International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **4**, April **2013**

ICIC International ©2013 ISSN 1349-4198
pp. **1439–1454**

A MIRROR-CHECKING ALGORITHM FOR EFFICIENT CIRCLE/ARC DETECTION

Shih-Hsuan Chiu[1,*], Ching-Chi Chen[2], Che-Yen Wen[3], Jun-Huei Lee[1]
Kuo-Hung Lin[1], Kuo-Liang Chung[1] and Yong-Huai Huang[4,5]

[1]Department of Materials Science and Engineering
[2]Graduate Institute of Automation and Control
National Taiwan University of Science and Technology
No. 43, Sec. 4, Keelung Rd., Taipei City 106, Taiwan
*Corresponding author: schiu@mail.ntust.edu.tw
{ D9104301; D9304301; k.l.chung; D9412903 }@mail.ntust.edu.tw

[3]Department of Forensic Science
Central Police University
No. 56, Shu-Ren Rd., Kuei-Shan, Taoyuan 33304, Taiwan
cwen@mail.cpu.edu.tw

[4]Institute of Computer and Communication Engineering
[5]Department of Electronic Engineering
Jinwen University of Science and Technology
No. 99, Anzhong Rd., New Taipei City 23154, Taiwan
yonghuai@ms28.hinet.net

Received January 2012; revised May 2012

Abstract. *The iterative multi-step scheme has been applied to circle/arc detection. It usually includes three main steps: picking initial points, finding correspondent searching points with predefined geometric properties, and obtaining candidate circles/arcs. A successful iteration for finding a candidate circle/arc depends on picking valid initial points and finding valid correspondent searching points, and the "valid" means those points must lie on the same target circles/arcs. That is, the iteration is redundant if initial points or searching points are invalid. In this paper, an efficient circle/arc detection method (MCD) based on a mirror-checking algorithm is proposed: we first randomly pick two initial points and construct two corresponding patterns by collecting neighbor points around the initial points. Then, we use the proposed mirror-checking algorithm to check whether the two patterns are mirrored. If they are not mirrored, we will ignore the initial points and find new ones to avoid redundant iteration; otherwise, a candidate circle can be obtained from the mirrored patterns. Instead of finding searching points within a predefined geometric region, the searching point is picked from the mirrored patterns. Based on the initial and searching points, the verification process is utilized to confirm whether the candidate circle is the target circle. From the experimental results, the proposed algorithm can efficiently reduce redundant iterations and executing time. In the meanwhile, we plug the proposed mirror-checking algorithm to the effective voting method (MEVM) and semi-random detection method (MSRD). The experimental results show that it can speed up the two methods and sift redundant iterations efficiently.*
Keywords: Mirror-checking algorithm, Multi-step method, Circle/Arc detection, Geometry property

1. **Introduction.** The Hough transform (HT) is one of the most useful technologies for circle/arc detection in machine vision [1-10]. It transforms every point in the spatial domain into n voting cells of the parameter space. That is, the HT converts the shape detection problem into peak detection in the parameter (or voting) space. However, the

main drawback of the HT is that it requires heavy loads for the 1-to-n mapping processing. That makes itself more difficult be applied to real-time systems.

Instead of using 1-to-n mapping scheme, many n-to-1 mapping methods have been proposed to reduce the loads for efficient circle/arc detection [11-30]. They use an n-to-1 mapping scheme to avoid spurious voting which happened in the 1-to-n mapping processing. The n-to-1 mapping scheme uses n points to decide the increment of one corresponding cell of the voting space. This can reduce the heavy loads of the HT. The randomized Hough transform (RHT) randomly picked three points (3-tuple) to obtain a candidate circle [24,25]. Since the probability of random picking 3-tuple points on the same circle at the same time is low, many RHT-based algorithms have been proposed to increase the sampling probability, such as picking the n-tuple points from the same group [15,22], and applying filtering to de-noising [16,23]. However, it still has the spurious voting problem while n-tuple points are not on the same target circle. A non-HT-based method, the efficient randomized circle detection algorithm (RCD), used the n-to-1 mapping method to detect circles without using the voting space or voting procedure [11]. It used a three step strategy: first, random picking 4-tuple points (seeds); second, determining a candidate circle with a distance criterion; finally, verifying the candidate circle with novel voting strategies [12,13]. However, in a noisy case, we can see that it is critical for random picking 4-tuple points on the same circle simultaneously.

A simple way to increase the probability of random picking seeds is to divide the first step into two sub-steps: picking initial points and finding searching points. That is, the new scheme includes three steps: picking n_i initial points, finding n_s correspondent searching points with predefined geometric properties, and obtaining a candidate circle. Chiu and Liaw (2005) proposed the effective voting method for circle detection (EVM) [14]. They picked two initial points (i.e., $n_i = 2$) to vote for only one candidate circle while the searching point (i.e., $n_s = 1$) is found from a pseudo circle. Zhang and Cao (2008) and Jiang (2009) used only one point as the initial point (i.e., $n_i = 1$) and found the searching points (i.e., $n_s = 2$) with the predefined geometric properties; however, their iteration for finding searching point is explosively increasing in noisy case [18,26]. Shang et al. (2009) proposed a semi-random detection based on right triangles inscribed in a circle (SRD); they used two random picked points as initial points (i.e., $n_i = 2$) and found searching points by following the vertical direction [20,21]. Any three of the points can be treated as a right triangle inscribed in a circle, and the found circle is treated as a candidate one. Chiu et al. (2010) proposed the fast randomized Hough transform (FRHT) to only pick one initial point (i.e., $n_i = 1$) and utilized the symmetric conception for finding the two searching points (i.e., $n_s = 2$) within a window region [27]. If above three points are valid, a reliable target circle can be obtained with the most votes. A successful iteration for finding a candidate circle/arc depends on picking valid initial points and finding valid correspondent searching points, and the "valid" means the initial and searching points are on the same target circle/arc. On the contrary, above methods encounter the same redundant problem that the iteration is redundant while initial points or searching points are not valid. Although Chiu et al. proposed the fast randomized method for efficient circle/arc detection (FRECD) and demonstrated that the initial point cannot be valid [28], it still needs to find valid searching points within a pseudo circle. Unfortunately, the number of redundant iterations is increased in a noisy or complex background case.

In this paper, a mirror-checking algorithm for efficient circle/arc detection method (MCD) is proposed: We first randomly pick two initial points and construct two corresponding patterns by collecting neighbor points around the initial points. We then use the proposed mirror-checking algorithm to check whether the two patterns are mirrored to each other. If they are not mirrored, we will ignore the initial points and find the new

ones to avoid redundant procedure. If they are mirrored, the following searching points are picked from the mirrored patterns, instead of finding within a predefined geometric region. Finally, we use the initial and searching points to obtain a candidate circle, and the verification process is utilized to confirm whether the candidate circle is the target circle. From the experimental results, the proposed MCD based on mirror-checking algorithm can efficiently reduce redundant iterations and executing time. In the meanwhile, we plug the proposed mirror-checking algorithm to the effective voting method (MEVM) and semi-random detection Method (MSRD). The experimental results show that it can speed up the two methods and sift redundant iterations efficiently. We may conclude that the proposed method provides the following truly significant and novel contributions:

1) Instead of finding searching points within a predefined geometric region, the searching point(s) are picked while obtaining the valid initial points.
2) A mirror-checking algorithm is applied to check whether the initial points are valid.

The rest of this paper is organized as follows. Section 2 is the proposed mirror-checking algorithm for efficient circle/arc detection (MCD). Section 3 shows the experimental results and discussions. Finally, some conclusions are addressed in Section 4.

2. **The Mirror-Checking Algorithm for Efficient Circle/Arc Detection (MCD).**

2.1. **Summary of geometric properties.** In Figure 1, there are four points P_a, P_b, P_d, and P_e on a circle, and their coordinates are (x_a, y_a), (x_b, y_b), (x_d, y_d) and (x_e, y_e), respectively. In order to simplify the explanation of the proposed algorithm, we summarize some well-known geometric properties of a circle as follows:
Property I:

We can find symmetric point pairs with respect to a middle-perpendicular line. For example, in Figure 1, we can find the symmetric point $P_e(x_e, y_e)$ from the point $P_d(x_d, y_d)$ with respect to $L_{\perp ab}$. Given P_a, P_b and P_d, the coordinate (x_e, y_e) can be obtained by following steps:

1) In Figure 1, there is a middle-perpendicular line $L_{\perp ab}$ with respect to the line $\overline{P_aP_b}$, and the line equation can be denoted as (general form):

$$Ax + By + C = 0. \tag{1}$$

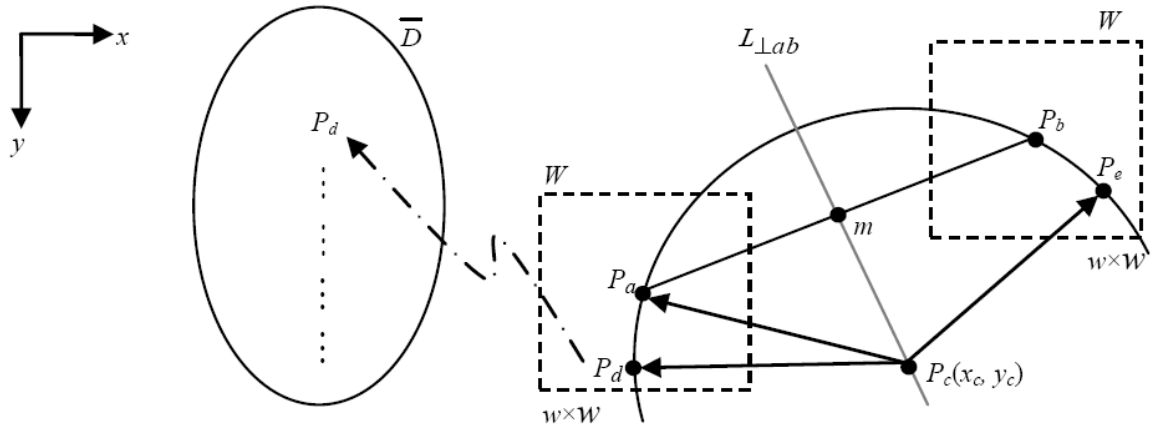


FIGURE 1. The illustration of the proposed method (MCD). In this case, $P_d \in \overline{D}$.

2) Supposing $P_d(x_d, y_d)$ and $P_e(x_e, y_e)$ form a symmetric point pair with respect to $L_{\perp ab}$, we can rewrite Equation (1) as:

$$A\left(\frac{x_d + x_e}{2}\right) + B\left(\frac{y_d + y_e}{2}\right) + C = 0. \tag{2}$$

Since $L_{\perp ab}$ and $\overline{P_dP_e}$ are perpendicular, their slopes are negative reciprocals of each other:

$$\frac{y_d - y_e}{x_d - x_e} \cdot \left(-\frac{A}{B}\right) = -1. \tag{3}$$

3) The coordinate (x_e, y_e) of point P_e can be computed from Equations (2) and (3):

$$\begin{cases} x_e = x_d - \frac{2 \cdot A(A \cdot x_d + B \cdot y_d + C)}{A^2 + B^2} \\ y_e = y_d - \frac{2 \cdot B(A \cdot x_d + B \cdot y_d + C)}{A^2 + B^2} \end{cases} \tag{4}$$

Property II:

Given P_a and P_b, the middle-perpendicular line, $L_{\perp ab}$, also can be obtained from the definition of the parametric form:

$$\begin{cases} x = x_m + v_x \cdot l \\ y = y_m + v_y \cdot l \end{cases}, \tag{5}$$

where $l \in [0, 1]$ denotes the scale factor, and $(x_m, y_m) = \left(\frac{x_a + x_b}{2}, \frac{y_a + y_b}{2}\right)$ denotes the coordinate of point m between P_a and P_b, and $(v_x, v_y) = (x_b - x_a, y_a - y_b)$ denotes the direction vector of $L_{\perp ab}$. We can denote any point on $L_{\perp ab}$ with Equation (5).

2.2. **The mirror-checking algorithm.** Given a $W \times H$ edge image with M edge points, we denote D as a set of all edge points:

$$D = \{P_i = (x_i, y_i) | i = 1, 2, \ldots, M\}, \tag{6}$$

where P_i denotes the edge point with coordinate of (x_i, y_i).

In Figure 1, let P_a and P_b be two initial points, we create two $w \times w$ windows, W_a and W_b, whose centers are P_a and P_b, respectively. According to Property I, all edge points within W_a are scanned one by one (except P_a). If we get a point, P_d, we will use Equation (4) to check if its correspondent symmetric point, $P_e \in D$, exists within W_b. If P_e is found, we store the P_d point to the set $\overline{D}$, where $\overline{D} \subseteq D$. After all points within the W_a region (except P_a) are scanned and checked by using Equation (4), we will treat the two initial points, P_a and P_b, are valid and provide enough symmetric level if the cardinality of the $\overline{D}$, $|\overline{D}|$, is bigger than a predefined threshold T_s (in this paper, we set $T_s = w/2$); otherwise, the two points, P_a and P_b, are invalid while $|\overline{D}|$ is less than a predefined threshold T_s.

2.3. **The proposed method (MCD).** In Figure 1, the proposed efficient method for circle/arc detection (MCD) is illustrated and described as follows:

Step 1: Initialization: Let I_P (I_V) be the iteration times of picking (valid) initial points, and $I_P = I_V = 0$ initially. T_i is the predefined iteration times that we can tolerate. Let N be the current detected circle number, and $N = 0$ initially. T_n is the target circle number.

Step 2: Initial point selection: We randomly pick two edge points from D, and the two points are denoted as P_a and P_b, respectively. In the meanwhile, we perform $I_P = I_P + 1$, where I_P denotes the iteration times of picking initial points from the set D.

Step 3: Mirror mapping for initial point selection (the mirror-checking algorithm): In this step, the picking P_a and P_b are applied to the mirror-checking algorithm to check whether the two random picking initial points are valid (see Section 2.2). If the two points are treated as valid, we perform $I_V = I_V + 1$ and forward to Step 4; Otherwise, we empty the set $\overline{D}$ and go back to Step 2.

Step 4: Obtaining a candidate circle: If $I_V = T_i$, we stop the whole procedure. According to Property II, any point $P_c(x_c, y_c)$ on the middle-perpendicular line, $L_{\perp ab}$, can be represented by using Equation (5):

$$\begin{cases} x_c = x_m + v_x \cdot l \\ y_c = y_m + v_y \cdot l \end{cases}, \tag{7}$$

where $x_m = (x_a + x_b)/2$, $y_m = (y_a + y_b)/2$ and $(v_x, v_y) = (x_b - x_a, y_a - y_b)$ can be obtained from the two initial points (P_a and P_b). We pick a point P_k from the set $\overline{D}$. Assuming that the P_k, P_a and P_b lie on the same circle which is centered at $P_c(x_c, y_c)$, we can obtain an equation $\left|\overline{P_cP_a}\right|^2 = \left|\overline{P_cP_k}\right|^2$ (where $|\bullet|$ denotes the Euclidean distance of two points.):

$$\begin{aligned} &[(x_m + v_x l) - x_a]^2 + [(y_m + v_y l) - y_a]^2 \\ &= [(x_m + v_x l) - x_k]^2 + [(y_m + v_y l) - y_k]^2 , \end{aligned} \tag{8}$$

Therefore, the scale factor l can be obtained by solving Equation (8) as:

$$l = \frac{(x_k^2 + y_k^2) - (x_a^2 + y_a^2) - 2[x_m(x_k - x_a) + y_m(y_k - y_a)]}{2[v_x(x_k - x_a) + v_y(y_k - y_a)]}. \tag{9}$$

For example, in Figure 1, we pick a point P_d from the set $\overline{D}$. The scale factor l can be obtained by using Equation (9):

$$l = \frac{(x_d^2 + y_d^2) - (x_a^2 + y_a^2) - 2[x_m(x_d - x_a) + y_m(y_d - y_a)]}{2[v_x(x_d - x_a) + v_y(y_d - y_a)]}. \tag{10}$$

We also obtain the coordinate of $P_c(x_c, y_c)$ and the correspondent radius r_c as:

$$r_c = \left|\overline{P_cP_a}\right| = \left|\overline{P_cP_d}\right|. \tag{11}$$

We go to Step 5. While all elements of the set $\overline{D}$ are checked and treated as invalid, we empty the set $\overline{D}$ and go back to Step 2.

Step 5: Verification: The circumference of edge points on the candidate circle can be described as:

$$D' = \left\{ P_j \left| \left| \sqrt{(x_j - x_c)^2 + (y_j - y_c)^2} - r_c \right| \le \varepsilon_r \right. \right\}, \tag{12}$$

where $D' \subseteq D$ and ε_r is a tolerant threshold value (we set $\varepsilon_r = 1$ in this paper). We can regard the candidate circle as the target circle while:

$$|D'| \ge 2\pi r_c T_r, \tag{13}$$

where $|D'|$ is the cardinality of D', and $T_r \in [0, 1]$ is the predefined existing rate of the target circle. If the candidate circle is confirmed as the target circle by Equation (13), we remove all elements of D' out of D; in the meanwhile, we let the current detected circle number $N = N + 1$ and empty the set $\overline{D}$; otherwise, we go back to Step 4. If $N = T_n$, we stop the whole procedure; otherwise, we go back to Step 2.

3. Experiments and Discussions.

3.1. Analysis of the computation.

In the iterative procedure, we use I_P to denote the iteration times of picking initial points. As two initial points are picked (e.g., P_a and P_b in Figure 1), the points within the created $w \times w$ window (e.g., W_a) are orderly checked by using Equation (4). Therefore, the iteration for the mirror-checking procedure is:

$$I_P \times (w^2 - 1), \tag{14}$$

where $(w^2 - 1)$ denotes the maximum number of points (except the center point) in the created window region.

While the valid initial points are obtained by using the proposed mirror-checking algorithm, the I_V is used to denote the iteration times. Then, the points in the set $\overline{D}$ are treated as searching points for obtaining candidate circles. Therefore, the iteration for obtaining a candidate circle is:

$$I_V \times (w^2 - 1), \tag{15}$$

where $(w^2 - 1)$ denotes the maximum cardinality of $\overline{D}$. The total iteration times (T) of the MCD required is the sum of Equations (14) and (15):

$$\begin{aligned} T(\text{MCD}) &\approx I_P \times (w^2 - 1) + I_V \times (w^2 - 1) \\ &\approx (I_P + I_V) \times w^2. \end{aligned} \tag{16}$$

Given a $W \times H$ image with total M edge points, we suppose there are N points on a circle; i.e., the point on the circumference is:

$$N = 2\pi r. \tag{17}$$

In this paper, the value of w is set as 20, which is much smaller than N. We have the relationship:

$$W \cdot H > M > N > w. \tag{18}$$

We use p to denote the probability of picking a valid point on a target circle, where p is approximated as:

$$p = \frac{N}{M}. \tag{19}$$

For the MCD, the probability of random picking initial points of P_a and P_b on the same target circle is:

$$P_{\text{MCD}} = \frac{N}{M} \cdot \frac{N-1}{M-1} \approx \left(\frac{N}{M}\right)^2 \approx p^2. \tag{20}$$

In general, the minimum iteration for picking valid initial points at least once (i.e., $I_V = 1$) is:

$$I_P = \frac{1}{p^2}. \tag{21}$$

Therefore, the total iteration times of the MCD required in Equation (16) is rewritten as:

$$T(\text{MCD}) \approx (I_P + 1) \times \left(w^2 - 1\right) \approx \left(\frac{w}{p}\right)^2, \tag{22}$$

where w is set as 20 in this paper, and $(w^2 - 1)$ denotes the total pixel number in the window region (except the center point).

3.1.1. *Computation comparison with the RCD.* For the RCD, the probability of randomly picking four initial points on the same target circle is:

$$P_{\mathrm{RCD}} = \frac{N}{M} \cdot \frac{N-1}{M-1} \cdot \frac{N-2}{M-2} \cdot \frac{N-3}{M-3} \approx p^4. \tag{23}$$

After an initial picking, a distance criterion is then utilized to check whether a candidate circle; therefore, the minimum iteration times (I_P) for RCD to pick four valid initial points at least once is:

$$T(\mathrm{RCD}) = \frac{1}{P_{\mathrm{RCD}}} = \frac{1}{p^4}. \tag{24}$$

We compare the iteration times of the MCD and RCD by using Equations (22) and (24), and the ratio of $T(\mathrm{MCD})$ and $T(\mathrm{RCD})$ is obtained by:

$$\frac{T(\mathrm{MCD})}{T(\mathrm{RCD})} \approx (p \cdot w)^2. \tag{25}$$

In this paper, w is set as 20. In other words, the proposed MCD will consume less computation iterations while p is decreasing. In general, while p is less than $\frac{1}{20}$, the MCD provides better performance than the RCD in computation iteration requirement.

3.1.2. *Computation comparison with the EVM.* When using the EVM to detect circles, we need to re-sample the M edge points with an interval d along the y-direction. When we set d as one (all edge points should be considered), the needed iteration is maximum and approximated as:

$$T(\mathrm{EVM}) \approx \frac{\left(\frac{M}{d}\right)\left(\frac{M}{d}-1\right)}{2} \approx \frac{M^2}{2}. \tag{26}$$

We compare the iteration times of the MCD and EVM by using Equations (22) and (26). The ratio of $T(\mathrm{MCD})$ and $T(\mathrm{EVM})$ is obtained by:

$$\frac{T(\mathrm{MCD})}{T(\mathrm{EVM})} \approx \left(\frac{w}{p}\right)^2 \cdot \left(\frac{2}{M^2}\right) = 2\left(\frac{w}{N}\right)^2. \tag{27}$$

According to Equation (18), we can see that the proposed MCD will consume less iteration times than that of the EVM.

3.1.3. *Comparison with the SRD.* To detect circle by using the SRD, the probability of picking valid initial points is:

$$P_{\mathrm{SRD}} = \left(\frac{2r}{H}\right) \cdot \left(\frac{2}{W} \cdot \frac{2-1}{W-1}\right) \approx \frac{4 \cdot r}{H \cdot W^2}, \tag{28}$$

where $\left(\frac{2r}{H}\right)$ denotes the probability of picking a valid line which is crossing to the target circle, and $\left(\frac{2}{W} \cdot \frac{2-1}{W-1}\right)$ denotes the probability that initial points are picked from the same target circle. Therefore, the minimum iteration times (I_P) for the SRD to pick valid initial points at least once is:

$$I_P(\mathrm{SRD}) = \frac{1}{P_{\mathrm{SRD}}} = \frac{H \cdot W^2}{4 \cdot r}. \tag{29}$$

Following the vertical direction of initial points, the iteration times of searching points is the height of image size; i.e., H. Therefore, the total iteration time of the SRD is:

$$T(\mathrm{SRD}) \approx I_P(\mathrm{SRD}) \cdot H \approx \frac{(W \cdot H)^2}{4 \cdot r}. \tag{30}$$

We compare the iteration times of the MCD and SRD by using Equations (22) and (30). The ratio of $T(\text{MCD})$ and $T(\text{SRD})$ is:

$$\frac{T(\text{MCD})}{T(\text{SRD})} \approx 4r \cdot \left(\frac{w}{N} \frac{M}{W \cdot H} \right)^2. \tag{31}$$

According to Equation (18), the proposed MCD will consume much less computation iteration than that of the SRD.

In what follows, several synthetic and realistic images are used to test and compare the performance of the methods (RCD, EVM, SRD and the proposed MCD). Table 1 shows the parameter settings of the four methods in our experiments. All of experiments are performed on Vista Operation System with Intel Core 2 Duo CPU (2.20 GHz) and executed (C++ programming) by using Borland C++ Builder 6.0.

TABLE 1. The parameter setting of the three methods. The "×" denotes that the parameter is not needed. The w, T_s, T_r and ε_r denote window size, scores threshold, existing rate and detection error, respectively.

Methods	w	$T_s(= w/2)$	T_r	ε_r
RCD	×	×	0.7	1
EVM	×	×	0.7	1
SRD	×	×	0.7	1
MCD (proposed)	20	10	0.7	1

3.2. **Synthetic image experiments.** Figure 2(a) shows a 256×256 circle image. The radius and circumference of the circle are 37 and 232 pixels, respectively. That is, there are 232 edge points on the circle. We impose different levels of salt and pepper noise on the image. The levels ($n\%$) range from 100% (i.e., adding 232 noise points) to 2000% (i.e., adding 4640 noise points). For example, Figures 2(b) and 2(c) are 1000% and 2000% noisy images of Figure 2(a), respectively. The thresholds T_i (the maximum iteration times that we can tolerate) and T_n (the excepted number of target circles) are set to be ∞ and 1, respectively. Figure 3 illustrates the average execution time (obtained from 100 simulations, t_{ave}) of the four methods for detecting the target circle in Figure 2(a) and its noisy images (please be noted that the execution time of mirror-checking algorithm of the MCD is already considered). It is clear to see that the execution time of the RCD and EVM are much longer than those of the SRD and MCD. After 200% noise imposed, the proposed MCD provides significant better performance than the SRD in t_{ave} required. As 2000% noises are imposed, the proposed MCD ($t_{ave} \approx 0.007$ seconds) is much faster than SRD ($t_{ave} \approx 0.5$ seconds), EVM ($t_{ave} \approx 10$ seconds), and RCD ($t_{ave} \approx 12$ seconds).

Figure 4(a) shows a 256×256 image with six target circles. There are totally 2154 edge points on circumferences of six circles. Again, we impose different levels of salt and pepper noise on the image. The levels ($n\%$) range from 20% (i.e., adding 430 noise points) to 200% (i.e., adding 4308 noise points). Figures 4(b) and 4(c) show the noise imposed images of Figure 4(a) by 100% and 200% (i.e., adding 2154 and 4308 noise points), respectively. The thresholds T_i (the maximum iteration times that we can tolerate) and T_n (the excepted number of target circles) are set to be ∞ and 6, respectively. Figure 5 illustrates the average execution time (t_{ave}) of four methods for detecting the circles in Figure 4(a) and its noisy images (please be noted that the time of mirror mapping of the MCD is already considered). It can be seen that the execution time of the MCD is much faster than the other ones. As 200% noises are imposed, the execution time (t_{ave}) of the proposed MCD is 0.049 seconds. As shown in Figure 6, $n\% = 200\%$, the MCD performs

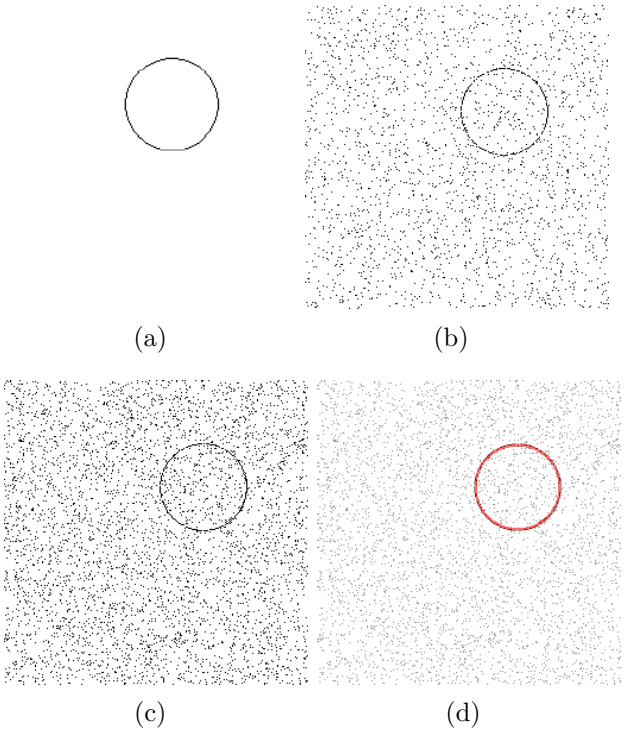


FIGURE 2. A synthetic image (single target circle): (a) an origin image with a circle; (b) imposing 1000% noise on the original image; (c) imposing 2000% noises on the original image; (d) the detection result of (c) by using the proposed MCD, and the average execution time (t_{ave}) obtained from 100 simulations is about 0.007 seconds.

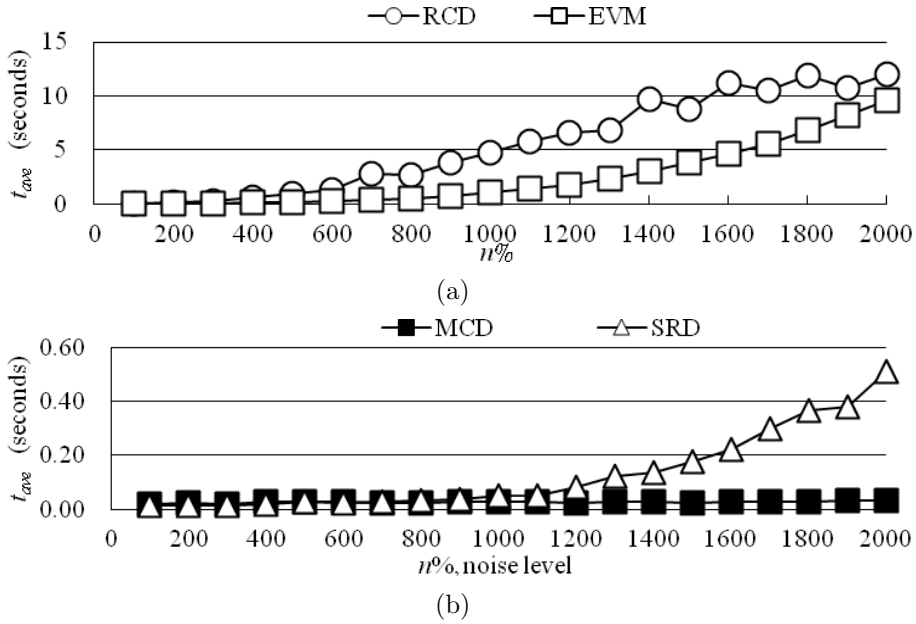


FIGURE 3. The experimental result of average execution time (seconds) of Figure 2: (a) RCD and EVM; (b) the SRD and the proposed MCD. The noise levels ($n\%$) range from 100% to 2000%, and the average execution time (t_{ave}) is obtained from 100 simulations.

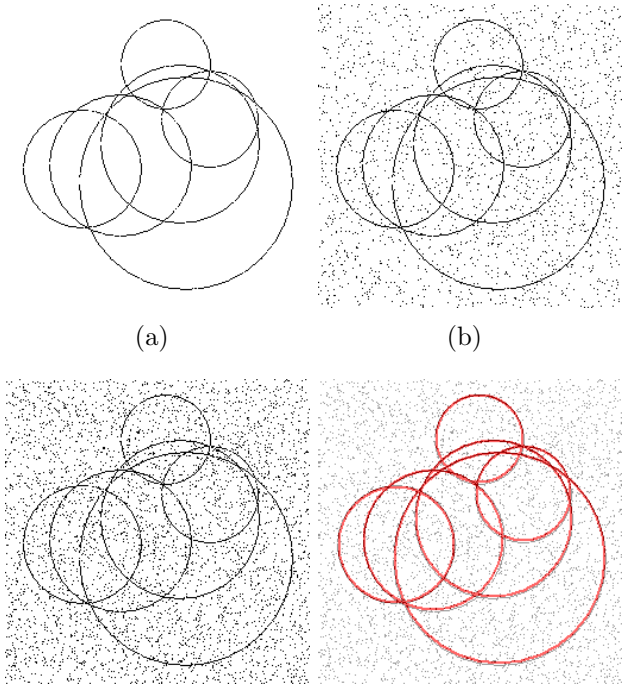


FIGURE 4. A synthetic image (six target circles): (a) an image with six circles; (b) imposing 100% salt and pepper noise on (a); (c) imposing 200% salt and pepper noise on (a); (d) the detection result of (c) by using the MCD, and the execution time obtained from 100 simulations is about 0.049 seconds.

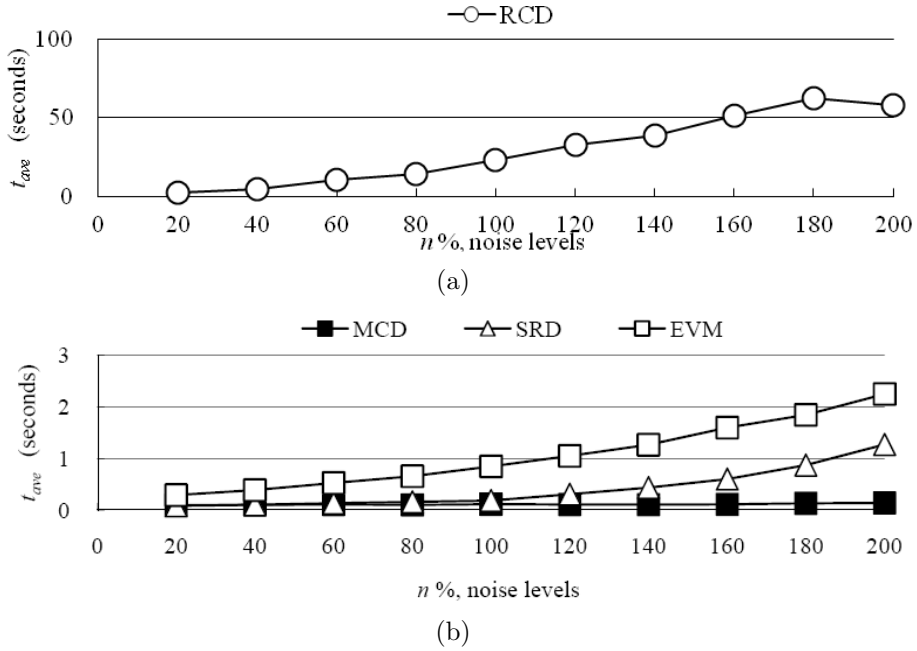


FIGURE 5. The experimental result of average execution time of Figure 4: (a) RCD; (b) the SRD, EVM and the proposed MCD. The noise levels ($n\%$) range from 20% to 200%, and the average execution time (t_{ave}) is obtained from 100 simulations.

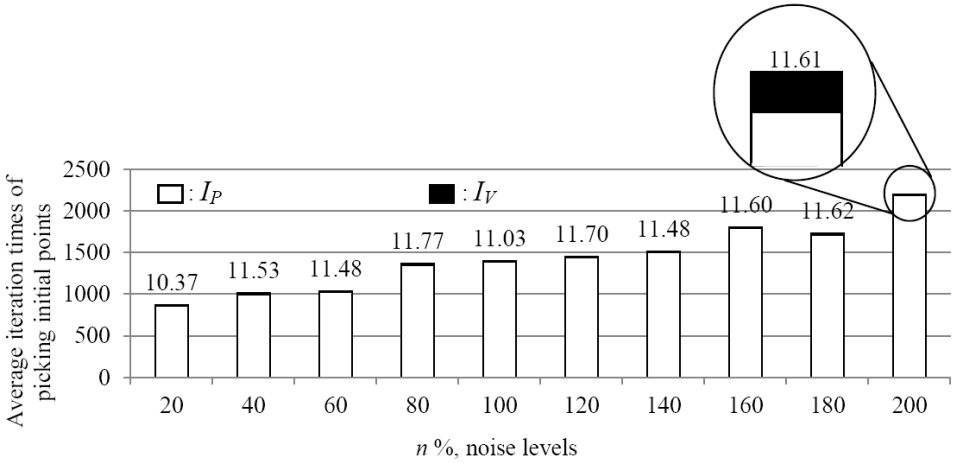


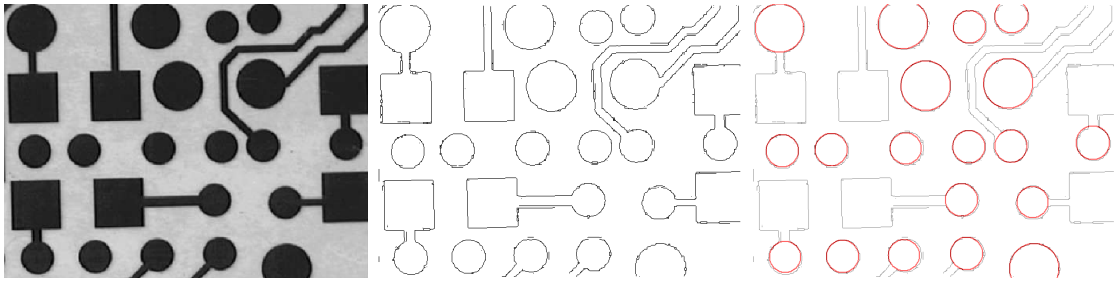
FIGURE 6. The average iteration times of picking initial points: The noise levels ($n\%$) range from 20% to 200%, and the average iteration times of I_V and I_P are obtained from 100 simulations. For example, in 200% noise level, the average I_V is 11.61 iterations while the average I_P is 2186.72 iterations. It denotes the mirror-checking algorithm can efficiently sift most redundant iteration while picking invalid initial points.

(a) (b) (c)

FIGURE 7. A realistic image (480 × 320) testing: (a) a rider wearing a helmet; (b) the edge image of (a); (c) the detection result of the proposed MCD. The T_r, T_n and T_i are set as 0.4, 1 and ∞, respectively.

about 2186.72 iteration times ($I_P = 2186.72$) for picking initial points (i.e., Step 2 in section 2.3); however, only about 11.61 iteration times (i.e., Step 3 in section 2.3) with valid initial points forward to the following steps (finding the correspondent searching points and verifying), and that is why the execution time required is much less than the other ones while noise increasing.

3.3. **Realistic image experiments.** Figure 7(a) is a realistic image (480 × 320) with a rider wearing a helmet. Our target is the helmet. In this case, T_r is set as 0.4, and the average execution time (t_{ave}) for the proposed MCD is about 1.22 (±0.85) seconds while $T_i = \infty$ (the maximum iteration times that we can tolerate) and $T_n = 1$. While the MCD can detect the helmet successfully, EVM and SRD cannot work well. For the RCD, the probability of picking four initial points on the target helmet is quiet low; the experimental result of average execution time is much more than that of MCD required. For the EVM, the initial points are picked from an interval d, and we will find a candidate circle within a pseudo circle range. That is, while the interval d is improper set, it is difficult to find a candidate circle. In this case, although it can detect the target arc while $d \leq 5$ (while $d > 5$, the EVM fails to detect the helmet), the computation time is increasing as d



(a) (b) (c)

FIGURE 8. A realistic image (640×480) testing: (a) a printed circuit board image; (b) the edge image of (a); (c) the detection result of the proposed MCD. The T_r, T_n and T_i are set as 0.7, 19 and ∞, respectively.

TABLE 2. The average numbers of detected target circles, N_{ave} (±S.D.), for the experiment of Figure 8. The total circle number in the printed circuit board image is 19, and T_i denotes the maximum iteration times that we can tolerate.

Methods	$T_i = 500$	$T_i = 1,000$	$T_i = 2,000$
RCD	0 (±0)	0.05 (±0)	0.12 (±0.03)
EVM	0 (±0)	0.15 (±0.11)	0.17 (±0.23)
SRD	11.03 (±2.27)	15.82 (±1.39)	17.97 (±0.79)
MCD (proposed)	18.16 (±0.67)	18.77 (±0.35)	19 (±0.0)

decreases. The average execution time (t_{ave}) for the EVM is about 13.21 (±0.02) seconds as $d = 5$. For the SRD, since it applies the geometry of a right triangle inscribed in a circle to circle detection, it will fail if the length of the target arc is less than a semicircle. In this case, the arc of the helmet is less than a semicircle, and the SRD cannot detect it.

In many practical applications, we may have no idea how many circles/arcs in an examined image. It is not easy to set proper T_i and T_n. A good circle/arc detection method should detect circles/arcs efficiently. That is, it can detect all target circles/arcs in a limited iterative procedure. Figure 8 shows a 640 × 480 printed circuit board image with 19 target circles (i.e., the accurate $T_n = 19$). If we let $T_i = \infty$ (the maximum iteration times that we can tolerate), all three methods can detect the target 19 circles successfully (see Figure 8(c)). If we change T_i as a constant value (as in Table 2, we let T_i = 500, 1,000, and 2,000), we may miss some target circles; i.e., the detected circle number N is less than T_n. We use N_{ave} to denote the average number of detected target circles from 100 simulations. We can see the proposed MCD almost finds all circles; in the meanwhile, the other methods miss some circles. For example, when $T_i = 500$, the MCD can find all circles most of time; on the other hand, the SRD misses about 8 circles and the EVM cannot find any circle.

3.4. **Plugging the mirror-checking algorithm with the effective voting method (MEVM) and semi-random detection method (MSRD).** We redo the experiment testing in Figure 2(a). The noise levels (n%) range from 100% to 2000%. Figure 9 illustrates the average execution time (obtained from 100 simulations, t_{ave}). While the EVM regards all picked initial points as valid and finishes all the following steps, the MEVM uses the mirror-checking algorithm to avoid redundant computations for finding correspondent searching points or circle detection while the initial points are invalid. As shown in Figure 10, as n% = 2000%, the MEVM performs 42,360 iteration times

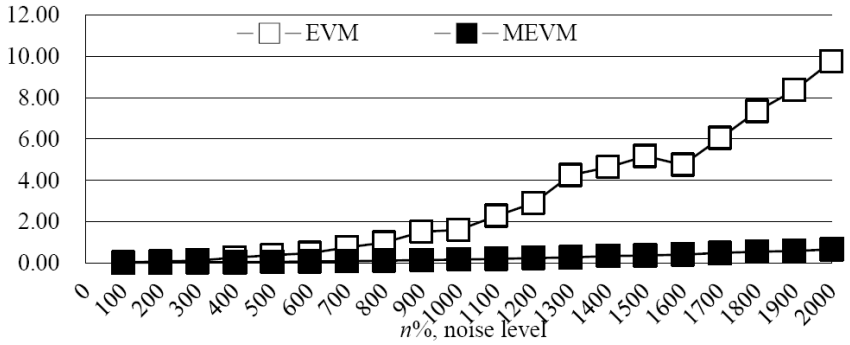


FIGURE 9. The execution time (seconds) of the EVM and MEVM (plugging the mirror-checking algorithm with them EVM) for testing the image in Figure 2. The noise levels ($n\%$) range from 100% to 2000%, and the average execution time (t_{ave}) is obtained from 100 simulations.

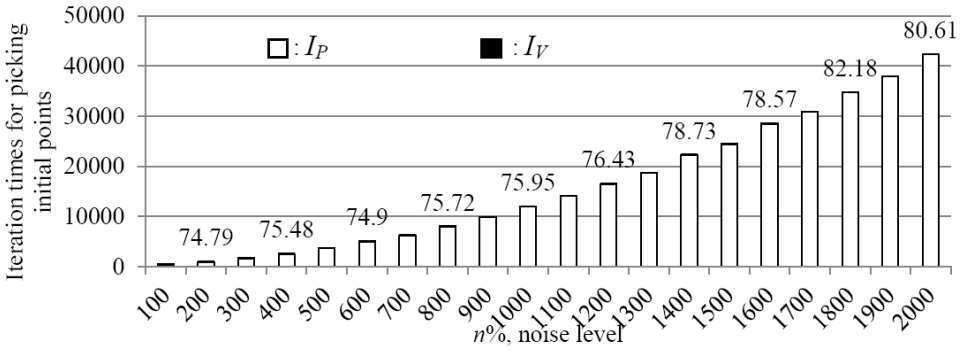


FIGURE 10. The experimental result of average iteration times of selecting initial points of the MEVM (plugging the mirror-checking algorithm with them EVM). The noise levels ($n\%$) range from 100% to 2000%, and the average iteration times of I_V and I_P are obtained from 100 simulations. For example in 2000% noise level, the iteration of I_V is approximated 80.61 times while I_P is over 42,360 times. It denotes the mirror-checking algorithm can efficiently sift most redundant iteration while selecting invalid initial points.

($I_P = 42,360$, which is obtained from 100 simulations) for initial points selection; however, about 80 iteration times ($I_V = 80.61$) with valid initial points forward to the following steps (finding the correspondent searching points and circle detection), and that is why the execution time required is much less than the EVM while noise increasing in Figure 9.

While the SRD regards all picked initial points as valid and finishes all the following steps, the MSRD uses the mirror-checking algorithm to avoid redundant computations for finding correspondent searching points or circle detection while the initial points are invalid. Figure 11 illustrates the average execution time (obtained from 100 simulations, t_{ave}). With 800% noise imposed, the MSRD provides significant better performance than the SRD in t_{ave}. While the SRD regards all random picked initial points as valid and finishes all the following steps, the MSRD applies the mirror-checking algorithm to avoid redundant computations for finding correspondent searching points or circle detection while the initial points are invalid. As shown in Figure 12, as $n\% = 2000\%$, the MSRD performs 220 iteration times ($I_P = 220$) for initial point selection (i.e., Step 2); however,

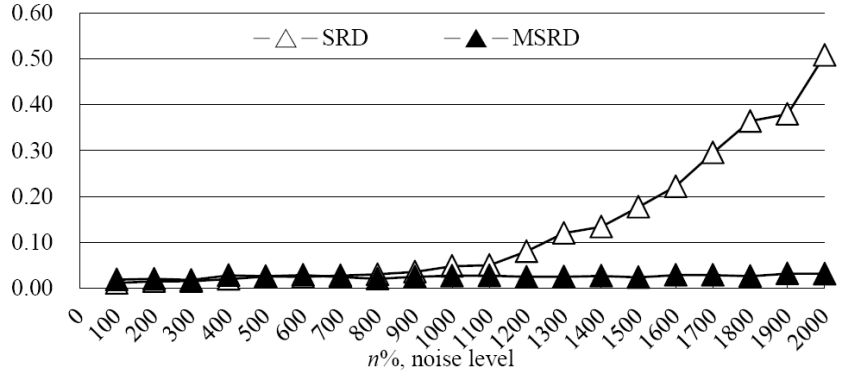


FIGURE 11. The execution time of the SRD and MSRD (plugging the mirror-checking algorithm with the SRD) for testing the image in Figure 2(a). The noise levels ($n\%$) range from 100% to 2000%, and the average execution time (t_{ave}) is obtained from 100 simulations.

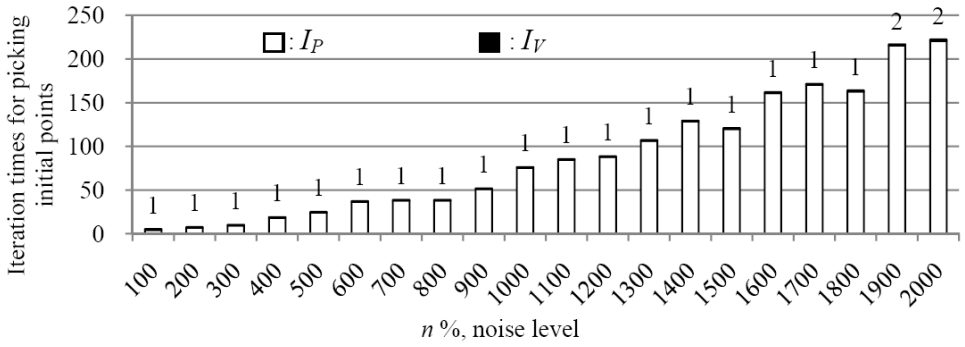


FIGURE 12. The experimental result of average iteration times of selecting initial points of the MSRD (plugging the mirror-checking algorithm with them SRD). The noise levels ($n\%$) range from 100% to 2000%, and the average iteration times of I_V and I_P are obtained from 100 simulations. For example, in 2000% noise level, the iteration of I_V is approximated 2 while I_P is over 220 times. It denotes the mirror-checking algorithm can efficiently sift most redundant iteration while selecting invalid initial points.

only two iteration times ($I_V = 2$) with valid initial points forward to the following steps (finding the correspondent searching points and verifying), and that is why the execution time required is much less than the SRD while noise increasing in Figure 11.

3.5. **Error analysis of the detected circle/arc.** In this section, we make some discussion for the quantization error of circle/arc detection. The error estimation in this thesis is to use two-norm computation, E, as follows:

$$E = \sqrt{(x_c - x')^2 + (y_c - y')^2 + (r_c - r')^2}, \tag{32}$$

where (x_c, y_c, r_c) and (x', y', r') respectively denote the detected and target circle parameters (center position and its radius) in the image space.

We may manually obtain the ground truth data of the target circle in this sample (see Figure 2(a)). However, it is a time consuming process and unreliable. On the contrary, we utilized a coarse-to-fine method to locate the ground truth of the targeted circle [30]. Table 3 shows the ground truth of the targeted circle, and it is accurate to two decimal places. We utilize the target circle in Figure 2(a) to obtain error estimation while imposing

TABLE 3. The ground truth of an artificially made circle, in which the center position is $(168, 90)$ with 37 radius in the image space.

The information of the target circle	x'	y'	r'
Artificially made circle in Figure 2(a)	168	90	37
Ground truth (coarse-to-fine method [30])	167.41	89.51	36.50

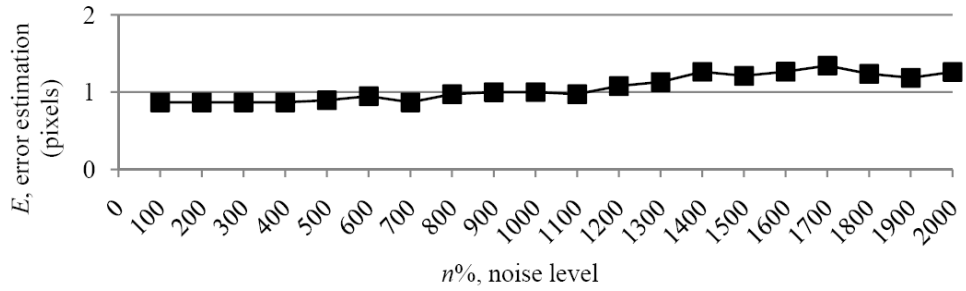


FIGURE 13. The error estimation of the MCD method, with respect to the ground truth obtained from the coarse-to-fine method [30]

noise from 100% to 2000%. Figure 13 illustrates the average error (E_{ave}) of the MCD method from the 100 simulations, and we can see that the error estimation is only around one pixel with respect to the ground truth.

4. **Conclusions.** The proposed MCD belongs to the strategy of the multi-step algorithm and uses initial points for circle detection. It uses a mirror-checking algorithm to sift and reduce redundant/heavy computation for insufficient searching steps. From the analysis of computation, we can see that the proposed MCD requires less computation than others, and the experimental results show that the execution time of the MCD is faster than that of the RCD, EVM and SRD. Not only the proposed mirror-mapping algorithm can avoid redundant computing efficiently, but also it can be plugged to the EVM and SRD to improve the redundant problems. However, a predefined threshold, $w/2$, is just empirical set for checking the symmetric level. How to decide a proper threshold for checking efficiently becomes an important issue. In the future, we will find a statistic/geometric way to decide a proper checking threshold instead of an empirical setting value.

REFERENCES

[1] T. D'Orazio, N. Ancona, G. Cicirelli and M. Nitti, A ball detection algorithm for real soccer image sequences, *Int. Conf. on Pattern Recognition*, DC, USA, vol.1, pp.210-214, 2002.

[2] J. Huang, X. You, Y. Y. Tang, L. Du and Y. Yuan, A novel iris segmentation using radial-suppression edge detection, *Signal Process*, vol.89, no.12, pp.2630-2643, 2009.

[3] Y. Luo, J. Liu and X. Shi, An improved method of RHT to localize circle applied in intelligent transportation system, *Int. Conf. on Audio, Language and Image Processing*, Shanghai, China, pp.335-338, 2008.

[4] N. Qiao, Y. Ye, Y. Huang, L. Liu and Y. Wang, Method of circle detection in PCB optics image based on improved point hough transform, *Proc. of SPIE – The International Society for Optical Engineering*, pp.1-5, 2009.

[5] X. Qingsong, S. Juan and L. Tiantian, A detection and recognition method for prohibition traffic signs, *Int. Conf. on Image Analysis and Signal Processing*, China, pp.583-586, 2010.

[6] M. Shahid and T. Nawaz, Iris detection based on pupil prospect point and horizontal projections, *Proc. of SPIE – The International Society for Optical Engineering*, Singapore, pp.1-6, 2010.

[7] M. Shamsi, P. B. Saad, S. B. Ibrahim, A. Rasouli and N. B. Abdulrahim, A new accurate technique for Iris boundary detection, *WSEAS Trans. on Comput.*, vol.9, no.6, pp.654-663, 2010.

[8] C. C. Wen, S. H. Chiu, J. J. Liaw and C. P. Lu, The safety helmet detection for ATM's surveillance system via the modified Hough transform, *IEEE Annual Int. Carnahan Conf. on Security Technology*, Taoyuan, Taiwan, pp.364-469, 2003.

[9] J. Wu, M. Si, F. Tan and C. Gu, Real-time automatic road sign detection, *Proc. of the 5th Int. Conf. on Image and Graphics*, Suzhou, China, pp.540-544, 2009.

[10] X. Zhu, R. M. Rangayyan and A. L. Ells, Detection of the optic nerve head in fundus images of the retina using the Hough transform for circles, *Journal of Digital Imaging*, vol.23, no.3, pp.332-341, 2010.

[11] T. C. Chen and K. L. Chung, An efficient randomized algorithm for detecting circles, *Computer Vision and Image Understanding*, vol.83, no.2, pp.172-191, 2001.

[12] K. L. Chung and Y. H. Huang, Speed up the computation of randomized algorithms for detecting lines, circles, and ellipses using novel tuning-and-LUT-based voting platform, *Applied Mathematics and Computation*, vol.190, no.1, pp.132-149, 2007.

[13] K. L. Chung and Y. H. Huang, A pruning-and-voting strategy to speed up the detection for lines, circles, and ellipses, *Journal of Information Science and Engineering*, vol.24, no.2, pp.503-520, 2008.

[14] S. H. Chiu and J. J. Liaw, An effective voting method for circle detection, *Pattern Recognition Letters*, vol.26, no.2, pp.121-133, 2005.

[15] S. H. Chiu and J. J. Liaw, A proposed circle/circular arc detection method using the modified randomized Hough transform, *Journal of the Chinese Institute of Engineers*, vol.29, no.3, pp.533-538, 2006.

[16] S. Y. Guo, X. F. Zhang and F. Zhang, Adaptive randomized Hough transform for circle detection using moving window, *Int. Conf. on Machine Learning and Cybernetics*, Dalian, China, pp.3880-3885, 2006.

[17] C. T. Ho and L. H. Chen, A fast ellipse/circle detector using geometric symmetry, *Pattern Recognition*, vol.28, no.1, pp.117-124, 1995.

[18] L. Y. Jiang, Fast detection of multi-circle with randomized Hough transform, *Optoelectronics Letters*, vol.5, no.5, pp.0397-0400, 2009.

[19] H. S. Kim and J. H. Kim, A two-step circle detection algorithm from the intersecting chords, *Pattern Recognition Letters*, vol.22, no.6-7, pp.787-798, 2001.

[20] F. Shang, J. Liu, X. Zhang and D. Tian, An improved circle detection method based on right triangles inscribed in a circle, *World Congress on Computer Science and Information Engineering*, Los Angeles, California, USA, pp.382-387, 2009.

[21] F. Shang, F. Wang, D. Tian and Z. Zhao, A method for circle detection based on right triangles inscribed in a circle, *Acta Optica Sinica*, vol.28, no.4, pp.739-743, 2009.

[22] M. Silveira, An algorithm for the detection of multiple concentric circles, *Lecture Notes in Computer Science: Theoretical Computer Science*, vol.3523, no.2, pp.271-278, 2005.

[23] C. Xing and J. Wang, An improved method for circle detection, *Lecture Notes in Computer Science: Theoretical Computer Science*, pp.463-468, 2008.

[24] L. Xu, E. Oja and P. Kultanen, A new curve detection method: Randomized Hough transform (RHT), *Pattern Recognition Letters*, vol.11, no.5, pp.331-338, 1990.

[25] L. Xu and E. Oja, Randomized Hough transform (RHT): Basic mechanisms, algorithms, and computational complexities, *CVGIP: Image Understanding*, vol.57, no.2, pp.131-154, 1993.

[26] M. Zhang and H. Cao, A new method of circle's center and radius detection in image processing, *Proc. of the IEEE Int. Conf. on Automation and Logistics*, Qingdao, China, pp.2239-2242, 2008.

[27] S. H. Chiu, J. J. Liaw and K. H. Lin, A fast randomized Hough transform for circle/circular arc recognition, *International Journal of Pattern Recognition and Artificial Intelligence*, vol.24, no.3, pp.457-474, 2010.

[28] S.-H. Chiu, K.-H. Lin, W.-Y. Wen, J.-H. Lee and H.-M. Chen, A fast randomized method for efficient circle/arc detection, *International Journal of Innovative Computing, Information and Control*, vol.8, no.1(A), pp.151-166, 2012.

[29] X. Zhang and L. Zhu, Alleviating the computational load of the probabilistic algorithms for circles detection using the connectivity represented by graph, *Machine Vision and Applications*, vol.22, no.4, pp.651-662, 2010.

[30] M. Atiquzzaman, Coarse-to-fine search technique to detect circles in images, *The International Journal of Advanced Manufacturing Technology*, vol.15, no.2, pp.96-102, 1999.