Pattern Recognition 47 (2014) 3693–3708

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A fast and effective ellipse detector for embedded vision applications

Michele Fornaciari [a,b], Andrea Prati [b,c,*], Rita Cucchiara [a,b]

[a] DIEF, University of Modena and Reggio Emilia, Via Vignolese, 905, 41125 Modena, Italy
[b] SOFTECH-ICT, Via Vignolese, 905, 41125 Modena, Italy
[c] DPPAC, University IUAV of Venice, Santa Croce 1957, 30135 Venezia, Italy

ARTICLE INFO

Article history:
Received 26 August 2013
Received in revised form
14 May 2014
Accepted 14 May 2014
Available online 27 May 2014

Keywords:
Real time ellipse detection
Hough transform
Selection strategy

ABSTRACT

Several papers addressed ellipse detection as a first step for several computer vision applications, but most of the proposed solutions are too slow to be applied in real time on large images or with limited hardware resources. This paper presents a novel algorithm for fast and effective ellipse detection and demonstrates its superior speed performance on large and challenging datasets. The proposed algorithm relies on an innovative selection strategy of arcs which are candidate to form ellipses and on the use of Hough transform to estimate parameters in a decomposed space. The final aim of this solution is to represent a building block for new generation of smart-phone applications which need fast and accurate ellipse detection also with limited computational resources.

1. Introduction

The recognition of geometrical shapes formally described by a mathematical model has a long tradition in pattern recognition. The attempts of finding new efficient solutions for detecting parametric shapes in noisy and cluttered images resulted in successful algorithms for lines and circles. They are based on accumulations/voting procedures (e.g. Hough–based methods), interpolation, curve fitting, and so on.

Similarly, the detection of ellipses has been often addressed in the past, although ellipses are more complex parametric curves due to the larger number of parameters. Ellipse detection is the starting point for many computer vision applications, since elliptical shapes are very common in nature and in hand-made objects. For instance, ellipse detection can be used in wheels detection [1], road sign detection and classification [2], object segmentation for industrial applications [3], automatic segmentation of cells from microscope imagery [3], pupil/eye tracking [4], and many more. With the advent of powerful mobile technologies for everyone and the spreading of new generations of smart-phones, the request of new applications running on these devices increased enormously. Despite their limitations, mobile devices are now powerful enough to enable on-board processing of complex data, including images and videos, allowing unprecedented capabilities in terms of applications. As a consequence, the scientific community has recently found large interest to image/video processing on smart-phones, often called *embedded* or *mobile vision* [5]. Possible applications range from real time object/person tracking [6], content-based retrieval of framed scene with markerless object recognition [7], face detection [8] (and possibly recognition), blind people aid for movement [9], etc.. As such, even though the advances in technology and multi-core processors will allow ever faster computation, keeping ellipse detection fast has the merit of allowing ever more complex computer vision applications.

Consequently, in this paper we present a new solution for very fast ellipse detection in real images. We choose arcs belonging to the same ellipse very quickly and very reliably by working at arc-level, instead of pixel-level, and by relying an innovative selection strategy. The ellipse center is then estimated exploiting the property of the midpoints of parallel chords, and remaining parameters are estimated accumulating votes in a decomposed parameter space. The good trade-off between efficiency (in the order of 10 ms per image) and accuracy makes this approach a proper candidate for implementation on mobile devices.

The rest of the paper describes the state-of-the-art in ellipse detection (Section 2), focusing mainly on fast algorithms. Section 3 describes in full details the proposed method which resulted to outperform all the existing fast methods Section 4 analyzes the novelties and the influence of the different parameters, also in comparison with other possible approaches. To prove the performance, Section 5 reports several experiments, both on synthetic and real images, including a dataset captured with mobile devices. Both efficiency and effectiveness are evaluated. Section 6 then summarizes the contributions of the paper.

* Corresponding author at: DPPAC, University IUAV of Venice, Santa Croce 1957, 30135 Venezia, Italy.
E-mail addresses: michele.fornaciari@unimore.it (M. Fornaciari), andrea.prati@iuav.it (A. Prati), rita.cucchiara@unimore.it (R. Cucchiara).

http://dx.doi.org/10.1016/j.patcog.2014.05.012
0031-3203/

2. Related works

The importance of ellipse detection in image processing is witnessed by the large amount of works present in the literature.

Most of the methods for ellipse detection rely on the Hough Transform – HT (or its variants) to estimate the parameters. Since an ellipse is analytically defined by five parameters, these methods try to overcome the main problem of a direct application of standard HT, which is a 5D accumulator. McLaughlin [10] rely on the randomized version of HT (RHT) and aim at reducing the memory usage using proper data structures. Lu and Tan [11] iteratively focus only on the points with higher probability to belong to the a single ellipse, thus reducing the parameter space to five 1D accumulators. A very common and memory efficient approach has been proposed by Xie and Ji [12] and Chia et al. [13], where four parameters are geometrically computed, estimating in a 1D accumulator the last one. Basca et al. [14] speeded up the method of Xie et al. using RHT, thus considering only a small random subset of the initial pairs of points. HT-based methods greatly suffer from noise (which includes both background noise and points belonging to different ellipses) which "dirties" the accumulator. Also, they are computationally intense and rather slow because of the voting procedure on a huge number of edge points combinations.

Instead of reducing the space dimensionality by means of strong assumptions, Aguado et al. [15] propose a decomposition of the parameter space: parameters are estimated in consecutive steps, leveraging previous results. Zhang and Liu [16] avoid unnecessary computations for those combinations of points that cannot lie to the same ellipse boundary by carefully selecting starting edge points.

Other approaches rely more heavily on the symmetry between the points on the boundary. Some methods [17–19] first find and analyze symmetry axes, estimating parameters using the HT, others [20] instead rely on symmetric relationships among boundary points and then adopt a least square fitting method.

All aforementioned methods start the estimation from sets of points, eventually selected according to some kind of geometric constraints. However, when considered unrelated to its neighbors, an edge pixel does not contribute significantly to a correct ellipse detection. A better characterization could be achieved using sets of connected edge pixels, i.e. *arcs*, which can be generated by linking short straight lines [1,21–24], splitting the edge contour [25–28], or validating connected edge pixels [29]. The ellipses parameters are obtained using ellipse fitting methods [30,31] on a reduced set of arcs, which are obtained grouping arcs according to their relative position and constraints on the curvature [1,21,22,25,26], or ellipse fitting error [23,24,27–29].

Most of the works present in the literature claim high detection accuracy. However, these results are validated mostly on a few synthetic images, and rarely on more than 10 real images, except [1,25]. The execution time for methods that claim to be fast or real-time [1,16,18,21–23,26] has been computed on few images as well, and may increase significantly on different kind of images. In this paper we present a novel method (preliminary works can be found in [32,33] for ellipse detection that results to be much faster than other state-of-the-art methods, while achieving similar or even better detection performance. We also present two annotated datasets (available on-line) of real images on which we tested both fast and effective methods for a fair evaluation.

3. Method description

We present a novel algorithm for fast ellipse detection designed for real-time performance on real world images. It first selects combination of arcs belonging to the same ellipse and then estimates its parameters via the Hough Transform in a decomposed parameter space. Let us first describe the overall procedure, as outlined in Fig. 1.

As first step arcs are extracted from the edge mask and classified in four classes according to their convexity. We classify edge pixels in two main directions according to their gradient phase and group 8-connected edge pixels in the same direction class to form *arcs*. Their quality is also improved removing short or straight arcs. Arcs are then classified according to their convexity, computed in a robust and efficient way. By combining the two classifications it is possible to assign each arc to a quadrant, in analogy with the final configuration in a Cartesian plane as depicted in Fig. 2(c). The method is tailored for the detection of visible ellipses, defined as having the boundary partially visible in at least three quadrants. Consequently we search for combinations of three arcs, called *triplets*, each belonging to a different quadrant. To avoid the combinatorial explosion, we select only triplets formed by arcs that satisfy three criteria based on convexity, mutual position and same pairwise estimated center. A selected triplet forms a *candidate* ellipse and, already knowing its center, we estimate the remaining three parameters in a decomposed Hough space requiring three 1D accumulators. Candidate ellipses are then validated according to the fitness of the estimation with the actual edge pixels. Since an ellipse may be supported by different triplets, multiple detections with slightly different parameters can be generated. We deal with multiple detections using a fast clustering procedure in the parameter space.

The following subsections will describe the different phases of the algorithm in detail.

3.1. Arc extraction

In this phase we extract arcs from the input image, first by detecting edge points, then grouping them in arcs, and finally classifying arcs based on edge direction and convexity.

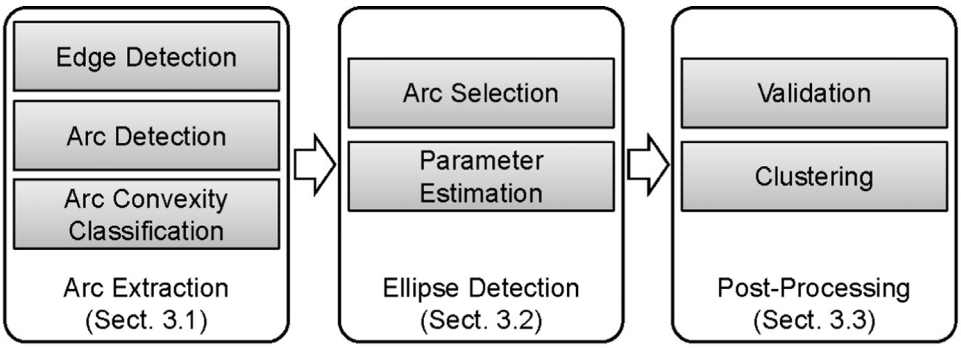

Fig. 1. Flowchart of the algorithm.

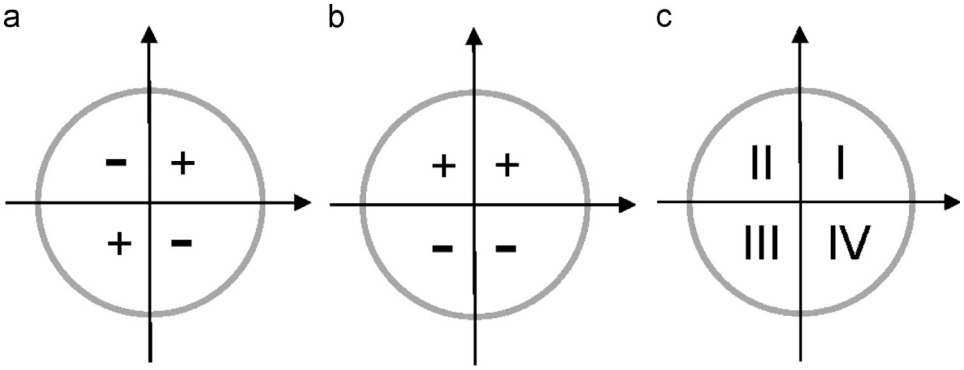

Fig. 2. Functions $\mathcal{D}, \mathcal{C}, \mathcal{Q}$. See the text for further details on these functions. (a) Directions $\mathcal{D}$. (b) Convexities $\mathcal{C}$. (c) Quadrants $\mathcal{Q}$.

3.1.1. Edge detection

Like most, also the proposed method begins analyzing the edge points, where every edge point $e_i = (x_i, y_i, \theta_i)$ is defined by its position x_i, y_i and the phase of the gradient θ_i. An edge detector is applied on the input image in order to obtain a set of edge points. We chose the Canny edge detector [34] with automatic thresholding[1] because of its properties of good detection, good localization and minimal response. In the detected edge points the gradient phase θ_i is obtained through the Sobel operator (already computed in the Canny algorithm).

3.1.2. Arc detection

Every edge point e_i is classified by the function $\mathcal{D} : e_i \rightarrow (+, -)$ into two main directions according to θ_i (Fig. 2(a)). We do not require accurate values of θ_i, and the classification $\mathcal{D}$ is simply done by checking the signs of the Sobel derivatives dx and dy:

$$\mathcal{D}(e_i) = \text{sign}(\tan(\theta_i)) = \text{sign}(dx) \cdot \text{sign}(dy) \tag{1}$$

We discard edge points lying on the classification boundary, i.e. with horizontal ($dy = 0$) or vertical ($dx = 0$) gradient direction. An *arc* α^k is formed by linking connected edge points of the same direction class:

$$\alpha^k = \{(e_1^k, \ldots, e_{N^k}^k) : \mathcal{D}(e_i^k) = \mathcal{D}(e_j^k), \forall i, j \wedge \mathit{Connected}(e_{i-1}^k, e_i^k)\} \tag{2}$$

where N^k represents the number of edge points belonging to the arc α^k and $\mathit{Connected}(e_{i-1}^k, e_i^k)$ verifies the 8-connectivity of two consecutive edge points.

By extension, we define the arc direction $\mathcal{D}(\alpha^k)$ as the direction class of its points. We also define $L^k \equiv e_1^k$ as the *left* end point of α^k, $R^k \equiv e_{N^k}^k$ as the *right* one, $M^k \equiv e_{\lfloor N^k/2 \rfloor}^k$ as the *middle* edge point, BB^k as the bounding box having L^k and R^k as nonadjacent vertices, and OBB^k as the oriented minimum area rectangle [35] enclosing all edge points $e_i^k \in \alpha^k$.

Some arcs α^k are not salient enough to characterize an ellipse and are readily removed: very short arcs ($N^k < Th_{length}$) that are mainly due to noise and arcs containing mostly collinear points (shortest side of $\text{OBB}^k < Th_{obb}$), thus not belonging to the curved boundary of an ellipse. The values of these parameters are investigated in Section 4.2.

3.1.3. Arc convexity classification

Every arc α^k is classified by the function $\mathcal{C} : \alpha^k \rightarrow (+, -)$ as having the convexity upward $(+)$ or downward $(-)$ (Fig. 2(b)). By construction, all the points $e_i^k \in \alpha^k$ lie inside BB^k, and divide it in two regions: we call U^k the region *under* the arc, and O^k the region *over* it (Fig. 3) computed with Algorithm 1.

[1] https://gist.github.com/egonSchiele/756833.

We find the convexity by comparing the areas of U^k and O^k:

$$\mathcal{C}(\alpha^k) = \begin{cases} + & \text{if } \mathit{Area}(\text{U}^k) > \mathit{Area}(\text{O}^k) \\ - & \text{if } \mathit{Area}(\text{U}^k) < \mathit{Area}(\text{O}^k) \end{cases} \tag{3}$$

Equal areas $\mathit{Area}(\text{U}^k)$ and $\mathit{Area}(\text{O}^k)$ implies that a meaningful convexity can not be determined (as in the case of inflexion points) and therefore the arc is discarded.

Algorithm 1. Get the convexity $\mathcal{C}(\alpha)$ of the arc α, when $\mathcal{D}(\alpha) = +$. Swap *area_O* and *area_U* when $\mathcal{D}(\alpha) = -$.

```
function GetConvexity (Point α[])
  N ← α.length()
  left ← α[0]
  right ← α[N − 1]
  current_x ← left.x
  area_O ← 0
  for i = 1 → N do
    if α[i].x ≠ current_x then
      area_O ← area_O + |α[i].y − left.y|
      current_x ← α[i].x
    end if
  end for
  area_BB ← |right.x − left.x| * |right.y − left.y|
  area_U ← area_BB − N − area_O
  if area_U > area_O then
    return +
  end if
  if area_U < area_O then
    return −
  end if
  if area_U = area_O then
    discard the arc
  end if
end function
```

Given the functions $\mathcal{D}$ and $\mathcal{C}$, for every arc α^k we can define the function $\mathcal{Q} : \alpha^k \rightarrow \{\text{I}, \text{II}, \text{III}, \text{IV}\}$, which maps α^k to its quadrant (Fig. 2(c)):

$$\mathcal{Q}(\alpha^k) = \begin{cases} \text{I} & \text{if } \langle \mathcal{D}(\alpha^k), \mathcal{C}(\alpha^k) \rangle = \langle +, + \rangle \\ \text{II} & \text{if } \langle \mathcal{D}(\alpha^k), \mathcal{C}(\alpha^k) \rangle = \langle -, + \rangle \\ \text{III} & \text{if } \langle \mathcal{D}(\alpha^k), \mathcal{C}(\alpha^k) \rangle = \langle +, - \rangle \\ \text{IV} & \text{if } \langle \mathcal{D}(\alpha^k), \mathcal{C}(\alpha^k) \rangle = \langle -, - \rangle \end{cases} \tag{4}$$

The chosen number of classes is four because it is the lowest number that allows to compute the convexity using Algorithm 1, thus reducing the number of triplets combination and allowing to generate edge with enough curvature.

In order to better understand the arc detection phase, Fig. 4 reports a toy example on a synthetic image where the same color corresponds to the same direction or quadrant.

3.2. Ellipse detection

We define a *candidate* ellipse $\mathcal{E}_i$ as a *triplet*, i.e. a set of three arcs $\tau^{abc}=(\alpha^a,\alpha^b,\alpha^c)$ that satisfy a set of criteria and are, thus, likely to belong to the same ellipse. The parameters of each $\mathcal{E}_i$ are then computed in a HT framework.

3.2.1. Arc selection strategy

Given N_α as the number of arcs in the image, the set $\mathcal{T}^0$ of all possible triplets will contain $\binom{N_\alpha}{3}$ elements. This number could be extremely high and most of triplets are composed by arcs that do not belong to the same ellipse: a large amount of computation will be wasted to estimate parameters for false detections. The goal of the selection strategy is to generate a subset of triplets containing only triplets τ^{abc} whose arcs belong to the same ellipse boundary.

We define a *pair* of arcs as $p^{ab}=(\alpha^a,\alpha^b)$. Thus, a triplet can also be defined as two pairs sharing an arc: $\tau^{abc}=\{(p^{ab},p^{dc})|\ \alpha^b\equiv\alpha^d\}$. The selection strategy first selects pairs whose arcs satisfy constraints on (i) convexity and (ii) mutual position. Then it computes the ellipse center assuming that both arcs lie on the same ellipse boundary. Finally, it finds a candidate ellipse as a triplet composed by two pairs that (iii) imply the same center.

The first constraint on convexity ensures that each pair is composed by arcs in subsequent quadrants (in counterclockwise order):

$$\mathcal{A}(p^{ab})=\begin{cases}\top & \text{if } (\mathcal{Q}(\alpha^a),\mathcal{Q}(\alpha^b))\in\{(\mathrm{I},\mathrm{II}),(\mathrm{II},\mathrm{III}),(\mathrm{III},\mathrm{IV}),(\mathrm{IV},\mathrm{I})\}\\ \bot & \text{otherwise}\end{cases}\tag{5}$$

where "$\top$" and "$\bot$" stand respectively for *true* and *false*. The subset $\mathcal{T}^1\subseteq\mathcal{T}^0$ is composed only of triplets whose pairs satisfy the first constraint:

$$\tau^{abc}\in\mathcal{T}^1\iff(\tau^{abc}\in\mathcal{T}^0)\wedge\mathcal{A}(p^{ab})\wedge\mathcal{A}(p^{dc})\tag{6}$$

The second constraint on mutual position discards pairs whose arcs are incoherent with the same elliptic shape. It is defined, with reference to Fig. 5, by the Boolean function:

$$\mathcal{M}(p^{ab})=\begin{cases}\top & \text{if } (\mathcal{Q}(\alpha^a),\mathcal{Q}(\alpha^b))\equiv(I,II)\ \wedge\ (L^a.x\gtrsim R^b.x)\\ \top & \text{if } (\mathcal{Q}(\alpha^a),\mathcal{Q}(\alpha^b))\equiv(II,III)\ \wedge\ (L^a.y\gtrsim L^b.y)\\ \top & \text{if } (\mathcal{Q}(\alpha^a),\mathcal{Q}(\alpha^b))\equiv(III,IV)\ \wedge\ (R^a.x\lesssim L^b.x)\\ \top & \text{if } (\mathcal{Q}(\alpha^a),\mathcal{Q}(\alpha^b))\equiv(IV,I)\ \wedge\ (R^a.y\lesssim R^b.x)\\ \bot & \text{otherwise}\end{cases}\tag{7}$$

where L and R are the leftmost and rightmost extrema of the arc, respectively, as defined in Section 3.1.2 and the inequalities $\gtrsim$ and $\lesssim$ are defined with a tolerance Th_{pos}, investigated in Section 4.2. The subset $\mathcal{T}^2\subseteq\mathcal{T}^1$ contains only triplets whose pairs respect the first and the second constraints:

$$\tau^{abc}\in\mathcal{T}^2\iff(\tau^{abc}\in\mathcal{T}^1)\wedge\mathcal{M}(p^{ab})\wedge\mathcal{M}(p^{dc})\tag{8}$$

The third constraint verifies whether the three arcs $\alpha^a,\alpha^b,\alpha^c$ lie on the boundary of the same ellipse or, equivalently, that the centers C^{ab}, C^{dc} implied by the pairs p^{ab}, p^{dc} of the triplet τ^{abc} are closer than a tolerance $Th_{centers}$ (see Section 4.2):

$$\mathcal{H}(\tau^{abc})=\begin{cases}\top & \text{if } C^{ab}\approx C^{dc}\\ \bot & \text{otherwise}\end{cases}\tag{9}$$

The subset $\mathcal{T}^3\subseteq\mathcal{T}^2$ contains only triplets whose pairs satisfy all constraints:

$$\tau^{abc}\in\mathcal{T}^3\iff(\tau^{abc}\in\mathcal{T}^2)\wedge\mathcal{H}(\tau^{abc})\tag{10}$$

Each triplet $\tau_i^{abc}\in\mathcal{T}^3$ is a candidate ellipse $\mathcal{E}_i$.

First and second constraints are very fast to compute and are still quite discriminative. The third is more complex, but is computed on a narrower set, and greatly improves the quality of candidate ellipses. More details on the performance of the selection strategy are available in Section 5.6.

3.2.2. Center estimation

We estimate the ellipse center C^{ab} for a given arc pair p^{ab} by means of a well known geometric property of ellipses: *the midpoints of parallel chords are collinear* [19], as shown in Fig. 6(a).

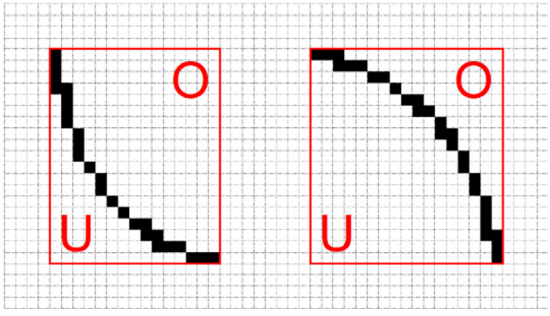

Fig. 3. Convexity classification.

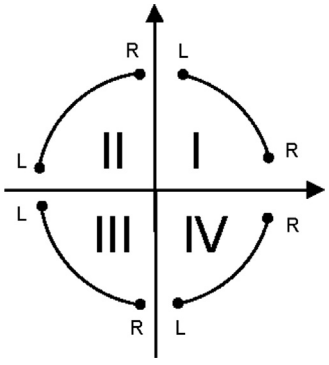

Fig. 5. Mutual position.

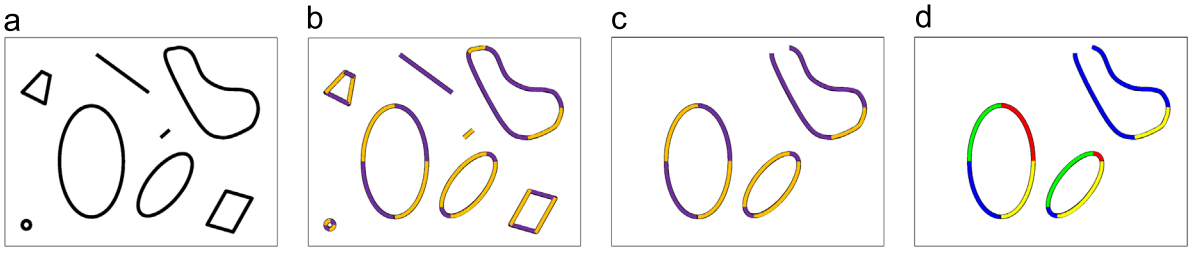

Fig. 4. Toy example of arc detection. Best viewed in colors. (a) Edge points. (b) Direction classification. (c) Removal of short and straight arcs. (d) Quadrants classification. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

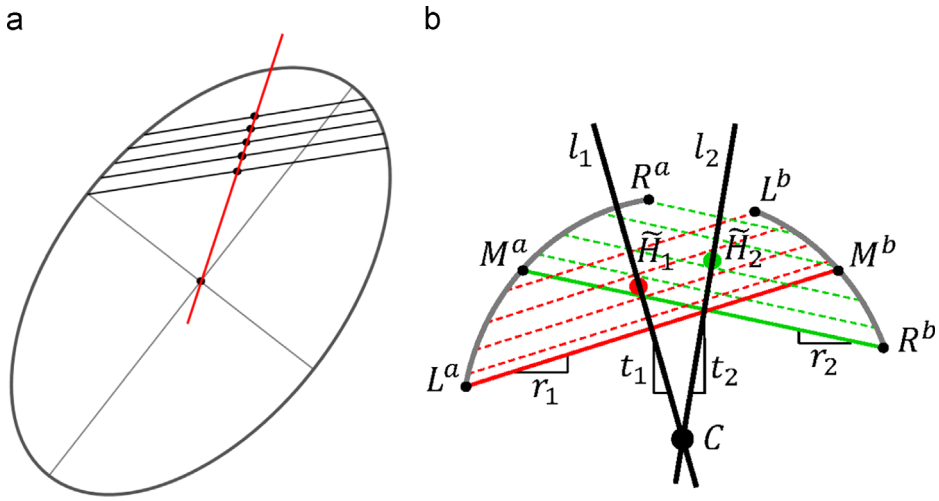

Fig. 6. Method for estimating the ellipse center. (a) The midpoints of a set of parallel chords and its center are collinear, (b) Geometric features to compute the center.

Thus, the intersection of two lines connecting the midpoints of two different sets of parallel chords is the ellipse center.

In order to find the center we need to generate two sets of parallel chords which are not parallel with each other. For clarity, we illustrate the procedure for the pair p^{ab} with reference to Fig. 6 (b). We generate two sets of N_s chords, parallel respectively to the lines $\overline{L^aM^b}$ and $\overline{R^bM^a}$ (being M the midpoint of the arcs as defined in Section 3.1.2), having slope r_1^{ab}, r_2^{ab}. The influence of the parameter N_s on the overall performance is investigated in Section 4.2. We call H_1^{ab}, H_2^{ab} the two sets of their midpoints. The two lines l_1^{ab} and l_2^{ab} intersecting the points in H_1^{ab} and H_2^{ab} will intersect in the center C^{ab}. However, especially in real images, the points in H_1^{ab}, H_2^{ab} are affected by noise and are not perfectly collinear, and the lines l_1^{ab}, l_2^{ab} have to be robustly estimated.

We estimate the slopes t_1^{ab}, t_2^{ab} of the lines l_1^{ab}, l_2^{ab} using a fast variant of the robust Theil-Sen estimator (Algorithm 2) [36] as the medians of the two sets of computed slopes S_1^{ab}, S_2^{ab}. We also generate the points $\tilde{H}_1^{ab}$, $\tilde{H}_2^{ab}$ whose coordinates are the medians of the coordinates of the points in H_1^{ab}, H_2^{ab}. We choose the median approach among other statistical measures to be consistent with the strategy of the Theil–Sen estimator, which adopts the median approach for its robustness to outliers.

Algorithm 2. Get the slope of the line best fitting the midpoints of parallel chords.

```
function GetSlope (Point midpoints[])
    middle ← midpoints.length()/2
    for i = 0 → middle do
        x1 ← midpoints[i].x
        y1 ← midpoints[i].y
        x2 ← midpoints[middle+1+i].x
        y2 ← midpoints[middle+1+i].y
        slope ← (y2−y1)/(x2−x1)
        S[i] ← slope
    end for
    return Median(S)
end function
```

Thus, given an arc pair p^{ab}, the coordinates of the center C^{ab} can be computed as follows (see Fig. 6(b)) (superscripts omitted for clarity):

$$C.x = \frac{\tilde{H}_2.y - t_2\tilde{H}_2.x - \tilde{H}_1.y + t_1\tilde{H}_1.x}{t_2 - t_1} \tag{11}$$

$$C.y = \frac{t_1\tilde{H}_2.y - t_2\tilde{H}_1.y + t_1t_2(\tilde{H}_1.x - \tilde{H}_2.x)}{t_2 - t_1} \tag{12}$$

We say that the centers C^{ab}, C^{dc} of two pairs p^{ab}, p^{dc} coincide and satisfy the third constraint of the selection strategy (Eq. (9)) if they lie within a given distance $Th_{centers}$ (see Section 4.2) which accounts for image noise. Following the toy example reported in Figs. 4 and 7 shows how the selection strategy works for the arc pointed by the arrow.

3.2.3. Parameter estimation

The ellipse parameters are estimated only for those triplets $\tau^{abc} = (p^{ab}, p^{dc})$ that satisfy the three selection strategy constraints. Since, by definition, the lines l_1, l_2 of the two pairs p^{ab}, p^{dc} should intersect the ellipse center, their pairwise intersection should always identify the center. However in noisy images this is rarely true. For a better estimation, we consider the ellipse center (x_c, y_c) as the median of the coordinates of a set of 7 points, consisting of the two centers C^{ab}, C^{dc}, their mean, and the other 4 intersection of the estimated lines $\{l_1^{ab} \cap l_1^{dc}, l_1^{ab} \cap l_2^{dc}, l_2^{ab} \cap l_1^{dc}, l_2^{ab} \cap l_2^{dc}\}$ as represented by gray points in Fig. 8.

In order to find the remaining parameters, the parameter space is decomposed as described in [15] in semi–axes ratio $N = B/A$ and orientation ρ. After the derivation of [15,16,37] the values of N and ρ are computed as

$$N = \begin{cases} N_+ & \text{if } N_+ \le 1 \\ 1/N_+ & \text{otherwise} \end{cases} \tag{13}$$

$$\rho = \begin{cases} \arctan(K_+) & \text{if } N_+ \le 1 \\ \arctan(K_+) + \dfrac{\pi}{2} & \text{otherwise} \end{cases} \tag{14}$$

where

$$\gamma = q_1q_2 - q_3q_4 \tag{15}$$

$$\beta = (q_3q_4 + 1)(q_1 + q_2) - (q_1q_2 + 1)(q_3 + q_4) \tag{16}$$

$$K_+ = \frac{-\beta + \sqrt{\beta^2 + 4\gamma^2}}{2\gamma} \tag{17}$$

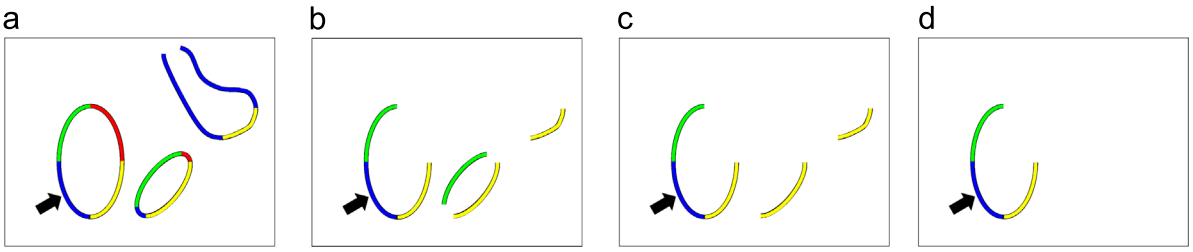

Fig. 7. Toy example of selection strategy. Best viewed in colors. (a) $\alpha^b \equiv \alpha^d$ is the pointed arc. Find α^a and α^c. (b) Constraint on convexity. (c) Constraint on mutual position. (d) Constraint on centers. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

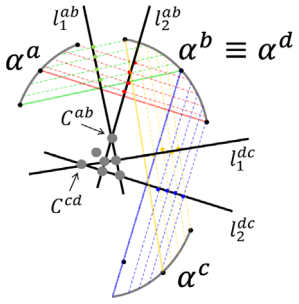

Fig. 8. Estimated center of the ellipse.

Table 1
Values to be assigned to q_1, q_2, q_3, q_4 to estimate N and ρ.

(α^a, α^b)		(α^d, α^c)	
q_1	q_2	q_3	q_4
r_1^{ab}	$S_1^{ab}[s],\ \forall s$	r_1^{dc}	$S_1^{dc}[s],\ \forall s$
r_1^{ab}	$S_1^{ab}[s],\ \forall s$	r_2^{dc}	$S_2^{dc}[s],\ \forall s$
r_2^{ab}	$S_2^{ab}[s],\ \forall s$	r_2^{dc}	$S_2^{dc}[s],\ \forall s$
r_2^{ab}	$S_2^{ab}[s],\ \forall s$	r_1^{dc}	$S_1^{dc}[s],\ \forall s$

$$N_+ = \sqrt{\frac{(q_1 - K_+)(q_2 - K_+)}{(1+q_1K_+)(1+q_2K_+)}} \tag{18}$$

One vote is accumulated for each combination of the parameters q_1, q_2, q_3, q_4, as reported by row in Table 1. Setting the values of q_1, q_3 to the slope of the parallel chords, we vary the values of q_2, q_4 with the slope of all lines computed by the Theil–Sen estimator (Algorithm 2). The values of N and ρ are then the highest peaks in the two 1D accumulators.

Given the values N and ρ, the value of the major semi–axis A is

$$A = A_x / \cos(\rho) \tag{19}$$

where:

$$x_0 = \frac{(x_i - x_c) + (y_i - y_c)K}{\sqrt{K^2+1}} \tag{20}$$

$$y_0 = \frac{-(x_i - x_c)K + (y_i - y_c)}{\sqrt{K^2+1}} \tag{21}$$

$$A_x = \sqrt{\frac{x_0^2N^2 + y_0^2}{N^2(K^2+1)}} \tag{22}$$

The value of A is estimated in a 1D accumulator considering as (x_i, y_i) every edge point e_i of the three arcs α^a, α^b, α^c, and taking the highest peak. The value of the last parameter, minor semi–axis B, is then

$$B = A \cdot N \tag{23}$$

3.3. Post–processing

The selection strategy defines three necessary, but not sufficient, conditions for the detection of an ellipse. Candidate ellipses must then be validated to remove false detections. Also, multiple triplets may be found on the boundary of the same ellipse, generating multiple detections of the same ellipse which need to be merged. Since the number of candidate ellipses is much less than the number of arcs, merging multiple detections is more efficient than finding all the arcs lying on the boundary of a same ellipse at an early stage.

3.3.1. Validation

A quality measure is assigned to each candidate ellipse $\mathcal{E}_i$. Recalling that we can get the position of a point $(\overline{x}_i, \overline{y}_i)$ with respect to the boundary of $\mathcal{E}_i$ by means of the ellipse equation:

$$f(\overline{x}_i, \overline{y}_i, \mathcal{E}_i) = \begin{cases} = 1 & \text{if } (\overline{x}_i, \overline{y}_i) \text{ is on the boundary of } \mathcal{E}_i \\ > 1 & \text{if } (\overline{x}_i, \overline{y}_i) \text{ is outside the boundary of } \mathcal{E}_i \\ < 1 & \text{if } (\overline{x}_i, \overline{y}_i) \text{ is inside the boundary of } \mathcal{E}_i \end{cases} \tag{24}$$

we can define the set $\mathcal{B} = \{(\overline{x}_i, \overline{y}_i) : |f(\overline{x}_i, \overline{y}_i, \mathcal{E}_i) - 1| < 0.1\}$ that contains the points that are close to the boundary. The score $\sigma \in [0, 1]$ summarizes how well the points of the three arcs composing $\mathcal{E}_i$ fit the boundary of the estimated ellipse:

$$\sigma = \frac{|\mathcal{B}|}{|\alpha^a| + |\alpha^b| + |\alpha^c|} \tag{25}$$

A candidate ellipse $\mathcal{E}_i$ with $\sigma > Th_{score}$ is considered as *valid*, otherwise as a false detection and is discarded.

3.3.2. Clustering

Multiple valid detections of the same ellipse are clustered by adopting a variant of the approach described in [38], which allows to assess the similarity of two ellipses $\mathcal{E}_i, \mathcal{E}_j$ comparing the distances between centers (Eq. (26)), axes (Eqs. (27) and (28)), and rotation (Eq. (29)) separately:

$$\delta_c = \sqrt{(\mathcal{E}_i.x_c - \mathcal{E}_j.x_c)^2 + (\mathcal{E}_i.y_c - \mathcal{E}_j.y_c)^2} < \min(\mathcal{E}_i.B, \mathcal{E}_j.B) \times 0.1 \tag{26}$$

$$\delta_a = (|\mathcal{E}_i.A - \mathcal{E}_j.A| / \max(\mathcal{E}_i.A, \mathcal{E}_j.A)) < 0.1 \tag{27}$$

$$\delta_b = (|\mathcal{E}_i.B - \mathcal{E}_j.B| / \min(\mathcal{E}_i.B, \mathcal{E}_j.B)) < 0.1 \tag{28}$$

$$\delta_\rho = \begin{cases} \dfrac{\measuredangle(\mathcal{E}_i.\rho - \mathcal{E}_j.\rho)}{\pi} < 0.1 & \text{if } \left(\dfrac{\mathcal{E}_i.B}{\mathcal{E}_i.A} < 0.9\right) \wedge \left(\dfrac{\mathcal{E}_j.B}{\mathcal{E}_j.A} < 0.9\right) \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

Ellipses $\mathcal{E}_i$ and $\mathcal{E}_j$ are considered as equivalent according to the function $\Delta : (\mathcal{E}_i, \mathcal{E}_j) \to (\top, \bot)$:

$$\Delta(\mathcal{E}_i, \mathcal{E}_j) = \bigwedge(\delta_c, \delta_a, \delta_b, \delta_\rho) \tag{30}$$

All valid ellipses are ordered by decreasing score and compared one by one with the center of each cluster by means of the function Δ. We consider the highest score ellipse as the center of a

given cluster. If the current ellipse cannot be assigned (i.e. is not equivalent) to any cluster, it becomes the center of a new cluster. The parameters are set according to [38].

4. Discussion on the method

This section summarizes the novel contributions of the proposed method, as well as the differences with other methods, and presents a thorough discussion on its parameters are presented.

4.1. Novelty and comparison

Arc generation. In the arc detection step (Section 3.1.2), the coarse gradient direction is computed by means of the Sobel derivatives. Other approaches, such as the Fast Line Extraction (FLE) of Kim et al. [39], approximate curved lines with short lines, and compute the gradient direction from the relationship between their starting and ending points. We compared the gradient computed using Sobel derivatives and FLE on the same data as in [39], i.e. various ellipses which have an axis ranging from 20 to 100 pixels, by fixing the other axis at 100 pixels. Fig. 9 shows the error between the two methods and the direction mathematically computed through differentiation for each boundary pixel according to the known ellipse parameters. Both the average and the maximum errors of our method are smaller than FLE. A more challenging test is reported in Table 2, where, among others, we show the performance on real images of our method and its variant that computes the direction of the gradient relying on FLE, namely [39] + Ours. These results show that the use Sobel derivatives is both efficient and accurate enough.

Arc detection. Arcs are detected by labeling connected edge points within the same direction class, avoiding time-demanding refinements such as the search for inflexion points or sharp turn

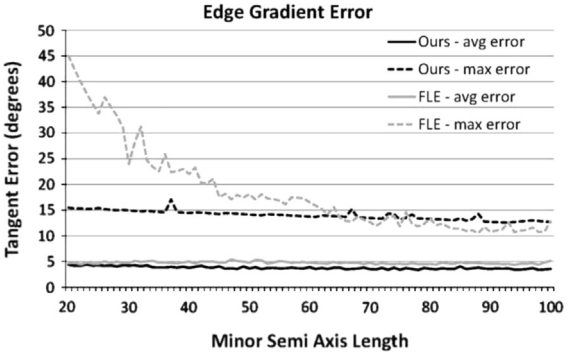

Fig. 9. The maximum and average error of our method and Fast Line Extraction [39].

Table 2
Average effectiveness and execution time (in milliseconds) on the three datasets. N/A = implementations on mobile devices not available.

Algorithm	Dataset Prasad		Dataset #1		Dataset #2	
	F-measure (%)	Time	*F*-measure (%)	Time	*F*-measure (%)	Time
[14]	23.98	134.98	31.77	684.31	28.84	N/A
[22]	33.47	7.85	42.58	18.95	45.88	N/A
[25]	**44.18**	158.32	45.12	1084.85	43.28	N/A
[16]	21.53	431.95	34.21	5591.5	45.06	N/A
Ours	43.70	**5.56**	**58.52**	**15.96**	**61.13**	**45.82**
[39] + Ours	34.04	8.39	54.44	25.98	58.72	N/A
Ours + [30]	39.72	7.02	48.93	22.77	44.86	N/A
Ours + [31]	39.78	7.43	48.82	20.44	44.70	N/A

[25,26], or junction points [27,28]. On one hand, this procedure will affect the performances in synthetic datasets explicitly designed to test the robustness of algorithm in particular scenarios, as reported in Section 5.4. On the other hand, the simplicity of the algorithm allows to achieve in real-time (in the order of milliseconds) state-of-the-art results (or even better) on real images, as confirmed by the experiments reported in Section 5.5.

Convexity. The convexity is computed by counting the number of pixels under the arc. Unlike the method of Zhang and Liu [16], this algorithm (Algorithm 1) is robust to thick edges and do not need angle estimation. Other methods to compute convexity cannot be applied: the method of Guil et al. [40] is suitable only for concentric ellipses, while the method of Prasad et al. [25] computes the convexity only in relation with another arc.

Selection strategy constraints. There are several works in the literature that adopt criteria to reduce the search space. However, we present a new set of efficient and effective criteria: (i) straight arcs are removed by thresholding the shortest side of their oriented bounding box; (ii) arcs are selected according to their convexity at arc level, rather than edge point level as in [16]; arcs are also selected according to their (iii) mutual position and (iv) implied center, without the need to compute a search region as in [25] or relying on ellipse fitting algorithms to find the center first [22].

Center computation. The ellipse center is computed exploiting the property of the midpoints of parallel chords [19]. It does not require angle estimations and is thus well suited for real world images, where the edges and gradient directions may be very noisy. This property has been already exploited in [18,19,41] to compute the center, first by finding symmetric points through horizontal and vertical scans in the image, and then by computing the pairwise intersection of the symmetric axes estimated via the HT. The proposed method presents several improvements: (i) parallel chords are not bounded to be horizontal or vertical; (ii) the procedure is performed on pair of arcs instead of the whole image, thus (iii) the two symmetric axes are computed by a simple line estimator, and (iv) there is only one intersection, i.e. the center. The center may be computed also relying on other methods, which however have some drawbacks for real time processing of real images. The property of the tangents [42,43] requires accurate edge gradient estimation. RANSAC [23] is robust to outliers but is not guaranteed to find the optimal solution or to converge within a given time. Circle fitting [1,21] or ellipse fitting [22,24–29] methods are dependent on the amount of noise.

Parameter estimation. The ellipse parameters are estimated as soon as a valid triplet is selected, clustering at a later stage multiple detections, if any. This procedure is opposed to every other method, where the grouping procedure aims at retrieving first all arcs that belong to the same ellipse, and then estimates its parameters. The proposed formulation overcomes the limitation of [15,16] to use 2D accumulators: it is optimized for the detection of a single ellipse at a time, thus needing only 1D accumulators to estimate rotation, axes ratio and major semi-axis length.

Clustering. By testing the similarity between two ellipses in the parameters space, the clustering method is much faster than evaluating their overlap as in [25], and the function Δ is more accurate than thresholding the Euclidean distance in the parameter space as in [14]. Differently from [38], the test on the center is related to the ellipse size, instead of the image size: as a results, ellipses with same parameters are clustered similarly regardless the image size.

4.2. Parameter selection

The proposed method is influenced by 5 parameters, Andrea: which may be considered too many, making the algorithm tuning

hard. However, their effect on the performance, both detection effectiveness and execution time, is discussed, showing either their negligible influence on the performance or an easy procedure to tune them. The parameters are tested on two datasets, detailed in Section 5.5, namely Dataset Prasad proposed by Prasad et al. [25] and Dataset #1 proposed in this paper.

Spurious edge and noise removal. During the preprocessing, as discussed in Section 3.1.2, we remove short and straight arcs, that are mainly due to noise and do not have enough curvature to contribute to the detection of an ellipse. We define as short those arcs shorter than Th_{length} (parameter #1). The influence of the parameter Th_{length} is investigated in Fig. 10, where it is clear that either including all edges (left side of the graph) or removing too many edges (right side) has negative impact on the detection effectiveness. The best results are obtained with values of Th_{length} between 2 and 32. However, as clearly depicted in Fig. 10(b), values lower than 8 are very time demanding. As a result, we set $Th_{length} = 16$.

Edges are defined as straight if the shortest side of the respective oriented bounding box is smaller than Th_{obb} (parameter #2). Fig. 11 shows the influence of this parameter on the performance. Both the detection effectiveness (Fig. 11(a)) and the execution time (Fig. 11(b)) decrease with values greater than 3 (Fig. 11(a)). We set $Th_{obb} = 3$ as a good trade-off.

Mutual position. The second constraint of the selection strategy (Section 3.2.1) forms arc pairs only for arcs with coherent mutual position, which is computed analyzing the relationships among the extrema of the arcs, with a tolerance Th_{pos} (parameter #3). In Fig. 12 we show that the impact of Th_{pos} on the method is negligible. We set $Th_{pos} = 1$ to gain some effectiveness (Fig. 12(a)) and keep the execution time at the minimum (Fig. 12(b)).

Distance between estimated centers. As discussed in Section 3.2.2, the third constraint of the selection strategy discards arc pairs whose computed centers are more distant than $Th_{centers} = \tau_{centers} \times$ image diagonal. As shown in Fig. 13, for $\tau_{centers}$ (parameter #4) greater than 0.02 there is no significant improvement in the detection effectiveness (Fig. 13(a)), but the computational time increases because more triplets are considered as valid (Fig. 13 (b) shows that the increased time is due to the estimation step). A value of $\tau_{centers} = 0.05$ that guarantees top effectiveness and still fast execution time is then selected.

Number of parallel chords. As described in Section 3.2.1, our method finds N_s (parameter #5) chords parallel to a given one. Considering all chords starting from each edge point of the arc α^a, i.e. $N_s = |\alpha^a|$, can be very time consuming and not useful to better locate the ellipse center. A more efficient approach is to consider fewer chords, starting from a subset of edge points sampled at regular intervals on the arc, to minimize the execution time and

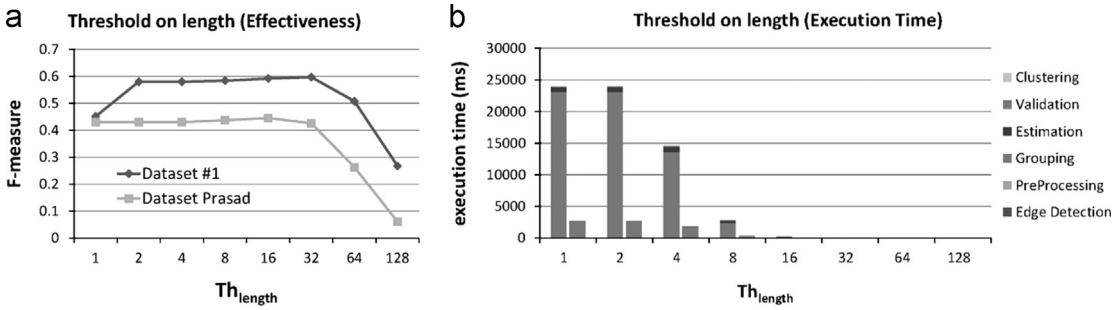

Fig. 10. Performance varying Th_{length}. (a) *F*-measure. (b) Execution time on Dataset #1 (left column) and Dataset Prasad (right column).

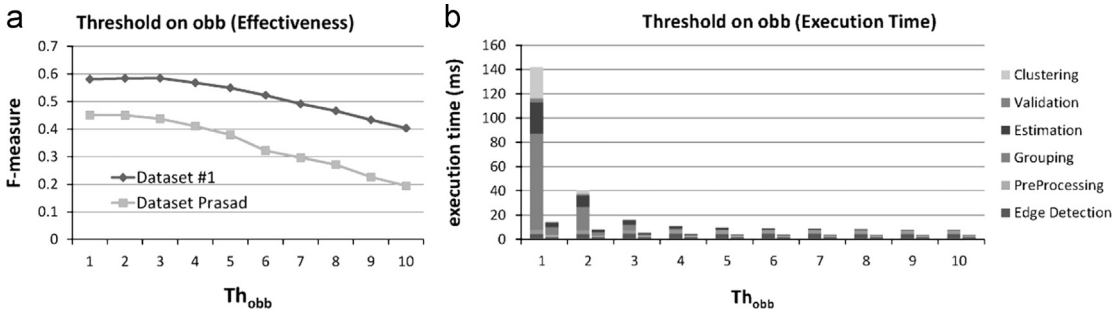

Fig. 11. Performance varying Th_{obb}. (a) *F*-measure. (b) Execution time on Dataset #1 (left column) and Dataset Prasad (right column).

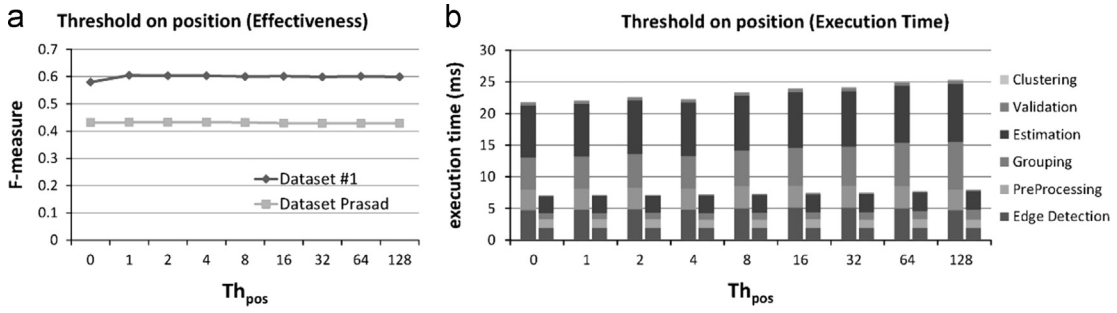

Fig. 12. Performance varying Th_{pos}. (a) *F*-measure. (b) Execution time on Dataset #1 (left column) and Dataset Prasad (right column).

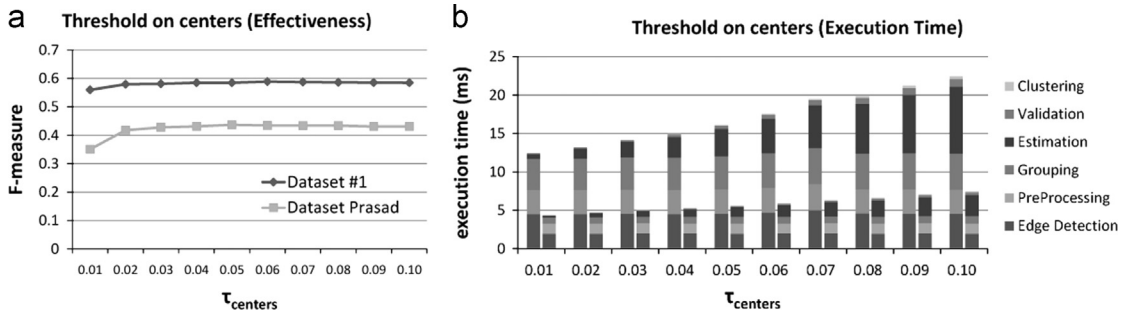

Fig. 13. Performance varying τ_{centers}, where ($Th_{centers} = \tau_{centers} \times$ image diagonal). (a) *F*-measure. (b) Execution time on Dataset #1 (left column) and Dataset Prasad (right column).

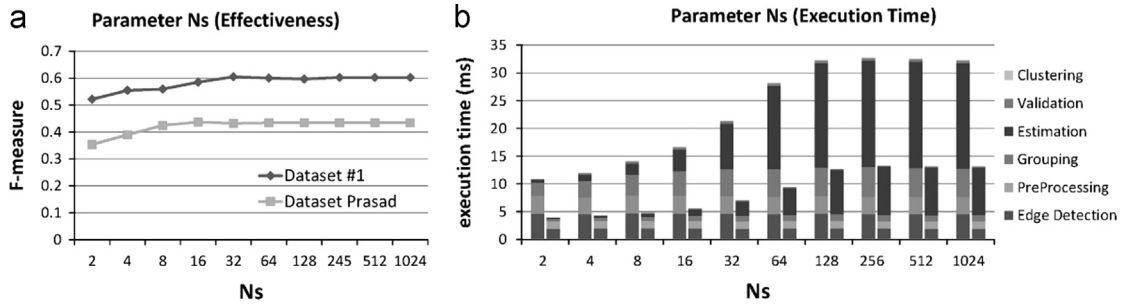

Fig. 14. Effectiveness and execution time varying N_s. (a) *F*-measure. (b) Execution time on Dataset #1 (left column) and Dataset Prasad (right column).

still allow a good estimation of the center. Fig. 14(b) shows that the time used for the evaluation step increases with N_s, while the other steps of the algorithm are basically stable. Moreover, Fig. 14 (a) shows that parameter N_s does not need to be greater than 32 to achieve the top performance. In general, the best tradeoff between accuracy and speed is obtained with $N_s \in [8, 32]$, that allows to avoid unnecessary computation and still to estimate accurately the parameters by using only the points on the selected arcs. Consequently, we set $N_s = 16$.

5. Experimental results

We perform a thorough set of tests of the proposed method, on both synthetic and real images datasets, evaluating its performance and comparing it with other state-of-the-art methods.

5.1. Evaluation metrics

We evaluated the performance of the algorithms in terms of *execution time* and *detection effectiveness*, according to the evaluation methodology of [25]. Detected and ground-truth ellipses are compared according to the following overlap ratio:

$$D = 1 - \frac{\text{count}(\text{XOR}(\mathcal{E}_1, \mathcal{E}_2))}{\text{count}(\text{OR}(\mathcal{E}_1, \mathcal{E}_2))} \tag{31}$$

where $\mathcal{E}_1$ and $\mathcal{E}_2$ are respectively the ground-truth ellipse and a valid detection. If the detected and the ground-truth ellipses have overlap ratio $D > D_0$, with $D_0 = 0.95$ for synthetic images and $D_0 = 0.8$ for real images, then a match is counted (true positive, TP); if a ground-truth ellipse does not have a match with any of the detected ellipses, then a miss is found (false negative, FN); finally, if the detected ellipse does not have a match with any of the ground-truth ellipses, it corresponds to a false positive (FP). According to these definitions the *detection effectiveness* is computed in terms of the *F*-measure.

5.2. Other methods

In order to show the performance of the proposed method, we compare it in terms of both execution time and detection effectiveness against other state-of-the-arts methods. Since the main feature of the proposed method is the ability to run in real–time still achieving good detection results, we selected among the methods reported in Section 1 the fastest and the most effective ones:

- the work of Basca et al. [14], which applies directly the RHT, randomly selecting an initial subset among all possible point pairs;
- the work of Zhang and Liu [16], which, after applying a point selection strategy, estimates the parameters in a decomposed parameter space in multiple steps;
- the work of Libuda et al. [22], which iteratively links small edges into arcs, till it obtains an elliptic shape;
- the work of Prasad et al. [25], which groups arcs according to edge curvature and convexity.

All methods are implemented in C++, except the one of Prasad et al. which is in Matlab. The methods of Zhang and Liu and Basca et al. have been reimplemented according to the respective papers, while the source code of Libuda et al. and Prasad et al. is available on-line.

In the experiments on real datasets (Section 5.5.1) we also evaluate three variants of our method to better motivate our choices:

- instead of relying on the Sobel derivatives to compute the gradient direction for edge pixels, we adopt the Fast Line Extractor of Kim et al. [39];
- instead of estimating the ellipse parameters via the Hough Transform in a decomposed space, we used the direct least square ellipse fitting algorithm of Fitzgibbon et al. [30];

- same as the previous case, but using the unconstrained, non-iterative least square based geometric ellipse fitting method "Ellifit" of Prasad et al. [31].

Their code is available on-line. The Fast Line Extractor is in C++, while the two ellipse fitting algorithms are in Matlab. In order to guarantee a fair comparison, the execution time of the programs in Matlab is scaled by a factor.[2] All experiments were executed without code parallelization on a PC with an Intel Core i7. Our method was also executed on a Samsung Galaxy S2.

5.3. Robustness to rotation, axes ratio and size

To investigate the working limits of the evaluated algorithms with respect to ellipse rotation, axes ratio and axes size, we created two datasets composed of automatically generated synthetic images of size 400×400, each containing a single ellipse without noise.

The first dataset contains 9100 images with fixed parameters $x_c = y_c = 200$ and $A = 100$, with B varying so that the axes ratio B/A ranges from 0 to 1 (step of 0.01) and orientation ρ ranging from 0° to 90° (step of 1°). The results of this evaluation are reported in Fig. 15 so that a pixel in the image is white if the ellipse with corresponding axes ratio (along horizontal axis) and orientation (along vertical axis) has been correctly detected, or black otherwise.

The second dataset contains 10,000 images with fixed parameters $x_c = y_c = 200$ and $\rho = 30^\circ$, with A ranging from 1 to 100 (step of 1) and B varying so that the axes ratio B/A ranges from 0 to 1 (step of 0.01). The results of the evaluation are reported in Fig. 16.

As expected, all the algorithms (i) are basically robust to orientation (no significant vertical asymmetry is noticeable in Fig. 15), (ii) fail when the axes ratio is near to 0, thus when ellipses degenerate into straight lines (leftmost part of graphs in both Figs. 15 and 16), and (iii) have difficulties detecting small ellipses (top part in Fig. 16), i.e. when ellipse boundaries are composed by very few pixels. Ellipses on the first synthetic dataset are large enough and the detection is insensitive to the variation of the parameter Th_{length} (Fig. 15(a)).

The method of Prasad et al. [25] performs extremely good on these datasets, being very robust to ellipse orientation, axes ratio and size, followed by the method of Basca et al. [14], which performs really good in single ellipse, noise-free images. The method of Zhang and Liu [16] fails in the detection of most of the ellipses. This is caused by its selection strategy, which is not able to correctly identify the ellipse center in case of too few samples which do not generate a peak in the 2D accumulator and, consequently, other parameters are wrongly estimated. Moreover, this method has some difficulties in the detection of ellipses with axes ratio near to 1 (rightmost part of Figs. 15(b) and 16(b)), i.e. ellipses degenerating into circles. Since most applications aim at detecting ellipses because they are a perspective transformation of circles, the non-detection of circles could represent an issue. The method of Libuda et al. [22] and ours have almost similar working conditions. Ours, however, appears to be more robust, while the high number of constraints in [22] cannot deal with some configurations of edge points. As shown in Fig. 16(a) varying the parameter Th_{length}, the algorithm is able to detect small ellipses, but at the cost of more execution time (see Section 4.2).

[2] After testing the execution time of a set of functions in C++ and Matlab, we found a scale factor of 50 to be a good average approximation.

5.4. Dataset of Chia et al. [24]

We report the tests on the synthetic datasets proposed in [24]. These datasets include three challenges: occluded ellipses, overlapped ellipses, and ellipses affected by salt-and-pepper noise. Being an edge linking method, we focused on the first two datasets as in [25], because the proposed method needs connected edge pixel to work and consequently is not suited to handle salt-and-pepper noise. Moreover, in case of salt-and-pepper noise, a simple pre-processing such the application of a median filter will remove the noise. The datasets consist of images of size $\lambda \times \lambda$, in which each image contains ηdifferent ellipses, with $\lambda = 300$ and $\eta \in \{4, 8, 12, 16, 20, 24\}$. The parameters of the ellipses, arbitrarily located within the image, are generated randomly with the value of the semi-axes in $[\lambda/30, \sqrt{\lambda^2+\lambda^2}/2]$. Fig. 17 shows the performance of our method, as well as the results obtained by Chia et al. [24] and Prasad et al. [25]. These two methods perform really well on these datasets: their careful selection of edge segments to be split or merged handles consistently cases of occlusion and overlapping. The proposed method, instead, aims at simplifying the arc extraction in order to run in real-time in real world images. This limitation is highlighted by the poor performance obtained on such challenging, yet synthetic datasets.

These results are due to the fact that in such small images the number of edge pixels for each ellipse is very limited. In order to ground this motivation to our poor performance, the original datasets have been upscaled by multiplying the parameter λ by a factor of 1, 2, 3, 4. As a consequence, ellipses are larger but still present the same challenges as in the original datasets. The results shown in Fig. 18 demonstrate that our method significantly improves the performance on the new datasets, confirming that in the presence of enough information the proposed method is able to detect very well overlapped or occluded ellipses.

5.5. Real datasets

Since the goal of the proposed method is to work in real time in real scenarios, we tested the methods on three datasets containing real world images: a portion of the dataset used in Prasad et al. [25] called "Dataset Prasad" hereinafter, and two datasets we created collecting a total of 1029 images.

Dataset Prasad is composed by the portion of data that are still available on-line of the dataset used in [25], which consists of 198 images out of the original 400.

Our Dataset #1 is composed of 400 real images containing elliptic shapes, collected from MIRFlickr and LabelMe repositories. The images from the first repository are high quality and mainly focused on a single object, while images from the second one have lower resolution, are more noisy and represent scenes containing different objects.

The major reason for the development of the proposed method is the capability to run in real–time on embedded devices such as smart-phones. As a consequence, we created Dataset #2 collecting several videos using a Samsung Galaxy S2 and selecting a total amount of 629 frames at the resolution of 640x480, typical of most smart-phone applications. Ellipse detection on this kind of images is very challenging due to varying lighting conditions and images blurred by motion and autofocus.

All images in Dataset #1 and Dataset #2 have been manually annotated and are publicly available on-line[3] for future comparisons. Regarding Dataset Prasad we relied on the given ground truth.

Figs. 19 and 20 show some statistics about the datasets. On one hand, in Dataset Prasad the number of small ellipses (i.e. with

[3] http://imagelab.ing.unimore.it/imagelab/ellipse/ellipse_dataset.zip.

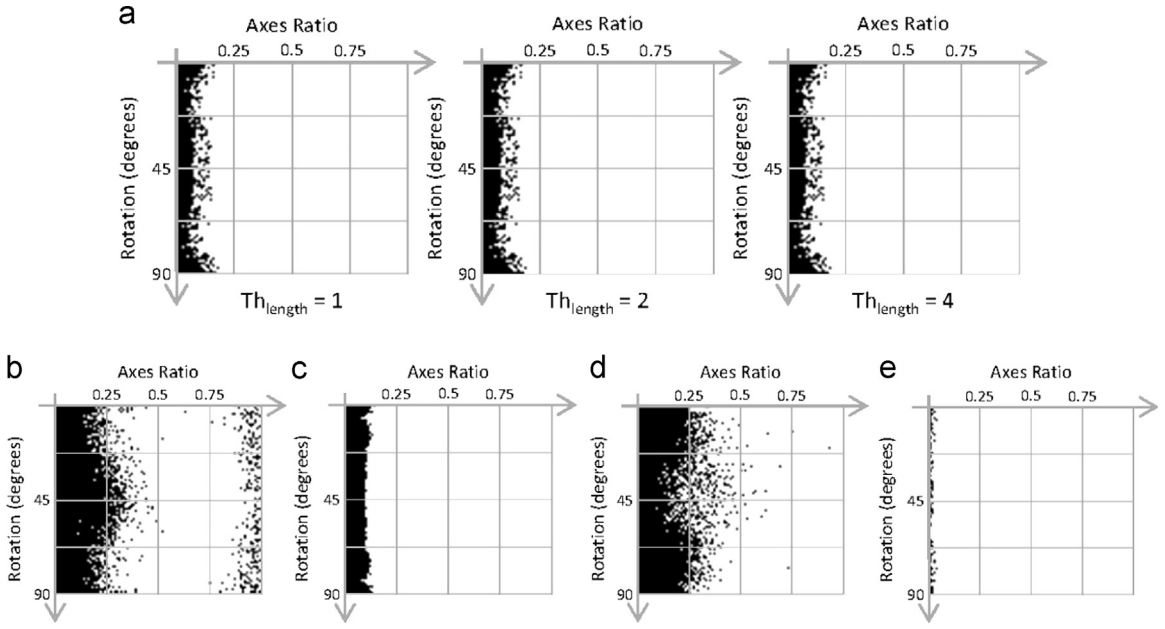

Fig. 15. Working conditions, with respect to rotation (vertical axis, from 0° to 90°) and axes ratio B/A (horizontal axis, from 0 to 1). (a) Ours, varying Th_{length}, (b) Zhang and Liu [16], (c) Basca et al. [14], (d) Libuda et al. [22], and (e) Prasad et al. [25].

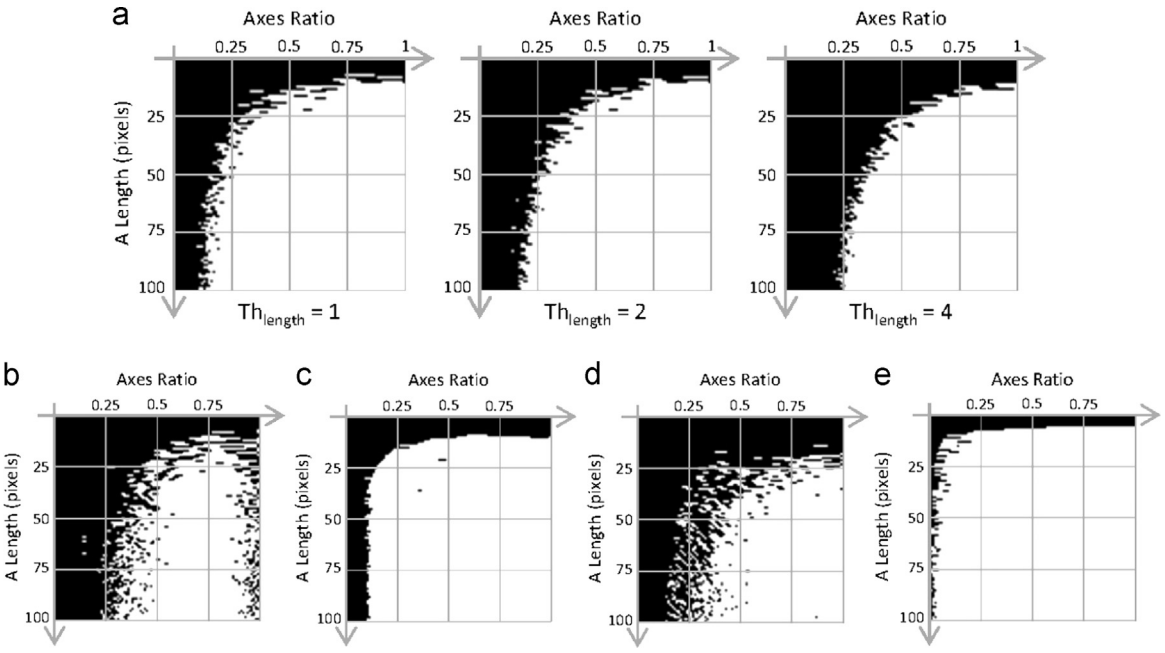

Fig. 16. Working conditions, with respect to major semi-axis length (vertical axis, from 1 to 100) and axes ratio B/A (horizontal axis, from 0 to 1). (a) Ours, varying Th_{length}, (b) Zhang and Liu [16], (c) Basca et al. [14], (d) Libuda et al. [22], and (e) Prasad et al. [25].

semi-major axis length shorter than 20 pixels) is very high compared to Dataset #1 and Dataset #2. On the other hand, in Dataset #1 and Dataset #2 most of the images contain few ellipses, while in Dataset Prasad the number of ellipses per image is more uniformly distributed.

5.5.1. *Results on real datasets*

We evaluated the effectiveness of the selected methods on the three datasets collecting the *F*-measure varying the threshold on the score (Th_{score}) of the detected ellipses for each image, as reported in Fig. 21 for Dataset Prasad, Fig. 22 for Dataset #1 and Fig. 23 for Dataset #2. We reported the highest *F*-measure value in Table 2. Our method results to be very effective in the detection of ellipses in real images.

In order to evaluate the speed, also reported in Table 2, for Dataset Prasad and Dataset #1 we collected the execution time while running all methods on a PC, while for Dataset #2 we collected the execution time while running our method on a smart-phone, namely a Samsung Galaxy S2. For this purpose, we developed an Android and OpenCV application which calls the C++ function for detecting ellipses via the Java Native Interface (JNI). Since, it was impossible to test the accuracy in real–time directly on the smart-phone due to the lack of a ground truth, all frames have been annotated and the effectiveness test was performed off-line on the PC. The execution time of our method, instead, was collected while running the application directly on the smart-phone filming the same scenes.

Dataset Prasad. We show the execution time breakdown for each processing step for all the compared algorithms in Table 3.

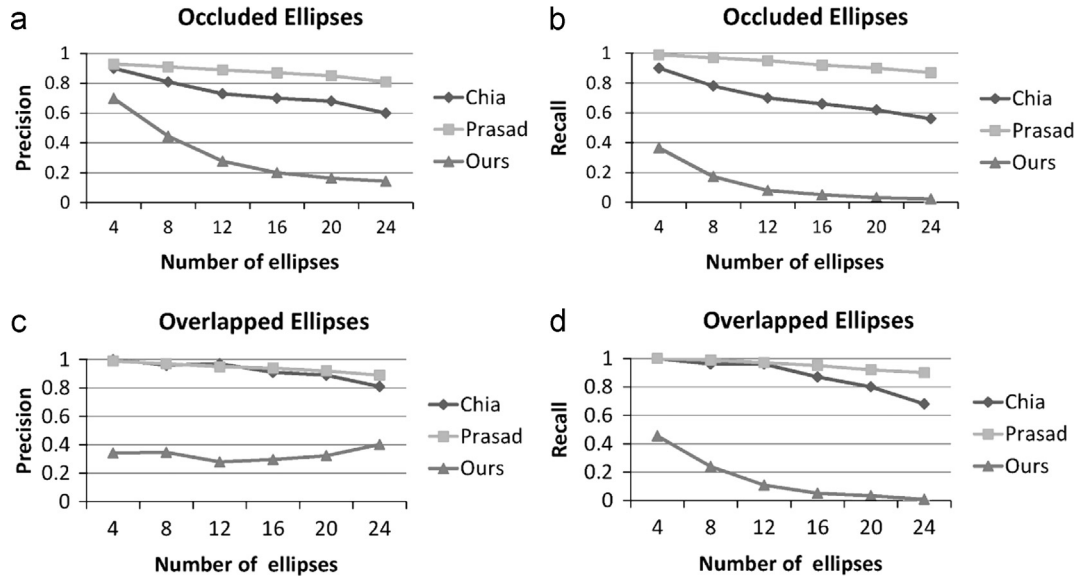

Fig. 17. Evaluation on Dataset Chia [24]. (a) Precision. (b) Recall. (c) Precision. (d) Recall.

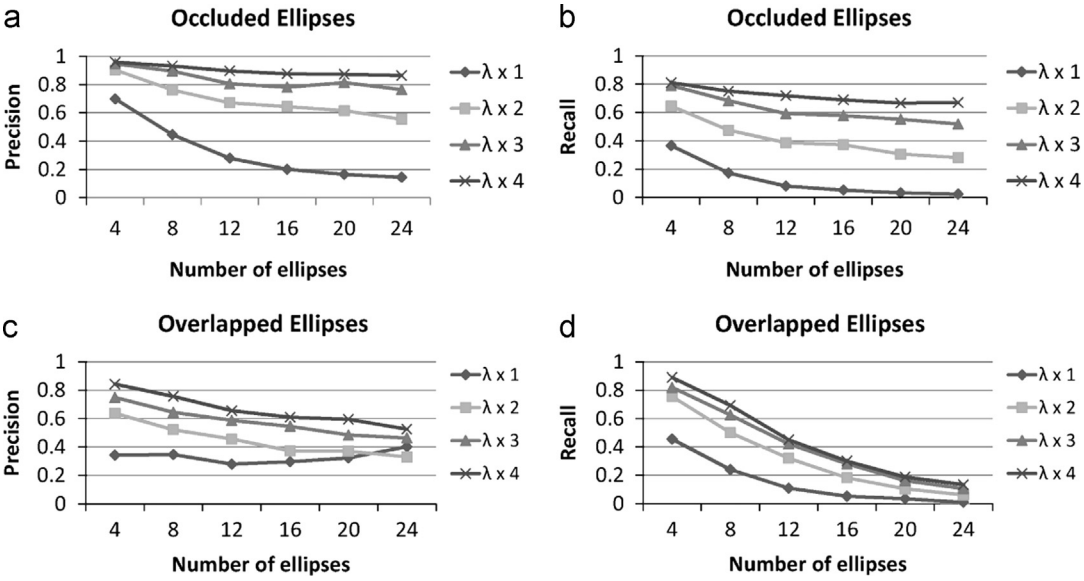

Fig. 18. Evaluation on Dataset Chia [24] of the proposed method increasing the value of λ. (a) Precision. (b) Recall. (c) Precision. (d) Recall.

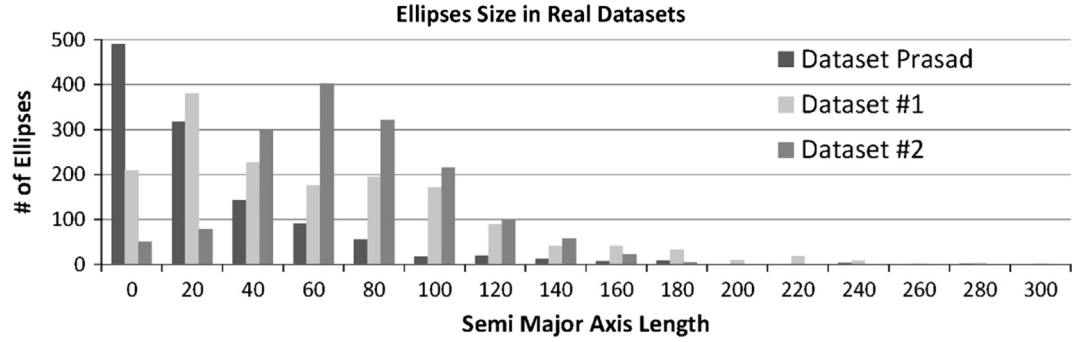

Fig. 19. Number of ellipses per length of the major semi-axes in real datasets.

The method of Prasad et al. [25] performs the best in terms of effectiveness (*F*-measure of 44.18%), but results to be quite slow. Most of the time is spent in the grouping step; also its clustering method based on the overlap ratio is very slow. Our method is the second one in terms of effectiveness (*F*-measure of 43.70%, very close to Prasad et al. [25]) and is the fastest with an execution time of 5.56 ms. Our poorer performance is mainly due the fact that this dataset contains a large amount of small ellipses, as shown in Fig. 19. Due to its working limits and to the trade-off considered in the parameter setting, our method is not able to detect about the

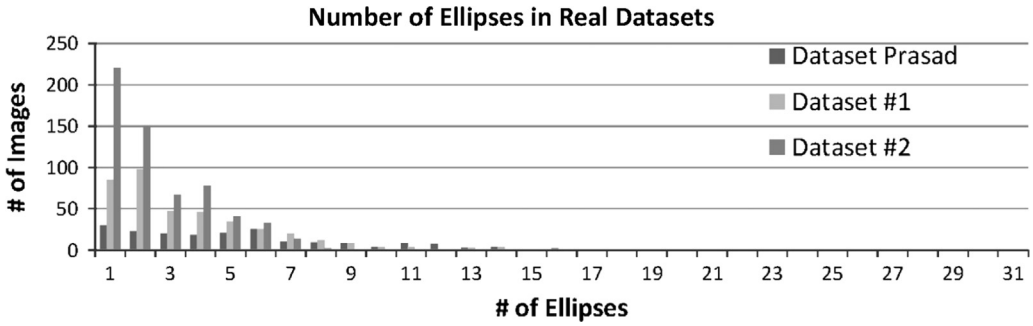

Fig. 20. Images per number of ellipses in real datasets.

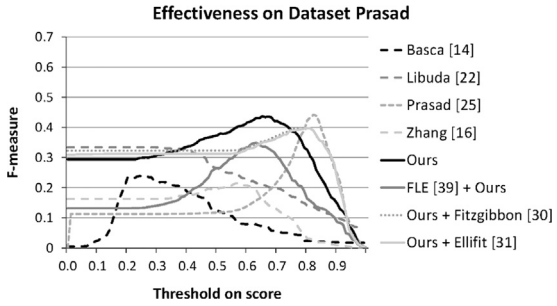

Fig. 21. Graph reporting the *F*-measure varying Th_σ on Dataset Prasad.

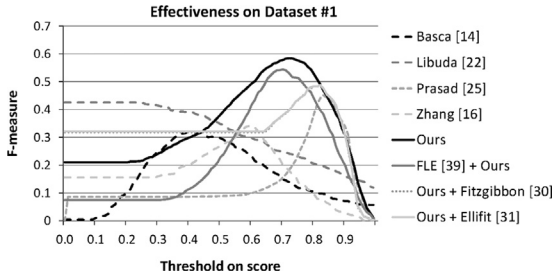

Fig. 22. Graph reporting the *F*-measure varying Th_σ on Dataset #1.

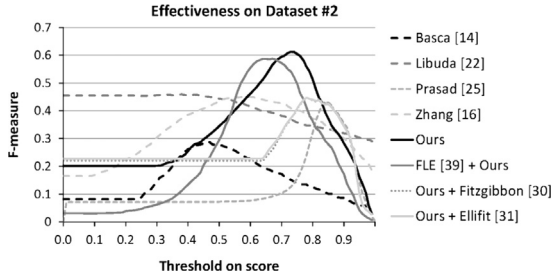

Fig. 23. Graph reporting the *F*-measure varying Th_σ on Dataset #2.

Table 3
Execution time breakdown on Dataset Prasad.

Step of the algorithm	[14]	[22]	[25]	[16]	Ours	[39] +Ours	Ours+ [30]	Ours+ [31]
Edge detection	2.28	2.21	2.30	2.33	1.96	2.29	2.36	2.36
Pre-processing	1.85	2.42	30.97	1.95	1.33	4.20	1.44	1.44
Grouping	0.00	2.44	67.48	0.04	0.84	0.81	1.47	1.47
Estimation	119.92	0.78	1.68	427.60	1.26	0.99	1.09	1.06
Validation	0.00	0.00	0.13	0.00	0.14	0.09	0.30	0.31
Clustering	10.92	0.00	55.76	0.02	0.03	0.01	0.36	0.78
Total	134.97	7.85	158.32	431.95	**5.56**	8.40	7.02	7.43

38% of the ellipses present in this dataset. However, the reported *F*-measure value demonstrates that our method is very effective in the detection of mid-sized or large ellipses (where the major semi axis *A* is larger than about 10–15 pixels). Moreover, this dataset contains also ellipses that are so distorted that are not detected by any other methods, as depicted in third line of Fig. 24. The variants of our method that rely on an ellipse fitting algorithm perform also quite well, while the implementation with the Fast Line Extractor is less effective. The method of Libuda et al. [22] has good execution time but quite low detection performance. The methods of Basca et al. [14] and Zhang and Liu [16] have very poor overall performance: most of the time is spent to estimate the parameters for a large number of edge pixel combinations.

Dataset #1. In Table 4 we report the execution time breakdown for all the algorithms. The two methods that work directly on single edge points, i.e. Zhang and Liu [16] and Basca et al. [14], have again the worst performance. The selection strategy of Zhang and Liu requires more computation, but leads to more accurate solutions with respect to random point selection of Basca et al. that suffers the amount of noise present in real images. The method of Libuda et al. [22] confirms to be very fast. The method of Prasad et al. [25] has good performances, but its arc extraction and grouping procedure are very time demanding. Our algorithm and its variants achieve the best performances in terms of both effectiveness and execution time.

Dataset #2. The results (see Table 2) regarding the effectiveness on Dataset #2 confirm the consideration reported for the other datasets. The method of Prasad et al. [25] is quite accurate, but not as much as in synthetic scenarios or its own dataset. The method of Libuda et al. [22] confirms average results, and the methods based on single edge pixels [14,16] still have very low performance. Our method has the highest *F*-measure value, and its variants achieve good performance as well. The average execution time on the smart-phone is 45.82 ms, demonstrating the real-time performance of the proposed method.

Some results of the tested methods on the three datasets are depicted in Figs. 24–26, respectively.

5.6. Selection criteria

The goal of the selection strategy is to discard as soon as possible those triplets whose arcs do not lie on the same ellipse boundary. This impacts both speed and detection effectiveness, avoiding further computation for triplets which are actually false detections. As clarified in Section 3.2.1, each step of the selection strategy selects a narrower subset: $\mathcal{T}^3 \subseteq \mathcal{T}^2 \subseteq \mathcal{T}^1 \subseteq \mathcal{T}^0$. The size of each subset and its ratio with respect to $\mathcal{T}^0$ are reported in Table 5. These values, computed as the average for all images in Dataset Prasad and Dataset #1, demonstrate the effectiveness of the selection strategy criteria.

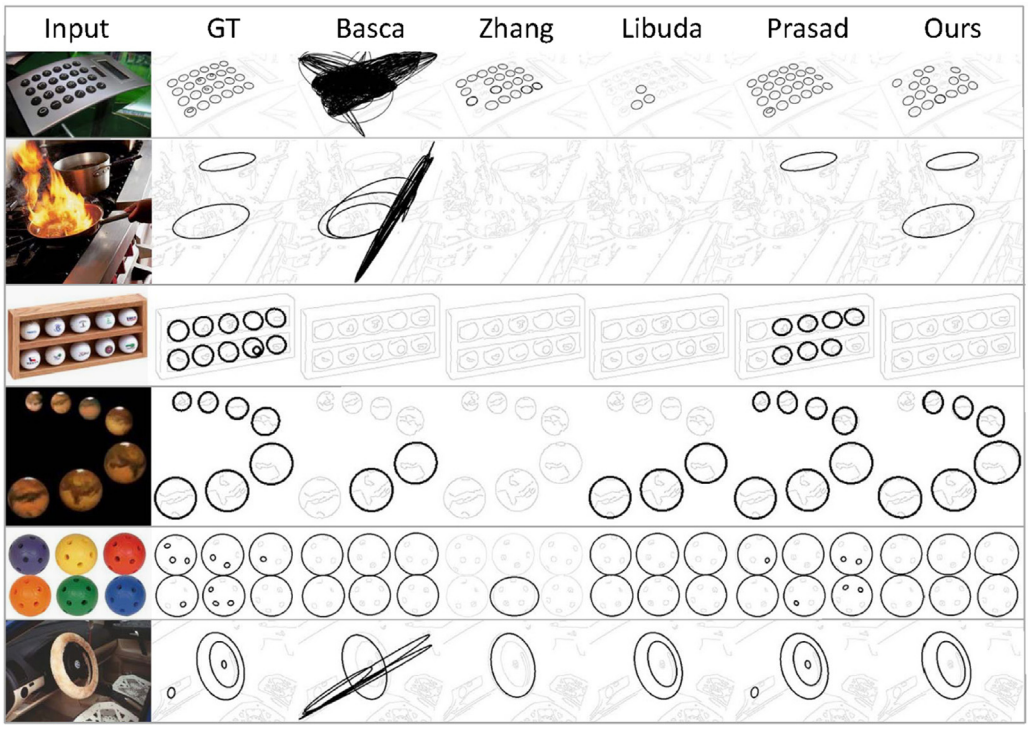

Fig. 24. Results on Dataset Prasad.

Table 4
Execution time breakdown on Dataset #1.

	[14]	[22]	[25]	[16]	Ours	[39] +Ours	Ours+ [30]	Ours + [31]
Edge detection	5.93	5.92	4.66	5.23	4.54	5.09	5.45	5.45
Pre-processing	5.77	5.47	102.81	4.68	3.14	10.22	3.11	3.11
Grouping	0.00	6.44	366.30	0.33	4.23	5.94	4.29	4.29
Estimation	611.05	1.11	4.48	5581.25	3.56	4.29	9.57	6.80
Validation	0.00	0.00	0.32	0.00	0.40	0.43	0.34	0.41
Clustering	61.56	0.00	606.28	0.04	0.07	0.01	0.29	0.36
Total	684.31	18.95	1084.85	5591.55	**15.96**	25.99	23.06	20.44

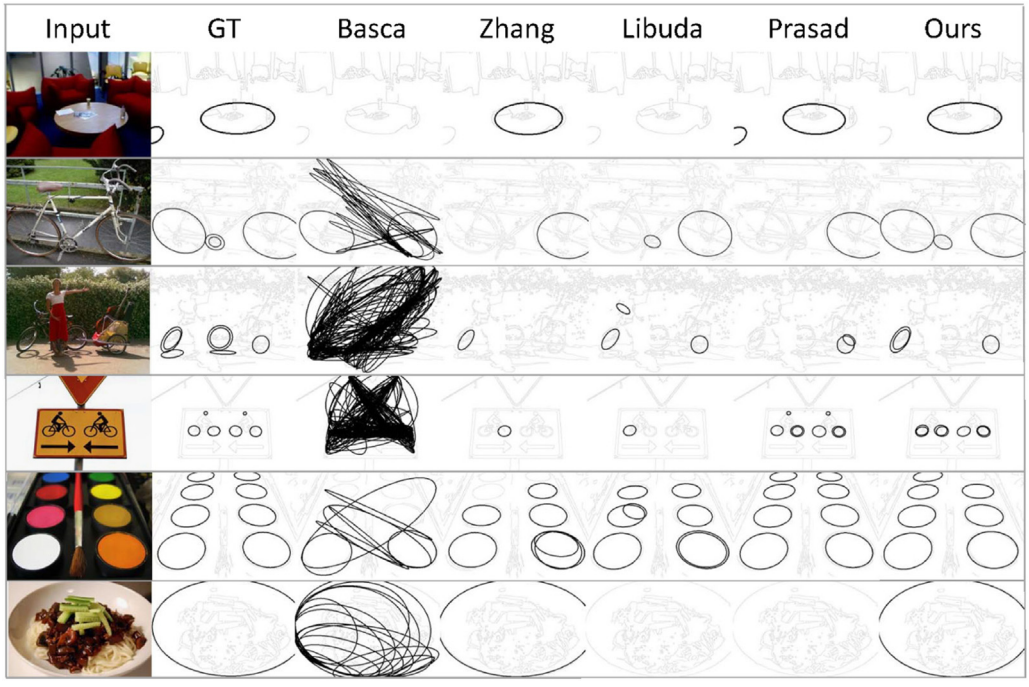

Fig. 25. Results on Dataset #1.

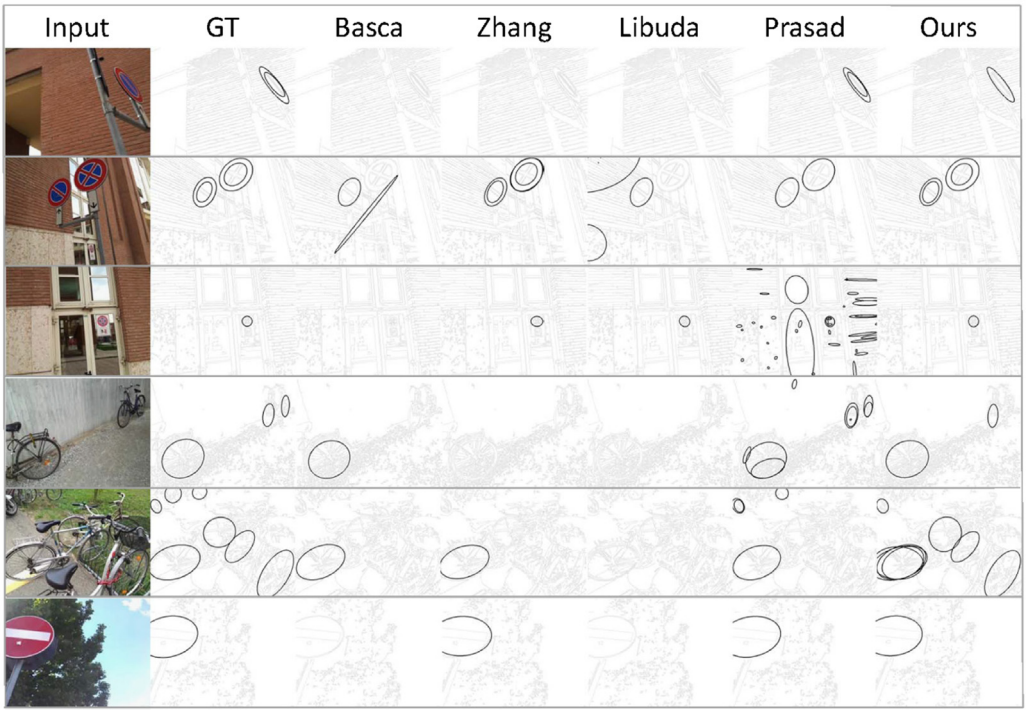

Fig. 26. Results on Dataset #2.

Table 5
Average number of triplets after applying the constraints.

Set	Dataset Prasad		Dataset #1	
	Avg. # triplets	% triplets	Avg. # triplets	% triplets
$\mathcal{T}^0$	44,514	100.00	261,019	100.00
$\mathcal{T}^1$	15,600	35.04	86,221	33.03
$\mathcal{T}^2$	2702	6.07	11,450	4.38
$\mathcal{T}^3$	112	0.25	323	0.12

5.7. Known limitations of our method

Our method relies on a very simple procedure to generate arcs. On one hand this guarantees a major speed-up, on the other hand it presents some drawbacks. The method is not able to split correctly arcs that presents inflexion or junction points, and it splits well shaped arcs spanning trough different quadrants. Small ellipses or ellipses with fragmented boundary are difficult to detect because arcs may result to be too short or without enough curvature. Also, when the ellipses have very low axes ratio arcs may be considered as straight lines and discarded as well. These limitations are highlighted by the poor performance on the synthetic datasets of Chia et al. in Section 5.4.

The selection strategy allows to significantly speed up the grouping procedure. However, it assumes that an ellipse has at least three arcs in different quadrants. This assumption does not always hold, and consequently some kind of occluded ellipses, such as well-shaped semi-ellipses, cannot be detected.

6. Conclusions

In this paper we proved that a very fast and accurate ellipse detection is feasible also with limited hardware resources as in the case of smart-phones. The main point is to shift the focus of interest from edge points to arcs (or parts of arcs) and, instead of providing an exhaustive search, to start selecting only arcs compatible with an elliptical shape. Our approach has been extensively analyzed (with particular focus on the influence of the most critical parameter) and compared with four state-of-the-art methods, resulting superior than them on real images in terms of trade-off between detection effectiveness and execution time.

Despite its performance, our approach is based on several assumptions which can lower its effectiveness compared with other methods in particular conditions. However, based on our experiments, the cases on which these assumptions do not hold are not frequent in real images and the overall improvement in terms of efficiency is worth this slight detection loss, especially when aiming, as in this paper, to a real-time implementation on a mobile device.

Conflict of interest

None declared.

Acknowledgments

This work has been partially funded by Regione Emilia Romagna under the Tecnopolo funding scheme.

References

[1] T. Cooke, A fast automatic ellipse detector, in: 2010 International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2010, pp. 575–580. doi: http://dx.doi.org/10.1109/DICTA.2010.102.
[2] A. Soetedjo, K. Yamada, Fast and robust traffic sign detection, in: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 2, 2005, pp. 1341–1346. doi: http://dx.doi.org/10.1109/ICSMC.2005.1571333.
[3] C. Teutsch, D. Berndt, E. Trostmann, 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.
[4] L. Świrski, A. Bulling, N. Dodgson, Robust real-time pupil tracking in highly off-axis images, in: Proceedings of the Symposium on Eye Tracking Research and Applications, ACM, 2012, pp. 173–176.

[5] G. Hua, Y. Fu, M. Turk, M. Pollefeys, Z. Zhang, Introduction to the special issue on mobile vision, Int. J. Comput. Vis. 96 (3) (2012) 277–279.
[6] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, Real-time detection and tracking for augmented reality on mobile phones, IEEE Trans. Vis. Comput. Gr. 16 (3) (2010) 355–368.
[7] P. Fockler, T. Zeidler, B. Brombach, E. Bruns, O. Bimber, Phoneguide: museum guidance supported by on-device object recognition on mobile phones, in: Proceedings of the 4th international conference on mobile and ubiquitous multimedia, ACM, 2005, pp. 3–10.
[8] M. Rahman, J. Ren, N. Kehtarnavaz, Real-time implementation of robust face detection on mobile platforms, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009, IEEE, 2009, pp. 1353–1356.
[9] C. Guida, D. Comanducci, C. Colombo, Automatic bus line number localization and recognition on mobile phones—a computer vision aid for the visually impaired, in: Image Analysis and Processing–ICIAP 2011, Springer, 2011, pp. 323–332.
[10] R.A. Mclaughlin, Randomized Hough Transform: Improved Ellipse Detection with Comparison, Technical Report, Univ. of Western Australia, 1998.
[11] W. Lu, J. Tan, Detection of incomplete ellipse in images with strong noise by iterative randomized hough transform (irht), Pattern Recognit. 41 (4) (2008) 1268–1279.
[12] Y. Xie, Q. Ji, A new efficient ellipse detection method, in: ICPR, 2002, pp. 957–960.
[13] A. Chia, M. Leung, H.-L. Eng, S. Rahardja, Ellipse detection with Hough transform in one dimensional parametric space, in: Proceedings of IEEE International Conference on Image Processing, vol. 5, 2007, pp. V-333–V-336.
[14] C. Basca, M. Talos, R. Brad, Randomized Hough transform for ellipse detection with result clustering, in: The International Conference on Computer as a Tool, 2005. EUROCON 2005, vol. 2, 2005, pp. 1397–1400.
[15] A.S. Aguado, E. Montiel, M.S. Nixon, On using directional information for parameter space decomposition in ellipse detection, Pattern Recognit. 29 (3) (1996) 369–381.
[16] S.-C. Zhang, Z.-Q. Liu, A robust, real-time ellipse detector, Pattern Recognit. 38 (2) (2005) 273–287.
[17] Y. Lei, K.C. Wong, Ellipse detection based on symmetry, Pattern Recognit. Lett. 20 (1) (1999) 41–47.
[18] C.-T. Ho, L.-H. Chen, A fast ellipse/circle detector using geometric symmetry, Pattern Recognit. 28 (1) (1995) 117–124.
[19] P.-Y. Yin, L.-H. Chen, New method for ellipse detection by means of symmetry, J. Electron. Imaging 3 (1) (1994) 20–29.
[20] W.-Y. Wu, M.-J.J. Wang, Elliptical object detection by using its geometric properties, Pattern Recognit. 26 (10) (1993) 1499–1509.
[21] E. Kim, M. Haseyama, H. Kitajima, Fast and robust ellipse extraction from complicated images, in: Proceedings of IEEE International Conference on Information Technology and Applications, 2002.
[22] L. Libuda, I. Grothues, K.-F. Kraiss, Ellipse detection in digital image data using geometric features, in: J. Braz, A. Ranchordas, H. Arajo, J. Jorge (Eds.), Advances in Computer Graphics and Computer Vision, Communications in Computer and Information Science, vol. 4, Springer, Berlin, Heidelberg, 2007, pp. 229–239.
[23] F. Mai, Y. Hung, H. Zhong, W. Sze, A hierarchical approach for fast and robust ellipse extraction, Pattern Recognit. 41 (8) (2008) 2512–2524.
[24] A.-S. Chia, S. Rahardja, D. Rajan, M. Leung, A split and merge based ellipse detector with self-correcting capability, IEEE Trans. Image Process. 20 (7) (2011) 1991–2006.
[25] D.K. Prasad, M.K. Leung, S.-Y. Cho, Edge curvature and convexity based ellipse detection method, Pattern Recognit. 45 (9) (2012) 3204–3221.
[26] T.M. Nguyen, S. Ahuja, Q. Wu, A real-time ellipse detection based on edge grouping, in: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 2009, pp. 3280 –3286.
[27] Z.-Y. Liu, H. Qiao, Multiple ellipses detection in noisy environments: a hierarchical approach, Pattern Recognit. 42 (11) (2009) 2421–2433.
[28] K. Hahn, S. Jung, Y. Han, H. Hahn, A new algorithm for ellipse detection by curve segments, Pattern Recognit. Lett. 29 (13) (2008) 1836–1841.
[29] Y. Qiao, S. Ong, Arc-based evaluation and detection of ellipses, Pattern Recognit. 40 (7) (2007) 1990–2003.
[30] A. Fitzgibbon, M. Pilu, R. Fisher, Direct least square fitting of ellipses, IEEE Trans. Pattern Anal. Mach. Intell. 21 (5) (1999) 476–480.
[31] D.K. Prasad, M.K. Leung, C. Quek, Ellifit: an unconstrained, non-iterative, least squares based geometric ellipse fitting method, Pattern Recognit. 46 (5) (2013) 1449–1465.
[32] M. Fornaciari, A. Prati, Very fast ellipse detection for embedded vision applications, in: 2012 Sixth International Conference on Distributed Smart Cameras (ICDSC), 2012, pp. 1–6.
[33] M. Fornaciari, R. Cucchiara, A. Prati, A mobile vision system for fast and accurate ellipse detection, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2013, pp. 52–53.
[34] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698.
[35] H. Freeman, R. Shapira, Determining the minimum-area encasing rectangle for an arbitrary closed curve, Commun. ACM 18 (7) (1975) 409–413.
[36] J. Matousek, Randomized optimal algorithm for slope selection, Inf. Process. Lett. (1991) 183–187.
[37] A. Fernandes, A Correct Set of Equations for the Real-Time Ellipse Hough Transform Algorithm, Technical Report, 2009.
[38] D. Prasad, M.K.H. Leung, Clustering of ellipses based on their distinctiveness: an aid to ellipse detection algorithms, in: 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 8, 2010, pp. 292–297. doi: http://dx.doi.org/10.1109/ICCSIT.2010.5564932.
[39] E. Kim, M. Haseyama, H. Kitajima, Fast line extraction from digital images using line segments, Syst. Comput. Jpn. 34 (10) (2003) 76–89.
[40] N. Guil, E. Zapata, Lower order circle and ellipse Hough transform, Pattern Recognit. 30 (10) (1997) 1729–1744.
[41] R. Chan, W.-C. Siu, Fast detection of ellipses using chord bisectors, in: 1990 International Conference on Acoustics, Speech, and Signal Processing, 1990. ICASSP-90, 1990, vol. 4, pp. 2201–2204. http://dx.doi.org/10.1109/ICASSP.1990.115997.
[42] P.-K. Ser, W.-C. Siu, Novel detection of conics using 2-d hough planes, IEE Proc. Vis. Image Signal Process. 142 (5) (1995) 262–270.
[43] H. Yuen, J. Illingworth, J. Kittler, Detecting partially occluded ellipses using the Hough transform, Image Vis. Comput. 7 (1) (1989) 31–37.

Michele Fornaciari received the M.S. degree in 2009 and is currently a Ph.D. student at the International Doctorate School in Information and Communication Technologies organized by the University of Modena and Reggio Emilia, Italy. He is also part of the SOFTECH research center in Modena, Italy. His research interests focus on video surveillance and mobile vision.

Andrea Prati received the M.S. degree in 1998 and the Ph.D. degree in 2001. He is an associate professor currently on the Department of Design and Planning in Complex Environments of the University IUAV of Venice. He collaborates on research projects at the regional, national, and European level. He is an author or coauthor of more than 100 papers in national and international journals and conference proceedings, he has been an invited speaker, organizer of workshops and journal's special issues, and a reviewer for many international journals in the field of computer vision and multimedia. He has also been the program chair of the 14th International Conference on Image Analysis and Processing (ICIAP), held in September 2007 in Modena, and of the ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC) 2011 and 2012. He is a senior member of the IEEE, and a member of the IEEE Computer Society, the ACM, and the GIRPR (IAPR Italy).

Rita Cucchiara received the M.S. degree in 1989 and the Ph.D. degree in 1992. She has been a full-time professor at the University of Modena and Reggio Emilia since 2005. Since 2008, she has been deputy dean of the Faculty of Engineering and heads the Imagelab laboratory (http://imagelab.ing.unimore.it). Since 2010, she has been Scientific Responsible of the ICT Platform in the High Technology Network of Emilia Romagna. Her research interests regard computer vision, pattern recognition, and multimedia systems. She is responsible of many research projects (national and EU projects), especially in people surveillance, urban security, and human-centered multimedia search of images and videos. She is the author of more than 200 papers in journals and international proceedings, and she acts as a reviewer for several international journals. She is in the EB of the MTAP and MVA journals, chair of several workshops on surveillance, track chair for ICPR 2012, and general chair of ICIAP 2007 and AI*IA 2009. She is a member of the IEEE, the IEEE Computer Society, and the ACM. Since 2006, she has been a fellow of the IAPR.