A two-step circle detection algorithm from the intersecting chords

Heung-Soo Kim, Jong-Hwan Kim *

Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Kusong-dong, Yusong-gu, Taejon, South Korea

Received 16 June 2000; received in revised form 11 December 2000

Abstract

This paper proposes a two-step circle detection algorithm using pairs of chords. It is shown how a pair of two intersecting chords locates the center of the circle. Based on this idea, in the first step, a 2D Hough transform (HT) method is employed to find the centers of the circles in the image. In the second step, a 1D radius histogram is used to compute the radii. The experimental results demonstrate that the proposed method can detect the circles effectively.

© 2001 Elsevier Science B.V. All rights reserved.

Keywords: Circle detection; Chord; Hough transform; Radius histogram

1. Introduction

Circle detection is one of the important problems in industrial vision applications such as automatic inspection and assembly (Davies, 1997). It has been researched using various methods (Duda and Hart, 1972; Kimme et al., 1975; Davies, 1988; Yuen et al., 1990). The Hough transform (HT) (Illingworth and Kittler, 1988; Leavers, 1993) has been widely used to extract analytic features, such as straight lines, circles and ellipses. The HT is robust against noises, clutters, object defects, shape distortions, etc. It can be regarded as an efficient implementation of a generalized matched filtering method. However, it requires massive computation and memory. Both complexities grow exponentially with the number of parameters, particularly O(n^2) for circles.

For circle detection, many researchers have developed the modified HT methods using the parameter decomposition and/or some geometric properties of circles to reduce the complexities. Yuen et al. (1990) have performed a comparative study of several HT-based techniques for circle finding. The parameter decomposition-based approaches usually start with the detection of the centers of the circles, then determine the radii. One of those properties is that the normal to a point on a circle passes through the center of the circle (Davies, 1987a; Illingworth and Kittler, 1987). Yip et al. (1992) used the property that two points on a circle whose tangent lines are parallel are the endpoints of the diameter of the circle. The above approaches require the gradient information of

---

*Corresponding author.

E-mail address: johkim@vivaldi.kaist.ac.kr (J.-H. Kim).

1 Associate Editor of the IEEE Transactions on Evolutionary Computation.

0167-8655/01/$ - see front matter © 2001 Elsevier Science B.V. All rights reserved.
PH: S 0 1 6 7 - 8 6 5 5 ( 0 1 ) 0 0 0 2 0 - 4
edge contours which are sensitive to noise (Davies, 1987b). The effect of noise on the edge direction information is generally larger than that on the edge position. There are several approaches without using the edge direction information. Chan and Siu (1990) proposed a fast ellipse detection based on the horizontal and vertical chord bisectors. Similarly, Ho and Chen (1995) proposed a fast detection algorithm of circles using a global geometric symmetry. It computed the circle center from the symmetrical vertical axis and the symmetrical horizontal axis. Sheu et al. (1997) used the symmetric axis information throughout the entire process to compute all five parameters. Goneid et al. (1997) developed the chord bisection method using a 1D array. Davies (1999) studied a simple chord bisection method for the rapid accurate location of ellipses. The method of Ioannou et al. (1999) is based on the property that the line perpendicular bisecting a chord of the circle passes through its center. Lei and Wong (1999) detected the symmetric axes and found pairs of two orthogonal axes whose intersections are the candidates of the centers. Its disadvantage is that straight lines in the image may make the detection of symmetric axes complex.

Yi (1998) proposed a fast finding and fitting algorithm using the geometric symmetry of the circle without HT. Yin (1999) proposed a hybrid scheme using GA and a local search to detect circles and ellipses.

In this paper, a two-step algorithm for finding circles from pairs of chords intersecting each other is proposed. In the first step, the 2D HT is used to compute the centers of the circles. Its idea is based on that a pair of chords can locate the center of the circle. After the points computed from pairs of chords being voted to the 2D accumulator array, the significant peaks are detected as the candidates of the center. In the second step, the 1D radius histogram is used to verify the circles and to compute their radii. The proposed method explores pairs of the chords and does not require any gradient information which may be sensitive to noise. Furthermore, a certain threshold value for detecting the peaks in the 2D accumulator array can be applied to the circles of different sizes because of the normalization of the count by the radius.

This paper is organized as follows. In Section 2, it is shown how a pair of the chords intersecting each other locates the center of the circle. Then, the circle detection method is described in details. In Section 3, the analysis of the proposed algorithm is given. In Section 4, the experimental results with synthetic and real images are demonstrated. Concluding remarks follow in Section 5.

2. Proposed method

2.1. Chord properties

In this section, we introduce the property that relates the center of a circle to its chord and a point on the chord.

Fig. 1(a) shows a circle and its chord $\overline{AB}$. Let $P$ be an interior point of the circle with its center at $O$ and $\overline{AB}$ be the chord passing through $P$ in a $\theta$ direction. The point $A$ is defined as the endpoint in the quadrant I or II of the Cartesian coordinate system centered at $P$, and the point $B$ defined as one in the quadrant III or IV. Let $\theta$ be a positive angle measured in a counterclockwise direction from $\overline{PX}$ to $\overline{OP}$ where $\theta \in [0^\circ, 180^\circ]$. Likewise, let $\phi$ be an angle measured from $\overline{PX}$ to $\overline{OP}$. Then,

$$|\overline{AP}| - |\overline{BP}| = 2|\overline{OP}| \cos(\theta - \phi).$$

(1)

The proof is straightforward in the following. It is well-known that the perpendicular bisector of any chord of a circle passes through its center $O$ (Rich, 1963). Let $C$ be the midpoint of the chord $\overline{AB}$. Then, $|\overline{AC}| = |\overline{BC}|$ and $\overline{OC} \perp \overline{AB}$. So, $|\overline{CP}| = |\overline{OP}| \cos(\theta - \phi)$. $|\overline{AP}| = |\overline{OP}| + |\overline{AP}| \cos(\theta - \phi)$, $|\overline{BP}| = (|\overline{PC}| + |\overline{CP}|) - (|\overline{BC}| - |\overline{CP}|) = 2|\overline{CP}|$. Therefore, $|\overline{AP}| - |\overline{BP}| = 2|\overline{OP}| \cos(\theta - \phi)$. 

In Fig. 1(a), $O(x_o, y_o)$ is related with the coordinates of $P(x_p, y_p)$ as follows:

$$x_o = x_p + a \cos \phi,$$

$$y_o = y_p - a \sin \phi,$$

(2)

where $a = |\overline{OP}|$ and the minus sign in the equation of $y_o$ reflects that $y$ value increases from top to bottom in the image coordinate system. If we have two chords passing through the point $P$, two equations are directly acquired from Eq. (1) so that $a$ and $\phi$ can
be solved with those two equations. As a result, 
\((x_0, y_0)\) can be computed from Eq. (2). The details
are given in the following section.

2.2. Computation for circle center

Consider two chords \(A_1B_1\) and \(A_2B_2\) passing 
through \(P(x_p, y_p)\) in the \(\theta_1\) and \(\theta_2\) directions 
\((\theta_1 \neq \theta_2)\), respectively, as shown in Fig. 1(b). Let 
\(d_1 = |A_1P| - |B_1P|\) and \(d_2 = |A_2P| - |B_2P|\). 

Then, \(d_1 = 2a\cos(\theta_1 - \phi)\) and \(d_2 = 2a\cos(\theta_2 - \phi)\) 
from Eq. (1). Solving for \(\phi\) by dividing two sides 
of the equation of \(d_1\) by ones of the equation of \(d_2\) 
and expanding the cosine functions, we get

\[
\tan \phi = \frac{d_2 \cos \theta_1 - d_1 \cos \theta_2}{d_1 \sin \theta_2 - d_2 \sin \theta_1}.
\]

Using \(a = d_1/(2\cos(\theta_1 - \phi))\) and Eq. (3), we get

\[
a \cos \phi = \frac{d_1}{2\cos(\theta_1 - \phi)} \cos \phi
= \frac{d_1}{2(\cos \theta_1 \cos \phi + \sin \theta_1 \sin \phi)}
= \frac{d_1}{2(\cos \theta_1 + \sin \theta_1 \tan \phi)}
= \frac{d_1 \sin \theta_2 - d_2 \sin \theta_1}{2 \sin(\theta_2 - \theta_1)}.
\]

Similarly,

\[
a \sin \phi = \frac{d_1 \sin \phi}{2 \cos(\theta_1 - \phi)} = \frac{d_1 \tan \phi}{2(\cos \theta_1 + \sin \theta_1 \tan \phi)}
= \frac{d_2 \cos \theta_1 - d_1 \cos \theta_2}{2 \sin(\theta_2 - \theta_1)}.
\]

Combining these results with Eq. (2), the center
\(O(x_o, y_o)\) of the circle is

\[
x_o = x_p + \frac{d_1 \sin \theta_2 - d_2 \sin \theta_1}{2 \sin(\theta_2 - \theta_1)},
\]

\[
y_o = y_p - \frac{d_2 \cos \theta_1 - d_1 \cos \theta_2}{2 \sin(\theta_2 - \theta_1)}.
\]

Eq. (4) shows us how the center of the circle can be 
computed from two intersecting chords. Specifically,
if we have an interior point \(P\) of the circle and 
two chords passing through it, \(d_1, d_2, \theta_1\) and \(\theta_2\) 
given by these chords can locate the center of the 
circle from Eq. (4). One of the simplest cases is for 
\(\theta_1 = 0^\circ\) and \(\theta_2 = 90^\circ\). In that case, it can be easily 
shown that \(x_o\) is the midpoint of the horizontal 
chord of \(\theta_1 = 0^\circ\) and \(y_o\) is the midpoint of the 
vertical chord of \(\theta_2 = 90^\circ\). This case is computationally 
efficient, but may not be robust because it 
uses only the vertical and horizontal chords. From 
this reason, the chords with randomly generated 
directions will be used along with Eq. (4) in this paper.
2.3. A two-step circle detection algorithm

Until now, it is assumed that the pair of the chords \((A_1B_1, A_2B_2)\) was known in the image. However, it would be difficult to identify and locate exactly the corresponding chords of each circle in the image with many objects. Besides, one evaluation from only two chords would not be reliable because there may be some uncertainties caused by digitization, noise, shape distortion, etc., in the image.

The HT can solve this problem. The outlines are as follows. Scanning the edge image from left to right and top to bottom, we find the edge points on two arbitrary lines passing through each pixel \((i, j)\) and compute the center candidates of the circles by applying the chord candidates generated from those edge points by Eq. (4), where each pixel \((i, j)\) corresponds to the point \(P\) in Fig. 1(b). The computed points are voted to the 2D accumulator array of the parameter space. After the voting process, the significant peaks which indicate the existence of the centers of the circles are detected through a peak detection method. Finally, applying the radius histogram method to each center candidate, we extract the radii and verify the circles.

In this work, the parameter space is congruent with the image space. This choice is natural but not necessary. The coarser parameter space may be more robust and less accurate. For the peak detection in the accumulator, nonmaximum suppression filtering before thresholding is used (Amir, 1990). The cells with smaller votes than any of the other cells in the neighborhood are set to 0. The size of the neighborhood relying on the minimum distance that the circle centers are to be separated, is set to \(5 \times 5\) pixels here. For the verification of the circles, the filtered radius histogram by Ioannou et al. (1999) is adopted, which is described by the following equation

\[
\frac{1}{4\sqrt{2}\pi r} \left[ \frac{-3r}{2(r - 2\Delta r)}, 1, 1, 1, \frac{3r}{2(r + 2\Delta r)} \right],
\]

where \(r\) is the radius and \(\Delta r\) is the size of the cells of the histogram. Its threshold does not depend on the radius of the circle and can effectively detect the radius.

The several notations are introduced before describing the proposed algorithm. \(E\) is a given thinned edge image, \(\text{ACC}[y|x]\) a 2D accumulator array, and \(I, II, III, IV\) four quadrants in the Cartesian coordinates centered at pixel \(P(i, j)\). For the chord passing through \(P\) in the \(\theta_1\) direction, its endpoint in the quadrant I or II is denoted by \(A_1\) and the other in the quadrant III or IV denoted by \(B_1\). \(d_{i_1}\) is the distance between \(P\) and \(A_1\), and \(d_{i_2}\) the one between \(P\) and \(B_1\). \(I_1(l_2)\) is the line passing through \(P\) in the \(\theta_1(\theta_2)\) direction. \(L_1\) represents a 2D array to store the positions of \(A_1\)‘s, the edge pixels on \(l_1\) in the quadrant I or II, and \(L_2\) one to store \(B_1\)‘s, the edge pixels on \(l_1\) in the quadrant III or IV. For the chord in the \(\theta_2\) direction, \(A_2, B_2, d_{i_2}, d_{i_2}, L_2\) and \(L_2\) are similarly defined. \((x_{i_1}, y_{i_1})\) and \((x_{i_2}, y_{i_2})\) represent the coordinates of \(A_1\) and \(A_2\), respectively. \(L_1\) is a list of the center candidates of the circles. The detailed algorithm is as follows.

**ALGO_CIRCLE_DETECTION**

1. Initialize all cells of \(\text{ACC}\) to 0.
2. Scan \(E\) from left to right and top to bottom.
3. For each pixel \(P(i, j)\) on \(E\), do the following steps 3.1 to 3.8:
   3.1. Initialize \(L_1, L_2, L_1\), and \(L_2\).
   3.2. Generate \(\theta_1\) and \(\theta_2\) randomly, where \(\theta_1 \neq \theta_2 \in [0^\circ, 180^\circ]\).
   3.3. For each edge pixel on the line \(l_1\),
       3.3.1. If the edge pixel is in the quadrant I or II, add it to \(L_1\).
       3.3.2. Otherwise, add it to \(L_2\).
   3.4. Go to step 3 until all edge pixels on \(l_1\) are considered.
   3.5. For the line \(l_2\), make \(L_2\) and \(L_2\) through the similar steps from 3.3 to 3.4.
   3.6. Compute \(d_{i_1}, d_{i_1}, d_{i_2}\), and \(d_{i_2}\) for each combination of four points \(A_1, B_1, A_2, B_2\) in \(L_1, L_1, L_2,\), and \(L_2\), respectively.
   3.7. If \((d_{i_1} + d_{i_2}) > d_{\min}\) and \((d_{i_2} + d_{i_2}) > d_{\min}\), do the following steps:
       3.7.1. Compute \((x_0, y_0)\) from Eq. (4) with
       \[d_1 = d_{i_1} - d_{B_1}\] and \[d_2 = d_{i_2} - d_{B_2}\].
       3.7.2. Compute
       \[r_1 = \sqrt{(x_0 - x_{i_1})^2 + (y_0 - y_{i_1})^2}\] and
       \[r_2 = \sqrt{(x_0 - x_{i_2})^2 + (y_0 - y_{i_2})^2}\].
3.7.3. If \(|r_1 - r_2| < \delta r_{\min}\), increment the
ACC at \((x_0, y_0)\) by
\[
\text{ACC}[y_0][x_0] = \text{ACC}[y_0][x_0] + 1/r^2,
\]
where \(r = (r_1 + r_2)/2\).

3.8. Go to step 3.6 until all combinations of
\(A_1s, B_1s, A_2s\) and \(B_2s\) are considered.

4. Go to step 3 until all pixels on \(E\) are considered.

5. Find all significant peaks in ACC and store them to LCC.

6. Extract the radii via the radius histogram of
each candidate of \(LCC\).

7. End.

\(LA_1\) and \(LB_1\) may have more than one point
because there may exist multiple edge points on
\(PA_1\) and \(PB_1\). Similarly, \(LA_2\) and \(LB_2\) may, too.

In the proposed algorithm, \(d_{\text{min}}\) is used to avoid
the sensitivity from too short chords of which the
extreme case is for two adjacent edge pixels. To
find the edge pixels on the line \(l_1\) (or \(l_2\)), the
intersection of the line with each of the grids which
define the pixels of the image is calculated.
It should be noted that only when the difference
between \(r_1\) and \(r_2\) is less than \(\delta r_{\min}\), the computed
result \((x_0, y_0)\) is voted to ACC. It is based on the idea
that the four endpoints of the two chords should
be equidistant from the center. The verification
with \(\delta r_{\min}\) prevents the points not belonging to the
circle perimeter from contributing to the voting.

The increment of ACC by Eq. (6) results in the
normalized peak values which are typically
around 5. Thus, a certain threshold can be used
to detect the peaks, regardless of the size of the
circles. Consider a discrete disk, which is defined
as a set of raster points consisting of a discrete
circle of the radius \(r\) and all points lying inside it.
\(D(r)\) denotes the number of point in the disk.
Then, \(\lim_{r \to \infty} D(r)/r^2 = \pi\) was proved by Kulpa
(1979). It can be seen that each pixel inside the
digital circle can always take any two chords
passing through itself and the number of such
pixels is equal to \(D(r)\). Thus, the peak value
normalized by \(1/r^2\) is expected to be approximate-
ly \(\pi\). However, through our experiments for
the circles with the radius between 30 and 100
pixels, the peak value ranges from 4 to 6, greater
than \(\pi\). It is because the intersection points of a
digital circle with the line containing its chord are
not always two, i.e., the endpoint of the chord
such as \(A\) or \(B\) is not unique because of the
digitization.

3. Analysis of the algorithm

In this section, the complexity and storage re-
quirements of the proposed method are analyzed
and the reduction of the computational loads is
discussed.

Consider an \(N \times N\) image. The proposed algo-

rithm requires one \(N \times N\) 2D array for ACC, and
five 2D arrays of the size less than \(3N \times 2\) for \(LA_1, LA_2, LB_1, LB_2\)
and the radius histogram. Thus, the
\(N \times N\) array is dominant storage requirement.
Finding the edges intersecting with two ran-
domly generated lines and voting by pairs of
chords generated from those edges have to be done
at each pixel. For \(x\) or \(y\) axis with \(N\) intervals,
there are at most \((N + 1)\) intersections between a
line and grids in the image. So, finding the pixels
on two lines requires maximum \(4(N + 1)\) calcula-
tions. Let \(N_c\) be the maximum number of the
endpoints stored in such arrays as \(LA_1, LA_2, LB_1\)
and \(LB_2\). Then, the combinations of four points
from those arrays are \(N_c^4\). The whole image has \(N^2\)
pixels, thus the total computation time of the algo-
rithm is \(O(N^3) + O(N^4N^2)\), where the first term
is the time for finding the edges intersecting with
the lines and the second one is for voting by pairs
of chords.
Although the complexity is high in the general
case, it can be significantly reduced in many
practical cases. If the circles to be detected in the
image are simple in the sense that they have few
objects or few edge points inside themselves, \(N_c\)
can be small (e.g., 3) so that \(O(N_c^4N^2)\) is less than
\(O(N^3)\) when \(N = 256\). In the process of finding the
edges on the lines, started from \(P\), i.e., making
such arrays as \(LA_1\) and \(LB_1\) in the algorithm, once
the intersections of the grids with the lines are
greater than \(N_c\), it can be stopped. Thus, the cal-
culations are less than \((N + 1)\). Another simple
strategy for the speedup is a kind of sampling
approach that the image is scanned every \(s\)
horizontal and vertical pixels so that voting at only those sampled pixels is done. In that case, the maximum peak value in ACC is decreased by the factor $x^2$. To reduce the computational complexity of trigonometric functions in Eq. (4), the lookup table for sine and cosine functions can be used. In addition, the edge thinning operations which may be time-consuming can be omitted if a little extra smoothing of peaks for the centers is tolerant. But, there is a tradeoff between the omission and the increased number of edge pixels.

4. Experimental results

The proposed algorithm ALGO_CIRCLE_DETECTION was implemented in C++ and executed on a PC with Pentium III 500 MHz CPU. Its performance was demonstrated with three synthetic and two real $256 \times 256$ gray images. Sobel operator detected the edges in the image and the resulted edge image was thinned. $\theta_1$ and $\theta_2$ were taken from integer values from 0 to 179, and the lookup table for sin and cos functions of each

Fig. 2. (a) Original image; (b) detected circles superimposed on the thinned edge image; (c) voting space from the $x$-side view by the proposed method; (d) voting space from the $x$-side view by the gradient method.
integer-valued $\theta$ was used. For the speedup discussed in the previous section, different values of $N_c$ and $s$ were tested. The thresholds for the edge magnitude, ACC and the radius histogram were set to 200, 1.5 and 0.5, respectively. $d_{\min} = 10$, $N_c = 4$ or 10 and $\Delta r = 1$ were used.

Fig. 2(a) shows a synthetic image with two circles where one circle is $(x_0, y_0, r) = (134, 134, 84)$ and the other one $(124, 144, 44)$. The first circle is almost twice as large as the second one and their centers are close to each other. Fig. 2(b) shows the circles detected by the proposed method ALGO_CIRCLE_DETECTION, superimposed on the thinned edge image. The ‘+’ mark indicates the detected circle centers. The results were $(134, 134, 84)$ and $(124, 144, 44)$ which coincided with the original ones. For the clearness, the accumulator array plotted from the $x$-side view is shown in Fig. 2(c). The peaks representing the circle centers were well separated. The peak values were 5.51 and 5.19 which are bigger than $\pi$, as before-mentioned, and

Fig. 3. (a) Original image; (b) result when adding salt and pepper noise of level 1; (c) result when adding salt and pepper noise of level 2; (d) voting space of (c) from the $x$-side view.
roughly equal regardless of their radii. On the contrary, the voting by the gradient method using edge direction information (e.g., Davies, 1997; Illingworth and Kittler, 1987) for the image filtered with $7 \times 7$ Gaussian filter suffered from two smoothing peaks (Fig. 2(d)). Even though not shown here, the result by the gradient method without filtering the image was much worse. That is because, roughness on the edge positions of the synthetic circles makes the edge orientation incorrect.

A synthetic image including several objects is shown in Fig. 3(a). The proposed algorithm was applied to the cases of adding the different levels of salt and pepper noise to the original thinned edge image. The detection results are shown in Fig. 3(b) and Fig. 3(c). The algorithm still worked well. In Fig. 3(d), the peaks indicating the centers are well-shaped even in the presence of noise.

Figs. 4(a) and 4(b) show a real image with 9 circles of different sizes and its thinned edge image,
respectively. In Fig. 4(c) showing the voting space from the x-side view, the peak value at \( x = 125 \) is about 9.85, which is bigger than \( 3\pi \). It is because there are three concentric circles. The detected circles are shown in Fig. 4(d).

The second real image is shown in Fig. 5(a) which contains two circles, particularly one circle occluded. Fig. 5(b) shows the thinned edge image for the median filtered image. Fig. 5(c) shows the 3-dimensional voting space by the proposed method. Even though one of two circles was occluded by other object, it was successfully detected, as shown in Fig. 5(d).

Table 1 shows the average execution time in five trials for the four test images under two different values of \( N_e \) and \( s \), respectively. The execution time for \( N_e = 4 \) was shorter than \( N_e = 10 \). The time for \( s = 3 \) was much shorter than \( s = 1 \), approximately by 9 (\( = 3^2/1^2 \)). As expected, the cases of smaller \( N_e \) and larger \( s \) took shorter execution time. However, if too small \( N_e \) is used for the circles containing other objects

Fig. 5. (a) Original image; (b) thinned edge image; (c) voting space; (d) detected circles superimposed on the thinned edge image.
Table 1
The average execution time (s) in five trials under different parameters

<table>
<thead>
<tr>
<th>Figure</th>
<th>$s = 1$</th>
<th>$s = 3$</th>
<th>$N_c = 4$</th>
<th>$N_c = 10$</th>
<th>$N_c = 4$</th>
<th>$N_c = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(a)</td>
<td>3.18</td>
<td>3.72</td>
<td>0.38</td>
<td>0.44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3(a)</td>
<td>4.50</td>
<td>9.52</td>
<td>0.54</td>
<td>1.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4(a)</td>
<td>6.84</td>
<td>24.93</td>
<td>0.82</td>
<td>2.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(a)</td>
<td>5.56</td>
<td>22.85</td>
<td>0.73</td>
<td>2.66</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6. (a) A partially occluded circle; (b) voting space from the proposed method ($s = 1, N_c = 10$); (c) mid-points of the horizontal chords; (d) detection result by the proposed method.

inside, the peak values of ACC may be significantly reduced, for instance, the biggest circle containing other eight circles in Fig. 4(a).

Although the comparison of the proposed method with other circle detection methods using the chord was not done thoroughly in this
work, we observed that most other techniques (i.e., Chan and Siu, 1990; Ho and Chen, 1995; Sheu et al., 1997; Goneid et al., 1997) used the vertical and horizontal chords and required the straight line HT (SHT) to detect the symmetrical vertical and horizontal axes. So, they would not work well in the case of the partially occluded circles and would spend much time for the voting process and the peak detection in the SHT. Fig. 6(a) shows the simple case of the circle partially occluded by a rectangle. Fig. 6(b) is the voting space by the proposed method when \( s = 1 \) and \( N_\text{c} = 10 \). It has a significant peak corresponding to the circle center. Fig. 6(c) is the blob image composed of the mid-points of the horizontal chords to find the candidate symmetrical vertical axes. Two short line segments belonging to the symmetrical vertical axis and one spurious vertical line segment appeared. It is not easy to detect them using the simple SHT and the partially occluded circle may not be detected. On the contrary, the proposed method uses the randomly selected orientations which enable the detection of the occluded circle (Fig. 6(d)), the lookup table to avoid the complex computation of the trigonometric function at the expense of the additional small memory, and the sampling rate \( s \) to speed up the detection process.

5. Conclusions

This paper proposed a two-step HT-based circle detection algorithm using pairs of chords intersecting each other. The formula for computing the center of the circle from a pair of the intersecting chords was derived. In the first step, based on the formula, the 2D HT method found robustly and effectively the centers of the circles in the image. In the second step, the radius of the circle was computed using the radius histogram. The proposed method does not use the edge direction information which is sensitive to noise. Through the normalization by \( r^2 \) in the voting of ACC, the thresholding is simple and general. The analysis of the proposed algorithm was given and the reduction of the complexity was discussed. The experimental results for synthetic and real images demonstrated that the proposed method works well for detecting the circles.

Acknowledgements

This work was supported by Ministry of Science and Technology (MOST) under grant 98-N9-01-01-A.

References
