A Fast Ellipse Detector Using Projective Invariant Pruning

Qi Jia, Xin Fan, Member, IEEE, Zhongxuan Luo, Lianbo Song, and Tie Qiu

Abstract—Detecting elliptical objects from an image is a central task in robot navigation and industrial diagnosis where the detection time is always a critical issue. Existing methods are hardly applicable to these real-time scenarios of limited hardware resource due to the huge number of fragment candidates (edges or arcs) for fitting ellipse equations. In this paper, we present a fast algorithm detecting ellipses with high accuracy. The algorithm leverage a newly developed projective invariant to significantly prune the undesired candidates and to pick out elliptical ones. The invariant is able to reflect the intrinsic geometry of a planar curve, giving the value of $-1$ on any three collinear points and $+1$ for any six points on an ellipse. Thus, we apply the pruning and picking by simply comparing these binary values. Moreover, the calculation of the invariant only involves the determinant of a $3 \times 3$ matrix. Extensive experiments on three challenging data sets with 650 images demonstrate that our detector runs 20%-50% faster than the state-of-the-art algorithms with the comparable or higher precision.

Index Terms—Ellipse detection, Projective invariant, Real-time.

1 INTRODUCTION

Ellipses are quite common in natural or artificial scenes. The detection of ellipses in a fast and reliable manner from real world images provides a powerful analysis tool for many computer vision applications such as wheels detection [1], biological cell division [2], and object segmentation for industrial applications [3]. Ellipse detection still remains unresolved as one of the classical tasks with long history. Most existing methods perform far from real time, which hinder their applications in reality.

The earliest ellipse detection algorithm dates back to the classical Hough transform (HT) that fits the parametric form of an ellipse using a voting scheme [4]. The standard HT approach extracts ellipses by finding the clusters in a five-dimensional (5D) parametric space, consuming a great deal of memory and time. The randomized HT (RHT) improves the performance by reducing the number of false alarms [5]. The iterative RHT (IRHT) speeds up RHT so significant by focusing on the candidates likely to be an ellipse that it only needs 1-D accumulators [6]. However, both RHT and IRHT are still quite slow attributing to the voting processing among numerous candidates, and the geometry relationships between points are also neglected during voting.

Researchers introduce algebraic or geometric constraints on points of an ellipse to screening candidates. Liang et al. [7] introduce the maximum correntropy criterion into the constrained least-square fitting to alleviate the influence of outliers. Mulleti et al. [8] use the finite rate of innovation sampling principle to fit noisy or partial ellipse. Both methods produce ellipses with less bias. However, they can only work on the image with one single ellipse. Xie and Ji exploit the symmetry of two points on ellipse, reducing the voting parameter to one [9]. However, it is time-prohibitive to enumerate every point pairs as elliptical candidates. Basca et al. [10] accelerate Xie et al.’s method with RHT by considering only a small random subset of initial point pairs. Zhang and Liu [11] use edge directional properties to reduce point combinations that lie out of the same ellipse boundary.

Many methods take into account the geometric constraints on arc segments as the symmetry between points brings too many candidate pairs. Kim et al. [12] extract arc segments approximated by short straight lines. Libuda et al. [13] and Prasad et al. [14] improve Kim et al.’s method with less memory usage. However, these candidate arcs connecting short line segments may merge intersected arcs from different ellipses, resulting in lower precisions [12], [13]. Nguyen et al. [15] detect ellipses upon arcs by edge grouping. Their method is able to handle incomplete ellipses, but fails to detect ellipses splitting into many short arcs. Some other works formulate the mergence of elliptical arcs as an assignment problem, and iteratively correct the detection results [16], [17], [18]. These methods have high detection rates, but suffer from heavy computational costs. Prasad et al. [19] merge elliptical arcs with the relationship score given by the center of the ellipse fitting the arcs. Recently, Fornaciari et al. [20] develop an ellipse detector that classifies elliptical arcs into four groups and estimates the ellipse parameters using the decomposed parameter space. There are still a number of candidate arcs in each group while their method renders a relatively faster detection than previous methods. Especially, it is quite time consuming to calculate every possible combinations of arcs from each group, not to mention that many of them are wrong combinations.

All aforementioned methods start the estimation from points or arcs with their positional constraints. Actually, there exist more constraints whether points or arcs are amenable to the analytical ellipse equation. In [21], RANSAC is used to randomly choose five points repeatedly until the ellipse determined by these five points closely passes through a maximum number of edge points. This method is inefficient as there exist so many combinations of five points and one has to calculate ellipse parameters for each
In this paper, we circumvent the high computational load by pruning and picking candidates using a projective invariant, named characteristic number (CN) [23]. The projective invariance of CN is introduced in [24] acting as geometric constraints for fiducial point localization under face pose changes. Later, Jia et al. employ this invariant property to construct a shape descriptor robust to perspective deformations [25]. For the first time, we explicitly take the advantage of the CN property giving the characterization of the intrinsic geometry of an underlying planar curve of points.

2 CHARACTERISTIC NUMBER ON LINE AND CONIC

In this section, we first introduce the general definition of the characteristic number (CN) and its property on lines and conic curves. Section 3 elaborates our fast ellipse detection algorithm with line pruning and arc picking using CN. Section 4 demonstrates experimental evaluations on accuracy and efficiency. Section 5 concludes the paper.

2.1 Characteristic number

The characteristic number extends the classical cross ratio in various respects, and reflects the intrinsic geometry underlying given points. The CN value of three collinear points is $-1$, while six points lying on a conic curve have a common CN value $+1$. We give the definition of $CN$ below [23].
The proof process also implies that the calculation of CNL is equal to the determinant of a $3 \times 3$ matrix. In this study, we use this property to prune line segments in ellipse detection.

\begin{equation}
CNL(P, Q) = CN(P, Q) = \prod_{i=1}^{3} \frac{b_i^{(1)}}{a_i^{(1)}} = -1,
\end{equation}

The proof process also implies that the calculation of CNL is achieved by the determinant of a $3 \times 3$ matrix. In this study, we use this property to prune line segments in ellipse detection.

### 2.3 Characteristic number on six points of a conic

We denote the line through two points $Q_1^{(1)}$ and $Q_1^{(2)}$ as $Q_1^{(1)}Q_1^{(2)},$ and the intersection of two lines as $< Q_1^{(1)}Q_1^{(2)}, Q_2^{(1)}Q_2^{(2)} >.$ Let $\{ Q_i^{(j)} \mid i = 1, 2, 3; j = 1, 2 \}$ be six distinct points on a conic (ellipse) as shown in Fig. 3, and $P_1, P_2,$ and $P_3$ be three intersection points of the lines connecting some pairs of the six points on the conic:

\begin{align}
P_1 = & < Q_3^{(1)}Q_3^{(2)}, Q_1^{(1)}Q_1^{(2)} >, \\
P_2 = & < Q_1^{(1)}Q_1^{(2)}, Q_2^{(1)}Q_2^{(2)} >, \\
P_3 = & < Q_2^{(1)}Q_2^{(2)}, Q_3^{(1)}Q_3^{(2)} >.
\end{align}

Similar to (3), each point of $\{ Q_i^{(j)} \mid i = 1, 2, 3; j = 1, 2 \}$ can be linearly represented by a pair of points from $\{ P_1, P_2, P_3 \}$.

\begin{align}
Q_1^{(1)} &= a_1^{(1)}P_1 + b_1^{(1)}P_2, \\
Q_1^{(2)} &= a_1^{(2)}P_1 + b_1^{(2)}P_2, \\
Q_2^{(1)} &= a_2^{(1)}P_2 + b_2^{(1)}P_3, \\
Q_2^{(2)} &= a_2^{(2)}P_2 + b_2^{(2)}P_3, \\
Q_3^{(1)} &= a_3^{(1)}P_3 + b_3^{(1)}P_1, \\
Q_3^{(2)} &= a_3^{(2)}P_3 + b_3^{(2)}P_1.
\end{align}

We have the characteristic number on six points of a conic (CNC) as (7) by substituting the representation coefficients $a_i^{(j)}$ and $b_i^{(j)}$ into (2), and the CNC value equals $+1.$ We apply this property of CN to pick arc segments likely lying on an ellipse.

\begin{equation}
CN(C(P, Q) = \prod_{i=1}^{3} \frac{2}{a_i^{(j)}} = 1.
\end{equation}

We provide a simple proof to $CN(C(P, Q) = +1$ based on Pascal’s hexagon theorem [26] as below.

**Proof** Let $\{ Q_i^{(j)} \mid i = 1, 2, 3; j = 1, 2 \}$ be six points on a conic. As shown in Fig. 3, we can obtain three more intersections as

\begin{equation}
\begin{aligned}
R_1 &= < Q_2^{(2)}Q_3^{(1)}, P_1P_2 >, \\
R_2 &= < Q_1^{(1)}Q_3^{(2)}, P_2P_3 >, \\
R_3 &= < Q_2^{(1)}Q_1^{(2)}, P_1P_3 >.
\end{aligned}
\end{equation}

Then $R_1, R_2,$ and $R_3$ can be represented by $\{ Q_i^{(j)} \mid i = 1, 2, 3; j = 1, 2 \}$ and $\{ P_1, P_2, P_3 \}$ through simple calculations as

\begin{equation}
\begin{aligned}
R_1 &= -| Q_2^{(2)}Q_3^{(1)}, P_2P_1 | + | Q_2^{(2)}Q_3^{(1)}, P_1P_2 |, \\
R_2 &= -| Q_1^{(1)}Q_3^{(2)}, P_3P_2 | + | Q_1^{(1)}Q_3^{(2)}, P_2P_3 |, \\
R_3 &= -| Q_1^{(2)}Q_3^{(1)}, P_3P_1 | + | Q_1^{(2)}Q_3^{(1)}, P_1P_3 |,
\end{aligned}
\end{equation}

where we use homogeneous coordination to represent a planar point as $A = [A(x), A(y), A(z)]^T,$ and $\{ A, B, C \}$ denotes the determinant of the $3 \times 3$ matrix given by the homogeneous coordinates of the three points $A, B,$ and $C$ as

\begin{equation}
[ A, B, C ] = \begin{vmatrix}
A(x) & B(x) & C(x) \\
A(y) & B(y) & C(y) \\
A(z) & B(z) & C(z)
\end{vmatrix}.
\end{equation}

As the homogeneous coordinate of a point in a projective plane is independent on the initial points constructing the plane, we specify $P_1 = (1, 0, 0)^T, P_2 = (0, 1, 0)^T,$ and $P_3 = (0, 0, 1)^T.$
Consequently, the points \( R_1, R_2 \) and \( R_3 \) can be represented by \( P_i, a^{(j)}_i \) and \( b^{(j)}_i \) \((i = 1, 2, 3; j = 1, 2)\) by substituting (6) into (9) as

\[
\begin{align*}
R_1 &= (b^{(2)}_2 b^{(1)}_3, -a^{(2)}_2 a^{(1)}_3, 0)^T, \\
R_2 &= (0, -b^{(1)}_1 b^{(2)}_3, a^{(1)}_1 a^{(2)}_3)^T, \\
R_3 &= (a^{(2)}_1 a^{(1)}_2, 0, -b^{(2)}_1 b^{(1)}_2)^T. \\
\end{align*}
\]

According to Pascal’s hexagon theorem [26], \( R_1, R_2, \) and \( R_3 \) are collinear, i.e.,

\[
|R_1, R_2, R_3| = 0. 
\]

The proof of (12) is provided in Appendix A.

Finally, we can obtain (7) by substituting (11) into (12). The proof is completed. Again, as seen from (9), the ratios of several determinants of \( 3 \times 3 \) matrices generate CNC.

### 3 Fast ellipse detection

In this section, we present our ellipse detector using the characteristic number to prune non-elliptical line segments and pick arc segments lying on an ellipse. These pruning and picking processes significantly reduce the number of arc candidates for final fitting, rendering fast detection. The complete detection procedure includes preprocessing, line pruning, arc selection, and parameter fitting and ellipse validation.

At the preprocessing step, edge points are detected and linked to generate arc segments, where short segments are removed as noise. We delete those arc segments likely to be lines detected by CNL values at the line pruning step. This step reduces the number of arc segments that are not parts of any ellipses. However, the possible arc combinations are still too many to efficiently fit elliptical parameters. At the following arc selection step, we firstly divide arc segments into four groups, and remove some impossible arc combinations across these groups according to their relatively positional relationships. Subsequently, we apply CNC to pick the arc combinations belonging to one ellipse. Only those picked arc segments are used to fit elliptic parameters in the last fitting and validation step. The pruning and picking with CNL and CNC significantly reduce the number of possible arc combinations that determine the computational load of the fitting. Also, these pruning and picking steps run fast so the overall detector is quite efficient.

#### 3.1 Preprocessing

Given an image, we firstly smooth the image to partially suppress noise, and apply the Canny edge detector [27] with default thresholds in Matlab to extract consecutive edge points. The edge detector outputs both the position \( x_i \) and \( y_i \), and gradient \( \tau_i \) on each edge point as \( e_i = (x_i, y_i, \tau_i) \), where \( i = 1, 2, ..., N \), \( \tau_i = dy/dx \), \( N \) is the number of edge points.

It is possible to apply the CN constraint on any three or six edge points to determine whether they lie in a line or ellipse. However, we have to calculate \( C^3_N \) and \( C^6_N \) point combinations for \( N \) edge points, resulting in high computational complexity. Moreover, most of these combinations come from noise, or different ellipses and lines, spending a large amount time on invalid CN calculations. Instead, we apply the constraints on arc segments.

In order to efficiently shear invalid arc combinations for later processing, we group arc segments into four sets corresponding to the arcs from one ellipse distributing in four quadrants as \( \text{Arc}_{I}, \text{Arc}_{II}, \text{Arc}_{III}, \) and \( \text{Arc}_{IV} \), named quadrant sets. In the preprocessing step, we separate edge points \( e_i (i = 1, 2, ..., N) \) into two groups \( \text{Arc}_{I} \cup \text{Arc}_{IV} \) and \( \text{Arc}_{II} \cup \text{Arc}_{III} \) by the signs of edge gradients \( \tau_i \) as the first stage of this arc grouping:

\[
\begin{align*}
\tau_i > 0, & \; e_i \in \text{Arc}_{I} \cup \text{Arc}_{IV}, \\
\tau_i < 0, & \; e_i \in \text{Arc}_{II} \cup \text{Arc}_{III}. \\
\end{align*}
\]

We link each edge point with the other edge points in its eight neighborhood from the same group with a breadth-first strategy until no edge point exists in the neighborhoods. Consequently, we separately generate a series of arc segments in two groups shown in Fig. 4. Figure 4(b) demonstrates the arcs by linking edge points from the Canny detector (shown in Fig. 4(a)) of the group \( \text{Arc}_{I} \cup \text{Arc}_{IV} \), and Fig. 4(c) shows the arc segments of \( \text{Arc}_{II} \cup \text{Arc}_{III} \).

As the second stage of arc grouping, we divide each group into two sets, eventually producing four sets. As shown in Fig. 5, the vertices \( (e_1(x), e_1(y)), (e_2(x), e_2(y)), (e_3(x), e_3(y)), (e_4(x), e_4(y)) \) constitute the bounding box of an arc with the length of \( t \), where the starting and ending edge points of the arc are \( e_1 \) and \( e_4 \), respectively. Denoting the difference between the numbers of pixels above and below (slashed and solid white blocks) can split the sets into two sets further. The last figure shows the splitting process.

Fig. 5. Grouping arcs into four sets. The first two figures are two arcs in \( \text{Arc}_{I} \cup \text{Arc}_{II} \). The difference between the numbers of pixels above and below (slashed and solid white blocks) can split the sets into two sets further. The last figure shows the splitting process.
Fig. 4. Arcs detection and grouping. (a) shows edge points from Canny detector; (b) shows the arcs in sets $\text{Arc}_{II}$ and $\text{Arc}_{IV}$; (c) shows the arcs in sets $\text{Arc}_{I}$ and $\text{Arc}_{III}$; (d) shows the result after removing noise and lines, and the arcs in the same set are labeled in the same color. There are four colors represent arcs in four sets.

Fig. 6. The segments after removing noise and lines. The top row gives the arc segments with edge linking, and the bottom provides those after noise and line pruning.

with edge linking, and the bottom provides those after noise and line pruning. Many segments from noise and lines disappear, significantly reducing the number of arc candidates.

3.3 Arc selection

In this step, we pick candidate arcs, that are likely to assemble an ellipse, across the four sets where noise and line segments have been removed. Specifically, two arcs, each with three edge points, are taken from two different sets (one arc per set) in order to construct CNC. As shown in Fig. 7, there are two arcs $\text{ar}_{c1}$ and $\text{ar}_{c2}$, where $Q^{(2)}_1$, $Q^{(1)}_1$ and $Q^{(2)}_2$ are the mid and two endpoints of $\text{ar}_{c1}$, respectively. Similarly, $Q^{(2)}_3$, $Q^{(1)}_2$ and $Q^{(2)}_3$ are the points in $\text{ar}_{c2}$. We intersect $Q^{(1)}_1$, $Q^{(2)}_1$ and $Q^{(1)}_2$, $Q^{(2)}_2$ at $P_1$, $Q^{(1)}_2$, $Q^{(2)}_2$ and $Q^{(1)}_3$, $Q^{(2)}_3$ at $P_2$, and $Q^{(1)}_2$, $Q^{(2)}_2$ and $Q^{(1)}_3$, $Q^{(2)}_3$ at $P_3$. Given these points in $\mathcal{P}$ and $Q$, we are able to have the representation coefficients $a^{(j)}_i$ and $b^{(j)}_i$ ($i = 1, 2, 3$ and $j = 1, 2$), and calculate the CNC value for this arc combination by substituting these coefficients into (7). Equation 7 also tells that the CNC value equals +1 if these two arc segments with six points come from one identical ellipse. Therefore, picking two arc candidates from one ellipse turns out to simply comparing the CNC value of the arc combination with the value +1. Similar to line pruning, we can also use a threshold ($T_{\text{HCNC}}$) that determines false negative for this process. The arc pair belonging the same ellipse obtains the value of CNC close to +1, and the absolute difference of CNC value and +1, represented as $\text{Dis}_\text{CNC}$, is close to 0, as shown in Fig. 8 where the triangle constructed by the six points from the pair is labeled in light green. The red-brown triangle indicates the arc pair having the CNC value greater than $T_{\text{HCNC}}$, where the two arc segments of this pair lie on different ellipses.

We only consider the combinations with arc segments from two adjacent quadrant sets, e.g., $\{\text{Arc}_I, \text{Arc}_{II}\}$ and $\{\text{Arc}_{II}, \text{Arc}_III\}$. The arc pair (labeled in green and blue) belonging the same ellipse obtains the value of CNC, $\text{Dis}_\text{CNC}$, close to +1, and the triangle is labeled in light green. The red-brown triangle indicates the arc pair (labeled in blue and pink) having the CNC value greater than $T_{\text{HCNC}}$. Points on arcs are labeled in purple, and vertices of triangles are labeled in grey.

Fig. 7. Construction of CNC based on two arcs. $Q^{(2)}_1$ is the mid points of $\text{ar}_{c1}$, while $Q^{(2)}_1$ and $Q^{(1)}_2$ are two endpoints of $\text{ar}_{c1}$. Similarly, $Q^{(2)}_3$, $Q^{(2)}_2$ and $Q^{(2)}_3$ are three points on $\text{ar}_{c2}$.

Fig. 8. Threshold ($T_{\text{HCNC}}$) is used to determine false negative for arc selection. The arc pair (labeled in green and blue) belonging the same ellipse obtains the value of CNC, $\text{Dis}_\text{CNC}$, close to +1, and the triangle is labeled in light green. The red-brown triangle indicates the arc pair (labeled in blue and pink) having the CNC value greater than $T_{\text{HCNC}}$. Points on arcs are labeled in purple, and vertices of triangles are labeled in grey.
The combinations with more arc segments are also able to yield more accurate parameter estimation in the later step. Unfortunately, noise and/or occlusions are likely to bring the absence of arc segments from one quadrant set. Hence, we constitute the combinations with three arcs from three different quadrant sets upon the arc pairs picked by CNC for later parameter fitting. There are four kinds of valid arc combinations from three quadrant sets for an ellipse, as shown in the first row of Tab. 1. We pick out these combinations in each kind by using both coordinate and CNC constraints together, where the coordinate constraint takes relative locations of arcs to remove invalid combinations. The calculation of CNC value is so instable that the determination of ellipse is no more effective. We exclude the combinations with arcs from two diagonal quadrant sets, avoiding this instability.

Specifically, the picking process for a three-arc combination begins with an arc segment in the middle quadrant set, and then proceed to those in the other two sets. We find an arc pair first, and then the third arc to form the combination by alternatively applying the coordinate and CNC constraints. We take the set combination \( \{ \text{Arc}_{I}, \text{Arc}_{II}, \text{Arc}_{III} \} \) in Tab. 1 as an example to illustrate the picking process. We start with one arc in \( \text{Arc}_{I} \), and test the pair of the arc of \( \text{Arc}_{II} \) and every arc segments in \( \text{Arc}_{III} \) with the coordinate constraint. If one pair meet the coordinate constraint, the CNC constraint is applied to this pair further. Subsequently, we search the set \( \text{Arc}_{III} \) to find the third arc segments forming a pair with the arc in \( \text{Arc}_{II} \) that follow the coordinate constraint. Herein, the CNC constraint runs only once in order to balance the speed and accuracy. We repeatedly run the picking process for every arc segments in \( \text{Arc}_{I} \), and find all valid three-arc combinations for \( \{ \text{Arc}_{I}, \text{Arc}_{II}, \text{Arc}_{III} \} \). Algorithm 1 details the picking process for this set combination. The similar process applies to the other three-set combinations in the first row of Tab. 1, and finally we have all valid three-arc combinations ready for parameter fitting given in the next section.

### 3.4 Parameter fitting

There are five parameters to determine an ellipse, including its center (two), orientation (one), and major and minor semi-axes (two). We follow the procedure in [20] to estimate all these parameters but the center whose calculation we provide below.

We estimate the center as the intersection of auxiliary lines generated from any two-point pairs on the three arcs picked by the previous procedure. As illustrated in the top figure of Fig. 10, \( S_1 \) and \( E_1 \) are two points on an arc segment, \( T_1 \) is the intersection of the two tangent lines from \( S_1 \) and \( E_1 \), and \( M_1 \) is the middle point of the line segment \( S_1E_1 \). It is proved that the line \( M_1T_1 \) passes through the ellipse center \( O \). For practical images, it is unnecessary that all such auxiliary line segments like \( M_1T_1 \) given by any point pairs on the elliptical arcs, but we are able to locate the center as the point closest to those lines by the least square fitting. In this study, we find that the accuracy of this fitting (estimation) also depends on the tangents of the points along the ellipse.

We use \( n_d \) (\( n_d = 16 \) for our experiments) parallel chords instead of tangent lines in order to minimize the effects from tangent deviations, as shown in the bottom figure of Fig. 10. The points \( F_1 \) and \( F_2 \) are the mid points of the arcs \( S_1E_1 \) and \( S_2E_2 \), respectively. We generate \( n_d \) chords parallel to \( S_1F_1 \), and so we do to the chord \( S_2F_1 \). The points \( M_1 \) and \( M_2 \) are mid-points of the two series of parallel chords to \( S_1F_1 \) and \( S_2F_1 \), respectively. The points of \( M_1 \) and \( M_2 \), lying on the line \( m_2 \), and the intersection of \( m_1 \) and \( m_2 \) determines the ellipse center \( O \). We estimate \( m_1 \) and \( m_2 \) using a fast variant of the robust Theil-Sen estimator [28] with two arcs in adjacent quadrant sets. Consequently, we obtain four lines through the ellipse center generated from a three-arc combination, yielding at most six pairwise intersections. The algebraic average of the coordinates of these six intersections is taken as the output ellipse center \((x_0, y_0)\).

### 3.5 Ellipses validation

Candidate ellipses are available after the parameter fitting step. There may exist false positives or duplicated ones in these candidates so that a validate step is necessary.

---

**Algorithm 1** Picking algorithm for a three-arc combination  
**Input:** \( \text{Arc}_{I}, \text{Arc}_{II}, \text{Arc}_{III} \)  
**Output:** valid arc combinations set  
for each \( \hat{\text{Arc}}_{c_1} \in \text{Arc}_{I} \) do  
for each \( \hat{\text{Arc}}_{c_2} \in \text{Arc}_{II} \) do  
if \( \hat{\text{Arc}}_{c_1}, \hat{\text{Arc}}_{c_2} \) do not meet coordinate constraints then  
Continue.  
end if  
if \( \text{Dis}_{\text{CNC}}(\hat{\text{Arc}}_{c_1}, \hat{\text{Arc}}_{c_2}) > \text{Th}_{\text{CNC}} \) then  
Continue.  
end if  
for each \( \hat{\text{Arc}}_{c_3} \in \text{Arc}_{III} \) do  
if \( \hat{\text{Arc}}_{c_1}, \hat{\text{Arc}}_{c_3} \) do not meet coordinate constraints then  
Continue.  
end if  
Add \( \{ \hat{\text{Arc}}_{c_1}, \hat{\text{Arc}}_{c_2}, \hat{\text{Arc}}_{c_3} \} \) to valid arc combinations set  
end for  
end for  
end for  
return valid arc combinations set
to pick up the center of a given cluster, removing duplicate
differences of ellipse parameters. A voting strategy is adopted
comparison. Larger the ratio between the total length of one three-arc combination
equation to calculate how it fits the equation. We count the number
one measures how many edge points fitting the corresponding
The second index accounts for arc lengths of three-arc com-
other state-of-the-art methods.
4 EXPERIMENTAL RESULTS AND ANALYSIS
We perform a series of experiments on data sets with both
All the experiments in this paper are executed on a desktop with
F-measure is defined as:
F-measure = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},
(14)
where
\text{Precision} = \frac{\text{\Omega}}{\text{\Omega}}, \text{Recall} = \frac{\text{\Psi}}{\text{\Gamma}}.
(15)
The symbol \( \Omega \) denotes the number of detected ellipses, and \( \Gamma \) indicates the number of ground-truth ellipses. \( \Psi \) is the number of correctly detected ellipses. The overlapping ratio of a detected ellipse \( \mathcal{E}_d \) to the ground truth \( \mathcal{E}_g \) is defined as:
\mathcal{M}(\mathcal{E}_d, \mathcal{E}_g) = \frac{\text{area}(\mathcal{E}_d \cap \mathcal{E}_g)}{\text{area}(\mathcal{E}_d) \cup \text{area}(\mathcal{E}_g)}.
(16)
where \( \text{area}(\mathcal{E}) \) is the number of pixels inside the ellipse \( \mathcal{E} \). The detected ellipse \( \mathcal{E}_d \) is considered as a correct detection if \( \mathcal{M}(\mathcal{E}_d, \mathcal{E}_g) > T \mathcal{H}_o \). The threshold \( T \mathcal{H}_o \) is set to 0.8 throughout our experiments, as did in [29].
Experimental data contain both real world and synthetic images. We use data sets with real world images to validate efficiency
1. The source code and resultant images of our detector can be found at https://github.com/dlut-dimt/ellipse-detector
and accuracy, while synthetic ones are designated to demonstrate robustness to noise and ellipse variations. Real world images are those from Dataset #1 and Dataset Prasad, the same as [20], for fair comparisons. Dataset #1 is composed of 400 images having elliptic shapes, collected from MIRFlickr and LabelMe repositories [30]. Those MIRFlickr images are of high quality, and most of them contain only one object (ellipse), while those from LabelMe are noisy images of low resolution, containing multiple objects. Dataset Prasad contains 198 real images from [29], where objects of oval shapes like human faces are regarded as ellipses. Besides evaluations on the original Dataset Prasad, we also construct Dataset #2 by selecting 50 images with rigorous ellipses in order to show the accuracy of our method on detecting ellipses. These rigorously elliptical shapes are quite common in industrial inspection and diagnosis.

4.2 Performance analysis

The effectiveness of line pruning and arc selection steps are illustrated by comparisons on detection results with and without these steps. We also provide empirical studies on the hyper-parameters involved in these steps. The parameters for ellipse fitting and validation are taken the same values as those in [20].

4.2.1 Performance analysis of line pruning

We remove short and straight line segments using CNL as discussed in Section 3.2. Our goal is to prune noise effects and lines in input images as much as possible but preserve arcs of ellipses. We perform line pruning on Dataset #1 by tuning \( Th_{CNL} \) from 0 to 5.0. Intuitively, the threshold \( Th_{CNL} \) limits the height of the triangle formed by the three edge points for CNL calculation. Therefore, the larger \( Th_{CNL} \) is, the more arcs are to be removed. The zero threshold, \( Th_{CNL} = 0 \), indicates no line pruning step included.

Table II lists the values of arc numbers, running time, and F-measure with varying \( Th_{CNL} \) values. There are averagely 181 arc segments without line pruning \( (Th_{CNL} = 0) \), and more segments are removed when increasing \( Th_{CNL} \) values. The computing time is 28.86ms, and F-measure is 0.4313 without line pruning. When noise and line segments are removed by CNL, the computing time is becoming lower and F-measure increases. We obtain the best performance, 8.52ms for computation and 0.4692 for F-measure when \( Th_{CNL} = 3.0 \). The computing time is still decreasing when more arc segments are pruned by the thresholds larger than 3.0. The values of F-measure slightly decrease due to increased false negative. These results show that the line pruning step using CNL is effective to remove noise and lines, and also alleviate the computational load for later steps. We set \( Th_{CNL} = 3.0 \) to balance the accuracy and computing time in all comparisons with the others.

| \( Th_{CNL} \) | arc num | avg. time(ms) | F-measure |
|---------------|---------|---------------|-----------|
| 0             | 181     | 28.86         | 0.4313    |
| 1             | 170     | 26.52         | 0.4338    |
| 2             | 108     | 13.75         | 0.4576    |
| 3             | 65      | **8.52**      | **0.4692**|
| 4             | 40      | 6.79          | 0.4499    |
| 5             | 26      | 6.17          | 0.4226    |

Fig. 11. Relationship between point coordinates and CNC values. Different colors indicate various CNC values given by the color bar on the right most. Five distinct points \( P_1, ..., P_5 \) on the ellipse are fixed, and the sixth point around the ellipse are varied to show the distribution of CNC values.

4.2.2 Performance analysis of arc selection

We use the geometric constraint of six points on arcs derived from CNC to pick up arc segments belonging to one ellipse. Theoretically, the CNC value of six points lying on an ellipse equals +1, but various imaging conditions (e.g., thermal noise and lens distortions) in practical applications may cause the value deviating from +1. As discussed in Section 3.3, we relax this hard constraint to a range in the vicinity of +1 determined by \( Th_{CNC} \). Herein, we perform experimental analysis on the relationship between point coordinates and CNC values. This analysis does not only give arise to an appropriate threshold, but also validates the effectiveness of the arc selection based on CNC.

Supposing an ellipse centering at the coordinate origin, we fix five distinct points \( P_1, ..., P_5 \) on the ellipse, and vary the sixth point around the ellipse to show the distribution of CNC values in Fig. 11. Different colors indicate various CNC values given by the color bar on the right most of Fig. 11. All the CNC values higher than 1.4 are colored in red-brown, while all those lower than 0.6 in blue. Figure 11 illustrates that most of CNC values with the sixth point close to the ellipse fall within the range from 0.6 to 1.4. There exist several regions where CNC values lie in the range while the sixth point locates far from the ellipse. These regions include the star-shaped area out of the ellipse between \( P_2 \) and \( P_3 \), and the bottom left and top right ends of the line stretching out of \( P_3 \). This observation can be explained by the fact that the CNC value of +1 indicates the points lying on a conic curve not only an ellipse. However, these ‘outliers’ have few effects on our detector, because other types of conic curves, e.g., hyperbola and parabola, seldom appear in practical industrial images, and the majority can be also removed by the coordinate constraint even if a few of them appear.

We further apply our detector on dataset #1 by varying \( Th_{CNC} \), the absolute deviation from +1, from 0 to 50, giving seventeen values in total. Figure 12 illustrates the values of computing time with varying thresholds. The blue line in Fig. 12 demonstrates the seventeen values of \( Th_{CNC} \) in an ascending order, also listed in the third row below the plot. The step values
vary with the value ranges of $Th_{CNC}$, i.e., 0.1 for the range from 0 to 0.6, 1.0 for 1 to 5, and 10 for values larger than 10. The orange and grey bars show the computing time with and without the arc selection step, whose values are listed in the first and second rows below, respectively. The average computing time is 6.96 ms for each image, one half of the detection time without CNC. 13.4ms, when we use the hard constraint $Th_{CNC} = 0$ picking only a small fraction of arc segments. More arc segments are included for parameter fitting, demanding more computing time, when increasing the threshold values. When $Th_{CNC}$ is larger than 20, the detector spends more time than that without the selection step since the calculation of CNC takes more time than what the constraint can save. As a result, the orange bars are higher than the grey ones for these $Th_{CNC}$ values in Fig. 12.

Figure 13 shows the values of F-measure (F-m) with varying threshold values. The F-measure for the detector without the CNC constraint is 0.4385 labeled in the gray bar, while ours in orange bars. The hard constraint yields a very low F-m 0.0033 since the choice excludes many arc segments slightly deviating from an ellipse, resulting in significant false negatives. The threshold 0.2 outputs the best F-m 0.4641, even higher than the detector without this constraint. This improvement shows that the CNC constraint is also able to exclude false positives in addition to decreasing the computing time from 13.40ms to 8.58ms. As expected, the values of F-m given by the thresholds larger than 20 are quite close to that without the selection step. In these cases, the selection step takes no effect on lowering down false positives. Therefore, the arc selection step using CNC is quite crucial to both efficiency and accuracy. We choose the threshold as $Th_{CNC} = 0.2$ to generate the best performance in the following experiments.

### 4.3 Comparisons with the state of the art

Firstly, we compare our detector with three recent arc-based methods, i.e., Zhang [11], Libuda [13], and Fornaciari [20], on Dataset #1. The set consists of images with different qualities and various numbers of target ellipses. The latest works of [7] and [8] are not so relevant as these three because both are point-based and applicable to scenarios where only one ellipse appear. The execution program of [20] is provided by the authors, and we take the results of [11] and [13] reported in [20]. Table III lists the comparisons on average running time and F-measure. The method of Zhang [11] performs the worst: the fitting on a large number of comparisons on average running time and F-measure. The method of Libuda [13] performs the best performance in the following experiments.

Several examples of the above methods on Dataset #1 are shown in Fig. 14 where the first column is the input images, and the second one lists the ground truth (GT). False negatives occur in all the results of Libuda [13] except for the fifth row. The method of Fornaciari [20] works quite close to ours that correctly detects almost all the ellipses in these images. It is worth noting that only our method successfully picks out the middle wheel partially occluded by the lady in the third image, while the other three fail. Compared with the ground truth, our detector outputs one extra ellipse out of the bottom right tray. Actually, one can find a dim elliptical trail along the tray, but the ground truth neglects it.

Secondly, we compare the execution time on each processing step with the-state-of-the-art on Dataset #1, as shown in Tab. IV. The method of Zhang [11] spends about 4243 ms on estimating the parameters due to huge numbers of possible point combinations. The method of Prasad [29] reduces the time on parameter estimation via grouping arcs with curvature and convexity, but the grouping spends additional 278ms. The method of Libuda [13] uses an iterative strategy, and the time spent on each of the first three steps equals about 4.5ms. Fornaciari’s detector [20], the fastest one among the existing, spends the most time 4.9ms on grouping arcs with their relative locations against the other steps. In contrast, our method only uses about half of the execution time of [20] on the grouping step, and the total execution time is less than two third of [20]. Thus, our detector reduces the time for the grouping step, the bottleneck for efficient detectors.

Further, we peer into the proposed detector to analyze how line pruning and arc selection accelerate detection. Table V illustrates the effects of CNL and CNC by listing the averaged values of arc numbers, arc combination numbers (CC) for parameter fitting, execution time and F-measure on Dataset #1. Without any processing using CNL nor CNC, possible averaged arcs and arc combinations (CC) are 181 and 2019, respectively. After line pruning, the arc number reduces to 65, about one third of of the original, and the CC value significantly decreases to 196, 6% of the original. Only 40 arc combinations remain for parameter fitting, shearing 98.5% combinations from the original. Naturally, the execution time drops down 93% from 129.33ms to 8.52ms. The value of F-measure also increases from 0.3163 to 0.4692 since many false positives are removed by these two steps using CNL and CNC. Green curves in the first and second rows of Fig. 15 present possible ellipses to be fitted without and with our pruning/selection steps, respectively. It is evident that the green curves in the second row are much less than those in the first row, intuitively showing the effectiveness of our CNL/CNC based processing to save time and improve accuracy.

**TABLE V**

| Method       | After arc detection | After line pruning | After arc selection |
|--------------|---------------------|--------------------|---------------------|
| arc Num      | 181                 | 65                 | 65                  |
| CC           | 2619                | 196                | 40                  |
| avg.time (ms)| 129.33              | 12.73              | 8.52                |
| F-measure    | 0.3163              | 0.4692             | 0.4090              |

Finally, we compare our detector with [20], giving the best performance among the existing, in terms of CC, F-measure, and execution time on three data sets including Dataset #1, Dataset #2, and Dataset Prasad. As shown in Tab. VI, our method outperforms [20] on all the three data sets. The last column shows...
Fig. 12. The values of computing time with varying thresholds. We vary the absolute deviation from $+1$, from 0 to 50, giving seventeen $Th_{CNC}$ values in total, labeled in blue line and listed in the third row. The orange and grey bars show the computing time with and without the arc selection step.

![Graph showing computing time with varying thresholds.](image1)

Fig. 13. The values of F-measure ($F_m$) with varying thresholds. We take the same deviation from $+1$ as did in Fig. 12, while the orange and grey bars show the F-measure with and without the arc selection step.

![Graph showing F-measure with varying thresholds.](image2)

TABLE IV
Execution times (ms) for each step compared with the-state-of-the-art on Dataset #1.

| Step                        | Libuda [13] | Prasad [29] | Zhang [11] | Fornaciari [20] | Our   |
|-----------------------------|-------------|-------------|------------|-----------------|-------|
| Edage detection             | 4.49        | 3.54        | 3.97       | 3.45            | 3.43  |
| Pre-processing              | 4.15        | 78.03       | 3.55       | 1.94            | 1.90  |
| Grouping                    | 4.89        | 278.01      | 0.25       | 4.90            | 2.53  |
| Estimation                  | 0.84        | 3.40        | 423.66     | 2.30            | 0.63  |
| Validation and Clustering   | 0.00        | 460.39      | 0.03       | 0.21            | 0.06  |
| Total                       | 14.38       | 823.38      | 4243.86    | 12.79           | 8.54  |
4.4 Robustness to ellipse variations and noise

In order to investigate the robustness to ellipse variations and noise, we use two synthetic data sets with different orientations and ratios of two semi-axes of ellipses, and also apply salt-and-pepper noise to real world images. The noise break arcs into several small fragments, which may affect the accuracy and efficiency.

The first synthetic data set consists of 9100 ellipses with various semi-axes ratios and orientations. One semi-axis is fixed as 100, and the other one varies so that the ratios range from 0.01 to 1 at the step of 0.01. Orientations vary from 1° to 90° at the step of 1°. The second data set contains 10000 images with a fixed center and orientation, showing changes on ratios and lengths of semi-axes. One semi-axis varies from 1 to 100 at the step of 1, and the other one changes accordingly so that axes ratios range from 0.01 to 1 at the step of 0.01. Both data sets come from [20] that generates synthetic 400 × 400 images, each containing one single ellipse without noise.

Figure 16 illustrates comparisons with [20], where black points indicate failures of detection. The results on the first data set are given in Fig. 16(a) and Fig. 16(b). The horizontal axis gives ratios of two semi-axes, and the vertical one shows orientations. Both methods are robust to orientation changes as long as axes ratios are larger than 0.25, but they fail in the cases of small axes ratios when ellipses degenerate into straight lines. The results on the second data set are shown in Fig. 16(c) and Fig. 16(d), where the vertical axis indicates lengths of major axes. The robustness of both methods is quite similar, working well on ellipses whose major axis is longer than 10 and axes ratio is larger than 0.25. Small arcs are likely to be pruned as noise when the ellipse only has a few pixels. Fortunately, one may tackle the problem of small ellipses by upscaling the image as did in [20]. Therefore, the proposed method is quite robust, and only fails in some extreme cases, e.g., small and extremely oblate ellipses, which are quite
TABLE VI
Testing results of the proposed methods compared with [20] on three datasets.

| Database        | avg. time (ms) [20] | F-m [20] | CC [20] | Improvement percentage |
|-----------------|---------------------|----------|---------|------------------------|
| Dataset Prasad  | 4.34                | 0.35     | 0.3059  | 19                     | 22.70%    | 6.44%    |
| Dataset #1      | 13.58               | 8.55     | 0.4692  | 40                     | 37.05%    | 7.00%    |
| Dataset #2      | 5.26                | 4.09     | 0.6271  | 34                     | 22.27%    | 6.41%    |

ONCLUSIONS

In this paper, an ellipse detector for real-time application is proposed. We trade off accuracy and efficiency and pay more attention to the execution time. We introduce a new geometry constraint to prune lines and select arcs belong to the same ellipse. The detector removes the straight arcs based on characteristic on line (CNL), and selects candidate elliptical arc combinations by characteristic on conic (CNC). Our method outperforms the-state-of-the-art by the experiments on real images, which can be used in real-time for various applications. In the future, we will improve our method to make it possible to detect quite small ellipses in images, which is a challenge for most existing methods.

APPENDIX A

PROOF OF PASCAL’S HEXAGON THEOREM

Pascal’s hexagon theorem: Let \{Q_{ij}\}_{i=1, 2, 3; j = 1, 2} be different points on a non-degenerate conic \(C\), as shown in Fig. 3.

Then three intersections

\[
\begin{align*}
R_1 &= Q_2^2 Q_3^1 Q_1^1, \\
R_2 &= Q_3^1 Q_1^1 Q_2^2, \\
R_3 &= Q_2^2 Q_1^1 Q_3^1.
\end{align*}
\]

are collinear.

Proof: The coordinate of \(R_1, R_2\) and \(R_3\) can be represented by \{Q_{ij}\}_{i=1, 2; j = 1, 2} as

\[
\begin{align*}
R_1 &= (Q_2^2 \times Q_3^1 \times Q_1^1), \\
R_2 &= (Q_3^1 \times Q_1^1 \times Q_2^2), \\
R_3 &= (Q_2^2 \times Q_3^1 \times Q_1^1).
\end{align*}
\]

where \(\times\) denotes the cross product of two points. Then \(R_1, R_2\) and \(R_3\) can be represented through simple calculations as

\[
\begin{align*}
R_1 &= |Q_2^2| Q_3^1 Q_1^1 |Q_1^2, \\
R_2 &= |Q_3^1| Q_1^1 Q_2^2 |Q_2^2, \\
R_3 &= |Q_2^2| Q_3^1 Q_1^1 |Q_3^1|.
\end{align*}
\]

To prove that \(R_1, R_2\) and \(R_3\) are collinear is equivalent to prove

\[
|R_1, R_2, R_3| = 0.
\]

Then we can substitute (19) into (20), and get the equivalent equation as

\[
\begin{align*}
&|Q_1^1, Q_2^2, Q_3^1| |Q_1^2, Q_2^2, Q_3^1| \\
&|Q_1^1, Q_2^2, Q_3^1| |Q_1^2, Q_2^2, Q_3^1| \\
&|Q_1^1, Q_2^2, Q_3^1| |Q_1^2, Q_2^2, Q_3^1|.
\end{align*}
\]

In order to prove it, we can replace any point \(Q_{ij}\) with the general point \(Q(x, y, z)\), taking \(Q_1^1\) for example, then we can
get the parametric equations of conic $C$ by other five points. As $Q_1^{(1)}$ is one point on conic $C$, it must meet (21). The proof is completed.

**ACKNOWLEDGMENTS**

We thank Dr. M. Fornaciari, Dr. S. Mulleti and Dr. T. Lu for providing their executables and insights. We also thank Dr. M. Fornaciari for his experimental data.

**References**

[1] T. Cooke, “A fast automatic ellipse detector,” in 2010 International Conference on Digital Image Computing: Techniques and Applications. IEEE, 2010, pp. 575–580.

[2] S. Zafari, T. Eerola, J. Sampo, H. Kalviainen, and H. Haario, “Segmentation of overlapping elliptical objects in silhouette images,” IEEE Transactions on Image Processing, vol. 24, no. 12, pp. 5942–5952, 2015.

[3] C. Teutsch, D. Berndt, E. Trostmann, and M. Weber, “Real-time detection of elliptic shapes for automated object recognition and object tracking,” in Electronic Imaging 2006. International Society for Optics and Photonics, 2006, pp. 60700I-60700I.

[4] J. Binningworth and J. Kittler, “A survey of the hough transform,” Computer vision, graphics, and image processing, vol. 44, no. 1, pp. 87–116, 1988.

[5] R. A. MaLaughlin, “Randomized hough transform: improved ellipse detection with comparison,” Pattern Recognition Letters, vol. 19, no. 3, pp. 299–305, 1998.

[6] W. Lu and J. Tan, “Detection of incomplete ellipse in images with strong noise by iterative randomized hough transform (irht),” Pattern Recognition, vol. 41, no. 4, pp. 1268–1279, 2008.

[7] J. Liang, Y. Wang, and X. Zeng, “Robust ellipse fitting via half-quadratic and semidefinite relaxation optimization,” IEEE Transactions on Image Processing, vol. 24, no. 11, pp. 4276–4286, 2015.

[8] S. Mulleti and C. S. Seelamantula, “Ellipse fitting using the finite rate of innovation sampling principle,” IEEE Transactions on Image Processing, vol. 25, no. 3, pp. 1451–1464, 2016.

[9] Y. Xie and Q. Ji, “A new efficient ellipse detection method,” in 16th International Conference on Pattern Recognition, vol. 2. IEEE, 2002, pp. 957–960.

[10] C. Basca, M. Talos, and R. Brad, “Randomized hough transform for ellipse detection with result clustering,” in EUROCON 2005-The International Conference on “Computer as a Tool”, vol. 2. IEEE, 2005, pp. 1397–1400.

[11] S.-C. Zhang and Z.-Q. Liu, “A robust, real-time ellipse detector,” Pattern Recognition, vol. 38, no. 2, pp. 273–287, 2005.

[12] E. Kim, M. Haseyama, and H. Kitajima, “Fast and robust ellipse extraction from complicated images,” in Proceedings of IEEE information technology and applications. Citeseer, 2002.

[13] L. Libuda, I. Grothues, and K.-F. Kraiss, “Ellipse detection in digital image data using geometric features,” in Advances in Computer Graphics and Computer Vision. Springer, 2007, pp. 229–239.

[14] D. K. Prasad, M. K. Leung, and C. Quek, “Ellifit: An unconstrained, non-iterative, least squares based geometric ellipse fitting method,” Pattern Recognition, vol. 46, no. 5, pp. 1449–1465, 2013.

[15] T. M. Nguyen, S. Ahuja, and Q. J. Wu, “A real-time ellipse detection based on edge grouping,” in IEEE International Conference on Systems, Man and Cybernetics. IEEE, 2009, pp. 3280–3286.

[16] F. Mai, Y. Hung, H. Zhong, and W. Sze, “A hierarchical approach for fast and robust ellipse extraction,” Pattern Recognition, vol. 41, no. 8, pp. 2512–2524, 2008.

[17] A. Y.-S. Chia, S. Rahardja, D. Rajan, and M. K. Leung, “A split and merge based ellipse detector with self-correcting capability,” IEEE Transactions on Image Processing, vol. 20, no. 7, pp. 1991–2006, 2011.

[18] T. Lu, W. Hu, C. Liu, and D. Yang, “Effective ellipse detector with polygonal curve and likelihood ratio test,” IET Computer Vision, vol. 9, no. 6, pp. 914–925, 2015.

[19] D. K. Prasad and M. K. Leung, “Clustering of ellipses based on their distinctiveness: An aid to ellipse detection algorithms,” in 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 8. IEEE, 2010, pp. 292–297.

[20] M. Fornaciari, A. Prati, and R. Cucchiara, “A fast and effective ellipse detector for embedded vision applications,” Pattern Recognition, vol. 47, no. 11, pp. 3693–3708, 2014.

[21] Y. Sugaya, “Ellipse detection by combining division and model selection based integration of edge points,” in 2010 Fourth Pacific-Rim Symposium on Image and Video Technology (PSIVT). IEEE, 2010, pp. 64–69.

[22] C. Liu and W. Hu, “Effective method for ellipse extraction and integration for spacecraft images,” Optical Engineering, vol. 52, no. 5, pp. 057 002–057 002, 2013.

[23] Z. Luo, X. Zhou, and D. X. Gu, “From a projective invariant to some new properties of algebraic hypersurfaces,” Science China Mathematics, vol. 57, no. 11, pp. 2273–2284, 2014.

[24] X. Fan, H. Wang, Z. Luo, Y. Li, W. Hu, and D. Luo, “Fiducial facial point extraction using a novel projective invariant,” IEEE Transactions on Image Processing, vol. 24, no. 3, pp. 1164–1177, 2015.

[25] Q. Jia, X. Fan, Y. Liu, H. Li, Z. Luo, and H. Guo, “Hierarchical projective invariant contexts for shape recognition,” Pattern Recognition, vol. 52, pp. 358–374, 2016.

[26] N. Stefanović and M. Milošević, “A very simple proof of pascals hexagon theorem and some applications,” Proceedings-Mathematical Sciences, vol. 120, no. 5, pp. 619–629, 2010.

[27] J. Canny, “A computational approach to edge detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, no. 6, pp. 679–698, 1986.

[28] J. Matoušek, “Randomized optimal algorithm for slope selection,” Randomization and approximation techniques in computer science, pp. 358–374, 1991.

[29] D. K. Prasad, M. K. Leung, and S.-Y. Cho, “Edge curvature and convexity based ellipse detection method,” Pattern Recognition, vol. 45, no. 9, pp. 3204–3221, 2012.

[30] R. Hartley and A. Zisserman, Multiple view geometry in computer vision. Cambridge university press, 2003.