Pattern Recognition 109 (2021) 107588

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

An occlusion-resistant circle detector using inscribed triangles

Mingyang Zhao [a,c], Xiaohong Jia [a,c,*], Dong-Ming Yan [b,c]

[a] KLMM, NCMIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China
[b] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China
[c] University of Chinese Academy of Sciences, China

ARTICLE INFO

Article history:
Received 4 April 2020
Revised 3 July 2020
Accepted 9 August 2020
Available online 10 August 2020

Keywords:
Circle detection
Inscribed triangle
Parameter estimation
Hough transform

ABSTRACT

Circle detection is a critical issue in pattern recognition and image analysis. Conventional geometry-based methods such as tangent or symmetry are sensitive to noise or occlusion. Area computation is more robust against noise, because it avoids differential calculations. Inspired by this characteristic, we present a novel method for fast circle detection using inscribed triangles. The proposed algorithm, which is robust to noise and resistant to occlusion, first extracts circular arcs by approximating line segments and identifying inflection points and sharp corners. To speed up the computation, irrelevant segments are filtered out through the triangle inequality. Arcs that belong to the same circle are then combined according to the position constraint and the inscribed triangle constraint. The circle parameters are further estimated by inscribed triangles based upon the Theil-Sen estimator and linear error refinement without the dependence of least-square fitting but still with the equivalent accuracy. Finally, candidate circles are verified to prune false positives through an inlier ratio rule, which jointly considers both distance and angle deviations. Extensive experiments are conducted on synthetic images including overlapping circles, and real images from four diverse datasets (three publicly available and one we built). Results are compared with those of representative state-of-the-art methods, and the proposed method is demonstrated to embraces several advantages: resistant to occlusion, more robust to noise, and better performance and efficiency.

1. Introduction

Shape analysis is a critical task in pattern recognition and computer vision. As a primitive geometric shape, circles appear every where in our daily life. Thus, the detection of circular objects has attracted intense attention in the past few decades, with efforts exerted to achieve higher accuracy and faster computational speed [50]. For instance, circle detection is widely used in the fields of spacecraft [9], industrial component measurements [31], pupil location [48], and medical image analysis [6,14]. It can also be used in circular traffic sign detection [4,5] and robotic vision system [40,54].

The reason for the constant emergence of new algorithms is the increasing complexity of the real world. Apart from detection accuracy and time, the major challenges are the presence of noise, cluttered backgrounds, occlusion, distortion, and image flaw. However, compared with human perception, current methods are still far from satisfactory as [47] pointed out. Therefore, the motivation of this work is to improve the robustness to noise and the resistance to occlusion for circular object detection.

The most classical method for circle detection is the Hough transform (HT) [20,25], which converts the estimation of circular parameters to the intersection of multiple 3D conic spaces [39] and where the highest peak in the accumulator is chosen as the final circle. However, HT takes a large memory space and high time consumption due to the voting for each edge point, and it is sensitive to the quantized interval of the accumulator. Noise also causes many false detections, especially when complex backgrounds exist.

Different from the enumeration manner of edge points in HT, randomized circle detection (RCD) [11] selects four edge points at each iteration. Three of these points are then used to define a circle solution, and the fourth point is used to check the compatibility, followed by the evidence collection and verification. Although RCD improves the computational efficacy, it still needs numerous attempts to generate candidate circles, because the probability of sampled points coming from the same circle is small, especially for images with noise or complicated scenarios.

Another line of study exploits the geometric properties of circles, such as tangent [61], gradient [32,35,44], curvature [38,58] and symmetry [24]. However, as [15] reports, gradient is more sensitive to noise than edge points owing to the

* Corresponding author.
E-mail addresses: zhaomingyang16@mails.ucas.ac.cn (M. Zhao), xhjia@amss.ac.cn (X. Jia), yandongming@gmail.com (D.-M. Yan).

https://doi.org/10.1016/j.patcog.2020.107588
0031-3203/

incorporation of differential calculation, which also holds for tangent and curvature. Moreover, the utilization of symmetry is based on the complete circle assumption and is thus unsuitable for occlusion and noise [62].

Inspired from the area computation that has been proven to be more robust to noise [37,41], in this study, we present a novel circle detection method that transforms the estimation of circular parameters to the computation of the triangle area. Specifically, we first design an innovative grouping strategy for arc segments on the basis of the relative position and inscribed triangle constraints, which can effectively handle occluded cases. Next, candidate circles are generated by leveraging inscribed triangles again that are guaranteed from the robust Theil-Sen statistical estimator [51]. This manner is more robust to noise, since area avoids the differentiation calculation. A linear error compensation is then applied to further improve the detection accuracy. Finally, a strict verification process is performed to reliably pick out true circles. Our method avoids conventional least-square fitting algorithms due to instability from incomplete data caused by occlusion or noise disturbance, but still has comparable accuracy as demonstrated by experiments. In a nutshell, the contributions of this paper are

1. a novel circle detector that is robust to noise and resistant to occlusion,
2. a totally geometry-based parameter estimation method through inscribed triangles without the dependence of least-square fitting but with equivalent accuracy, and
3. a new real-world dataset that we constructed with the sufficient examination of circle detection algorithms.

Detailed explanations of the above points are presented in the following sections. Section 2 reviews related circle detection algorithms from the perspective of candidate circle generation and verification. The overview of the proposed method is presented in Section 3, followed by the procedure explanations in Section 4. Section 5 reports the experimental results including the (a) evaluation of arc grouping and eccentricity study, (b) parameter threshold analysis, (c) circle detection comparison in terms of performance and speed with five representative state-of-the-art methods in four real-world datasets, and (d) robustness analysis by a series of Gaussian white noise. The conclusions and future work are summarized in Section 6.

2. Related work

HT [25] is a broadly used geometric primitive detector, as summarized in a recent survey [39]. It was first applied in Duda and Hart [20] to detect circles, a method named as circle Hough transformation (CHT). As indicated in Mukhopadhyay and Chaudhuri [39], the computational complexity and memory requirements of CHT are $O(m^3)$ and $O(n^3)$, respectively, where m is the number of pixels and n is the quantized bins.

To improve voting efficiency and reduce storage, a large amount of work has been proposed [3,22,23], such as randomized HT [52,55,56] and probabilistic HT [33,46,60], both of which randomly pick a small number of edge pixels as input for voting to speed up execution. However, these methods are still time consuming especially when noise exists. Yang et al. [57] divide the input image as several specific spaces and adopt a local voting manner. This method is more stable than HT but has more parameters for tuning. Djekoune et al. [18] propose an incremental modification by approximating the conventional trigonometric functions using polynomials. Nevertheless, an iterative step must be introduced. Su et al. [49] replace the accumulative voting of HT by a minimizing approach, which represents each circle through equally distributed segments. However, the symmetric hypothesis and fixed segment number are not suitable for the occluded case. Other variants usually combine HT with geometric properties to reduce storage space. Kimme et al. [32] utilize gradient direction to constrain voting ranges, but this method only processes one circle at a time. Yip et al. [59] and Kerbyson et al. [30] decompose the 3D accumulator to 2D by first locating circle centers followed by resolving radii. However, as [3] points, these methods inevitably bring aliasing artifacts for accumulator arrays.

Different from the voting scheme of HT, RCD methods take a sampling checking process. For instance, Chen et al. [11] directly sample four edge pixels from the image, three of which are used to define a circle followed by a compatibility checking, that is, whether or not the fourth point lies on the circle. If the checking holds, then an inlier computation is further executed. In [13], an improved RCD is proposed by involving multiple evidence checking, which additionally depends on the gradient to recognize candidate circles. However, it also needs numerous attempts to obtain valid sampling for real images, because the probability acquiring sampled edge points from the same circle is small. To enhance efficiency, Jia et al. [26] narrow down the sampling scope to connected edge segments. Marco et al. [16] utilize isophote curvatures [53] to confine sampling points. Similar to [26], Zhang et al. [62] sample points from edge segments and devise an isosceles triangle based on gradient computation to determine circles. Jiang et al. [28] use the difference area of false circles to update the sampling. However, these sampling strategies either involve differential calculation or are influenced by noise.

Another line of work explores geometric properties to improve the circle detection performance. Compared with HT and RCD, these methods are usually more efficient [2]. For instance, Ho et al. [24] rely on symmetry to locate circle centers by horizontally and vertically scanning the input image. Similarly, Rad et al. [44] find circles according to the symmetry of gradient pair vectors. However, these methods easily suffer from occlusion. Kim et al. [31] derive the circle center by solving a linear equation, which is defined by two intersecting chords. Le et al. [34] use the perpendicular bisectors of two chords to determine centers. Although these methods can achieve better robustness against noise, the computational complexity of finding intersecting points is their main drawback [62]. Yuan et al. [61] transform the edge points to a 1D power histogram based on circle power theorem, and the peak is detected to locate circles. In their method, the tangent is calculated. Akinlar et al. [2] utilize line segments to approximate edges by an edge drawing parameter free algorithm [1]. They first pick out edge segments that are composed of at least three consecutive line segments and then cluster arcs on the basis of the center distance by frequently evoking the least-square fitting algorithm. However, when edges are broken as small parts due to occlusion or severe noise, there will emerge large fitting deviations and, therefore, true circles will be lost. Lu et al. [35] adopt a similar manner but add a polarity constraint defined by the gradient orientation followed by two instances of circle fitting. Different from them, our method avoids gradient calculation and directly estimates circular parameters from inscribed triangles without depending on fitting algorithms while still maintaining equivalent accuracy.

3. Overview

Given an input image, the proposed framework follows three steps to detect circles (Fig. 1). An overview of the framework is presented in the following.

3.1. Circular arc extraction

To circumvent tuning parameters of the Canny edge detector [10], we extract the edge map of the input image by a non-

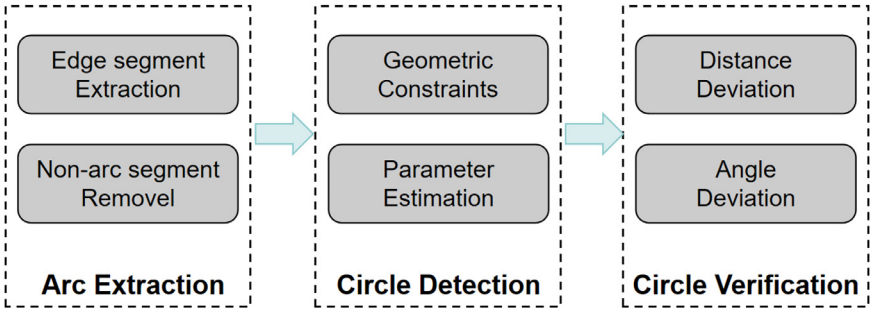
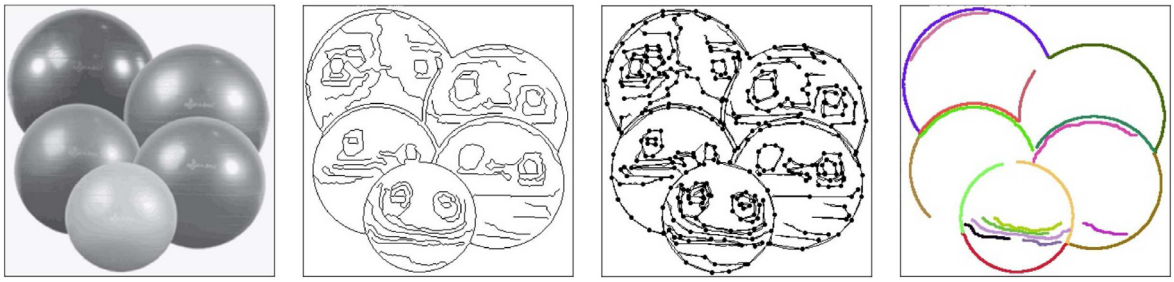


Fig. 1. Flowchart of the proposed circle detection method.



(a) (b) (c) (d)

Fig. 2. Circular arc extraction. From the left to right column: (a) A test image in occlusion; (b) The extracted edge map; (c) The edge curve is approximated by a series of line segments; (d) Finally extracted arcs. Different arcs are denoted by different colors.

parametric method. According to the definition of circular arcs, we then leverage line segments to approximate edges and to obtain circular arcs by splitting these line segments at inflection points or sharp corners. To speed up the following processing, we further remove arcs with low curvature and small length by the triangle inequality constraint and the length constraint, respectively, because they are either too straight or too short to constitute a circle.

3.2. Circle detection

For the extracted arcs, two geometric constraints are introduced to group arcs that belong to the same circle. The first one is the relative position constraint, which actually reflects the convexity of arcs. The second one is based on the inscribed triangles and is thus named as inscribed triangle constraint. After co-circular arcs are identified, inscribed triangles are again utilized together with the robust Theil-Sen estimator to generate candidate circles, followed by a linear error refinement step to further improve the detection accuracy.

3.3. Circle verification

Owing to the inevitable bias caused by discrete pixels, there may exist false circles. To prune false detections and pick out circles with high confidence, we perform an effective verification step that simultaneously takes distance and angle deviations into account.

4. Methodology

4.1. Circular arc extraction

4.1.1. Pre-processing stage

The purpose of the pre-processing stage is to extract the edge curve, which is usually obtained by Canny detector [10]. However, the effect of this stage is influenced by threshold tuning. To avoid this, we adopt the EDPF [1] method to detect edges. EDPF first detects anchors without setting any parameters and then anchors are joined together by drawing an edge curve. The edge curve is further verified by the *a contrario* principle, which is based on the Helmholtz perceptual mechanism and Gestalt theory [17]. A test image and its occluded edge curve are shown in Fig. 2(a) and (b), respectively.

4.1.2. Arc extraction

An edge segment is defined as an arc segment if it turns in the same direction and its turning angles are within a certain threshold. With this definition, arcs can be extracted by recognizing inflection points and sharp corners as illustrated in Fig. 3.

To this end, a series of line segments is applied to approximate the extracted edge curve for representing the curvature characteristic. Ramer-Douglas-Peucker algorithm [19,45] is adopted here due to its accuracy and computational efficiency. As shown in the left side of Fig. 3, an edge curve is approximated by a series of line segments $L = \{l_i \in \mathbb{R}^2\}_{i=1}^{n}$, and their endpoints are denoted as dominant points. The directional vector of a line segment l_i is denoted as $\overrightarrow{l}_i$. Then the end dominant point of $l_i \in L$ is regarded as a sharp corner if

$$\cos\theta_i = \frac{\overrightarrow{l}_i}{\|\overrightarrow{l}_i\|_2} \cdot \frac{\overrightarrow{l}_{i+1}}{\|\overrightarrow{l}_{i+1}\|_2} \geq \cos T_\alpha, \tag{1}$$

where θ_i is the angle between the two consecutive line segments l_i and l_{i+1}, and T_α is the angle threshold. All thresholds will be analyzed in Section 5.2.2. To recognize inflection points, the cross product of the vectors are utilized, that is, if

$$\left(\frac{\overrightarrow{l}_{i-1}}{\|\overrightarrow{l}_{i-1}\|_2} \times \frac{\overrightarrow{l}_i}{\|\overrightarrow{l}_i\|_2}\right) \cdot \left(\frac{\overrightarrow{l}_i}{\|\overrightarrow{l}_i\|_2} \times \frac{\overrightarrow{l}_{i+1}}{\|\overrightarrow{l}_{i+1}\|_2}\right) = -1, \tag{2}$$

then the start dominant point of $\overrightarrow{l}_i$ is seen as an inflection point. Therefore, A is an inflection point while B is a sharp corner in the right side of Fig. 3. Next, edge curves are split at these points to

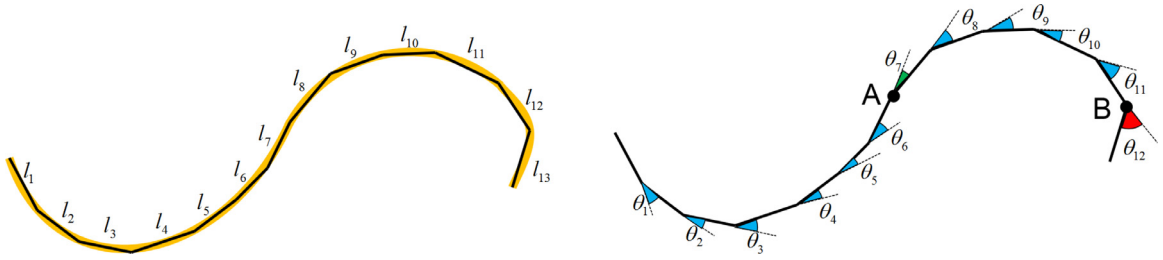


Fig. 3. Determination of inflection points and sharp corners. Left: an edge curve is approximated by thirteen line segments; Right: two arcs are obtained by identifying the inflection point A and the sharp corner B.

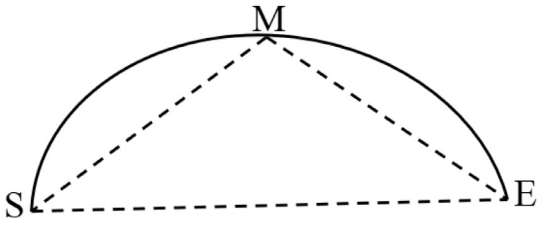


Fig. 4. Removel of line segments based on the triangle inequality.

obtain arc segments. However, there may emerge line or short segments that do not contribute to circle detection. To speed up the following processing, similar in Jia et al. [27], edge segments whose number of pixels less than 16 are simply removed regardless of the image resolution. A strategy based on triangle inequality is then devised to filter out line segments. Specifically, let S, E, and M be the two endpoints and the midpoint of an edge segment in Fig. 4, respectively. It will be deemed as a line segment if

$$\|S-M\|_2+\|M-E\|_2 \simeq \|S-E\|_2.$$

In practice, we evaluate the bend degree τ by

$$\tau = 1-\frac{\|S-E\|_2}{\|S-M\|_2+\|M-E\|_2}. \tag{3}$$

Therefore, $\tau \in (0, 1)$. A larger τ indicates a more curved edge segment, and thus segments whose τ is small are filtered out. For Fig. 2(b), the approximated line segments and extracted circular arcs are presented in Fig. 2(c) and (d), respectively. To speed up execution, the arc whose distance between two endpoints is less than 3 is regarded as a completed circle without the following arc grouping process, but still with the final strict verification.

4.2. Circle detection

For all extracted arcs $\Phi = \{a_i\}_{i=1}^n$, to circumvent the instability of direct circle fitting for broken arcs, we first group the arcs that belong to the same circle together. To shear invalid arc pairs for later processing, we first utilize chords to identify the relative position of two arcs a_i, $a_j \in \Phi$. Then, we retain the arc pairs that satisfy the position constraint and further estimate their centers and radii on the basis of two inscribed triangles. At the same time, inliers regarding arc points are calculated to further confirm cocircular arcs. The motivation is that if the estimated two circles by the two arcs share the similar center and radius parameters, it is highly likely that this arc pair belongs to the same circle.

4.2.1. Relative position constraint

The proposed relative position constraint actually reflects the concavity and convexity relationship between two arcs. Specifically, for each $a_i \in \Phi$, we first compute the line equation l_i joining its two endpoints S_i and E_i (Fig. 5) as below:

$$l_i : A_i x + B_i y + C_i = 0.$$

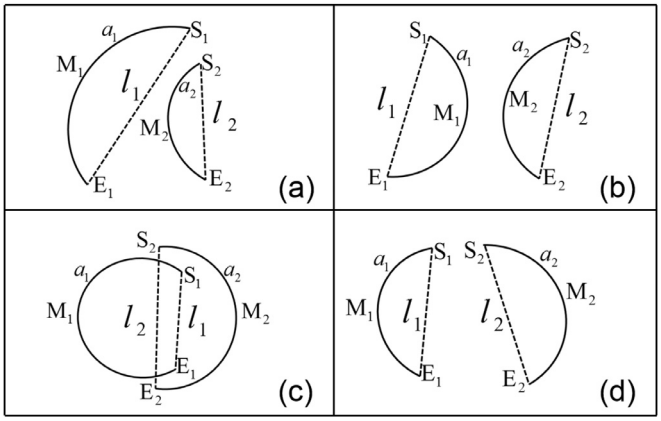


Fig. 5. Relative position constraint for arc grouping. In (a), (b), and (c), the two arcs can not be grouped together, while the two arcs in (d) can be grouped.

We then use

$$s_i(P) := sign(l_i(P)) \in \{+1, -1\}$$

to represent the relative position of the point P about l_i. For another arc a_j, a_i and a_j are said to satisfy the relative position constraint if the formulas

$$\begin{cases} s_i(S_j)\cdot s_i(E_j) > 0 \ \wedge \ s_i(M_i)\cdot s_i(S_j)\cdot s_i(E_j) < 0, \\ s_j(S_i)\cdot s_j(E_i) > 0 \ \wedge \ s_j(M_j)\cdot s_j(S_i)\cdot s_j(E_i) < 0. \end{cases} \tag{4}$$

hold. Here, S_*, E_*, and M_* are the two endpoints and the middle point of the arc a_*, respectively. Eq. (4) indicates that for l_i, S_j and E_j lie in one side of it while M_i lies in the other side, and simultaneously, S_i and E_i lie in one side of l_j while M_j lies in the other side. We enumerate position cases in Fig. 5, and from Eq. (4), we know the case (d) is the only valid one.

4.2.2. Inscribed triangle constraint

Arcs that belong to the same circle must share similar centers and radii. Thus we propose a novel method that utilizes the geometric characteristic of inscribed triangles to estimate circular parameters for each arc. First, arcs are sorted with respect to length in descending order, and then group starting from the longest arc. Arcs that satisfy the relative position constraint with the longest one are picked out. The specific process is presented in the following.

Kindly, see Fig. 6 for reference. Let a_1 and a_2 be two arcs, and their endpoints and midpoints are denoted as S_i, E_i, and $M_i (i = 1, 2)$, respectively. For the triangle $\triangle S_1M_2E_1$, the following formula holds:

$$\begin{cases} \dfrac{|S_1E_1|}{\sin\angle S_1M_2E_1} = \dfrac{|S_1M_2|}{\sin\angle S_1E_1M_2} = \dfrac{|E_1M_2|}{\sin\angle E_1S_1M_2} = 2R_1, \\ S_{\triangle S_1M_2E_1} = \dfrac{1}{2}|S_1M_2||E_1M_2|\sin\angle S_1M_2E_1. \end{cases} \tag{5}$$

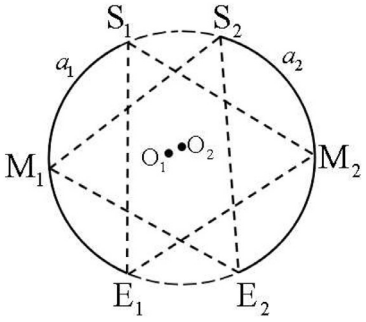


Fig. 6. Estimation of centers and radii by inscribed triangles $\triangle S_1M_2E_1$ and $\triangle S_2M_1E_2$, where a_1 and a_2 are two circular arcs, S_i, E_i, and M_i $(i = 1, 2)$ are the start point, end point, and middle point of them. The estimated centers are O_1 and O_2.

where $|^*|$ denotes the length of the edge *, and $S_{\triangle S_1M_2E_1}$ denotes the area of $\triangle S_1M_2E_1$. After a similar process is conducted for $\triangle S_2M_1E_2$, the estimated radius of a_1 and a_2 can be obtained as

$$R_1 = \frac{|S_1M_2||M_2E_1||E_1S_1|}{4S_{\triangle S_1M_2E_1}}, R_2 = \frac{|S_2M_1||M_1E_2||E_2S_2|}{4S_{\triangle S_2M_1E_2}}. \tag{6}$$

In addition to radii, the corresponding centers of a_1 and a_2 are further estimated. Let the coordinates of S_1, M_2, and E_1 be (x_{S_1}, y_{S_1}), (x_{M_2}, y_{M_2}) and (x_{E_1}, y_{E_1}), and the estimated circular center of a_1 be $O_1(x_{O_1}, y_{O_1})$. Then it holds that

$$\begin{aligned} x_{O_1} &= \frac{\begin{vmatrix} x_{M_2}^2 + y_{M_2}^2 - x_{S_1}^2 - y_{S_1}^2 & (y_{M_2} - y_{S_1}) \\ x_{E_1}^2 + y_{E_1}^2 - x_{M_2}^2 - y_{M_2}^2 & (y_{E_1} - y_{M_2}) \end{vmatrix}}{2((x_{M_2} - x_{S_1})(y_{E_1} - y_{M_2}) - (x_{E_1} - x_{M_2})(y_{M_2} - y_{S_1}))}, \\ y_{O_1} &= \frac{\begin{vmatrix} (x_{M_2} - x_{S_1}) & x_{M_2}^2 + y_{M_2}^2 - x_{S_1}^2 - y_{S_1}^2 \\ (x_{E_1} - x_{M_2}) & x_{E_1}^2 + y_{E_1}^2 - x_{M_2}^2 - y_{M_2}^2 \end{vmatrix}}{2((x_{M_2} - x_{S_1})(y_{E_1} - y_{M_2}) - (x_{E_1} - x_{M_2})(y_{M_2} - y_{S_1}))}. \end{aligned} \tag{7}$$

Note that the denominator of the above expressions is not equal to 0 given that S_1, E_1, and M_2 are the vertices of the inscribed triangle $\triangle S_1M_2E_1$, which is non-collinear. Similarly, let the center derived from $\triangle S_2M_1E_2$ be $O_2(x_{O_2}, y_{O_2})$. Thus, the bias between the two radii and centers are

$$|R_1 - R_2| < T_r, \|O_1 - O_2\|_2 < T_o. \tag{8}$$

All thresholds will be discussed in Section 5.2.2. If Eq. (8) holds, then we further verify inlier ratio regarding the two arcs a_1 and a_2. Here, the average $R = (R_1 + R_2)/2$ and $O = (O_1 + O_2)/2$ are respectively taken as the radius and center of the potential circle C of the two arcs. Let the edge point numbers of a_1 and a_2 be $|a_1|$ and $|a_2|$, and then the inlier ratio IR_{12} is calculated by $IR_{12} = n/(|a_1| + |a_2|)$, where n is the number of edge points that are no more than one pixel from C. If $IR_{12} > 0.5$, then a_1 and a_2 can finally be grouped together. That is, two arcs a_i, $a_j \in \Phi$ that have satisfied the relative position constraint can be combined in the same group G if

$$|R_i - R_j| < T_r \ \wedge \ \|O_i - O_j\|_2 < T_o \ \wedge \ IR_{ij} > 0.5.$$

The grouping process is continued until all circular arcs find their corresponding groups G_1, G_2, ..., G_m. The arc grouping procedure is summarized in Algorithm 1. The grouped result of the extracted arcs in Fig. 2(d) is presented in Fig. 7(a).

4.2.3. Estimation of circular parameters

After arc grouping, we utilize the robust Theil-Sen estimator [51] based on inscribed triangles to effectively estimate circular parameters. To this end, more inscribed triangles are constructed as illustrated in Fig. 8. Specifically, the estimation is divided into four cases. **Case 1:** For the single arc case in Fig. 8(a), we estimate three times for the center and radius by using $\triangle SME$, $\triangle SAE$, and $\triangle SBE$.

Case 2: For two arcs in Fig. 8(b), in addition to the obtained $\triangle S_1M_2E_1$ and $\triangle S_2M_1E_2$, we further add $\triangle S_1E_2E_1$, $\triangle S_1S_2E_1$, $\triangle S_2E_1E_2$, and $\triangle S_2S_1E_2$ to obtain a total of six inscribed triangles.

Case 3: For three or more arcs, we directly utilize the centers and radii that have been calculated for every two arcs in Section 4.2.2. For instance, there are a total of six inscribed triangles for three arcs as shown in Fig. 8(c).

Case 4: For the complete circle, we evenly sample five edge points on this circle and construct five inscribed triangles $\triangle CAD$, $\triangle BEC$, $\triangle ADB$, $\triangle ECA$, and $\triangle DBE$ to estimate the circular parameters as shown in Fig. 8(d).

On the basis of the Theil-Sen estimator, the median of the obtained centers and radii of each case are adopted as the final circular parameters. The estimation results for grouped arcs in Fig. 7(a) are presented in Fig. 7(b).

4.2.4. Refinement of circular parameters

Owing to the discrete property of pixels, the estimated circles may not perfectly match the ideal circle. Therefore, a linear error compensation like that in De Marco et al. [16] is adopted here to further improve the parameter precision. Assume that an ideal circle with center $(\bar{x}_c, \bar{y}_c)$ and radius $\bar{R}_c$ is defined by the following equation:

$$(x - \bar{x}_c)^2 + (y - \bar{y}_c)^2 = \bar{R}_c^2.$$

The detected circle with parameter $(\tilde{x}_c, \tilde{y}_c, \tilde{R}_c)$ affected by errors $(\Delta x_c, \Delta y_c, \Delta R_c)$ will then be formulated as

$$(x - (\tilde{x}_c - \Delta x_c))^2 + (y - (\tilde{y}_c - \Delta y_c))^2 = (\tilde{R}_c - \Delta R_c)^2, \tag{9}$$

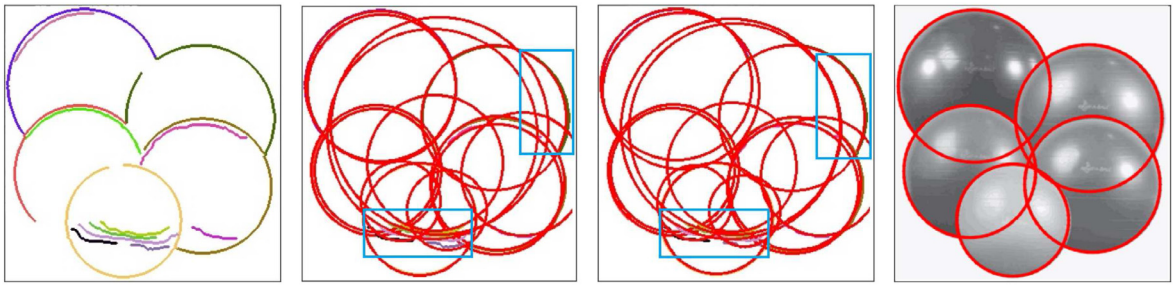


Fig. 7. Circle detection for Fig. 2. From the left to right column: (a) Grouped co-circular arcs. Different groups are indicated by different colors; (b) Estimated circles by inscribed triangles. we add blue boxes to reveal relatively large deviations although the estimated circles match the arcs well overall; (c) Refined circles. Blue boxes show that the circles now match arcs better after the linear error refinement; (d) Finally detected circles after verification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

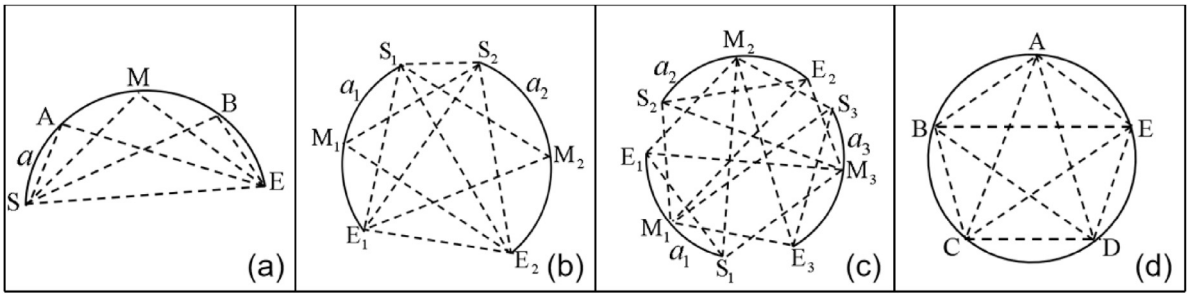


Fig. 8. Estimation of circular centers and radii. (a) For the single arc case, $\triangle SME$, $\triangle SAE$, and $\triangle SBE$ are used. (b) For the two arc case, $\triangle S_1M_2E_1$, $\triangle S_2M_1E_2$, $\triangle S_1E_2E1$, $\triangle S_1S_2E_1$, $\triangle S_2E_1E_2$, and $\triangle S_2S_1E_2$ are used. (c) For the three or more arc case, inscribed triangles of each arc pair that have been obtained in Section 4.2.2 are used. (d) For the complete circle, five inscribed triangles $\triangle CAD$, $\triangle BEC$, $\triangle ADB$, $\triangle ECA$, and $\triangle DBE$ are used.

and the estimation error regarding the detected circle is denoted by

$$E(x_i, y_i) = (x_i - \tilde{x}_c)^2 + (y_i - \tilde{y}_c)^2 - \tilde{R}_c^2, i = 1, 2, \ldots, N,$$

where N is the number of edge points in an arc group.

Reformulating Eq. (9), we obtain

$$((x - \tilde{x}_c) + \Delta x_c)^2 + ((y - \tilde{y}_c) + \Delta y_c)^2 = (\tilde{R}_c - \Delta R_c)^2.$$

Given that ignoring the second-order error terms Δx_c^2 and Δy_c^2 has little influence over the final result [29], the following over-determined linear system is obtained:

$$-2\begin{pmatrix} x_1 - \tilde{x}_c & y_1 - \tilde{y}_c & \tilde{R}_c \\ x_2 - \tilde{x}_c & y_2 - \tilde{y}_c & \tilde{R}_c \\ \vdots & \vdots & \vdots \\ x_N - \tilde{x}_c & y_N - \tilde{y}_c & \tilde{R}_c \end{pmatrix}\begin{pmatrix} \Delta x_c \\ \Delta y_c \\ \Delta R_c \end{pmatrix} = \begin{pmatrix} E(x_1, y_1) \\ E(x_2, y_2) \\ \vdots \\ E(x_N, y_N) \end{pmatrix},$$

which can be simply denoted as

$$-2\mathbf{AX} = \mathbf{E}. \tag{10}$$

Since arc points $\{(x_i, y_i) \in \mathbb{R}^2\}_{i=1}^N$ are different from one another, the coefficient matrix $\mathbf{A}$ is full column rank, which concludes that $\mathbf{A}^T\mathbf{A}$ is invertible, where $\mathbf{A}^T$ is the transpose of $\mathbf{A}$. Then the error vector $\mathbf{X}$ is obtained by

$$\mathbf{X} = -\frac{1}{2}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{E}. \tag{11}$$

The accuracy of estimated parameters can be further improved on the basis of $\mathbf{X}$. The refinement result of the circles in Fig. 7(b) is presented in Fig. 7(c).

4.3. Verification of candidate circles

After refinement, a verification process is applied to depress false circles among candidate circles. Here, the inlier ratio is utilized. Different from traditional methods that only consider distance deviation in general, this study takes both distance and angle deviations into account to obtain more reliable results.

As illustrated in Fig. 9, let the center and radius of a candidate circle C be $O(x_O, y_O)$ and R, respectively. Next, the inlier set I is generated by taking two constraints into account. An edge point $P_j(x_{P_j}, y_{P_j})$ from its arc group G is seen as an inlier if both distance and angle constraints hold as follows:

$$\mathrm{I} = \{P_j \in G \,|\, |\|O - P_j\|_2 - R| \le T_d \ \wedge \ \frac{\mathbf{n}_{P_j}}{\|\mathbf{n}_{P_j}\|_2} \cdot \frac{\nabla_{P_j}C}{\|\nabla_{P_j}C\|_2} \ge \cos T_\theta\}, \tag{12}$$

where

$$\nabla_{P_j}C = (x_{P_j} - x_O, y_{P_j} - y_O)^T$$

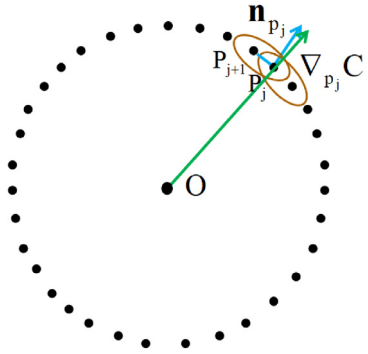


Fig. 9. The normal $\mathbf{n}_{p_j}$ and the gradient $\nabla_{p_j}C_i$ at point P_j, where $\mathbf{n}_{p_j}$ is obtained by two adjacent edge points P_j, P_{j+1}.

and

$$\mathbf{n}_{P_j} = \mathbf{R}(x_{P_j} - x_{P_{j+1}}, y_{P_j} - y_{P_{j+1}})^T = (y_{P_{j+1}} - y_{P_j}, x_{P_j} - x_{P_{j+1}})^T.$$

$\mathbf{R}$ is the rotation matrix with the rotation angle equal to $\pi/2$. T_d is the threshold that evaluates the distance from edge points to the candidate circle, and T_θ is the angle threshold between $\mathbf{n}_{p_j}$ and $\nabla_{p_j}C$. As suggested by Lu et al. [36], we typically set $T_d = 2$ and $T_\theta = 20^\circ$. Then the inlier ratio IR is calculated by $IR = \frac{|\mathrm{I}|}{2\pi R} \in [0, 1]$. A candidate circle C is accepted to be a true circle if its $IR \ge T_{ir}$. Given that the completeness between candidate circles generated by arc grouping and completed circles are different, different thresholds T_{ir} and T_{irc} are also set for them. All thresholds will be analyzed in detail in Section 5.2.2. The final detected circles of Fig. 2(a) after verification are shown in Fig. 7(d).

5. Experimental results

We implement a series of experiments that include synthetic and real images to test the performance of the proposed method. Synthetic images are created to test the arc grouping effect based on the relative position and inscribed triangle constraints, and real images are used to test the general performance, such as accuracy, speed, and robustness. We compare our method with five representative state-of-the-art methods from three categories. The first category is voting-based method of which the most frequently used is HT implemented in OpenCV (HoughCircles) [7], as it effectively narrows down the voting space by means of gradient directions. The improved version of HT named curvature-aided HT (CACD) [58]. The second is sampling-based method such as four-point RCD [11]. The last is geometry-based method which includes edge-drawing with false detection control (ED) [2] and the method with arc-support line segments (AS) [35]. Source codes of these methods are publicly available online. The HT, RCD, ED, AS

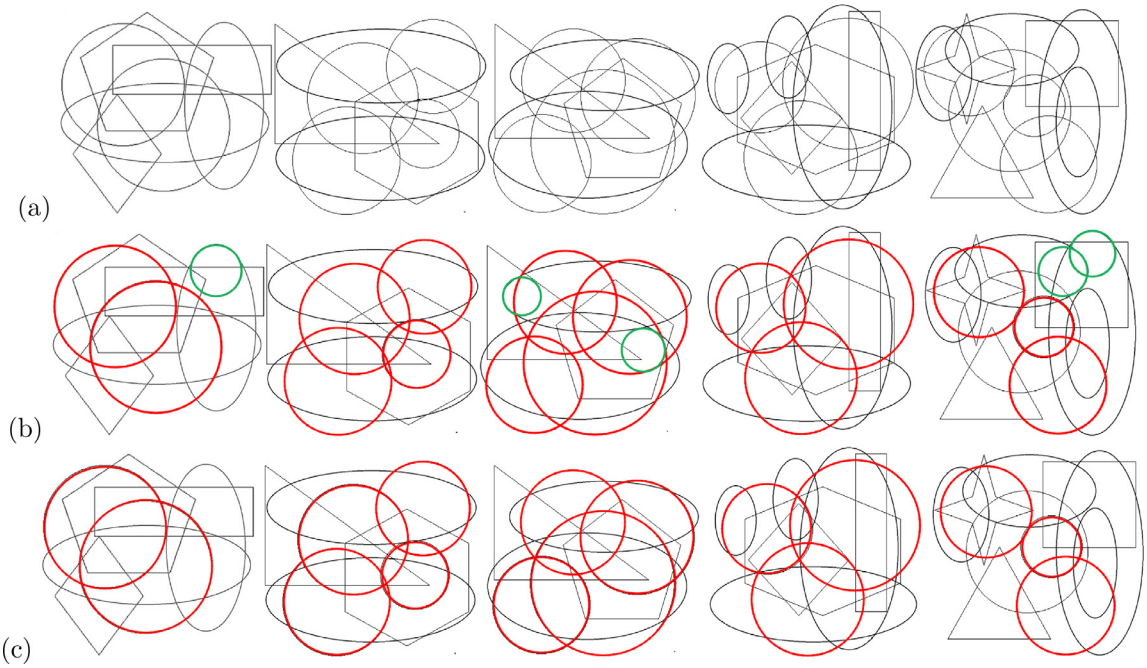


Fig. 10. Test on synthetic images with overlapping. (a) Each image has 2 to 4 circles with severe interferences by other shapes. (b) The detection results of ED, which reports several false circles indicated by green color. (c) The detection results of our method, which almost detects all circles. Rightly detected circles are superimposed by red color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

methods and ours are executed in C++, while CACD is in MATLAB. All experiments are conducted on the same desktop with an Intel Core i7 3.60 GHz CPU and 4 GB RAM without parallel computing.

5.1. Synthetic test of arc grouping and eccentricity study

To assess the effectiveness of the proposed arc grouping strategy and the estimation accuracy by inscribed triangles, we generate a synthetic test set of overlapping circles. As presented in Fig. 10(a), each image contains two to four overlapping circles meanwhile, multiple geometric primitives such as triangles and ellipses are added to aggravate interference. Each circle is broken up as a couple of scattered arc segments, which make the arc grouping sufficiently tough. In addition, the coexistence of elliptic arcs further disturbs the circular arcs. To evaluate quantitatively the accuracy of circular parameters estimated by inscribed triangles, we compare the proposed method with ED that relies on the least-square fitting [8]. All thresholds involved in this test are the same as those in Section 5.2.2.

The detection results of ED and our method are presented in Fig. 10(b) and (c). ED reports as many right circles as possible but also detects false circles for the first, second, and fifth images caused by ellipses, indicating that the verification of ED may not be sufficient. By contrast, the proposed method almost successfully detects all circles and does not have false circles. The average differences of center and radius (*i.e.*, $(\Delta x_o, \Delta y_o)$, Δr) are also calculated between the true circles and the estimated ones for each image. The result is reported in Table 1 where the proposed method is shown to have comparable accuracy with ED, through we directly estimate parameters from inscribed triangles rather than depending on the least-square fitting.

Apart from the accuracy test, we further implement a robustness test for the proposed method against edge point variation. To this end, we manually draw the images in Fig. 10(a) along their edges to obtain the jittering effect, as shown in Fig. 11. The detection results are presented in the second row of Fig. 11, which shows that although edges become rough, the estimated circles by inscribed triangles still match the ground truths well.

Table 1
Test results on overlapping images. $(\Delta x_o, \Delta y_o)$, Δr means the difference between estimated parameters and the ground truths; ΔT means the missing number of true circles, and F means the number of detected false circles. Our method has comparable accuracy with ED yet without false detections.

	ED				Ours			
	$(\Delta x_o, \Delta y_o)$	Δr	ΔT	F	$(\Delta x_o, \Delta y_o)$	Δr	ΔT	F
Syn1	(0.06, 0.41)	0.68	0	1	(0.59, 1.2)	0.52	0	0
Syn2	(0.54, 0.42)	0.39	0	0	(0.42, 0.35)	0.78	0	0
Syn3	(0.41, 0.25)	0.23	0	2	(0.61, 0.28)	0.84	0	0
Syn4	(0.81, 0.34)	0.69	0	0	(0.49, 0.71)	0.84	0	0
Syn5	(0.38, 0.45)	0.58	−1	2	(0.39, 0.54)	0.75	−1	0

It is interesting to see whether our method can detect circles that like ellipses. For this purpose, a sequence of images is generated as illustrated in the first row of Fig. 12, starting with a circle of radius equal to 5 and then slowly changing to an ellipse with growing eccentricity. The circle detection results are shown in the second row of Fig. 12. Moreover, we report the axis length and the inlier ratio for each image in Table 2. Given that the inlier ratio of completed circles is set as $T_{irc} = 0.5$, we can detect the ellipse with eccentricity $e = 0.30$. On the other hand, if we slightly lower the inlier ratio bar for ellipses, then we can accept the ellipse with $e \leq 0.42$ as a true circle. However, as eccentricity increases, the inlier ratio declines rapidly. Hence, for ellipses with high eccentricity, our method can be more suitably applied for ellipse approximation rather than for reliable circle detection.

5.2. Evaluation of the proposed method

Besides synthetic tests where self evaluation is possible, we also assess the quality of real image detection, on four publicly challenging datasets that have different characteristics:

Dataset Mini. A common test set containing 10 images: coins (256 × 256 pixels), cake (256 × 256), insulator (256 × 192),

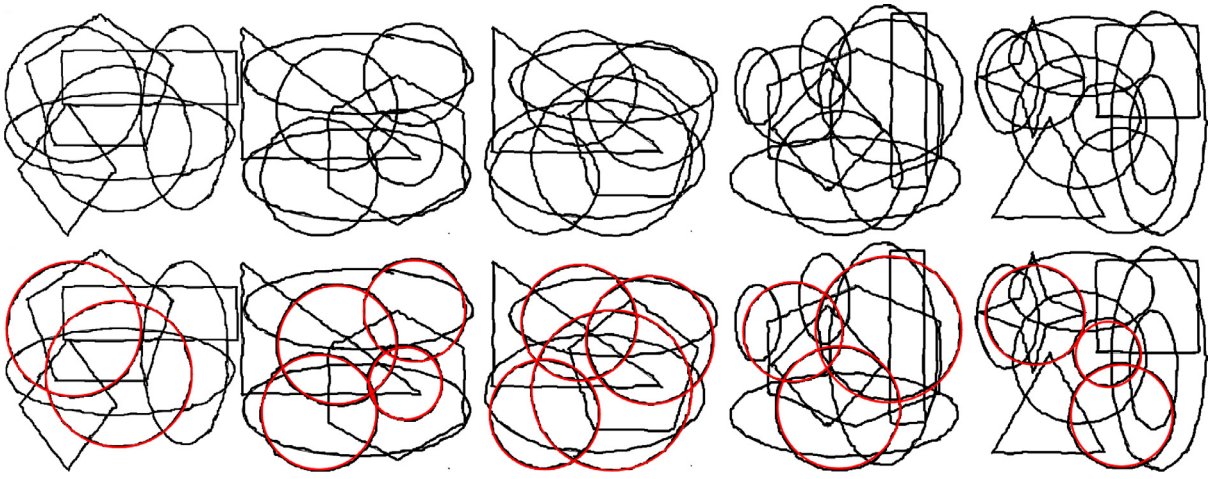


Fig. 11. Test on synthetic images with jittering. The first row is hand-drawn images and the second row is the detection results of our method. Detected circles are superimposed by red color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

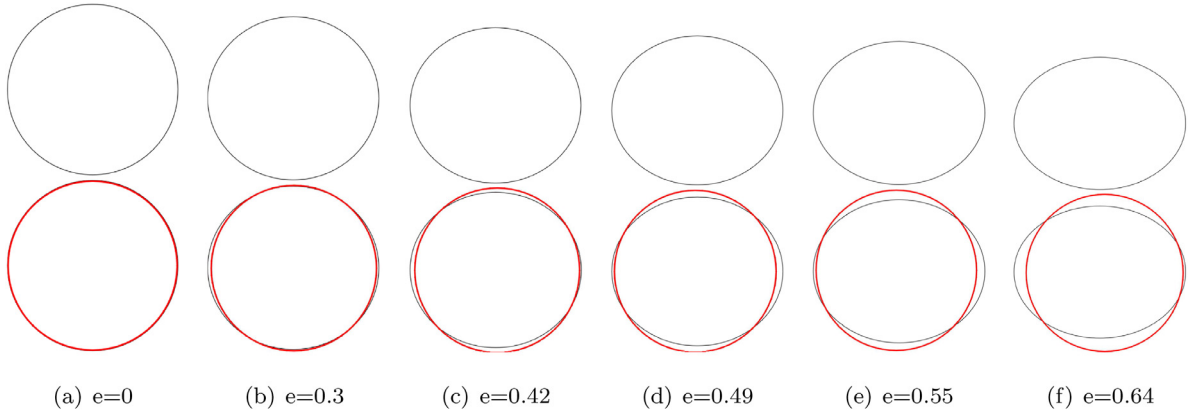


Fig. 12. A sequence of images with growing eccentricity starting from a circle. The first row is the raw images and the second row is the estimated circles. The letter 'e' represents the corresponding eccentricity.

Table 2
Records of a sequence of images with growing eccentricity. When the eccentricity is less than 0.42, we can accept that ellipse as a true circle.

Images	Major axis	Minor axis	Eccentricity	Inlier ratio
E1	10	10	0	0.91
E2	10.50	10	0.30	0.74
E3	11	10	0.42	0.37
E4	11.50	10	0.49	0.26
E5	12	10	0.55	0.18
E6	13	10	0.64	0.09

gobang (256 × 256), plates (400 × 300), logo (283 × 344), speaker (437 × 393), stability-ball (209 × 210), ball (340 × 340), and Swatch (309 × 356). This dataset was first used in Akinlar and Topal [1] and it later became a benchmark used by others [16,62]. For consistency, we also test the proposed method on this dataset and it keeps the pixel size indicated in the brackets the same as in Akinlar and Topal [1] without scaling. Low resolution, spectral reflection, and occlusion are the major challenges in this dataset.

Dataset GH. A complex real-world dataset downloaded from GitHub [43], consisting of 258 gray images from different real-life scenarios. The blurred boundary and occlusion by different objects make the detection task sufficiently tough. The large scope of radius variation also disturbs the detection performance.

Dataset PCB. An industrial dataset for the printed circuit board (PCB) involving 100 images. Each image includes at least one circle. For images with multiple circles, they usually have a concentric structure. In addition, substantial noise and blurring edges further aggravate the detection difficulty.

Dataset MY. A new dataset named MY, which includes 111 real images, is provided. These images are collected by smart phones from different scenarios, such as indoors or outdoors, day or night. Different from the first three datasets, all the images are colorful in MY. Compared with dataset GH, MY contains more circles in each image, and the perspective distance is farther. Hence, more cluttered backgrounds are involved. As the camera perspective may not be perfectly perpendicular to scenes, near-elliptic circles may emerge. We labeled all images manually ourselves.[1]

5.2.1. Comparison metrics

The effectiveness and running time of the proposed method are compared with those of previous ones. For effectiveness, to be objective, we adopt three well-known metrics from information retrieval into our context, namely, *precision, recall*, and *F-measure*. These metrics are defined as

$$\text{Precision} = \frac{|\text{TP}|}{|\text{TP} + \text{FP}|}, \text{Recall} = \frac{|\text{TP}|}{|\text{TP} + \text{FN}|}.$$

[1] The four datasets can be downloaded from https://github.com/zikai1/CircleDetection.

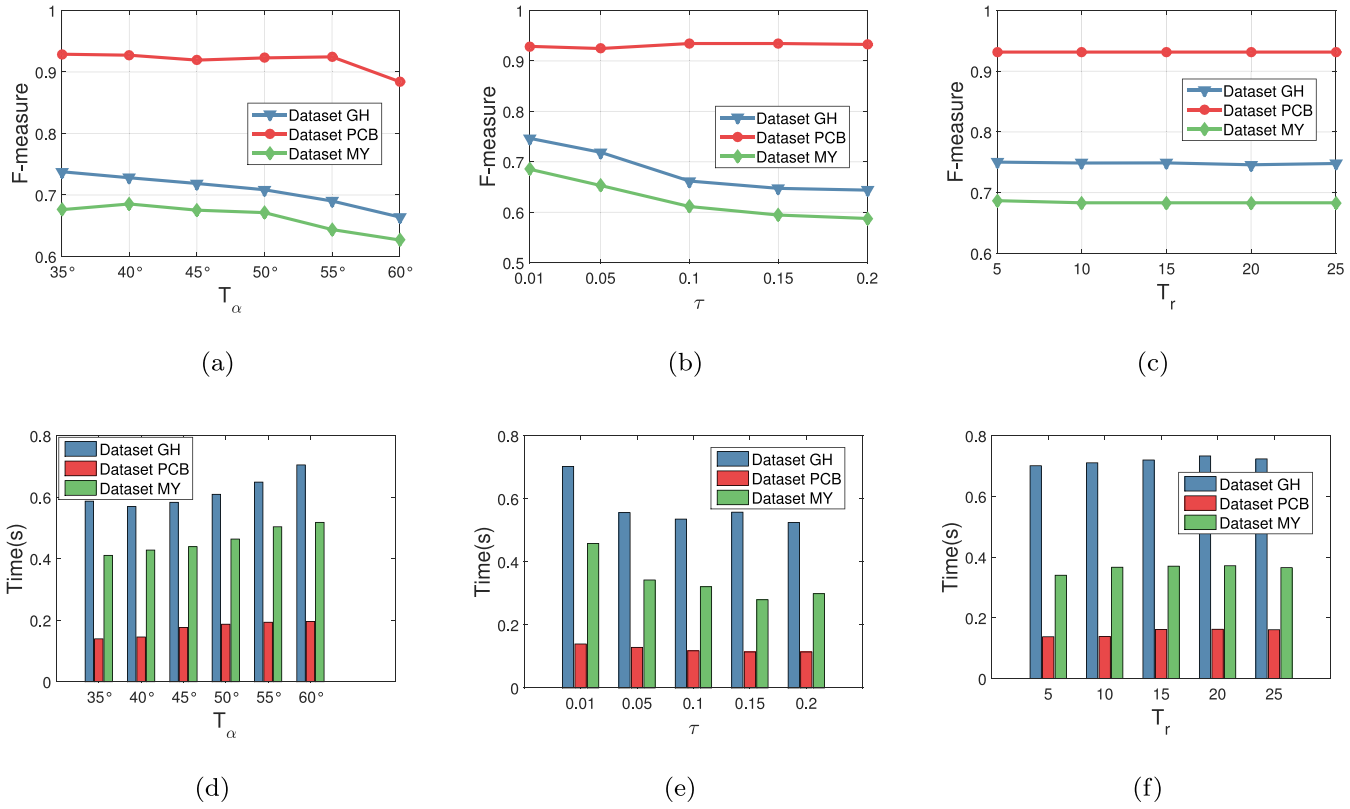


Fig. 13. Statistics of parameter threshold and running time for T_α, τ, and T_r.

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Note that F-measure is a comprehensive performance metric. A detected circle C_d is considered a true positive (TP) if its overlapping ratio regarding the ground truth C_t is not less than 0.8 (suggested by Fornaciari et al. [21], Jia et al. [27], Lu et al. [35], Prasad et al. [42]); otherwise, it is a false positive (FP), and a ground truth not rightly recognized is seen as a false negative (FN). We adopt the manner of [12] to define the overlapping ratio between C_d and C_t:

$$\text{Overlap Ratio}(C_d, C_t) = \frac{area(C_d) \cap area(C_t)}{area(C_d) \cup area(C_t)},$$

where $area(C_*)$ denotes the number of pixels inside the circle C_*. Besides the effectiveness metric, average detection time is also taken to evaluate the algorithm speed.

5.2.2. Threshold analysis

Our method mainly involves six parameters, namely, T_α, T_r, T_o, τ, T_{ir}, and T_{irc}. However, due to the complexity of images, we cannot fix a set of thresholds to obtain the best performance for each image. To reveal the sensitivity of the proposed method regarding parameter thresholds, a series of statistical experiments of F-measure and running time is conducted. We perform each experiment by changing one parameter at a time while fixing the others.

In the pre-processing step, T_α used to recognize sharp corners is crucial for generating circular arcs. Fig. 13(a) indicates that $T_\alpha = 35^\circ$ or 40° can obtain a better F-measure than other angles can, taking the running time in Fig. 13(d) into consideration, we choose $T_\alpha = 40^\circ$ for use. Parameter τ is devised to filter out line segments and then to speed up detection. As shown in Fig. 13(b), with τ increasing, the F-measure monotonically decreases for datasets GH and MY yet keeps stable for dataset PCB. The result indicates that a larger τ will remove more curved edge segments and thus reduce detection effects. Hence, $\tau = 0.01$ is chosen for better effect. T_r in Fig. 13(c) and T_o in Fig. 14(a) are constraint parameters for arc grouping. As can be seen, the F-measure keeps the utmost stability at the range [5, 25] for both T_r and T_o revealing that the estimated centers and radii of arcs from the same circle by inscribed triangles are very close. Their running time in Figs. 13(f) and 14(d) tells us that they also share similar detection speeds among [5, 25]. Therefore, both T_r and T_o are set to be equal to 5. In the verification step, T_{ir} and T_{irc} play the role of recognizing TPs from candidate circles. Fig. 14(b) presents the variation tendency of F-measure when T_{ir} starts from 0.3 to 0.5. Owing to occlusion or noise disturbance, the F-measure gradually declines with the rise in T_{ir}. However, the time consumption for different T_{ir} values is similar, which demonstrates that the verification step spends relatively low time. As for T_{irc}, the F-measure is more stable for datasets GH and MY than for dataset PCB. Therefore, $T_{ir} = 0.3$ and $T_{irc} = 0.5$ are chosen to trade off occlusion and noise interference.

On the other hand, we tune threshold parameters for other compared methods so that their best combinations are used to obtain prime detection results. For instance, the bin size of HT is tuned from 0.5 to 1, and the distance between adjacent centers is tuned from 0 to half-width of the input image.

5.2.3. Performance comparison

We first report the detection results for dataset Mini. As there is no ground truth provided, we manually label the images as shown in Fig. 15. The F-measure of different methods is presented in Fig. 16, while the running time is reported in Table 3. Fig. 16 shows that geometry-based methods have better performance than sampling-based and voting-based methods, and the F-measure of our method for each image is greater than 0.8, which achieves the highest F-measure among all methods. ED and AS

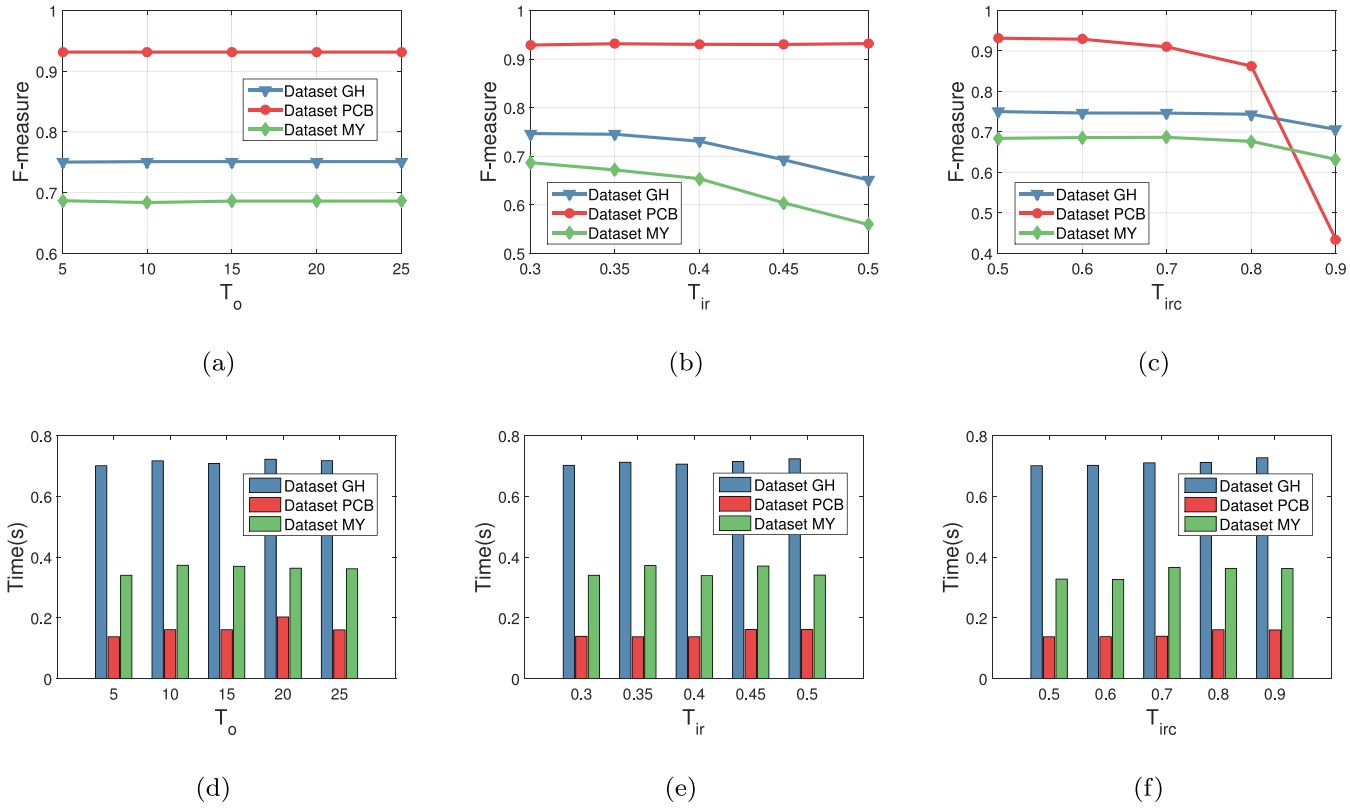


Fig. 14. Statistics of parameter threshold and running time for T_o, T_{ir}, and T_{irc}.

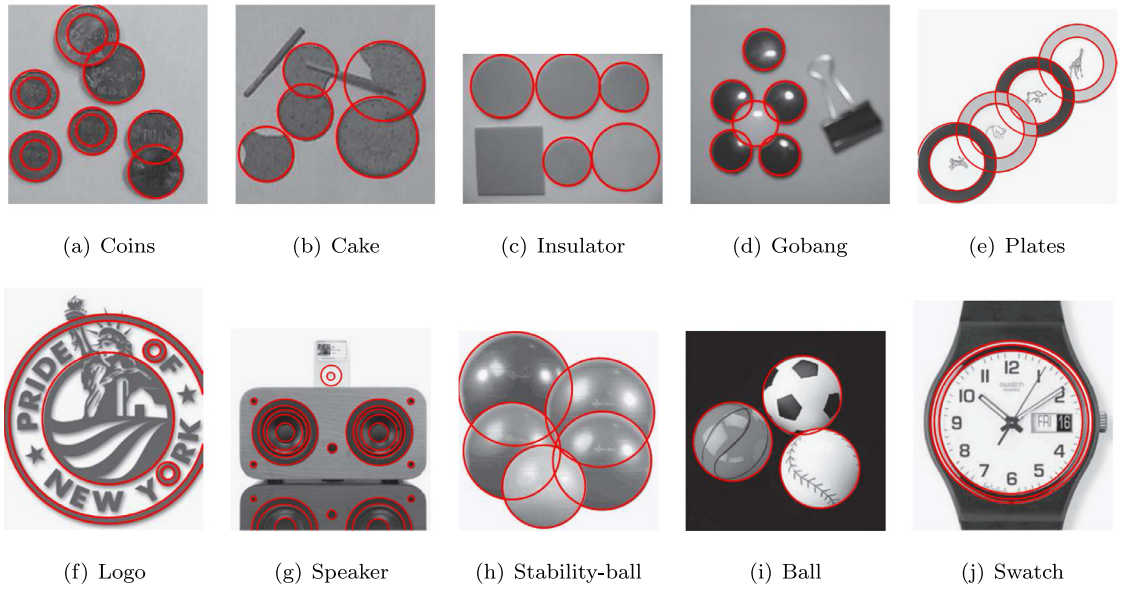


Fig. 15. We manually label the ten images in dataset Mini, and the labeled circles are superimposed by red color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

share similar detection results, but with larger fluctuation than ours. HT has the maximal fluctuation, which indicates its sensitivity to different images.

The detection results are presented in Fig. 17. HT and CACD methods lose many TPs, which demonstrating the weakness of the HT class, that is, the distance between nearly detected centers must be greater than a certain tolerance. Especially, HT is sensitive to concentric circles, which can be concluded from the sixth, seventh and last rows. By contrast, RCD detects many FPs, which are caused by the random sampling from complex edge points and the lack of effective verification. Although ED and AS attain relatively satisfactory performance, they also miss TPs, such as the first two images. Note that both ED and AS take the letters “D”and “O” in the sixth and last images as circles, which indicates that their verification may not be sufficient. We also report the detection time in Table 3, with each time calculated by the average of 100 ex-

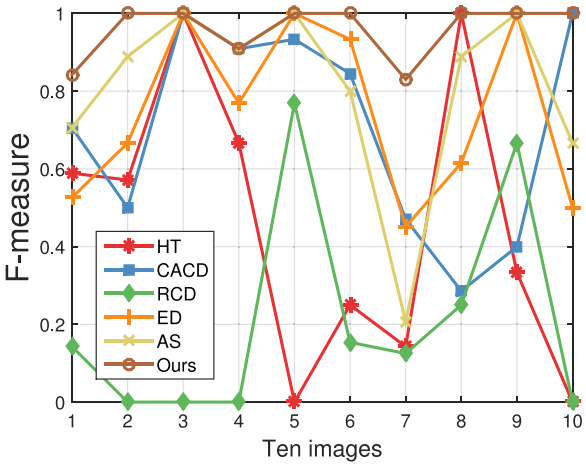


Fig. 16. F-measure of the ten images. Our method achieves the best F-measure along with the minimal fluctuation.

Table 3
Time consumption (s) for the ten test images. Our method has very close time consumption to ED on average. The bold value indicates the best performance.

Images	HT	CACD	RCD	ED	AS	Ours
Coins	0.07	0.52	5.76	0.10	0.38	0.13
Cake	0.05	1.26	6.87	0.09	0.37	0.11
Insulator	0.03	1.32	5.12	0.05	0.32	0.06
Gobang	0.04	0.93	6.03	0.05	0.32	0.06
Plates	0.11	3.30	6.11	0.10	0.48	0.18
Logo	0.87	2.23	4.62	0.12	0.50	0.17
Speaker	0.08	1.25	7.81	0.20	0.47	0.20
Stability-ball	0.03	3.90	4.03	0.14	0.39	0.21
Ball	0.10	0.73	9.47	0.13	0.48	0.14
Swatch	0.73	1.07	5.66	0.17	0.50	0.18
Average	0.21	1.65	6.15	0.11	0.42	**0.14**

periments. The table shows that RCD spends the most detection time, which roots from its exhaustive manner. HT is fast, except in the Logo and the Swatch cases, but its average speed is not the highest, revealing its sensitivity to complex images. In addition, our method offers very close time consumption to that of ED but with better detection performance and is faster than other four methods on average.

Next, a more comprehensive test is conducted on datasets GH, PCB, and MY. They contain a total of 258 + 100 + 111 = 469 real-world images. The detection results are reported in Table 4. From an overall perspective, we discover that geometry methods surpass voting-based and sampling-based methods in both effectiveness and speed. Specifically, the proposed method achieves the best F-measure on datasets GH and PCB, and obtains the second highest F-measure on dataset MY but with the fastest detection speed. ED has higher recall but lower precision than ours, which indicates the lack of strict verification. AS performs well on its own dataset but is inferior on other datasets, especially on dataset GH. Owing to the disturbance of cluttered backgrounds from real world, RCD spends a long time to sample, but the effect is still far from satisfactory. Compared with RCD, HT has better performance and achieves the highest precision on dataset PCB. However, HT is tough in concentric circles, which can be concluded from the low recall on dataset PCB. HT is also sensitive to the complexity of images as indicated from the large time gap between datasets GH and PCB. CACD, an improved version of HT lifts the F-measure overall but also causes precision reduction in the first two datasets. Sample examples are presented in Fig. 18.

Our last experiment is to test the robustness of the proposed method for noisy images. For this purpose, the function *imnoise(img, 'gaussian', mean, variance)* in MATLAB with zero mean and variance ranging from 1% to 35% is used to generate Gaussian white noise, as demonstrated in Fig. 19. The quantitative evaluation is reported in Fig. 20 which shows that when the noise level is zero, ED, AS, and the proposed method can correctly detect all circles. However, when the noise level increases to 8%, AS has a sharp decrease to zero, because noise adversely influences the accurate calculation of gradient. ED also loses half of the TPs. Owing to the use of area computation in our method, we can effectively avoid differential calculation and thus provide more robust detection results. When the noise level is between [10%, 20%], our method gives the right detection for four circles, but ED declines to only one. The reason is that noise has broken the edge into short parts, while ED still needs the length of edge segments to be greater than three consecutive line segments. With noise level exceeding 24%, the extracted edge is heavily affected, resulting in the severe fragmentation of edge segments. Hence it is difficult to generate circles by grouping very small arcs. When the noise level reaches 35% and more, because the edge map has been very fragmented, all methods fail to detect TPs. However, ED, AS, and our method report no FPs as shown in the right side of Fig. 20. By contrast, HT, CACD, and RCD report false detections, which demonstrate that the performance of voting-based or sampling-based methods is inferior when there exists substantial noise. Note that all experiments are conducted without denoising or rescaling the image.

Algorithm 1 Arc grouping process.

input: Arc set $\Phi = \{a_1, a_2, a_3, \cdots, a_n\}$; Radius and center difference thresholds: T_r, T_o.
output: Arc group set Ω

Sort $a_i \in \Phi$ in descending order w.r.t. length by the quick sort method.
Initialize $\Omega \leftarrow \emptyset$;
while $\Phi \neq \emptyset$ **do**
  Select the longest arc $a_i \in \Phi$;
  Initialize the group $G \leftarrow \{a_i\}$;
  for $a_j \in \Phi \backslash G$ **do**
    Determine S_j, M_j and E_j of a_j;
    for $a_k \in G$ **do**
      Determine S_k, M_k and E_k of a_k;
      Judge the relative position of a_j and a_k by Eq. (4);
      if the relative position constraint holds **then**
        Compute the radii R_j, R_k, centers O_j, O_k, and inlier ratio IR_{jk} by Eq. (6) and Eq. (7);
        if $|R_j - R_k| < T_r \wedge \|O_j - O_k\|_2 < T_o \wedge IR_{jk} > 0.5$ **then**
          $G \leftarrow G \cup \{a_j\}$;
        else
          Continue;
        end if
      else
        Continue;
      end if
    end for
  end for
  Push G in Ω;
  $\Phi \leftarrow \Phi \backslash G$;
end while

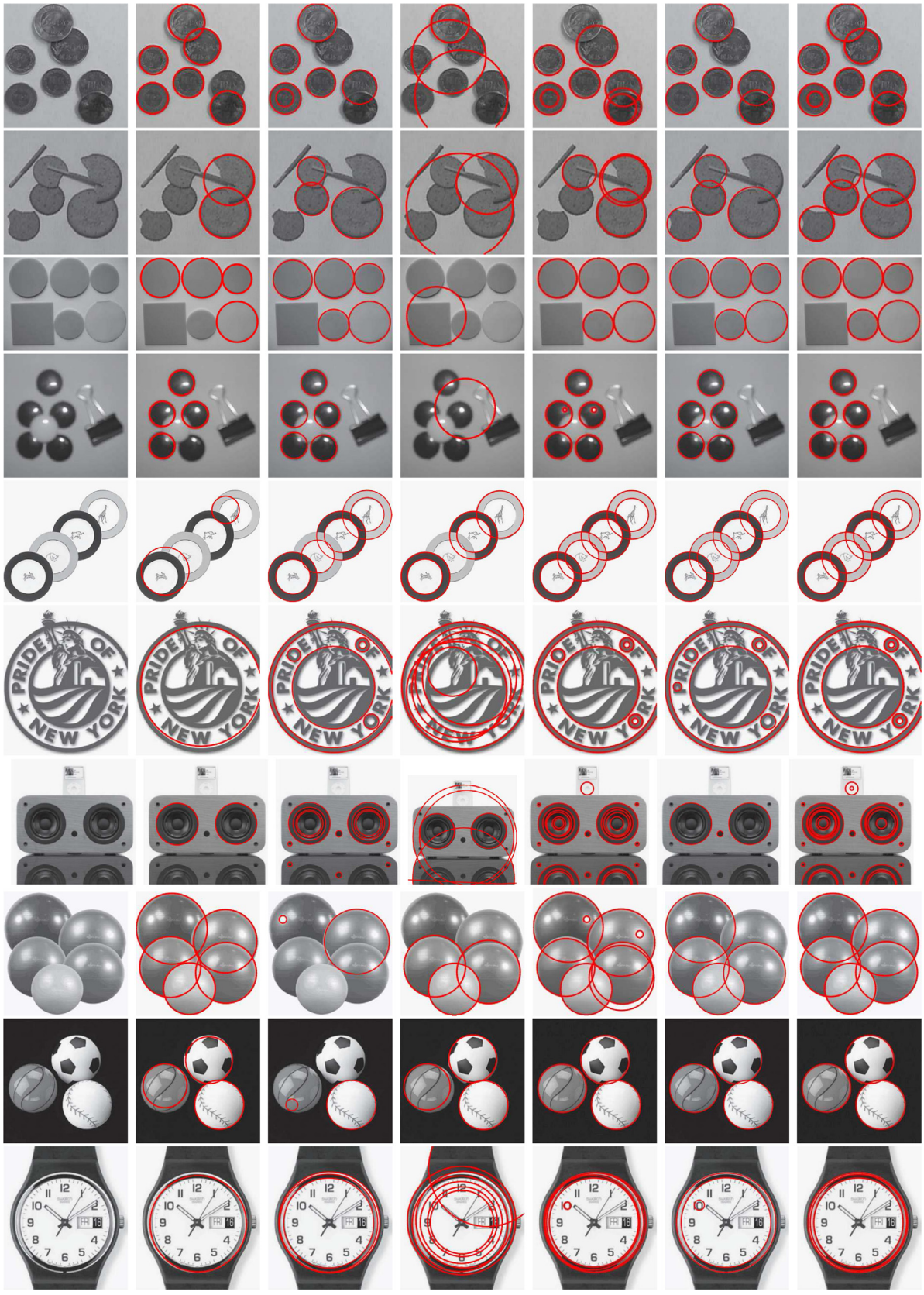


Fig. 17. Circle detection results on dataset Mini, which is widely used by other algorithms. From the left to right column: input image, HT [7], CACD [58], RCD [11], ED [2], AS [35] and ours. As can be seen, the proposed method obtains better performance than others.

Table 4
Circle detection results of all methods for all datasets. Our method obtains the highest F-measure on datasets GH and PCB, and the second F-measure on dataset MY. Bold values indicate the best performance.

	Dataset GH				Dataset PCB				Dataset MY			
	F-measure	Precision	Recall	Time(s)	F-measure	Precision	Recall	Time(s)	F-measure	Precision	Recall	Time(s)
HT	0.36	0.34	0.39	10.53	0.32	0.99	0.19	**0.03**	0.28	0.34	0.24	8.30
CACD	0.38	0.30	0.50	3.62	0.74	0.67	0.84	3.00	0.56	0.54	0.58	3.42
RCD	0.03	0.01	0.43	13.87	0.01	0.01	0.13	12.41	0.03	0.02	0.18	9.32
ED	0.70	0.64	0.78	**0.46**	0.90	0.85	0.95	0.15	**0.70**	0.65	0.75	0.36
AS	0.42	0.39	0.45	1.00	0.92	0.93	0.91	0.40	0.67	0.69	0.66	0.78
Ours	**0.75**	0.76	0.74	0.55	**0.95**	0.96	0.94	0.16	0.69	0.80	0.60	**0.34**

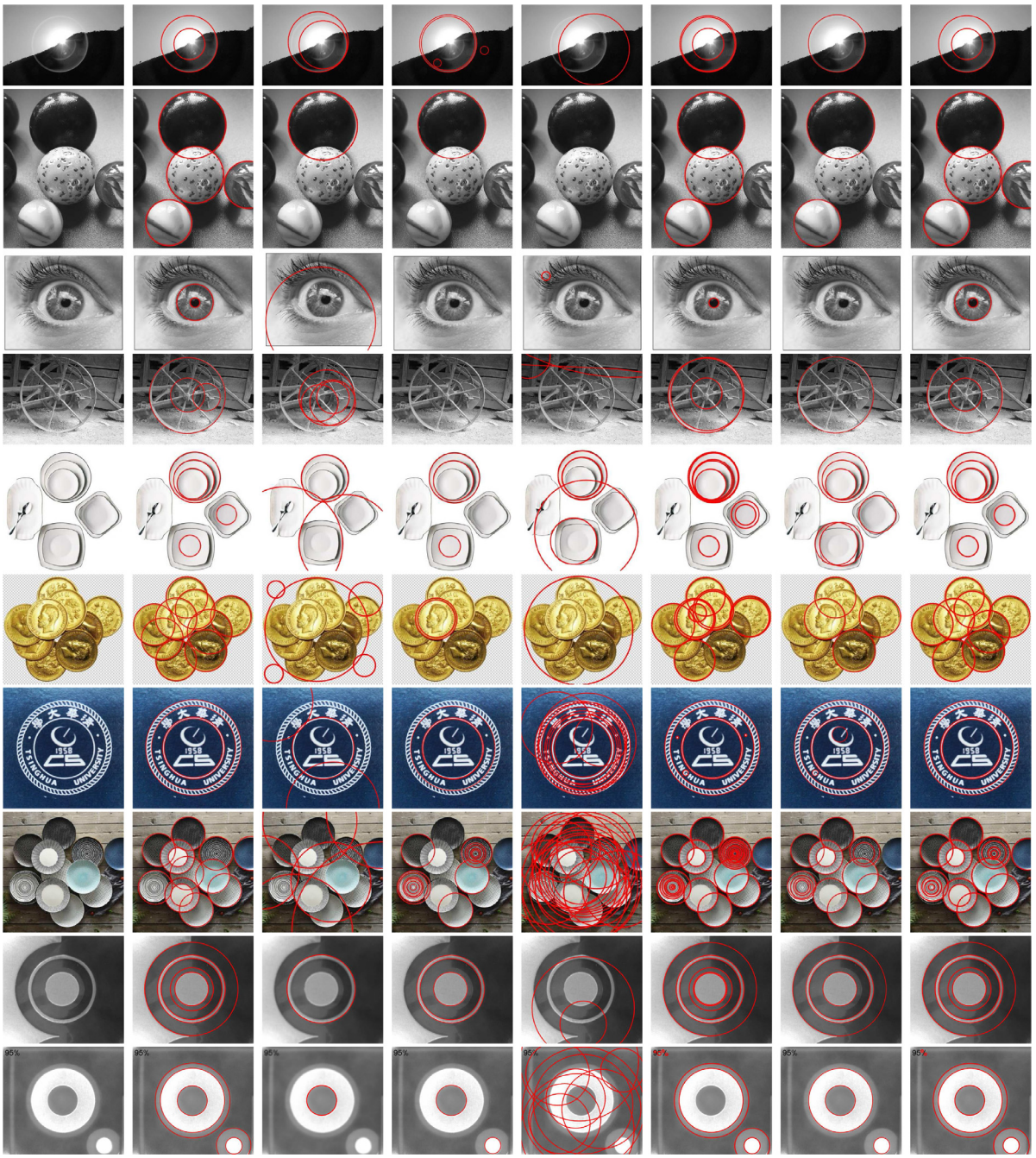
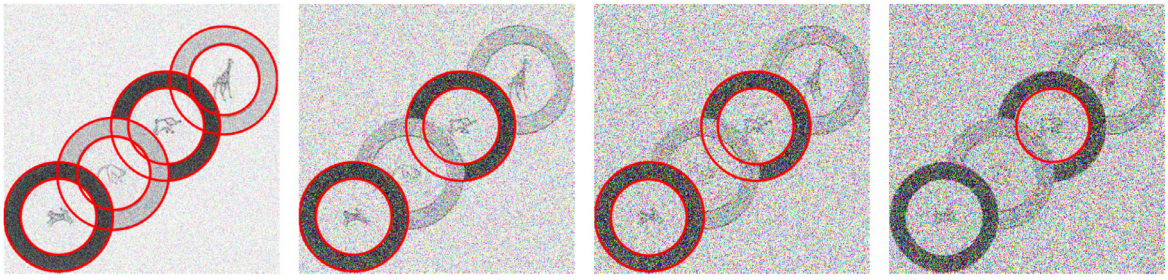


Fig. 18. Circle detection samples from datasets GH (the first four rows), MY (the second four rows), and PCB (the last two rows) for all algorithms. From the left to right column: input image, ground truth, HT [7], CACD [58], RCD [11], ED [2], AS [35] and ours. As can be seen, the proposed method obtains better performance than others.



(a) $\sigma = 2\%$, 8 circles (b) $\sigma = 10\%$, 4 circles (c) $\sigma = 20\%$, 4 circles (d) $\sigma = 30\%$, 1 circle

Fig. 19. Circle detection results of our method at 2%, 10%, 20% and 30% Gaussian noise levels.

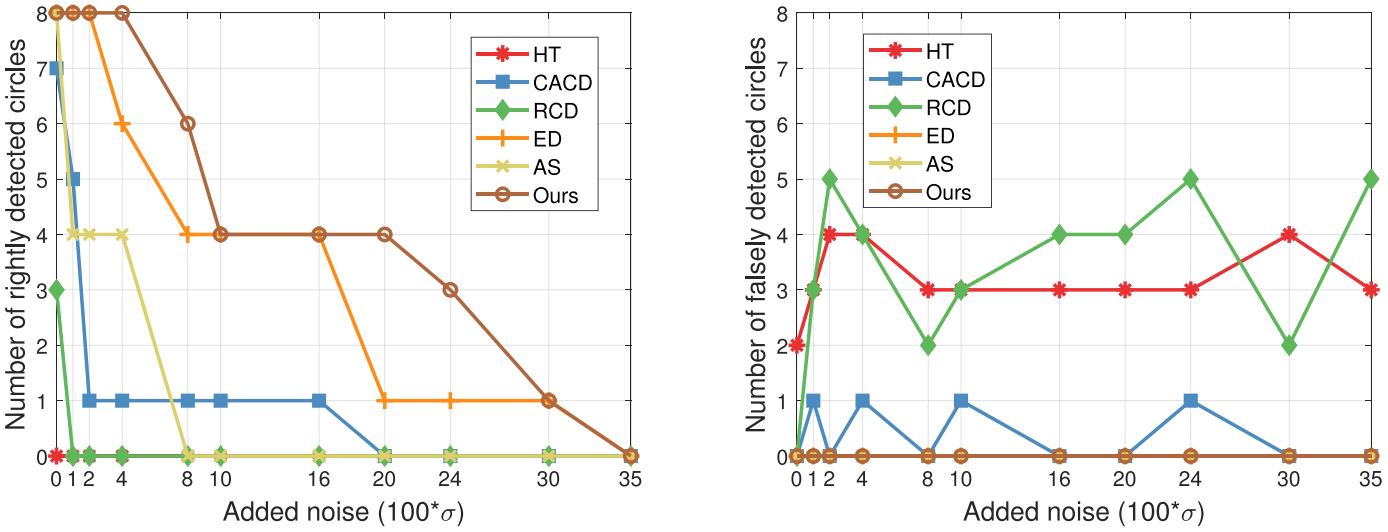


Fig. 20. Circle detection results at different Gaussian noise levels. The number of rightly and falsely detected circles with noise ranging from 1% to 35% are presented in the left and right, respectively.

6. Conclusions

In this paper, we present a novel circle detection algorithm based on inscribed triangles following three main steps. Given an image, our method first extracts circular arcs by removing irrelevant line segments and then devises a comprehensive arc grouping strategy combining the relative position constraint as well as the inscribed triangle constraint. Next, inscribed triangles are utilized further to estimate circular parameters, followed by a refinement step and a verification step. The proposed method can accurately and effectively recognize circles in different scenarios. Inscribed triangles in this work are designed not only to group arcs, and can thus deal with occluded circles, but also to estimate centers and radii of candidate circles. In addition, due to the avoidance of differential computation, inscribed triangles are more robust against noise.

Experiments on synthetic images and four real-world datasets are compared to five representative state-of-the-art methods, and better performance is obtained by our method. Although the proposed method is more robust to noise, its recall will also decline when the noise level is high enough. Thus, a simple denoising step will help. In the future, we plan to apply the proposed method to detect circles in vision systems, such as industrial component measurement and robotic manipulation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61872354 and 61772523), the National Key R&D Program of China (2019YFB2204104), the Beijing Natural Science Foundation (L182059 and Z190004), the Intelligent Science and Technology Advanced subject project of University of Chinese Academy of Sciences (115200S001), the Open Research Fund Program of State key Laboratory of Hydroscience and Engineering, Tsinghua University (sklhse-2020-D-07), and Alibaba Group through Alibaba Innovative Research Program.

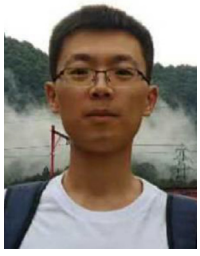
Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patcog.2020.107588.

References

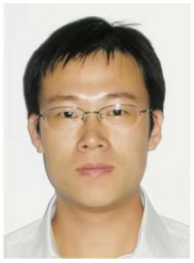
[1] C. Akinlar, C. Topal, EDPF: a real-time parameter-free edge segment detector with a false detection control, Int. J. Pattern Recognit. Artif. Intell. 26 (01) (2012) 1255002.
[2] C. Akinlar, C. Topal, Edcircles: a real-time circle detector with a false detection control, Pattern Recognit. 46 (3) (2013) 725–740.
[3] T.J. Atherton, D.J. Kerbyson, Size invariant circle detection, Image Vis. Comput. 17 (11) (1999) 795–803.
[4] N. Barnes, A. Zelinsky, Real-time radial symmetry for speed sign detection, in: IEEE Intelligent Vehicles Symposium, 2004, 2004, pp. 566–571.
[5] S.K. Berkaya, H. Gunduz, O. Ozsen, C. Akinlar, S. Gunal, On circular traffic sign detection and recognition, Expert Syst. Appl. 48 (2016) 67–75.
[6] J. Bewes, N. Suchowerska, D. McKenzie, Automated cell colony counting and analysis using the circular hough image transform algorithm (CHiTA), Phys. Med. Biol. 53 (21) (2008) 5991.

[7] Gary R. Bradski, A. Kaehler, Learning OpenCV - computer vision with the OpenCV library: software that sees, O'Reilly, 2008.
[8] R. Bullock, Least-square circle fit, 2017. https://www.dtcenter.org/sites/default/files/community-code/met/docs/write-ups/circle_fit.pdf.
[9] J. Cai, P. Huang, L. Chen, B. Zhang, An efficient circle detector not relying on edge detection, Adv. Space Res. 57 (11) (2016) 2359–2375.
[10] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. (6) (1986) 679–698.
[11] T.-C. Chen, K.-L. Chung, An efficient randomized algorithm for detecting circles, Comput. Vis. Image Underst. 83 (2) (2001) 172–191.
[12] A.Y.-S. Chia, S. Rahardja, D. Rajan, M.K. Leung, A split and merge based ellipse detector with self-correcting capability, IEEE Trans. Image Process. 20 (7) (2010) 1991–2006.
[13] K.-L. Chung, Y.-H. Huang, S.-M. Shen, A.S. Krylov, D.V. Yurin, E.V. Semeikina, Efficient sampling strategy and refinement strategy for randomized circle detection, Pattern Recognit. 45 (1) (2012) 252–263.
[14] E. Cuevas, D. Oliva, M. Díaz, D. Zaldivar, M. Pérez-Cisneros, G. Pajares, White blood cell segmentation by circle detection using electromagnetism-like optimization, Comput. Math. Methods Med. 2013 (2013) 395071.
[15] E. Davies, The effect of noise on edge orientation computations, Pattern Recognit. Lett. 6 (5) (1987) 315–322.
[16] T. De Marco, D. Cazzato, M. Leo, C. Distante, Randomized circle detection with isophotes curvature analysis, Pattern Recognit. 48 (2) (2015) 411–421.
[17] A. Desolneux, L. Moisan, J.-M. Morel, From gestalt theory to image analysis: A probabilistic approach, volume 34, 2007.
[18] A.O. Djekoune, K. Messaoudi, K. Amara, Incremental circle hough transform: an improved method for circle detection, Optik 133 (2017) 17–31.
[19] D.H. Douglas, T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica 10 (2) (1973) 112–122.
[20] R.O. Duda, P.E. Hart, Use of the hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.
[21] M. Fornaciari, A. Prati, R. Cucchiara, A fast and effective ellipse detector for embedded vision applications, Pattern Recognit. 47 (11) (2014) 3693–3708.
[22] S.-y. Guo, X.-f. Zhang, F. Zhang, Adaptive randomized hough transform for circle detection using moving window, in: 2006 International Conference on Machine Learning and Cybernetics, 2006, pp. 3880–3885.
[23] J.H. Han, L. Kóczy, T. Poston, Fuzzy hough transform, Pattern Recognit. Lett. 15 (7) (1994) 649–658.
[24] C.-T. Ho, L.-H. Chen, A fast ellipse/circle detector using geometric symmetry, Pattern Recognit. 28 (1) (1995) 117–124.
[25] P.V. Hough, Method and means for recognizing complex patterns, 1962.
[26] L.-q. Jia, C.-z. Peng, H.-m. Liu, Z.-h. Wang, A fast randomized circle detection algorithm, in: 2011 4th International Congress on Image and Signal Processing, 2, 2011, pp. 820–823.
[27] Q. Jia, X. Fan, Z. Luo, L. Song, T. Qiu, A fast ellipse detector using projective invariant pruning, IEEE Trans. Image Process. 26 (8) (2017) 3665–3679.
[28] L. Jiang, Z. Wang, Y. Ye, J. Jiang, Fast circle detection algorithm based on sampling from difference area, Optik 158 (2018) 424–433.
[29] K. Kanatani, Y. Sugaya, Y. Kanazawa, Ellipse fitting for computer vision: implementation and applications, Synth. Lect. Comput. Vis. 6 (1) (2016) 1–141.
[30] D. Kerbyson, T. Atherton, Circle detection using hough transform filters (1995).
[31] H.-S. Kim, J.-H. Kim, A two-step circle detection algorithm from the intersecting chords, Pattern Recognit. Lett. 22 (6–7) (2001) 787–798.
[32] C. Kimme, D. Ballard, J. Sklansky, Finding circles by an array of accumulators, Commun. ACM 18 (2) (1975) 120–122.
[33] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic hough transform, Pattern Recognit. 24 (4) (1991) 303–316.
[34] T. Le, Y. Duan, Circle detection on images by line segment and circle completeness, in: 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3648–3652.
[35] C. Lu, S. Xia, W. Huang, M. Shao, Y. Fu, Circle detection by arc-support line segments, in: 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 76–80.
[36] C. Lu, S. Xia, M. Shao, Y. Fu, Arc-support line segments revisited: an efficient high-quality ellipse detection, IEEE Trans. Image Process. 29 (2020) 768–781.
[37] S. Manay, B.-W. Hong, A.J. Yezzi, S. Soatto, Integral invariant signatures, in: European Conference on Computer Vision, 2004, pp. 87–99.
[38] T.D. Marco, D. Cazzato, M. Leo, C. Distante, Randomized circle detection with isophotes curvature analysis 48 (2) (2015) 411–421.
[39] P. Mukhopadhyay, B.B. Chaudhuri, A survey of hough transform, Pattern Recognit. 48 (3) (2015) 993–1010.
[40] P. Pásztó, P. Hubinskỳ, Mobile robot navigation based on circle recognition, J. Electr. Eng. 64 (2) (2013) 84–91.
[41] H. Pottmann, J. Wallner, Q.-X. Huang, Y.-L. Yang, Integral invariants for robust geometry processing, Comput. Aided Geom. Des. 26 (1) (2009) 37–60.
[42] D.K. Prasad, M.K. Leung, S.-Y. Cho, Edge curvature and convexity based ellipse detection method, Pattern Recognit. 45 (9) (2012) 3204–3221.
[43] L. Qiankun, Dataset gh, 2019. https://github.com/duguqiankun. Accessed January 2020.
[44] A.A. Rad, K. Faez, N. Qaragozlou, Fast circle detection using gradient pair vectors, in: Dicta, 2003, 2003, pp. 879–887.
[45] U. Ramer, An iterative procedure for the polygonal approximation of plane curves, Comput. Graph. Image Process. 1 (3) (1972) 244–256.
[46] D. Shaked, O. Yaron, N. Kiryati, Deriving stopping rules for the probabilistic hough transform by sequential analysis, Comput. Vis. Image Underst. 63 (3) (1996) 512–526.
[47] E.H.B. Smith, B. Lamiroy, Circle detection performance evaluation revisited, in: International Workshop on Graphics Recognition, 2015, pp. 3–18.
[48] M. Soltany, S.T. Zadeh, H.-R. Pourreza, Fast and accurate pupil positioning algorithm using circular hough transform and gray projection, in: International Conference on Computer Communication and Management, 2011.
[49] Y. Su, X. Zhang, B. Cuan, Y. Liu, Z. Wang, A sparse structure for fast circle detection, Pattern Recognit. 97 (2020) 107022.
[50] R. Szeliski, Computer vision: algorithms and applications, 2010.
[51] H. Theil, A rank-invariant method of linear and polynomial regression analysis, in: Henri Theil's Contributions to Economics and Econometrics, 1992, pp. 345–381.
[52] A. Torii, A. Imiya, The randomized-hough-transform-based method for great–circle detection on sphere, Pattern Recognit. Lett. 28 (10) (2007) 1186–1192.
[53] M. Van Ginkel, J. Van de Weijer, L. Van Vliet, P. Verbeek, Curvature estimation from orientation fields, in: 5th Annual Conference of the Advanced School for Computing and Imaging, Heijen, NL, June 15–17, 1999.
[54] J. Xavier, M. Pacheco, D. Castro, A. Ruano, U. Nunes, Fast line, arc/circle and leg detection from laser scan data in a player driver, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 3930–3935.
[55] L. Xu, E. Oja, Randomized hough transform (RHT): basic mechanisms, algorithms, and computational complexities, CVGIP 57 (2) (1993) 131–154.
[56] L. Xu, E. Oja, P. Kultanen, A new curve detection method: randomized hough transform (RHT), Pattern Recognit. Lett. 11 (5) (1990) 331–338.
[57] H. Yang, J. Luo, Z. Shen, W. Wu, A local voting and refinement method for circle detection, Optik 125 (3) (2014) 1234–1239.
[58] Z. Yao, W. Yi, Curvature aided hough transform for circle detection, Expert Syst. Appl. 51 (2016) 26–33.
[59] R.K. Yip, P.K. Tam, D.N. Leung, Modification of hough transform for circles and ellipses detection using a 2-dimensional array, Pattern Recognit. 25 (9) (1992) 1007–1022.
[60] A. Yla-Jaaski, N. Kiryati, Adaptive termination of voting in the probabilistic circular hough transform, IEEE Trans. Pattern Anal. Mach. Intell. 16 (9) (1994) 911–915.
[61] B. Yuan, M. Liu, Power histogram for circle detection on images, Pattern Recognit. 48 (10) (2015) 3268–3280.
[62] H. Zhang, K. Wiklund, M. Andersson, A fast and robust circle detection method using isosceles triangles sampling, Pattern Recognit. 54 (2016) 218–228.



Mingyang Zhao is currently a Ph.D. candidate at Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences (CAS). His research interests include: computer vision, pattern recognition and machine learning.

Xiaohong Jia is an associate professor at Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences (CAS). She received her Ph.D. and Bachelor's degree from the University of Science and Technology of China in 2009 and 2004, respectively. Her research interests include computer graphics, computer aided geometric design and computational algebraic geometry.



Dong-Ming Yan is a professor at the National Laboratory of Pattern Recognition of the Institute of Automation, Chinese Academy of Sciences (CAS). He received his Ph.D. from Hong Kong University in 2010 and his Master's and Bachelor's degrees from Tsinghua University in 2005 and 2002, respectively. His research interests include computer graphics, computer vision, geometric processing and pattern recognition.