Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 190 (2007) 132–149
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Speed up the computation of randomized algorithms for detecting lines, circles, and ellipses using novel tuning- and LUT-based voting platform

Kuo-Liang Chung *,[1], Yong-Huai Huang

Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC

Abstract

Shape detection is a fundamental problem in image processing field. In shape detection, lines, circles, and ellipses are the three most important features. In the past four decades, the robustness and the time speedup are two main concerned issues in most developed algorithms. Previously, many randomized algorithms were developed to speed up the computation of the relevant detection successfully. This paper does focus on the time speedup issue. Based on Bresenham's drawing paradigm, this paper first presents a novel lookup table (LUT)-based voting platform. According to the proposed LUT-based voting platform, we next present a novel computational scheme to significantly speed up the computation of some existing randomized algorithms for detecting lines, circles, and ellipses. Moreover, the detailed time complexity analyses are provided for the three concerned features under our proposed computational scheme and these derived nontrivial analyses also show the relevant computational advantage. Under some real images, experimental results illustrate that our proposed computational scheme can significantly speed up the computation of some existing randomized algorithms. In average, the execution-time improvement ratios are about 28%, 56%, and 48% for detecting lines, circles, and ellipses, respectively, and these improvement ratios are vary close to the theoretic analyses.

Keywords: Bresenham's drawing paradigm; Circles; Ellipses; Lines; Randomized algorithms

1. Introduction

Shape analysis is a fundamental problem in image processing field [1–3]. In shape analysis, lines, circles, and ellipses are the most three important features since they often occur in the image. In the past four decades, the robustness and the time speedup are two main concerned issues in most developed algorithms. These

* Corresponding author.
E-mail address: klchung@csie.ntust.edu.tw (K.-L. Chung).
[1] Supported by National Council of Science of ROC under contract NSC95-2218-E011-009.

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.01.012

developed algorithms are partitioned into two categories, namely the deterministic algorithms and the nondeterministic algorithms.

Most of the existing deterministic algorithms are based on the Hough transform (HT) [4,5]. Assume the input image have been transformed into its edge map. The HT maps each edge pixel into a parameter space, i.e. (ρ, θ)-space where ρ denotes the normal distance and θ denotes the normal angle. Although these HT-based algorithms have robustness gains, a large amount of computing time is needed in the voting process of the relevant accumulator arrays which are used to emulate the corresponding parameter spaces. Therefore, many improved HT-based algorithms [6–26] have been developed to reduce the computing time by utilizing some geometrical properties and the gradient information to decompose the parameter space into fewer dimension parameter space. However, these computing time improvement still can not meet the real-time demand.

Based on the randomized approach, many nondeterministic algorithms have been developed to speed up the computing time to meet the real-time demand. Among these developed nondeterministic algorithms [27–44], Kiryati et al. [30] presented a probabilistic HT (PHT) to detect lines. The PHT only uses a small, randomly selected subset of edge pixels to perform the HT on the (ρ, θ)-space. Later, several improved versions of the PHT were developed and they are the progressive PHT (PPHT) [39], the gradient-based PPHT (GPPHT) [42], and the one with novel stopping rules [35]. Employing the candidate line concept, Xu et al. [29,31] presented an efficient randomized HT (RHT) for line detection. For convenience, Xu et al.'s method is called the RHTL. Following the RHTL, some improved versions [36,40] were investigated. Recently, plugging the pruning-and-voting (PAV) strategy into the RHTL, the PRHTL [44] was developed to speed up the candidate line-based randomized algorithms.

For detecting circles, the probabilistic circular HT (PCHT) was proposed by Ylä-Jääski and Kiryati [32]. Based on the idea in the PHT, the PCHT uses a very efficient stopping scheme, which is measured by the stability test via the ranks of the highest peaks in the accumulator array, to speed up the execution-time performance and increase the robustness performance. In [29,31], an efficient randomized HT approach for circle detection, which is called the RHTC, was presented by Xu et al. Each time, the RHTC selects three edge pixels randomly and then their corresponding mapped points in the parameter space are collected by voting on the accumulator array or the link-list data structure. Based on a parameter-free approach, an improved randomized algorithm [41], called the RCD, was presented to detect circles. Later, the PAV-based RCD [44], called the PRCD, was developed to speed up the RCD. Employing some geometrical properties [12], McLaughlin [37] proposed a RHT-based approach for detecting ellipses. For convenience, McLaughlin's two-stage method is called the RHTE. Experimental results show that the RHTE has better execution-time when compared to the previous two methods [15,12]. Combining the PAV strategy and the RHTE, the PRHTE [44] was proposed to improve the execution-time performance of the RHTE.

This paper does focus on presenting a new computational scheme to speed up the computational requirements in several existing randomized algorithms for detecting the three concerned features. Based on Bresenham's drawing paradigm [45–48], this paper first presents a novel lookup table (LUT)-based voting platform. According to the proposed LUT-based voting platform, we next present a novel computational scheme to significantly speed up the computation required in the previous RHTL [29,31], RCD [41], and RHTE [37] for detecting lines, circles, and ellipses, respectively, while preserving the same robustness. Moreover, for each concerned feature, a solid time complexity analysis of our proposed computational scheme is provided to show the relevant computational advantage. In average, experimental results show that the execution-time improvement ratios are about 28%, 56%, and 48% for detecting lines, circles, and ellipses, respectively, and these improvement ratios are vary close to our theoretic time complexity analyses. Further, experimental results also show that plugging our proposed LUT-based voting platform into the previous three algorithms can lead to higher execution-time improvement ratios when compared to the PAV-based algorithms [44].

The remainder of this paper is organized as follows. In Section 2, according to Bresenham's drawing paradigm, the proposed LUT-based voting platform is presented. In Section 3, based on the proposed tuning- and LUT-based voting platform, a novel computational scheme and its analyses of computation gains are presented. In Section 4, the proposed computational scheme leads to significantly speed up the computation of some existing randomized algorithms for detecting lines, circles, and ellipses. In addition, the related nontrivial time complexity analyses are provided. Section 5 demonstrates the experimental results. Finally, Section 6 addresses some concluding remarks.

2. The proposed novel LUT-based voting platform

This section is partitioned into two subsections. In Section 2.1, we first review the previous candidate feature-based randomized algorithms (CFRAs), such as the RHTL, the RCD, and the RHTE. In the same subsection, we also point out the computational bottleneck existed in the previous CFRAs. In order to overcome the computational bottleneck in the CFRAs, based on Bresenham's drawing paradigm [45,46,48,47], our proposed LUT-based voting platform is presented in Section 2.2.

2.1. The previous candidate feature-based randomized algorithms: CFRAs

Definition 1 (*Candidate Features*). Lines, circles, or ellipses determined by a few randomly selected edge pixels are called candidate features (CFs).

Definition 2 (*CF-based randomized algorithm*). For the CF-based randomized algorithms (CFRAs), such as the RHTL, the RCD, and the RHTE, the coordinates of all edge pixels $v_l = (x_l, y_l)$ are initially stored in the set V. For determining the CF, the CFRA first randomly select a few edge pixels from V to determine the CF. After the CF is detected, the voting process is applied to calculate the distance between each of the remaining edge pixels and the CF, then the counting step is performed to sum up the number of edge pixels lying on the CF. From the counting result, the CFRA can finally determine whether the CF is a true feature or not.

By Definition 2, the CFRA consists of the following four steps:

Step 1. (Initialization) Save each edge pixel $v_l = (x_l, y_l)$ to the set V. Let C_f and C_v denote the failure counter and the voting counter, respectively. Two thresholds T_f and T_{min} are used where T_f denotes the number of failures that we can tolerate; T_{min} is used to determine whether the CF is a true feature or not.

Step 2. (Determine the CF) Randomly picking a few edge pixels out of V to determine the CF. For line detection, the RHTL takes two edge pixels to determine the candidate line. For circle detection, the RCD takes four edge pixels to determine the candidate circle. For ellipse detection, three edge pixels are selected by the RHTE to construct the candidate ellipse.

Step 3. (Voting process) Set the voting counter C_v to 0. For each edge pixel $v_l = (x_l, y_l)$ in V, the distance between the CF and the edge pixel v_l, say $d_{l \to F}$, is calculated. If the value of $d_{l \to F}$ is less than or equal to the threshold $\triangle$, say $\triangle = 1$, perform $C_v = C_v + 1$.

Step 4. (Determine the true feature) Using the final value of C_v, the CFRA can determine whether the CF is a true feature or not. In the RHTL, if the value of C_v is greater than or equal to T_{min}, the CF is a true feature. In the RCD and RHTE, the CF is a true feature if the ratio of C_v over the perimeter of the CF is greater than the threshold T_{min} (the estimation for the perimeter of the ellipse is shown in Appendix 1). If the CF is a true feature, we take these C_v edge pixels out of V and go to Step 2. Otherwise, the CF is viewed as a false feature and perform $C_f = C_f + 1$. If $C_f = T_f$, then stop; otherwise, go to Step 2.

After running many testing images for the three previous CFRAs, experimental results indicate the following property.

Property 1 (Computational bottleneck in the CFRA). *On the Pentium* 4 *personal computer with* 1.8 *GHz and the C programming language, the ratios of the execution-time required in the voting process* (*see Step* 3 *in the above CFRA*) *over the total execution-time for the RHTL, the RCD, and the RHTE are about* 65%, 91%, *and* 72.5%, *respectively. Consequently, the voting process is the computational bottleneck in the CFRA.*

From Property 1, to reduce the execution-time required in the voting process significantly, the proposed LUT-based voting platform is presented in the following subsection.

2.2. The proposed LUT-based voting platform

Instead of calculating the distance between the edge pixel and the CF in the voting process of the CFRA, our proposed LUT-based platform only performs one simple query such that whether the edge pixel is lying

on the CF or not can be determined fast. The proposed LUT-based platform first creates a 2-D binary array, say A_{lut}, which is based on the Bresenham's drawing paradigm. Given the related parameters of one CF, that CF can be drawn on an integer domain very fast. For example, given two end points, the corresponding line can be drawn very fast and concerns only simple integer additions and shift operations.

Definition 3 (*LUT-based platform*). According to the related parameters of the CF, using Bresenham's drawing paradigm to draw the CF on the 2-D binary array where the bit '1' denotes the feature point. The array A_{lut} associated with the drawn CF is called the LUT-based platform.

Let the related parameters of the candidate circle be the center (x_c, y_c) and the radius r. For example, given $(x_c, y_c) = (16, 17)$ and $r = 12$, by Definition 3, the corresponding LUT-based platform is depicted in Fig. 1.

Definition 4 (*Voting on LUT-based platform*). The (x, y)-coordinate of each edge pixel, say $v_l = (x_l, y_l)$, is used as the key to perform the query on the LUT-based platform A_{lut}. If $A_{\text{lut}}(x_l, y_l) = 1$, it means that the edge pixel v_l is lying on the CF and it contributes a vote to the CF.

By Definition 4, the number of edge pixels lying on the CF are summed up to determine whether the CF is a true feature or not.

Definition 5 (*Cleaning up the LUT-based voting platform*). Before detecting the next feature in the image, using the Bresenham's drawing paradigm again, each entry of the CF in the LUT-based voting platform is reset by changing its value from '1' to '0'.

From Definitions 3–5, the CFRA associated with our proposed LUT-based voting platform is presented as follows. For convenience, the LUT-based CFRA is called the LCFRA. In the following six steps, the proposed LUT-based voting platform is realized by Steps 3–5.

Step 1. (Initialization) After performing the initialization process in Step 1 of the CFRA, we reset the LUT-based voting array $A_{\text{lut}}(W, H)$ where W and H denote the width and height of testing image.

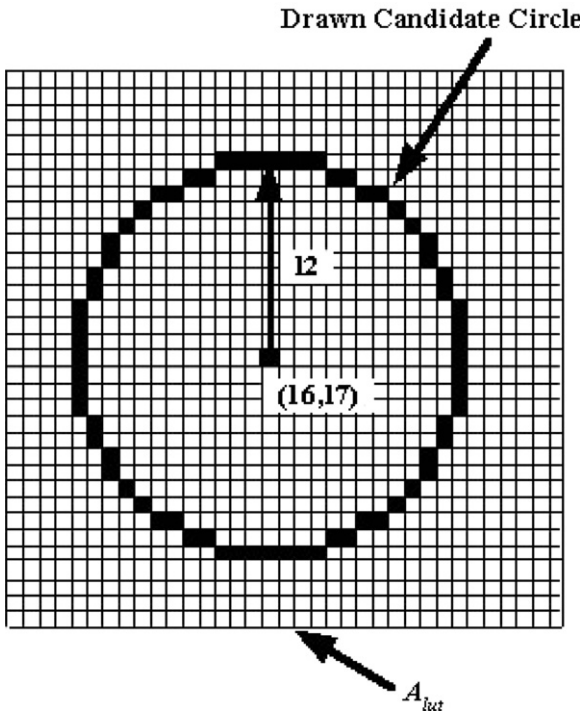

Fig. 1. The LUT-based platform for the candidate circle with center (16, 17) and radius 12.

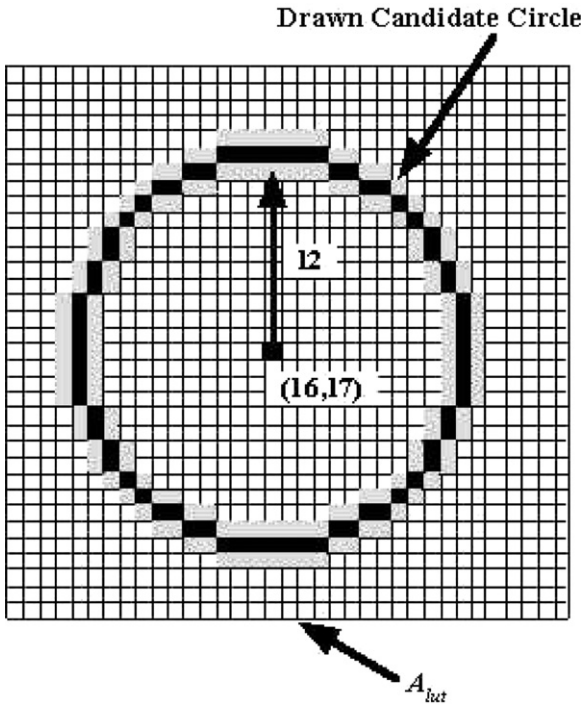

Fig. 2. The banded LUT-based voting platform with bandwidth 1.

Step 2. (Determine the CF) This step is equal to Step 2 in the CFRA.

Step 3. (Building up the LUT-based voting platform) By Definition 3, Bresenham's drawing paradigm is applied to build up the LUT-based voting platform.

Step 4. (Voting process) Set the voting counter C_v to 0. By Definition 4, for each edge pixel $v_l = (x_l, y_l)$ in the edge set V, if $A_{lut}(x_l, y_l) = 1$ holds, perform $C_v = C_v + 1$.

Step 5. (Cleaning up the LUT-based voting platform) By Definition 5, Bresenham's drawing paradigm is applied to clean up the LUT-based voting platform.

Step 6. (Determine the true feature) This step is equal to Step 4 in the CFRA.

In order to detect the CFs with higher tolerance, the proposed LUT-based voting platform can be extended to a banded LUT-based voting platform where the bandwidth is specified by the user. Let the bandwidth be 1, then the banded LUT-based voting platform of Fig. 1 is shown in the banded candidate circle of Fig. 2.

3. The proposed tuning- and LUT-based computational scheme: TLCFRA

Plugging the tuning concept into the CFRA associated with the LUT-based voting platform, which has been described in Section 2, a new computational scheme is presented in this section. In addition, the analyses of its computational gains are provided.

Before demonstrating the computational gains of our proposed new tuning- and LUT-based computational scheme, for each pixel, the voting-time required in the past CFRA is evaluated first. In the voting process of the CFRA, it first calculates the distance between the edge pixel $v_l = (x_l, y_l)$ and the CF, then one comparison is used to decide whether v_l is lying on the CF or not, and one extra addition is needed in the counting step if v_l is lying on the CF. Consequently, in worst case, the voting-time required for each edge pixel in the CFRA can be represented by $t_v = t_{dis} + t_c + t_a$ where t_{dis} denotes the time for the distance calculation; t_c and t_a denote the time required to perform one comparison and one addition, respectively. Following the voting-time notation t_v, the CFRA takes $T_v = m \times t_v = m \times (t_v + t_c + t_a)$ time in the voting process where m denotes the number of edge pixels in the current edge set.

Let t_{lut} denote the time required to build up or clean up the CF on the LUT-based voting platform A_{lut}. From Step 4 of the LCFRA, for each edge pixel, the voting process needs one comparison to determine whether the edge pixel is a CF point or not and one addition is needed for increasing the counter C_{v} by one. In worse case, Step 4 of the LCFRA needs $t_{\text{v}}^{\text{lut}} = t_{\text{c}} + t_{\text{a}}$ time. From Step 3 and Step 5 in the LCFRA, for each CF, $2t_{\text{lut}}$ time is needed to build up and clean up the LUT-based voting array A_{lut}. For m edge pixels, the voting-time required in Steps 3–5 of the LCFRA is at most $T_{\text{v}}^{\text{lut}} = m \times t_{\text{v}}^{\text{lut}} + 2t_{\text{lut}} = m \times (t_{\text{c}} + t_{\text{a}}) + 2t_{\text{lut}}$ time.

Definition 6 (*Tuning strategy*). The voting-time improvement ratio of our proposed LCFRA over the previous CFRA is represented by $I_{\text{v}} = \frac{T_{\text{v}} - T_{\text{v}}^{\text{lut}}}{T_{\text{v}}} = \frac{t_{\text{dis}}}{t_{\text{dis}}+t_{\text{c}}+t_{\text{a}}} - \frac{2t_{\text{lut}}}{m(t_{\text{dis}}+t_{\text{c}}+t_{\text{a}})}$. If $\frac{t_{\text{dis}}}{t_{\text{dis}}+t_{\text{c}}+t_{\text{a}}} - \frac{2t_{\text{lut}}}{m(t_{\text{dis}}+t_{\text{c}}+t_{\text{a}})} > 0$ holds, the computational gain of our proposed LCFRA can be guaranteed. On the other hand, when $m \geqslant \frac{2t_{\text{lut}}}{t_{\text{dis}}}$, our proposed LUT-based voting platform can replace the voting process in the previous CFRA to reduce the voting-time in the voting process. Otherwise, when $m < \frac{2t_{\text{lut}}}{t_{\text{dis}}}$, the voting process of the CFRA is used. From Definition 6, the proposed LCFRA associated with the above tuning strategy is called the TLCFRA.

In what follows, the average voting-time improvement ratio of the TLCFRA is analyzed. Let M denote the number of edge pixels in the initial edge set V and let the minimal number of edge pixels be $\frac{2t_{\text{lut}}}{t_{\text{dis}}}$ when applying the tuning strategy. Assume m is a random variable and the distribution of m is uniform for $\frac{2t_{\text{lut}}}{t_{\text{dis}}} \leqslant m \leqslant M$. The average voting-time improvement ratio of TLCFRA over the CFRA can be calculated by

$$\begin{aligned} I_{\text{v}} &= \frac{1}{M}\left(\int_0^{\frac{2t_{\text{lut}}}{t_{\text{dis}}}-1} 0\,\mathrm{d}m + \int_{\frac{2t_{\text{lut}}}{t_{\text{dis}}}}^{M} \frac{t_{\text{dis}}}{t_{\text{dis}}+t_{\text{c}}+t_{\text{a}}} - \frac{2t_{\text{lut}}}{m(t_{\text{dis}}+t_{\text{c}}+t_{\text{a}})}\,\mathrm{d}m\right) \\ &= \frac{t_{\text{dis}}}{t_{\text{dis}}+t_{\text{c}}+t_{\text{a}}} - \frac{2t_{\text{lut}}}{M(t_{\text{dis}}+t_{\text{c}}+t_{\text{a}})}\left(1 + \ln\frac{Mt_{\text{dis}}}{2t_{\text{lut}}}\right). \end{aligned} \tag{1}$$

By Eq. (1), we have the following result.

Lemma 1. *The average voting-time improvement ratio of the TLCFRA over the CFRA is* $I_{\text{v}} = \frac{t_{\text{dis}}}{t_{\text{dis}}+t_{\text{c}}+t_{\text{a}}} - \frac{2t_{\text{lut}}}{M(t_{\text{dis}}+t_{\text{c}}+t_{\text{a}})}\left(1 + \ln\frac{Mt_{\text{dis}}}{2t_{\text{lut}}}\right)$.

Lemma 1 indicates that I_{v} only reflects the average voting-time improvement ratio of the TLCFRA over the CFRA. From Property 1, let the ratio of the average voting-time over the total execution-time be R when plugging our proposed tuning- and LUT-based computational scheme into the three previous CFRAs, such as the RHTL [29,31], the RCD [41], and the RHTE [37]. We have the following result.

Lemma 2. *The average execution-time improvement ratio of TLCFRA over the CFRA is* $I_T = R \times I_{\text{v}}$.

4. Applications to speed up existing randomized algorithms for detecting lines, circles, and ellipses

In this section, the applications of our proposed tuning- and LUT-based computational scheme are presented to speed up the previous CFRAs for detecting lines, circles, and ellipses; the relevant three improved randomized algorithms are called the TLRHTL, the TLRCD, and the TLRHTE, respectively. Moreover, the related nontrivial time complexity analyses are provided to show the computational gains of the three proposed TLCFRAs.

4.1. Speed up the RHTL for detecting lines

For each input edge pixel $v_l = (x_l, y_l)$, by Lemma 1, the time t_{dis} required in the RHTL is first considered. For detecting lines, the RHTL needs a 2-D accumulator array, say ACC, for implementing the (ρ, θ)-space where the value of each entry in ACC is initialized to 0. Each time, two edge pixels are randomly selected from the edge set to determine a possible line $L_{\rho,\theta}$ and the value of $ACC(\bar{\rho}, \bar{\theta})$ is increased by 1 where $\bar{\rho}$ and $\bar{\theta}$ denote the quantized ρ and the quantized θ, respectively. If the value of $ACC(\bar{\rho}, \bar{\theta})$ is equal to the specified threshold,

the possible line $L_{\rho,\theta}$ becomes a candidate line. The distance between the candidate line $L_{\rho,\theta}$ and the input edge pixel $v_l = (x_l, y_l)$ is calculated by

$$d_{l\to(\rho,\theta)} = |\rho - (x_l \cos\theta + y_l \sin\theta)|. \tag{2}$$

If v_l donates one vote to the candidate line $L_{\rho,\theta}$, the value of $d_{l\to(\rho,\theta)}$ must be less than or equal to the bandwidth of the banded LUT-based voting array. In Eq. (2), the values of $\cos\theta$ and $\sin\theta$ can be reused once they have been calculated. From Eq. (2), the calculation of $d_{l\to(\rho,\theta)}$ needs two additions, two multiplications, and one absolute operation. Consequently, we have the following property.

Property 2. *In the RHTL, for each input edge pixel, the time required for calculating the distance $d_{l\to(\rho,\theta)}$ in Eq. (2) is bounded by $t_{\text{dis}} = (2t_{\text{a}} + 2t_{\text{m}} + t_{abs})$ where t_m and t_{abs} denote the time required for one multiplication and one absolute operation, respectively.*

Form Definition 3 in Section 2.2, when a candidate line $L_{\rho,\theta}$ is determined, based on Bresenham's line drawing method [45], the following procedure is used to draw the banded candidate line (BCL) on the LUT-based array. For saving space of the context, the following BCL Drawing procedure only considers $L_{\rho,\theta}$ with the slope between 0 and 1.

Procedure BCL Drawing:

Step 1. Input two end points (x_1, y_1) and (x_2, y_2) of the candidate line $L_{\rho,\theta}$. Let $x_d = | x_1 - x_2 |$ and $y_d = | y_1 - y_2 |$. If $x_1 > x_2$, perform $x = x_2$, $y = y_2$, and $x_e = x_1$; otherwise, perform $x = x_1$, $y = y_1$, and $x_e = x_2$. The time required in this step is $5t_{\text{as}} + 2t_{\text{a}} + 2t_{\text{abs}} + t_{\text{c}}$ where the symbol t_{as} denotes the execution-time required for one assignment operation.

Step 2. Perform $A_{\text{lut}}(x, y) = 1$, $A_{\text{lut}}(x, y-1) = 1$, and $A_{\text{lut}}(x, y+1) = 1$ to draw the first three line points of $L_{\rho,\theta}$. The time required in this step is $3t_{\text{as}} + 2t_{\text{a}}$.

Step 3. Initialize the decision parameter as $p = 2y_d - x_d$ where p is used to determine where the next line point should be drawn. The time required in this step is $t_{\text{as}} + t_{\text{a}} + t_{\text{m}}$.

Step 4. Let $x = x + 1$. If $p < 0$, perform $p = p + 2y_d$; otherwise, perform $y = y + 1$ and $p = p + 2(y_d - x_d)$. Then we set $A_{\text{lut}}(x, y) = 1$, $A_{\text{lut}}(x, y-1) = 1$, and $A_{\text{lut}}(x, y+1) = 1$ to draw the line points. This step requires at most $6t_{\text{as}} + 6t_{\text{a}} + t_{\text{m}} + t_{\text{c}}$ time.

Step 5. If $x \geqslant x_e$, then stop; otherwise, go to Step 3. This step only needs t_{c} time.

It is known that A_{lut} is of size $W \times H$ and let $l_{\text{ave}} = \frac{\max(W,H)}{2}$ denote the average number of the line points for $L_{\rho,\theta}$, then the time required for performing Bresenham's BCL Drawing procedure is shown below.

Lemma 3. *Performing the BCL Drawing procedure on A_{lut} needs $t_{\text{lut}} = (9 + 6l_{\text{ave}})t_{\text{as}} + (5 + 6l_{\text{ave}})t_{\text{a}} + (1 + l_{\text{ave}})t_{\text{m}} + (1 + 2l_{\text{ave}})t_{\text{c}} + 2t_{abs}$ time.*

Proof. In the BCL Drawing procedure, since Step 1, Step 2, and Step 3 are only performed once, the time required in the three steps is $9t_{\text{as}} + 5t_{\text{a}} + t_{\text{m}} + 2t_{\text{abs}} + t_{\text{c}}$. Because Step 4 and Step 5 must be performed l_{ave} times until the condition $x < x_e$ in Step 5 is violated, the time required in the two steps is $(6t_{\text{as}} + 6t_{\text{a}} + t_{\text{m}} + 2t_{\text{c}})l_{\text{ave}}$. Summing up all the time requirement, the total time required in the BCL Drawing procedure is $t_{\text{lut}} = (9 + 6l_{\text{ave}})t_{\text{as}} + (5 + 6l_{\text{ave}})t_{\text{a}} + (1 + l_{\text{ave}})t_{\text{m}} + (1 + 2l_{\text{ave}})t_{\text{c}} + 2t_{\text{abs}}$. □

In our experiments, $t_{\text{as}}, t_{\text{a}}, t_{\text{m}}, t_{\text{abs}}$, and t_{c} are 1.2, 2.2, 2.2, 6, and 3.3 ns (nano second), respectively, by using the Pentium 4 computer with 1.8 GHz. Thus, from Property 2, we have $t_{\text{dis}} = 2t_{\text{a}} + 2t_{\text{m}} + t_{\text{abs}} = 14.8$ ns and from Lemma 3, we have $t_{\text{lut}} = 39.3 + 29.2l_{\text{ave}}$ ns.

Following Definition 6 in Section II.C, when the condition $m \geqslant \frac{2t_{\text{lut}}}{t_{\text{dis}}}$ holds, the computational advantage of our proposed TLRHTL can be guaranteed. From $t_{\text{dis}} = 14.8$ ns and $t_{\text{lut}} = 39.3 + 29.2l_{\text{ave}}$ ns, the tuning strategy in our proposed TLRHTL is defined as follows.

Definition 7 (*Tuning strategy for line detection*). In our proposed TLRHTL, if $m \geqslant 5.311 + 3.946l_{\text{ave}}$ holds, our proposed LUT-based voting platform is used; otherwise, the voting process of the RHTL is used.

By using the above tuning strategy, the average voting-time improvement ratio of our proposed TLRHTL over the previous RHTL is shown below.

Theorem 1. *The average voting-time improvement ratio of our proposed TLRHTL over the previous RHTL is* $I_v = 0.729 - \frac{3.872+2.877l_{ave}}{M}\left(1+\ln\frac{7.4M}{39.3+29.2l_{ave}}\right)$.

Proof. From Lemma 1 and Definition 7, the average voting-time improvement ratio of our proposed TLRHTL over the previous RHTL is $I_v = \frac{t_{dis}}{t_{dis}+t_c+t_a} - \frac{2t_{lut}}{M(t_{dis}+t_c+t_a)}\left(1+\ln\frac{Mt_{dis}}{2t_{lut}}\right) = 0.729 - \frac{3.872+2.877l_{ave}}{M}\left(1+\ln\frac{7.4M}{39.3+29.2l_{ave}}\right)$. □

By Property 1, Lemma 2, and Theorem 1, we have the following result.

Corollary 1. *The average theoretical execution-time improvement ratio of our proposed TLRHTL over the previous RHTL is about* $I_T^L = I_v \times R = I_v \times 0.65 = 0.4739 - \frac{2.517+1.87l_{ave}}{M}\left(1+\ln\frac{7.4M}{39.3+29.2l_{ave}}\right)$.

Suppose a 256×256 image is used in the line detection experiment. From the definition of l_{ave}, we have $l_{ave} = 128$. Putting $l_{ave} = 128$ into Definition 7, it yields the tuning condition $m \geqslant 511$.

Corollary 2. *Assume* $M = 3000$, *the theoretical average time improvement ratio of our proposed TLRHTL over the previous RHTL is* 0.251.

4.2. Speed up the RCD for detecting circles

In the RCD, it first randomly selects four edge pixels each time to determine a candidate circle $C_{x_c,y_c,r}$ in terms of the center (x_c, y_c) and the radius r. When $C_{x_c,y_c,r}$ is determined, for each input edge pixel v_l, the time required for computing the distance between $C_{x_c,y_c,r}$ and v_l is calculated by

$$d_{l\to(x_c,y_c,r)} = \left|\sqrt{(x_l-x_c)^2+(y_l-y_c)^2} - r\right|. \tag{3}$$

If $d_{l\to(x_c y_c r)} \leqslant \triangle$, say $\triangle = 1$, v_l is said to lie on $C_{x_c,y_c,r}$. Consequently, we have the following property.

Property 3. *In the RCD, the time required for calculating the distance* $d_{l\to(x_c y_c r)}$ *is bounded by* $t_{dis} = 4t_a + 2t_s + t_r + t_{abs}$ *where* t_s *and* t_r *denote the time required to perform one square operation and one square-root operation, respectively.*

In our proposed TLRCD, once a candidate circle $C_{x_c,y_c,r}$ is determined, based on Bresenham's circle drawing method [46], the following procedure is used to draw the banded candidate circle (BCC) on the LUT-based array.

Procedure BCC Drawing:

Step 1. Input the center (x_c, y_c) and radius r of $C_{x_c,y_c,r}$. Set $x = 0$ and $y = r$. The position $(0, r)$ denotes the first circle point on the circumference of $C_{0,0,r}$ where $C_{0,0,r}$ denotes the circle obtained by shifting the center of $C_{x_c,y_c,r}$ to coordinate $(0, 0)$. The time required in this step is $2t_{as}$.

Step 2. Let $x_a = x + x_c$ and $y_a = y + y_c$, then perform $A_{lut}(x_a, y_a) = 1$, $A_{lut}(x_a, y_a + 1) = 1$, and $A_{lut}(x_a, y_a - 1) = 1$. Due to the symmetric property for the eight octants, the three symmetric circle points can be obtained for each octant. The time required in this step is $40t_{as} + 32t_a$.

Step 3. Initialize the decision parameter as $p = 1 - r$ where p is used to determine where the next circle point should be drawn. The time required in this step is $t_{as} + t_a$.

Step 4. If $p < 0$, the next circle point along $C_{0,0,r}$ is drawn at the position $(x + 1, y)$. In addition, perform $x = x + 1$ and $p = p + 2x + 1$; otherwise, the next circle point along $C_{0,0,r}$ is $(x + 1, y + 1)$. Further, perform $x = x + 1$, $y = y + 1$, and $p = p + 2x + 1$. This step requires at most $3t_{as} + 4t_a + t_m + t_c$ time.

Step 5. Let $x_a = x + x_c$ and $y_a = y + y_c$, then we set $A_{lut}(x_a, y_a) = 1$, $A_{lut}(x_a, y_a + 1) = 1$, and $A_{lut}(x_a, y_a - 1) = 1$. Due to the symmetric property for the eight octants, the time required in this step is $40t_{as} + 32t_a$.

Step 6. If $x \geqslant y$, then stop; otherwise, go to Step 4. This step only needs t_c time.

Let $r_{ave} = \frac{\min(W,H)}{4}$ denote the average radius of the candidate circle, then the time required for performing the BCC Drawing procedure is shown below.

Lemma 4. *Performing the BCC Drawing procedure on A_{lut} needs* $t_{lut} = \left(43 + 43\frac{r_{ave}}{\sqrt{2}}\right)t_{as} + \left(33 + 36\frac{r_{ave}}{\sqrt{2}}\right)t_a + \frac{r_{ave}}{\sqrt{2}}(t_m + 2t_c)$ *time.*

Proof. In the BCC Drawing procedure, since Step 1, Step 2, and Step 3 are only performed once, the time required in the three steps is $43t_{as} + 33t_a$. Because we apply the symmetric property of the eight octants to the BCC Drawing procedure, Step 4, Step 5, and Step 6 are performed $\frac{r_{ave}}{\sqrt{2}}$ times until the condition $x < y$ in Step 6 is violated. After running Step 4, Step 5, and Step 6 $\frac{r_{ave}}{\sqrt{2}}$ times, it takes $43\frac{r_{ave}}{\sqrt{2}}t_{as} + 36\frac{r_{ave}}{\sqrt{2}}t_a + \frac{r_{ave}}{\sqrt{2}}(t_m + 2t_c)$ time. Summing up all the time requirement, the total time required in the BCC Drawing procedure is $t_{lut} = \left(43 + 43\frac{r_{ave}}{\sqrt{2}}\right)t_{as} + \left(33 + 36\frac{r_{ave}}{\sqrt{2}}\right)t_a + \frac{r_{ave}}{\sqrt{2}}(t_m + 2t_c)$. □

In our experiments, it is shown that t_s and t_r are 2.2 and 28 ns, respectively. Thus, from Property 3, t_{dis} and t_{lut} can be calculated by $t_{dis} = 4t_a + 2t_s + t_r + t_{abs} = 47.2$ ns and from Lemma 4, we have $t_{lut} = 124.2 + 139.6\frac{r_{ave}}{\sqrt{2}}$ ns.

Following Definition 6 in Section 3, when the condition $m \geqslant \frac{2t_{lut}}{t_{dis}}$ holds, the computational advantage of our proposed TLRCD can be guaranteed. From $t_{dis} = 47.2$ ns and $t_{lut} = 124.2 + 139.6\frac{r_{ave}}{\sqrt{2}}$ ns, the tuning strategy in our proposed TLRCD is described below.

Definition 8 (*Tuning strategy for circle detection*). In our proposed TLRCD, if $m \geqslant 5.263 + 5.915\frac{r_{ave}}{\sqrt{2}}$ holds, our proposed LUT-based voting platform is used; otherwise, the voting process of the RCD is used.

Using the above tuning strategy, the average voting-time improvement ratio of our proposed TLRCD over the previous RCD is shown below.

Theorem 2. *The average voting-time improvement ratio of our proposed TRCD over the previous RCD is* $I_v = 0.896 - \frac{4.714 + 5.298\frac{r_{ave}}{\sqrt{2}}}{M}\left(1 + \ln\frac{23.6M}{124.2 + 139.6\frac{r_{ave}}{\sqrt{2}}}\right)$.

Proof. From Lemma 1 and Definition 8, the average voting-time improvement ratio of our proposed TLRCD over the previous RCD is $I_v = \frac{t_{dis}}{t_{dis}+t_c+t_a} - \frac{2t_{lut}}{M(t_{dis}+t_c+t_a)}\left(1 + \ln\frac{Mt_{dis}}{2t_{lut}}\right) = 0.896 - \frac{4.714 + 5.298\frac{r_{ave}}{\sqrt{2}}}{M}\left(1 + \ln\frac{23.6M}{124.2 + 139.6\frac{r_{ave}}{\sqrt{2}}}\right)$. □

By Property 1, Lemma 2, and Theorem 2, we have the following result.

Corollary 3. *The average theoretical execution-time improvement ratio of our proposed TLRCD over the previous RCD is about* $I_T^C = 0.815 - \frac{4.29 + 4.821\frac{r_{ave}}{\sqrt{2}}}{M}\left(1 + \ln\frac{23.6M}{124.2 + 139.6\frac{r_{ave}}{\sqrt{2}}}\right)$.

Given a 256×256 image, from the definition of r_{ave}, we have $r_{ave} = 64$. Putting $r_{ave} = 64$ into Definition 8, it yields the tuning condition $m \geqslant 273$. Let $M = 3000$, we have the result.

Corollary 4. *The average time improvement ratio of our proposed TLRCD over the previous RCD is* 0.563.

4.3. Speed up the RHTE for detecting ellipses

In the RHTE [37] by McLaughlin, it selects three edge pixels each time to determine the candidate ellipse. From the elliptic equation, an ellipse can be expressed by

$$f(y - y_c)^2 + e(x_l - x_c)(y - y_c) + d(x_l - x_c)^2 = 1, \tag{4}$$

where (x_c, y_c) is the center of the ellipse; d, e, and f are the other three parameters satisfying $d > 0$, $f > 0$, and $4df - e^2 > 0$. Then the 5-tuple parameters (x_c, y_c, d, e, f) can be converted to 5-tuple parameters (x_c, y_c, a, b, θ) for $\theta = \frac{\arctan(\frac{e}{d-f})}{2}$, $a = \sqrt{\frac{1}{d\cos^2\theta + e\sin\theta\cos\theta + f\sin^2\theta}}$, and $b = \sqrt{\frac{1}{f\cos^2\theta - e\sin\theta\cos\theta + d\sin^2\theta}}$ [49], where θ denotes the orientation angle of the ellipse and the two parameters a and b denote the half lengths of the two axes.

We can use a simple measure to calculate the distance between the edge pixel and the candidate ellipse $E_{x_c,y_c,a,b,\theta}$ to determine whether the candidate ellipse is a true ellipse or not. When $a \geqslant b\,(b > a)$, we consider two cases: the first case is that the possible ellipse orientation, i.e. $\theta\,(\theta + \pi/2)$, satisfies $0 \leqslant \theta \leqslant \pi/4\,(\pi/2 \leqslant \theta \leqslant 3\pi/4)$ and the second case is that $\pi/4 < \theta \leqslant \pi/2\,(3\pi/4 \leqslant \theta \leqslant \pi)$ is hold. For the first case, we use the vertical distance between the edge pixel and the candidate ellipse to determine whether the candidate ellipse is a true ellipse. Given an edge pixel $v_l = (x_l, y_l)$, if the vertical line $X = x_l$ passes through $E_{x_c,y_c,a,b,\theta}$, then Eq. (4) has two solutions (x_l, y_1) and (x_l, y_2). Thus, the vertical distance between $v_l = (x_l, y_l)$ and the ellipse in Eq. (4) is equal to the minimum of $|\,y_l - y_1\,|$ and $|\,y_l - y_2\,|$. Therefore, the distance between $E_{x_c,y_c,a,b,\theta}$ and v_l is calculated by

$$d_{l\to(x_c,y_c,e,d,f)} = \min\left\{\left|y_l - \frac{-e(x_l - x_c) \pm \sqrt{[e(x_l - x_c)]^2 - 4f[d(x_l - x_c)^2 - 1]}}{2f} - y_c\right|\right\}.$$

If $d_{l\to(x_c,y_c,e,d,f)} \leqslant \triangle$, v_l is said to lie on $E_{x_c,y_c,e,d,f}$.

By the same arguments in the first case, for the second case, we use the horizontal distance between the edge pixel and the candidate ellipse to determine whether the candidate ellipse is a true ellipse. For saving the space of context, we omit the similar discussion.

If $[e(x - x_c)]^2 - 4f[d(x_l - x_c)^2 - 1] < 0$, the remaining operations for computing a are unnecessary to be calculated. For different v_l, the values of $2f, 4f$, and $-e$ can be reused in Eq. (5) once they are calculated in advance. From Eq. (5), the calculation of $d_{l\to(x_c,y_c,e,d,f)}$ needs nine additions, four multiplications, two square operations, one division, one square-root operation, one absolute operation, and one comparison. We have the following property.

Property 4. *In the RHTE, for each edge pixel, the time required for calculating the distance $d_{l\to(x_c,y_c,e,d,f)}$ in Eq. (5) is bounded by $t_{dis} = 9t_a + 4t_m + 2t_s + t_d + t_r + t_c + t_{abs}$ where t_d denotes the time required to perform one division.*

We now consider time t_{lut} required in the proposed TLRHTE. When a candidate Ellipse $E_{x_c,y_c,a,b,\theta}$ has been determined, based on Bresenham's ellipse drawing method [47], the following procedure is used to draw the banded candidate ellipse (BCE) on the LUT-based array.

Procedure BCE Drawing:

Step 1. Input the center (x_c, y_c), two axes a and b, and the orientation angle θ of the candidate ellipse $E_{x_c,y_c,a,b,\theta}$. Set $x = 0$, $y = b$, $a_1 = a^2$, $b_1 = b^2$, $a_2 = 2a_1$, and $b_2 = 2b_1$. The position $(0, b)$ denotes the first drawn point on A_{lut} for the ellipse $E_{0,0,a,b,0}$ where $E_{0,0,a,b,0}$ denotes the ellipse obtained by shifting the center of $E_{x_c,y_c,a,b,\theta}$ to coordinate $(0,0)$ and rotating the orientation angle of $E_{x_c,y_c,a,b,\theta}$ to 0. The time required in this step is $6t_{as} + 2t_m + 2t_s$.

Step 2. Let $x_a = y\sin\theta + x_c$ and $y_a = -y\cos\theta + y_c$, then perform $A_{lut}(x_a, y_a) = 1$, $A_{lut}(x_a, y_a + 1) = 1$, $A(x_a, y_a - 1) = 1$, $A_{lut}(x_a + 1, y_a) = 1$, $A_{lut}(x_a - 1, y_a) = 1$, $A_{lut}(x_a + 1, y_a - 1) = 1$, $A_{lut}(x_a - 1, y_a + 1) = 1$, $A_{lut}(x_a + 1, y_a + 1) = 1$, and $A_{lut}(x_a - 1, y_a - 1) = 1$. Due to the symmetric property for the four quadrants, the symmetric nine ellipse points can be obtained for each quadrant. The time required in this step is $44t_{as} + 56t_a + 12t_m$.

Step 3. Initialize the decision parameter as $p = b_1 - a_1 b + 0.25a_1$ where the parameter p is used to determine where the next ellipse point should be drawn. Let $p_x = 0$ and $p_y = a_2 y$. The time required in this step is $3t_{as} + 2t_a + 3t_m$.

Step 4. If $p < 0$, the next ellipse point along the ellipse $E_{0,0,a,b,0}$ is drawn at the position $(x + 1, y)$. In addition, perform $x = x + 1$, $p_x = p_x + b_2$, and $p = p + b_1 + p_x$; otherwise, the next point along the ellipse $E_{0,0,a,b,0}$ is $(x + 1, y - 1)$. Further, perform $x = x + 1$, $y = y - 1$, $p_x = p_x + b_2$, $p_y = p_y - a_2$ and $p = p + b_1 + p_x - p_y$. This step requires at most $5t_{as} + 7t_a + t_c$ time.

Step 5. Let $x_a = x\cos\theta + y\sin\theta + x_c$ and $y_a = x\sin\theta + y\cos\theta + y_c$, then perform $A_{lut}(x_a, y_a) = 1$, $A_{lut}(x_a, y_a + 1) = 1$, $A_{lut}(x_a, y_a - 1) = 1$, $A_{lut}(x_a + 1, y_a) = 1$, $A_{lut}(x_a - 1, y_a) = 1$, $A_{lut}(x_a + 1, y_a - 1) = 1$, $A_{lut}(x_a - 1, y_a + 1) = 1$, $A_{lut}(x_a + 1, y_a + 1) = 1$, and $A_{lut}(x_a - 1, y_a - 1) = 1$. Due to the symmetric property for the four quadrants, the symmetric nine points can be obtained for each quadrant. The time required in this step is $44t_{as} + 56t_a + 12t_m$.

Step 6. If $p_x < p_y$, then go to Step 4; otherwise, go to Step 7. This step only needs t_c time.

Step 7. Initialize the decision parameter as $p = b_1(x+0.5)^2 + a_1(y-1)^2 - a_1b_1$. The time required in this step is $t_{as} + 4t_a + 3t_m + 2t_s$.

Step 8. If $p > 0$, the next ellipse point along the ellipse $E_{0,0,a,b,0}$ is drawn at position $(x, y-1)$. In addition, perform $y = y-1$, $p_y = p_y - a_2$, and $p = p + a_1 - p_y$; otherwise, the next point along $E_{0,0,a,b,0}$ is $(x+1, y-1)$. Further, perform $x = x+1$, $y = y-1$, $p_x = p_x + b_2$, $p_y = p_y - a_2$, and $p = p + a_1 - p_y + p_x$. This step requires at most $5t_{as} + 7t_a + t_c$ time.

Step 9. Let $x_a = x\cos\theta + y\sin\theta + x_c$ and $y_a = x\sin\theta + y\cos\theta + y_c$, then perform $A_{lut}(x_a, y_a) = 1$, $A_{lut}(x_a, y_a+1) = 1$, $A_{lut}(x_a, y_a-1) = 1$, $A_{lut}(x_a+1, y_a) = 1$, $A_{lut}(x_a-1, y_a) = 1$, $A_{lut}(x_a+1, y_a-1) = 1$, $A_{lut}(x_a-1, y_a+1) = 1$, $A_{lut}(x_a+1, y_a+1) = 1$, and $A_{lut}(x_a-1, y_a-1) = 1$. Due to the symmetric property for the four quadrants, the symmetric nine points can be obtained for each quadrant easily. The time required in this step is $44t_{as} + 56t_a + 12t_m$.

Step 10. If $y \leqslant 0$, then stop; otherwise, go to Step 8. This step only needs t_c time.

Let $a_{ave} = \frac{\min(W,H)}{4}$ denote the average half length of major axis of the candidate ellipse, then the time required for performing the BCE Drawing procedure is shown below.

Lemma 5. *Performing the BCE Drawing procedure on A_{lut} needs $t_{lut} = (54 + 49a_{ave})t_{as} + (62 + 63a_{ave})t_a + (20 + 12a_{ave})t_m + 2a_{ave}t_c + 4t_s$ time.*

Proof. In the BCE Drawing procedure, since Steps 1, 2, 3, and 7 are only performed once, the time required in the three steps is $54t_{as} + 62t_a + 20t_m + 4t_s$. Further, the two subsets of the remaining six steps, one consisting of Step 4, Step 5, and Step 6 and the other consisting of Step 8, Step 9, and Step 10, are perform a_{ave} times. The time required for performing each of these two subsets is $(49t_{as} + 63t_a + 12t_m + 2t_c)$. After running the two subsets, it takes $(49t_{as} + 63t_a + 12t_m + 2t_c)a_{ave}$ time. Summing up all the time requirement, the total time required in the BCE Drawing procedure on the LUT-based array A_{lut} is $t_{lut} = (54 + 49a_{ave})t_{as} + (62 + 63a_{ave})t_a + (20 + 12a_{ave})t_m + 2a_{ave}t_c + 4t_s$. □

From Property 4, the time required for the distance calculation in the RHTE is bounded by $t_{dis} = 9t_a + 4t_m + 2t_s + t_d + t_r + t_c + t_{abs}$ time. From Lemma 5, it is known that it takes $t_{lut} = (54 + 49a_{ave})t_{as} + (62 + 63a_{ave})t_a + (20 + 12a_{ave})t_m + 2a_{ave}t_c + 4t_s$ time to create the LUT-based voting platform for detection ellipses. In our experiment, t_d is 23.8 ns by using the Pentium 4 computer with 1.8 GHz. From Property 4, we have $t_{dis} = 9t_a + 4t_m + 2t_s + t_d + t_r + t_c + t_{abs} = 94.1$ ns and from Lemma 5, we have $t_{lut} = 254 + 230.4a_{ave}$ ns.

Following Definition 6, the tuning strategy in our proposed TLRHTE is shown below.

Definition 9 (*Tuning strategy for ellipse detection*). In our proposed TLRHTE, if $m \geqslant 5.399 + 4.897a_{ave}$ holds, our proposed LUT-based voting platform is used; otherwise, the voting process of the RHTE is used.

By using the above tuning strategy, we have the following result.

Theorem 3. *The average voting-time improvement ratio of our proposed TLRHTE over the previous RHTE is* $I_v = 0.945 - \frac{5.1+4.627a_{ave}}{M}\left(1 + \ln\frac{47.05M}{254+230.4a_{ave}}\right)$.

Proof. From Lemma 1 and Definition 9, the average voting-time improvement ratio of our proposed TLRHTE over the previous RHTE is $I_v = \frac{t_{dis}}{t_{dis}+t_c+t_a} - \frac{2t_{lut}}{M(t_{dis}+t_c+t_a)}(1 + \ln\frac{Mt_{dis}}{2t_{lut}}) = 0.945 - \frac{5.1+4.627a_{ave}}{M}(1 + \ln\frac{47.05M}{254+230.4a_{ave}})$. □

By Property 1, Lemma 2, and Theorem 3, we have the following two results.

Corollary 5. *The average theoretical execution-time improvement ratio of our proposed TLRHTE over the previous RHTE is about* $I_T^E = 0.685 - \frac{3.698+3.355a_{ave}}{M}\left(1 + \ln\frac{47.05M}{254+230.4a_{ave}}\right)$.

Corollary 6. *Putting $a_{ave} = 64$ into Definition 9, it yields the tuning condition $m \geqslant 319$. Let $M = 3000$, the average time improvement ratio of our proposed TLRHTE over the RHTE is* 0.449.

5. Experimental results

In this section, some experimental results are demonstrated to evaluate the performance of the concerning algorithms. Since our proposed TLRHTL, TLRCD, and TLRHTE inherit the robustness of the previous RHTL, RCD, and RHTE, respectively, we only evaluate the execution-time performance among them. In addition, we also provide the comparison between our proposed TLCFRAs and the previous PAV-based algorithms [44], such as the PRHTL, the PRCD, and the PRHTE, to emphasize the computation advantage of our proposed LUT-based voting platform. All the experiments are implemented by using the Pentium 4 personal computer with 1.8 GHz and the C programming language.

5.1. Experiments for line detection

Four 256×256 images (see Fig. 3) are taken to evaluate the execution-time performance between the RHTL and our proposed TLRHTL. In our experiments, the Sobel edge detection is used to obtain the four edge maps of Fig. 3. For the four testing images, after running our proposed TLRHTL, the resulting detected lines are the same as the RHTL and are shown in Fig. 4. The execution-time comparison and execution-time improvement ratios for the four testing images are shown in Table 1. The execution-time is measured in terms of milliseconds. From Table 1, experimental results reveal that our proposed TLRHTL is faster than the previous RHTL for the four testing images and the average execution-time improvement ratio is 28.1%. From

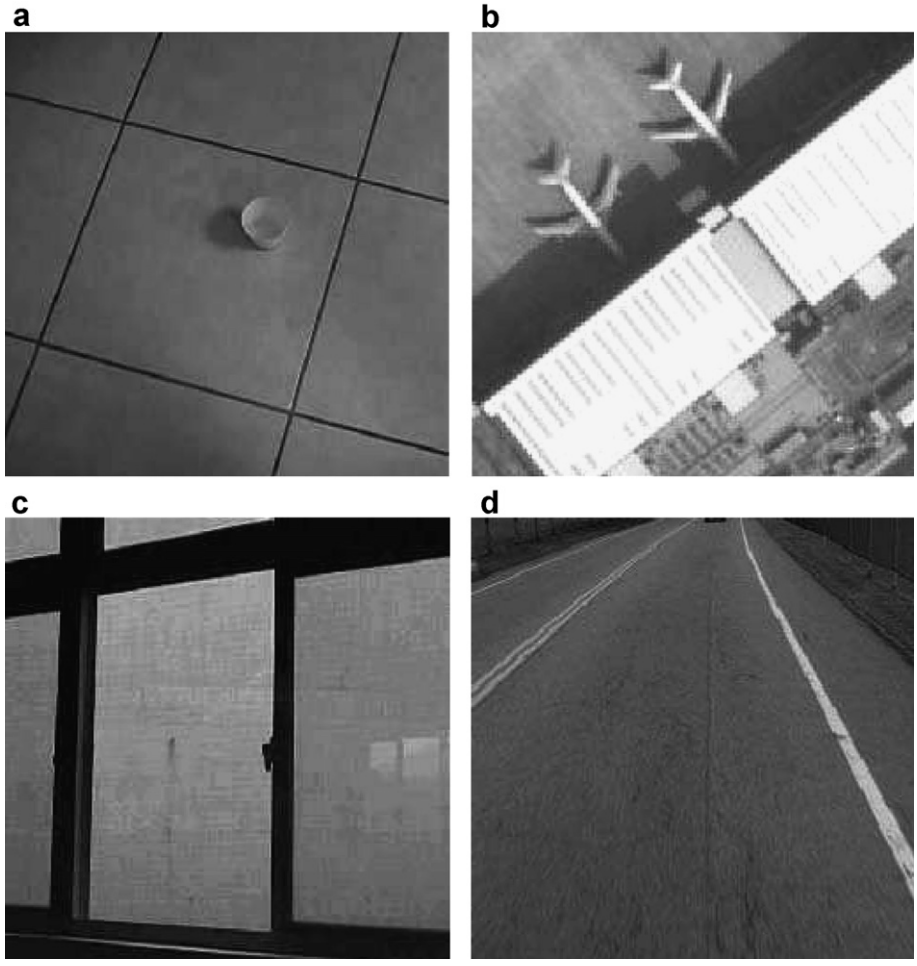

Fig. 3. The four testing images: (a) the floor image, (b) the airport image, (c) the window image and (d) the road image.

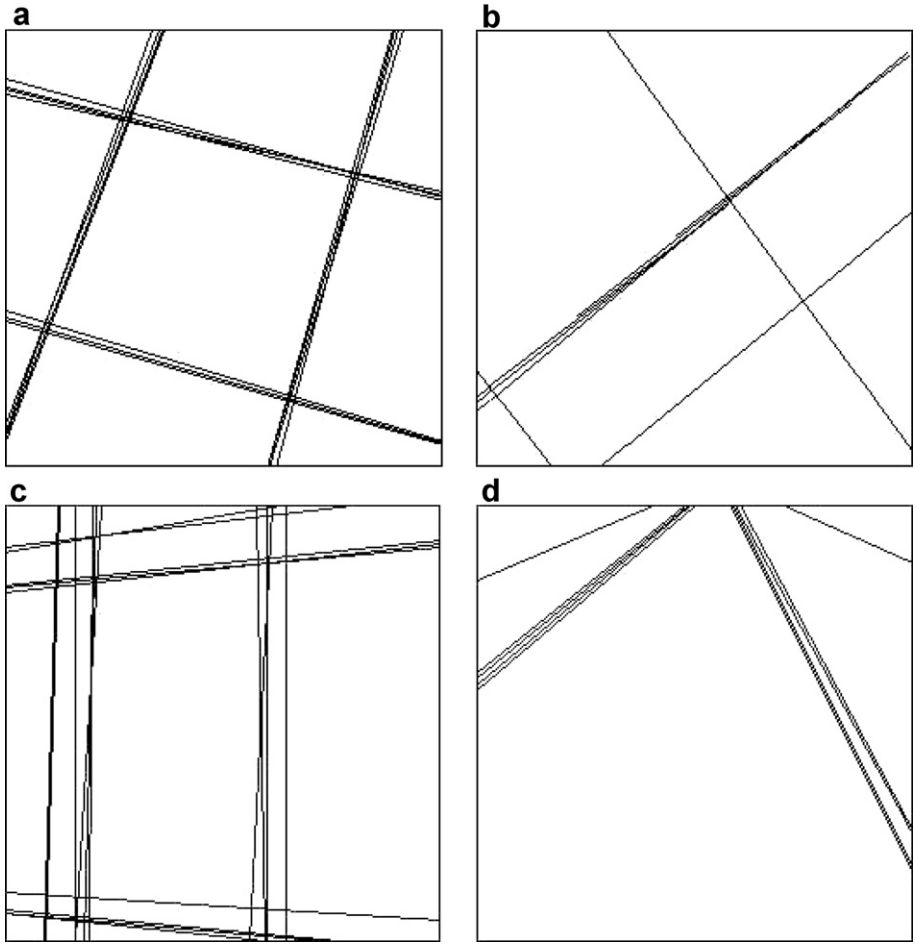

Fig. 4. The detected lines of Fig. 3: (a) the floor image, (b) the airport image, (c) the window image and (d) the road image.

Table 1
Execution-time comparison for the RHTL, the PRHTL and the TLRHTL

	Image			
	Floor	Airport	Window	Road
RHTL	32	105	54	52
PRHTL	28	95	46	46
TLRHTL	24	75	39	36
$\frac{TLRHTL-RHTL}{RHTL}$	0.250	0.286	0.278	0.308
$\frac{TLRHTL-PRHTL}{PRHTL}$	0.143	0.211	0.152	0.217

Corollary 2, it is known that the theoretical execution-time improvement ratio is 25.1% which nearly meets the practical execution-time improvement ratio 28.1%. Table 1 also shows that our proposed TLRHTL has better execution-time performance when compared to the previous PRHTL [44] and the average execution-time improvement ratio is 18.1%.

5.2. Experiments for circle detection

Three 256×256 images and one 352×240 image (see Fig. 5) are used to evaluate the time performance between the RCD and our proposed TLRCD. For the four testing images, after running our proposed

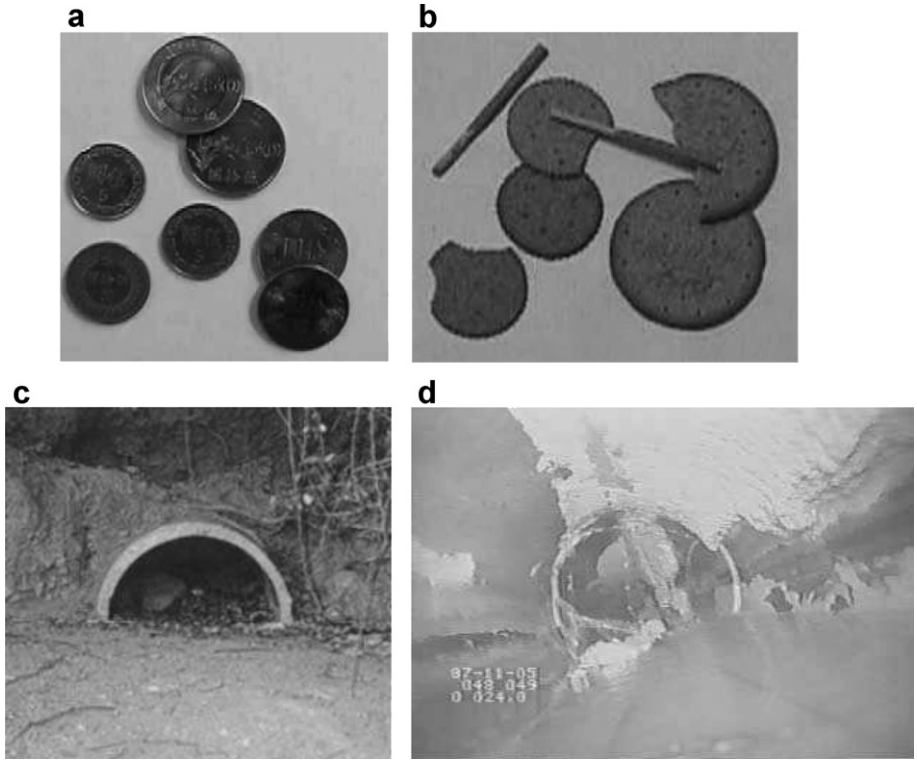

Fig. 5. The four testing images: (a) the coin image, (b) the cracker image, (c) the culvert image and (d) the sewage pipe image.

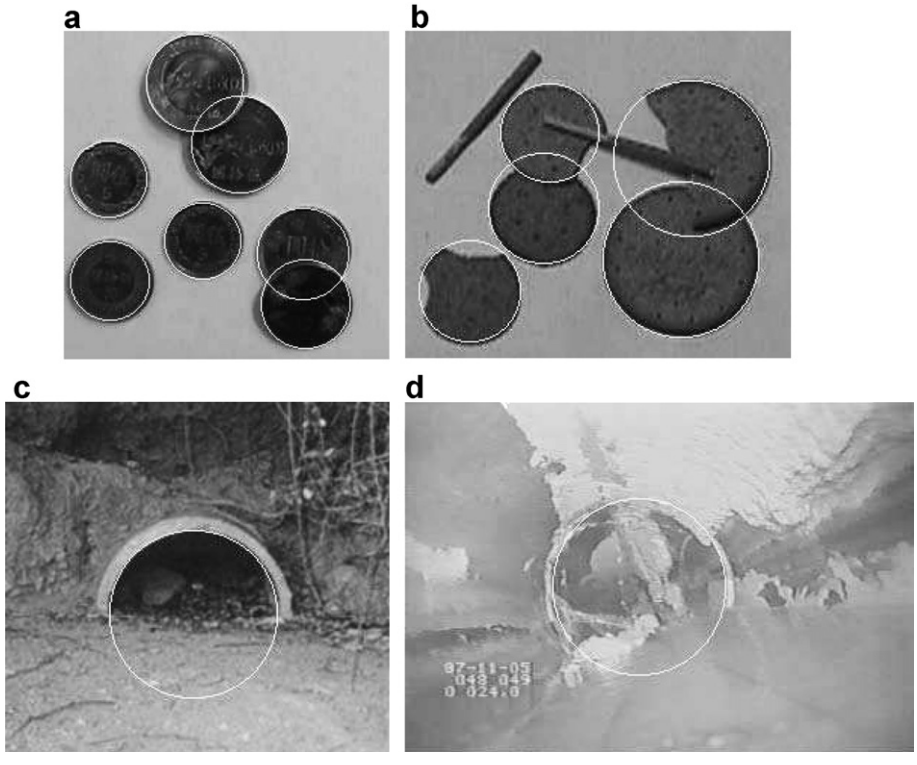

Fig. 6. The detected circles of Fig. 5: (a) the coin image, (b) the cracker image and (c) the culvert image and (d) the sewage pipe image.

TLRCD, the detected circles are the same as the RCD and are illustrated in Fig. 6. The execution-time comparison and execution-time improvement ratios for the four testing images are shown in Table 2. The execution-time is still measured in terms of milliseconds. From Table 2, experimental results reveal that our proposed TLRCD is faster than the previous RCD for the four testing images and the average execution-time improvement ratio is 56.5%. From the Corollary 4, it is known that the theoretical execution-time improve-

Table 2
Execution-time comparison for the RCD, the PRCD, and the TLRCD

	Image			
	Coin	Cracker	Culvert	Sewage pipe
RCD	107	37	117	484
PPRCD	81	29	88	382
TLRCD	51	16	52	187
$\frac{RCD-TLRCD}{RCD}$	0.523	0.568	0.556	0.614
$\frac{PRRCD-TLRCD}{PRCD}$	0.370	0.448	0.409	0.510

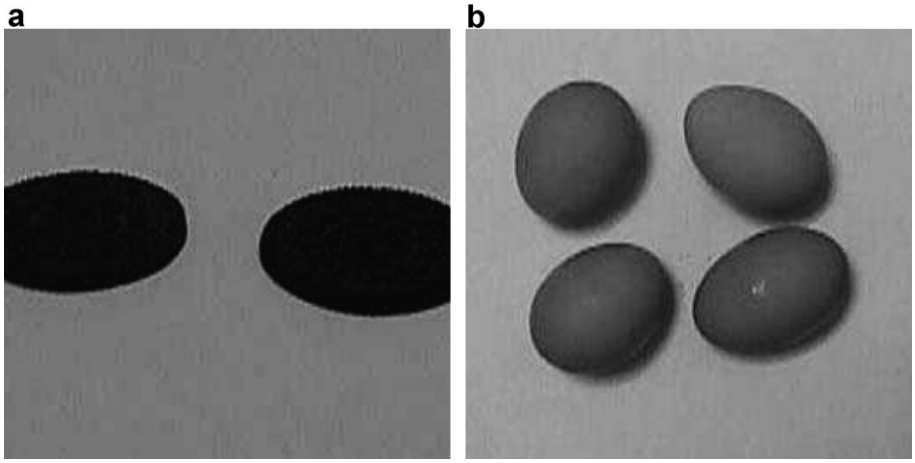

Fig. 7. The two testing images: (a) the cookies image and (b) the eggs image.

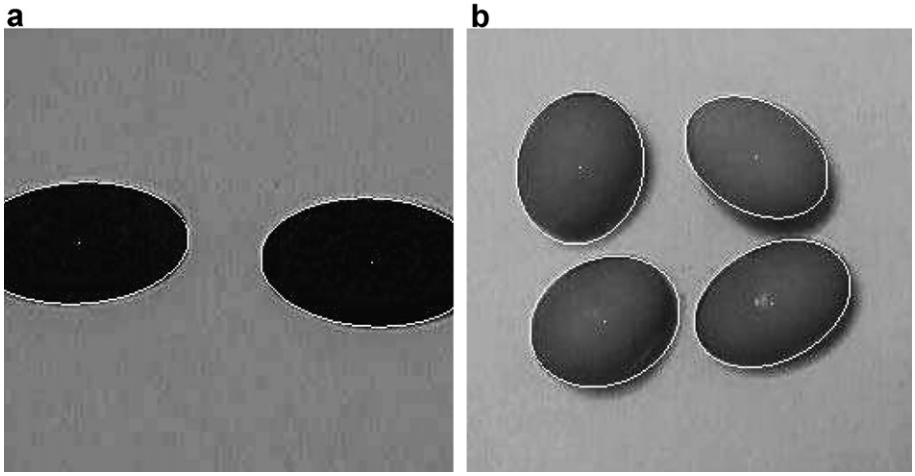

Fig. 8. The detected ellipses of Fig. 7: (a) the cookies image and (b) the eggs image.

Table 3
Execution-time comparison for the RHTE, the PRHTE, and the TLRHTE

	Image	
	Cookies	Eggs
RHTE	53	221
PRHTE	41	170
TLRHTE	27	118
$\frac{TLRHTE-RHTE}{RHTE}$	0.491	0.466
$\frac{TLRHTE-PRHTE}{PRHTE}$	0.341	0.306

ment ratio is 56.3% which nearly meets the practical execution-time improvement ratio 56.5%. From Table 2, the execution-time performance of our proposed TLRCD is better than that of the PRCD [44] and the average execution-time improvement ratio is 43.4%.

5.3. Experiments for ellipse detection

Two 256×256 images (see Fig. 7) are used to evaluate the time performance between the RHTE and our proposed TLRHTE. For the two testing images, after running our proposed TLRHTE, the detected ellipses are the same as the RHTE and are shown in Fig. 8. The execution-time improvement ratios for the two testing images are shown in Table 3. From Table 3, experimental results reveal that our proposed TLRHTE is faster than the previous RHTE for the two testing images and the average execution-time improvement ratio is 47.9%. From Corollary 6, it is known that the theoretical execution-time improvement ratio is 44.9% which is close to the practical execution-time improvement ratio 47.9%. Table 3 also shows that our proposed TLRHTE has 32.4% average execution-time improvement ratio when compared to the PRHTE [44].

6. Conclusion

We have presented the proposed LUT-based voting platform and tuning strategy. Plugging our two newly proposed techniques into the previous three CFRAs, such as the RHTL, the RCD, and the RHTE, for detecting lines, circles, and ellipses, respectively. Our proposed three improved randomized algorithms, namely the TLRHTL, the TLRCD, and the TLRHTE, have also be presented. Moreover, we have provided the detailed time complexity analysis for each proposed improved algorithm. Under the same robustness, experimental results show that the execution-time improvement ratios of our proposed three TLCFRAs are about 28%, 56%, and 48% for detecting lines, circles, and ellipses, respectively, and these improvement ratios are vary close to the theoretic analyses. Experimental results also show that our proposed three TLCFRAs have higher execution-time improvement ratios when compared to the PAV-based algorithms [44].

Acknowledgement

The authors would like to appreciate Dr. T.C. Chen for his valuable comments in Appendix 1.

Appendix 1. Proof for the estimated perimeter of ellipse

The parametric representation of the ellipse can be expressed by $x = a\sin t$ and $y = b\cos t$. We consider the first case when $a \geqslant b$. By differentiating the above two parametric equations, we have $\mathrm{d}x = a\cos t\,\mathrm{d}t$ and $\mathrm{d}y = -b\sin t\,\mathrm{d}t$. Let $\mathrm{d}s^2 = \mathrm{d}x^2 + \mathrm{d}y^2$, then it yields

$$\mathrm{d}s^2 = (a^2\cos^2 t + b^2\sin^2 t)\,\mathrm{d}t^2 = [a^2(1-\sin^2 t) + b^2\sin^2 t]\,\mathrm{d}t^2 = [a^2 - (a^2-b^2)\sin^2 t]\,\mathrm{d}t^2$$

and we have $\mathrm{d}s = \sqrt{a^2 - (a^2-b^2)\sin^2 t}\,\mathrm{d}t = a\sqrt{1 - \frac{a^2-b^2}{a^2}\sin^2 t}\,\mathrm{d}t$.

Let $x = \frac{\sqrt{a^2-b^2}}{a}\,(\leqslant 1)$, then we have $\mathrm{d}s = a\sqrt{1-x^2\sin^2 t}\,\mathrm{d}t$. Thus, the perimeter of the ellipse, s, can be obtained by calculating

$$s = 4\int_0^{\frac{\pi}{2}} \mathrm{d}s = 4a\int_0^{\frac{\pi}{2}} \sqrt{1 - x^2\sin^2 t}\,\mathrm{d}t.$$

From $\sqrt{1 - x^2\sin^2 t} = 1 - \frac{1}{2}x^2\sin^2 t - \frac{1}{2\times 4}x^4\sin^4 t - \cdots$, we further have

$$s = 4a\left[\frac{\pi}{2} - \frac{x^2}{2}\int_0^{\frac{\pi}{2}} \sin^2 t\,\mathrm{d}t - \frac{x^4}{2\times 4}\int_0^{\frac{\pi}{2}} \sin^4 t\,\mathrm{d}t - \cdots\right].$$

Because of $\int_0^{\frac{\pi}{2}} \sin^{2n} t\,\mathrm{d}t = \frac{1\times 3\times 5\cdots(2n-1)}{2\times 4\times 6\cdots 2n}\times\frac{\pi}{2}$, we thus have $s = 2a\pi[1 - (\frac{1}{2})^2x^2 - (\frac{1\times 3}{2\times 4})^2\frac{x^4}{3} - (\frac{1\times 3\times 5}{2\times 4\times 6})^2\frac{x^6}{5}\cdots]$. Since each term after the second term in the bracket is rather small, the perimeter of the ellipse is approximated by $s = 2a\pi[1 - (\frac{1}{2})^2x^2]$.

By the same arguments, for the second case when $a < b$, the perimeter of the ellipse can be approximated by $s = 2b\pi[1 - (\frac{1}{2})^2y^2]$ where $y = \frac{\sqrt{b^2-a^2}}{b}$. Combining the results for the two cases, let $L = \max(a, b)$ and $L = \min(a, b)$, then the perimeter of the ellipse can be estimated by the formula $s = 2\pi L - \frac{\pi(L+S)(L-S)}{2L}$. □

References

[1] E.R. Davies, Machine Vision, Academic Press, New York, 1997.
[2] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison Wesley, New York, 1992.
[3] M. Sonka, V. Hlavac, R. Boyle, Image Processing, Analysis and Machine Vision, PWS, New York, 1998.
[4] P.V.C. Hough, Method and means for recognizing complex patterns, US Patent 3,069,654, December 18, 1962.
[5] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Communications of the ACM 15 (1972) 11–15.
[6] C. Kimme, D. Ballard, J. Sklansky, Finding circles by an array of accumulator, Communications of the ACM 18 (1995) 120–122.
[7] S. Tsuji, F. Matsumoto, Detection of ellipses by a modified Hough transform, IEEE Transactions on Computers 27 (1978) 777–781.
[8] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognition 13 (1981) 111–122.
[9] J. Illingworth, J. Kittler, The adaptive Hough transform, IEEE Transactions on Pattern Analysis and Machine Intelligence 9 (1987) 690–698.
[10] J. Illingworth, J. Kittler, Survey: a survey of the Hough transform, Computer Vision, Graphics, and Image Processing 44 (1988) 87–116.
[11] E.R. Davies, Finding ellipses using the generalized Hough transform, Pattern Recognition Letter 9 (1989) 87–96.
[12] H.K. Yuen, J. Illingworth, J. Kittler, Detection partially occluded ellipses using the Hough transform, Image and Vision Computing 7 (1989) 31–37.
[13] R.K.K. Yip, P.K.S. Tam, D.N.K. Leung, Modification of Hough transform for circles and ellipses detection using a 2-dimensional array, Pattern Recognition 25 (1992) 1007–1022.
[14] V.F. Leavers, Survey: which Hough transform, CVGIP: Image Understanding 58 (1993) 250–264.
[15] C.T. Ho, L.H. Chen, A fast ellipse/circle detector using geometric symmetry, Pattern Recognition 28 (1995) 117–124.
[16] C.T. Ho, L.H. Chen, A high-speed algorithm for elliptical object detection, IEEE Transactions on Image Processing 5 (1996) 547–550.
[17] D. Ioannou, W. Huda, A.F. Laine, Circle recognition through a 2D Hough transform and radius histogramming, Image and Vision Computing 17 (1999) 15–26.
[18] Y.T. Ching, Detecting line segments in an image – a new implementation for Hough transform, Pattern Recognition Letters 22 (2001) 421–429.
[19] Q. Ji, R.M. Haralick, Error propagation for the Hough transform, Pattern Recognition Letters 22 (2001) 813–823.
[20] H.S. Kim, J.H. Kim, A two-step circle detection from the intersecting chords, Pattern Recognition Letters 22 (2001) 787–798.
[21] A.A. Sewisy, F. Leberl, Detection ellipses by finding lines of symmetry in the images via an Hough transform applied to straight, Image and Vision Computing 19 (2001) 857–866.
[22] S. Climer, S.K. Bhatia, Local lines: a linear time detector, Pattern Recognition Letters 24 (2003) 2291–2300.
[23] E.R. Davies, Truncating the Hough transform parameter space can be beneficial, Pattern Recognition Letters 24 (2003) 129–135.
[24] Y. Furukawa, Y. Shinagawa, Accurate and robust line segment extraction by analyzing distribution around peaks in Hough space, Computer Vision and Image Understanding 92 (2003) 1–25.
[25] K.U. Kasemir, K. Betzler, Detecting ellipses of limited eccentricity in images with high noises levels, Image and Vision Computing 21 (2003) 221–227.
[26] C.P. Chau, W.C. Shiu, Adaptive dual-point Hough transform for object recognition, Computer Vision and Image Understanding 96 (2004) 1–16.
[27] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography, Communications of the ACM 24 (1981) 381–395.
[28] M.A. Fischler, R.C. Bolles, Intelligence: The Eye, the Brain, and the Computer, Addison Wesley, 1987.
[29] L. Xu, E. Oja, P. Kultanan, A new curve detection method: randomized Hough Transform (RHT), Pattern Recognition Letters 11 (1990) 331–338.

[30] N. Kiryati, Y. Eldar, A.M. bruckstein, A probabilistic Hough transform, Pattern Recognition 24 (1991) 303–316.
[31] L. Xu, E. Oja, Randomized Hough transform (RHT): basic mechanisms, algorithms, and computational complexities, CVGIP: Image Understanding 57 (1993) 131–154.
[32] A. Ylä-Jääski, N. Kiryati, Adaptive termination of voting in probabilistic Hough transform, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (9) (1994) 911–915.
[33] H. Kälviäinen, P. Hirvonen, L. Xu, E. Oja, Probabilistic and non-probabilistic Hough transforms: overview and comparisons, Image and Vision Computing 13 (4) (1995) 239–252.
[34] R.A. McLaughlin, Randomized Hough transform: better ellipses detection with comparison, IEEE TENCON. Digital Signal Processing Application 1 (1998) 409–414.
[35] D. Shaked, O. Yaron, N. Kiryati, Deriving stopping rules for the probabilistic Hough transform by sequential analysis, Computer Vision and Image Understanding 63 (3) (1996) 512–526.
[36] H. Kälviäinen, P. Hirvonen, An extension to the randomized Hough transform exploiting connectivity, Pattern Recognition Letters 18 (1997) 77–85.
[37] R.A. McLaughlin, Randomized Hough transform: improve ellipses detection with comparison, Pattern Recognition Letters 19 (1998) 299–305.
[38] N. Kiryati, H. Kälviäinen, S. Alaoutinen, Randomized or probabilistic Hough transform: unified performance evaluation, Pattern Recognition Letters 21 (2000) 1157–1164.
[39] J. Matas, C. Galambos, J. Kittler, Robust detection of lines using the progressive probabilistic Hough transform, Computer Vision and Image Understanding 78 (1) (2000) 119–137.
[40] V. Kyrki, H. Kälviäinen, Combination of local and global line extraction, Real-Time Imaging 6 (2000) 79–81.
[41] T.C. Chen, K.L. Chung, An efficient randomized algorithm for detecting circles, Computer Vision and Image Understanding 83 (2001) 172–191.
[42] C. Galambos, K. Kittler, J. Matas, Gradient based progressive probabilistic Hough transform, IEE Proceedings – Vision, Image and Signal Processing 148 (3) (2001) 158–165.
[43] D. Walsh, A.E. Raftery, Accurate and efficient curve detection in image: the importance sampling Hough transform, Pattern Recognition 35 (2002) 1421–1431.
[44] K.L. Chung, Y.H. Huang, A pruning-and-voting strategy to speed up the detection for lines, circles, and ellipses, Journal of Information Science and Engineering, in press.
[45] J.E. Bresenham, Algorithm for computer control of a digital plotter, IBM System Journal 4 (1965) 25–30.
[46] J.E. Bresenham, A linear algorithm for incremental digital display of circular arcs, Communications of the ACM 20 (1977) 100–106.
[47] M.R. Kappel, An ellipse-drawing algorithms for faster displays, Fundamental Algorithms for Computer Graphics (1985) 257–280.
[48] D. Hearn, M.P. Baker, Computer Graphics, third ed., Prentice-Hall, New York, 1997.
[49] W. Wernick, Analytic Geometry, Silver Burdett, New York, 1968.