China Instrument and Control Society
Instrumentation
INSTRUMENTATION, Vol. 12, No. 3, September 2025 https://doi.org/10.15878/j.instr.202500283

Article

A High-quality Ellipse Detection Method for Machine Vision Based on Geometric Constraints and Hierarchical Clustering

Lin Zhang[1,2], Xuan Liu[1,2], Chen Zhang[3], Yuqing Hou[1,2], Xiaowei He[1,2], Sheng Tang[1,2,*]

[1] School of Electronic Information, Northwest University, Xi'an, China

[2] State-Province Joint Engineering and Research Center of Advanced Networking and Intelligent Information Services, Northwest University, Xi'an, China

[3] Institute of Photonics and Photonic Technology, Northwest University, Xi'an, China

[*] Corresponding author email: tangsheng@nwu.edu.cn

Abstract: In machine vision, elliptical targets frequently appear within the camera's region of interest (ROI). Ellipse detection is essential for shape detection and geometric measurements in machine vision. However, existing ellipse detection algorithms often face issues such as high computational complexity, strong dependence on initial conditions, sensitivity to noise, and lack of robustness to occlusions. In this paper, we propose a fast and robust ellipse detection method to address these challenges. This method first utilizes edge gradient and curvature information to segment the curve into circular arcs. Next, based on the convexity of the arcs, it divides them into different quadrants of the ellipse, groups and fits the arcs according to multiple geometric constraints at a low computational cost. Finally, it reduces the parameter space for hierarchical clustering and then segments the complete ellipse into several sectors for verification. We compare our method across seven datasets, including five public image datasets and two from industrial camera scenes. Experimental results show that our method achieves a precision ranging from 67.1% to 98.9%, a recall ranging from 48.1% to 92.9%, and an F-measure ranging from 58.0% to 95.8%. The average execution time per image ranges from 25 ms to 192 ms, demonstrating both high accuracy and efficiency.

Keywords: ellipse detection; geometric constraints; hierarchical clustering; camera datasets

Citation: Lin Zhang, Xuan Liu, Chen Zhang, Yuqing Hou, Xiaowei He, Sheng Tang."A High-quality Ellipse Detection Method for Machine Vision Based on Geometric Constraints and Hierarchical Clustering." Instrumentation 12, no.3 (September 2025). https://doi.org/10.15878/j.instr.202500283

1 Introduction

Ellipses are among the most common nonlinear geometric shapes in the real world, making ellipse detection techniques crucial in the fields of machine vision and image processing. These techniques can address a wide range of real-world problems, including industrial inspection, attitude measurement, camera calibration, medical research, biological cell division, traffic sign recognition, face recognition, and object tracking on robotic platforms[1-13]. However, there are still a few generalized solutions for ellipse detection in real-world images, which often feature complex backgrounds, motion blur, varying lighting conditions, occlusions, and noise. These challenges result in poor performance of existing methods, either in terms of detection accuracy, execution time, or both. Moreover, ellipses are not always fully visible and may be partially occluded or hidden by other objects, leading to incomplete ellipse curves in the image. In regions where the boundaries of two objects intersect, edge contours may be broken, further complicating the detection of elliptical geometry in real images.

In industrial vision systems, where objects are often

viewed from varying angles and under different lighting conditions, ellipses commonly appear incomplete or distorted. This makes reliable detection particularly difficult, and motivates our focus on robust methods tailored for such cases.

2 Related Work

Through literature research, we know that current ellipse detection methods are mainly divided into three categories: mathematical model-based methods, machine learning-based methods, and edge-link-based methods.

Mathematical model-based methods for ellipse detection typically use parametric equations of ellipses. The most classical ellipse fitting methods can be categorized into the Hough Transform (HT) and the Least Squares (LS) method. The Hough Transform operates in a 5D parameter space, utilizing a voting mechanism to identify ellipse parameters, which can be computationally and memory-intensive. To mitigate these issues, advancements have emerged, notably the Random Hough Transform (RHT) [14] that randomly selects subsets of edge points for each detection iteration. Tang and Srihari[15] further refined this approach with the Constrained Random Hough Transform (CRHT), enhancing accuracy by incorporating smoothing, distance, and curvature constraints. Additionally, the Probabilistic Hough Transform (PHT) introduced by N. Kiryati[16] reduced computational load by randomly sampling a subset of edge points for the transform. Addressing the limitations of these techniques, Guil and Zapata[17] introduced the Fast Elliptic Hough Transform (FEHT), adopting a novel focusing algorithm to replace the traditional polling process in a single parameter space. This innovation reduces noise sensitivity and computational overhead, albeit with limitations stemming from the extensive the overlooking of a large number of candidate votes, as well as the overlooking of geometric interdependencies among points during the voting phase. The Least Squares approach estimates ellipse parameters by aligning with edge points, yet it is notably susceptible to the initial parameter choices and the influence of noise and outliers, which can significantly skew the fitting outcomes. To counter the detrimental effects of outliers, several modified Least Squares techniques have been devised. For instance, the Iterative Reweighted Least Squares (IRLS) method[18] employs robust estimators or kernels to mitigate the impact of outliers. Liang et al. [19] introduced a constrained least squares fitting technique incorporating a maximum entropy criterion, specifically designed to tackle the outlier issue. Furthermore, A. Fitzgibbon[20] suggested integrating ellipsoid-shaped constraints into the least squares framework, enhancing robustness, especially against outliers. Although these methods have notably strengthened the ellipse fitting resilience, they are still constrained by high computational demands and time consumption, primarily due to the nonlinear optimization and iterative nature of the algorithms.

Machine learning-based methods detect ellipses by constructing theoretical models and loss functions for the observed points. For instance, Arellano and Dahyot[21] used the concept of point set alignment to detect ellipses with a Gaussian Mixture Model (GMM), named GMMED. Zhao et al. [22] also employed GMM to model ellipses and optimized the solution using a simulated annealing algorithm. With the rapid development of deep learning, convolutional neural network (CNN) -based methods have become more robust and efficient, especially in complex scenes. Dong et al. [24] developed the Ellipse R-CNN model for detecting ellipse-like shapes enriched with texture information. Li[25] created an anchor-point-based ellipse detector based on Faster R-CNN, achieving efficient detection in complex texture environments. However, this two-stage method is limited by specific object classes. Wang[26] proposed an effective generalized ellipse detector based on data augmentation without anchors, called ElDet. This method relies heavily on edge information and uses three loss functions to collectively enhance the CNN model's performance in detecting complete ellipses.

Nevertheless, the model tends to be biased when detecting ellipses that are significantly larger or smaller than the overall image. These methods have undergone extensive data-driven training, making machine learning-based ellipse detection useful for specific object classes. However, for industrial ellipse detection, edge information is preferable as it more accurately describes the underlying geometry.

Edge linking methods leverage continuous edge data from images to identify ellipses. Liu et al.[27] introduced a hierarchical strategy grounded in the idea that any ellipse segment can self-identify in reconstruction through geometric principles, decreasing computation and storage demands but still grappling with missed detections and false alarms. Prasad et al.[28] refined the search space for arc grouping by incorporating edge curvature and arc convexity. Fornaciari et al. [8] devised yet another ellipse detector(YAED), which filters arcs into ellipses based on geometric constraints and estimates parameters by segmenting the parameter space. Jia et al. [29] built upon YAED with CNED, a method enhancing projective invariance to eliminate unproductive ellipse candidates. Dong[30] enhanced YAED's grouping strategy, addressing three-arc ellipse detection and implementing a geometrical feedback system to refine detections, minimizing errors. Patraucean et al.[31] introduced ELSD, a parameter-free approach for line segments and elliptical arcs, utilizing LSD for iterative line segment search and concurrent detection/grouping. Lu et al. [1] presented an LSD-based method for ellipse detection, which constructs arc support groups from line segments and enhances accuracy and efficiency through local and global search strategies. Meng[32] developed a circular arc adjacency

matrix technique, leveraging curvature and region constraints to sparsify the matrix, echoing Prasad's[28] work. Zhao[33] built upon Jia's[29] concept of projective invariance, proposing coherent chord computation and crossover ratios for precise ellipse detection. Zhou[34] devised a top-down fitting strategy, overcoming Fornaciari's[8] limitations by detecting small, flat ellipses.

Model-based ellipse detection encounters real-time hurdles and is sensitive to initialization, whereas machine learning methods rely on extensive training data. Among them, edge segment detection stands out as an efficient strategy for multiple ellipse recognition in digital images, with lower computational overheads. Edge-chain methods strive to group arcs of the same ellipse but are prone to errors in arc detection, particularly in intricate or occluded images. In this work, we introduce a novel high-performance ellipse detection approach tailored for machine vision, integrating geometric constraints and hierarchical clustering. Our technique smartly pairs arcs based on geometric features like smoothness, curve positioning, and ellipse center confidence, subsequently estimating parameters through voting in a decomposed space. Section 2 reviews prevalent ellipse detection techniques. Section 3 outlines our proposed method. Section 4 examines parameter impacts and benchmarks our approach against others. Finally, Section 5 encapsulates the key contributions of this paper.

3 Description of the Methodology

Arcs are extracted from the edge poll and categorized into four classes based on the concavity of the arcs. We classify the edge pixels according to their gradient phase in the two main directions and group eight connected edge pixels in the same direction to form arcs. Their quality is also improved by removing short arcs or straight lines, and then the arcs are grouped according to their concavity and convexity, and the ellipses are fitted robustly and efficiently. Each step of the proposed ellipse detection method's algorithm is described in detail in the following sub-sections.

3.1 Edge-Weighted Fusion

Circular arc detection requires high accuracy due to the significant variations in shape and curvature. Different edge detection operators emphasize distinct features such as gradient and curvature, making it challenging to achieve consistent results. To address this, we propose an algorithm that fuses the outputs of various edge detection operators by assigning weights to each pixel position. This fusion algorithm enhances overall accuracy, produces more continuous edges, and mitigates the impact of noise. Additionally, it minimizes the dependency on the parameters of individual operators and can detect edges efficiently across different scales. Consequently, the algorithm is more comprehensive and robust.

By fusing this information, multiple features can be combined to improve the discrimination of circular arcs. Firstly, two edge images will be obtained from the input image using a Canny edge detector with automatic thresholding and the Sobel operator. Then, the gradients of the edge images and edge points will be fused using mean weighting, as shown in Figure 1. The gradient information derived from this weighted fusion is more coherent and smoother compared to that obtained from a single operator. This coherence facilitates faster and more accurate subsequent filtering and connection of circular arcs based on the gradient information. We can get the weighted edge image elements.

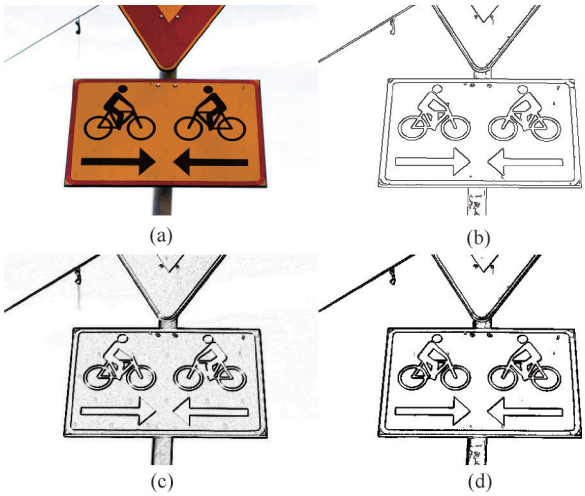

Fig.1 (a) Original image (b) Canny operator detects edges (c) Sobel operator detecting edges (d) Mean-weighted fused edges

To efficiently cut invalid arc combinations for post processing, we group arc segments into four sets corresponding to the arcs of an ellipse distributed in four quadrants$ArcI$、$ArcII$ 、$ArcIII$、$ArcIV$, which are called arc quadrant sets. In the preprocessing step, we separate edges pixels$p_i$according to the direction of the edge gradientτ_i, as the first stage of this arc grouping, and the gradient sign function$\mathrm{T}(p_i)$ is defined as equation (1).

$$\mathrm{T}(p_i)=\begin{cases}+, if \tan(\tau_i)>0\\ -, if \tan(\tau_i)<0\end{cases} \tag{1}$$

We define$D(C_T)$ as the direction of an arc located in a quadrant. Thus, arcs with positive gradient directions are located in the first or third quadrant $D(C_T)\in\{ArcI\cup ArcIII\}$, while arcs with negative gradient directions are located in the second or fourth quadrant $D(C_T)\in\{ArcII\cup ArcIV\}$, as shown in Figure 2. A property that will be used again for grouping arcs in a subsequent procedure.

3.2 Arc Segmentation Grouping

In this stage, we extract arcs from images segmented into quadrants, and further process the edge images to obtain smoother and more accurate results. First, we use the 8-connected region algorithm to extract continuous

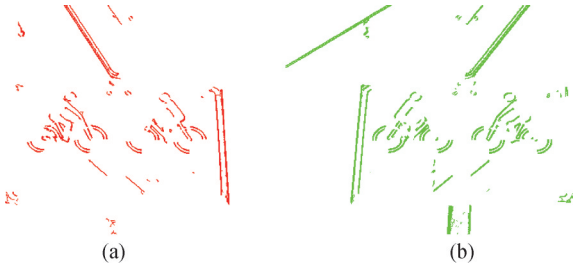

Fig.2 (a) curves at *Arc I* and *Arc III*
(b) curves at *Arc II* and *Arc IV*

edge contours. Next, we select contours with smooth curvature and exclude edge contours shorter than 12 pixels, as they may be influenced by noise or lack the curvature necessary for the ellipse detection process. Finally, we filter the arcs based on edge direction and convexity to identify those that are likely elliptical, grouping them into arcs, which are then classified.

3.2.1 Extraction of Elliptical Arcs

Through the above processing, we obtain a set of continuous curve edges with high geometric similarity to elliptical arcs, and the extracted curves are given by equation (2).

$$C_i = \begin{bmatrix} x_0 & x_1 & \dots & x_n \\ y_0 & y_1 & \dots & y_n \end{bmatrix} \tag{2}$$

Where(x_i, y_i) denotes the coordinates of each point and the curve consists of by *n*points .

After getting the curve, we can calculate the curvature of each curve point by point using equation (3). The curvature reflects the degree of curvature of the curve at each point and helps to determine the local shape of the curve.

$$K_i = \frac{\left|\dot{x}_i \ddot{y}_i - \ddot{x}_i \dot{y}_i\right|}{\left(\dot{x}_i^{\,2} + \dot{y}_i^{\,2}\right)^{3/2}} \tag{3}$$

Here x_i, y_i denote the horizontal and vertical coordinates of the pixel on the curve *C* respectively.

To simplify the curvature calculation, we use the center difference method to approximate the derivatives. The specific formulas for the first and second-order derivatives of x_i are as equation (4).

$$\begin{cases} \dot{x}_i = \left[x_{i+1} - x_{i-1}\right]/2 \\ \ddot{x}_i = \left[x_{i+2} - 2x_i + x_{i-2}\right]/4 \end{cases} \tag{4}$$

After calculating the curvature, we connect edge pixels with similar curvature to form continuous curves. We then classify these contours into closed and non-closed contours based on the distance between their endpoints. Closed contours are considered highly indicative of elliptical shapes, and thus, we prioritize selecting and fitting these closed edges. If a candidate ellipse passes validation, it is deemed the final detection result. If not, the same processing steps are applied to the remaining edges. This methodology ensures highly accurate and robust ellipse detection, thereby enhancing the overall reliability of the detection process.

Due to the inherent smoothness of curvature changes in elliptical arcs, we achieve smooth arc segments by identifying and segmenting key pixels, including turning points and inflection points. As illustrated in Fig. 3, turning points are characterized by significant changes in curvature, whereas inflection points mark locations where the direction of curvature reverses. Upon identification of these key points, the curve is represented as a series of line segments. This approach involves decomposing the original curve into smaller, more manageable segments, which are subsequently reconnected through the key points. This method ensures the generation of a continuous and smooth arc that closely approximates the original elliptical arc, represented as an arc *C* consisting of a series of line segments$\{l_1, l_2, ..., l_n\}$

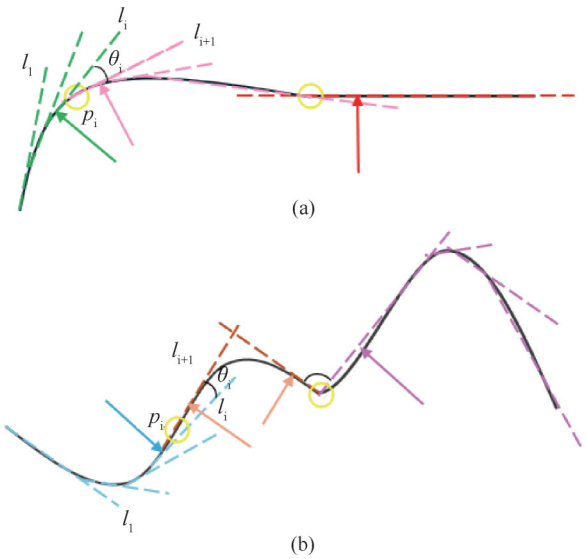

Fig.3 (a) Curves with large changes in curvature (b) curves with large changes in curvature direction. Specifically, *l* represents the line segments of the fitted curve ($i = 1, 2, 3\ldots$), $\theta_i$denotes the angle of intersection of the line segmentsl_i and l_{i+1}. $p_i$can be seen as a corner in (a) or a point of inflection in (b)

The RDP algorithm[35] is employed to approximate the curve *C* with a set of line segments *C*:$\{l_1, l_2, ..., l_n\}$, leveraging a tolerance threshold $T_t$to mitigate computational expenses. This approximation enables subsequent analyses to focus solely on the endpoints of these line segments, rather than encompassing all constituent pixels. The vector angle $\theta_1, \theta_2, \cdots, \theta_{m-1}$, defined between consecutive line segment pairs $l_i \sim l_{i+1}$, assumes values within the interval $[-\pi, \pi]$. To assess the angular variation between the initial angle A and a subsequent angle B exhibiting a notable change, a threshold T_θ is established. The identification of key pixels along the curve *C*is then contingent upon the satisfaction of specific constraints, thereby determining the critical pixels based on these prescribed conditions $\left|\theta_1 - \theta_j\right| > T_\theta$.

When the first angle θ_1 and the i^{th} angle θ_i have the same sign but are very different, this indicates that the

pixel is at the corner. And when the sign of the i^{th} angle $\theta_1$changes with respect toθ_i , it indicates that the direction of the curvature has changed and these points are the inflection points. Algorithm 1 splits the curve into arc segments by recognizing these corners and inflection points. A new set of consecutive edges is generated and edge points near the corners are removed to obtain purer elliptical arcs.

Algorithm 1 Arc selection

Input: $\mathbf{T}_\theta$, curves: C, curvature threshold: $\mathbf{T}_k$
Output: the arc vector: S

Initialize arc vector $S = \emptyset$
repeat
  Fit each curve C_i by line segments, store in L;
  Calculate angles θ formed by consecutive line segments in L;
  # Store endpoints and their curvature in P;
  $P \leftarrow \{(C_i[1], K_i[1])\}$
  repeat
    if $|\theta_1 - \theta_j| > \mathbf{T}_\theta$ **then**
      for m = 1 to the length of C_i **do**
        if $(L[j].x, L[j].y) = (C_i[m].x, C_i[m].y)$ **then**
          $P \leftarrow P \cup \{(C_i[m], K_i[m])\}$
        end if
      end for
    end if
  until traversed every line in L
  $P \leftarrow P \cup \{(C_i[-1], K_i[-1])\}$
  repeat
    for k = 2 to the size of P **do**
      if $|P[k][1] - P[k-1][1]| > \mathbf{T}_k$ **then**
        Reserve $C_i[k]$ in $S[k]$.
      end if
    end for
  until traversed the set P
until traversed every curve C_i

In Algorithm 1, we will also check if the curvature changes. If the curvature remains constant or changes within a given threshold, the entire curve is treated as a straight segment and discarded; the remaining portion of the curve, they are treated as smooth arc segments, which could potentially represent the edges of an ellipse or other circular structure, and the algorithm stores the curvature information of the two endpoints of these segments for subsequent processing. This information is crucial in the following steps as it will be used to categorize further whether these arc segments are convex or concave arcs.

3.2.2 Arc Convexity Classification

When dealing further with smooth circular arc segments, they are indeed divided into two categories based on their concavity: convex and concave arcs. During the process of arc segmentation, we obtained the curvature information $K_i[1]$ at the starting point and the curvature information $K_i[-1]$ at the endpoint of the arc. The concavity or convexity of the arc is determined by the curvature at these two endpoints. According to equation (5), if the curvature of both endpoints of a curve is positive, then the curve is concave; if the curvature of both endpoints is negative, then the curve is convex.

$$\Omega(C) = \begin{cases} -, if(K_i[1] \& K_i[-1]) > 0 \\ +, if(K_i[1] \& K_i[-1]) < 0 \end{cases} \tag{5}$$

If the curvature of the two endpoints is either positive or negative, or both are zero, the arc C cannot be unambiguously classified as a convex or concave arc in the conventional sense, and will be screened out of the subsequent geometrical analysis or identification process. Based on $\mathrm{T}(p_i)$ and the above $\Omega(C)$, each arc C can be divided into the following quadrants, shown as equation (6).

$$\Gamma(C) = \begin{cases} ArcI, \langle \mathrm{T}(p_i), \Omega(C) \rangle = \langle +, + \rangle \\ ArcII, \langle \mathrm{T}(p_i), \Omega(C) \rangle = \langle -, + \rangle \\ ArcIII, \langle \mathrm{T}(p_i), \Omega(C) \rangle = \langle -, - \rangle \\ ArcIV, \langle \mathrm{T}(p_i), \Omega(C) \rangle = \langle +, - \rangle \end{cases} \tag{6}$$

where Γ denotes the quadrant in which the arc is located. For example, consider an arc belonging to $\{A\mathrm{rc}I \cup A\mathrm{rc}III\}$, which is classified into the third quadrant ($A\mathrm{rc}III$) if the curvature of the two endpoints is positive according to the edge point p_i, otherwise (it falls in the first quadrant ($A\mathrm{rc}I$) . Thus, according to equation (6), when these arc segments are part of an ellipse, then they can be divided into regions similar to the four quadrants based on their position on the ellipse.

3.2.3 Grouping of Arcs Based on Geometric Constraints

Based on the above arc categorization, we select two arcs $\varepsilon_{\mathrm{ab}} = (C_a, C_b)$ as quadrant constraint arc sets from the arcs distributed in four quadrants, indicating that the arcs may belong to the same ellipse. Accordingly, the six combinations of the four quadrant constraint arc sets are ($A\mathrm{rc}I, A\mathrm{rc}II$)($A\mathrm{rc}I, A\mathrm{rc}III$) ($A\mathrm{rc}I, A\mathrm{rc}IV$)($A\mathrm{rc}II, A\mathrm{rc}III$) ($A\mathrm{rc}II, A\mathrm{rc}IV$) and ($A\mathrm{rc}III, A\mathrm{rc}IV$), as shown in Figure 4.

Subsequently, we employ a grouping method based on the relative positions of arcs [30]. As illustrated in Figure 4, we use the constraint conditions described by equation (7) to define the relative positions between two eligible arcs, thereby discarding arc segments that cannot form the same ellipse.

$$\begin{cases} C_a^L(x) > C_b^R(x) & if(C_a, C_b) \in (ArcI, A\mathrm{rc}II) \\ C_a^R(y) > C_b^L(y) \& C_a^R(x) > C_c^R(x) & if(C_a, C_b) \in (A\mathrm{rc}I, A\mathrm{rc}III) \\ C_a^R(y) > C_{\mathrm{b}}^L(\mathrm{y}) & if(C_a, C_b) \in (A\mathrm{rc}I, A\mathrm{rc}IV) \\ C_{\mathrm{a}}^L(\mathrm{y}) > C_{\mathrm{b}}^L(y) & if(C_a, C_b) \in (A\mathrm{rc}II, A\mathrm{rc}III) \\ C_{\mathrm{a}}^L(\mathrm{y}) > C_{\mathrm{b}}^R(y) \& C_b^L(x) > C_{\mathrm{a}}^R(x) & if(C_a, C_b) \in (A\mathrm{rc}II, A\mathrm{rc}IV) \\ C_a^L(x) > C_b^R(x) & if(C_a, C_b) \in (A\mathrm{rc}III, A\mathrm{rc}IV) \end{cases} \tag{7}$$

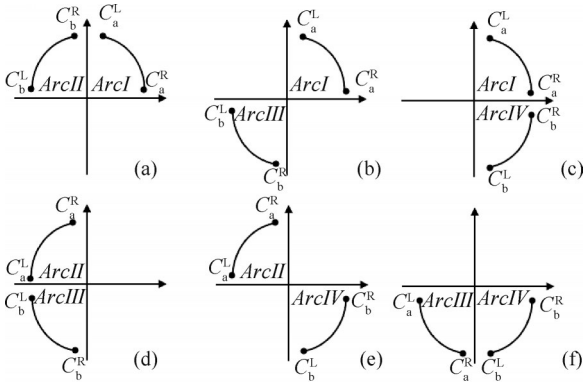

Fig.4 The six grouping stages. Specifically, *ArcI*, *ArcII*, *ArcIII*, *ArcIV* represents the four quadrants where the arcs are located. where $C_i^L(x)$ and $C_i^L(y)$ denote the value of the left endpoint of the arc, $C_i^R(x)$ and $C_i^R(y)$denote the value of the left endpoint of the arc, respectively.

We define the set of two arcs that meet specific conditions of a candidate ellipse as $\varepsilon_{ab}=(C_a, C_b)$, as these two arcs may belong to the same ellipse. Additionally, the positions of the centers of two arc segments can serve as a significant constraint for selecting potential sets of arc segments. If the centers of two arc segments are located within a specific range in the same region, these two arc segments likely belong to the same ellipse.

3.3 Initial Elliptic Set Formation

3.3.1 Determination of tthe Center of the Ellipse

A classic and effective method for determining the center of an ellipse is based on the property of the midpoints of parallel chords. This method leverages a crucial geometric characteristic of ellipses: the lines connecting the midpoints of any two parallel chords intersect at the center of the ellipse. As illustrated in Figure 5, by calculating multiple such lines and finding their intersection, the center of the ellipse can be approximately determined. However, this method may encounter challenges when dealing with short arc segments. When the arc segment length is very small, the distance between the nearest parallel chords formed by the segment itself also decreases, which can result in significant errors in the calculated center position. Additionally, when using the tangents at the endpoints of the arc segments to aid in determining the center, the slopes of these tangents may be affected by image noise and the digitization process, further reducing the accuracy of the detection.

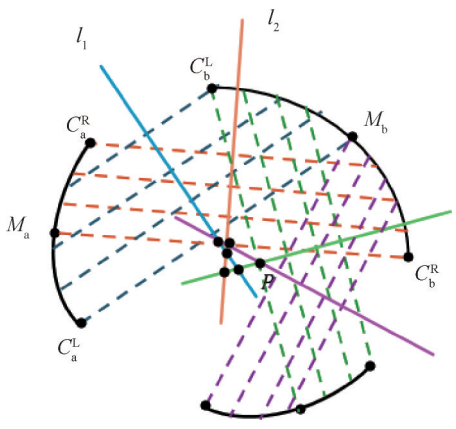

Fig.5 Parallel chord midpoints to determine the center of an ellipse

The center of the ellipse is the first and most critical parameter to calculate, as it serves as the key point for localization. To enhance the robustness and efficiency of this calculation, as shown in Figure 6, we first determine the midpoints of arcs C_a and C_b, denoted as M_a and M_b, respectively. The four endpoints of the pair of arcs obtained in the previous steps are designated as C_a^R, C_a^L, C_b^R, and C_b^L. Subsequently, tangents are drawn through these four endpoints and the two midpoints of the elliptical arcs. On both sides of the arcs, these tangents intersect in pairs at points P_1, P_2, P_3, and P_4. We further construct two line segments: a vertical line $l_1$passing through the intersection point P_1 and midpoint m_1, where m_1 is the midpoint of the line segment connecting the endpoint C_a^R and the midpoint M_a of arc C_a; and another vertical line l_3 passing through the intersection point P_3 and midpoint m_3, where m_3 is the midpoint of the line segment connecting the endpoint C_b^R and the midpoint M_b of arc C_b. On a pair of arcs, the straight lines l_i and l_j must pass through the center of the ellipse[37], where i=1,2;j=3,4.

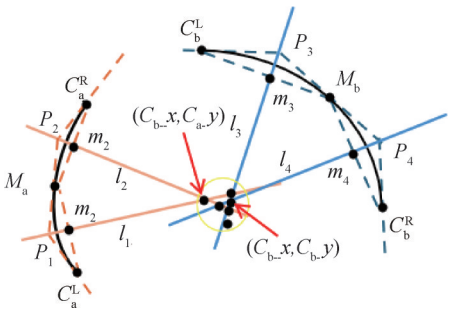

Fig.6 Ellipse center calculation method

In the above procedure we know the coordinates and gradients of the arc endpoints and arc midpoints (x_i, y_i, τ_i), then the coordinates of the midpoints m_1 and m_2 that we use to construct the vertical line segments can be expressed by equation (8).

$$\begin{aligned} m_1.x &= \frac{C_a^L.x + M_a.x}{2}, m_1.y = \frac{C_a^L.y + M_a.y}{2} \\ m_2.x &= \frac{C_a^R.x + M_a.x}{2}, m_2.y = \frac{C_a^R.y + M_a.y}{2} \end{aligned} \tag{8}$$

The coordinates of the intersection points P_1 and P_2 of the three tangents to the arcC_a can be calculated by equation (9).

$$P_1.x = \frac{M_a.y - M_a.\tau \times M_a.x - C_a^L.y + C_a^L.\tau \times C_a^L.x}{C_a^L.\tau - M_a.\tau}$$

$$P_1y = \frac{M_a.\tau \times C_a^L.y - C_a^L.\tau \times M_a.y + C_a^L.\tau \times M_a.\tau(M_a.x - C_a^L.x)}{C_a^L.\tau - M_a.\tau}$$

$$P_2.x = \frac{M_a.y - M_a.\tau \times M_a.x - C_a^R.y + C_a^R.\tau \times C_a^R.x}{C_a^R.\tau - M_a.\tau}$$
$$P_2y = \frac{M_a.\tau \times C_a^R.y - C_a^R.\tau \times M_a.y + C_a^R.\tau \times M_a.\tau(M_a.x - C_a^R.x)}{C_a^R.\tau - M_a.\tau} \tag{9}$$

We use equation (10) to calculate the slopes q_1 and $q_3$of the line arc C_a from its two endpoints $C_a^L$and C_a^R respectively to its midpoint m .

$$q_1 = \frac{M_a.y - C_a^L.y}{M_a.x - C_a^L.x}, q_3 = \frac{M_a.y - C_a^R.y}{M_a.x - C_a^R.x} \tag{10}$$

Continuing the calculation gives the slopes q_2 and q_4 of the line l_1,l_2 through the arc C_a.

$$q_2 = \frac{P_1.y - m_1.y}{P_1.x - m_1.x}, q_4 = \frac{P_2.y - m_2.y}{P_2.x - m_2.x} \tag{11}$$

Thus, given an arc, the coordinates of the center $(C_a.x, C_a.y)$ can be derived according to the proposed method as equation (12).

$$C_a.x = \frac{m_2.y - q_4 \times m_2.x - m_1.y + q_2 \times m_1.x}{q_2 - q_4}$$
$$C_a.y = \frac{q_2 \times m_2.y - q_4 \times m_1.y + q_2 q_4 (m_2.x - m_1.x)}{q_2 - q_4} \tag{12}$$

From the above analysis, we can get a pair of arcs centers. We consider that the two arcs of the candidate ellipse $\varepsilon_{ab} = (C_a, C_b)$ can contain the same ellipse if and only if the distance between the centers of the two arcs $(C_a.x, C_a.y)$and$(C_b.x, C_b.y)$ exists for a given threshold T_c. Furthermore, the four constructed lines l_1 , l_2 , l_3 , and l_4 intersect in pairs at four points, any of which may be considered a potential center of the candidate ellipse. The average of these intersection point coordinates, along with the average of the centers of the pair of arcs, is also considered a potential center of the candidate ellipse. Consequently, for a pair of arcs, eight potential ellipse centers are calculated. In the next section, these intersection points and calculated centers will be used to estimate the other parameters of the ellipse.

3.3.2 Elliptic Parameter Estimation

Having determined the center of the ellipse, the next step is to compute the remaining parameters of the ellipse. To achieve this, we decompose the parameter space of the ellipse and compute the two parameters, *N*and *K*, as described in references[31,38]. Here, *N* represents the ratio of the minor axis length *B* to the major axis length *A*of the ellipse, and *K* is represented as $K = \tan\varphi$, where denotes the orientation of the ellipse.

The equation for the operator *N* represented by*K* is as equation (13).

$$N = \sqrt{\frac{(q_1 - K)(q_2 - K)}{(1 + q_1 K)(1 + q_2 K)}} \tag{13}$$

where q_1, q_2 is derived from the first pair of points on one arc, and is derived from another pair of points on another arc. Therefore, the equation (14).

$$K = \pm\sqrt{1 - \frac{\beta}{\alpha}} \tag{14}$$

Where$\alpha = q_1 q_2 - q_3 q_4$, $\beta = q_2 q_4 (q_3 - q_1) + q_1 q_3 (q_4 - q_2) + (q_1 + q_2 - q_3 - q_4)$

For the found pair of points, according to equation (13) successive updates of the aggregator *N*—*K*, we can get the values of the topmost points *N* and φ in the two 1D aggregators. In this step, the elliptic polar coordinate equation and the coordinate transformation relation are applied to convert the coordinates of a point on the arc of the circle from the world coordinate system (x_i, y_i) to the elliptic coordinate system (x_e, y_e).

In two-dimensional space, coordinate transformations are accomplished by equation (15).

$$\begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x_i - x_c \\ y_i - y_c \end{bmatrix} \tag{15}$$

From equations (14-15), we obtain the equations (16-17).

$$x_e = \frac{(x_i - x_c) + (y_i - y_c)K}{\sqrt{K^2 + 1}}$$
$$y_e = \frac{(y_i - y_c) - (x_i - x_c)K}{\sqrt{K^2 + 1}} \tag{16}$$

$$A_x = \sqrt{\frac{x_e^2 N^2 + y_e^2}{N^2 (K^2 + 1)}} \tag{17}$$

Finally, by defining the major axis *A* as $A = A_x / \cos\varphi$ and the minor axis *B* as $B = A \cdot N$, we obtain a set of parameters $(x_c, y_c, A, B, \varphi)$ for the candidate ellipse.

3.4 Elliptic Clustering

Due to the interference of external features, an ellipse may be segmented into two elliptical arcs, thus extracting two ellipses from the same object. Therefore, we need to cluster the results of multiple detections of the same ellipse. Considering the presence of duplicate points in the initial ellipse set, an effective clustering method to trim them is especially important, both to keep isolated points and to suppress non-extremely large points. To prevent data explosion, we first map the data (circle center, semiaxis, and direction) into three low-dimensional spaces for clustering. An iterative mean drift clustering method is used to identify acceptable cluster centers within the region of interest (ROI). For this purpose, we developed a hierarchical clustering methodthat incorporates the Gaussian mean shift. The method decomposes the 5D elliptic parameter space clustering problem into two 2D space (centers and semi-axes), and one 1D space (directions) clustering problem. The similarity of the two ellipses is evaluated by comparing the differences in the ellipse parameters. Specifically, an eigenvector is defined as $V_i = \{(x_i, y_i), (a_i, b_i), \varphi_i\}_{i-1}^{n}$,where (x_i, y_i) represents the center coordinates of an ellipse,(a_i, b_i) represents the long and short semi-axes, and φ_i is the rotation angle of the ellipse.

The specific clustering process is detailed in Algorithm 2. Initially, mean clustering is performed on the centers of the ellipses. After obtaining the cluster centers, the minimum Euclidean distance for each ellipse center in the initial set is calculated, identifying the ellipse center closest to the smallest cluster. Subsequently, clustering is performed on the semi-axes and rotation angles. Finally, the concatenation of these three parameters is considered to constitute a valid set of ellipses.

Algorithm 2 Ellipse Clustering

Input: Initial ellipse set: $\mathbf{E}_I$, Ellipse parameter: V_i
Output: Cluster ellipse set: $\mathbf{E}_C$, Ellipse number: N

Initialize set: $\mathbf{E}_C = \emptyset$
Cluster elliptic centers of $\mathbf{E}_I$ using Gaussian Mean Shift
Obtain N_c cluster centers $(x, y)_1, (x, y)_2, ..., (x, y)_{N_c}$
for each elliptic cluster center $(x, y)_j$ **do**
    Compute Euclidean distance $d_{\min}$ between $(x, y)_j$ and V_i
    if $d_{\min}$ equals the minimum distance found so far **then**
        Add V_i to set A_j
    end if
end for
Implement clustering based on semi-axes for each initial ellipse subset A_j
Obtain $N_{c,a}$ cluster centers $(a, b)_1, (a, b)_2, ..., (a, b)_{N_{c,a}}$
for each semi-axes cluster center $(a, b)_k$ **do**
    Compute Euclidean distance $d_{\min}$ between $(a, b)_k$ and V_i
    if $d_{\min}$ equals the minimum distance found so far **then**
        Add V_i to set $A_{j,k}$
    end if
end for
Implement clustering based on orientations for each initial ellipse subset $A_{j,k}$
Obtain $N_{c,a,\varphi}$ cluster centers $\varphi_1, \varphi_2, ..., \varphi_{N_{c,a,\varphi}}$
for each orientation cluster center φ_s **do**
    Compute Euclidean distance $d_{\min}$ between φ_s and V_i
    if $d_{\min}$ equals the minimum distance found so far **then**
        Add V_i to set $A_{j,k,s}$
    end if
end for
Combine all sets Aj, k, s into $\mathbf{E}_C$: $\mathbf{E}_C \leftarrow \bigcup_{j,k,s} A_{j,k,s}$
Ellipse count: $N \leftarrow N + |A_{j,k,s}|$

Numerous findings have shown[1] show that applying the clustering method to the initial set of ellipses generates purer candidate ellipses. However, the benefit of clustering after our ellipse parameter space decomposition is twofold. First, a hierarchical clustering method is employed to address the complexity of directly measuring the distance between two ellipses. Each ellipse parameter (center, orientation, semi-axes) is treated as an individual entity for clustering. This approach allows for a more detailed analysis of the distribution and relationships of ellipses, as the variation of one parameter does not directly affect the clustering of other parameters. Secondly, the hierarchical method significantly outperforms direct clustering in terms of speed and accuracy. The reduction in dimensionality and the independent processing of parameters contribute to this improvement, ensuring that the clustering results are both reliable and accurate.

3.5 Elliptic Validation

In the candidate ellipse fitting process described above, we cover almost all sets of ellipses in the image and reduce duplicate ellipses. However, some combinations may form false ellipses, so ellipse validation is necessary to improve the quality of detected ellipses. We use a validation method that combines sectorization and a voting mechanism to ensure the accuracy of ellipse fitting.

First, we divide the ellipse into several equal-sized sectors. Then, voting is performed based on these consecutive sectors to check the completeness of the ellipse. Specifically, we collect the center points of each sector on the circumference of the potential ellipse and control the gradient alignment of these points. If the center point on the sector satisfies the alignment condition, the changed sector is marked as valid. The collection and validation of valid sectors is performed in the following steps:

(1) Determination of Consecutive Sectors: We need to identify a candidate ellipse by at least three consecutive valid sectors.

(2) Voting mechanism: Each successive sequence of valid sectors provides a vote equal to the number of sectors in the sequence.

(3) Threshold setting: A predetermined threshold is set for the minimum number of votes, which determines the algorithm's tolerance for the occluded portion of the ellipse. A candidate ellipse can be labeled as a detected ellipse only if its number of votes exceeds this threshold.

During the validation process, all edge points contributing to the votes are removed from the edge segment to prevent double-counting. If the candidate's ellipse does not reach the preset threshold for the number of votes, the sampling process is restarted until the end condition is met. By this method, not only is the appearance of false ellipses is reduced, but also the purity and accuracy of the detection results are improved.

4 Experimental Results

In this section, we assess the performance of the proposed method concerning detection effectiveness and execution time. For evaluating detection effectiveness, we employ the method described in[40]. This method utilizes three metrics: (1) Precision, (2) Recall, and (3) F-measure, which are defined as equation (18-20).

$$Precision = \frac{\Psi}{\mathrm{N}} \tag{18}$$

$$Recall = \frac{\Psi}{\mathrm{M}} \tag{19}$$

$$F-measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{20}$$

Where Ψdenotes the number of correctly detected ellipses, Ndenotes the total number of detected ellipses, and Mdenotes the number of ground-true ellipses.

4.1 Data Set Selection

We conducted a comprehensive set of tests on both synthetic and real image datasets to evaluate the performance of the proposed method and compare it with other state-of-the-art methods. We used five publicly available datasets downloaded from the internet and two datasets created using a PCO Dicam PRO camera during the direct laser writing process to assess the performance of the proposed ellipse detection method. We publicly disclose these datasets tohttps://github. com/daixi028/Dataset-Ellispes.

Two of the natural datasets are Dataset Prasad[27] and Dataset #2[8]. Dataset Prasad consists of 198 images containing small ellipses. Dataset #2 comprises 629 frames of images captured by a mobile phone under varying lighting conditions from several video sequences.

The two industrial datasets are Dataset PCB[39] and Dataset Camera. Dataset PCB contains 100 industrial PCB images with concentric structures. The Dataset Camera consists of 100 images of complete or partial ellipses captured by our industrial camera under different lighting conditions.

The method was also tested on three synthetic datasets. The first two datasets consist of 600 images with elliptical occlusions[8] and 600 images with overlapping ellipses[40], respectively. Each image is 300 × 300 pixels in size and contains $n \in \{4, 8, 12, 16, 20.24\}$ ellipses. The last dataset includes 200 phase maps used during the imaging process to load onto a spatial light modulator to alter the beam phase, featuring complex concentric ellipses or internal structures.

The presentation of the above dataset information is shown in Table 1.

Table 1 Seven datasets used in the experiment

Dataset	Prasad	#2	PCB	Camera	Occlude	Overlap	Phase
Number of images	198	629	100	100	600	600	200
Resolution	360 × 360	640× 480	500 × 500	1280 × 1024	300 × 300	300 × 300	1080× 1080

We evaluate the performance of the proposed method by comparing it with six state-of-the-art elliptical detectors: Lu et al. [1], Jia et al. [29], Fornaciari et al. [8], Prasad et al. [27], ELSD[29], and Liu et al. [27]. The C++ or MATLAB source code for these methods is available online, and all methods were executed using their default parameters. All experiments were conducted on a PC equipped with 64 GB of RAM and an Intel Core i7 processor.

4.2 Threshold Determination

The performance of the ellipse detection method depends on the selection of threshold parameters, primarily T_t, T_θ, and T_c, as shown in Figure 7. However, it is not guaranteed that a single set of threshold parameters will achieve optimal ellipse detection performance across all images. Additionally, selecting appropriate control parameters for each image to establish an accurate mathematical model is quite challenging. Therefore, we use statistical data from the F-measure and execution time (*ms*) t to evaluate performance, adjusting one threshold parameter at a time while keeping the others fixed. These parameters were tested on the aforementioned two natural datasets and two industrial datasets.

Figures 7(a) and (d) illustrate the impact of threshold T_t on performance and execution time. When T_t exceeds 10, detection accuracy and efficiency decrease, while the execution time decreases with increasing T_t . We chose a compromise setting of $T_t = 10$ (with $T_\theta = 25$ and $T_c = 25$). As shown in Figures 7(b) and (e), the optimal value of threshold T_θ for detection efficiency is 35 (with $T_t = 10$ and $T_c = 25$), because the execution time decreases monotonically with increasing . Figures 7(c) and (f) show that the optimal values of threshold T_c for detection efficiency are 20 and 25, but the execution time for 25 is longer than for 20. Balancing accuracy and speed, we chose $T_c = 20$ (with $T_t = 10$ and $T_\theta = 35$). We consider other threshold parameters as inherent parameters, as they can be fixed empirically and do not significantly impact the detection process.

To further evaluate the sensitivity of the proposed method to key thresholds, we conducted controlled experiments on three synthetic datasets. We independently varied the RDP segmentation threshold T_t, the angle thresholdT_θ for arc grouping, and the ellipse center distance threshold T_c, while fixing the other two parameters. For Dataset Occlude and Dataset Overlap, due to their similar image resolution and target scale, the optimal performance was achieved with$T_t = 8$, $T_\theta = 25$, and $T_c = 20$. In contrast, for the more complex Dataset Phase, which includes diverse occlusion patterns and ellipse sizes, the best detection accuracy was obtained at$T_t = 10$, $T_\theta = 30$and $T_c = 30$. These results demonstrate the adaptability of our method to varying image characteristics while highlighting reasonable default settings for different scenarios.

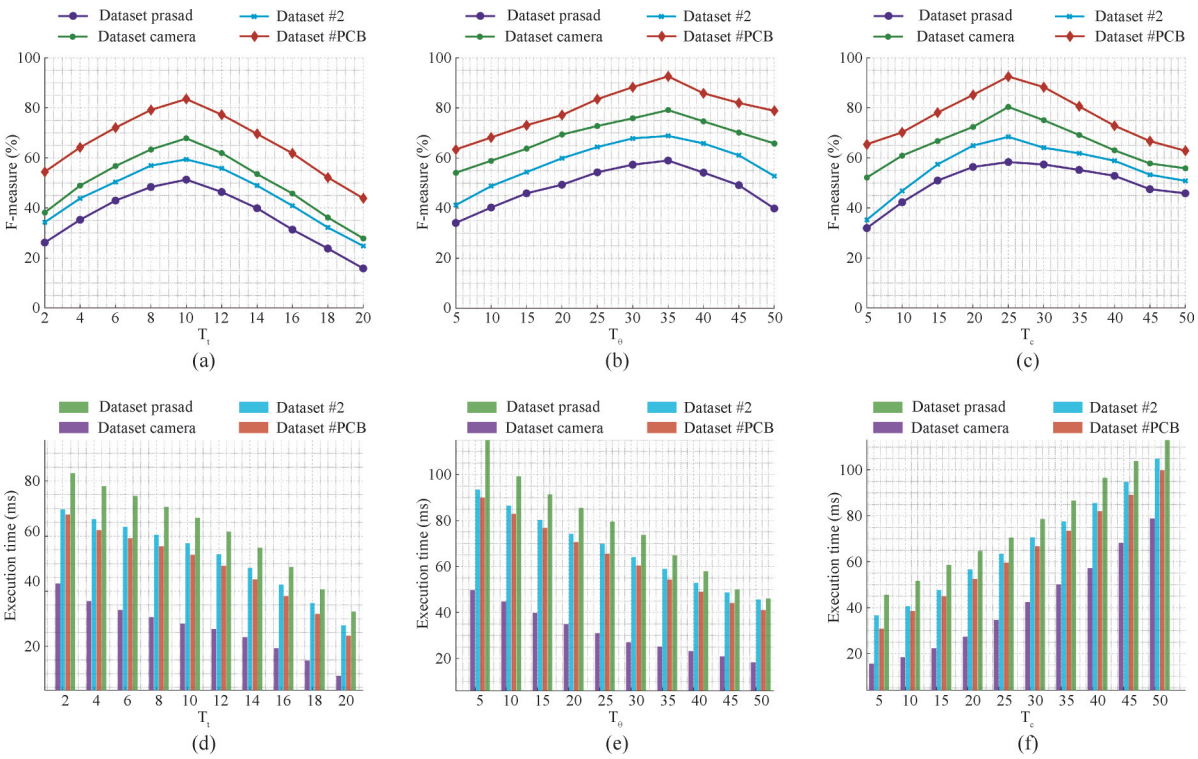

Fig.7 Threshold determinations. As the thresholds T_l, T_θ,and T_c are changed, (a-c)represent the relative F-measure and (d-f) represent the corresponding execution time (*ms*).

4.3 Experiment on the Dataset

Figure 8 presents the Precision, Recall, and F-measure of different algorithms across seven datasets, including two natural datasets, two industrial datasets, and three synthetic datasets. Table 2 displays the execution time of these methods on the same datasets.

In the two natural datasets, the methods proposed by Liu and Prasad not only demonstrate low accuracy but are also extremely time-consuming. 'Liu's hierarchical strategy, involving multiple detection stages, contributes to its long computation time. Prasad's method suffers from a brute-force strategy where arcs are matched against all possible ellipses in their vicinity, which greatly increases computational overhead. Although Lu's method enhances accuracy by building on ELSD, its execution time remains over 100 ms, which is unsuitable for real-time tasks. 'Fornaciari's and 'Jia's algorithms offer the shortest execution times due to efficient edge pruning and clustering strategies, but they fall short in accuracy compared to our method. Our algorithm achieves higher precision, recall, and F-measure, while maintaining competitive execution times, demonstrating a better balance between speed and accuracy.

Table 2 Execution time (*ms*) of different algorithms on two natural datasets, two industrial datasets and three synthetic dataset

	Liu[27]	Prasad[28]	Fornaciari[8]	Jia[29]	ELSD[31]	Lu[1]	our
Dataset Prasad	949.12	3958.20	12.61	8.42	1324.38	158.52	**25.31**
Dataset #2	4120.28	9351.70	47.25	32.75	4483.06	437.95	**55.05**
Dataset PCB	6171.20	1075.83	58.25	43.83	3625.61	169.53	**51.56**
Dataset Camera	9351.20	1843.43	70.84	61.52	2371.46	349.03	**63.17**
Dataset Occlude	257.34	5773.96	188.21	169.32	4572.73	2176.64	**162.48**
Dataset Overlap	282.48	6132.03	193.75	168.04	4860.97	2272.83	**178.27**
Dataset Phase	4379.75	6971.49	237.64	224.63	9500.15	5798.04	**191.96**

The visual results on the natural datasets further confirm this conclusion, as shown in Figure 9. 'Liu's method performs poorly due to its multi-stage detection pipeline, while 'Prasad's exhaustive arc matching leads to high time consumption and lower detection quality.

On the industrial datasets, specifically Dataset PCB and Dataset Camera, our method shows strong adaptability. For the PCB dataset, our results are comparable to those of 'Lu's method in terms of accuracy, but our approach requires significantly less computation

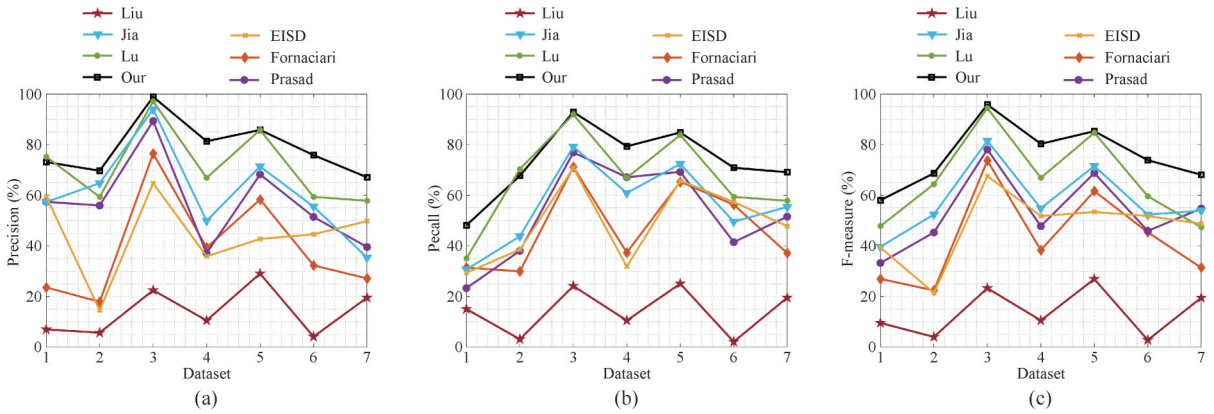

Fig.8 Precision, Recall and F-measure of different algorithms. From one to seven, they are Dataset Prasad, Dataset #2, Dataset PCB, Dataset Camera, Dataset Occlude, Dataset Overlap, Dataset Phase

Fig.9 Detection results in two natural datasets.

time. This efficiency is crucial in industrial environments where processing speed is often a key requirement. For the Dataset Camera, which includes many incomplete and noisy ellipses, our algorithm achieves the best overall performance, outperforming others in all three evaluation metrics.

The detection results on these two industrial datasets are shown in Figure 10. They demonstrate that 'Fornaciari's and 'Jia's methods struggle in these scenarios because their algorithms rely on detecting ellipses from arcs spanning three quadrants, making them unsuitable for the partial ellipse structures commonly seen in industrial images. While Lu's method can detect arcs, it may incorrectly filter out valid candidates during verification. Our method avoids these pitfalls, maintaining both accuracy and robustness under industrial constraints.

Regarding the synthetic datasets, our method once again demonstrates strong robustness, especially in handling occlusions. As illustrated in Figure 11, our method achieves the highest F-measure under occlusion conditions, showing its ability to accurately detect ellipses even when portions of the shape are missing. Although Lu's method slightly outperforms ours in overlapping ellipse scenarios, our method still performs competitively and offers much better execution speed. 'Fornaciari's and 'Jia's methods maintain high speed but exhibit significantly lower F-measure values, indicating a trade-off between speed and detection quality. ELSD and Prasad's methods perform reasonably well when ellipse density is low, but their performance deteriorates rapidly in more complex scenes. 'Liu's method shows poor

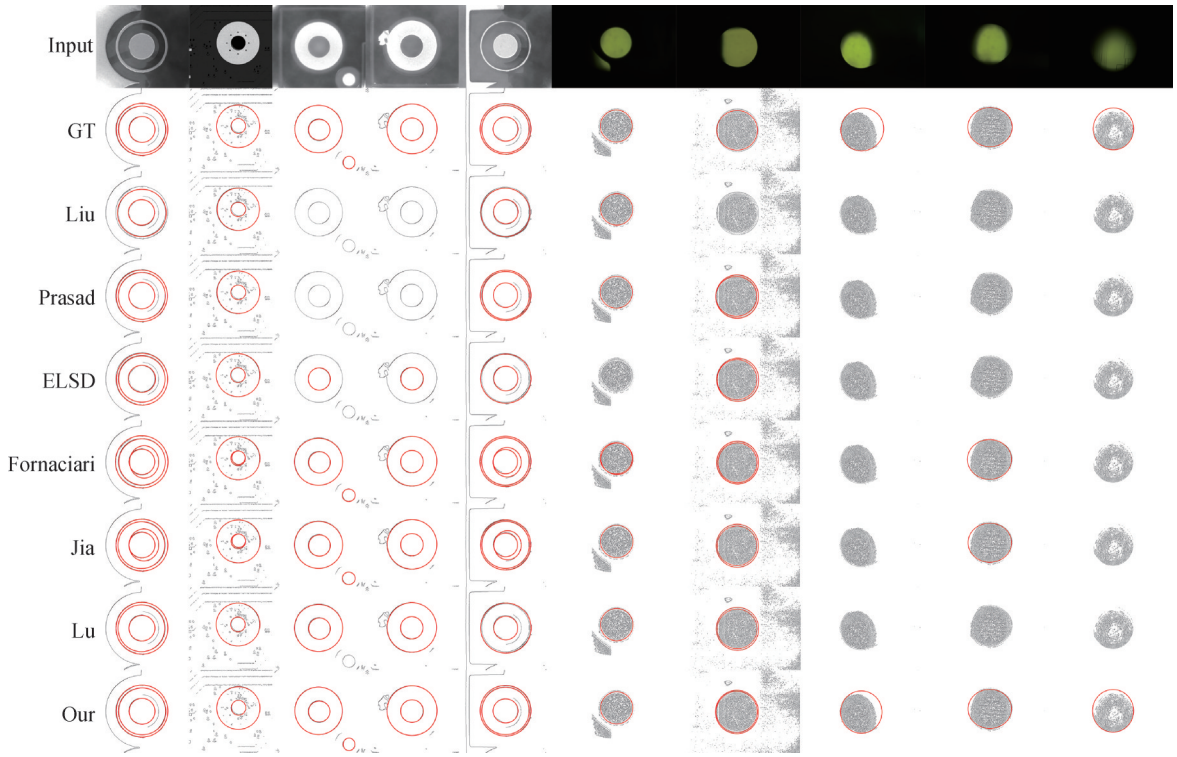

Fig.10 Detection results in two industrial datasets.

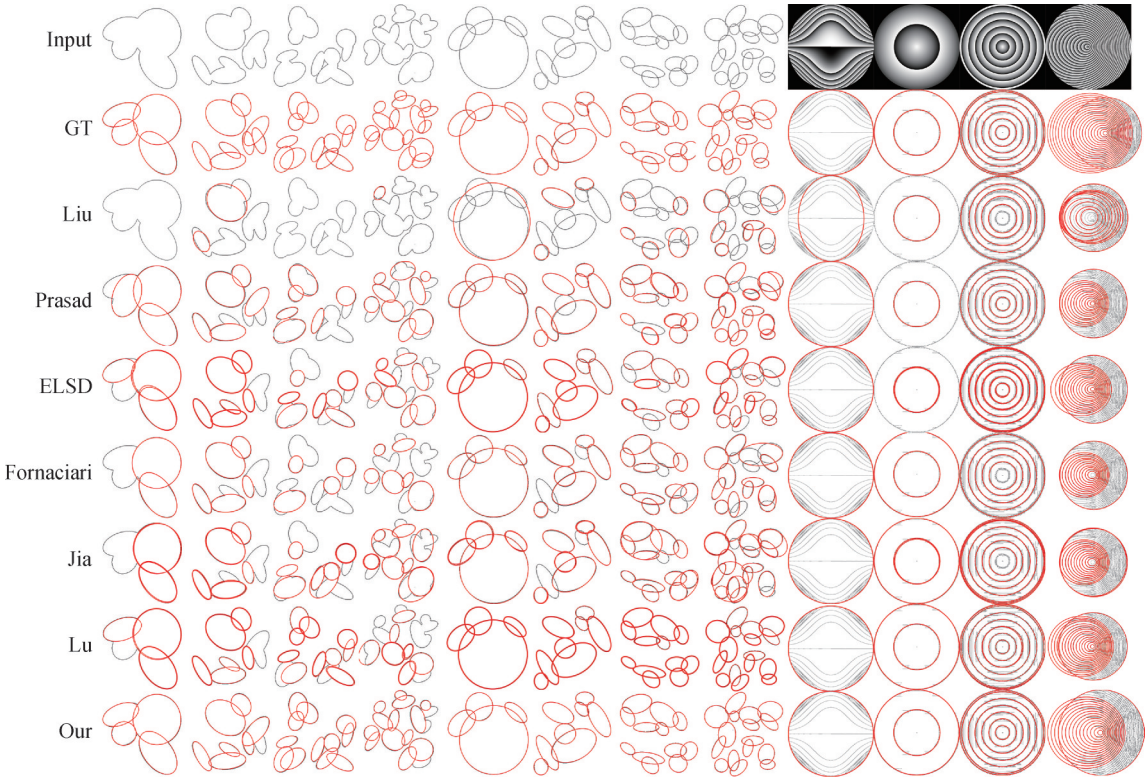

Fig.11 Detection results in three synthetic datasets.

performance in both accuracy and runtime. In concentric ellipse detection, most methods including ours, Fornaciari's, Jia's, and 'Lu's achieve good results. However, Lu and Fornaciari tend to miss some ellipses, and 'Jia's clustering strategy occasionally produces redundant detections.

Overall, our algorithm demonstrates superior generalization and robustness across different types of datasets. It is capable of handling diverse and complex scenes, including real-world industrial data and synthetic images with varying degrees of difficulty. These results highlight the effectiveness of our arc grouping strategy

and verification mechanism, making our approach well-suited for both accuracy-critical and time-sensitive applications.

5 Conclusion

This paper investigates the problem of ellipse detection in various scenarios and images captured by industrial cameras. Specifically, we propose an ellipse detection method based on geometric constraints and hierarchical clustering, which achieves high detection efficiency and low execution time. The balance between low execution time and detection efficiency is achieved through (a) effective pruning of curves that aid in ellipse detection, (b) intelligent grouping of arcs that are likely to form ellipses, (c) robust utilization of precomputed gradient information across multiple steps, and (d) hierarchical clustering of the five-dimensional ellipse parameter space. Moreover, stringent ellipse verification ensures high localization accuracy and robustness while rejecting false positives. Moreover, rigorous ellipse verification reduces false positives and improves the purity and accuracy of detected ellipses.

We conducted extensive experiments, fine-tuning and validating method parameters to achieve optimal performance. Compared with six existing methods, our approach demonstrates significant advantages in both detection efficiency and execution time, successfully detecting ellipses with varying degrees of incompleteness in images captured by industrial cameras. Future work will focus on further enhancing the detection capability of severely occluded ellipses and well-shaped semi-ellipses.

Author Contribution:

Lin Zhang: Conceptualization, Methodology, Software, Data curation, Formal analysis, Visualization, Writing original draft, Writing review & editing. Xuan Liu: Investigation, Data curation, Software, Validation, Writing – review & editing.Chen Zhang: Formal analysis, Visualization, Writing – review & editing. Yuqing Hou: Data curation, Software, Validation. Xiaowei He: Resources, Project administration. Sheng Tang: Conceptualization, Supervision, Methodology, Project administration, Funding acquisition, Writing – review & editing.

Funding Information:

This paper is supported by National Major Scientific Research Instrument Development Project of China (No. 51927804) and Science Fund for Shaanxi Provincial Department of Education's Youth Innovation Team Research Plan under Grant (No. 23JP169).

Data Availability:

The related dataset of this paper can be accessed from the following link:

https://github.com/daixi028/Dataset-Ellispes

Conflicts of Interest:

The authors declare no competing interests.

Dates:

Received 13 March 2025; Accepted 25 August 2025; Published online 30 September 2025

References

[1] Lu C, Xia S, Shao Met al. (**2019**) Arc-support line segments revisited: an efficient and high-quality ellipse detection. *IEEE Trans. Image Process* 29:769-781.

[2] Meng C, Xue J, Hu Z. (**2015**) Monocular position-pose measurement based on circular and linear features. *2015 Int. Conf. Digit. Image Comput. Techn. Appl. (DICTA) 1* -8.

[3] Chen X D, Qian C, Zhao Met al. (**2024**) Improving ellipse fitting via multi-scale smoothing and key-point searching. *Pattern Recognition* 151: 110432

[4] Tao L, Xia R, Zhao Jet al. (**2023**) A high-accuracy circular hole measurement method based on multi -camera system. *Measurement* 207: 112361

[5] Zafari S, Eerola T, Sampo Jet al. (**2015**) Segmentation of Overlapping Elliptical Objects in Silhouette Images. *IEEE Trans. Image Process* 24: 5942-5952.

[6] Garcés Y, Guerrero A, Hidalgo Pet al. (**2016**) Automatic detection and measurement of viral replication compartments by ellipse adjustment. *Sci. Rep* 6: 36505.

[7] Abeyrathna D, Life T, Rauniyar Set al. (**2021**) Segmentation of Bacterial Cells in Biofilms Using an Overlapped Ellipse Fitting Technique. *2021 IEEE Int. Conf. Bioinform. Biomed (BIBM) 3548* -3554.

[8] Fornaciari M, Prati A, Cucchiara R.(**2014**) A fast and effective ellipse detector for embedded vision applications. *Pattern Recognition* 47: 3693-3708.

[9] Li H, Lin Z, Shen Xet al. (**2015**) A convolutional neural network cascade for face detection. *IEEE Conf. Comput. Vis. Pattern Recognit(CVPR) 5325* -5334.

[10] Zhu X, Ramanan D. (**2012**) Face detection, pose estimation, and landmark localization in the wild. *IEEE Conf. Comput. Vis. Pattern Recognit(CVPR)* . 2879-2886.

[11] Dong H, Sun G, Pang W C, et al.(**2018**) Fast ellipse detection via gradient information for robotic manipulation of cylindrical objects. *IEEE Robotics and Automation Letters 2754*-2761.

[12] Salehian S S M, Khoramshahi M, Billard.(**2016**) A dynamical system approach for softly catching a flying object: theory and experiment. *IEEE Trans. Rob* 32: 462-471.

[13] Ono K, Ogawa T, Maeda Yet al.(**2014**) Detection, localization and picking up of coil springs from a pile. *2014 IEEE Int. Conf. Robot. Autom 3477*-3482.

[14] McLaughlin R A. (**1998**) Randomized hough transform: improved ellipse detection with comparison. *Pattern Recognition Letters* 19: 299-305.

[15] Tang Y, Srihari S N. (**2011**) Ellipse detection using sampling

constraints. *2011 IEEE Int. Conf. on Image Process 1045*- 1048.

[16] Kiryati N, Eldar Y, Bruckstein A M. (**1991**) A probabilistic hough transform. *Pattern Recognit* 24: 303-316.

[17] Guil N, Zapata E. (**1997**) Lower order circle and ellipse Hough transform. *Pattern Recognit* 30: 1729-1744.

[18] Zhang Z (**1997**) Parameter estimation techniques: A tutorial with application to conic fitting. Image Vis. *Comput* 15: 59-76.

[19] Liang J, Wang Y, Zeng X. (**2015**) Robust ellipse fitting via half-quadratic and semidefinite relaxation optimization. *IEEE Trans. Image Process* 24: 4276-4286.

[20] Fitzgibbon A, Pilu M, Fisher R B. (**1999**) Direct least square fitting of ellipses. *IEEE Trans. Pattern Anal. Mach. Intell* 21: 476-80.

[21] Arellano C, Dahyot R. (**2016**) Robust ellipse detection with Gaussian mixture models. *Pattern Recognition* 58: 12-26.

[22] Zhao M. et al. (**2021**) Robust ellipse fitting using hierarchical Gaussian mixture models. *IEEE Trans. Image Process* 30: 3828-3843.

[23] Jian B, Vemuri B C. (**2011**) Robust point set registration using Gaussian mixture models. *IEEE Trans. Pattern Anal. Mach. Intell* 33: 1633-1645.

[24] Dong W, Roy P, Peng C, Isler V (**2021**) Ellipse R-CNN: Learning to infer elliptical object from clustering and cclusion. IEEE Trans. *Image Process* 30:2193-2206.

[25] Li Y. (**2019**) Detecting lesion bounding ellipses with gaussian proposal networks. *International Workshop on Machine Learning in Medical Imaging(MLMI) 337* -344.

[26] Wang T, Lu C, Shao Met al. (**2022**) An anchorfree general ellipse object detector. *Asian Conf. Comput. Vis 2580* -2595.

[27] Liu Z Y, Qiao H. (**2009**) Multiple ellipses detection in noisy environments: a hierarchical approach. *Pattern Recognit* 42: 2421-2433.

[28] Prasad D Ket al. (**2012**) Edge curvature and convexity based ellipse detection method. *Pattern Recognit* 45: 3204-3221.

[29] Jia Q, Fan X, Luo Zet al. (**2017**) A fast ellipse detectorusing projective invariant pruning. *IEEE Trans. Image Process* 26: 3665-3679.

[30] Dong H, Prasad D K, Chen I M. (**2018**) Accurate detection of ellipses with false detection control at video rates using a gradient analysis. *Pattern Recognit* 81: 112-130.

[31] Pătrăucean V, Gurdjos P, Von Gioi R G. (**2012**) A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. *2012 Eur. Conf. Comput* 12: 572-585

[32] Cai M, Zhao X, Xiang Z, Fu G. (**2020**) Arc adjacency matrix-based fast ellipse detection. *IEEE Trans. Image Process* 29: 4406-4420.

[33] Zhao M, Jia X, Ma L, Hu L, Yan D . (**2024**). Coherent chord computation and cross ratio for accurate ellipse detection. *Pattern Recognition* 146: 109983.

[34] Zhou H, Han L, Zhu S,Yan H. (**2024**) A high-precision ellipse detection method based on quadrant representation and top-down fitting. *Pattern Recognition* 154:110603

[35] Hershberger J E, Snoeyink J. (**1992**) Speeding up the Douglas-Peucker line-simplification algorithm. *University of British Columbia* , Department of Computer Science 134-143.

[36] Barwick D S. (**2009**) Very fast best-fit circular and elliptical boundaries by chord data. *IEEE Trans. Pattern Anal. Mach. Intell* 31: 1147-1152.

[37] Ser P K, Siu W C. (**1995**) Novel detection of conics using 2-D Hough planes. *IEE Proceedings-Vision* , Image and Signal Processing 142: 262-270.

[38] Fernandes A. (**2009**) A correct set of equations for the real-time ellipse hough transform algorithm. *Technical Report* .

[39] Lu C, Xia S, Huang Wet al. (**2017**) Circle detection by arc-support line segments. *2017 IEEE Int. Conf. Inf. Process (ICIP) 76* -80.

[40] Chia, Alex Y S, Rahardja S, Rajan D, Leung, M K. (**2011**) A Split and Merge Based Ellipse Detector With Self-Correcting Capability. *IEEE Trans.Image Process* 20: 1991-2006.