PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 38 (2005) 273–287

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

A robust, real-time ellipse detector

Si-Cheng Zhang, Zhi-Qiang Liu*

School of Creative Media, City University of Hong Kong, Hong Kong, China

Received 16 September 2003; received in revised form 12 January 2004; accepted 1 March 2004

Abstract

In this paper, we present an improved ellipse detector that may be used in real-time face detection. In this algorithm we first extract edges from the image using a robust edge detector, which is then followed by a rule-based method. Finally, we decompose the parameter space of the Hough transform to achieve computational efficiency. Our extensive experimental results show indeed that the proposed detector is capable of detecting ellipse features with an excellent accuracy under various image conditions at a high speed (10 frames per second on a Pentium-III 500 MHz PC).
© 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Ellipse detection; Real-time; Edge convexity; Hough transform

1. Introduction

Ellipse detection has been an important task in pattern recognition. Over the last two decades researchers have developed many approaches to ellipse detection. For instance, there are techniques based on the Hough transform (HT) and its variants [1–7], least-squares fitting [8], RANSAC [9], genetic algorithms [10], fuzzy logic [11,12], methods combined with geometrical properties [5–7,10,12], and more recently, competitive learning algorithms [14]. Broadly, these algorithms can be divided into two groups: voting/clustering and optimization. Methods in the first group, which include HT, RANSAC, fuzzy logic, and competitive learning, map sets of points to the parameter space to detect the ellipses through accumulators/clustering. These methods are robust against outlier and occlusion, but computationally expensive. The randomized Hough transform (RHT) [4] has been a popular method in recent years. The second group, which includes least-square, genetic algorithm, optimizes some objective functions for fitting ellipses. Although these approaches are generally accurate, they require pre-processing such as segmentation and grouping. The direct least-squares technique [8] is efficient for fitting ellipse in scattered data. To speed up the process, some geometric properties of ellipses are used [7,13].

Since its introduction in the early 1960s, the HT [1] and its variants have been successfully used in many problems. A major advantage of HT is its insensitivity to imperfect data. HT is able to detect the geometry shapes such as lines, circles and ellipses in various situations; however, computationally it is inefficient which increases with the complexity of shapes. For instance, to detect circles, it requires three-dimensional (3-D) features: center coordinates (x, y), and radius (r), whereas to detect elliptic shapes it needs five parameters: center coordinates (x, y), semi-axis (a, b) and orientation (ρ).

When using HT to detect ellipses [2], the accumulators can become very large (5-D), which makes it inefficient, let alone to perform in real time. One approach is to decompose the parameter space into smaller subspaces. An efficient technique is the piecewise decoupling and decomposing approach based on edge gradient. To speedup the search in the parameter space, another technique, the *focusing strategy*, has become increasingly popular. With these improvements, some researchers have proposed new HT algorithms for the detection of elliptic primitives, among which the fast Hough Transform (FHT) [3] has become popular for the its computational efficiency. Although these methods improve

* Corresponding author. Tel.: +852-2194-2832; fax: +852-2788 -7165.
E-mail address: zliu@cs.mu.oz.au (Z.-Q. Liu).

0031-3203/$30.00 © 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2004.03.014

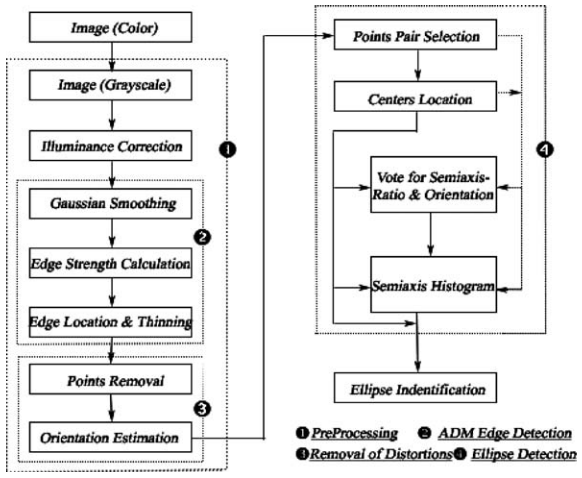

Fig. 1. Architecture of our algorithm.

efficiency to some extent, it is still not useful in many real-time applications, e.g., face detection and recognition in security and surveillance systems.

In this paper, we present a fast, robust ellipse detection algorithm. In this method, we integrate edge directional properties for decomposing the HT parameter space for ellipses and make a full use of candidate point pairs to achieve computational efficiency. To investigate the effectiveness of our algorithm, we have carried out extensive experiments. Fig. 1 illustrates the system architecture.

The remainder of this paper is organized as follows: Section 2 introduces a robust edge detection algorithm and post-processing that removes spurious data and estimates edge convexity. In Section 3 we present in detail the new ellipse detection technique. We demonstrate the performance of the proposed ellipse detection system in Section 4, where we evaluate the experimental results under different testing conditions and make some comparisons, which show that our approach is efficient, robust and accurate. The conclusions are given in Section 5.

2. Edge detection and post-processing

2.1. Overview

In order to detect ellipse effectively in an image we must extract reliable features. This is accomplished in two stages: edge detection to extract the edge information and post-processing to remove spurious data in the edge image for further evaluation.

To detect elliptic shapes, it is necessary to detect edges in the image which is an important stage as the quantity and quality of edge data will affect greatly the performance of the system. Our system first converts the color image in RGB to the gray-scale image. We then apply the Absolute Difference Mask (ADM) algorithm [15] to the image to produce the edge image, together with the information about the edge strength and direction. After we have obtained the edge image, we post-process the edge image by removing spurious edge points to build an edge point map from which we estimate edge convexity based on neighborhood directions. In this way, we obtain necessary and non-redundant information for ellipse detection, which eliminates unnecessary computation.

The above process generates three data sets: an edge-point map, an edge-convexity map and an edge-strength map. The edge-point map is the main source data for the selection of point pairs in Section 3.2. The edge-convexity map plays a key role in the ellipse detection algorithm for feature extraction. And the edge-strength map is used in the voting process in the HT.

2.2. Edge detection

In our application, we are mainly concerned with the detection of quadric/elliptic features. In addition, the system must be able to perform robustly in real time. To achieve these we need a good edge detection algorithm that must satisfy the following three criteria:

(i) Unique edge response.
(ii) Least dependence of control parameters, e.g., threshold values.
(iii) Efficient computation.

The first condition ensures that an edge is unique, which helps reduce the computation and maintain a good accuracy. Over the years numerous edge detectors have been developed [16]. Unfortunately, most common edge detectors, such as Sobel and Robert, usually produce multi responses for an edge, which significantly interfere with data analysis, e.g., the identification of an object, detection of an ellipse, etc. The ability to uniquely define an edge at the right location also improves the reliability and greatly reduces the computational time. The second condition makes the algorithm robust over a large range of different image qualities, which is one of the most desirable properties for a system to perform consistently and reliably in real-world applications. The last condition stipulates that in order for the detector to perform in real time, we must adopt early strategies that are computationally efficient and produce accurate results. For instance, many complex pixel-level operations are quite time-consuming and should be modified while not affecting the result significantly. Although the three conditions are simple enough, they serve as practical guidelines in developing our real-time edge detector.

For the task at hand, through experiments we here found that the ADM algorithm is suitable edge detection algorithm as compared to other more popular edge detectors [17,18]. A major advantage of the ADM algorithm is its simplicity and robust performance, although the original ADM is not

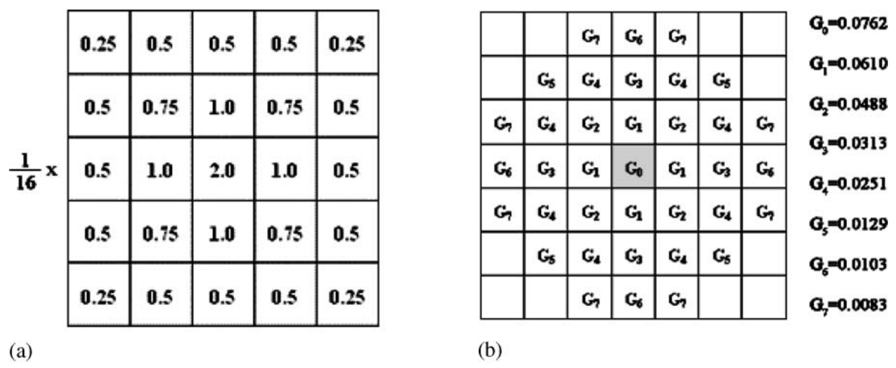

Fig. 2. (a) Original ADM mask; (b) Gaussian circle mask.

the most efficient detector. To improve its efficiency, we propose a three-step operation on gray-scale images:

(i) applying the semi-Gaussian smoothing mask,
(ii) calculating the edge strength and direction at each pixel, and
(iii) locating the final edge.

The first step is to reduce the effect of random noise in the image, as most edge detectors do. The next step calculates the edge strength and direction at each point. Finally, a simplified non-maximum suppression technique is used to determine the final edge with a single-pixel width. In the following, we briefly present the modified ADM algorithm.

The smoothing mask of the original ADM is illustrated in Fig. 2(a), where we change the coefficients to the normalized Gaussian values by Eq. (1) and transform the shape into a circle mask as shown in Fig. 2(b), which more effectively smoothes the image. To maintain computational efficiency, we use a lookup table (LUT) that is calculated offline. Using a 37-pixel (radius = 3) mask is adequate for removing the random noise. The entire operation requires only some arithmetic additions and an index search.

$$G(i,j) = A\mathrm{e}^{-(i^2+j^2)/2\sigma^2} \qquad A \in R^+,\ i,j = 0,1,2,3\ldots\ . \tag{1}$$

The ADM is applied to the smoothed image to calculate the edge strength and direction at every pixel location. We follow the same steps as that in Ref. [15]. A simplified non-maximum suppression of edge strength is performed within a 3×3 mask to determine the possible edge points. Then a threshold of edge strength is applied to the output to produce the localized, single-pixel-wide edge. We adopt the original idea and store the strength and direction information for later use. After these modifications, the algorithm satisfies the three conditions mentioned previously. The detected edge is singe-pixel (unique) wide. The output is affected only by a single parameter, namely, *the edge threshold*. Since the entire computation involves only arithmetic additions, it is very efficient. We will present comparisons in Section 4.

2.3. Post-processing

After edge extraction, we obtain information about edge locations, strength and direction which will be used in ellipse detection. However, since the edge extraction algorithm also introduces noise and distortion into the processed data, it is necessary to remove these effects to make ellipse detection more effective and efficient.

2.3.1. Removal of spurious points

One of the most common side effects of edge detectors is the spurious edge points which must be removed. Broadly, there are two kinds of such points: *completely isolated* points and *excessively connected* points that may happen at some boundary intersections. In our system, an isolated edge point is defined as the point that does not have neighborhood points in a 3×3 window as shown in Fig. 3(a). Although this definition may result in some edge points being deleted from the edge map, it is very effective for the ellipse detection algorithm proposed in this paper. As we will discuss in Section 3.1 such isolated points contribute only to confusion in the selection of point pairs in ellipse detection and make the detection more computationally costly.

Our ellipse detection algorithm directly uses disconnected edge segments as they lend themselves naturally for computing directions. As a result, it does not need to perform the linking operation. In fact our algorithm is so stable that it can perform very well even on scattered edge components. Further, our experiments show that even without the intersection points it is still able to detect ellipses reliably. Therefore, for efficiency we remove also the edge points at intersections in the edge map.

For this we define two points that are considered to be *connected* if and only if they are *within* the distance of $\sqrt{2}$ (excluding $\sqrt{2}$). According to this definition, an *excessively connected* point is surrounded by at least three connected points within a 3×3 window shown in Fig. 3(b). For comparison, Fig. 3(c) shows a case that is not *excessively connected*. Our experiment results have shown that it is

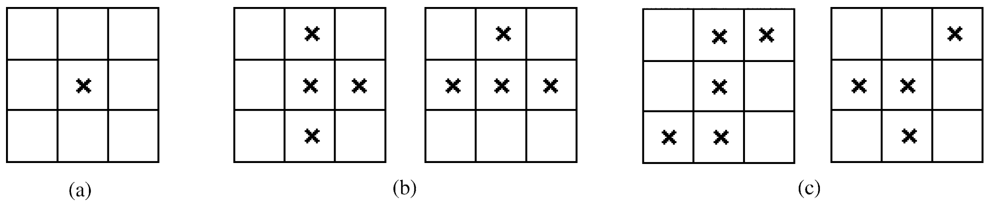

Fig. 3. Removal of spurious points.

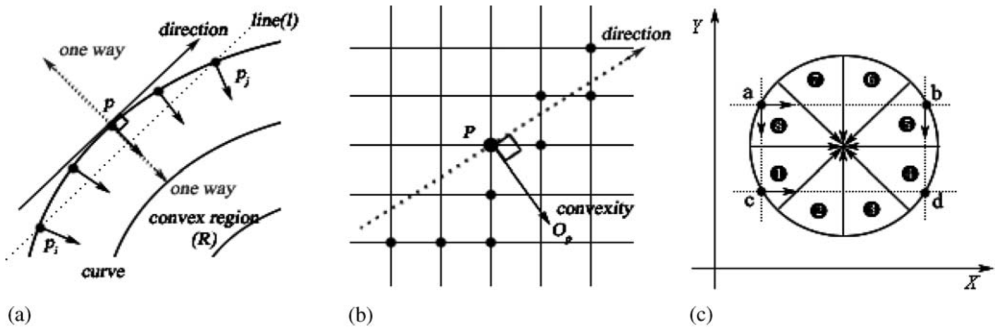

Fig. 4. Estimation of edge convexity.

necessary to remove *excessively connected* points, to avoid the interference in the calculation of convexity to be discussed below.

2.3.2. Estimation of edge convexity

Although edge directional information is useful in edge linking or labeling algorithms, it cannot be applied directly for more complex features, such as quadratic features, which is also suggested in Ref. [6]. As shown in Fig. 4(a), the convexity associated with an edge point plays a crucial role in the detection of elliptic shapes. However, the convexity is usually estimated at edge segments of the entire curve, which can be computationally very costly [19]. Therefore, in the following we propose a simplified and effective algorithm to calculate the convexity.

As shown in Fig. 4(a), convexity is a property of a convex region R in which a line l between any two points, P_i and P_j, lies also in R, which applies to the border of R. We may think a convex curve in terms of the border of the convex region. Therefore, we may estimate convexity by making use of both the edge tangential information and the edge's neighborhood points in the edge map.

Let us look at the edge points within 5×5 window in the *cleaned* edge map obtained previously (see Fig. 4(b)). Intuitively, along the edge, the side that contains more edge points is more likely to indicate a convex region, as in a small neighborhood the direction and the convexity of a curve (an edge segment) change only slightly. For computational efficiency, we divide the quadrant into N equal sections as shown in Fig. 4(c) (here, $N = 8$) and label them, respectively, by a number from ❶ to ❽ to represent their associated convexities. At every edge point P, first we calculate the direction in terms of the tangent slope of the edge k, and then count the neighboring edge points on both sides of the tangent in a $M \times M$ window by Eq. (2), based on which we determine the convexity at this edge point by a vector O_p as shown in Fig. 4(b)

$$C_p = \left\lceil \frac{\left(\arctan(k) + \pi + \eta \cdot \frac{\pi}{2}\right) \cdot N}{2\pi} \right\rceil \bmod N,$$

$$\eta = \operatorname{sgn}\left(\sum_{y-y_0 \geqslant k(x-x_0)} Edge(x,y) - \sum_{y-y_0 < k(x-x_0)} Edge(x,y) \right), \tag{2}$$

where $N = 2^n$, $M = 2m + 1$, $|x - x_0| \leqslant m$, $|y - y_0| \leqslant m$, $m, n = 1, 2, 3 \ldots$

There are two special cases to be considered: (1) if the difference value of edge strength in X direction or Y direction or both are equal to zero, which means no directional information at current point is available, or Eq. (2) if the number of edge points is equal on both sides, which implies no meaningful convexity can be determined. In the first case, for the sake of efficiency and without much sacrifice of accuracy in ellipse detection, we simply exclude it from future computation, whereas in the second case we assign a convexity according to its preceding edge point in the selection of point pairs to be discussed below.

3. Ellipse detection

3.1. Overview

Recently, two groups of researchers have proposed independently a fast ellipse Hough transform (FEHT) for ellipse detection [5,6]. Based on the edge gradient information, they proposed to use the geometric property: two edge points on the ellipse can help to locate the center point and thus to estimate the other four parameters in multiple stages. The combination of geometry and multistage processing further reduces the computational complexity. However, these algorithms are sensitive to the quality of edge map and their efficiency greatly relies on the number of candidate points.

In this section, we present a new algorithm, real-time ellipse Hough transform (RTEHT), which is able to detect ellipses in real time while maintaining an excellent accuracy. In our ellipse detector, we use the edge detector described in Section 2.2. In order to achieve a good detection accuracy, we employ a heuristic scheme for the selection of point pairs. This scheme strives to look for more meaningful clues with fewer candidate points, which reduces the computational cost. We make a full and consistent use of the point pairs throughout the entire process. In this way, we can achieve a good efficiency and accuracy. To further reduce the processing time, we follow the lazy evaluation strategy and use efficient data structures in implementation. Furthermore, our algorithm has the built-in flexibility to detect not only exactly fitted ellipse primitives but also *ellipse-like* shapes which makes it very useful in many real-world applications.

Broadly, the algorithm has four steps:

(1) Selecting point pairs to be used in the following three steps. This is a crucial step as a set of properly selected point pairs will ensure the success of the entire algorithm.
(2) Locating all possible ellipse centers. This represents the main computational component of the algorithm.
(3) Computing vote for the semi-axis ratio and orientation in the two-dimensional accumulator. For each candidate ellipse center we look for all possible semi-axis ratios and orientations.
(4) Estimating the long axe a or short axe b of the ellipse candidate with each set of the four parameters (center x *and* y, semi-axis ratio a/b and orientation) obtained in the previous steps.

In the following sections, we will present the algorithm and its implementation in detail.

3.2. Selection of point pairs

The selection of point pairs is a crucial step in the algorithm, as the number of candidate points directly determines the computational cost and the quality of the selected point pairs is critical to detection accuracy.

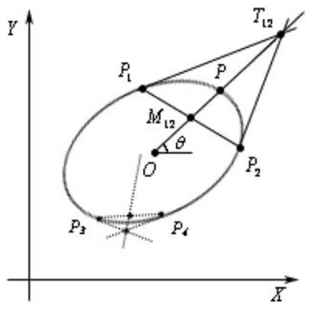

Fig. 5. Selection of point pairs.

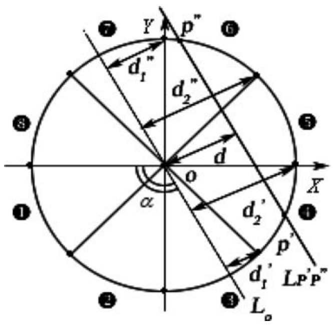

Fig. 6. Convexity matching scheme.

As illustrated in Fig. 5, P_1 and P_2 are two points on an ideal ellipse, T_{12} is the intersection of the two tangent lines from P_1 and P_2, and M_{12} is the middle point of the segment line P_1P_2. It has been proved that for an ideal ellipse, M_{12} and T_{12} are on the line that crosses the center O [3]. Therefore, if enough such point pairs are available, we can locate the center point of the ellipse. Based on this observation, researchers have proposed different schemes to search for point pairs [2–6,13]. In this paper, we present a new approach that further improves the search process.

In order to obtain a minimum number of point pairs that are most useful in detecting ellipses, in our algorithm, we consider the following three aspects.

(i) *Associated convexities*: The most crucial consideration in our approach is the convexity matching scheme. Geometrically, two points that consist of a pair on an ellipse should have a certain relationship in convexity. For instance, let us consider a line that passes through the center of a circle (see L_0 in, Fig. 6). It will make two intersecting points (a pair of points) on the circle that are on the exact opposite sides of the circle and their associated convexities are also opposite to each other. Such a relationship changes smoothly when the line moves away from the center (see $L_{p'p''}$ in Fig. 6). Therefore, we set up one rule as Eq. (3) that governs such a relationship in the selection of point pairs, so as to make sure that they most likely belong to the same elliptic shape. With the method introduced in Section 2.3, we can divide the quadrant equally into N partitions to represent the convexity regions; and Fig. 6 shows the matching scheme when $N = 8$. In case of unique edge response, this rule results in a more accurate determination of whether two points are on the same elliptic shape. As described in the following, this rule is important in that it influences not only the effectiveness of the selection operation, but also the accuracy of ellipse detection

$$C_{p'} = \left(\left\lceil \frac{\alpha \cdot N}{2\pi} \right\rceil + i \right) \bmod N \quad \text{if } d'_{i-1} \leqslant d < d'_i,$$
$$C_{p''} = \left(\left\lceil \frac{(\alpha + \pi) \cdot N}{2\pi} \right\rceil - j \right) \bmod N \quad \text{if } d''_{j-1} \leqslant d < d''_j, \tag{3}$$

where

$$d'_i = \sin\left(\beta + \frac{2\pi \cdot (i-1)}{N}\right) \cdot r,$$
$$d''_j = \sin\left(\frac{2\pi}{N} - \beta + \frac{2\pi \cdot (j-1)}{N}\right) \cdot r,$$
$$\beta = \alpha \bmod \frac{2\pi}{N}, \quad 0 \leqslant \alpha < 2\pi,\ N = 2^n,$$
$$i, j = 1, \ldots, n-1, \quad d'_0 = d''_0 = 0.$$

(ii) *Appropriate distance*: Two points of interest should be within a reasonable range; that is, if two points are located beyond this range, they may contribute little to correct detection. For example, the two points P_3 and P_4 in Fig. 5 would cause more error if they were selected as the candidate pair since their positions are too close. For efficiency, we can define a target range in which an ellipse may exist. In our algorithm, for instance for a pair of horizontal points, we measure the distance Dis_x by taking into account of the convexity constraint in Eq. (4). In this way, we will be able to find all matched edge points for each edge point to generate the correct candidate pairs. As a result, this rule constrains the search for point pairs only in a specified range and thereby reduces the computational cost.

(iii) *Reasonable angles*: Finally, two tangent lines that intersect must also be within a reasonable angle range. For instance, two parallel lines are obviously unreasonable. Through extensive experiments we have found that the calculation of intersection is more accurate when the angle is around 90°. In our algorithm, we specify a range of angles, which enables us to exclude most points that produce false clues for later calculations.

With these three rules, the algorithm is able to extract a set of high-quality point candidates for detecting the elliptic shapes reliably.

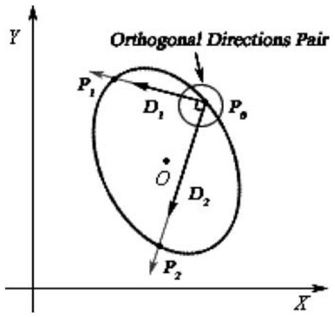

Fig. 7. Stream's Scan process.

$$Dis_x(P_i, P_j) = \begin{cases} [T_{x1}, T_{x2}] & \text{if } \begin{cases} C_i = 2 & \text{or} \\ C_i = 7 & \text{or} \\ C_i = 1 & \text{and } C_i = 3 \text{ or} \\ C_i = 8 & \text{and } C_j = 6 \end{cases} \\ [T_{x3}, T_{x4}] & \text{otherwise} \end{cases} \tag{4}$$

where $T_{x1} = A_{\min}$, $T_{x2} = \sqrt{2}A_{\max}$, $T_{x3} = \min(\sqrt{2}A_{\min} < A_{\max})$ is the target size of the long axe a, and C_i is the convexity of point P_i.

To further improve computational efficiency, in the search of the candidate points in the edge map, we have developed a *Stream's Scan* process, which searches, from a source point (e.g., P_0 in Fig. 7), for point pairs along two lines that are perpendicular to each other. Referring to Fig. 7, when applying the above three rules along D_1 or D_2, the source point P_0 and the matched point P_1 or P_2 are selected as a pair, respectively. At every edge point, the process proceeds continuously with the following three steps. First, we estimate the possible target convexities that match the source point according to rule (i), then we search the targets within a distance defined by rule (ii) from the source point, and finally we find the matched points that satisfy rule (iii) to generate the right candidate pairs. The reason for orthogonal directions is for another important rule (namely, the "4–2–1" rule) governing the point pairs grouping to be introduced in Section 3.3. All the qualified paring points are found and stored in a dynamic link table with grouping information, which avoid iterative sliding search operation for subsequent groupings such as Ref. [5]. The scanning process needs to pass only once through the edge map and acts much like a stream process, which results in high efficiency.

In order to avoid repetition in the proceeding, the scanning order in each direction is adaptively adjusted according to the convexity at the current source point. More specifically, for instance as Fig. 4(c) shows, we search point pairs in only horizontal and vertical (HV) scanning directions. Initially the convexity is 8 at point $\boldsymbol{a}$, the target convexity may be at the opposite quadrants within an angle range indicated by 5 or 6 in the horizontal scanning direction, and by 1 or 2 in the vertical scanning direction. Say, a vertical scan from point ***a*** arrives at point ***c*** that has an associated convexity 1. The next targets from point ***c*** are convexities 3 and 4 in the horizontal scanning direction, and none in the vertical direction from point ***c*** as it has been scanned from point ***a*** to ***c***. This approach is flexible in that it allows to be executed along multiple direction pairs *simultaneously* to improve performance. In our implementation, in a similar fashion, we also use the positive diagonal and negative diagonal (PN) scanning simultaneously with HV. As a result, the scanning process is quite efficient in producing useful point pairs. The comparison for various scanning schemes will be discussed in Section 4.

3.3. Location of center

As illustrated in Fig. 5, if a line set consists of such lines as $T_{12}M_{12}$, the center of ellipse can be derived from the line intersections belonging to the set. Due to unavoidable distortions, the intersections are usually distributed in a small range *around* the exact centers. Therefore, they can be found by searching the local maxima of 2-D accumulator for intersections:

$$\frac{((x-a_0)\cos(\rho)+(y-b_0)\sin(\rho))^2}{a^2} + \frac{((x-a_0)\sin(\rho)+(y-b_0)\cos(\rho))^2}{b^2} = 1, \tag{5}$$

where (a_0, b_0) is the center of ellipse, $\rho \in (0, 2\pi]$ is the orientation of ellipse with respect to x axis and (a, b) are the semi-axis.

If we assume points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, we can derive the following relationships from Eq. (5):

$$x_{T12} = \frac{y_2 - \phi_2' x_2 - y_1 + \phi_1' x_1}{\phi_1' - \phi_2'},$$

$$y_{T12} = \frac{\phi_2' y_1 - \phi_1' y_2 + \phi_1'\phi_2'(x_2 - x_1)}{\phi_1' - \phi_2'},$$

$$x_{M12} = \frac{x_1 + x_2}{2}, \qquad y_{M12} = \frac{y_1 + y_2}{2},$$

$$q_1 = \frac{y_2 - y_1}{x_2 - x_1},$$

$$q_2 = \frac{y_{T12} - y_{M12}}{x_{T12} - x_{M12}} = \frac{(\phi_1' + \phi_2')(y_1 - y_2) + 2\phi_1'\phi_2'(x_2 - x_1)}{2(y_1 - y_2) - (\phi_1' + \phi_2')(x_2 - x_1)}, \tag{6}$$

where ϕ_1' is the tangent angles at edge point P_1 and P_2, respectively, q_1 and q_2 are the slopes of lines P_1P_2 and $T_{12}M_{12}$, respectively.

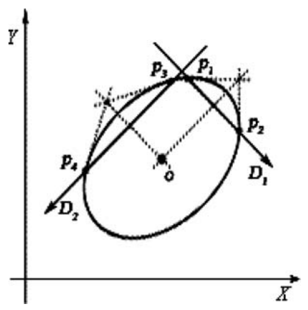

Fig. 8. "4–2–1" rule in grouping.

The intersection (a_0, b_0) can be calculated as follows:

$$a_0 = \frac{y_{M34} - q_4 x_{M34} - y_{M12} + q_2 x_{M12}}{q_2 - q_4},$$

$$b_0 = \frac{q_2 y_{M34} - q_4 y_{M12} + q_2 q_4 (x_{M34} - x_{M12})}{q_2 - q_4}. \tag{7}$$

Eq. (7) shows locating the center by two pairs of points on ellipse, which we call the "*four-points-two-directions-one-intersection*" (in short "4–2–1") process. Fig. 8 illustrates the "4–2–1" rule: four edge points (points P_1 and P_2, points P_3 and P_4) located at two orthogonal directions, respectively (direction D_1 and D_2, such as PN) can make one intersection (point O). For simplicity, we will refer to the four points in the "4–2–1" process as 4P in the following sections. It is another rule we set up in our algorithm that governs the grouping of point pairs. While maintaining the independence of parameter decomposition in FHT, the "4–2–1" rule compensates to some extent for the loss of connectivity in contents, which differs from other algorithms available in the literature such as Refs. [5,6]. Since it is necessary to use the accumulator to identify all possible values, we must consider the following three aspects to improve the quality of the output:

(i) *Compute intersections*: How to group two pairs of points from two candidate sets to produce the intersection. Through experiments, we have discovered that two pairs of points that are oriented perpendicularly to each other usually intersect more closely to the exact center. From such two sets of point pairs, we compute all the intersections using the "4–2–1" process to fill in the polling accumulator. In order to carry out the "4–2–1" process sequentially, we used a "neighborhood window" (see Fig. 9): within a small window, we search for two point pairs along orthogonal directions, each of which should have one and only one point in the window. This ensures that such two point pairs are associated with the same candidate ellipse.
(ii) *Verify intersections*: To determine the eligibility of the computed intersections, we remove those intersections that physically exist but are beyond a reasonable range. Further, if the convexities of the $4P$ do not focus

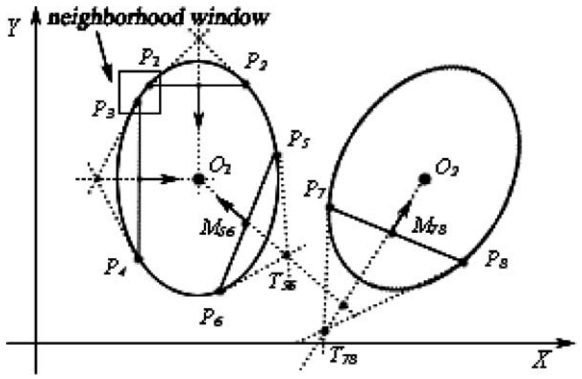

Fig. 9. Location of centers.

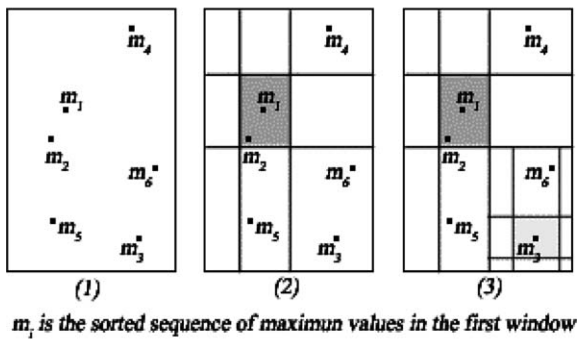
m_4
m_1
m_2
m_6
m_5
m_3
(1)
(2)
(3)
m_i is the sorted sequence of maximun values in the first window

Fig. 10. Search of intersections.

towards the same center, the intersection obtained by Eq. (7) is illegal in view of the ellipse geometric property. For instance, as shown in Fig. 9, although P_5 and P_7 could be included in a "neighborhood window" and the line $T_{56}M_{56}$ intersects with $T_{78}M_{78}$ the intersection does not indicate a legitimate center, since the convexities associated with P_5P_6 and P_7P_8 tell that the two pairs cannot share an ellipse. The solution is to define some markers for matching according to the convexities associate with the point pairs during the selection stage. This constraint ensures that the intersection exists only within the convexity regions of the $4P$. In our algorithm, we require that the horizontal point pairs intersect only with the vertical ones whose convexities converge orthogonally toward the same center. With the measures of verification, we can effectively reduce erroneous intersections, and also improves efficiency.

(iii) *Search for centers among the intersections*: To search for local maxima in the 2-D accumulator of intersections, we propose a simplified focusing algorithm to improve efficiency, which is inspired by Califano and Bolle [20]. The focusing stratery has three steps shown in Fig. 10. First, the local maximum is found in the current window (initially it is the entire image) and a sorted sequence of values that are above a threshold is obtained. Then the window is non-uniformly divided into 3×3 sub-windows with the middle one centered at the location of the local maximum. Finally, in each sub-window, repeat the second step according to the sorted sequence. Our modified focusing algorithm is different from Ref. [20] in that the dimensions of the sub-windows are task dependent, determined adaptively by local distribution density estimated by histogram and predefined threshold. It is a semi-recursive process in that it subdivides only at the locations of the possible local maxima. As a result, the focusing is able to quickly find all the qualified local maxima.

3.4. Vote for the semi-axis ratio and orientation

In order to find the relationship between the semi-axis ratio and orientation of an ellipse, we decomposed an ellipse into two orthogonally projected ellipses. Here we adopted the ellipse polar equation as in the following equation for the sake of geometric property

$$[(x_\rho(\theta) - a_0), (y_\rho(\theta) - b_0)] = [(x(\theta) - a_0), (y(\theta) - b_0)] \begin{bmatrix} \cos(\rho) & \sin(\rho) \\ \sin(\rho) & \cos(\rho) \end{bmatrix}, \quad (8)$$

where $(x_\rho(\theta), y_\rho(\theta))$ is the point with orientation θ respect to the long axe a if defining

$$a_x = a\cos(\rho), \quad b_x = -b\sin(\rho),$$

$$a_y = a\sin(\rho), \quad b_y = b\cos(\rho),$$

$$N = \frac{b_y}{a_x} = \frac{b}{a}, \quad K = \frac{a_y}{a_x} = \tan(\rho), \quad (9)$$

after some complicated algebraic derivations as proved in Ref. [5] we get finally,

$$-N^2 = \frac{(q_1 - K)(q_2 - K)}{(1 + q_1K)(1 + q_2K)}. \quad (10)$$

Eq. (10) has four parameters: q_1, q_2, N and K, where q_1 and q_2 are slopes for lines P_1P_2 and $T_{12}M_{12}$, respectively, and N and K denote semi-axis ratio (b/a) and orientation (ρ) respectively. For each center candidate, we look for its associated point pairs to calculate q_1 and q_2 using Eq. (6). Then for each point pairs found, the N–K accumulator is continuously updated according to Eq. (10). The size of the accumulator depends on the reasonable range of N–K. The search scheme for local maxima in the accumulator is similar to the focusing algorithm introduced in the previous section. Based on the local maxima, the corresponding orientation (ρ) can be computed using anti-trigonometric operations.

Since in Eq. (10), only a pair of points is used in computing N–K, it is not enough to improve the accuracy. In our algorithm, we use $4P$ that contributes to locating the center

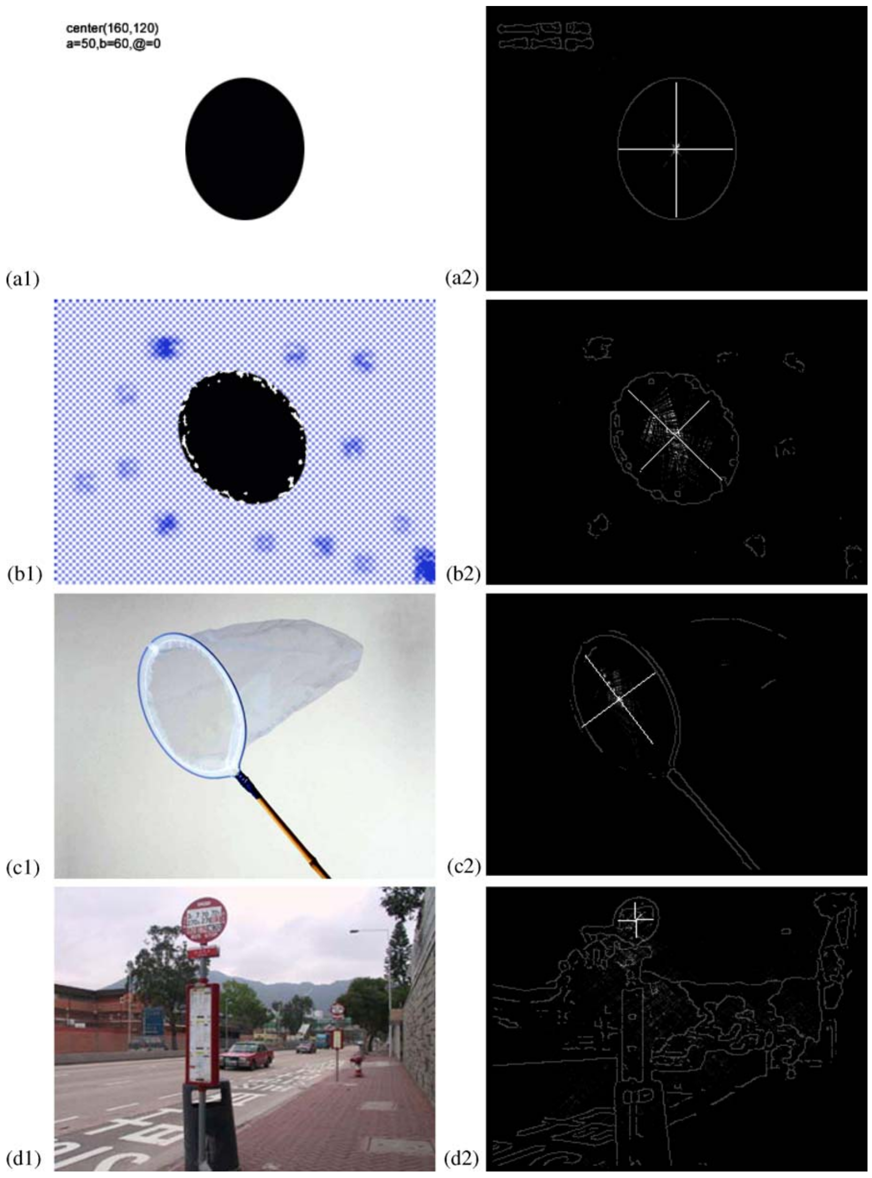

Fig. 11. Detection of single ellipse.

in the "4–2–1" process, which are grouped into two pairs of points. From Eq. (10) we can see that the $4P$ are sufficient for a definite solution for K:

$$K = \pm\sqrt{1 + \frac{(q_1q_2+1)(q_3+q_4)-(q_3q_4+1)(q_1+q_2)}{q_1q_2-q_3q_4}}, \tag{11}$$

where q_1q_2 belongs the point pairs in the first direction and q_3q_4 the other. For the $4P$, we can calculate the corresponding N using Eq. (10).

Since the above mentioned $4P$ and one intersection in Eq. (11) are closely related, it is important to assign this solution a higher weight when updating the accumulator. The resulting N–K accumulator consists of weighted voting values that are useful in generating local maxima.

3.5. *Estimation of semi-axis*

After the four parameters (a_0, b_0, N, K) have been computed from two 2-D accumulator it remains to estimate the long axe a and the short axe b. After some complex

Table 1
Experiment results of single ellipse detection

Image in Fig. 11	Image size	Accuracy on parameters			Efficiency on stages (ms)						
		(a_0, b_0) (%)	(N, K) (%)	(A, B) (%)	Preprocessing		Ellipse detection				Total processing time
					Edge detection	Info. adjustment	Pairs selection	Center location	N–K vote	A–B estimation	
(al–a2)	320 × 240	98	96	95	40	10	10	15	10	5	90
(b1–b2)	320 × 240	94	92	90	55	10	15	15	10	5	110
(c1–c2)	320 × 240	92	90	89	60	10	15	20	10	5	120
(d1–d2)	320 × 240	90	88	86	70	10	20	25	15	5	145

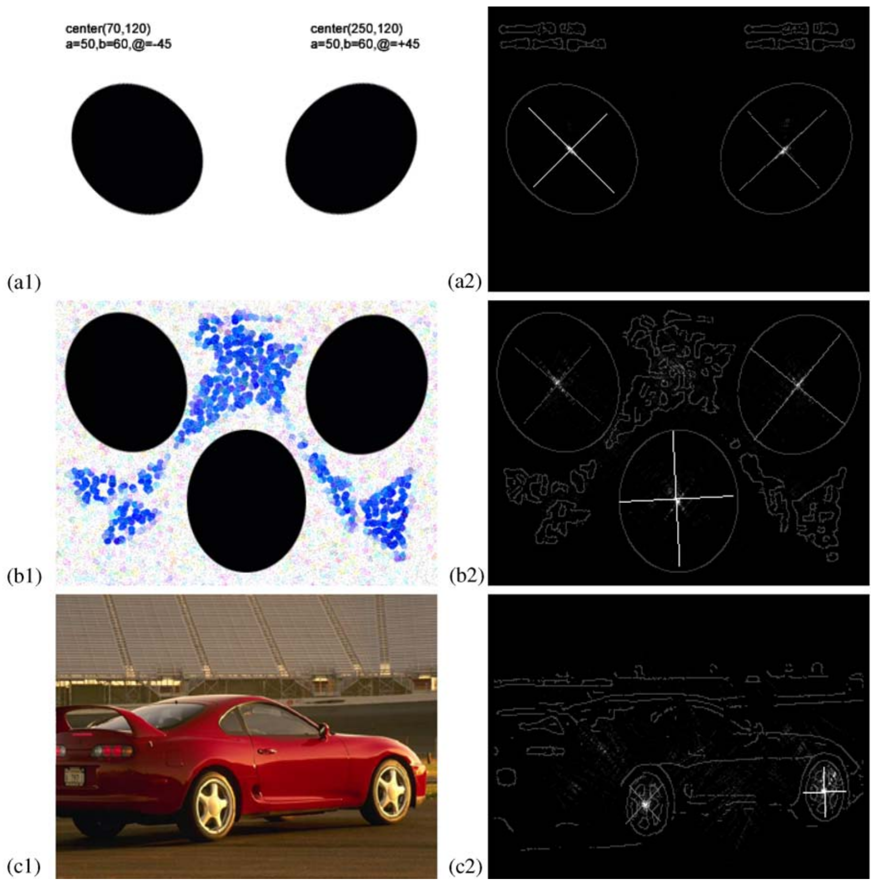

Fig. 12. Detection of multiple ellipses.

derivations from Eqs. (8)–(10), we obtain the following:

$$x_0 = \frac{(x - a_0)}{\sqrt{K^2 + 1}} + \frac{(y - b_0)K}{\sqrt{K^2 + 1}},$$

$$y_0 = \frac{(x - a_0)K}{\sqrt{K^2 + 1}} + \frac{(y - b_0)}{\sqrt{K^2 + 1}},$$

$$a_x = \sqrt{\frac{x_0^2 N^2 + y_0^2}{N^2(1 + K^2)}}, \tag{12}$$

where (x, y) are the image coordinates.

Applying the same principle as that in Ref. [5], we generate a histogram for estimating a_x. Then the values of the long axe a and the short axe b can be computed from

Table 2
Relationship between efficiency and the number of candidate ellipses

Image type	Image size	Possible ellipse instances	Detected result	Total processing time (ms)	Processing time per ellipse (ms)
Synthetic clear image	320 × 240	1	1	90	90
(such as Fig. 12 a1–a2)	320 × 240	2	2	120	60
	320 × 240	3	3	135	45
Synthetic dirty image	320 × 240	1	1	110	110
(such as Fig. 12 b1–b2)	320 × 240	2	2	130	75
	320 × 240	3	3	145	48
Real world image	320 × 240	1	1	125	125
(such as Fig. 12 c1–c2)	320 × 240	2	2	150	75
	320 × 240	3	3	160	53

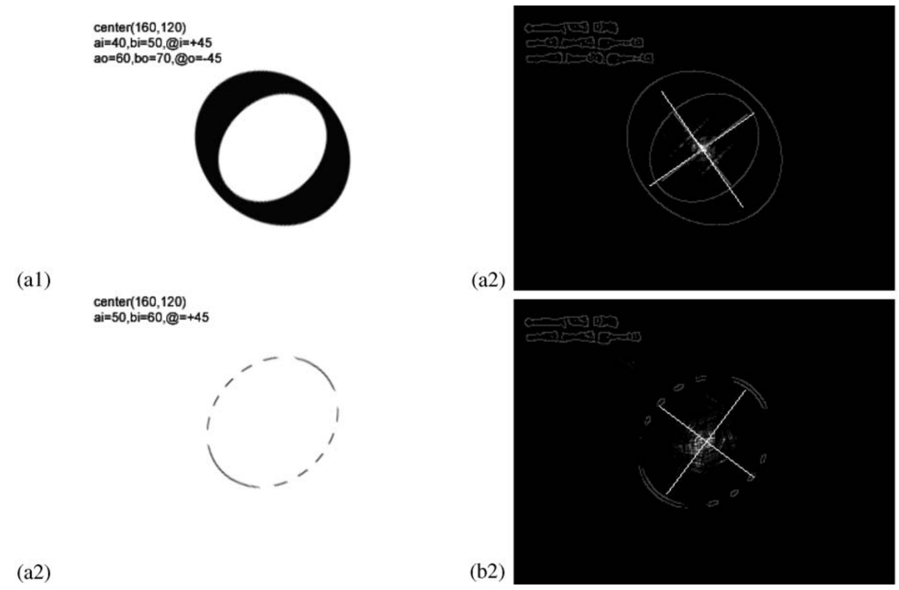

Fig. 13. Special cases.

Eq. (9). Finally, an ellipse candidate is identified with a parameter set (a_0, b_0, N, K, a or b).

4. Experiment results and evaluation

We have conducted extensive experiments with various image qualities. The algorithm is implemented in C++ with MS Visual C++ 6.0 compiler on MS Windows2K running on a desktop PC with Intel PIII 500 MHz processor and 256M RAM. First, we apply the algorithm to synthetic images that have an idea ellipse with and without noise. Then we test images with real-world objects. Fig. 11 shows some of the test images and results. The experiment results are listed in Table 1. We can see some general trends from the table: the accuracy of parameter estimation decreases in the order of scene complexity in terms of the number of objects in the image, but is very well maintained to a reasonable level, especially on the locations of possible centers. In terms of efficiency, the processing time for each stage increases slightly with the scene complexity. The preprocessing stage took almost half of the total processing time, mostly on edge detection.

Fig. 12 shows the detection of multiple ellipses in images. Table 2 shows the relationship between efficiency and the number of candidate ellipses. From the table we can see that the computational efficiency per-ellipse is improved significantly. Figs. 13(a2) and (b2) show that the algorithm has correctly detected concentric ellipses and partial occluded ellipse respectively. Fig. 14 shows the detection of ellipse-like features, where the rectangles indicate the evidence of ellipse-like shapes. From these figures we

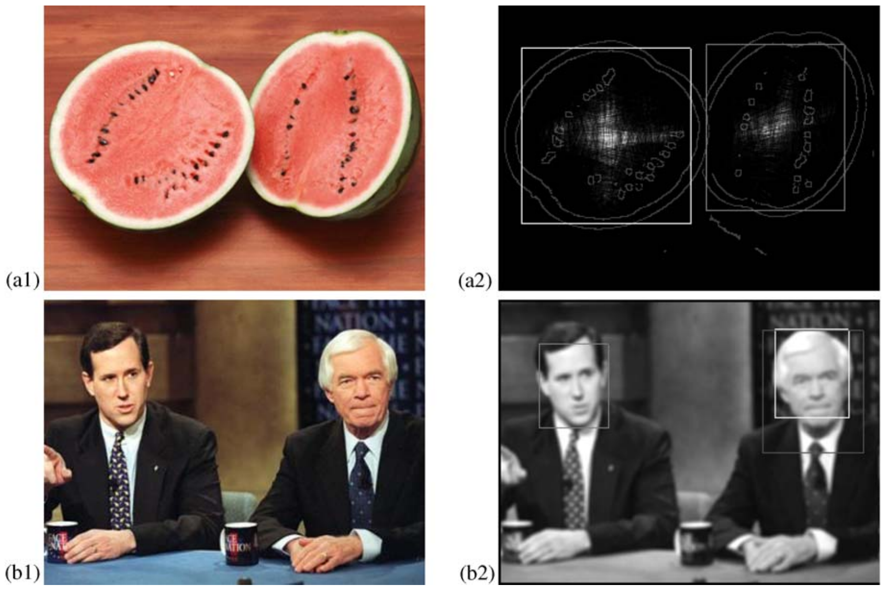

Fig. 14. Detection of ellipse-like shapes.

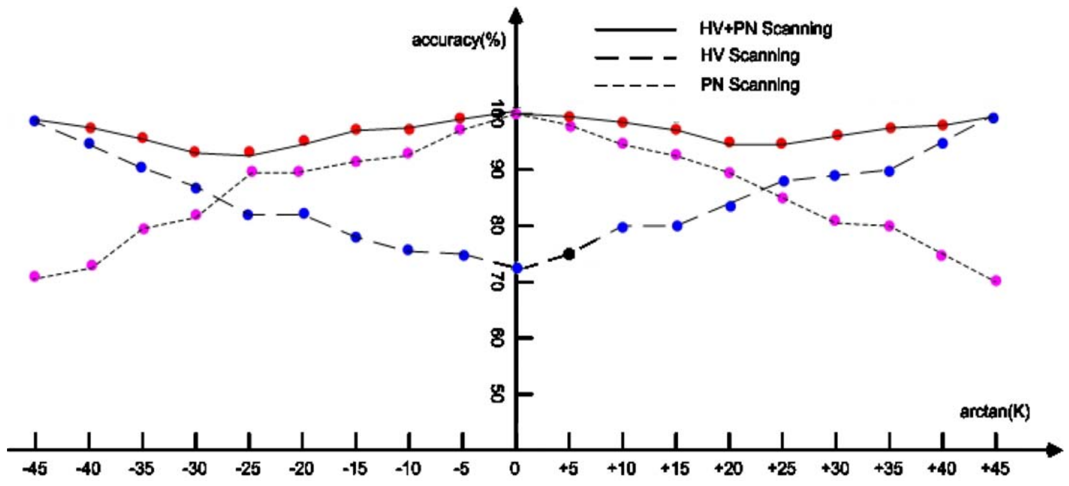

Fig. 15. Relationships between accuracy of N–K detection and different orientation.

can see that such special situations can be handled very well.

We have also carried out experiments to find out the relationships between the accuracy N–K calculation and various ellipse orientations with different scanning schemes mentioned in Section 3.2. Fig. 15 shows the results, from which we can see that by combining multiple pairs of scanning directions in the scanning operation, we can indeed improve the accuracy.

In order to find out the influence of edge detection techniques on ellipse detection, we also conducted experiments using three representative detectors: modified ADM, Canny and SUSAN. Fig. 16 shows the edges of the same image with a 5×5 smoothing mask. Table 3 shows the results. It is evident from the results that ADM is less dependent on the control parameters and computationally more efficient, which make ADM a preferred choice in our algorithm.

To further demonstrate the effectiveness of our algorithm, we re-implemented the algorithms in Refs. [5,6] based on the pseudocode in Refs. [6,21]. Table 4 shows the results under the same condition, which indeed show that our

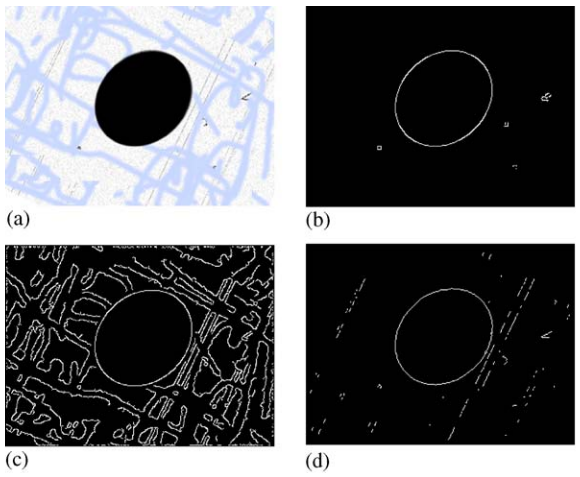

Fig. 16. Different edge detectors on the performance of the ellipse detection algorithm: (a) original image; (b) modified ADM detector; (c) canny detector; (d) SUSAN detector.

algorithm is far more superior due to the careful selection of point pairs and matching scheme. Moreover, we have also performed comparisons with some HT-based ellipse detectors summarized in Ref. [4]: Randomized Hough Transform (RHT), Probabilistic Hough Transform (PHT) and Geometric Symmetry Hough Transform (GSHT). Fig. 17 shows that proposed algorithm exceeds both in computational efficiency and detection accuracy by a big margin.

5. Conclusions

In this paper we have presented a new algorithm based on edge convexity, and FHT for real-time ellipse detection. This algorithm selects a least number of highly informative point pairs for detecting ellipses, which has enabled us to detect ellipses with real-time efficiency and excellent accuracy. Our experimental results have shown that the proposed algorithm is efficient, robust, accurate, capable of detecting multiple ellipses, and effective to gather evidence for partial ellipses or even ellipse-like shapes.

To further improve our algorithm, we are investigating ways to reduce errors in convexity matching caused by the biases resulted from image sampling, and developing more adaptive matching rules to achieve a better performance in accuracy and efficiency. We are now using this algorithm in face detection in real-time applications, which will be presented in another paper.

6. Summary

In this paper, we present an improved ellipse detector that may be used in real-time face detection. In this algorithm we first extract edges from the image using a robust edge

Table 3
Comparisons between different edge detectors

Edge detectors	Detector type	Unique edge response	Control parameters	Edge quality		Efficiency in computation		
				Edge details	Curvature reservation	Paris selected	Edge detection (ms)	Ellipse detection (ms)
ADM	Advanced	True	1	Coarse	Good	473	50	100
SUSAN	Advanced	True	5	Fine	Best	624	55	120
Canny	Advanced	True	3	Normal	Normal	809	65	140
Laplacian–Gaussian	Second-order step	False	1	Fine	Poor	1041	100	180

Table 4
Comparisons with FEHT

Image type	Accuracy (only $[a_0, b_0]$ is list)			Efficiency in computation		
	FEHT in Ref. [5] (%)	FEHT in Ref. [6] (%)	RTEHT (%)	FEHT in Ref. [5] (ms)	FEHT in Ref. [6] (ms)	RTEHT (ms)
Synthetic clear image (such as Fig. 11 a1–a2)	99	97	98	3640	2520	90
Synthetic dirty image (such as Fig. 11 b1–b2)	94	93	94	3980	2790	110
Real world image (such as Fig. 11 c1–c2)	90	88	92	4400	3250	120

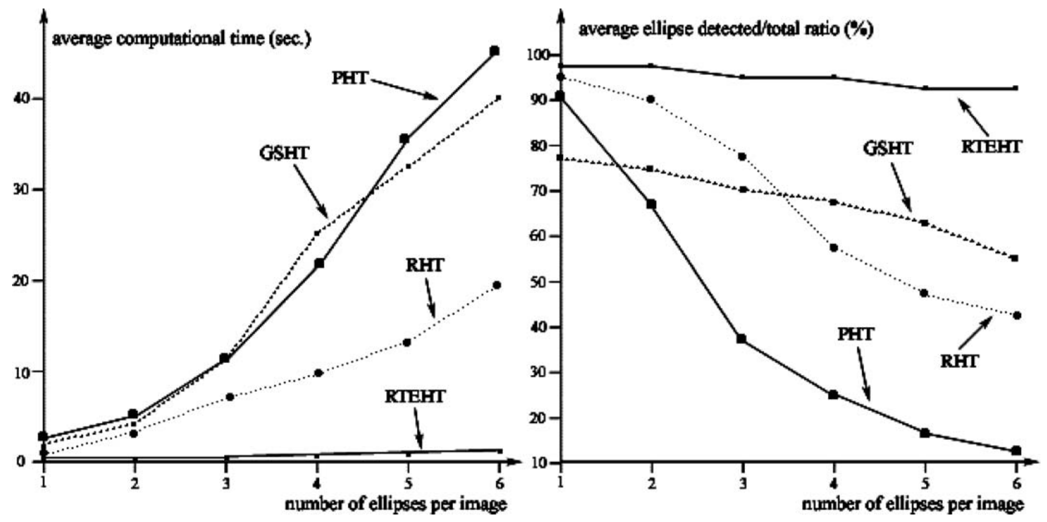

Fig. 17. Comparisons between typical ellipse detectors over serials of images.

detector, which is then followed by a rule-based method. Finally, we decompose the parameter space of the Hough transform to achieve computational efficiency. Our extensive experimental results show indeed that the proposed detector is capable of detecting ellipse features with an excellent accuracy under various image conditions at a high speed (10 frames per second on a Pentium-III 500 MHz PC).

Acknowledgements

This project was supported in part by a Hong Kong Research Grants Council (RGC) Project No. CityUH.K #9040690-873, a Strategic Development Grant (SDG) Project No. #7010023-873, an Applied Research Grant Project No. #9640002-873 and RCMT Project No. #9360080-873, from City University of Hong Kong.

References

[1] P.V.C. Hough, Method and means for recognizing complex patterns, US Patent, 3069654, 1962.
[2] S. Tsuji, F. Matsumoto, Detection of ellipses by a modified Hough transformation, IEEE Trans. Comput. 27 (8) (1978) 777–781.
[3] H. Li, M.A. Lavin, R.J. Le Master, Fast Hough transform: a hierachical approach, J. Comput. Vision Graphics Image Process. 36 (1986) 139–161.
[4] R.A. McLaughlin, Randomized Hough transform: improved ellipse detection with comparison, Pattern Recognition Lett. 19 (3–4) (1998) 299–305.
[5] A.S. Aguado, M.E. Montiel, M.S. Nixon, On using directional information for parameter space decomposition in ellipse detection, Pattern Recognition 28 (3) (1996) 369–381.
[6] N. Guil, E.L. Zapata, Lower order circle and ellipse Hough transform, Pattern Recognition 30 (10) (1997) 1729–1744.
[7] A. Sewisy, F. Leberl, Detection ellipses by finding lines of symmetry in the images via an hough transform applied to straight lines, Image Vision Computing 19 (12) (2001) 857–866.
[8] A. Fitzgibbon, M. Pilu, R.B. Fisher, Direct least square fitting of ellipses, IEEE Trans. PAMI 2l (5) (1999) 477–480.
[9] Yu Chin Cheng, Samuel C. Lee, A new method for quadratic curve detection using K-RANSAC with acceleration techniques, Pattern Recognition 28 (5) (1995) 663–682.
[10] Peng-Yeng Yin, A new circle/ellipse detector using genetic algorithms, Proceedings of the IPPR on CVGIP Taipei, Taiwan, 1998, pp. 362–368.
[11] R. Krishnapuram, O. Nasraoui, H. Frigui, The Fuzzy C spherical shells algorithms: a new approach, IEEE Trans. Neural Networks 3 (5) (1992) 663–671.
[12] R.N. Dave, Generalized fuzzy c-shells clustering and detection of circular and elliptical boundaries, Pattern Recognition 25 (7) (1992) 713–721.
[13] C. Ho, L. Chan, A fast ellipse/circle detector using geometric symmetry, Pattern Recognition 28 (1) (1995) 117–124.
[14] Y.J. Zhang, Z.Q. Liu, Self-splitting competitive learning: a new on-line clustering paradigm, IEEE Trans. Neural Networks l3 (2) (2002) 369–380.
[15] F.M. Alzahrani, T. Chen, A real-time edge detector: algorithm and VLSI architecture, Real-Time Imaging 3 (1997) 363–378.
[16] M. Heath, S. Sarkar, T. Sanocki, K. Bowyer, Comparison of edge detectors: a methodology and initial study, Comput. Vision Image Understanding 69 (1) (1998) 38–54.
[17] J. Canny, A computational approach to edge detection, IEEE Trans. PAMI 8 (6) (1986) 679–697.
[18] S.M. Smith, J.M. Brady, SUSAN—a new approach to low level image processing, Int. J. Comput. Vision 23 (1997) 45–78.
[19] M.J. Donahue, S.I. Rokhlin, On the use of level curves in image analysis, Image Understanding 57 (3) (1993) 185–203.

[20] A. Califano, R.M. Bolle, The multiple window parameter transform, IEEE Trans. PAMI 14 (12) (1992) 1157–1170.

[21] Mark Nixon, Alberto Aguado, Feature Extraction & Image Processing, Newnes, Oxford, 2002, pp. 16l–214.

About the Author—SI-CHENG ZHANG received his B.Sc. degree in Computer Science from University of Science and Technology of China in 2001. Since then, he has been a Ph.D. student in the School of Creative Media, City University of Hong Kong. His research interests include image processing, computer vision and pattern recognition.

About the Author—ZHI-QIANG LIU is currently a Professor with School of Creative Media, City University of Hong Kong and Department of Computer Science and Software Engineering, the University of Melbourne. He received a M.A.Sc. degree in Aerospace Engineering from the Institute for Aerospace Studies, The University of Toronto, and a Ph.D. degree in Electrical Engineering from The University of Alberta, Canada. In addition to doing research on neural-fuzzy systems, machine learning, human media systems, computer vision, mobile computing, and computer networks, he also enjoys bush/beach walking, classic music, gardening, and serving the community.