Received January 12, 2021, accepted January 29, 2021, date of publication February 3, 2021, date of current version February 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3056795

Multiscale Detection of Circles, Ellipses and Line Segments, Robust to Noise and Blur

ONOFRE MARTORELL, ANTONI BUADES, AND JOSE LUIS LISANI
Department of Mathematics and Computer Science, Institute of Applied Computing and Community Code (IAC3), Universitat de les Illes Balears, 07122 Palma, Spain

Corresponding author: Onofre Martorell (o.martorell@uib.cat)

The authors acknowledge the Ministerio de Ciencia, Innovación y Universidades (MCIU), the Agencia Estatal de Investigación (AEI) and the European Regional Development Funds (ERDF) for its support to the project TIN2017-85572-P.

ABSTRACT This paper proposes a basic taxonomy of image contours. Our goal is to classify smooth curves into five categories, namely, circles, ellipses, line segments, arcs of circles and arcs of ellipses. These geometrical structures have been chosen as they serve as input of many computer vision tasks. The proposed strategy is applied on a set of initial disjoint contours, which are grouped together to form the aforementioned structures. These, in turn, are validated using an a contrario approach that guarantees a reduced number of false detections. The use of a multiscale strategy permits the detection at different resolution levels, which makes the method robust to noise and blur.

INDEX TERMS Line segment detection, circle detection, ellipse detection, a contrario validation.

I. INTRODUCTION

The detection of line segments, circular arcs, elliptical arcs, as well as complete circles and ellipses, is an important topic in the fields of image processing and computer vision. Detecting these structures in digital images is a challenging task due to the variety of scenarios where they appear. An ideal detector should be able to identify multiple instances of the sought structure in the same image irrespective of their size, work with synthetic, natural and noisy images, have high detection rates and good accuracy, and produce few or no false detections.

The detection of these geometrical structures is used as a component in many other high-level tasks, such as iris and pupil detection [1], [2], traffic sign detection [3], [4], 3D calibration [5], automatic inspection of manufactured products [6] or autonomous driving in unknown environments [7]. The degree of blur or noise may differ in each situation, hence, designing a reliable fully automatic general-purpose detector for a certain geometric primitive is a challenging task.

Algorithms for ellipses and circles detection can be divided into two main categories: pixel-based methods, where edge pixels are grouped together using the Hough Transform (HT) or other clustering method (e.g. RANSAC); and contour-based methods, where edge curves are used as input of the detection algorithm. The former are slow (since many edge pixels are present in the images) and depend on many parameters, while the latter are faster and more accurate since the validation of an ellipse/circle candidate requires that several edge curves belong to its support, and not just a bunch of (possibly spatially unconnected) edge points.

Our detector falls into this second category, but four important aspects distinguish it from previously published works: 1) It takes as input a set of smooth curves obtained using a biologically-inspired model of good continuation of the image edges [8], [9]. This avoids the need of edge-linking or curve-parsing pre-processing steps. 2) The input curves are grouped together to form circles or ellipses using a RANSAC-like approach that permits the union of sparsely located contours. 3) A multiscale strategy is used to combine the detections obtained at different resolutions, which makes the method robust to blur and noise. 4) We propose a common strategy for the detection of several geometrical structures, not only circles and ellipses, but also arcs of circles, line segments and arcs of ellipses.

It must be noted that the three last features of the proposed method could be applied on any set of input contours, provided that they have already been parsed into smooth curves. In this sense, the use of the technique described in [8], [9] for contour extraction may be considered as optional. However, the fact that the contours extracted with this technique are already parsed into different curves depending on their smoothness and connectivity makes them particularly useful for our purposes. Fig. 1 compares the contours obtained by

The associate editor coordinating the review of this manuscript and approving it for publication was Mingbo Zhao.

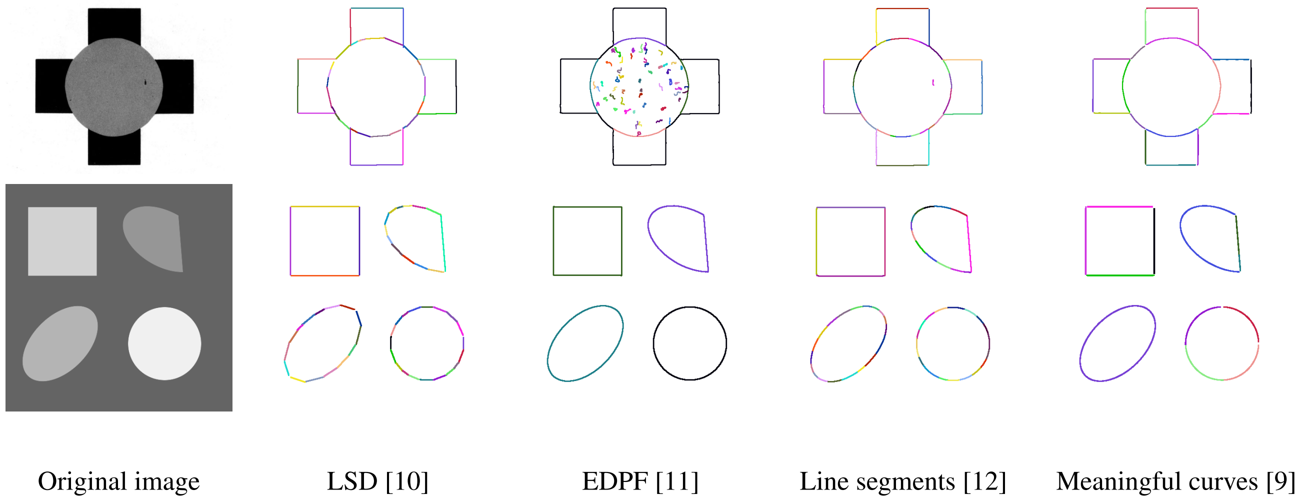


FIGURE 1. From left to right: original image, segments detected by LSD [10], contours detected by EDPF [11], which are the ones used by EDcircles [12] and [13], the line segments from Kovesi's algorithm [14] used as input by Mai *et al.* [15] or [16], among others and contours detected by [9] which we take as input for our detector. Observe that the method in [9] directly provides a segmentation of the contours in circles, ellipses, line segments and arcs of curves. However, the algorithm doesn't classify the curves in such categories. The current paper does.

different edge detectors used by state-of-the art contour-based methods for circle/ellipse detection: LSD [10] (used in [17]–[20]), EDPF [11] (used in [12]), Kovesi method [14] (used in [15] and [16], among others) and the one used by our method [9]. Each extracted curve is displayed with a different color.

Observe that smooth curves are split into several pieces by LSD [10] and Kovesi [14], which implies that a merging step is needed to group them. On the other hand, EDPF [11] extracts as a single curve contours belonging to different geometrical structures (arcs of circles and segments in the example), which means that a splitting step is necessary to parse them. The detector [9] represents entire regular contours by a single curve, parsed at corners and T-junctions. In very simple cases (see Fig. 1) a validation step is all that is needed to classify the obtained contours as circles, ellipses or line segments. The need for the combination of partial detections into a few geometrical structures arises in more complex situations (Fig. 2). The current paper describes such strategy and shows that the obtained results are competitive with respect to the state-of-the-art.

Fig. 2 illustrates the output of our geometry analysis for a particular example. Some of the initial curves at different scales are grouped into circles and ellipses, the rest are combined to form line segments, circular arcs or elliptical arcs. Curves that do not fall into any of these categories are not displayed in the final output. Different colors are used to represent each type of curve.

As mentioned above, an important contribution of the proposed strategy for the obtention of ellipses, circles and line segments, with respect to previously published works, is its multiscale nature. Geometrical structures appear in natural images with a certain degree of blur that depends on the camera resolution and acquisition conditions. Most detection methods extract the contours at a single image resolution, which may lead to missed detections. By combining the detection results obtained for contours extracted at different resolutions we are able to detect geometrical structures even if the contours are slightly blurred or noisy.

In the following sections the proposed method shall be described in detail and its results compared with the state-of-the-art. The paper is organized as follows: in Section II the more relevant literature on circles, ellipses and line segments detection is reviewed; in Section III a complete overview of the proposed method is given, and a detailed description of all its steps is provided in Sections IV and V; several experimental results and comparisons with existing methods are performed in Section VI; finally, the conclusions of our work are presented in Section VII.

II. STATE-OF-THE-ART

Many methods have been proposed in the literature for the detection of different geometrical structures in the images. In this section we review the most relevant works on circle detection, ellipse detection and line segments detection. We shall classify these methods into two big categories, namely, pixel-based approaches and contour-based approaches. The former take as input unconnected edge-pixels (usually extracted using Canny's edge detector or a similar technique) and try to sort out which ones of these pixels belong to the sought geometrical structure. The latter use chained sets of pixels describing the image's contours and try to combine them to form the desired structures.

The most popular line segment detectors fall into the category of pixel-based methods. The first algorithms used the Hough Transform [21], [22], but they were slow and produced several false detections. A modified version of the algorithm, KHT [23], achieved robust detections by using an improved

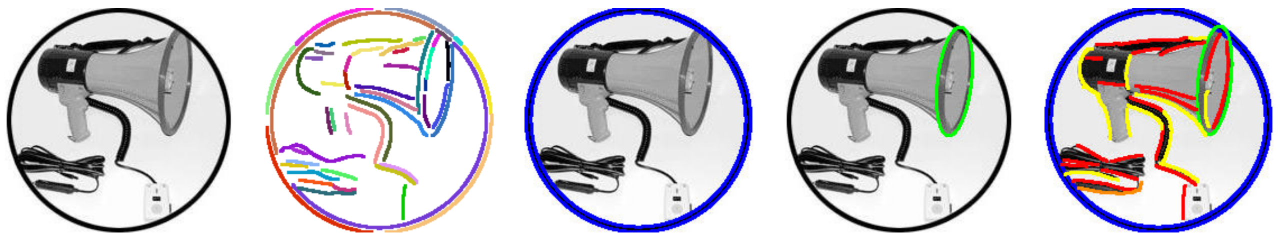


FIGURE 2. Geometry analysis obtained with the proposed method. From left to right: original image, contours detected using the method in [8], [9] (at a particular scale), detected circles (blue), ellipses (green) and joint detection including line segments (red), circular arcs (yellow) and elliptical arcs (orange). The initial contours at different scales are grouped into circles and ellipses, the rest are combined to form line segments, circular arcs or elliptical arcs. Curves that do not fall into any of these categories are not displayed in the final output. For simplicity only contours at a particular scale are displayed even if the procedure is multi-scale. See Fig. 5 for a more complete version of the figure.

voting scheme. A shortcoming of these methods is that they generate long lines, rather than segments. Etemadi [24] proposed to chain adjacent edge-pixels from which straight segments could be extracted, and Burns [25] linked pixels having the same orientation, but both methods produced too many segments and were very sensitive to noise. Grompone proposed (LSD algorithm, [10]) to combine Burns' idea with the a contrario validation approach [26]. This approach, also known as Helmholtz principle, states that no meaningful structures shall be perceived in a white noise image, and therefore the detection thresholds are automatically adjusted to that effect. LSD produces good results in general and it is used as the basis of other algorithms aiming at obtaining polygonal approximations of the image contours, as we shall see later in this section. EDLines [27] also reports good results and is faster than LSD. This method uses the Edge Drawing edge detector [28] to obtain chains of pixels from which line segments are extracted and validated using the a contrario principle.

Concerning circle detection, among the pixel-based approaches, the first published methods were based on the Circular Hough Transform (CHT) [29]–[32] adapting the original technique used for detecting line segments [21]. For circle detection, the method is as follows: first, a discrete 3D parameter space for the three parameters of the circle is created, where each point has an accumulator bin; after that, each edge pixel contributes to increase the accumulator of the points in the parameter space corresponding to all the circles centered at the pixel. At the end, when all the edge pixels have been explored, the circles are detected as the local peaks in the parameter space. This class of methods consume excessive memory to store the 3D parameter space and need a huge amount of time for voting and finding the local peaks. Moreover, many parameters, such as the bin size, affect significantly the final performance of the methods, and have to be finely tuned. To overcome the limitation of memory consumption, some authors tried an approach that uses a 2D accumulator bin instead of 3D [33], but it still has some false detections and low accuracy. After the first CHT variations with only small changes, many other variants appeared trying to overcome their drawbacks, such as probabilistic HT [34], [35], randomized HT [36], [37], fuzzy HT [38], etc. The probabilistic HT (PHT) methods [34], [35] randomly choose a subset of the edge pixels to accelerate the Hough Transform method. In [36], Xu *et al.* propose a randomized version of the Hough transform (RHT) that successively samples subsets of points from the set of edge pixels. The size of these subsets depends on the model to detect, e.g. two points for detecting a line and three for detecting a circle. The parameters estimated from each subset are updated in the parameter space for later finding the local peaks that correspond to detected figures. This permits to remove the parameter space sampling, since only those values estimated from the subsets are taken into account. In addition, the parameters' counters reflect the times that those where proposed rather than the actual number of edge pixels voting for them. In a theoretical framework, Kiryati *et al.* [39] have shown that RHT is faster than PHT in high quality images but PHT is more robust than RHT in images contaminated by noise and errors.

For ellipses detection, a straightforward extension of the Hough Transform method would imply to increase the dimensionality of the parameter space to five and find the peaks in this new space [22]. However, as with circles, the increase in dimensionality increases also the time and memory consumption of the method. As before, to alleviate these issues, many methods have appeared trying to overcome these limitations, such as the probabilistic HT [40], the randomized HT [41] or the method from Yuen *et al.* [42]. As an alternative to the Hough Transform, recently other methods using histograms of power ratios of points in circles and ellipses have been proposed [43], [44]. However, they suffer from the same drawbacks than the previously described methods in terms of quantization of the parameter space and noise sensitivity.

Another group of methods belonging to the pixel-based family comprises those based on the RANSAC algorithm [45]–[48]. These approaches appear in an effort to present an alternative to the HT algorithms and overcome their limitations. Among all the methods in this family, one of the most popular is the Random Circle Detection, RCD, presented by Teh-Chuan Chen and Kuo-Liang Chung in [45]. In this paper, they propose to select sets of four edge pixels and compute all the circles containing any three of them. If one of the circles is close to the non-used point, it is considered as a good detection. Finally, the good detections are validated and only circles containing a large enough number of edge pixels are kept. Since the number of sets of four edge pixels in an image may be huge, RCD involves many computations and leads to several initial good detections that are finally discarded after the validation step, which makes the algorithm inefficient.

Other approaches formalize the detection as an optimization problem. Least-square fitting methods [49]–[51] fall into this category, but also Bayesian methods [52] and Gaussian mixture models [53] have been used to solve the problem. The main drawback of these techniques is that they can only fit one primitive at a time, and therefore the fitting process must be repeated after removing all the edge pixels belonging to one of the already detected structures.

Contour-based approaches, also called edge-following methods, detect geometrical structures by taking advantage of the connectivity between edge pixels. For circle and ellipse detection the main strategy is to detect arcs of curves and then to group them forming the desired structure. The main differences between these methods reside in the criteria used to extract the arcs, in the way these arcs are grouped together and in how the parameters of the detected structures are computed.

Some methods [16], [54]–[57] rely on curvature or closeness properties to link edge points and obtain arcs of curves. Other methods resort to polygonal approximations of the image contours, based on the detection and linking of short line segments, computed using the LSD algorithm [10] as in [17]–[20], EDLines [27] as in [12], Kovesi's code [14] as in [15], Leung's method [58] as in [59] or the Ramer-Douglas-Peucker algorithm [60], [61] as in [62]. Edge segments extracted with the EDPF algorithm [11] are used in [13].

The extracted arcs of curves may be merged using different strategies: in some cases an initial estimation of the circles or ellipses parameters is computed using a least-squares fitting method and arcs supporting similar structures, or sharing similar parameters, are grouped together [12], [13], [15], [59]; in other cases arcs are combined based on convexity and position constraints [18], [54], [55], [62] or on geometric properties of ellipses or circles [20], [57].

With respect to the different strategies employed to compute the parameters of the detected structures, RANSAC-based approaches [15], [56] and voting schemes similar to the Hough transform [16], [54], [55], [57], [62] have been used for this purpose. However, these methods rely on the accurate calculation of local geometric features (curvature, tangent lines, etc.) and are sensitive to noise and partial occlusions. As an alternative, different least-square fitting methods applied on the extracted curves have been proposed [12], [13], [17]–[20], [59], [63].

These methods depend on several parameters that must be tuned in order to reduce the number of false detections. In some cases [17], [18], [20], [62] a minimum angular coverage or edge-pixel support is required to validate a detection. In other cases the parameters are learned using a ground-truth dataset [54], [55], [57], or are computed from the image itself using a saliency score [16]. In [12], [19] the a contrario approach, already mentioned at the beginning of this section, is used to automatically adjust the detection thresholds.

The method presented in this paper can be related to these previous works in the sense that the final goals of the detector are similar, but with the important novelty that the proposed detector uses a unified methodology for the detection of all such structures. The same procedure is applied progressively to group the initial contours forming full circles, then full ellipses, and finally the remaining contours are classified into circular or elliptical arcs or line segments. Very few published works aim at such a complete classification of the image contours. Some of the earlier attempts at the joint detection of straight lines and circular arcs in contours were reported in [64] and [24]. In [65] West *et al.* proposed a method for the segmentation of curves into line segments and elliptical arcs. These methods used an edge map as input and were very sensitive to noise. Recently, Pătrăucean *et al.* [19] proposed to jointly detect ellipses, circles and line segments with a controlled number of false detections. Similarly, the method in [12] can be used for the detection of both circles and ellipses with small eccentricity, but doesn't handle the detection of line segments. Finally, Wolters and Koch [63] proposed a method to extract the topology of an image, by classifying the image edges points into lines or parabolic arcs.

III. OVERVIEW OF THE PROPOSED METHOD

The proposed method is summarized in Algorithm 1. The input of the algorithm is the set of curves extracted with the method described in [8], [9], which we denote $\mathcal{S}_\sigma$. Each curve $s \in \mathcal{S}_\sigma$ is composed of a group of pixels with a continuity in orientation (what we call a *smooth contour*). These contours are either isolated or meet at T-junctions and corners. The only parameter of the extraction method is σ, that determines the spatial scale of the extracted contours: blurred contours may be extracted by using large values of the parameter, although the spatial location of the results is more accurate when σ is small. Initially, the detection algorithm is applied independently on the contours extracted at each scale, but in a posterior step, the detections obtained at different scales are combined.

The proposed method proceeds identically for the detection of circles and ellipses. For simplicity, we describe the method for circle detection. The algorithm proceeds by finding circles that fit the extracted contours. Starting with an initial contour $s \in \mathcal{S}_\sigma$, a least-squares method is used to compute the parameters of the circle that best fits the pixels in s. If the fitting error is small the rest of curves in $\mathcal{S}_\sigma$ are examined using a RANSAC-like strategy to check if they belong to the circle. If they do, the parameters of the circle are recomputed using the additional curves and the process is iterated until no more curves can be added to the group. The curves belonging to this group are removed from $\mathcal{S}_\sigma$ provided that the computed circle is validated by angular coverage and using an a contrario criterion. Therefore, only (almost) full circles perceptually meaningful are considered as valid detections. The whole process is repeated using the next contour in $\mathcal{S}_\sigma$. The circles detector is applied for different values of the scale parameter σ and the detections at different scales are combined to obtain the final result. A complete account of the circles detection stage is given in Section IV.

At this point of the algorithm, for each scale σ, $\mathcal{S}_\sigma$ contains only the curves that do not belong to a circle with enough image coverage. The previous process is repeated but computing in this case the parameters of the ellipse that better fits the pixels in each curve $s \in \mathcal{S}_\sigma$. By using the same strategy described above, at the end of this stage all the (almost) full ellipses in the image have been detected by the method. Section IV-G explains the details of how the circles detection method may be adapted for the detection of ellipses. One may think that the detection of circles and ellipses could be done simultaneously, since circles are special cases of ellipses. However, the validation of each detected structure depends on an a contrario argument (see Section IV-D) that assigns smaller 'meaningfulness' to those structures that have the potential to appear more in an image. Ellipses, having a higher number of degrees of freedom, can appear in more configurations in the image than circles. Therefore, a circle tested as a 'ellipse' gets a 'meaningfulness' score smaller than the same circle tested as a 'circle', and could be missed by the detector.

The search for global structures is limited to circles and ellipses. We do not aim at joining small segments into lines, since this is a non bounded structure, and this grouping would prevent a subsequent grouping into polygonal structures (which shall be the subject of future research).

The remaining curves in $\mathcal{S}_\sigma$, for the different values of σ, are fused into a single set $\mathcal{S}'$ and they are finally classified as circular arcs, line segments, elliptical arcs or none of the above. The process is described in Section V.

Algorithm 1 Overview of the Method

```
Input: Image: $I$
Input: Set of $\sigma$ values (increasing order): $M_\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_{M-1}, \sigma_M\}$
Output: Set of circle detections: $\mathcal{C}$
Output: Set of ellipse detections: $\mathcal{E}$
Output: Set of segment detections: $\mathcal{L}$
Output: Set of circular arcs detections: $\mathcal{A}$
Output: Set of elliptical arcs detections: $\mathcal{Q}$
1: for $\sigma \in M_\sigma$ do
2:     $\mathcal{S}_\sigma \leftarrow$ Contour detection($I, \sigma$){(Method from [9])}
3: end for
4: $\mathcal{C}, \mathcal{S}_\sigma \leftarrow$ Multiscale circle detection($\mathcal{S}_\sigma$) {Algorithm 2}
5: $\mathcal{E}, \mathcal{S}_\sigma \leftarrow$ Multiscale ellipse detection($\mathcal{S}_\sigma$) {Algorithm 2}
6: $\mathcal{L}, \mathcal{A}, \mathcal{Q} \leftarrow$ Multiscale line segment, circular arc and elliptical arc detection($\mathcal{S}_\sigma$) {Algorithm 3}
```

IV. CIRCLES AND ELLIPSES DETECTION

The steps of the algorithm for circles and ellipses detection are detailed in the following subsections. The complete process is described in Algorithm 2. In the last subsection it is explained how the algorithm may be adapted for the detection of ellipses.

Algorithm 2 Multiscale Circles/Ellipses Detector

```
Input: Set of contours $\mathcal{S}_\sigma$, extracted at scales $\sigma \in M_\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_M\}$, $\sigma_1 < \sigma_2 < \cdots < \sigma_M$
Output: Set of circles/ellipses detections: $\mathcal{C}$
Output: Set of image contours not belonging to detected circles/ellipses: $\mathcal{S}_\sigma$
1: //Single scale detection
2: for $\sigma \in M_\sigma$ do
3:     $\mathcal{C}_\sigma \leftarrow \emptyset$ {Set of detected circles/ellipses at scale $\sigma$}
4:     for $s \in \mathcal{S}_\sigma$ do
5:         $c_0 \leftarrow$ Initial_estimation($s$){Section IV-A}
6:         $c^* \leftarrow$ RANSAC_refinement($c_0, \mathcal{S}_\sigma$){Section IV-B}
7:         Validation($c^*, \mathcal{S}_\sigma$){Sections IV-C and IV-D}
8:         if Validation is OK then
9:             $\mathcal{C}_\sigma \leftarrow c^*$
10:            Remove inliers from $\mathcal{S}_\sigma$
11:        end if
12:    end for
13: end for
14: //Multiscale detection
15: $\mathcal{C} \leftarrow \mathcal{C}_{\sigma_M}$
16: for $\sigma \in \{\sigma_{M-1}, \cdots, \sigma_2, \sigma_1\}$ do
17:     if $c \in \mathcal{C}_\sigma$ is not repeated in $\mathcal{C}$ then
18:         $\mathcal{C} = \mathcal{C} \cup \{c\}$
19:     end if
20:     if $c \in \mathcal{C}_\sigma$ is tangent or equal to $c' \in \mathcal{C}$ then
21:         $\mathcal{C} = (\mathcal{C} \setminus \{c'\}) \cup \{c\}$
22:     end if
23: end for
24: //Remove contours in the support of detected circles/ellipses
25: for $\sigma \in M_\sigma$ do
26:     for $s \in \mathcal{S}_\sigma$ do
27:         if $s$ belongs to the support of some $c \in \mathcal{C}$ then
28:             $\mathcal{S}_\sigma \leftarrow \mathcal{S}_\sigma \setminus \{s\}$
29:         end if
30:     end for
31: end for
```

A. INITIAL ESTIMATION

Given a set of curves $\mathcal{S}_\sigma$ extracted at a given scale σ, each curve s in the set is checked to decide whether it belongs to a circle or not. First, the parameters of the circle c_0 (center and radius) that better fits the points in s are estimated using a least-squares algorithm [66]. In order to quickly reject wrong detections, the estimated circle is kept only if two conditions are satisfied:

(i) At least a percentage p_d of the circle is inside the domain of the image, where p_d is a parameter that we set to 80% in all of our tests.

(ii) Most of the points of the contour s are well fitted by the circular approximation. The fraction of contour points that fit the circumference is computed as

$$f_{c_s} = \frac{\#\{P \in s \,|\, d(P, c_0) < T_d(r)\}}{\#\{P \in s\}}, \tag{1}$$

where $d(P, c_0)$ is the distance from point P to the circumference and the threshold $T_d(r)$ is an increasing function of the radius, thus allowing farther distances for big circles than for smaller ones.[1] In practice, we require that $f_{c_s} > T_{pd}$, with $T_{pd} = 87.5\%$.

B. COMPUTING THE CIRCLE SUPPORT

From the initial circle candidate c_0 an iterative process is started in order to group together all the curves in $\mathcal{S}_\sigma$ that belong to the support of the circle. The support of a circle c is composed of all those contours that satisfy condition (ii). The process is as follows:

1) Consider $k = 1$, the index of the first step in the iterative process.
2) Compute $E \subseteq \mathcal{S}_\sigma$, the set of curves in $\mathcal{S}_\sigma$ that intersect the circle c_0, excluding s (the initial curve from which c_0 was computed). We consider that a curve $s' \in \mathcal{S}_\sigma$ intersects c_0 if the distance between at least one of its points to the circumference is not bigger than 1 pixel. If E is empty then stop the refinement process and keep as final result c_0.
3) Randomly choose a subset of curves R from E and define $S = \{s\} \cup R$.
4) Use algorithm [66] to fit a circle c_k to the curves in S and compute n_k, the number of contours in $\mathcal{S}_\sigma$ than are well approximated by c_k (using condition (ii)). These are the inliers at iteration k.
5) Increase k and iterate steps 3 and 4 a maximum of n times. The method requires few iterations, since the maximum number of different tests is bounded by $2^{|E|}$, where $|E|$ is the number of curves in E, which is generally small. We set a maximum number of iterations, $n = 25$, for exceptional cases.
6) Keep the circle c_k with highest n_k. Denote it as c^*.

In order to improve the robustness of the result, the above process is repeated N times, using as initial circle c_0 the best estimation obtained from the previous repetition. In our implementation we fix N to 4 repetitions.

Due to the random nature of the curve grouping process (step 3), it can be regarded as a RANSAC algorithm. Moreover, the inliers are computed over the whole set of curves S_σ instead of the set of intersecting curves E, as inspired by the LO-RANSAC method [67].

Fig. 3 illustrates our RANSAC method. In this example, the circle has been split in several curves by the contour detector. In Fig. 3d the starting contour s used to estimate the circle is drawn in green, the initially estimated circle c_0 is displayed in blue and the set of intersected contours E is marked in red. It must be noticed that the initial circle approximation does not fit perfectly the actual circle, and that E contains a straight segment that is not part of the circle's boundary. After the first repetition of RANSAC (Fig. 3e), this segment is discarded and only valid contours are kept. In the following repetitions (Fig. 3f to 3i) the algorithm refines the detection and the final set E is composed of all the contours actually belonging to the circle's boundary.

C. VALIDATION BY ANGULAR COVERAGE

The validation conditions applied so far ((i) and (ii)) guarantee that the detected circle is (mostly) inside the image and that most of the pixels in its support are well approximated by it. However, very short circular arcs may satisfy these conditions. In order to detect circles with enough contour support, we proceed as follows.

Denote as $\mathcal{P}_{c^*}$ the set of pixels belonging to the inliers of the circle c^*, estimated in subsection IV-B, and let O denote the center of the circle. For each pixel P in $\mathcal{P}_{c^*}$ compute the angle of the vector $\overrightarrow{OP}$ with respect to the horizontal, and store the results in a histogram. This histogram is built using L bins, and each bin k ($k \in \{0, \cdots, L-1\}$) counts how many contour points have an angle in the range $\left[\frac{2\pi}{L}k, \frac{2\pi}{L}(k+1)\right)$. We use $L = 36$ in the current implementation.

A circle is validated if the percentage of non-empty bins in the histogram (i.e. its angular support) is high enough:

$$\frac{\#\{k \in \{0, \cdots, L-1\}, \text{bin}_k \text{ is non-empty}\}}{L} > T_a \tag{2}$$

where T_a is a parameter that we set to 60%.

D. A CONTRARIO VALIDATION

The non-accidentalness principle of perception (or Helmholtz principle) states that a geometric structure is perceptually meaningful only when its expectation is low under random conditions. Based on this premise, the a contrario framework for the validation of geometric structures was introduced in [68] and further developed in [69]. The goal of this approach is to compute, for each structure, its expected number of occurrences in a random situation. Only structures having a low enough expected number of occurrences are validated as meaningful. In order to perform this computation, a background model describing the statistics of the data under the random hypothesis must be defined. The a contrario approach has been used in many domains [70]–[74] to control the number of false detections (also called number of false alarms, or NFA) of a detector. By defining the NFA as the expected number of occurrences of the detected structure under the random hypothesis, the detection thresholds may be adjusted in order to guarantee that this number is small. We apply this approach to the validation of the detected

[1] We use a logistic function $T_d(r) = \frac{B}{1+C\cdot e^{-D\cdot r}}$, where $C = \frac{B}{A} - 1$. The values A, B and D are set experimentally to 0.85, 5 and 0.025, respectively. For small circles, the threshold has a value close to 0.85 while for large circles, the parameter is close to 5

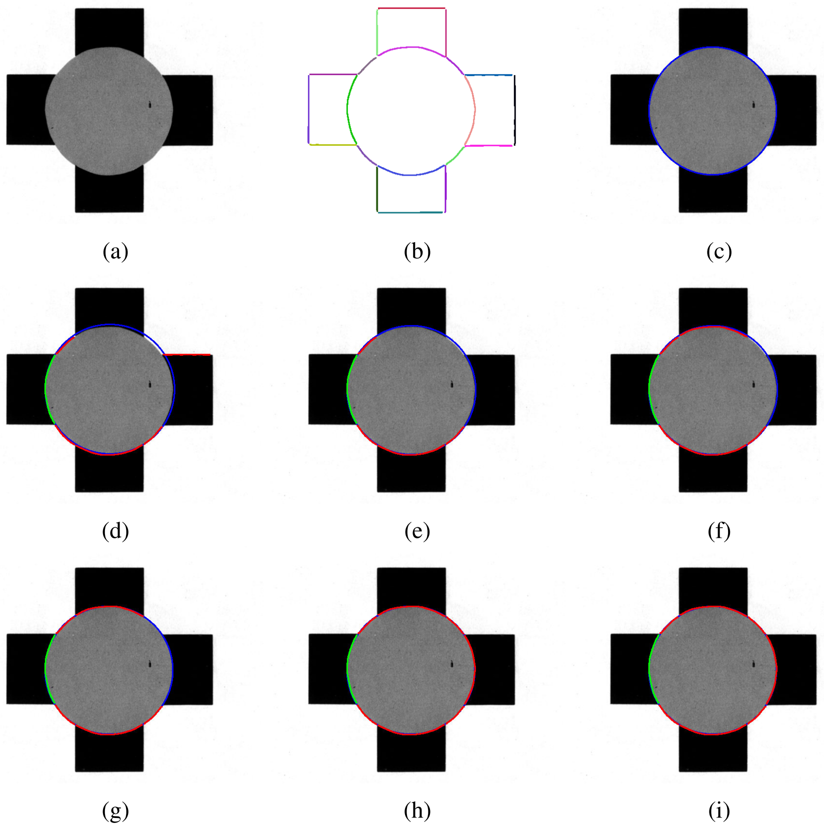


FIGURE 3. Illustration of RANSAC algorithm for contours grouping. First row: (a) original image, (b) contours detected, (c) final result. Other images: starting contour (green), estimated circle (blue) and intersected contours (red) after each repetition of RANSAC.

circle in a way similar to the one described in [10], where Grompone *et al.* present an algorithm to detect line segments in a given image. A similar method was also used for the validation of circles and ellipses in [12] and [75].

Given a detected circle c^* and $\mathcal{P}_{c^*}$, the set of pixels belonging to its inlier contours, we first count how many of these pixels have a gradient vector *aligned* with the normal vector of the circle at the pixel. By aligned we mean that the angular difference between these two vectors is below a given threshold τ. Since the direction of the gradient vector is not reliable when its magnitude is too small, pixels whose gradient magnitude is smaller than 1 are not considered in the computations. That is, a pixel $q \in \mathcal{P}_{c^*}$ is considered to be aligned with the normal vector of the circle if

$$\text{Angle}(\nabla I(q), \text{dir}_\perp(tan_{c^*}(q))) < \tau \;\text{ and }\; |\nabla I(q)| > 1 \quad (3)$$

where $\nabla I(q)$ is the gradient of the image I at point q and $\text{dir}_\perp(tan_{c^*}(q))$ is the direction orthogonal to the tangent to the circle c^* at q.

Next, the probability of having such a quantity of pixels aligned by chance is computed. For that, we use as background model the hypothesis that the angular differences are uniformly distributed in $[0, \pi]$ in a random situation. Therefore, the probability of finding a pixel with an angular difference smaller than τ is $p = \tau/\pi$. If the circumference is composed of n pixels, the probability of k of them being aligned with the image gradient is computed using the binomial probability distribution

$$B(n, k, p) = \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{(n-j)}. \quad (4)$$

Although this probability may be very small, this doesn't ensure that the detected circle is perceptually meaningful, since, if many such circles are tested using the same approach, it is possible that some of them appear in a random image merely by chance. The correct way to assess the meaningfulness of the circle is to compute its NFA, by multiplying its probability of occurrence by the number of potentially tested circles, or N_{tests}. This number may be estimated using the formula proposed in [75]: $N_{\text{tests}} = M^{\text{df}/2}$, where M is the number of image pixels and df the degrees of freedom of the detected structure (three in the case of circles, two for its center and one for its radius). To summarize, the NFA of the detected circle, given an angular precision τ, is

$$NFA = N_{\text{tests}} \cdot B(n, k, p). \quad (5)$$

The detection is meaningful if $NFA < \varepsilon$, where ε is a threshold usually set to 1 in most of the published works, which corresponds to allowing less than 1 false detection per image. We use this value for the detection of circles.

In our implementation, instead of selecting a particular value for τ, we propose to test several values $\tau_i \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. A detection will be declared valid if its NFA is less than ε for any of the $p_i = \tau_i/\pi,\ i = 1, \ldots, 5$. Formally, this is equivalent to increase the number of tests to $5\ N_{\text{tests}}$. Hence, the final test is

$$5 \cdot N_{tests} \cdot B(n, k, p_i) < \varepsilon,\ \ i = 1, \ldots, 5. \tag{6}$$

Fig. 4 shows some examples of how the two proposed validation steps reduce the number of wrongly detected circles in an image.

E. ITERATION OF THE DETECTOR

If the circle c^* detected in subsection IV-B is deemed valid, then the curves $s \in \mathcal{S}_\sigma$ than belong to its support (the inliers computed in subsection IV-B) are removed from the list of contours, and the process (subsections IV-A to IV-E) is iterated until all the remaining curves have been tested.

F. MULTISCALE DETECTION

The above subsections describe how the meaningful circles of the image may be detected given the set of contours obtained at a given scale σ. The contours extracted at finer scales (low values of σ) are better located but more fragmented than the ones obtained at coarser scales (higher values of σ). Moreover, blurred circles may only be detected for high values of σ. For these reasons different sets of detections $\mathcal{S}_\sigma$ are obtained at each scale. In this section we shall explain how to combine these detections to obtain a single set $\mathcal{C}$. This final set cannot be a mere union of the different $\mathcal{C}_\sigma$'s, since similar detections may appear at different scales, giving raise to duplicated detections. We then follow the following procedure:

1) We initialize $\mathcal{C}$ to the set of circle detections obtained with the largest σ.
2) We proceed in descending order of σ and compare the detections in $\mathcal{C}_\sigma$ to the ones in $\mathcal{C}$:
   - If the detection is not identical or tangent to an existing one in $\mathcal{C}$, it is added to $\mathcal{C}$. We consider that two circles are identical if the distance between their centers and radii is smaller than T_s. Two circles are considered tangent to each other if a certain percentage P_t of points of the circle with the largest radius is at a distance smaller than T_t pixels from the other circle. We use $T_s = 4.0$ pixels, $P_t = 15\%$ and $T_t = 2.0$ pixels in our implementation.
   - If the detections are identical or tangent, the one at the finest resolution is kept and the existing one is removed from $\mathcal{C}$. This ensures that the better located detections are kept in the final result.
3) Finally, at each scale σ we remove from each set $\mathcal{S}_\sigma$ those contours belonging to the detected circles in $\mathcal{C}$. Condition (ii) in subsection IV-A is applied to decide that a contour belongs to the support of a circle.

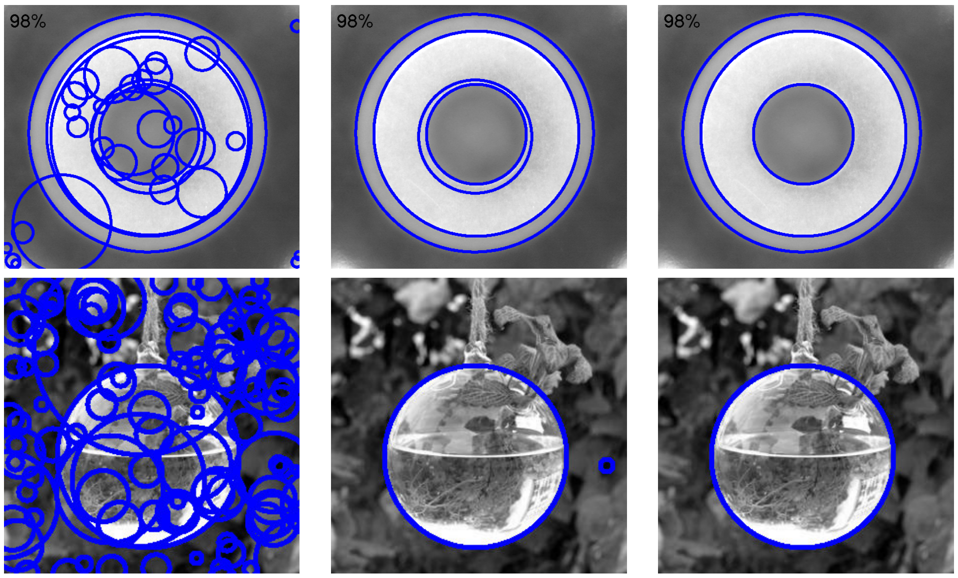
The pseudo-code of the multi-scale algorithm for circles is provided in Algorithm 2. In our tests we have combined the results obtained with $\sigma \in \{1.5, 2.0, 2.5, 3.0\}$.

G. DETECTION OF ELLIPSES

The detection of ellipses by the previous procedure needs some minor adaptations:

1) In the initial estimation step (subsection IV-A) the parameters of the ellipse e_0 (center, semi-major and semi-minor axes, and rotation with respect to the horizontal) are estimated using the least-squares algorithm from Halir *et al.* [76] which is a modification of the widely used solution from Fitzgibbon *et al.* [77]. Both methods provide a direct solution of the least-squares problem for ellipses, but the first one is numerically more stable. Moreover, in Equation (1), the parameter r of function $T_d(r)$ is now the length of the minor axis of the ellipse.
2) In many cases, the procedure described above is able to provide good initial estimates of the image ellipses. However, when the initial set of curves is excessively fragmented, it is possible that the initial ellipse doesn't intersect curves of $\mathcal{S}_\sigma$ other that s, thus preventing the refinement process. We therefore proceed by computing the initial estimate using pairs of curves, instead of a single curve.
   After applying the algorithm with a single curve as initialization, we create all possible pairs with curves not belonging to any of the already detected ellipses. We discard those pairs satisfying at least one the following conditions:
   (i) The pair of curves is non co-elliptical. This condition was first introduced by Cakir *et al.* [13] and its goal is to identify those pairs of curves whose concavities are towards opposite directions, i.e. the two curves are located back-to-back. To find if two curves are non co-elliptical, we compute the segment between the centers of the corresponding ellipses. If this segment intersects one of the two curves, the pair of curves is non-co-elliptical.
   (ii) One of the curves of the pair is well-fitted by a line. We apply the same validation used for segment detection presented in section V.

   These two criteria permit to quickly discard many pairs of contours, thus reducing the time of computation. For each validated pair, we consider the union of the two contours as initial curve and apply the procedure proposed in Section IV-B to add to the detection all the curves that belong to the support of the ellipse. If an ellipse is detected, all pairs containing any of the contours taking part of the ellipse are discarded.
3) The only modification in the a contrario validation (subsection IV-D) concerns the number of tests used to



(a) Before validation (b) After validation by angle coverage (c) After a contrario validation

FIGURE 4. From left to right: circles detected by our algorithm before the validation step; remaining circles after the validation by angle coverage; remaining circles after a contrario validation.

compute the NFA. Since the ellipse has five degrees of freedom, now $N_{\text{tests}} = M^{\frac{5}{2}}$. As commented in Section III, an ellipse is a more flexible geometric shape than a circle. Moreover, as it can be seen here, the value of N_{tests} is higher for ellipses. Therefore, it will be harder for a set of contours to pass the test for ellipses than for circles, making it useful to independently detect both types of structures.

4) Finally, for the combination of detections at multiple scales (subsection IV-F), in order to consider two ellipses to be identical, we need to additionally compare their orientations.

V. DETECTION OF LINE SEGMENTS AND ELLIPTICAL AND CIRCULAR ARCS

After the application of the algorithm described in Section IV, the sets $\mathcal{S}_\sigma$ contain, for each σ, the image contours that do not belong to the support of the detected structures. The final step of the proposed method is aimed at classifying them as circular arcs, line segments, elliptical arcs or none of the above. As in previous sections, we decided to separate the detection of circular and elliptical arcs.

The algorithm that we shall use for this classification is similar to the one proposed in previous sections. The main difference lies in how extracted features at different resolutions are merged. In Section IV, the groups of contours are classified as circles or ellipses before being combined at the final stage. In the current case, contours belonging to different scales are fused prior to classification. This permits to quickly remove duplicates and to join fragmented contours.

Let us first comment the part analogous to the algorithm in Section IV:

1) For each scale σ and for each contour in the set $\mathcal{S}_\sigma$, the parameters of the circle and the ellipse that better fits its pixels are computed using a least-squares algorithm [66] and [76], respectively, as in section IV. Also, the parameters of the line l_0 passing through the two more distant points of the contour are computed.
2) The validation criterion (ii) is used to remove, from each set $\mathcal{S}_\sigma$, the contours that are not well approximated neither by the estimated circle or ellipse nor by the line. In the case of lines, the parameter r in the distance threshold function is taken as the distance between the two more distant points in the contour.
   The a contrario validation is also applied to eliminate non meaningful contours. In the case of segments the maximum number of tests is set to $N_{\text{tests}} = M^2$.

Since the goal is to classify individual contours, the grouping procedure described in subsection IV-B is skipped.

Let us now describe the fusion process which takes as input the remaining curves at each scale after the previous validation test, $\mathcal{S}'_\sigma$. We denote by $\mathcal{S}$ the final set of detected arcs and segments at the end of the process. This set is initialized with the contours obtained at the coarsest scale (largest value of σ). We combine this set with the other sets of contours in descending order of σ. We compare one contour s_1 from $\mathcal{S}$ with each contour s_2 from $\mathcal{S}'_\sigma$ and we check if they both could be part of a common circular arc, line segment or elliptical arc:

1) We consider $s_\cup$, the union, and $s_\cap$, the intersection, of the curves s_1 and s_2. One pixel may belong to $s_\cap$ if at least

one point from the other curve is spatially close and shares a similar orientation. The orientation of the points is obtained at the output of the contour detector [9]. The curves are fused if the intersection is non empty.

2) Since the contours belong to different scales, the intersecting part of their union tends to be thick. In order to reduce this thickness we filter the intersection $s_\cap$.
Let $q \in s_\cap$ the point to be filtered. For each point $p_i \in s_\cup$, $i = 1, \dots, n$, we compute the weight

$$w_i = e^{(q-p_i)/\sigma_f^2}, \; i = 1, \dots, n, \tag{7}$$

with $\sigma_f = 10$. Then, we compute a line by minimizing the weighted least-squares error

$$\sum_{i=0}^{n} w_i(ax_i + by_i + c)^2, \tag{8}$$

where a, b and c are the parameters of the general form of a line defined as $ax + by + c = 0$. Finally, the filtered point, denoted by q_f and belonging to $\tilde{s}_\cap$, is defined as the orthogonal projection of q to the line minimizing 8.

$$q_f = \left(\frac{b(bq_x - aq_y) - ac}{a^2 + b^2}, \frac{a(-bq_x + aq_y) - ac}{a^2 + b^2} \right). \tag{9}$$

The use of a linear approximation, instead of the commonly used filtering technique consisting in replacing the point by the barycenter of the points in its neighborhood, permits to preserve the curvature, specially near the end points of the contours. It could be shown that the previous scheme is equivalent to a mean curvature motion of the intersection set, provided that we could parametrize $s_\cap$ as a curve.

3) At the end, the new curve s_f is defined as

$$s_f = \tilde{s}_\cap \cup (s_\cup \setminus s_\cap) \tag{10}$$

that is, the union of the filtered points and the points not belonging to the intersection.

4) We check that the filtered contour s_f may be approximated by a line, a circular arc or an elliptical arc using the a contrario validation method described in previous sections. If the test is positive then the filtered curve is added to $\mathcal{S}$.

5) The previous steps are iterated until all the curves in $\mathcal{S}$ and $\mathcal{S}_\sigma$ have been checked.

Finally, since all the contours in $\mathcal{S}$ have been validated as linear segments, circular arcs or elliptical arcs, we just need to classify them. Contours that do not qualify as linear segments are then tested as circulars arcs. If they fail the test, then they are tested as elliptical arcs. We have chosen this classification order because circular and elliptical arcs can be easily fitted to almost any contour. Indeed, straight contours might be well approximated by a large enough circle.

Algorithm 3 Multiscale Detection of Line Segments, Circular Arcs and Elliptical Arcs

```
Input: Set of contours $\mathcal{S}_\sigma$, extracted at scales $\sigma \in M_\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_M\}, \sigma_1 < \sigma_2 < \cdots < \sigma_M$
Output: Set of segment detections: $\mathcal{L}$
Output: Set of circular arcs detections: $\mathcal{A}$
Output: Set of elliptical arcs detections: $\mathcal{Q}$
1: //Initial selection of valid contours
2: for $\sigma \in M_\sigma$ do
3:   $\mathcal{S}'_\sigma \leftarrow \emptyset$
4:   for $s \in \mathcal{S}_\sigma$ do
5:     $c, l \leftarrow$ Compute circle and line($s$)
6:     if $s$ satisfies (ii) and (6) for $c$ or $l$ then
7:       $\mathcal{S}'_\sigma \leftarrow \mathcal{S}'_\sigma \cup s$
8:     end if
9:   end for
10: end for
11: //Multiscale fusion
12: $\mathcal{S} \leftarrow \mathcal{S}'_{\sigma_M}$
13: for $\sigma \in \{\sigma_{M-1}, \cdots, \sigma_2, \sigma_1\}$ do
14:   for $s' \in \mathcal{S}'_\sigma$ do
15:     repeat
16:       $\mathcal{I} \leftarrow \emptyset$ {Set of contours in $\mathcal{S}$ than have intersection with $s'$ }
17:       for $s \in \mathcal{S}$ do
18:         if $s'$ and $s$ have at least one close point then
19:           $\mathcal{I} \leftarrow \mathcal{I} \cup \{s\}$
20:         end if
21:       end for
22:       if $\mathcal{I} = \emptyset$ then
23:         $\mathcal{S} \leftarrow \mathcal{S} \cup \{s'\}$
24:       else
25:         $s \leftarrow$ first element in $\mathcal{I}$
26:         $s_f \leftarrow$ Filter($s' \cup s$)
27:         $c, l, e \leftarrow$ Compute circle, line and ellipse($s_f$)
28:         if $s_f$ satisfies (ii) and (6) for $l$, $c$ or $e$ then
29:           $\mathcal{S} \leftarrow \mathcal{S} \setminus \{s\}$
30:           $s' \leftarrow s_f$
31:         end if
32:       end if
33:     until $\mathcal{I} = \emptyset$
34:   end for
35: end for
36: //Classification
37: $\mathcal{L}, \mathcal{A}, \mathcal{Q} \leftarrow \emptyset$
38: for $s \in \mathcal{S}$ do
39:   $c, l, e \leftarrow$ Compute circle, line and ellipse($s$)
40:   if $s$ satisfies (ii) and (6) for $l$ then
41:     $\mathcal{L} \leftarrow \mathcal{L} \cup s$
42:   else if $s$ satisfies (ii) and (6) for $c$ then
43:     $\mathcal{A} \leftarrow \mathcal{A} \cup s$
44:   else if $s$ satisfies (ii) and (6) for $e$ then
45:     $\mathcal{Q} \leftarrow \mathcal{Q} \cup s$
46:   end if
47: end for
```

VI. EXPERIMENTATION

Most methods in the literature are aimed at the detection of a specific type of geometrical structure, either circles, ellipses or line segments. For this reason the experimentation section has been divided in several subsections, each of them comparing one stage of our method with methods in the literature that perform a similar detection. First, in Section VI-A, we compare the proposed method with methods detecting circles. After that, in Section VI-B, we compare our method against algorithms for ellipse detection. Next, in Section VI-C, we compare our method with methods that join edge points and classify them as segments or circular and elliptical arcs. Finally, in Section VI-D we explore the robustness of the detections under varying conditions of noise and blur. For a better understanding of the text, methods without an specific name, will be assigned a name according to the first author and the object they detect.

In all the experiments, the results obtained with our method will be displayed with the following color code: blue for circles, green for ellipses, red for line segments, yellow for circular arcs and orange for elliptical arcs. Fig. 5 shows a complete example of detection. In the top row of this figure the original image and the result of the proposed method are displayed. In the next four rows we can see the results of the detector at different scales. From left to right we have: extracted contours, detected circles, detected ellipses, and remaining contours that do not belong to any of the structures detected in the previous steps but that are partially fitted by a line, a circle or an ellipse. The last row shows, from left to right, the results of the multiscale procedure: fusion of circles, fusion of ellipses, and fusion of the remaining contours. The contours displayed in the last image are classified into circular arcs, elliptical arcs or line segments.

A. CIRCLE DETECTION

We evaluate our circle detection algorithm by comparing it with state-of-the-art methods: EDcircles [12], the algorithm of Lu *et al.* [18] (Lu_circles), RCD [45] and the circular Hough transform (CHT) (Matlab implementation [33], [78]). The codes for [12], [18] and [19] are made accessible by their authors[2][3][4] and we implemented RCD [45] following the corresponding article. All four methods have been evaluated using their default parameters. The Matlab implementation of CHT requires a minimum and maximum possible radius, which have been set to 5 pixels and to half of the minimum of the width and height of the image, respectively.

The evaluation has been carried out on three different datasets, some of them kindly provided by the authors of the compared detection methods: Industrial Printed Circuit Board (PCB)[2] image dataset, Natural image dataset (Natural)[2] and AUCDB. The PCB dataset contains 100 images of printed circuit board with circles on it. The Natural dataset contains 100 images, combining real images, as well as some images from the PCB dataset. AUCDB provides 225 images divided in several categories: Balls, Circles, Satellite, Structure, Synthetic, TD and TS (Traffic Signs), where each category contains images related to its name. The ground truth is available for the datasets AUCDB and PCB. We have built ourselves the ground truth for the Natural dataset and made it available at http://researchtami.uib.es/circles_groundtruths/.

In order to reliably compare all the previously mentioned methods, we have computed both qualitative and quantitative measures.

1) VISUAL COMPARISON

We evaluate the ability to correctly detect circles while not producing false detections. In many cases, this may be a subjective comparison, since every observer may have a different idea of which structures should be detected.

Fig. 6 compares the performance of all the methods with images from the Natural dataset. We display the original images as well as the detections by CHT [33], [78], EDcircles [12], RCD [45], Lu_circles [18] and our method. All the circles have been drawn in blue, overlayed onto the original image. The first row presents an image with many concentric circles. In this image, RCD has many false detections and none of them corresponds to a real circle. CHT is not able to detect any circle. Finally, Lu_circles, EDcircles and our method produce results with no false detections, but our method is the only one that detects all the circles correctly. Lu_circles fails to detect two of the circles and EDcircles has one circle that is not correctly detected, since it is tangent to another circle. In the second row there are two kinds of circles: the ones formed only by a contour line, and the filled ones. RCD does not detect any of them, CHT misses many and EDCircles fails at detecting the filled ones. Lu_circles and our method are able to detect all of them. The basketball in the third row just contains one circle, the outer one. All the methods, except for CHT, are able to detect this circle. EDcircles and CHT have false detections.

The examples in Fig. 7 correspond to the AUCDB database. In the first row CHT is able to detect two circles almost correctly, it only fails on the length of the radius, due to the discretization of the parameter space. Moreover, it has four false detections. In this image, EDcircles and Lu_circles fail to detect one and two circles, respectively, and they do not have any false detections. RCD has many false detections due to the grass texture, which creates many edge points that are considered as parts of a circle by this method. Finally, our method detects the four balls without false detections. In the second row none of the methods gets false detections. However, they fail to detect some of the circles, except for our method, which is able to detect all of them. In this example, Lu_circles is the method that detects less circles. In the last row our method fails to detect one of the five balls, while EDcircles is able to detect all of them. This missed detection

[2]https://github.com/AlanLuSun/Circle-detection
[3]https://github.com/CihanTopal/ED_Lib
[4]https://github.com/viorik/ELSDc

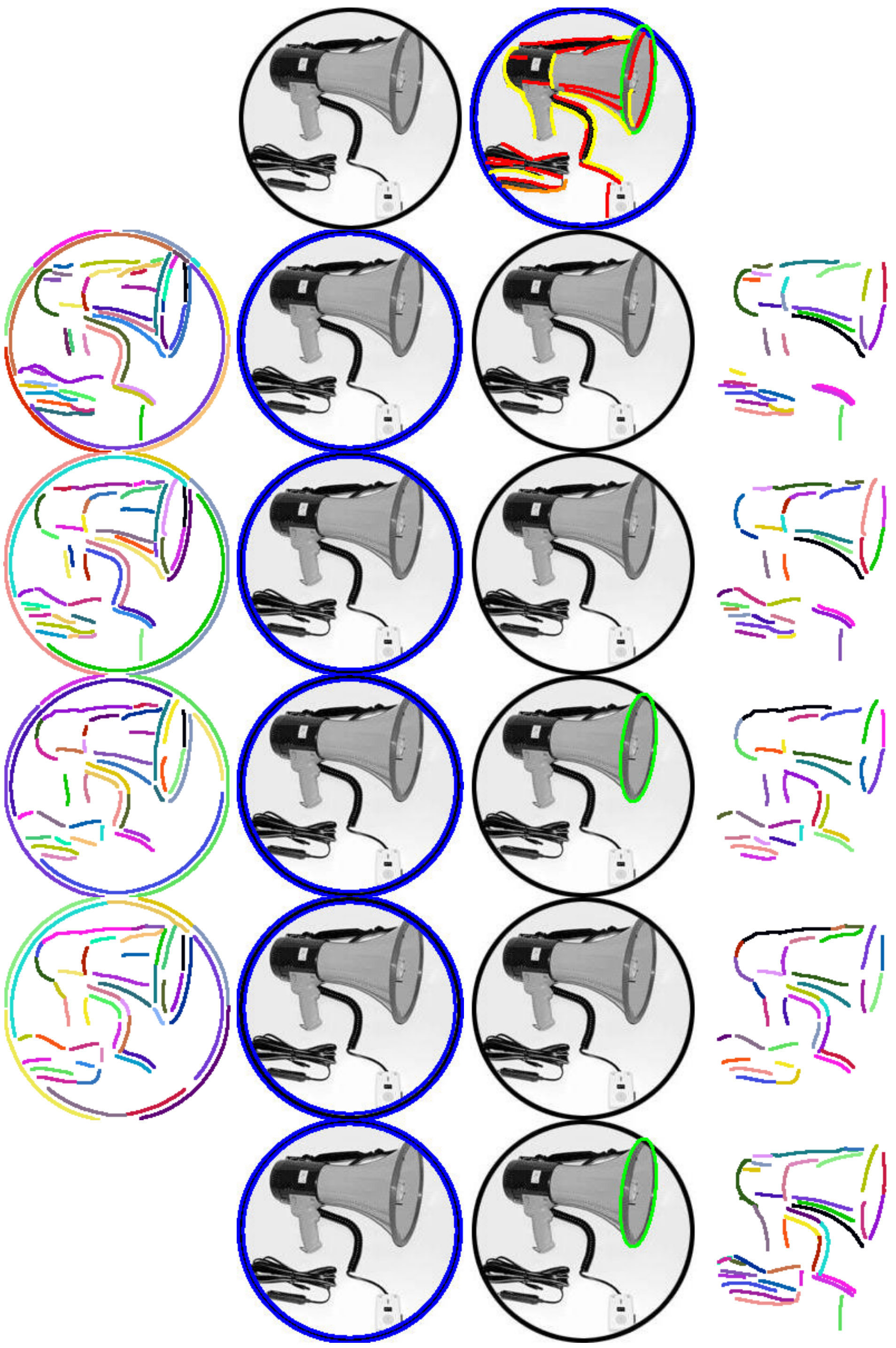


FIGURE 5. Illustration of the proposed detection process. First row: original image and final detection result. Different colors are used to label the different structures: blue for circles, green for ellipses, red for line segments, yellow for circular arcs and orange for elliptical arcs. Second to fifth rows: detection results for different values of the scale parameter ($\sigma = 1.5, 2.0, 2.5$ and 3.0). From left to right we observe: initial contours, detected circles, detected ellipses, remaining contours that can be fitted either by a line segment, a circular arc or an elliptical arc. In the final row are displayed, from left to right, the results of the multiscale procedure: fusion of circles, fusion of ellipses, fusion of remaining contours.

by our method is due to the fact that we are requiring a validation by angle coverage too high for the visible portions of this ball.

In order to summarize the qualitative results, let's mention that Lu_circles produces quite accurate results and no false detections in the images from the Industrial and Natural

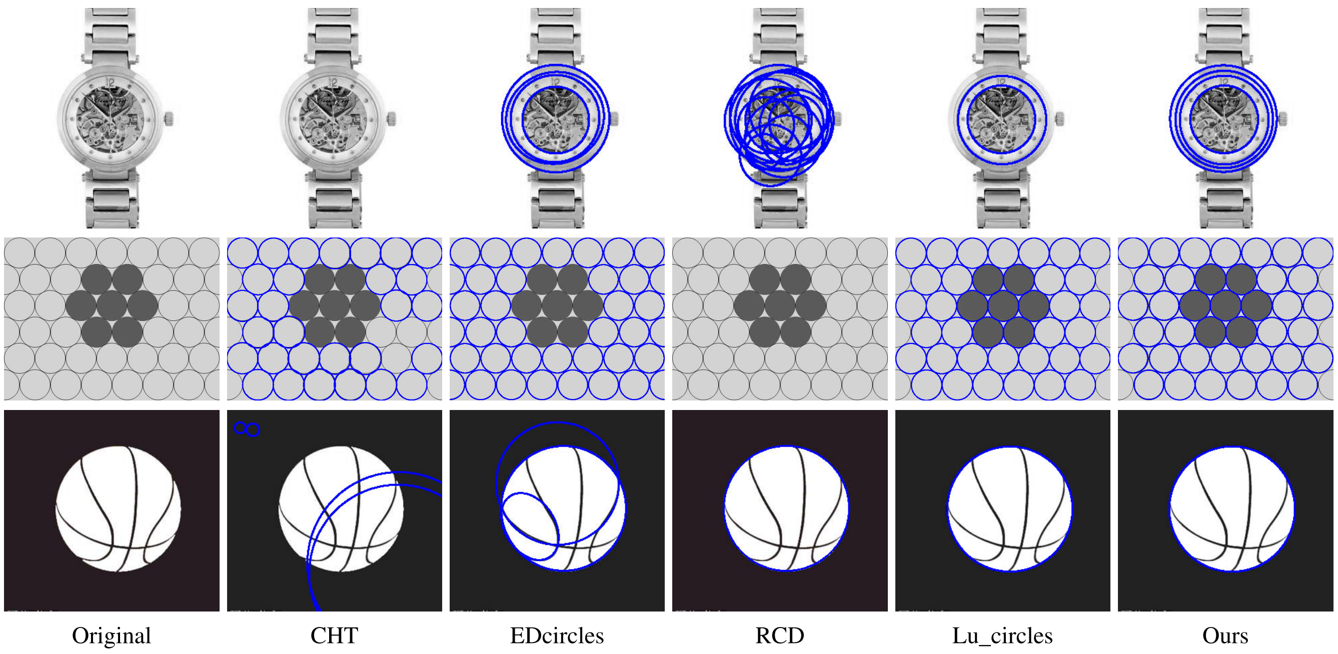


FIGURE 6. Circle detection comparison for images from the Natural dataset. From left to right: original image, CHT [33], EDcircles [12], RCD [45], Lu_circles [18] and our method. The detected circles are drawn in blue.

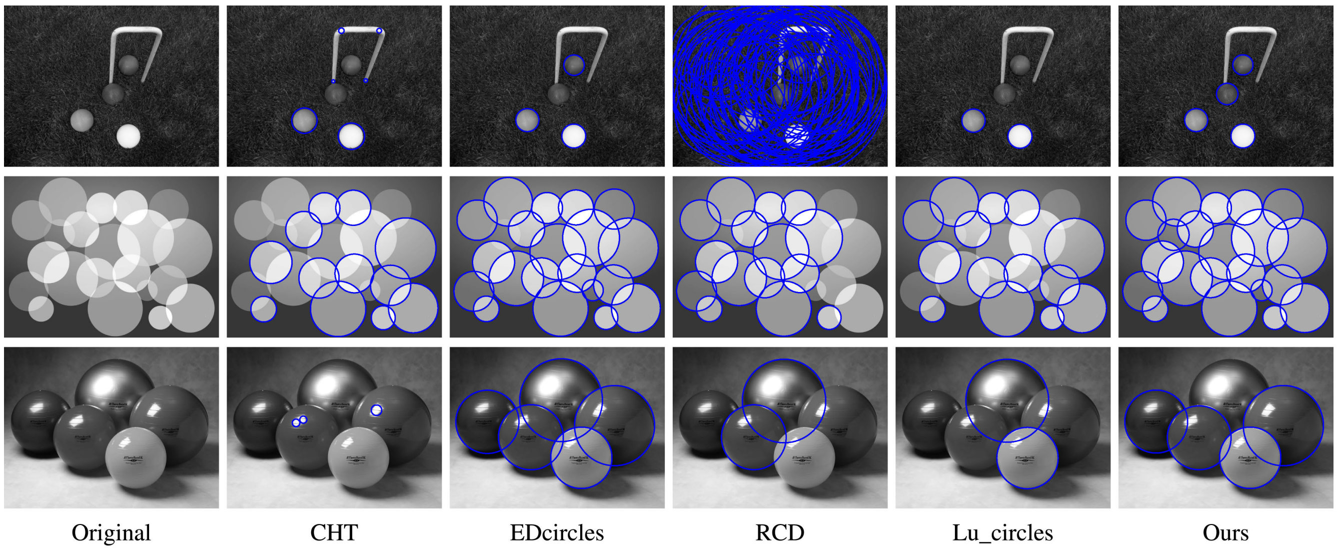


FIGURE 7. Circle detection comparison for images from the AUCDB dataset. From left to right: original image, CHT [33], EDcircles [12], RCD [45], Lu_circles [18] and our method. The detected circles are drawn in blue.

datasets, which were taken from their repository. However, this method has lower performance in the AUCDB dataset. EDCircles has very good results in the AUCDB dataset which they originally used for testing, but has a lower performance in the Natural one. Our method gives the best compromise in all datasets.

2) NUMERICAL COMPARISON

In order to quantitatively assess the results, we use the common precision and recall metrics, which are respectively defined as

$$\text{precision} = \frac{TP}{TP+FP}, \quad \text{recall} = \frac{TP}{TP+FN} \tag{11}$$

where TP are true positive detections, FP false positive detections and FN false negative detections. We consider that a detected circle is a true positive if there exists a circle in the ground-truth set such that the distance between their centers and radii is lower than $T = 4$. Since our method only considers as detections those circles that are almost completely contained in the image domain (condition (i)), to make a fair

TABLE 1. Average precision and recall results on three image datasets for circle detection and the average of all the averages. In each case, the two best results of precision and recall have been highlighted.

Dataset	PCB dataset		AUCDB dataset		Natural dataset		Average	
Method	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Lu_circles	**0.974**	0.623	**0.948**	0.534	**0.757**	0.500	**0.893**	0.552
EDcircles	0.503	**0.708**	**0.948**	**0.787**	0.697	**0.666**	0.716	**0.720**
CHT	0.145	0.117	0.108	0.209	0.285	0.223	0.179	0.183
RCD	**0.789**	0.552	0.146	0.109	0.231	0.181	0.389	0.281
Ours	0.778	**0.787**	**0.935**	**0.732**	**0.761**	**0.674**	**0.825**	**0.731**

comparison we eliminated both from the detections of the other methods and from the ground-truth all the circles that do not satisfy this condition.

Quantitative results of precision and recall for the methods under comparison can be found in Table 1, as well as the average precision and recall for the three datasets.

In the PCB database, our method has the best recall and Lu_circles the highest precision. Lu_circles has a very good precision but a poor recall in all databases. Similarly, RCD has high precision but a pretty low recall in PCB. Moreover, EDcircles performs worse than our method in terms of precision and recall for this database.

Results in AUCDB are quite similar for EDcircles and for our approach. Except for the precision of Lu_circles, the results from the other methods are significantly inferior. AUCDB is a very challenging dataset, since it contains many different kinds of circles. For the Natural database, our method presents the best precision and recall values, except for the precision of Lu_circles.

The average precision and recall in all three databases corroborates the previous observations. The proposed method has the best average recall and average precision except for the precision from Lu_circles which is slightly higher at the cost of a much lower recall than our method. Regarding the other methods, EDcircles seems to offer a good compromise between precision and recall.

B. ELLIPSE DETECTION

We evaluate the ellipse detection algorithm by comparing our results with the results of some of the state-of-the-art methods: the algorithm of Mai *et al.* [15] (Mai_ellipses), the algorithm of Fornaciari *et al.* [55] (Fornaciari_ellipses), the algorithm of Jia *et al.* [57] (Jia_ellipses) and EDcircles [12], which also computes ellipses. The codes for Fornaciari_ellipses and Jia_ellipses are made accesible by the corresponding authors[5][6] and we implemented Mai_ellipses, following the corresponding article.

The evaluation has been carried out on three different datasets provided by Fornaciari *et al.* [55][7]: Dataset Prasad, Random Images Dataset and Smartphone Images Dataset. The first one was originally created by Prasad *et al.* [16] and it had 400 images taken from the Caltech 256 dataset [79], but due to a hardware problem, the authors lost part of the annotations and now there are only 198 images, which are the ones used in this experimentation and shared by Fornaciari *et al.* The other two datasets were created by Fornaciari *et al.* The first one consists of 400 real images collected from MIRFlickr and LabelMe repositories and the second one contains 629 images taken with an smartphone.

EDcircles and our method detect circles and ellipses separately. Hence, to compare the results we will plot circles and ellipses equally for the methods that detect both geometrical structures.

1) VISUAL COMPARISON

Fig. 8 shows examples of our method applied to images from Random and Prasad datasets and a comparison with the other state-of-the-art methods. In the first row, Mai_ellipses is unable to detect the two ellipses in the glasses, while Jia_ellipses and Fornaciari_ellipses detect only the one on the right side (Fornaciari_ellipses detects several instances). EDcircles and our method detect both ellipses, although EDcircles has a false detection.

In the second example there are two ellipses describing the silhouette of the frying pan and the boiling pot. Mai_ellipses detects parts of both silhouettes, but the ellipses detected are wrong. EDcircles and Jia_ellipses detect only one of them. Finally, Fornaciari_ellipses and our method detect both of them, but Fornaciari_ellipses has also two false alarms.

In the third row all the methods find the ellipse at the bottom of the image, but the one detected by Mai_ellipses is not well located. Regarding the biggest ellipse, EDcircles, Mai_ellipses and our method detect it, but, as in the previous examples, the detection of Mai_ellipses is only partially well-fitted. Finally, EDcircles detects one of the circles in the microwave and our method detects both of them. Although the two circles are valid detections, the ground-truth does not include them. This reduces the precision of our method in the posterior numerical comparison.

In the last row all the methods detect correctly a subset of the observed ellipses. However, all the methods have some

[5]https://sourceforge.net/projects/yaed/

[6]https://github.com/dlut-dimt/ellipse-detector

[7]http://www.imagelab.ing.unimore.it/imagelab2015/ellipse/ellipse_dataset.zip

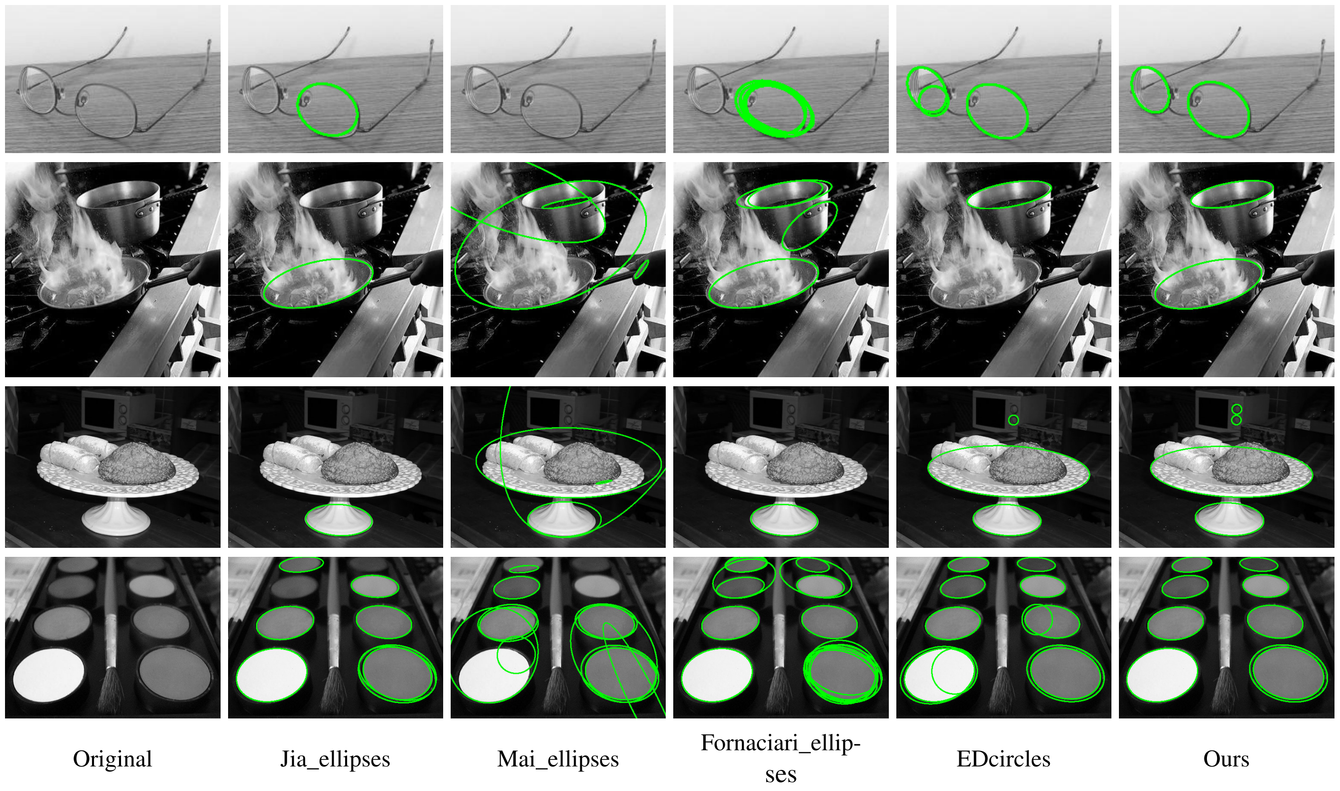


FIGURE 8. Ellipse detection comparison with images from Prasad and Random datasets. From left to right: original image, Jia_ellipses [57], Mai_ellipses [15], Fornaciari_ellipses [55], EDcircles [12] and our method. The detected ellipses are drawn in green.

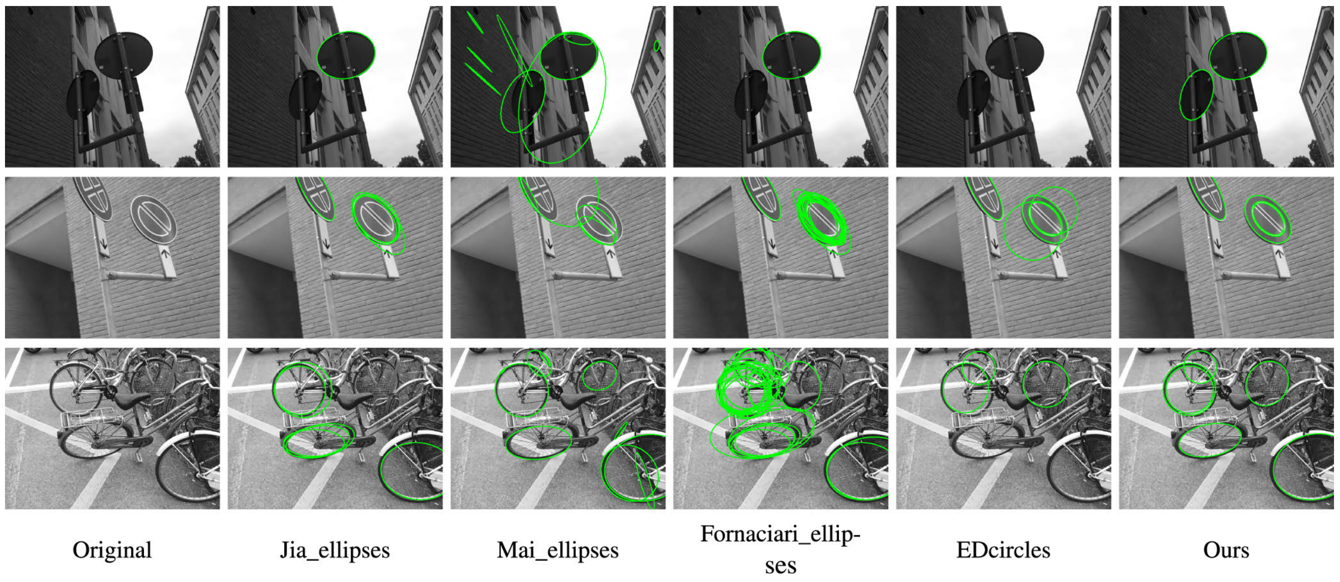


FIGURE 9. Ellipse detection comparison with images from the Smartphone dataset. From left to right: original image, Jia_ellipses [57], Mai_ellipses [15], Fornaciari_ellipses [55], EDcircles [12] and our method. The detected ellipses are drawn in green.

false detections, generally due to duplicate detections of the same ellipse, or caused by the union of contours belonging to different ellipses. In our case, we detect two ellipses in the bottom-right ellipse, but in fact, the boundary of the object in that area is thick enough to consider it as two separate detections.

Fig. 9 displays examples of detection on images from the Smartphone dataset. The first example contains two ellipses, although most methods only detect the one on the right hand side. EDcircle fails to detect both of them, and Mai_ellipses exhibits many false detections. The proposed method correctly detects both.

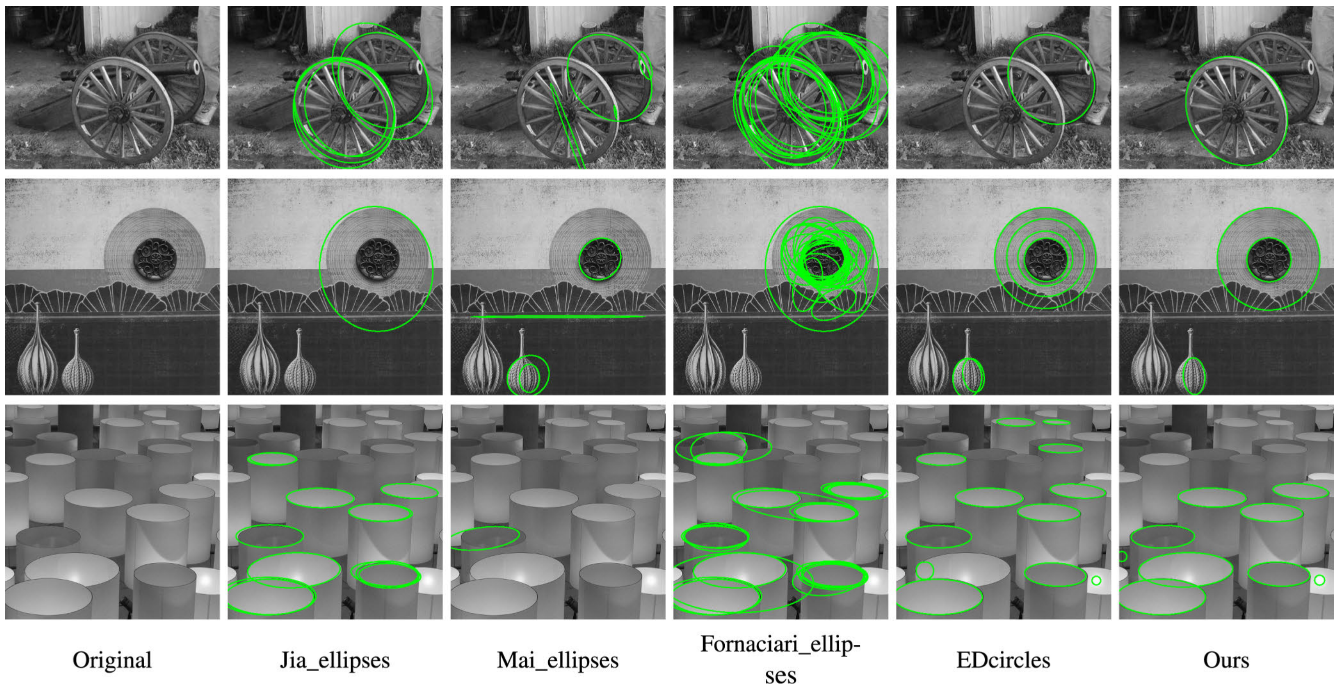


FIGURE 10. Ellipse detection failures of our method. We selected particular examples where our method fails in detecting all the ellipses or detects more than expected. From left to right: original image, Jia_ellipses [57], Mai_ellipses [15], Fornaciari_ellipses [55], EDcircles [12] and our method. The detected ellipses are drawn in green.

In the second row most methods present false detections. Jia_ellipses and Fornaciari_ellipses detect ellipses in the two traffic signs but many of them are duplicated. All the detections by Mai_ellipses are incorrect. EDcircles correctly detects the ellipses but also produces two false detections, while our method obtains correct results (although the inner ellipse on the traffic sign on the right is detected as two ellipses due to the thickness of the drawing line).

The example in the last row is challenging since several circles and ellipses, some of them partially occluded, are present in the image. With the exception of the proposed method, the tested algorithms are unable to detect all of them. Moreover, some of those detections are not well located. This is the case of EDcircles and Mai_ellipses: the first method detects only one of the wheels as an ellipse, but with a small error in the location; Mai_ellipses is not able to place well the ellipse in the center of the image. Moreover, Mai_ellipses, Jia_ellipses and Fornaciari_ellipses have false detections.

In Fig. 10 we display examples for which our method presents a lower performance. We selected particular examples where our method fails in detecting all the ellipses or detects more than expected. For the first example, we detect only one of the two wheels of the cannon. In the second example we have a false detection, although its shape is similar to an ellipse. In the last example, we fail to detect some of the ellipses. The comparison with the other methods shows that these are challenging examples, and that even in our worst case scenario, the results are competitive with the state-of-the-art.

To summarize the qualitative results on ellipse detection, let's mention that all the methods produce quite accurate results, but all of them have some drawback: Fornaciari_ellipses produces many false detections, due to the inaccurate clustering of the detections in the final stage of the method; Jia_ellipses sometimes misses detections and in other occasions produces many duplicates of the true detections; EDcircles fails to detect some of the ellipses and has some false circle detections. Finally, Mai_ellipses produces many false detections and sometimes the actual ellipses are only detected partially, as part of a bigger ellipse. Our method exhibits a good performance in all the examples.

2) NUMERICAL COMPARISON

In this section we perform a numerical comparison of the state-of-the-art methods on the three ellipse datasets presented at the beginning of the section. We use the same measures as in section VI-A2, but in this case, an ellipse is considered to be correctly detected if the common area between the detection and the ground truth is larger than 0.75 times the area of the bigger ellipse.

Table 2 shows the results for state-of-the-art methods on the three ellipse datasets. In all three datasets, and in average, our method gets the best precision and the second recall. Fornaciari_ellipses having the best recall, has the worst precision among all methods, which is corroborated in the visual comparison section. In the Prasad dataset, EDcircles has a better recall, but a significantly worse precision compared to our method.

TABLE 2. Average precision and recall results on three image datasets for ellipse detection and the average of all the averages. In each case, the two best results of precision and recall have been highlighted.

Dataset	Prasad dataset		Random dataset		Smartphone dataset		Average	
Method	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Jia_ellipses	0.429	0.254	0.368	0.443	0.284	0.588	0.360	0.428
Mai_ellipses	0.208	0.090	0.208	0.270	0.150	0.384	0.189	0.248
Fornaciari_ellipses	0.197	0.339	0.119	**0.547**	0.085	**0.738**	0.134	**0.541**
EDcircles	**0.575**	**0.403**	**0.505**	0.523	**0.625**	0.571	**0.568**	0.500
Ours	**0.752**	**0.394**	**0.621**	**0.540**	**0.629**	**0.663**	**0.667**	**0.532**

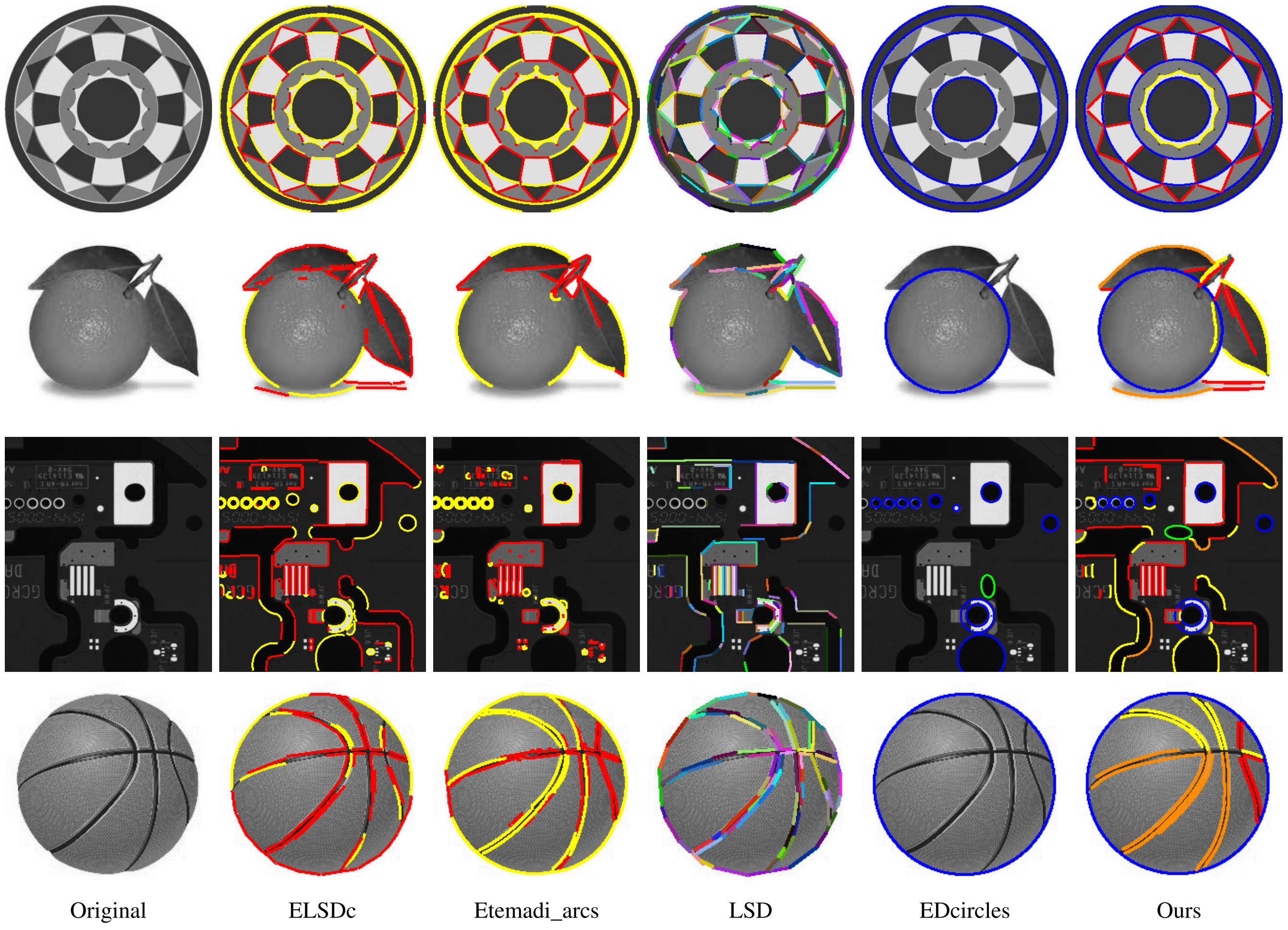


FIGURE 11. Line segments and circular arc detections in images from the Natural dataset. From left to right: original image, ELSDc [19], Etemadi_arcs [24], LSD [10], EDcircles [12] and our method. The detected ellipses are drawn in green, the circles in blue, the detected lines in red, the circular arcs in yellow and the elliptical arcs in orange.

C. TAXONOMY OF IMAGE CONTOURS

In this section we shall evaluate the performance of our whole detection chain compared with other detectors from the literature. We compare against ELSDc [19], which detects segments and arcs of ellipses; Etemadi's method [24] (Etemadi_arcs), which detects segments and circular arcs; LSD [10], which detects segments; and EDcircles [12], which detects circles and ellipses.

ELSDc is made available by their authors[8]; Etemadi_arcs is part of a toolkit created by the author named Object Recognition Toolkit (ORT) and a more recent version can be downloaded from a github repository[9]; LSD has an online implementation at www.ipol.im [80], where the source code is available for download. In this section, all the images are taken from the Natural and Prasad datasets used in the previous sections.

Fig. 11 displays the results obtained on images from the Natural dataset. In all the examples the detected circles are drawn in blue, ellipses in green, circular arcs in yellow, line segments in red and elliptical arcs in orange. The

[8]https://github.com/viorik/ELSDc

[9]https://github.com/encuadro/encuadro/tree/master/c/ort/ORT-2.3

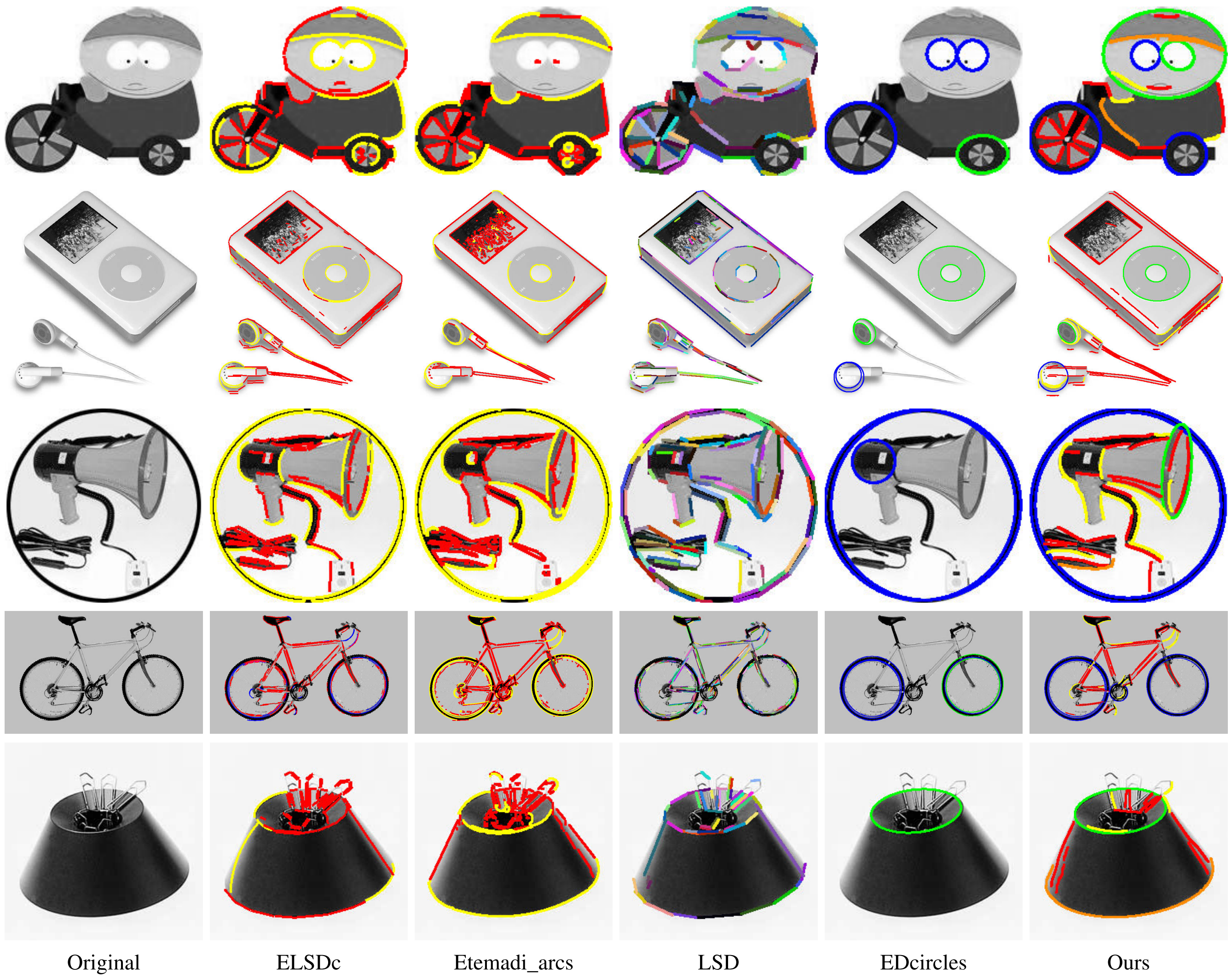


FIGURE 12. Line segments and circular arc detections in images from the Prasad dataset. From left to right: original image, ELSDc [19], Etemadi_arcs [24], LSD [10], EDcircles [12] and our method. The detected ellipses are drawn in green, the circles in blue, the detected lines in red, the circular arcs in yellow and the elliptical arcs in orange.

first example contains several concentric circles, some short curves close to the central circle and many segments belonging to geometrical shapes. The two biggest circles in the outer edge of the figure, and the smallest circle in the center, are detected by all the methods, although LSD detects them as several segments. The two middle sized circles are more difficult to detect, since they result from the union of several small dark and bright shapes. EDcircles does not detect the smaller one, ELSDc and Etemadi_arcs detect parts of them as segments, LSD detects them as several segments and our method detects them both. Regarding the linear segments, all the methods except EDcircles detect all of them correctly. Finally, there are several small arcs close to the smallest circle. All these shapes are well classified as arcs by our method, while ELSDc and Etemadi_arcs classify some of them as arcs and the rest as segments.

In the orange image of the second example all the methods are able to detect the main shape correctly, although ELSDc and Etemadi_arcs miss the areas close to the shadow of the fruit and close to the leaves. Apart from that, ELDSc, Etemadi_arcs and our method detect the leafs as contours, but the first two methods fail to classify them as arcs. Finally, ELSDc, LSD and our method detect the shadow of the object.

A complex circuit board is displayed in the third row. In this example, ELSDc detects all the main contours on the image, although it classifies incorrectly some of them as segments or arcs. Etemadi_arcs classifies well all the detected contours, but misses the low contrasted contours. EDcircles finds all the circles in the image, but it has a false ellipse detection. Our method detects most circles and classifies correctly the other contours as circular arcs, segments or elliptical arcs.

In the last example ELSDc and Etemadi_arcs detect parts of the contour of the basketball as line segments. Also, some of the curves inside the ball are incorrectly classified as line segments by these methods. Our method correctly detects the

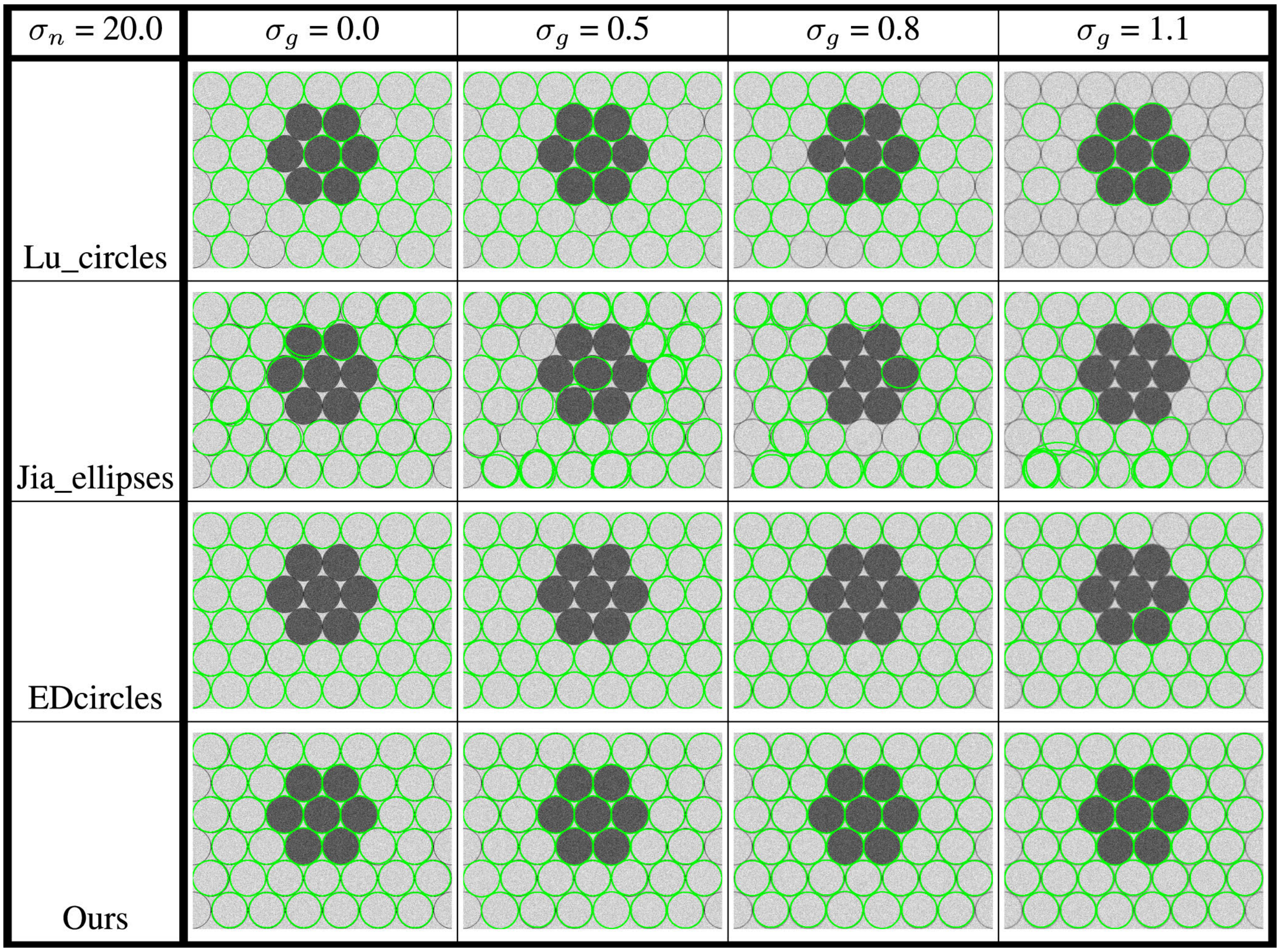


FIGURE 13. Detection comparison with noise and varying blur conditions.

big circle and most of the inner contours are classified as arcs, either circular or elliptical, except for some on the right side of the ball, which are more straight and hence classified as segments.

Fig. 12 shows examples of detection on images from the Prasad dataset. The first example shows an image of a cartoon character. ELSDc detects almost all the main contours, except for the big one delimiting the person from the bike, but classifies incorrectly some of them. Etemadi_arcs is not able to find some of the contours belonging to the head and none of the contours belonging to the eyes. EDcircles is not able to find the ellipse in the head and the ellipse in the right wheel is not perfectly aligned with the real one. Finally, our method finds all circular and elliptical shapes, although one eye is detected as a circle and the other as an ellipse.

The second example shows a music player and a pair of headphones. In this example ELSDc detects most parts of the circular shapes as arcs and the other contours are well classified as segments. Etemadi_arcs is only able to detect half of the big ellipse in the music player as an arc. Moreover, it detects many small contours on the screen of the device, which do not give relevant information about the image. EDcircles is able to detect all the circular shapes, as well as LSD. Finally, our method also detects all circular shapes and correctly classifies other contours.

The third image consists of a megaphone inscribed in a circle. All the methods are able to detect the outer circle and they do it as a double detection, one for the interior boundary and one for the exterior boundary. This is because the circle is thick enough to provide two separate zones with high gradient, where each of them provides one of the detections. Apart from that, EDcircles is not able to detect the ellipse in the megaphone and it has a false detection. All the other methods provide a good classification of the detected contours. Similar comments apply to the next two examples.

For the last row, ELSDc only classifies part of the top and bottom circular areas as arcs, and the other parts are classified as segments. Moreover, it classifies the left boundary of the object as an arc, when it is a segment. Etemadi_arcs has some small and noisy contours close to the paperclips which do not give any information about the image and classifies well the large contours describing the shape of the object. EDcircles

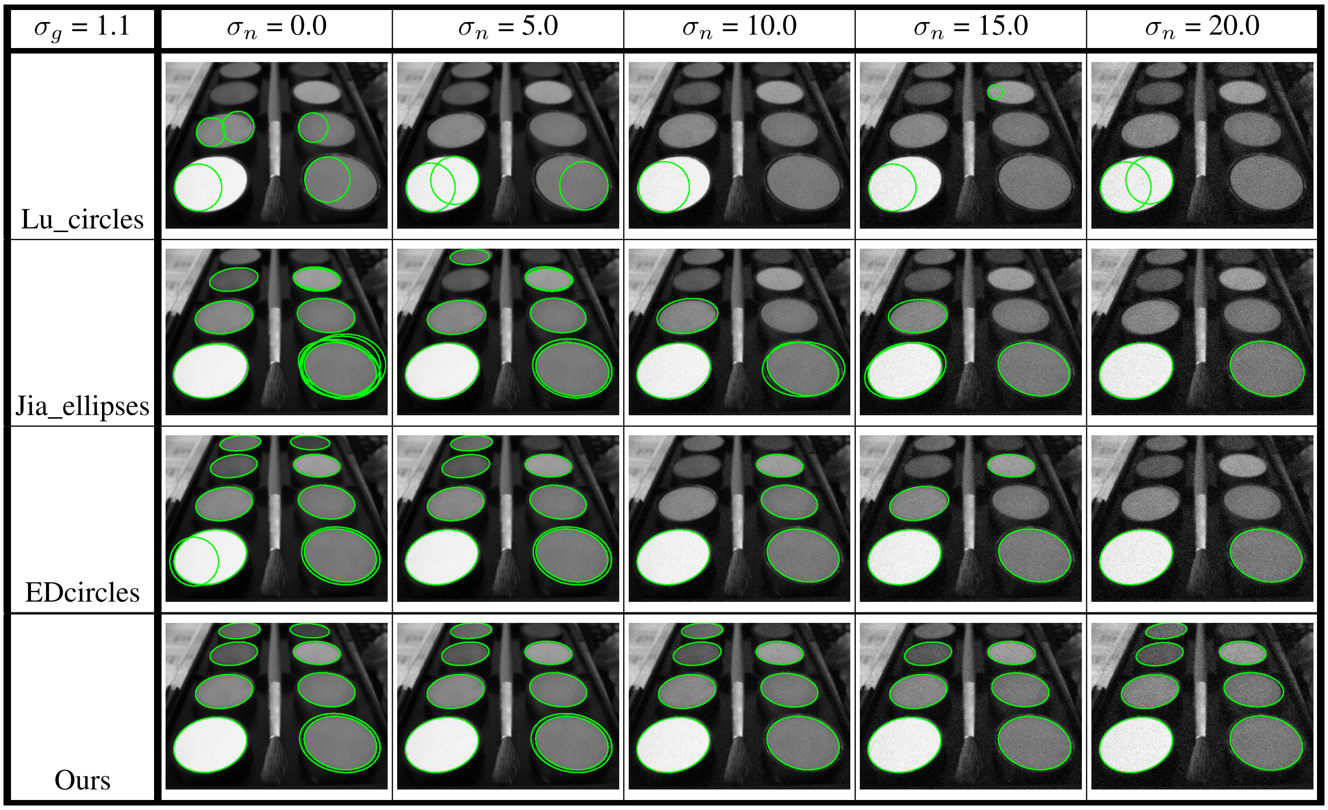


FIGURE 14. Detection comparison with blur and varying noise conditions.

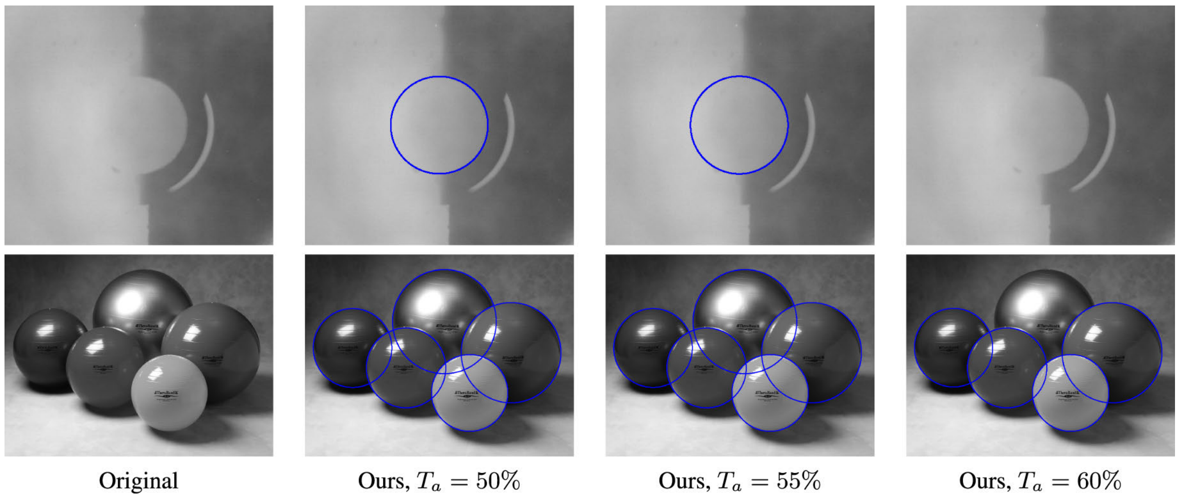


FIGURE 15. Circle detection comparison. Original image and the results of our method with different values of the coverage threshold T_a. We have set $T_a = 60\%$ as default value in order to avoid the detection of semi-circles (top row), although this might slightly increase the rate of missed detections (bottom row).

is only able to detect the ellipse on the top of the object. Our method detects one ellipse and an elliptical arc on the other one, and, as well as LSD, is able to detect and classify well the contours describing the shape of the object.

D. ROBUSTNESS TO BLUR AND NOISE

In this section we will explore the robustness to varying conditions of noise and blur of the methods achieving the highest performance in the previous sections. More specifically, the methods used in this section are Lu_circles [18], Jia_ellipses [57], EDcircles [12] and our method.

We apply the methods to a set of noisy and blurred images. Each of these test images is obtained by applying a Gaussian convolution of standard deviation σ_g and adding a white Gaussian noise of standard deviation σ_n to the original image.

TABLE 3. Average precision and recall results on three image datasets for circle detection for different values of parameter T_a. The chosen parameter is a good compromise for the recall and precision measures, which respectively decrease and increase when being more strict with the coverage criterion.

Parameter value	PCB dataset		AUCDB dataset		Natural dataset		Average	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
$T_a = 50\%$	0.712	0.844	0.878	0.816	0.669	0.692	0.753	0.784
$T_a = 55\%$	0.759	0.820	0.912	0.767	0.729	0.677	0.800	0.755
Default: $T_a = 60\%$	0.778	0.787	0.935	0.732	0.761	0.674	0.825	0.731
$T_a = 65\%$	0.796	0.787	0.944	0.702	0.785	0.660	0.842	0.716

TABLE 4. Average precision and recall results on three image datasets for ellipse detection for different values of parameter T_a. The chosen parameter is a good compromise for the recall and precision measures, which respectively decrease and increase when being more strict with the coverage criterion.

Parameter value	Prasad dataset		Random dataset		Smartphone dataset		Average	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
$T_a = 50\%$	0.535	0.429	0.400	0.594	0.378	0.747	0.438	0.590
$T_a = 55\%$	0.668	0.421	0.546	0.572	0.574	0.718	0.596	0.570
$T_a = 60\%$	0.725	0.402	0.606	0.545	0.640	0.683	0.657	0.543
Default: $T_a = 65\%$	0.752	0.394	0.621	0.540	0.629	0.663	0.667	0.532
$T_a = 70\%$	0.774	0.369	0.661	0.512	0.673	0.611	0.703	0.497

TABLE 5. Average precision and recall results on six image datasets for circle and ellipse detection and the average of all the averages for different values of parameter T_{pd}.

Parameter value	PCB dataset		AUCDB dataset		Natural dataset		Average	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
$T_{pd} = 85\%$	0.784	0.803	0.931	0.735	0.747	0.669	0.817	0.736
Default: $T_{pd} = 87.5\%$	0.778	0.787	0.935	0.732	0.761	0.674	0.825	0.731
$T_{pd} = 90\%$	0.786	0.781	0.942	0.716	0.770	0.660	0.833	0.719
$T_{pd} = 92.5\%$	0.788	0.781	0.946	0.705	0.782	0.656	0.839	0.714
	Prasad dataset		Random dataset		Smartphone dataset		Average	
$T_{pd} = 85\%$	0.732	0.402	0.607	0.557	0.614	0.696	0.651	0.552
Default: $T_{pd} = 87.5\%$	0.752	0.394	0.621	0.540	0.629	0.663	0.667	0.532
$T_{pd} = 90\%$	0.777	0.381	0.657	0.519	0.693	0.627	0.709	0.509
$T_{pd} = 92.5\%$	0.794	0.363	0.679	0.492	0.726	0.574	0.733	0.476

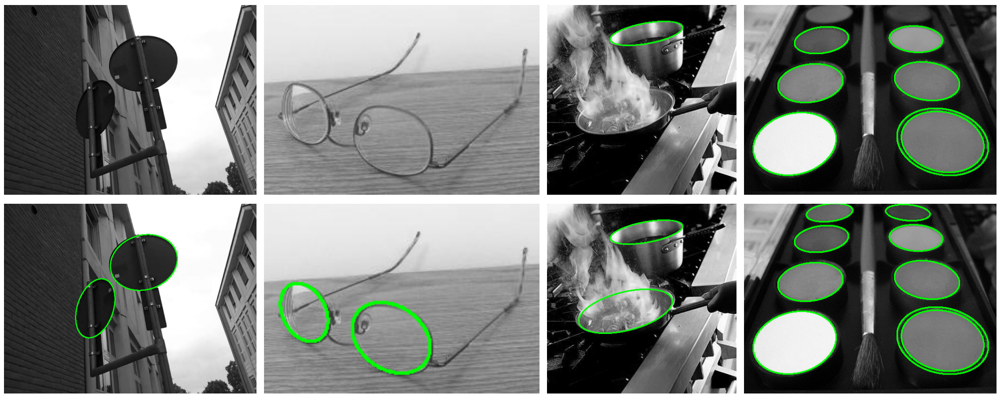


FIGURE 16. Selected examples which clearly benefit from using pairs of curves as initialization for ellipse detection. First row: results using only one curve for initialization. Second row: results using two curves for initialization.

We consider the values $\sigma_n \in \{0.0, 5.0, 10.0, 15.0, 20.0\}$ and $\sigma_g \in \{0.0, 0.5, 0.8, 1.1\}$.

Fig. 13 and 14 compare some of the obtained detections. We fix for each figure either σ_n or σ_g, and vary the other parameter. We display for all the methods the detections in green, regardless of whether they correspond to circles or ellipses.

In Fig. 13 we add a noise of $\sigma_n = 20$ and vary σ_g. Lu_circles shows to be robust to noise since it detects almost every circle without the presence of blur. However, as we

TABLE 6. Average precision and recall results on three image datasets for circle detection and the average of all the averages. We compare the effect of using one or two curves for the initial circle detection.

Parameter value	PCB dataset		AUCDB dataset		Natural dataset		Average	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
$k = 1$	0.785	0.790	0.945	0.707	0.776	0.653	0.835	0.717
Default: $k = 2$	0.778	0.787	0.935	0.732	0.761	0.674	0.825	0.731

TABLE 7. Average precision and recall results on three image datasets for ellipse detection and the average of all the averages. We compare the effect of using one or two curves for the initial ellipse detection.

Parameter value	Prasad dataset		Random dataset		Smartphone dataset		Average	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
$k = 1$	0.783	0.354	0.667	0.467	0.701	0.487	0.717	0.436
Default: $k = 2$	0.752	0.394	0.621	0.540	0.629	0.663	0.667	0.532

TABLE 8. Average precision and recall results on six image datasets for circle and ellipse detection and the average of all the averages for different values of parameter *N*, the number of refinements of the RANSAC-like contour grouping method.

Parameter value	PCB dataset		AUCDB dataset		Natural dataset		Average	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
$N = 1$	0.791	0.787	0.939	0.724	0.759	0.657	0.830	0.723
$N = 2$	0.785	0.787	0.941	0.726	0.757	0.664	0.828	0.726
Default: $N = 4$	0.778	0.787	0.935	0.732	0.761	0.674	0.825	0.731
$N = 6$	0.782	0.792	0.934	0.724	0.758	0.666	0.825	0.727
	Prasad dataset		Random dataset		Smartphone dataset		Average	
$N = 1$	0.753	0.388	0.624	0.529	0.657	0.659	0.678	0.525
$N = 2$	0.755	0.393	0.629	0.533	0.650	0.659	0.678	0.528
Default: $N = 4$	0.752	0.394	0.621	0.540	0.629	0.663	0.667	0.532
$N = 6$	0.759	0.395	0.628	0.537	0.650	0.660	0.679	0.531

increase the value of σ_g, this method detects less circles. Jia_ellipses is less robust to noise since has many double detections and the circles are not well located. Detections also tend to disappear as σ_g increases. EDcircles has similar detections when σ_g varies, although the central solid circles are not identified. Our method detects the same structures in all cases.

We fix $\sigma_g = 1.1$ in Fig. 14 and vary σ_n. The original image for this experiment is a palette of makeup where the perspective makes each circle to look as an ellipse. Since Lu_circles is devoted to circle detection, it fails in this example. Jia_ellipses gives a reasonable detection for the noise free image, but quickly degrades as the image gets noisier. EDcircles loses detections as noise increases, but it doesn't introduce false detections. Our result is stable, except for the detection lost when increasing the noise. This ellipse is poorly contrasted in the original image and becomes undetectable in the image as noise increases, even for our eye.

E. PARAMETER ANALYSIS

In this section we analyze the parameters that have a higher influence in the performance of the proposed method. The values of the rest of the parameters can vary in a relatively large range without producing meaningful changes in the results. The main parameters are: T_a, the minimum angular coverage for the validation of circles and ellipses; T_{pd}, the minimum percentage of points in a contour needed to decide that it fits a circle or an ellipse; the initial number of contours used to compute the support of circles and ellipses; and N, the number of refinements of the RANSAC-like contour grouping method described in Section IV-B.

In Fig. 15 we display the results of applying our method to the detection of circles with different values of T_a. The default value is $T_a = 60\%$, but for certain examples the use of $T_a = 50\%$ or 55% permits to detect more circles (bottom example). However, it also implies the detection of semi-circles as the ones in the top row, which we believe cannot be interpreted as a circle. In Tables 3 and 4 we show the performance of our method for circle and ellipse detection, respectively, when we modify threshold T_a. The results show that the precision increases as we increase the value of the threshold T_a. At the same time, the recall decreases, being the chosen parameter a good compromise for both.

In Table 5 we show the performance of our method when the parameter T_{pd} is modified. As it can be seen, in all datasets the precision increases as we increase the value of T_{pd}. However, the recall is lower with higher values of T_{pd}. We have chosen as default value $T_{pd} = 87.5\%$ because it gives the best balance between precision and recall.

TABLE 9. For different datasets, this table shows the average computation time per pixel of the proposed method. The average percentage of time devoted to the detection of circles, ellipses and other structures is also displayed.

Dataset	Time per pixel	% time for circles	% time for ellipses	% time for others
PCB dataset	4.50 ms	16.89 %	14.03%	69.08%
AUCDB dataset	9.19 ms	19.41%	27.02%	53.57%
Natural dataset	8.10 ms	21.92%	23.36%	54.72%
Prasad dataset	12.85ms	19.06%	34.20%	46.74%
Random dataset	17.18 ms	20.75%	37.54%	41.70%
Smartphone dataset	17.60 ms	22.90%	32.61 %	44.49%
Average	11.57 ms	20.16%	28.13%	51.72%

We also evaluate the difference in the performance of our method when one or two curves are used as initial estimates of a circle or ellipse contour. We have discussed in Section IV-G that the use of two curves increases the performance in the ellipse detection step, but the same setting can be used for the detection of circles. We display in Fig. 16 selected examples which clearly benefit from using pairs of curves as initialization. In Table 6 and 7 we can see the increase in performance when using this option, for circles and ellipse detection. In both cases, there is a significant increase in the recall, without penalizing too much the precision.

Finally, in Table 8 we show the performance of our method with different values of N. We observe that the precision and the recall do not have a large variation when changing the value of the parameter N. We did not include a table for the number of iterations n of each refinement, set to $n = 25$, because increasing its value does not modify the results. Compared to classical RANSAC applications this parameter is not critical for our procedure since the number of possible combinations of the selected curves is often lower than n.

F. COMPUTATIONAL COMPLEXITY

It is difficult to bound the number of operations per pixel in the proposed method. This is due to the several steps involved in the detection process and the fact that this number might actually depend on the number of initial contours detected, as well as its length.

We run our algorithm on a 3.2GHz Intel Core i7-8700 processor on an Ubuntu system. We measured the time of computation for each image used in the experimentation and normalized it by the number of pixels. This time accounts for the multi-scale detection of circles, ellipses and identification of segments and circular and elliptical arcs. Table 9 shows the mean time per pixel on each dataset, as well as the percentage of time devoted to circle detection, ellipse detection and contour merging and classification. For example, for an image of 512×512 pixels the detection takes on average 3.01 s. 20% of this time is devoted to circle detection, 28% to ellipse detection and 52% to contour merging and classification.

VII. CONCLUSION

We have presented in this paper a unified framework for the classification of the image contours into different geometric structures, namely, circles, ellipses, line segments, circular arcs and elliptical arcs. To the best of our knowledge, this is the first work to propose such a complete taxonomy of the image contours. The proposed technique involves several algorithms but has a common methodology: contours are detected at several scales, in a first step these contours are grouped to form circles, the ones that do not belong to the support of a circle are then grouped to form ellipses and, finally, those that do neither belong to circles nor ellipses are classified as line segments, circular arcs, elliptical arcs or none of the above. In order to reduce the number of false detections a validation step that assesses the perceptual meaningfulness of the detected structures is used. Among the technical novelties developed to fulfill the classification task we may mention a RANSAC-like technique for the grouping of sparsely distributed contours into circles or ellipses, a multiscale strategy for the combination of detections obtained at different scales, and a contour filtering method than permits the fusion of contours from different scales.

Multiple experiments illustrate the performance of the proposed methodology for the detection of the different geometric structures. Our method shows superior performance than state-of-the-art techniques in the tests performed on several datasets. Moreover, its multiscale character endows the method with robustness to noise and blur, as our experiments corroborate. In future work, we shall investigate the grouping of line segments into polygonal structures.

ACKNOWLEDGMENT

The authors would like to thank Changsheng Lu for kindly providing the ‘Industrial PCB dataset’ and its ground truth, and the ‘Natural Images dataset’, used in the Experimental section.

REFERENCES

[1] Y. Ebisawa, “Robust pupil detection by image difference with positional compensation,” in *Proc. IEEE Int. Conf. Virtual Environ., Hum.-Comput. Interfaces Meas. Syst. (VECIMS)*, May 2009, pp. 143–148.

[2] J. B. Hiley, A. H. Redekopp, and R. Fazel-Rezai, “A low cost human computer interface based on eye tracking,” in *Proc. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBS)*, Aug. 2006, pp. 3226–3229.

[3] F. Larsson and M. Felsberg, “Using Fourier descriptors and spatial models for traffic sign recognition,” in *Proc. Scand. Conf. Image Anal.*, 2011, pp. 238–249.

[4] A. Arlicot, B. Soheilian, and N. Paparoditis, "Circular road sign extraction from street level images using colour, shape and texture databases maps," in *Proc. Workshop Laserscanning*, 2009, pp. 205–210.

[5] C. Rothwell, A. Zisserman, C. Marinos, D. Forsyth, and J. Mundy, "Relative motion and pose from arbitrary plane curves," *Image Vis. Comput.*, vol. 10, no. 4, pp. 250–262, May 1992.

[6] L. da Fontoura Costa and R. M. Cesar, Jr., *Shape Analysis and Classification: Theory and Practice*. Boca Raton, FL, USA: CRC Press, 2010.

[7] P. Smith, I. D. Reid, and A. J. Davison, "Real-time monocular SLAM with straight lines," in *Proc. Brit. Mach. Vis. Conf.*, Edinburgh, U.K., Sep. 2006.

[8] A. Buades, R. G. Von Gioi, and J. Navarro, "Contours, corners and T-junctions detection algorithm," *Image Process. Line*, vol. 8, pp. 24–36, Feb. 2018.

[9] A. Buades, R. G. von Gioi, and J. Navarro, "Joint contours, corner and t-junction detection: An approach inspired by the mammal visual system," *J. Math. Imag. Vis.*, vol. 60, pp. 1–14, Sep. 2017.

[10] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.

[11] C. Akinlar and C. Topal, "EDPF: A real-time parameter-free edge segment detector with a false detection control," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 26, no. 1, Feb. 2012, Art. no. 1255002.

[12] C. Akinlar and C. Topal, "EDCircles: A real-time circle detector with a false detection control," *Pattern Recognit.*, vol. 46, no. 3, pp. 725–740, Mar. 2013.

[13] H. I. Cakir, C. Topal, and C. Akinlar, "An occlusion-resistant ellipse detection method by joining coelliptic arcs," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 492–507.

[14] P. D. Kovesi. *MATLAB and Octave Functions for Computer Vision and Image Processing*. Accessed: Feb. 5, 2021. [Online]. Available: http://www.peterkovesi.com/matlabfns/

[15] F. Mai, Y. S. Hung, H. Zhong, and W. F. Sze, "A hierarchical approach for fast and robust ellipse extraction," *Pattern Recognit.*, vol. 41, no. 8, pp. 2512–2524, Aug. 2008.

[16] D. K. Prasad, M. K. H. Leung, and S.-Y. Cho, "Edge curvature and convexity based ellipse detection method," *Pattern Recognit.*, vol. 45, no. 9, pp. 3204–3221, Sep. 2012.

[17] T. Le and Y. Duan, "Circle detection on images by line segment and circle completeness," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3648–3652.

[18] C. Lu, S. Xia, W. Huang, M. Shao, and Y. Fu, "Circle detection by arc-support line segments," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 76–80.

[19] V. Patraucean, P. Gurdjos, and R. G. Von Gioi, "Joint a contrario ellipse and line detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 788–802, Apr. 2017.

[20] C. Lu, S. Xia, M. Shao, and Y. Fu, "Arc-support line segments revisited: An efficient high-quality ellipse detection," *IEEE Trans. Image Process.*, vol. 29, pp. 768–781, 2020.

[21] P. V. Hough, "Machine analysis of bubble chamber pictures," in *Proc. Int. Conf. High Energy Accel. Instrum.*, vol. 590914, 1959, pp. 554–558.

[22] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.

[23] L. A. F. Fernandes and M. M. Oliveira, "Real-time line detection through an improved Hough transform voting scheme," *Pattern Recognit.*, vol. 41, no. 1, pp. 299–314, Jan. 2008.

[24] A. Etemadi, "Robust segmentation of edge data," in *Proc. Int. Conf. Image Process. Appl.*, 1992, pp. 311–314.

[25] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting straight lines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 4, pp. 425–455, Jul. 1986.

[26] A. Desolneux, L. Moisan, and J.-M. Morel, "Edge detection by Helmholtz principle," *J. Math. Imag. Vis.*, vol. 14, no. 3, pp. 271–284, May 2001.

[27] C. Akinlar and C. Topal, "EDLines: A real-time line segment detector with a false detection control," *Pattern Recognit. Lett.*, vol. 32, no. 13, pp. 1633–1642, Oct. 2011.

[28] C. Topal and C. Akinlar, "Edge drawing: A combined real-time edge and segment detector," *J. Vis. Commun. Image Represent.*, vol. 23, no. 6, pp. 862–872, Aug. 2012.

[29] E. R. Davies, "A modified Hough scheme for general circle location," *Pattern Recognit. Lett.*, vol. 7, no. 1, pp. 37–43, Jan. 1988.

[30] J. Illingworth and J. Kittler, "A survey of the Hough transform," *Comput. Vis., Graph., Image Process.*, vol. 44, no. 1, pp. 87–116, 1988.

[31] J. Matas and C. G. J. Kittlery, "Progressive probabilistic Hough transform," in *Proc. Brit. Mach. Vis. Conf.*, 1998, pp. 14–17.

[32] C. Kimme, D. Ballard, and J. Sklansky, "Finding circles by an array of accumulators," *Commun. ACM*, vol. 18, no. 2, pp. 120–122, Feb. 1975.

[33] T. J. Atherton and D. J. Kerbyson, "Size invariant circle detection," *Image Vis. Comput.*, vol. 17, no. 11, pp. 795–803, Sep. 1999.

[34] D. Shaked, O. Yaron, and N. Kiryati, "Deriving stopping rules for the probabilistic Hough transform by sequential analysis," *Comput. Vis. Image Understand.*, vol. 63, no. 3, pp. 512–526, May 1996.

[35] H. Muammar and M. Nixon, "Approaches to extending the Hough transform," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 1989, pp. 1556–1559.

[36] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: Randomized Hough transform (RHT)," *Pattern Recognit. Lett.*, vol. 11, no. 5, pp. 331–338, May 1990.

[37] L. Xu, "Randomized Hough transform (RHT): Basic mechanisms, algorithms, and computational complexities," *Comput. Vis. Image Understand.*, vol. 57, no. 2, pp. 131–154, Mar. 1993.

[38] J. H. Han, L. Kóczy, and T. Poston, "Fuzzy Hough transform," *Pattern Recognit. Lett.*, vol. 15, no. 7, pp. 649–658, Jul. 1994.

[39] N. Kiryati, H. Kälviäinen, and S. Alaoutinen, "Randomized or probabilistic Hough transform: Unified performance evaluation," *Pattern Recognit. Lett.*, vol. 21, nos. 13–14, pp. 1157–1164, Dec. 2000.

[40] N. Kiryati, Y. Eldar, and A. M. Bruckstein, "A probabilistic Hough transform," *Pattern Recognit.*, vol. 24, no. 4, pp. 303–316, Jan. 1991.

[41] J. Y. Goulermas and P. Liatsis, "Incorporating gradient estimations in a circle-finding probabilistic Hough transform," *Pattern Anal. Appl.*, vol. 2, no. 3, pp. 239–250, Aug. 1999.

[42] H. K. Yuen, J. Illingworth, and J. Kittler, "Detecting partially occluded ellipses using the Hough transform," *Image Vis. Comput.*, vol. 7, no. 1, pp. 31–37, Feb. 1989.

[43] B. Yuan and M. Liu, "Power histogram for circle detection on images," *Pattern Recognit.*, vol. 48, no. 10, pp. 3268–3280, Oct. 2015.

[44] M. Liu, B. Yuan, and J. Bai, "Ellipse detection on images using conic power of two points," in *Proc. BMVC*, 2018, p. 215.

[45] T.-C. Chen and K.-L. Chung, "An efficient randomized algorithm for detecting circles," *Comput. Vis. Image Understand.*, vol. 83, no. 2, pp. 172–191, Aug. 2001.

[46] K. Chung and Y. Huang, "A pruning-and-voting strategy to speed up the detection for lines, circles, and ellipses," *J. Inf. Sci. Eng.*, vol. 24, no. 2, pp. 503–520, 2008.

[47] K.-L. Chung, Y.-H. Huang, S.-M. Shen, A. S. Krylov, D. V. Yurin, and E. V. Semeikina, "Efficient sampling strategy and refinement strategy for randomized circle detection," *Pattern Recognit.*, vol. 45, no. 1, pp. 252–263, Jan. 2012.

[48] L. Jiang, Z. Wang, Y. Xu, M. Liu, and J. He, "An improved randomized algorithm for detecting circles," *AMSE J.*, vol. 59, no. 1, pp. 177–188, 2006.

[49] S. J. Ahn, W. Rauh, and H.-J. Warnecke, "Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola," *Pattern Recognit.*, vol. 34, no. 12, pp. 2283–2303, Dec. 2001.

[50] D. S. Barwick, "Very fast best-fit circular and elliptical boundaries by chord data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1147–1152, Jun. 2009.

[51] J. Liang, M. Zhang, D. Liu, X. Zeng, O. Ojowu, K. Zhao, Z. Li, and H. Liu, "Robust ellipse fitting based on sparse combination of data points," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2207–2218, Jun. 2013.

[52] D. Liu and J. Liang, "A Bayesian approach to diameter estimation in the diameter control system of silicon single crystal growth," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 4, pp. 1307–1315, Apr. 2011.

[53] C. Arellano and R. Dahyot, "Robust ellipse detection with Gaussian mixture models," *Pattern Recognit.*, vol. 58, pp. 12–26, Oct. 2016.

[54] H. Dong, D. K. Prasad, and I.-M. Chen, "Accurate detection of ellipses with false detection control at video rates using a gradient analysis," *Pattern Recognit.*, vol. 81, pp. 112–130, Sep. 2018.

[55] M. Fornaciari, A. Prati, and R. Cucchiara, "A fast and effective ellipse detector for embedded vision applications," *Pattern Recognit.*, vol. 47, no. 11, pp. 3693–3708, Nov. 2014.

[56] L.-Q. Jia, C.-Z. Peng, H.-M. Liu, and Z.-H. Wang, "A fast randomized circle detection algorithm," in *Proc. 4th Int. Congr. Image Signal Process. (CISP)*, vol. 2, 2011, pp. 820–823.

[57] Q. Jia, X. Fan, Z. Luo, L. Song, and T. Qiu, "A fast ellipse detector using projective invariant pruning," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3665–3679, Aug. 2017.

[58] M. K. Leung and Y.-H. Yang, "Dynamic two-strip algorithm in curve fitting," *Pattern Recognit.*, vol. 23, nos. 1–2, pp. 69–79, Jan. 1990.
[59] A. Y.-S. Chia, S. Rahardja, D. Rajan, and M. K. Leung, "A split and merge based ellipse detector with self-correcting capability," *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 1991–2006, Jul. 2011.
[60] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Comput. Graph. Image Process.*, vol. 1, no. 3, pp. 244–256, Nov. 1972.
[61] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica, Int. J. Geograph. Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, Dec. 1973.
[62] H. Dong, I.-M. Chen, and D. K. Prasad, "Robust ellipse detection via arc segmentation and classification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 66–70.
[63] D. Wolters and R. Koch, "Combined precise extraction and topology of points, lines and curves in man-made environments," in *Proc. German Conf. Pattern Recognit.* Cham, Switzerland: Springer, 2017, pp. 115–125.
[64] A. Pridmore, J. Porrill, and J. Mayhew, "Segmentation and description of binocularly viewed contours," *Image Vis. Comput.*, vol. 5, no. 2, pp. 132–138, May 1987.
[65] G. A. West and P. L. Rosin, "Multistage combined ellipse and line detection," in *Proc. BMVC*. London, U.K.: Springer, 1992, pp. 197–206.
[66] R. Bullock, "Least-squares circle fit," Develop. Testbed Center, Boulder, CO, USA, Tech. Rep., 2006, pp. 1–3. [Online]. Available: https://www.dtcenter.org/sites/default/files/community-code/met/docs/write-ups/circle_fit.pdf
[67] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *Proc. Joint Pattern Recognit. Symp.* Berlin, Germany: Springer, 2003, pp. 236–243.
[68] A. Desolneux, L. Moisan, and J.-M. Morel, "Meaningful alignments," *Int. J. Comput. Vis.*, vol. 40, no. 1, pp. 7–23, Oct. 2000.
[69] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis: A Probabilistic Approach*, vol. 34. New York, NY, USA: Springer, 2007.
[70] A. Desolneux, L. Moisan, and J. Morel, "A grouping principle and four applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 4, pp. 508–513, Apr. 2003, doi: 10.1109/TPAMI.2003.1190576.
[71] J. L. Lisani and J.-M. Morel, "Detection of major changes in satellite images," in *Proc. Int. Conf. Image Process. (ICIP)*, vol. 1, Sep. 2003, pp. 941–944, doi: 10.1109/ICIP.2003.1247119.
[72] J. Delon, A. Desolneux, J.-L. Lisani, and A. B. Petro, "A nonparametric approach for histogram segmentation," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 253–261, Jan. 2007, doi: 10.1109/TIP.2006.884951.
[73] J. L. Lisani, L. Rudin, and A. Buades, "Fast video search and indexing for video surveillance applications with optimally controlled false alarm rates," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2011, pp. 1–6, doi: 10.1109/ICME.2011.6012151.
[74] J.-L. Lisani, S. Ramis, and F. J. Perales, "A contrario detection of faces: A case example," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 2091–2118, Jan. 2017, doi: 10.1137/17M1118774.
[75] V. Pătrăucean, P. Gurdjos, and R. G. Von Gioi, "A parameterless line segment and elliptical arc detector with enhanced ellipse fitting," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2012, pp. 572–585.
[76] R. Halır and J. Flusser, "Numerically stable direct least squares fitting of ellipses," in *Proc. 6th Int. Conf. Central Eur. Comput. Graph. Vis. (WSCG)*, vol. 98, 1998, pp. 125–132.
[77] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 476–480, May 1999.
[78] T. Atherton and D. Kerbyson, "Using phase to represent radius in the coherent circle Hough transform," *IEE Colloq. Hough Transforms*, 1993, pp. 1–5.
[79] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category database," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. 7694, 2016.
[80] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A line segment detector," *Image Process. Line*, vol. 2, pp. 35–55, Mar. 2012.

ONOFRE MARTORELL received the degree in mathematics from the University of Balearic Islands, in 2016, and the M.Sc. degree in computer vision from the Autonomous University of Barcelona, in 2017. He is currently pursuing the Ph.D. degree with the University of Balearic Islands under a predoctoral grant.

His research interests include computer vision and mathematical image analysis. He is also a member of the Institute of Applied Computing and Community Code (IAC3).

ANTONI BUADES received the Ph.D. degree in mathematics from the University of Balearic Islands (UIB), Spain, in 2006.

He is currently an Associate Professor with UIB. His research interests include mathematical analysis of digital images and video, particularly, restoration, demosaicking, super-resolution, registration, medical imaging, satellite imaging, and deep learning. He is also a member of the Institute of Applied Computing and Community Code (IAC3).

JOSE LUIS LISANI received the Ph.D. degree in computer science and applied mathematics from the Universities of Illes Balears, Spain, and the Ph.D. degree from Paris-Dauphine, France, in 2001.

He is currently an Associate Professor with the University of Illes Balears. He has coauthored the book *A Theory of shape identification* (Springer LNM, 2008). His research interests include the analysis and processing of color images and video sequences. He is also a member of the Institute of Applied Computing and Community Code (IAC3).