEEE, 2009: 382−387.

[20] BARWICK D S. Very fast best-fit circular and elliptical boundaries by chord data [J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2009, 31(6): 1147−1152.

[21] CHERNOV N. Circular and linear regression: Fitting circles and lines by least squares [M]. London, UK: CRC Press, 2010: 47−49.

[22] AIGNER M, SIR Z, JUTTLER B. Evolution-based least squares fitting using pythagorean hodograph spline curves [J]. Computer Aided Geometric Design 2007, 24(6): 310−322.

[23] BULLOCK R. Least-squares circle fit [EB/OL]. [2006−10]. www.dtcenter.org at:met-users-docs-write ups-circle fit.pdf.

(Edited by YANG Bing)