Real-Time Robotic Manipulation of Cylindrical Objects in Dynamic Scenarios Through Elliptic Shape Primitives

Huixu Dong, Ehsan Asadi, Guangbin Sun, Dilip K. Prasad, and I-Ming Chen, *Fellow, IEEE*

***Abstract*—Robotic manipulation employs the object detection in images to create a scene awareness and locate an object's pose. In dynamic scenarios, fast multiobject detection and tracking are crucial. Many objects commonly found in household and industrial environments are represented by cylindrical shapes. Thus, it is available for robots to manipulate them through the real-time detection of elliptic shape primitives formed by the circular tops of these objects. We devise an efficient algorithm of the detection of elliptic shape primitives, which in turn enables robust and real-time robotic manipulations of such objects. The proposed algorithm incorporates the information of elliptic edge curvature, splits complex curves into arcs, classifies the arcs into different quadrants of a candidate elliptic shape, determines the quality of arc selection for ellipse fitting, and then retrieves the corresponding elliptic shape primitive. Our algorithm provides either faster or more accurate ellipse detection results than the current state-of-the-art methods, irrespective of challenging scenarios such as occluded or overlapping ellipses. This is verified by performance comparison with six state-of-the-art elliptic shape detection algorithms on four public image datasets. The algorithm has been integrated on robots to demonstrate the ability to carry out accurate robotic manipulations (tracking, grasping, and stacking) of cylindrical objects in real time. We show that the robotic manipulator, empowered by the elliptic shape primitive algorithm, performs well in complex manipulation experiments as well as dynamic scenarios.**

***Index Terms*—Cylindrical object detection, dynamic scenarios, ellipse detection, robotic manipulation.**

I. Introduction and Related Work

AS ROBOTS are being incorporated in sociourban scenarios, robotic manipulation in dynamic and cluttered scenarios with sufficient accuracy and real-time execution is needed. We discuss specific manipulation tasks of robotic perception and grasp of objects in cluttered and dynamic scenes. A comprehensive literature review of robotic perception and grasp is presented in [1] and [2]. Early perception for robotic grasp makes use of a set of two-dimensional (2-D) key points and the preferred edge features to detect the object and deduce the object position [3]–[5]. Since an RGB-D sensor is available at a low cost, the robotic perception based on the three-dimensional (3-D) descriptor is more feasible than ever. This work further combines local template descriptor, shape, depth, and scene semantic as features to match the objects [6], [7]. Learning methods have proven effective for a wide range of robotic perception problems, allowing a perception system to learn a mapping from some feature sets to various visual properties [8], [9]. The grasp synthesis problem is formulated as finding a successful grasp configuration, which is a planar position and an orientation visualized by rectangle representation [10], [11]. However, this method is not suitable for precise positioning for manipulations. Better precision can indeed be accomplished with modern object detection and deep learning techniques [12]. However, their performance is training dependent which is computationally demanding and limited by the training dataset. Yet, the execution time while detection may not necessarily be real time [13]. As different grasp types are needed for different objects, some researchers define grasp strategies based on the shape primitives [14], i.e., cylinders, boxes, or spheres. For example, most cylindrical objects can be grasped with a lateral cylindrical wrap or top grasp using two or more fingers, depending on the type of the gripper, while other strategies such as prismatic grasp is preferred for a cubic object.

Cylindrical objects are characterized by ellipses in the images depicting the circular top of cylinder at most angles of observation; see Fig. 1 for an example. Many objects commonly found in household (e.g., cans, cups, bottles, batteries, candles) and industrial environments (e.g., plastic/metal caps, pipes, screw holes) are represented by cylindrical shape models. Tracking and robotic manipulation of cylindrical objects in dynamic scenarios can be enabled by the real-time and accurate detection of elliptic shape primitives. Although this approach hugely simplifies the problem of creating robotic scene awareness in the context of cylindrical objects, detection of elliptic shapes for this application is challenging due to the following reasons. The manipulative action of robot implies that the scene acquired by robot's camera changes dynamically and the object of interest may get occluded behind other objects or may appear overlapping with other objects. Detection of occluded or overlapping ellipses in real time in complex scenes is challenging.

Besides being relevant to robotic manipulation [15]–[21], ellipse detection in images has been deemed useful in other

Manuscript received August 14, 2017; revised August 14, 2018; accepted August 27, 2018. This paper was recommended for publication by Associate Editor Y. Sun and Editor A. Kheddar upon evaluation of the reviewers' comments. *(Corresponding author: Huixu Dong.)*

The authors are with Nanyang Technological University, Singapore 639798, Singapore (e-mail: dong0076@e.ntu.edu.sg; asadi.ehsan@yahoo.com; guangbin.sun@gmail.com; dilipprasad@gmail.com; michen@ntu.edu.sg).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2018.2868804

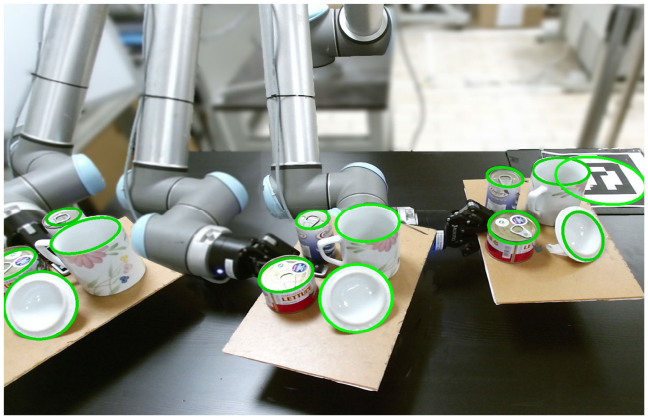

Fig. 1. Tracking ellipses in a dynamic scenario.

applications as well, including industrial inspection [22], medical diagnosis [23], [24], recognition of traffic signs [25], security [26], autonomous navigation [15], [27], and tracking targets [28], [29]. Thus, there is a body of work related to ellipse detection in the allied field of image processing and computer vision. For the benefit of interested readers, we have included a review of contemporary ellipse detection approaches in the Appendix. We note here that the current methods of ellipse detection are unsuitable for robotic manipulation. Robotic manipulation requires real-time and accurate detection of elliptic shape primitives, whereas the current methods are marred with slow execution due to heavy computational loads or low ellipse detection accuracy, sometimes both. In this paper, we address the need of real-time and accurate detection of elliptic shape primitives for the purpose of robotic manipulation.

In this paper, we propose a novel solution for robotic manipulation of cylindrical objects through a fast and accurate ellipse detection approach from the edge map of RGB or gray images captured using robotic camera sensor. Ellipse detection performance, as relevant to manipulation task, can be characterized by *F*-measure (balance between precision and recall of detection of ellipses) and average time taken by our ellipse detection algorithm to detect the ellipses in each image. Our approach balances precision and recall using gradient and curvature properties of arcs from elliptic shapes to cluster suitable arcs and fit ellipses to the clustered arcs. Real-time execution is achieved by two mechanisms. First, gradient and curvature properties are computed only once and reused in several steps of the algorithm. Second, computationally inexpensive and simple mechanisms for clustering and ellipse fitting have been devised. In addition, multiple filters at various stages incrementally exclude irrelevant arcs, thus improving the *F*-measure and reducing the computation time simultaneously. We show superior performance in ellipse detection as compared to state-of-the-art algorithms and demonstrate its successful incorporation in robotic manipulation experiments in dynamic scenarios.

We highlight the novelties of our work. Foremost, our work provides a first solution, to the best of our knowledge, for robotic manipulation of cylindrical objects in dynamic environments using elliptic shape primitives only. The second novelty is our vision of repeated use of inexpensively computed but robust geometric properties of arcs of ellipses, such as smoothness, continuity, concavity–convexity, etc. Our algorithm also incorporates several novel aspects. First, we use a novel way of analyzing convexity and concavity properties, which is more accurate and costs less computation than the contemporary methods [30]. Removal of inflexion points and turning points make it further robust. Second, we use a simple and cost-effective approach for determining ellipse centers with better accuracy than the existing approaches. Third, we use a new arc group filtering strategy, namely distance constraint of ellipse fitting for selecting better arc groups, and employ length ratios to validate ellipse detections.

The rest of this paper is organized as follows. Section II presents the proposed algorithm for detection of elliptic shape primitives. Section III investigates the algorithm controls and compares its performance with the state-of-the-art algorithms in ellipse detection. It also verifies suitability of the algorithm for robotic manipulation. Section IV presents integration of the algorithm in a robotic system. Section V provides results of robotic experiments for stationary cylinders as well as dynamic scenarios. This paper is concluded in Section VI.

II. Detection of Elliptic Shapes in Images

Our algorithm for detection of elliptic shapes in images comprises four main blocks, namely, preprocessing, arc processing, ellipse fitting, and ellipse validation.

A. Preprocessing

Edge image of the input image is obtained using Canny edge detector [31] with autothresholding. Every edge point p_i with coordinates (x_i, y_i) is classified into two classes "+" and "–," based on the sign of the gradient η_i at p_i, as follows:

$$\mathrm{X}(p_i) = \begin{cases} + \text{ if } \tan(\eta_i) > 0 \\ - \text{ if } \tan(\eta_i) < 0. \end{cases} \tag{1}$$

Although it is difficult to obtain the accurate values of η_i in digital images [32], $\mathrm{X}(p_i)$ is not sensitive to the accuracy of η_i. Sobel derivatives compute η_i inexpensively. The edge points that encounter trivial cases of dx or dy being zero are discarded. We note that if an edge curve is hypothetically on an ellipse, the edge points with the positive gradient directions, i.e., $\mathrm{X}(p_i) = +$, rest on the first and third quadrants of the ellipse $(p_i \in \{\mathrm{I} \cup \mathrm{III}\})$ while the other arcs belong to the second and fourth quadrants of the ellipse $(p_i \in \{\mathrm{II} \cup \mathrm{IV}\})$, as illustrated in Fig. 2. Edge curves are identified using 8-connectivity of two consecutive edge points.

First stage of filtering is now incorporated in two steps. Short edge curves may not have sufficient information for ellipse detection. Thus, curves with the number of edge points less than T_n are removed. In the second step, edge curves with almost no curvature, i.e., almost linear curves are removed. We use an oriented bounding box with the minimum area, which is denoted as $\mathrm{OBB}_{\min}$, to enclose a curve [33]. If the minimum side length of $\mathrm{OBB}_{\min}$ is less than a threshold T_O, we can discard this curve.

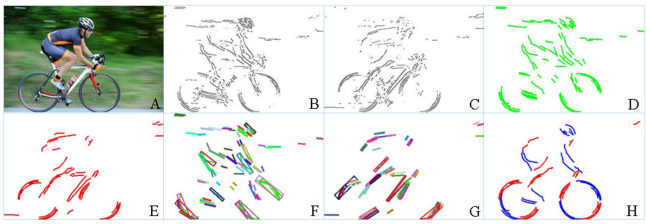

Fig. 2. Example of preprocessing. (A) Initial image. (B) and (C) Edge images along two edge directions by gradient signs. (D) and (E) Edge images after the process of 8-connectivity. (F) and (G) Arcs enclosed by oriented bounding boxes. (H) Image after preprocessing.

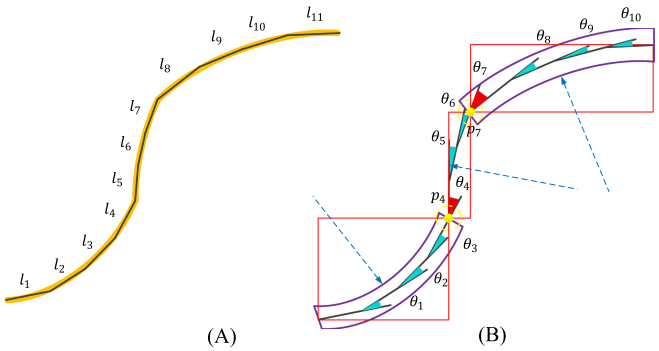

Fig. 3. (A) Curve fitted by a series of line segments. Arc extraction by splitting an arc at a change of direction of curvature p_4 and a large change of curvature p_7. Line segments with arrows represent the directions of curvatures of arcs and the red rectangle frames denote bounding boxes enclosing arcs extracted.

B. Arc Processing

This block is divided into three parts. The first part extracts smooth and continuous arcs. The second part classifies into four quadrants. The third part groups the arcs such that an ellipse can be fit on each group.

1) Arc Selection: In this step, continuous and smooth arcs are identified from the edge curves. This is done by splitting curves where sudden changes in curvature are encountered. The sudden change may be in terms of the amount (referred to as turning point) or the direction of change (referred to as an inflexion point). We perform this as follows. We fit an edge curve by a series of line segments using [34] so that the curve can be denoted as $\{l_1, l_2, \ldots, l_N\}$, as shown in Fig. 3(A). The line segment approximation of a curve reduces the computational cost since the subsequent calculation is only performed on the dominant points of the curve (i.e., the endpoint of the line segments of the curve) instead of all the points. The vector angle between consecutive line segments is defined as $\theta_i (i = 1, 2, \ldots, n)$ whose direction is from l_i to l_{i+1} and the range lies in between $-\pi$ and π. We set a threshold (T_θ) to evaluate the amount of the change between the ith angle θ_i and the $(i+1)$th angle θ_{i+1}. Thus, turning corners are determined as $|\boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i| > T_\theta$. T_θ is set sufficiently high to conclude a large change in amount as well as direction change in a curvature. Large difference between two consecutive angles (θ_i, θ_{i+1}) of the same sign indicates turning points and the large difference between two consecutive angles (θ_i, θ_{i+1}) of opposite signs

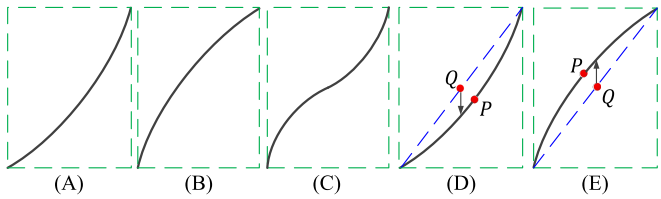

Fig. 4. Arc classification based on the convexity–concavity. A_u and A_l represent the upper area and lower area, respectively. (A) $A_u > A_l$. (B) $A_u < A_l$. (C) $A_u = A_l$. P and Q denote the midpoints of the arc and the line segment formed by the two endpoints of arc, respectively.

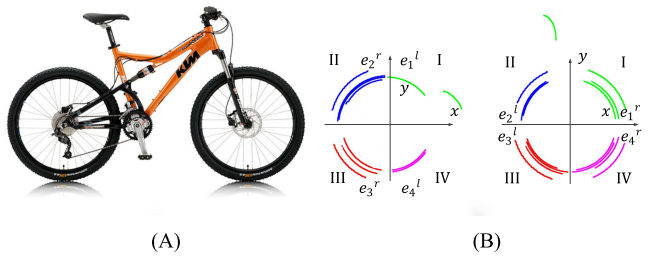

Fig. 5. Arc classification. ${e_i}^l$ and ${e_i}^r$ denote the left and right endpoints of an arc.

indicates inflexion points. The algorithm of arc extraction is given in Algorithm I after the Appendix. The filtering steps of preprocessing block are used again to discard arcs with insufficient length or curvature.

2) Arc Grouping: Arcs from the same ellipse must satisfy the corresponding convexity–concavity for each other. In our approach, we define convexity and concavity as follows (illustrated in Fig. 4): if the midpoint position of an arc is higher than the midpoint position of a line segment formed by the endpoints of an arc, the arc is convex; if it is lower, then the arc is concave; and if a decision cannot be made, the arc is discarded. We denote the concavity and convexity as

$$\Theta(e) = \begin{cases} + \text{ if } e \text{ is } & \text{convex} \\ - \text{ if } e \text{ is } & \text{concave.} \end{cases} \tag{2}$$

From our definition, the arcs in the first and second quadrants of an ellipse are convex and the arcs in the third and fourth quadrants are concave. Thus, $\Theta(e)$ and $\mathrm{X}(p_i)$ of the arc are used to classify each arc into an unique quadrant; see Fig. 5, for example. We note that authors in [30] and [35] use the difference between the upper and lower areas to determine the convexity–concavity of arcs. Computation of areas is more expensive than computation of the relative positions. Moreover, these methods fail if the arc has inflexion points; see Fig. 4(C), for example. Since our algorithm already removes inflexion points, its performance on the detection accuracy is better than others. Verifications are provided in the experiments.

3) Arc Grouping Based on Geometric Constraints: A set of three arcs $\tau_{\mathrm{abc}} = (e_a, e_b, e_c)$ are chosen as a triplet depending on the below grouping constraints, which indicate that the arcs likely belong to the same ellipse. Accordingly, four combinations of a triplet of arcs that do not belong

to the same quadrant are made, as follows: $(e_a, e_b, e_c) \in \{(\mathrm{I}, \mathrm{II}, \mathrm{III}), (\mathrm{I}, \mathrm{II}, \mathrm{IV}), (\mathrm{II}, \mathrm{III}, \mathrm{IV}), (\mathrm{I}, \mathrm{III}, \mathrm{IV})\}$ where a, b, c are the indexes of different arcs, as shown in Fig. 5. We choose a pair of arcs $p_{\mathrm{ab}} = (e_a, e_b)$ from the two subsequent quadrants in counterclockwise order, such as $(e_a, e_b) \in \{(\mathrm{I}, \mathrm{II}), (\mathrm{II}, \mathrm{III}), (\mathrm{III}, \mathrm{IV}), (\mathrm{IV}, \mathrm{I})\}$. Thus, a triplet is made up of two pairs of arcs sharing an arc, $\tau_{\mathrm{abc}} = \{(p_{\mathrm{ab}}, p_{\mathrm{cd}}) | \ e_b \equiv e_d\}$. We use the relative position among arcs to sort arcs sets. For instance, in Fig. 5, along the horizontal axis, the coordinates of ${e_1}^l$ and ${e_4}^l$ must be bigger than that of ${e_2}^r$ and ${e_3}^r$.

Similarly, along the vertical axis, the coordinates of ${e_3}^l$ and ${e_4}^r$ must be smaller than that of ${e_2}^l$ and ${e_1}^r$ (see Fig. 5). A tolerance of 1 pixel between two sides of inequalities is included because the pixels on the coordinate axes have been discarded. The locations of centers of such arcs are also considered as an additional constraint for selecting potential arcs. If the centers of such arcs in a triplet rest on the same area within a range, it is highly likely that they fit the same ellipse. Sometimes, arcs belonging to different ellipses may merge together to fit one ellipse. The constraint of the distances between midpoints of arcs and the centers of arcs is applied to avoiding false detections.

C. Ellipse Fitting and Parameter Determination

We split the task of determining the parameters of ellipses into three subtasks, confirming the elliptic centers, using distance constraint for ellipse fitting, and determining the remaining parameters.

1) Confirming the Elliptic Centers: The geometric characteristics used for retrieving the centers of the ellipses are presented here. The geometric properties of points and tangents in ellipses are used for finding elliptic centers [36]–[40]. The elliptic center is the intersection of the lines connecting the midpoints of chords and the intersection formed by the tangents through the endpoints of arcs. Although the method can detect the center of a small ellipse, the slopes of tangents could contain many errors due to the image noise and the digitization of the image [35], as shown in Fig. 6(A). Our new approach avoids this case, as described next.

The arc pair whose arcs satisfy two geometric constraints 1) convexity–concavity; and 2) relative positions are chosen for calculating elliptic center. The center is estimated based on the maximum distances among endpoints of arcs for a pair of arcs. Specifically, endpoints of a pair of arcs can form $\binom{4}{2}$ line segments. Fig. 6(B) shows an example. We extract endpoints (P_1, P_3, P_4) forming the two longest line segments (P_1P_3, P_1P_4). (P_{13}, P_{14}) are intersection points of the corresponding tangents. (M_{13}, M_{14}) are midpoints of the two line segments (P_1P_3, P_1P_4). The elliptic center is the intersection of two lines (l_{13}, l_{14}) passing through intersection points (P_{13}, P_{14}) and midpoints (M_{13}, M_{14}).

The coordinates and gradient of endpoints of an arc are expressed as $\{x_i, y_i, \theta_i | i = 1, 2, 3, 4\}$. The coordinates of midpoints (M_{13}, M_{14}) of the line segments P_1P_3 and P_1P_4 are derived by (3). Similarly, the coordinates of intersections (P_{13}, P_{14}) are expressed in (4). We calculate the slope q_1 of

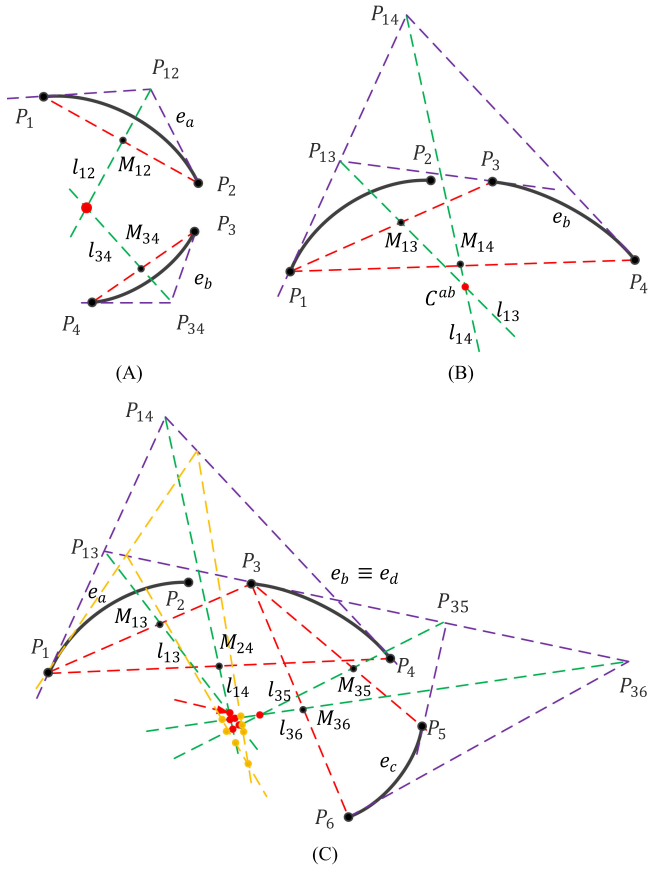

Fig. 6. Illustration of the calculation method of the elliptic center. The methods of estimating the elliptic center proposed by (A) others and (B) and (C) us. (B) Red line segments are the two longest ones in all line segments. P_i denotes endpoints of arcs e_a, e_b, e_c; P_{ij} represents the intersection of the tangent lines through P_i and P_j. M_{ij} is the midpoint of the line segment P_iP_j. l_{ij} indicates the line formed by P_{ij} and M_{ij}. The shifted intersection points (yellow dots) are formed by the shifted lines (yellow lines) joining the midpoints of the chords and the intersections of the tangent lines when the slope of the line through the midpoint P_{34} of the arc a_b is not correct due to image noise. $i, j = 1, 2, 3, 4, 5, 6$.

the line segment P_1P_3 by (5) and the slope q_2 of the line l_{13} passing M_{13} and P_{13} is obtained through (6):

$$
\begin{aligned}
x_{M_{13}} &= \frac{x_1 + x_3}{2}, y_{M_{13}} = \frac{y_1 + y_3}{2} \\
x_{M_{14}} &= \frac{x_1 + x_4}{2}, y_{M_{14}} = \frac{y_1 + y_4}{2}
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
x_{P_{13}} &= \frac{y_1 - \theta_1 x_1 - y_3 + \theta_3 x_3}{\theta_3 - \theta_1}, \\
y_{P_{13}} &= \frac{\theta_1 y_3 - \theta_3 y_1 + \theta_3 \theta_1 (x_1 - x_3)}{\theta_3 - \theta_1} \\
x_{P_{14}} &= \frac{y_1 - \theta_1 x_1 - y_4 + \theta_4 x_4}{\theta_4 - \theta_1}, \\
y_{P_{14}} &= \frac{\theta_1 y_4 - \theta_4 y_1 + \theta_4 \theta_1 (x_1 - x_4)}{\theta_4 - \theta_1}
\end{aligned} \tag{4}
$$

$$
q_1 = \frac{y_1 - y_3}{x_1 - x_3}, q_3 = \frac{y_1 - y_4}{x_1 - x_4} \tag{5}
$$

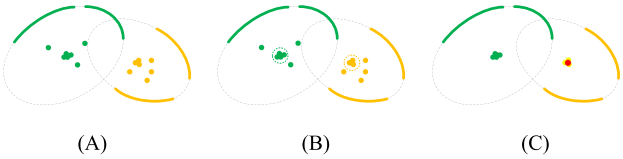

Fig. 7. Procedure of a center estimation by the mean-shift clustering algorithm. (A) Intersections generated by two pairs of arcs. (B) Center clusters enclosed by the dash circles with the mean-shift algorithm. (C) Determined center.

$$q_2 = \frac{y_{P_{13}} - y_{M_{13}}}{x_{P_{13}} - x_{M_{13}}} = \frac{(\theta_3 + \theta_1)(y_3 - y_1) + 2\theta_3\theta_1(x_1 - x_3)}{2(y_3 - y_1) - (\theta_3 + \theta_1)(x_1 - x_3)},$$
$$q_4 = \frac{y_{P_{14}} - y_{M_{14}}}{x_{P_{14}} - x_{M_{14}}} = \frac{(\theta_4 + \theta_1)(y_4 - y_1) + 2\theta_4\theta_1(x_1 - x_4)}{2(y_4 - y_1) - (\theta_4 + \theta_1)(x_1 - x_4)}. \tag{6}$$

Thus, the coordinates of the center C^{ab} can be derived according to the proposed method as follows:

$$x_{C^{\mathrm{ab}}} = \frac{y_{M_{14}} - q_4 x_{M14} - y_{M13} + q_2 x_{M_{13}}}{q_2 - q_4},$$
$$y_{C^{\mathrm{ab}}} = \frac{q_2 y_{M_{14}} - q_4 y_{M_{13}} + q_2 q_4 (x_{M_{14}} - x_{M_{13}})}{q_2 - q_4}. \tag{7}$$

Using the same method, we can obtain the center of a pair of arcs $p_{\mathrm{cd}} = (e_c, e_d)\ |e_d \equiv e_b$. We consider that a set of arcs $\tau_{\mathrm{abc}} = \{(p_{\mathrm{ab}}, p_{\mathrm{cd}})|\ e_b \equiv e_d\}$ can consist of the same ellipse if and only if the distance of the centers of two pairs $(p_{\mathrm{ab}}, p_{\mathrm{cd}})$ lies within a given threshold T_c.

Theoretically, six intersections formed by a set of arcs τ_{abc} should be coincident with each other. However, due to the presence of noise and inaccurate edge angles, the geometric centers will shift into neighborhoods of initial points such that these centers show a distribution in the parameter space of ellipse, as illustrated in Fig. 6(C). Unlike conventional detection methods of ellipse centers by obtaining the median of the coordinates of a set of intersections in [30] and [41], we also generate the center whose coordinates are obtained by the iterative mean-shift clustering algorithm [42] in Fig. 7(C).

2) Distance Constraint for Ellipse Fitting: After determining elliptic center, we filter the arc groups to select better groups for ellipse fitting and reduce false positives. In general, the distances between points on arcs and the elliptic center should fall in a suitable range. For simplicity and computation efficiency, we just use midpoints of arcs to get the distance between a point on an arc and the elliptic center. We define the following function $\Gamma(e_a, e_b, e_c)$ to describe the distance constraint for discarding the false-positive detections:

$$\Gamma(e_a, e_b, e_c) = \frac{\max(d_a, d_b, d_c)}{\min(d_a, d_b, d_c)} < T_f$$

where $\max(d_a, d_b, d_c)$ and $\min(d_a, d_b, d_c)$ are the estimated minimum and maximum values among three distances d_a, d_b, d_c between midpoints P_a, P_b, P_c of three arcs e_a, e_b, e_c and the center C estimated in the previous section, respectively, as seen in Fig. 8. If the distance constraint $\Gamma(e_a, e_b, e_c) > T_f$ is satisfied, the arc is retained for the detection of ellipses. The threshold

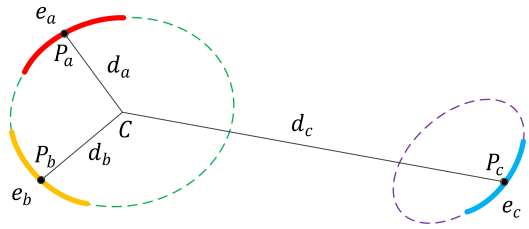

Fig. 8. Arcs at different ellipses merging one ellipse.

T_f is set to 5 in this work. The method ensures that an incorrect ellipse corresponding to these three arcs is not detected since $\Gamma(e_a, e_b, e_c)$ of a triplet of arcs is beyond the corresponding threshold T_f.

3) Estimation of Ellipse Parameters: To obtain the remaining parameters, we decompose the parameter space of an ellipse for the ratio N of the ellipse minor semiaxes length B to major semiaxes length A and other defined parameter K ($K = \tan\rho$, ρ is the orientation of an ellipse), as described in [30], [40], [43], and [44]. The equation that expresses N in terms of K is provided as

$$N^2 = -\frac{(q_1 - K)(q_2 - K)}{(1 + q_1 K)(1 + q_2 K)} \tag{8}$$

with $\alpha = q_1 q_2 - q_3 q_4$ and $\beta = q_2 q_4 (q_3 - q_1) + q_1 q_3 (q_4 - q_2) + (q_1 + q_2 - q_3 - q_4)$. Thus,

$$K = \pm\sqrt{1 - \frac{\beta}{\alpha}}. \tag{9}$$

For each pair of points above, the $N - K$ accumulator is continuously updated according to (8) and we can get the highest peaks of the values of N and ρ are in two one-dimensional accumulators. Here, the ellipse polar equations and coordinate transformation relations are applied to transforming the coordinate (x_i, y_i) of a point on an arc in the world coordinate system to the coordinate (x_o, y_o) in the ellipse coordinate system. In two dimensions, the coordinate transformation is done in the following way:

$$\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} \cos\rho & \sin\rho \\ -\sin\rho & \cos\rho \end{bmatrix} \begin{bmatrix} x_i - x_c \\ y_i - y_c \end{bmatrix}. \tag{10}$$

According to (8)–(10), we obtain the following equations:

$$x_o = \frac{(x_i - x_c) + (y_i - y_c) K}{\sqrt{K^2 + 1}},$$
$$y_o = \frac{(y_i - y_c) - (x_i - x_c) K}{\sqrt{K^2 + 1}}. \tag{11}$$

Referred to [43], A_x is estimated as follows:

$$A_x = \sqrt{\frac{x_o^2 N^2 + y_o^2}{N^2 (1 + K^2)}}. \tag{12}$$

Finally, combining $A_x = A\cos\rho$ and $= B/A$, we can identify an ellipse candidate with a parameter set (x_c, y_c, N, K, A), as shown in Fig. 9.

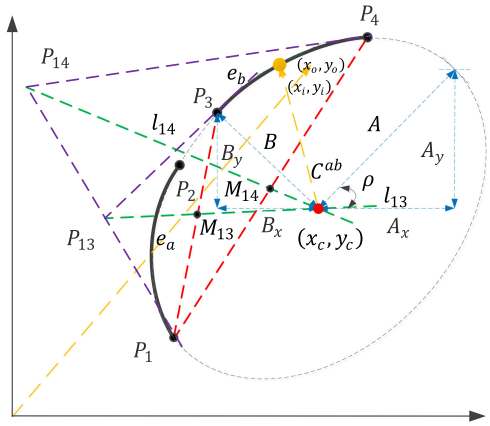

Fig. 9. Geometric schematic of parameter estimation.

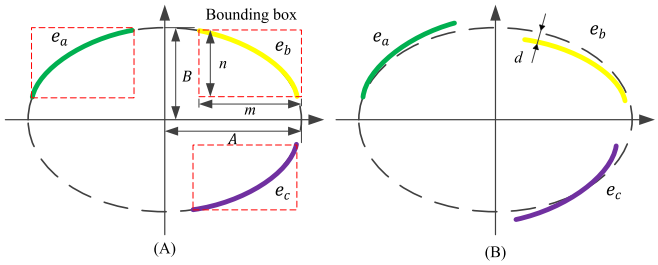

Fig. 10. (A) Ellipse validation by the ratio of the sum of m and n to the sum of A and B. (B) Ellipse validation via arc circumference ratio.

D. Ellipse Validation

Arcs that satisfy the three geometric constraints above still may consist of an invalid ellipse, such as false positives or duplicated ones. Therefore, we perform ellipse validation for filtering false detections by identification of invalid or duplicate ellipses.

1) Validation by the Ratio of the Length: The ratio of the sum of the long and width lengths of bounding box enclosing an arc to the sum of the major and minor semiaxes lengths (A, B, respectively) can be regarded as a measurement of circumference. Such scheme is less sensitive to the quantization problem in the pixel count feature than that of just considering independent pixel separately. Assuming that an ellipse is fit by the triplet $\tau = (a_1, a_2, a_3)$ enclosed by bounding boxes with the size of $m_i \times n_i (i = 1, 2, 3)$, we define a function $\mathrm{K}(\tau)$ as follows:

$$\mathrm{K}(\tau) = \frac{\sum_{i=1}^{3}(m_i + n_i)}{3(A + B)}. \tag{13}$$

A higher value of $\mathrm{K}(\tau)$ implies a larger probability of a candidate ellipse being a real one, as illustrated in Fig. 10(A). If $\mathrm{K}(\tau)$ exceeds a threshold (T_l), the ellipse candidate can be verified further by the below constraint; otherwise, it is discarded. We define a score $s(\mathrm{K}(\tau))$ of $\mathrm{K}(\tau)$, to represent the probability of a set of arcs consisting of a real ellipse. Based on a related concept, Prasad *et al.* [36], [23] use the angular circumference ratio which needs the computation of the angle subtended by each arc on the center, which increases additional burden. However, our validation strategy is significantly faster.

2) Validation via the Ratio of the Circumference: Supposing that the coordinate of point on an arc is (x_i, y_i), we put (x_i, y_i) into the following equations:

$$X = \frac{\left[(x_i - x_c)\cos\rho + (y_i - y_c)\sin\rho\right]^2}{A^2},$$
$$Y = \frac{\left[(y_i - y_c)\cos\rho - (x_i - x_c)\sin\rho\right]^2}{B^2} \tag{14}$$

where x_c and y_c represent the center and ρ donates the orientation of a detected ellipse. In Fig. 10(B), the differential distance d between a detected elliptic arc and an edge of a real ellipse arc is given as follows:

$$d = |X + Y - 1|. \tag{15}$$

If d is less than a threshold (T_d), it implies that the pixel (x_i, y_i) is close to the edge of the detected ellipse. We define a function $\psi(\tau_{\mathrm{abc}})$, which is the ratio of the number of a set of pixels $\mathcal{B}$ satisfying the above constraint to the number of pixels in the triplet τ_{abc} used for detecting an ellipse

$$\psi(\tau_{\mathrm{abc}}) = \frac{N_{\mathcal{B}}}{N_a + N_b + N_c} \tag{16}$$

where N_a, N_b, and N_c are the numbers of pixels on the three arcs, respectively, and $N_{\mathcal{B}}$ is the number of pixels in $\mathcal{B}$. If $\psi(\tau_{\mathrm{abc}})$ is more than a threshold (T_ψ), the detected ellipse is considered as a valid one; otherwise, this ellipse is given up. The corresponding score $s(\psi(\tau_{\mathrm{abc}}))$ whose value is $\psi(\tau_{\mathrm{abc}})$ is defined for clustering.

3) Clustering by Similarities Among Detected Ellipses: Multiple candidate ellipses may be fit to form the same "real" ellipse as multiple arc sets may belong to the same "real" ellipse. In order to ensure a "real ellipse" is just made up of one arc set, such candidate ellipses are clustered depending on the order of the scores [36]. All valid ellipses with the same center are ranked according to the total decreasing score $(s(\mathrm{K}(\tau)) + s(\psi(\tau_{\mathrm{abc}})))$. Then, we use the method proposed by Bascca *et al.* [45] to assess the similarity of two ellipses. Specifically, a feature vector is defined as $V(x_c, y_c, A, B, \rho)$ where (x_c, y_c) represents the coordinates of an ellipse center, A and B denote the lengths of the semimajor and semiminor axes, and ρ is the orientation of ellipse. The Euclidean distance between two feature vectors is used as the distinctiveness measure:

$$D(V, W) = \sqrt{\sum_{i=1}^{5}(E_{V,i} - E_{W,i})^2} \tag{17}$$

where $E_{V,i}$ and $E_{W,i}$ denote the ith parameters in the vectors V and W. If $D(V, W)$ is less than a threshold T_s, the ellipses V and W are concluded to belong to the same ellipse cluster. When an ellipse cannot be assigned to any cluster, it becomes a reference for a new cluster.

III. Algorithm Controls and Performance

In this section, we investigate the suitable values of the thresholds which control the performance of our algorithm. Further, we compare the performance of our algorithm with state-of-the-art ellipse detection algorithms, namely Jia *et al.* [41], Fornaciari

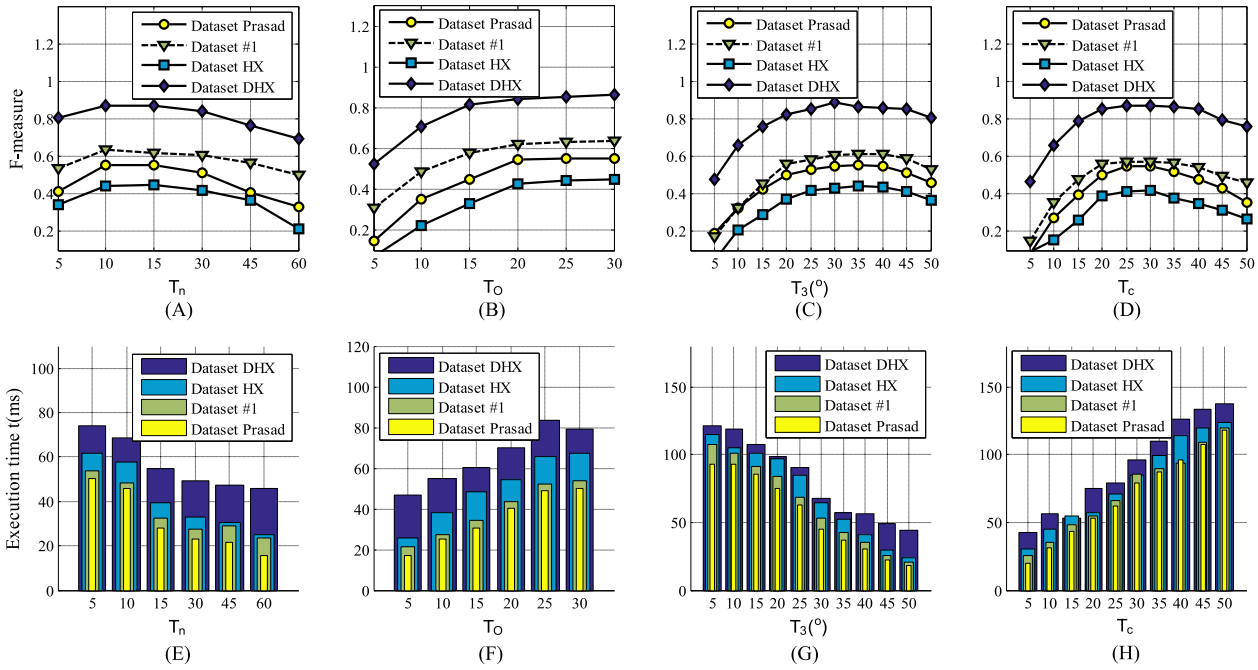

Fig. 11. Effectiveness values and execution times for four thresholds T_n, T_O, T_θ, and T_c. (A) Threshold on curve lengths. (B) Threshold on side ratios. (C) Threshold on intersection angles. (D) Threshold on centre distances. (E) Threshold on curve lengths. (F) Threshold on side ratios. (G) Threshold on intersection angles. (H) Threshold on centre distances.

et al. [30], Prasad *et al*. [36], Bai *et al*. [46], Liu *et al*. [47], and Mai *et al*. [48]. Their source codes in C++ or MATLAB are available online. All the experiments are performed on a PC with 8 GB RAM and an Intel Core i7 processor.

We use four public datasets with different characteristics, including Dataset Prasad [36], dataset#1 [30], Dataset HX and Dataset DHX. Dataset HX presents images taken in diverse backgrounds from sociourban scenarios, including households, industries, roads, etc. These images are randomly chosen from 16 categories in the ImageNet repository [49]. Dataset DHX is especially relevant to robotic manipulation since it contains images with cylindrical objects in household or office environments. These images are collected randomly by 20 volunteers. For taking each image, the volunteer placed a camera approximately 50 cm from the front edge of a platform and 50 cm above the platform surface, tilted down by approximately 45°. Some pictures are captured in a different perspective around the same type of scenario.

A. Evaluation Metrics

Performance metrics, such as *F*-measure and the execution time t [30], [36], [50], are used for quantitative comparison. For a detected ellipse ε_d and the ground-truth ellipse ε_g, the overlap ratio Φ is the Jaccard index of similarity between the ellipses ε_d and ε_g [36]. If the overlap ratio Φ satisfies $\Phi > \Phi_0$ with $\Phi_0 = 0.8$, the detected ellipse is considered a correct detection; otherwise, it is counted as a miss. Precision and Recall values are computed as follows:

$$\text{Precision} = \frac{\Omega}{N}, \ \text{Recall} = \frac{\Omega}{M} \tag{18}$$

where Ω donates the number of correctly detected ellipses, N is the number of detected ellipses, and M is the number of ground-truth ellipses. Based on (18), $F-$measure is obtained as follows:

$$F-\text{measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{19}$$

B. Investigation of Algorithm Controls

Performance of the ellipse detection method depends on the choice of threshold parameters $T_n, T_O, T_\theta, T_c, T_l, T_d, T_\psi$, and T_s. It is likely that one set of values cannot suit all images. However, general rules of thumb can be derived for each parameter, fixing other threshold parameters, such that balanced performance can be achieved in terms of $F-$measure and the computation time (t), as illustrated in Figs. 11 and 12.

$F-$measure drastically reduces for T_n>15, while the execution time remains stable in Fig. 11(A). T_n is therefore set as 15. Fig. 11(B) shows the effect of the threshold T_O. It is clear that the detection effectiveness does not vary significantly but the computation time increases when the threshold $T_O > 20$. Thus, we choose 20 as the value of T_O.

As observed in Fig. 11(C), the best performance is obtained with values of T_θ between 30 and 40 for $F-$measure. The execution time t decreases as values of T_θ increase. As a result, we set $T_\theta = 35°$. As clearly illustrated in Fig. 11(D), the best thresholds are 25 and 30 for the detection effectiveness, but the execution time of a 30 threshold is more time demanding than that of a 25 threshold. We choose $T_c = 25$ as a good tradeoff to gain a good effectiveness and keep an execution time an acceptable value. For a ratio threshold $T_l > 0.6$, there is a significant decline in the detection effectiveness and the

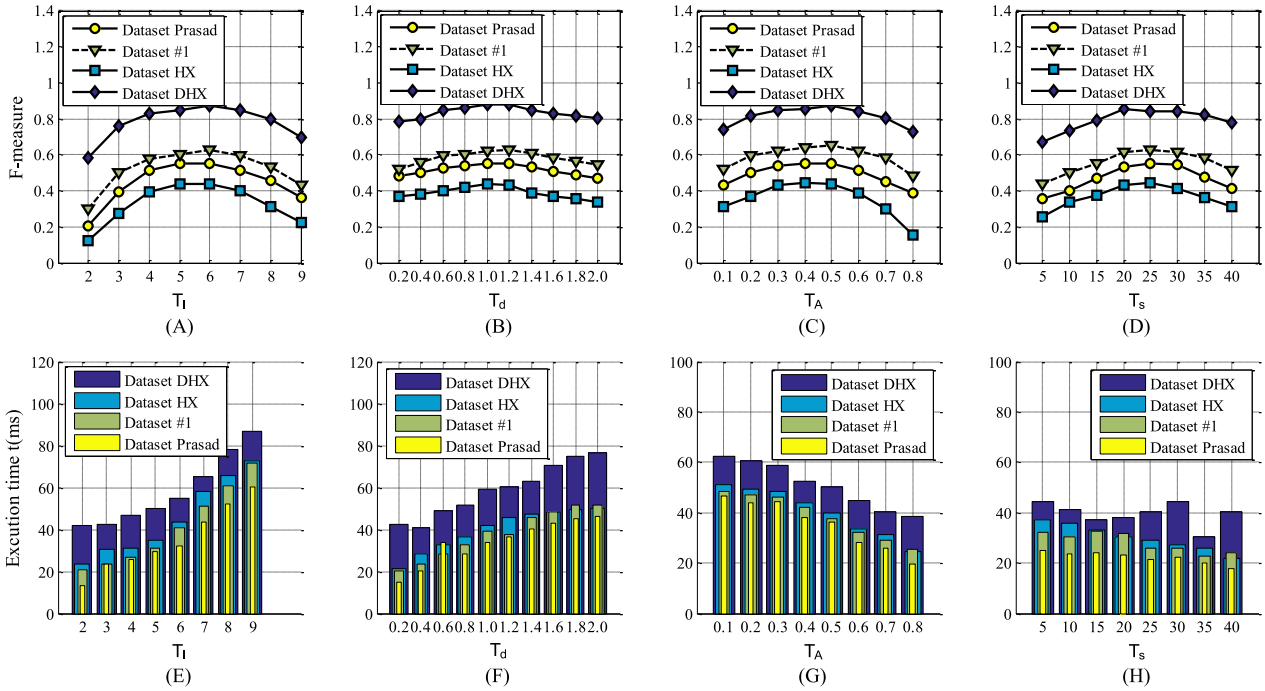

Fig. 12. Effectiveness values and execution times for four thresholds T_l, T_d, T_ψ, and T_s. (A) Threshold on distance ratios. (B) Threshold on pixel distances. (C) Threshold on numbers of points. (D) Threshold on similarities. (E) Threshold on distance ratios. (F) Threshold on pixel distances. (G) Threshold on numbers of points. (H) Threshold on similarities.

computational time also decreases due to the time decrease of the clustering step. We set $T_l = 0.5$ because this value can guarantee the best effectiveness and an acceptable execution time comparing with a value of $T_l = 0.4$, as shown in Fig. 12(A). Fig. 12(b) illustrates that the detection effectiveness is basically stable, while the time used for detecting ellipses slightly increases with the threshold T_d. In this case, the best tradeoff between the detection effectiveness and cost time is confirmed with $T_d = 1$. The detection effectiveness dramatically decreases when the threshold $T_\psi > 0.5$ because more arc sets are considered as invalid ones, while the fluctuation of the execution time is small in general, as shown in Fig. 12(C). We adopt $T_\psi = 0.5$. Setting $T_s > 25$ has negative impact on the detection effectiveness, as illustrated in Fig. 12(D). Finally, we adopt $T_s = 25$.

We note that the detector can be tuned for the best performance in a specific scenario and for a specific application by using a learning-based approach to determining the best values of thresholds. Here, our motivation is to show that empirically chosen parameters, not necessarily fine tuned for the best performance, can also provide a good detection performance. This indicates the versatility of our method.

C. Performance Comparison

The results of average $F-$measure values and average execution time are presented for four datasets in Table I. We compared the presented detector with six state-of-the-art model-based methods by means of the same evaluation metrics, used in four public datasets with different characteristics. Considering the overall effectiveness, indicated by *F*-measure concerning speed, precession, and recall factors together, our method outperforms other six methods in three out of four datasets, while ranked second for the first dataset.

The methods of Bai, Liu, and Mai present relatively lower performances, especially losing ellipses in images with complex backgrounds. Liu's method has good execution speed but very poor effectiveness. Bai's method has better effectiveness on large ellipses than Mai's method but still remains low. The method of Prasad *et al*. outperforms all other methods in its own dataset (Dataset Prasad). Our method is in the second place and close to Prasad's method in terms of the $F-$measure. Prasad's method is very time demanding though it shows relatively good effectiveness for detecting small and distorted ellipses. The results of execution time show that Jia's method is the fastest one and its effectiveness is better than Fornaciari's method. Fornaciari's and Jia's detectors show excellent execution speeds but suffer in detection effectiveness due to poor performance for small, occluded ellipses. In other three datasets, the proposed method is superior to other methods for the effectiveness, especially in detecting small ellipses and occluded and overlapping ellipses. In terms of average execution speeds, the execution time of our approach is less than 1% of the execution time of Prasad's method and falls behind methods of Fornaciari and Jia. However, the detection speed of our algorithm is still acceptable for real-time applications. Since the frequency of the video (see Supplementary Material) of Kinect is around 30 Hz, the proposed detector, as well as Jia's and Fornaciari's detectors, can perform real-time ellipse detections. With this as a sufficient speed, the proposed method's superior *F*-measure provides the obvious advantage. The detection examples from these datasets are provided in Figs. 13 and 14.

TABLE I
COMPARISON RESULTS FOR FOUR DATASETS

		F-measure	*Time(ms)*	*Precision*	*Recall*			*F-measure*	*Time(ms)*	*Precision*	*Recall*
Dataset Prasad	Mai 2008	0.2535	962.2	0.4135	0.2269	**Dataset HX**	Mai 2008	0.2573	597.8	0.3109	0.2431
	Liu2009	0.0950	1049	0.0700	0.1505		Liu2009	0.0534	1560	0.0653	0.0575
	Bai 2009	0.2395	3470	0.2195	0.2890		Bai 2009	0.1028	4140	0.0830	0.2181
	Prasad 2012	0.7428	8300	0.8512	0.6813		Prasad 2012	0.3910	3260	0.3364	0.4669
	Fornaciari 2014	0.3661	12.61	0.8541	0.2330		Fornaciari 2014	0.3472	19.35	0.8825	0.2161
	Jia 2017	0.4332	8.42	0.7390	0.3064		Jia 2017	0.4032	15.69	0.8165	0.2677
	Ours[1]	0.5172	14.51	0.6214	0.4429		Ours[1]	0.4128	22.16	0.7375	0.2866
	Ours	**0.5549**	**14.97**	**0.6437**	**0.4876**		Ours	**0.4467**	**22.52**	**0.7836**	**0.3124**
Dataset #1	Mai 2008	0.2604	1979.5	0.3299	0.2463	**Dataset DHX**	Mai 2008	0.2248	1217.04	0.175	0.3737
	Liu2009	0.1170	3960	0.1248	0.1415		Liu2009	0.0342	1272.04	0.035	0.0621
	Bai 2009	0.2121	136085	0.2420	0.1909		Bai 2009	0.0648	48962.13	0.0369	0.4256
	Prasad 2012	0.4512	17130	0.4425	0.4610		Prasad 2012	0.8519	2272.0	0.8823	0.8217
	Fornaciari 2014	0.5716	16.55	0.7117	0.4777		Fornaciari 2014	0.4093	40.37	0.3585	0.4769
	Jia 2017	0.5733	12.61	0.6161	0.5361		Jia 2017	0.3377	34.68	0.2475	0.5314
	Ours[1]	0.6133	20.35	0.7946	0.4994		Ours[1]	0.8475	41.86	0.9118	0.7917
	Ours	**0.6378**	**20.83**	**0.8312**	**0.5174**		Ours	**0.8906**	**43.30**	**0.9347**	**0.8505**

Note that Ours[1] represents the proposed method with the determination strategy of ellipse's center provided in [36]–[40].

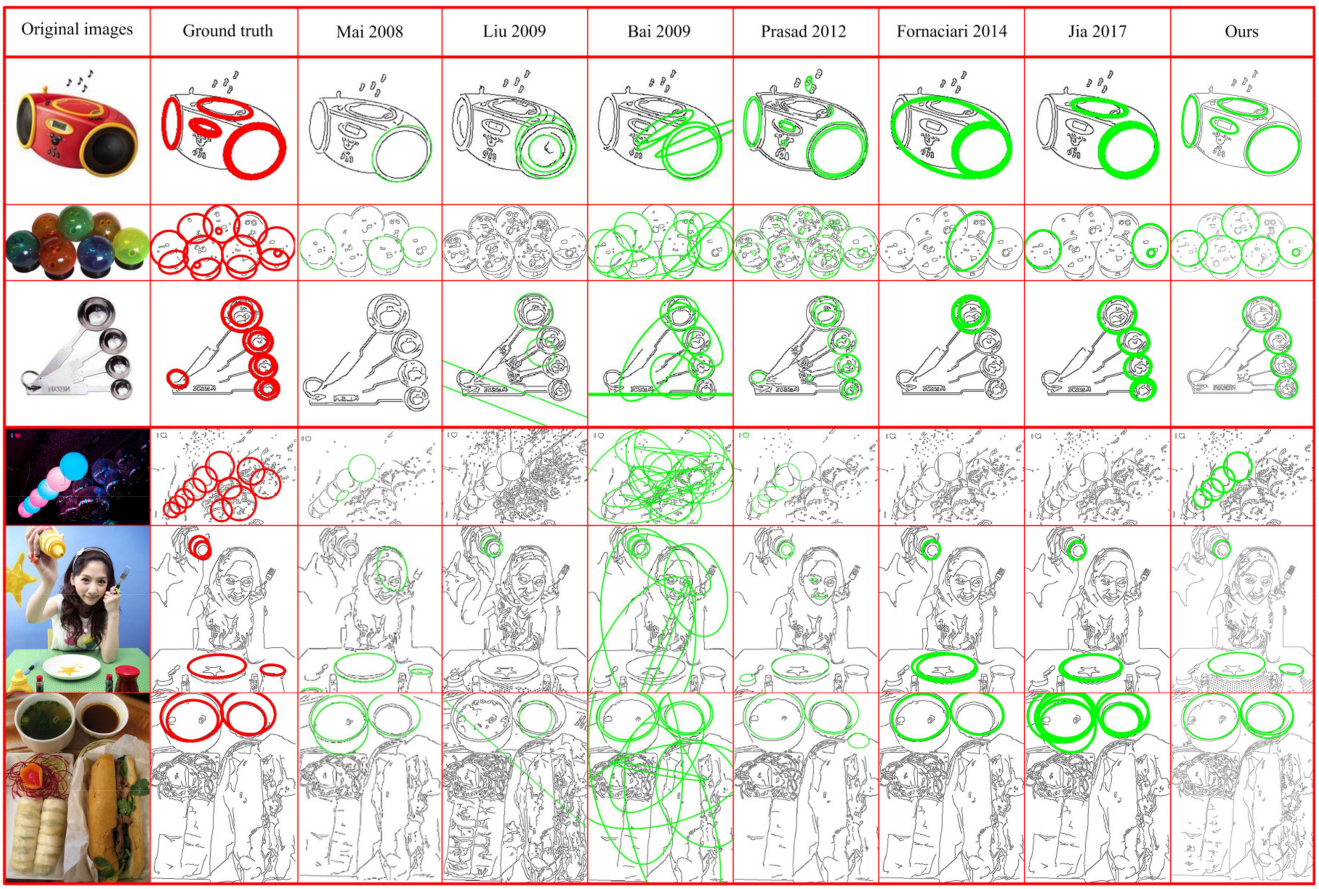

Fig. 13. Detection practical examples for Dataset Prasad (the first throw rows) Dataset #1(the last three rows).

In terms of the individual factors, for one of the datasets, the ellipse detector proposed by Prasad *et al*. demonstrates a better precision and recall performance than our method but at a significantly higher computation time. Due to the computational burden, Prasad's method is mostly suitable for postprocessing applications where real-time results are not required. The two other methods presented by Fornaciari *et al*. and Jia *et al*. indicate a slightly lower computation time than ours but cannot cope well with detecting small and occluded ellipses, resulting in a high rate of wrongly detected ellipses and a low recall rates in complex scenes.

The detection speed of our method is in an acceptable range for real-time applications. Its recall is generally better than Fornaciari's and Jia's recalls and comparable to that of Prasad. Its

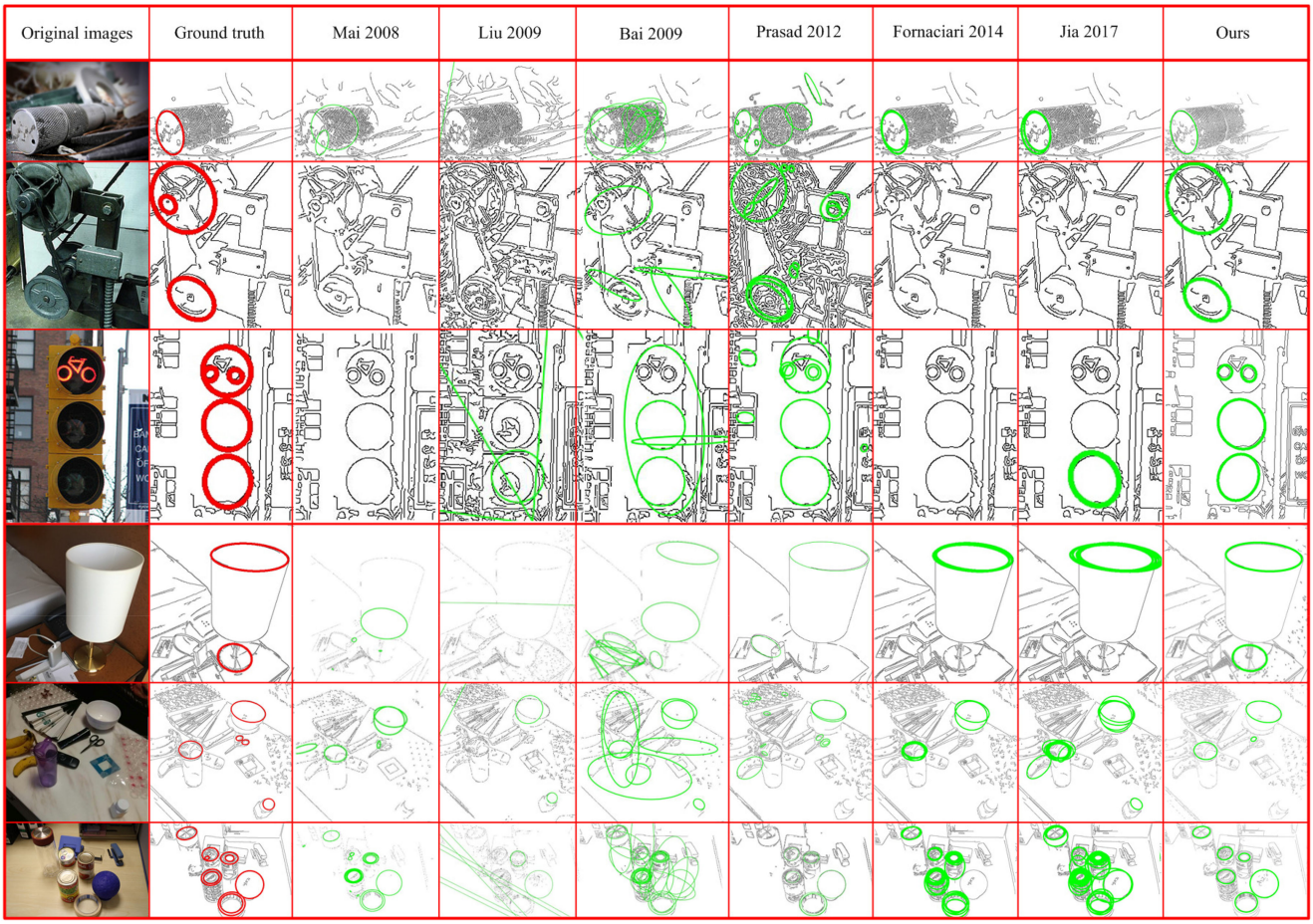

Fig. 14. Detection practical examples for Dataset HX (the first three rows) and Dataset DHX (the last three rows).

precision is comparable to Fornaciari's and Jia's ones. Thus, our method demonstrates efficient real-time performance for dynamic scenarios with unstructured scenes, in which there are multiple ellipses which vary in sizes and might be partially occluded.

Here, we discuss the reason of competitive advantages of the proposed method against other methods. A curve is split into arcs based on arc curvatures such that our method presents a good effectiveness in detecting small, occluded, and overlapping ellipses. Indeed, even though the arc extraction and arc grouping cost additional computations, they are effective in reducing the computations in the subsequent steps such as the arc grouping and arc clustering. Consequently, the detection process can still be real time.

The reasons regarding false detections are also mentioned. When ellipses are heavily obscured, it may result in false positives (overdetection) because of irrelevant combinations of arcs belonging to different ellipses. In an arc extraction procedure, it sometimes splits the elliptic curves into many small arcs which do not contribute significant information for ellipse detection due to the lack of curvature, which may generate false negatives (missing detection). We conclude that if the detection accuracy is of paramount importance, for example, in life critical scenarios, the proposed method is not the most suitable. But for most practical applications, where a good balance of detection effectiveness and speed is sought, the proposed methods provide a good solution

Here, we compare the proposed strategy of determining the center of ellipse with the existing works to verify the effect on the detection performance. In experiments, we adopt the proposed detectors with the determination of ellipse's ellipse based on the mean-shift algorithm and with that of confirming the ellipse's center provided by [36]–[40] for comparing the detection performances. As shown in Table I, our method with the proposed strategy of confirming ellipse's center costs more time than the other one. We add the tangent parameters formed by another pair of arcs and use the mean-shift algorithm to improve the accuracy of estimating the center, which consumes additional computational burden. However, the proposed method of estimating the center performs better than the existing works on the four public datasets.

D. *Robustness Analysis of Ellipse Detection in Real Time*

We perform several experiments to further evaluate the algorithm's real-time performance for robotic manipulation of cylindrical objects under actual static or dynamic conditions. The first experiment is to investigate the robustness of our ellipse detection method in a sequence of images captured from a static scene with a cluttered background. In the second

Fig. 15. Real-time ellipse detection in a sequence of 214 images. A cluttered background with light and dark colors is used to verify the robustness of the presented detector.

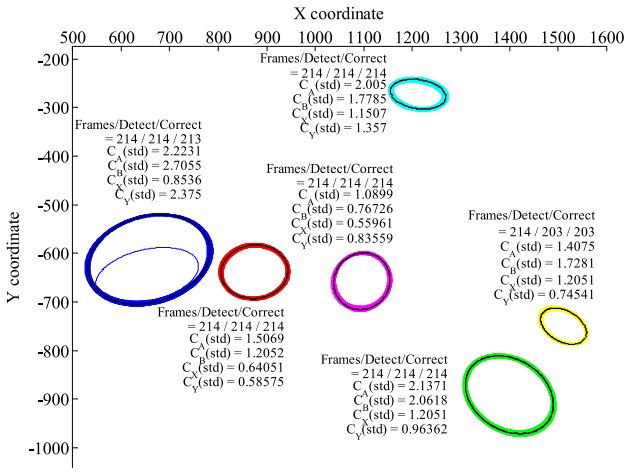

Fig. 16. Statistic results of detecting six ellipses in a video stream.

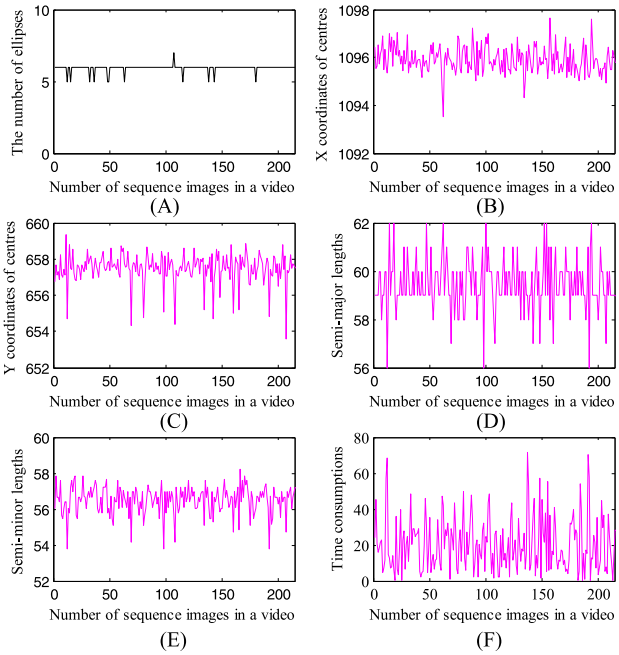

Fig. 17. Detailed detection performance in all image frames for one of the objects. (F) Execution time for each image.

experiment, the detector is evaluated in a dynamic scene in which the image sequence includes multiple moving objects with the cylindrical shape.

1) Robustness Analysis of Ellipse Detection in Real Time: We use a camera streaming a video of six cylindrical objects placed on a table, as illustrated in Fig. 15. The video contains 214 image frames and our ellipse detector is applied to all images for analyzing the robustness of the method over a continued duration. The statistical data of detection results for all six ellipses in images are shown in Fig. 16. As expected, the algorithm is fast and robust to the threshold parameters in detecting the ellipses with a fixed center and orientation and the detector can reach stable detection in a video stream successfully. The detection successes for the ellipses appearing from the left to the right in the horizontal axis for 214 input images are 213, 214, 214, 214, 214, and 203 times, correspondingly. The standard deviations of four parameters (the center coordinates, semimajor and semiminor lengths) for the six detected ellipses are in the ranges of (0.5596, 1.2051), (0.5857, 2.3750), (1.0899, 2.2231), and (0.7673, 2.7055). The standard deviations of orientations vary from 0.0160 to 0.0849 radian. Moreover, the ratios of the mean values of the estimated ellipses parameters (X_c, Y_c, A, B, O) to the ground-truth values for six ellipses are 0.9862, 0.9313, 0.9756, 0.9139, and 0.9723, respectively.

As shown in Fig. 17(A), the number of detected ellipses keeps a stable level over time. Fig. 17(B)–(E) demonstrates four parameters of one of the detected ellipses (shown in pink color in Fig. 16) as a reference case for analyzing the detection robustness. Basically, the coordinates (X_c, Y_c) of ellipse center and semiaxis lengths (A, B) have fluctuations within small ranges. Specifically, we use the ranges of parameters to investigate how geometric configurations (e.g., the center coordinates and semiaxis lengths) affect the performance of our detector. One semiaxis varies from 56 to 62, and the other changes within the range from 53.8 to 58.2. The center is fixed as (1095.782, 667.643). The fluctuation ranges of its horizontal coordinate are from 1093.5 to 1097.6. The vertical coordinate varies between 653.6 and 659.4.

2) Ellipse Detection in Dynamic Scenarios: To evaluate the performance of the presented detector in dynamic scenarios, we construct a set of eight experiments of moving cylindrical objects at different speeds, and compare our results with the two real-time detectors, namely Fornaciari and Jia. In the experiments presented here, we use five objects: four cans with different sizes, and a cup. The objects are randomly placed on a board which is carried with the help of the experiment platform including a robotic arm-UR10 and a 2-finger Robotiq gripper introduced in the next section. We use a simple motion model where the manipulator base joint is rotated around 55° at different speeds for illustrating the performance. A Microsoft Kinect camera mounted on a tripod next to the setup is used to capture the images. While executing the robot motion, the image sequence acquired by the camera is processed in real time using the proposed algorithm. The parameters of all detected ellipses

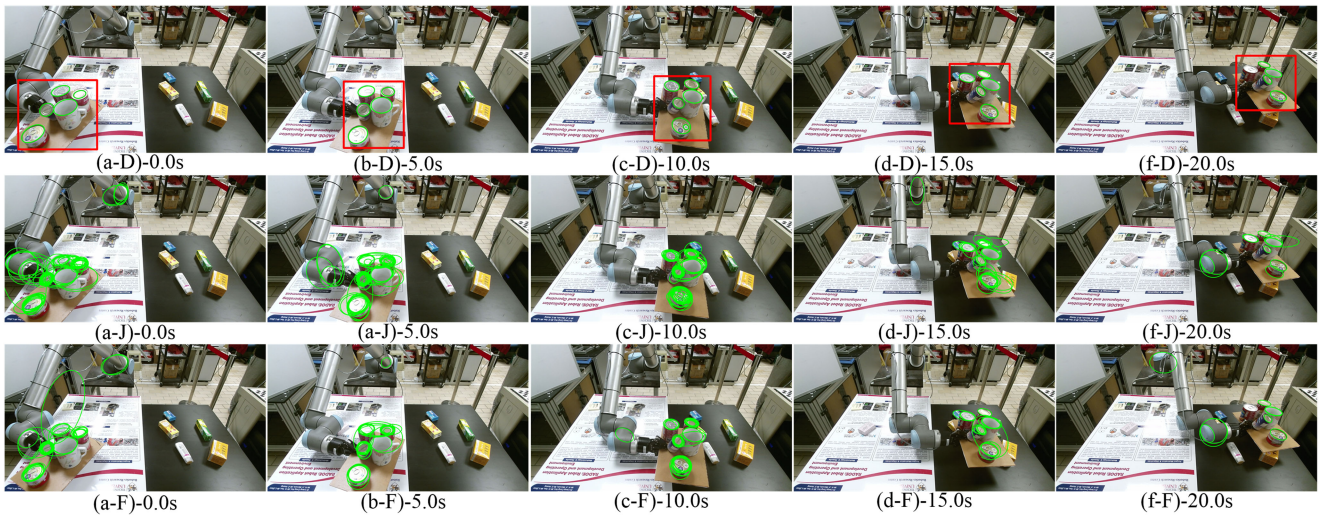

Fig. 18. Snapshots of dynamic detections at 40% of full speed of robotic motion for comparing the presented detector (the first row) with Jia's detector (the second row) and Fornaciari's detector (the third row) at 0.2-Hz sampling frequency.

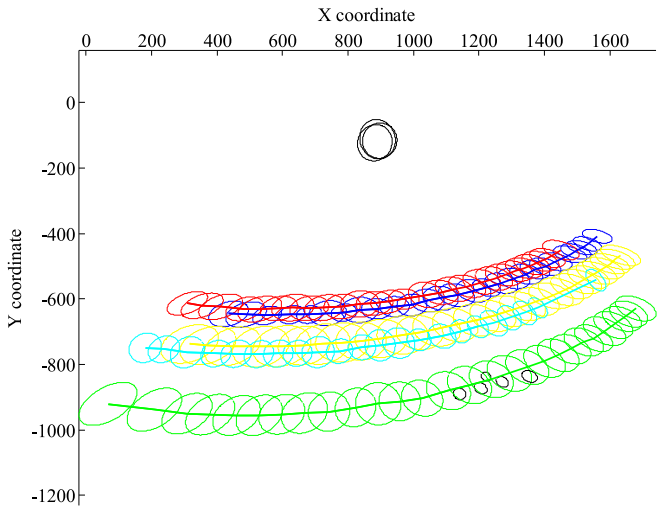

Fig. 19. Sequence of detected ellipses in a dynamic scenario with multiple moving cylindrical objects.

in each image frame are recorded independently at every time step. Its center coordinates describe the position of ellipses in the frames.

The snapshots of the robot status and our detection results are shown in the top of Fig. 18 at a sampling rate of 0.2 Hz. All detected ellipses in all image frames are shown in Fig. 19 to demonstrate the trajectories of the objects. The results of the methods of Jia and Fornaciari are depicted in the second and third rows of Fig. 18, respectively. These detectors cannot discard false ellipses as they always merge close ellipses, which results in excessive fault detections. Regarding quantitative evaluation, *F*-measure, precision, and recall of our method are 0.7873, 0.9571, and 0.6814, respectively. However, the methods of Fornaciari and Jia have worse performances with *F*-measure (0.2934, 0.3130), precision (0.3412, 0.3845), and recall (0.2573, 0.2639).

TABLE II
COMPARISON RESULTS FOR VARIOUS SPEEDS

	F-measure	*Precision*	*Recall*
1.0 of full speed	0.7885	0.9192	0.7110
0.9 of full speed	0.7707	0.9388	0.6715
0.8 of full speed	0.7797	0.9153	0.6930
0.7 of full speed	0.7506	0.9265	0.6520
0.6 of full speed	0.7873	0.9571	0.6814
0.5 of full speed	0.7842	0.9599	0.6753
0.4 of full speed	0.7501	0.8425	0.6760
0.3 of full speed	0.8154	0.9474	0.7277

To explore the effect of different moving speeds on the detection performance, eight experiments with various angular velocities of base motion between [0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0] of the full speed (180°/s) of the robotic arm are performed. As seen in Table II for experimental results of eight trials, we achieve stable values of *F*-measure, precision, and recall. The presented detector can keep a high detection effectiveness, yielding more than 75.01% *F*-measure for the different moving speeds. According to the given experimental results, the motion speed of objects does not have a significant effect on the performance of the ellipse detector.

Failures in these experiments come from two sources. The first source is the cluttered background, due to which the presented ellipse detector cannot sometimes distinguish "real" elliptic edges and "false" edges from the background. These situations occur rarely—we observed 4 out of 19 failures. Another source is that when the object is far away from the camera, the ellipses become smaller gradually until the detector cannot succeed with such small ellipses.

IV. SYSTEM AND PLATFORM OF ROBOTIC MANIPULATIONS

The proposed ellipse detector could be applied to several robotics applications enabling the robot to mainly depend on 2-D information for rapid detection and tracking of cylindrical

objects. For instance, the ellipse detector could be deployed to track cylindrical objects carried by a service robot, or to track products on a conveyor belts for picking and placing purposes, as well as to detect cylindrical mechanical parts from a cluttered set of different shape objects for sorting them or track circular shafts and circular holes simultaneously for assembling purposes. Here, we present a sample integration of the algorithm in an actual robotic manipulation system.

The robotic manipulation platform is equipped with a 6-DOF industrial robot arm—Universal Robot 10 (for tracking cylindrical objects above and grasping static cylindrical objects) or Universal Robot 5 (for grasping dynamic cylindrical objects) and a 2-finger Robotiq 85 gripper mounted on robotic end-effector. The robot arm-UR5 and UR10 have the same repeatability accuracy of ± 0.1 mm, have a reach of 95 and 140 cm with the gripper, and can support payloads of 5 and 10 kg, respectively. Without tactile feedback, the 2-Finger Robotiq 85 gripper has two articulated underactuated fingers that each have two joints (two links per finger) driven by a single actuator. They automatically adapt to the shape of the object. The Kinect sensor that is capable of providing high-quality registered videos of both color and depth is fixed on a Pan-and-Tilt approximately 1.5 m above the ground near a table. The calibration accuracy between Kinect sensor and robot arm base is up to an average of 3 mm. A linear guide with an operation range between 0 and 1 m is used to carry cylindrical objects. Without an encoder, the stepper motor rotates a round and the slide on the linear guide moves 75 mm. The system itself has deviations from the calibration between the camera and the robotic base, as well as the registration of the RGB image and the depth image together for an RGB-D camera. The registration deviation has the less effect than the calibration deviation. The offset generated by two kinds of deviations is a constant value. Thus, we can make up for these two deviations by performing several robotic grasps to obtain this offset. Moreover, we also can accept these two deviations if the gripper grasps a relatively small object with the size less than the maximum expanding size of the gripper. The robot uses the average value of the information of detecting an ellipse to perform a grasp, which improves the accuracy of robotic grasp.

The problem of motion planning [51] is addressed using the RRT algorithm in Open Motion Planning Library [52] and Robot Operating System [53] to produce continuous motions that connects the configuration of end-effector between its stationary position and a goal configuration for grasping while avoiding collision with obstacles. The motion is represented as a path in configuration space. The point clouds acquired by the Kinect camera are also used for collision avoidances as well, so as to ensure collision-free motion planning and safe manipulation.

V. Robotic Manipulation Experiments

We present two robotic manipulation experiments in this section. The first is manipulation of static cylindrical objects in a human–robot collaboration task. The second involves two robotic manipulation tasks in different scenarios with dynamic cylindrical objects.

A. Static Cylindrical Objects

Here, we performed a picking and stacking experiment with cylindrical objects, e.g., food cans. Additional information is needed to capture the 3-D coordinate of the detected object to be manipulated. Here, a Microsoft Kinect sensor is deployed for obtaining two types of information of RGB and depth. The ellipse detector utilizes the RGB image to perform ellipse detection and the centers of the detected ellipses are provided as 2-D pixel coordinates. The 2-D pixel coordinates of the centers are converted to 3-D world coordinates so that the robotic manipulator can move to the location and grasp the cylindrical object. The depth image, which is an encoded grayscale image with each value measuring the distance to the Kinect camera, is used to obtain the 3-D world coordinates of the detected centroid. We collect 20 times for each cylindrical object to calculate the average value of the centers of 20 ellipses to avoid the issue caused by the case of the partial missing 3-D data in the Kinect sensor. The 3-D coordinates of the centers are then acquired from the point cloud by mapping the 2-D pixels coordinates to the registered depth in the point cloud. As a grasped cylindrical object stands along the vertical direction, the orientation information is known to a robot. Obtaining the 3-D coordinate of ellipse center by the scheme above, a robot can make a configuration to realize a grasp.

While various grasp types are needed for different objects, most of the cylindrical objects are grasped with a lateral cylindrical wrap using a two-finger gripper if there are no collision issues. Otherwise, a top grasp could be used to pick up an object placed in a cluttered environment.

We consider a human–robot collaboration task, in which the human operator feeds some objects frequently to a working station that is a table. The robot's task is to detect the arrival of new cylindrical objects on the right side of the table and subsequently stack them on the left half-side of the table. Thus, the robot requires fast detection of multiple cylindrical objects on the table. It should also be aware of changes in the scene to identify the new objects fed by the operator, to update the position of the latest objects stacked on the left side and to track the status of an object once it moves from the right side to the left. We perform this task by deploying the proposed detector. Thus, the cylindrical objects are identified and tracked by detecting the top of the objects as ellipses. Once a new ellipse is detected in the right side of the table, see Fig. 20(A-1)–(A-3), the 3-D position of the top of the object is located as described above. Then, the motion planning program infers the picking and placing paths. Then, the inverse kinematic and robotic arm gets automatically triggered to pick up the object and stack it on the other side of the table. Once the ellipse moves to the left side of the table, the manipulated object is tracked using the real-time detection of ellipses, see Fig. 20(B-1)–(B-3), and its state from a new detected object (ellipses are shown in green) is changed to a stacked object (ellipses are shown in red). Finally, once the object is placed on the back plate on the table or stacked on the top of another object, as shown in Fig. 20(C-1)–(C-3), the targeted position for stacking the next object is updated by getting the 3-D position of the latest ellipse on the left side.

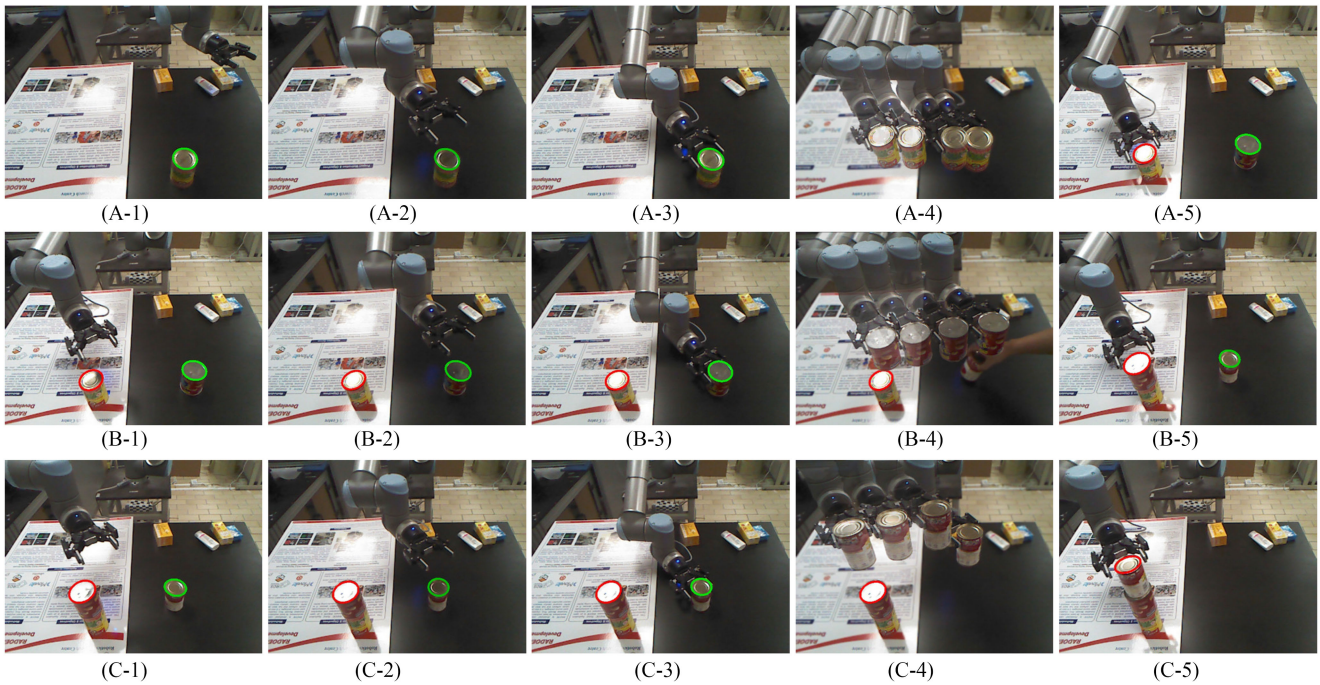

Fig. 20. Stacking food cans. Place the first can (A—1, 2, 3), stack the second can on the first can (B—1, 2, 3), and stack the third can on the second can (C—1, 2, 3).

The results of ellipse detection and the state of robot arm at the picking instances, shown in Fig. 20, indicate successful positioning of objects with cylindrical shapes for robot manipulation purposes, such as picking, stacking, and placing. The images indicate the detected objects using RGB images captured by the Kinect camera, which shows the states of the robot arm-UR10 and gripper at manipulating instances. We performed five groups of stacking experiments. The successful rate of stacking cylindrical objects reaches 100% as it is easy for our detector and the proposed grasping strategy to realize ellipse-based stacking actions in static environments.

B. *Dynamic Cylindrical Objects*

We carried out two experiments with moving cylindrical objects to validate the performance of our ellipse detector for robotic manipulation in dynamic scenarios. The first experiment demonstrates the success rates of grasping cans placed on a conveyor moving at different speeds. The second experiment imitates an industrial line to pick up the cans from a linear conveyor and put them in a box.

Our robotic grasp strategy consists of three main steps: perception, inference, and manipulation. In dynamic grasp experiments, we first use the proposed detector to detect the ellipse. Then, given the position where the object is placed by the depth information, the robot infers the next position of the moving object on the linear slide and robotic arm configuration for grasp. Accordingly, the robot moves to the pregrasp position to wait for the object to be manipulated. Finally, the robot adjusts the direction of the gripper to grasp the object, carry it to the destination, and release it.

In particular, the process is as follows. After manual placing of a can on the slider, the ellipse detector captures and tracks the position of the moving can. Then, the grasp plan is determined by the estimating the direction and 2-D location of the object's center. The manipulation task is defined as grasping the moving object, moving it 20 cm above the linear guide, and carrying it until the gripper reaches the destination. Before robotic manipulation, the arm stops at the home position for about 1 s to detect a moving object and determine the 7-DOF gripper configuration. The robotic system plans a suitable path to a "pregrasp" position, by using inverse kinematics, which is defined as 10 cm away from the actual grasping configuration along the approach vector (the normal to the palm of the gripper). After reaching the "pregrasp" position, the gripper waits for the object to arrive in front of the gripper. When that happens, the robot moves 2 cm ahead of the newest object position, to perform the grasp action. If it is outside the operating range of the conveyor, the robot waits for the new pregrasp position from the returning phase. This step is needed to avoid the gripper approaching the object early or late and prevent pushing the object off. Finally, the robot moves the grasped object to the destination, and the gripper opens.

In this experiment, we did not use the point cloud to deduce the spatial coordinate of the cylindrical object. The bottlenecks are the relatively low update rate of point clouds and inaccurate depth data due to the noise and week registration characteristic using the Kinect sensor. Initially, we tried to obtain a group of depth information, calculate an average value, and map to the z-coordinate, but the accuracy was poor. Our alternative method, to avoid the use of the point cloud in dynamic scenarios, is to employ the 2-D data in RGB images and two reference positions

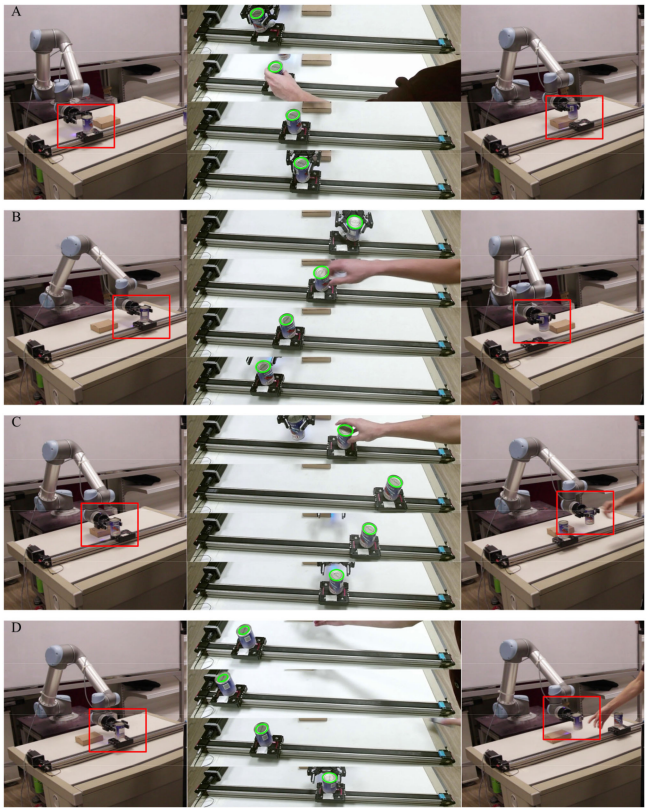

Fig. 21. Robot grasping cans on the conveyor with four speeds in a dynamic scenario. The target is enclosed by a red frame. The speeds of A, B, C, and D are 2.5, 5.0, 7.5, 10.0 cm/s, respectively. Several detection examples for each speed are shown in the middle column.

to determine the position of the object on the linear guide. In this way, the 3-D position of the center of the object and the corresponding pixel values of the image at the two endpoints of the conveyer are recorded once as reference information. Then, the slope of the object's path from the conveyer belt relative to the robotic base is obtained. With this information, the pose of moving objects along the conveyor can be extrapolated using only pixel values of the center of moving ellipses obtained at a rate of video capturing.

1) Grasps of Moving Objects With Different Speeds: In the first set of grasp experiments, our goal is to grasp a can that is placed on a moving linear guide with four different speeds. Fig. 21 shows the performance at each step. We perform four sets of experiments with different speeds, at 0.025, 0.050, 0.075, and 0.100 m/s, 30 times for each speed. The main factor affecting the time between picks is the operation range of the conveyor. Specifically, when the planned pregrasp position is beyond the operation range of the conveyor, the robot has to wait for detecting ellipse and sampling the speed of the can in the returning phase.

The success rate of grasping objects over 30 runs at each speed is also summarized in Table III. The robot successfully grasped 118 times out of 120 tries, realizing an overall success rate of 98.33%. Specifically, three out of four experimental sets were accomplished without a single failure. The high rate of successful grasps indicates the robustness of the proposed

TABLE III
EXPERIMENTAL RESULTS OF ROBOTIC GRASPS IN DYNAMIC SCENARIOS

Speed (cm/s)	Trials	Successes (single-way)	Successes (double-way)	Failures	Success rate	Time(s)
2.5	30	30	0	0	100%	0.89
5.0	30	29	1	0	100%	0.51
7.5	30	30	0	0	100%	0.47
10.0	30	25	3	2	93.3%	0.35
Results	120/t	114t	4/t	2	98.33%/a	0.47/a

/*t* and /*a* mean the total value and the average value, respectively.

detector in dynamic scenarios. Table III also shows two cases in which the grasps failed. For these two failure cases, the object was touched or knocked down by one finger when the gripper approached it. The fast movement of the can results in the imprecise scheduling of grasp execution for the fourth set of experiments. In practical grasp experiments, our robotic gripper does not have any force or tactile feedback so that even a slight error in perception and path planning could result in a grasp failure. We believe that methods based on force or tactile feedback would complement the grasping system and help to build a more robust grasping system. However, these results provide strong evidence that the proposed elliptic shape detector can be applied in real scenarios.

2) Dynamic Picking and Placing Cans in an Industrial Line: In the second experiment, the task is to pick up the cans from the linear guide and place them in a box. Fig. 22 shows screenshots of picking and placing the objects. Our system first infers the position of the moving object by ellipse detection and then plans a path to manipulate it on a table. The work scene is set with ten cans placed densely on two borders of the table and two persons put cans on the slider randomly. This is a "close-loop" automatic system. Once detecting the position of the object by the proposed detector, the robot will manipulate it to the placed destination—the box (near to the table) until the end. The robot attempts to grasp an object only once. The algorithm stream for detecting cylindrical objects continues to seek the next object so that the robot can automatically initialize the next grasp action.

The grasps are considered successful if the robot can grasp and move the object to the destination. We declare failure if the robot does not achieve a grasp or drops the object during or after lifting. Since the maximum opening of the gripper is slightly bigger than that of the can, small errors in position estimation have a significant effect on a grasp outcome. This is not surprising since the rounded and curved shape made it prone to rolling out of the gripper as it closed. The speed of the linear guide is fixed to 5.0 cm/s. In each trial, the robot picks up the can from the slider and places it in the designated area. Similarly, having an estimated position by the ellipse detector, the robot moves to the pregrasp position and wait for the object coming to the front of the gripper. The planning and execution time for the robot must match the time from obtaining the position of the ellipse by the detector to the position where the object arrives in the front of the gripper. After getting the object, the robot places the can in a box near the table. We ran 30 picking and placing experiments using ten cans. For the 28 rounds of experiments, we obtained a 93.3% grasp success rate (2 grasp failures out of

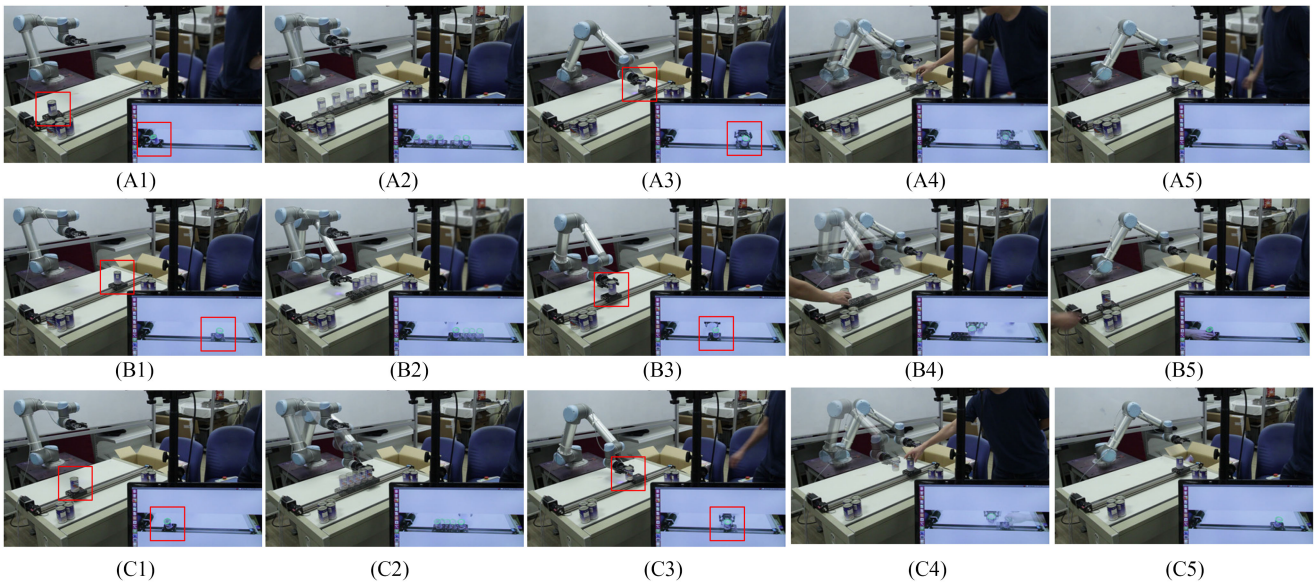

Fig. 22. Steps of robot picking and placing cans on (A), (C) outward phase and (B) returning phase. The target is enclosed by a red frame.

30 grasp attempts). Of the two failures, they were due to the estimation of inaccurate positions, resulting in collisions of the fingers with the object during the grasp attempt.

VI. Conclusion and Future Work

In this paper, we consider the problem of detecting ellipses with sufficient accuracy and speed to permit robotic manipulation of cylindrical objects in manipulating a cylinder object in real time in complicated scenes. Segmenting arcs based on a changing amount of arc curvatures and grouping arcs depending on the arc convexity–concavity and mutual positions result not only in improving the detection effectiveness (F − measure) but also reduces the detection time considerably.

Compared with six other recent methods of detecting ellipses, our extensive experiments demonstrate that our algorithm presents significant advantages for detecting ellipses on images with complex backgrounds in terms of tradeoff between the detection effectiveness and detection time. In our robotic experiments, the robot successfully tracks a cylindrical object to operate in real time, recognizes a cylinder object, blocks and also sorts cylinder objects from ones with different shapes in a cluttered environment.

In the future, we will update this algorithm to detect even smaller ellipses, heavily occluded ellipses, and well-shaped semiellipses. We will also consider segmenting cylindrical objects by the presented ellipse detector and edge information based on deep learning to get 3-D coordinates, which allows a robot to sort multiple circle and elliptic mechanical components in industrial environments.

The proposed ellipse detection algorithm is relevant for visual tracking methods for robotics and we wish to explore this aspect also in the future work. Visual tracking methods that use shape-based models directly benefit from elliptic shape detection. Further, visual tracking methods that employ key point detectors can also benefit from our algorithm. Our algorithm identifies geometric relationships among multiple arcs and thus indicates key points representing elliptic features.

Moreover, the proposed method cannot deduce the object orientation. Future works will focus on further obtaining the pose (positions and orientations) of cylindrical objects, as well as incorporating other sensing modalities and complementary perception methods to leverage the robotic manipulation in more complicated scenarios. For example, we will consider segmenting cylindrical objects by the presented ellipse detector while capturing edge information based on deep learning to get both positions and orientations, which allows planning more advanced manipulation strategies for cylindrical objects with various configurations in complex and dynamic industrial environments.

Additional video is available on https://www.youtube.com/watch?v=UvszHZs8R1k.

Appendix
Review of Ellipse Detection Approaches

Here, we present a brief overview of the existing approaches of ellipse detection in images. Almost all ellipse detection methods work with edge maps of the images since they are amenable to parametric analysis, geometric analysis, as well as algebraic analysis of the properties of ellipses. Modified versions of the basic Hough transform (HT) for detecting shapes from planar sets of points were proposed for ellipse detection. The versions include standard HT [54], randomized HT [55], and probabilistic HT [56]. It has also been used in combination with other geometric or algebraic approaches, where HT is used for determination of s subset of parameters or is used with certain subset of the original image determined suitable for ellipse detection through geometric or algebraic properties. Examples include Cakir *et al.* [57], Chen *et al.* [58], and Prasad *et al.* [36]. The algebraic and statistical approaches such as those utilizing least squares fitting approaches [35], random sample consensus [48],

etc., benefit by supplying edges selected specifically as likely belonging to ellipses. Notably, they are significantly more computationally efficient than HT-based approaches.

Thus, selection of arcs based on geometric properties is useful in such approaches too. Further, the robustness and accuracy of ellipse detection are better if the ellipse is represented with as large circumference of it as available. However, the available circumference of ellipse may be broken into small curves due to occlusion, overlap with other objects, and image noise. Thus, while identifying arcs belonging to an ellipse, it is also worthwhile to cluster such arcs and use collectively for obtaining the ellipse parameters. This has been the fundamental guiding principle behind the ellipse detection approaches developed over the last decade when successive improvement in the accuracy of ellipse detection was observed. Unsurprisingly, these methods comprise the current state-of-the-art algorithms in ellipse detection from images. We summarize these methods briefly here. A geometric criterion is used to select candidate pairs of arc segments that belong to the same ellipse by searching the elliptic arcs from edge contours in [59] and [60]. Subsequently, these short arc segments are merged to fit an ellipse by a statistical regression method [61]. Similarly, in several ellipse detection methods [35], [36], [48], [62], [63], short straight lines are detected to approximate arc segments and these arcs segments are accumulated and merged into ellipses. Subsequently, these arcs are grouped according to the convexity–concavity [30], [64], [65], and geometric constraints [50]. Bai *et al*. [46] and Jia *et al*. [41] include the property of elliptical concavity to group arcs when clustering candidate ellipses. Symmetry of arcs in an image is utilized in [66] and [67]. The distance information is adopted to evaluate registration of a set of arcs [68], [69]. Finally, a number of methods proposed by authors in [34], [48], [70], and [71] have attempted to improve the detection accuracy with iterative approaches.

The computational cost of the methods above is typically smaller than that of HT-based methods, and the accuracy of these methods is better than the methods that use only HT or algebraic methods. Nevertheless, they are still limiting for real-time robust ellipse detections as we describe next. The time taken by these methods is determined by the time spent on analysis of the geometric properties, identifying the clusters of arc, determining the parameters of ellipse for each cluster of arcs, and then filtering the detected ellipses based on saliency metrics (confidence of detection) or redundancy (similar ellipses detected by two different clusters). Unless computationally efficient techniques are used for each step and information is reused across the steps, the overall computation time can be prohibitive for real application. This is even more prominent if the attempt is to detect ellipses with not only good precision but also good recall. Good recall requires considering as many clusters or as refined clusters as feasible and good precision requires as stringent clustering and filtering as possible, both imposing on the time spent on clustering arcs and filtering ellipses. Thus, methods that are fast compromise the detection performance measured through *F*-measure (harmonic mean of precision and recall) and methods that perform well are slow.

The Arc Extraction Algorithm

Algorithm I: Arc Extraction by splitting a curve at sudden changes.

Input: T_θ, curves
Output: arcs that may consist of ellipses in a container C
For $i = 0$ to the size of curves **do**
  Fit curve i by line segments;
  Calculate the angles θ formed by consecutive line segments
  For $j = 0$ to the size of line segments **do**
    If (abs($\boldsymbol{\theta}_{j+1} - \boldsymbol{\theta}_j$) $> T_\theta$)
      For $m = 0$ to the size of the ith curve **do**
        If (sign($\boldsymbol{\theta}_j$) $\cdot$ sign($\boldsymbol{\theta}_{j+1}$) < 0)
          If (line segments $[j]$.x$==$curve_$i[m]$.x &&
            line segments $[j]$.y$==$curve_$i[m]$.y)
              Reserve these points in the container B;
        Else
          If (line segments $[j+1]$.x$==$curve_$i[m]$.x
          &&line segments $[j+1]$.y$==$curve_$i[m]$.y)
              Reserve these points in the container B;
  For $k = 0$ to the size of B **do**
    For $n = B[k]$ to $B[k+1]$ **do**
      Reserve curve $[i][n]$ in the container C.

REFERENCES

[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proc. 2000 IEEE Int. Conf. Robot. Autom.*, vol. 1, 2000, pp. 348–353.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—A survey," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.

[3] C. F. Olson and D. P. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *IEEE Trans. Image Process.*, vol. 6, no. 1, pp. 103–113, Jan. 1997.

[4] A. Agrawal, Y. Sun, J. Barnwell, and R. Raskar, "Vision-guided robot system for picking objects by casting shadows," *Int. J. Robot. Res.*, vol. 29, nos. 2/3, pp. 155–173, 2010.

[5] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, A. Agrawal, and H. Okuda, "Pose estimation in heavy clutter using a multi-flash camera," in *Proc. 2010 IEEE Int. Conf. Robot. Autom.*, 2010, pp. 2028–2035.

[6] L. Bo, X. Ren, and D. Fox, "Learning hierarchical sparse features for RGB-(D) object recognition," *Int. J. Robot. Res.*, vol. 33, no. 4, pp. 581–599, 2014.

[7] D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, "Grasping novel objects with depth segmentation," in *Proc. 2010 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 2578–2585.

[8] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Proc. 2016 IEEE Int. Conf. Robot. Autom.*, 2016, pp. 3406–3413.

[9] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, nos. 4/5, pp. 705–724, 2015.

[10] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: Learning using a new rectangle representation," in *Proc. 2011 IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3304–3311.

[11] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Proc. 2015 IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1316–1322.

[12] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, nos. 4/5, pp. 421–436, 2018.

[13] M. Schwarz, A. Milan, A. S. Periyasamy, and S. Behnke, "RGB-D object detection and semantic segmentation for autonomous manipulation in clutter," *Int. J. Robot. Res.*, vol. 37, nos. 4/5, pp. 437–451, 2018.

[14] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, 2003, pp. 1824–1829.

[15] C. Korpela, M. Orsag, and P. Oh, "Towards valve turning using a dual-arm aerial manipulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 3411–3416.

[16] K. Ono *et al.*, "Detection, localization and picking up of coil springs from a pile," in *Proc. 2014 IEEE Int. Conf. Robot. Autom.*, 2014, pp. 3477–3482.

[17] K. Liu, H. Li, and Z. Sun, "Ellipse detection-based bin-picking visual servoing system," in *Engineering Creative Design in Robotics and Mechatronics*. Hershey, PA, USA: IGI Global, 2013, pp. 114–121.

[18] J. Su, Z.-Y. Liu, H. Qiao, and C. Liu, "Pose-estimation and reorientation of pistons for robotic bin-picking," *Ind. Robot*, vol. 43, no. 1, pp. 22–32, 2016.

[19] C. Do, T. Schubert, and W. Burgard, "A probabilistic approach to liquid level detection in cups using an RGB-D camera," in *Proc. 2016 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2075–2080.

[20] C. J. Phillips, M. Lecce, and K. Daniilidis, "Seeing glassware: From edge detection to pose estimation and shape recovery," in *Proc. Robot. Sci. Syst.*, vol. 3, 2016.

[21] J. A. Piepmeier and H. Lipkin, "Uncalibrated eye-in-hand visual servoing," *Int. J. Robot. Res.*, vol. 22, nos. 10/11, pp. 805–819, 2003.

[22] J. Liang, Y. Wang, and X. Zeng, "Robust ellipse fitting via half-quadratic and semidefinite relaxation optimization," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4276–4286, Nov. 2015.

[23] S. Zafari, T. Eerola, J. Sampo, H. Kälviäinen, and H. Haario, "Segmentation of overlapping elliptical objects in silhouette images," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5942–5952, Dec. 2015.

[24] J. Liu *et al.*, "Automated robotic measurement of 3D cell morphologies," *IEEE Robot. Autom. Lett.*, 2016.

[25] A. Soetedjo and K. Yamada, "Fast and robust traffic sign detection," in *Proc. 2005 IEEE Int. Conf. Syst., Man, Cybern.*, vol. 2, 2005, pp. 1341–1346.

[26] 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.

[27] J. Faigl, T. Krajník, J. Chudoba, L. Přeučil, and M. Saska, "Low-cost embedded system for relative localization in robotic swarms," in *Proc. 2013 IEEE Int. Conf. Robot. Autom.*, 2013, pp. 993–998.

[28] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 462–471, Apr. 2016.

[29] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1049–1065, Oct. 2014.

[30] 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, 2014.

[31] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[32] D. K. Prasad, M. K. H. Leung, C. Quek, and M. S. Brown, "DEB: Definite error bounded tangent estimator for digital curves," *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4297–4310, Oct. 2014.

[33] H. Freeman and R. Shapira, "Determining the minimum-area encasing rectangle for an arbitrary closed curve," *Commun. ACM*, vol. 18, no. 7, pp. 409–413, 1975.

[34] 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.

[35] D. K. Prasad, M. K. Leung, and C. Quek, "ElliFit: An unconstrained, non-iterative, least squares based geometric ellipse fitting method," *Pattern Recognit.*, vol. 46, no. 5, pp. 1449–1465, 2013.

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

[37] D. K. Prasad and M. K. Leung, "An ellipse detection method for real images," in *Proc. 25th IEEE Int. Conf. Image Vis. Comput. New Zealand*, 2010, pp. 1–8.

[38] K. C. Chen, N. Bouguila, and D. Ziou, "Quantization-free parameter space reduction in ellipse detection," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7622–7632, 2011.

[39] M. Cicconet, K. Gunsalus, D. Geiger, and M. Werman, "Ellipses from triangles," in *Proc. IEEE Int. Conf. Image Process.*, 2014, pp. 3626–3630.

[40] S.-C. Zhang and Z.-Q. Liu, "A robust, real-time ellipse detector," *Pattern Recognit.*, vol. 38, no. 2, pp. 273–287, 2005.

[41] 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.

[42] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, Aug. 1995.

[43] A. S. Aguado, M. E. Montiel, and M. S. Nixon, "On using directional information for parameter space decomposition in ellipse detection," *Pattern Recognit.*, vol. 29, no. 3, pp. 369–381, 1996.

[44] A. Fernandes, "A correct set of equations for the real-time ellipse Hough transform algorithm," Technical Report, 2009. [Online]. Available: http://centria.csites.fct.unl.pt/file.php%3Fcode=97c5c800fdc8dd8b842e16511e81d2e4

[45] C. Basca, M. Talos, and R. Brad, "Randomized Hough transform for ellipse detection with result clustering," in *Proc. Int. Conf. "Comput. as a Tool" 2005*, vol. 2, 2005, pp. 1397–1400.

[46] X. Bai, C. Sun, and F. Zhou, "Splitting touching cells based on concave points and ellipse fitting," *Pattern Recognit.*, vol. 42, no. 11, pp. 2434–2446, 2009.

[47] Z.-Y. Liu and H. Qiao, "Multiple ellipses detection in noisy environments: A hierarchical approach," *Pattern Recognit.*, vol. 42, no. 11, pp. 2421–2433, 2009.

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

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. 2009 IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

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

[51] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *Int. J. Robot. Res.*, vol. 31, no. 9, pp. 1021–1043, 2012.

[52] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

[53] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. Int. Conf. Robot. Autom. Workshop Open Source Softw.*, Kobe, Japan, 2009, vol. 3, no. 3.2, p. 5.

[54] 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, 1972.

[55] 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, 1990.

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

[57] H. I. Cakir, B. Benligiray, and C. Topal, "Combining feature-based and model-based approaches for robust ellipse detection," in *Proc. 2016 24th IEEE Eur. Signal Process. Conf.*, 2016, pp. 2430–2434.

[58] S. Chen, R. Xia, J. Zhao, Y. Chen, and M. Hu, "A hybrid method for ellipse detection in industrial images," *Pattern Recognit.*, vol. 68, pp. 82–98, 2017.

[59] Q. Ji and R. M. Haralick, "A statistically efficient method for ellipse detection," in *Proc. 1999 IEEE Int. Conf. Image Process.*, vol. 2, 1999, pp. 730–734.

[60] H. Dong, G. Sun, W.-C. Pang, E. Asadi, D. K. Prasad, and I.-M. Chen, "Fast ellipse detection via gradient information for robotic manipulation of cylindrical objects," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2754–2761, Oct. 2018.

[61] R. Grbić, D. Grahovac, and R. Scitovski, "A method for solving the multiple ellipses detection problem," *Pattern Recognit.*, vol. 60, pp. 824–834, 2016.

[62] E. Kim, M. Haseyama, and H. Kitajima, "Fast and robust ellipse extraction from complicated images," in *Proc. IEEE Int. Conf. Inf. Technol. Appl.*, 2002.

[63] L. Libuda, I. Grothues, and K.-F. Kraiss, "Ellipse detection in digital image data using geometric features," in *Advances in Computer Graphics and Computer Vision*. Berin, Germany: Springer, 2007, pp. 229–239.

[64] H. Dong, I.-M. Chen, and D. K. Prasad, "Robust ellipse detection via arc segmentation and classification," in *Proc. 2017 IEEE Int. Conf. Image Process.*, 2017, pp. 66–70.

[65] 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, 2018.

[66] Y. Xie and Q. Ji, "A new efficient ellipse detection method," in *Proc. IEEE 16th Int. Conf. Pattern Recognit.*, vol. 2, 2002, pp. 957–960.

[67] J. Ni, M. K. Singh, and C. Bahlmann, "Fast radial symmetry detection under affine transformations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 932–939.

[68] C. Liu and W. Hu, "Effective method for ellipse extraction and integration for spacecraft images," *Opt. Eng.*, vol. 52, no. 5, 2013, Art. no. 057002.

[69] C. Arellano and R. Dahyot, "Robust ellipse detection with Gaussian mixture models," *Pattern Recognit.*, vol. 58, pp. 12–26, 2016.
[70] T. Lu, W. Hu, C. Liu, and D. Yang, "Effective ellipse detector with polygonal curve and likelihood ratio test," *IET Comput. Vis.*, vol. 9, no. 6, pp. 914–925, 2015.
[71] B.-K. Kwon, Z. Teng, T.-J. Roh, and D.-J. Kang, "Fast ellipse detection based on three point algorithm with edge angle information," *Int. J. Control Autom. Syst.*, vol. 14, no. 3, pp. 804–813, 2016.

Huixu Dong received the B.Sc. degree in mechatronics engineering from the Harbin Institute of Technology, Harbin, China, in 2013, and is currently working toward the Ph.D. degree in mechanical & aerospace engineering at the Robotics Research Centre of Nanyang Technological University, Singapore.

His current research interests include robotic perception and grasp, robot-oriented computer vision, robot-oriented artificial intelligence and optimization design of robotic grasping mechanism.

Ehsan Asadi received the B.Sc. and M.Sc. degrees both in mechanical engineering, in 2002 and 2005, respectively, and the Ph.D. degree in aerospace engineering from the Politecnico di Milano, Milan, Italy, in 2013.

He is a Postdoctoral Research Fellow with the Robotics Research Center, Nanyang Technological University, Singapore. He was with various research groups and robotics companies including, Poli-rotorcraft, Italy, and Terabee S.A.S, France. His main research interests include sensor fusion, vision-aided inertial navigation, simultaneous localization and mapping, robot vision and field robotics.

Guangbin Sun received the B.Sc., M.Sc., and Ph.D. degrees from Northeast University, Shenyang, China, in 2007, 2009, and 2015, respectively, all in mechatronics engineering.

During the period between 2010 and 2012, he was an exchange Ph.D candidate with Carnegie Mellon University. He is currently a Research Fellow with Nanyang Technological University, Singapore. His current research interests include motion planning of humanoid robot and industrial robotic arm as well as human–robot interaction.

Dilip K. Prasad received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Dhanbad, India, in 2003, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2013, both in computer science and engineering.

He is currently a Research Fellow with Nanyang Technological University. He has authored more than 60 internationally peer-reviewed research articles. His current research interests include image processing, pattern recognition, and computer vision.

I-Ming Chen received the B.S. degree in mechanical engineering from National Taiwan University, Taipei, Taiwan, in 1986, and the M.S. and Ph.D. degrees in aerospace engineering from the California Institute of Technology, Pasadena, CA, USA, in 1989 and 1994, respectively.

He is a Full Professor with the School of Mechanical and Aerospace Engineering, previous directions of Robotics Research Centre and Intelligent System Centre in Nanyang Technological University (NTU), Singapore.

Dr. Chen is a Fellow of ASME and General Chairman of 2017 IEEE International Conference on Robotics and Automation. He is a Senior Editor for IEEE TRANSACTIONS ON ROBOTICS. He also acts as the Deputy Program Manager of A∗STAR SERC Industrial Robotics Program to coordinate project and activities under this multi-institutional program involving NTU, National University of Singapore, SIMTech, A∗STAR I2R, and Singapore University of Technology and Design. He works on many different topics in robotics, such as mechanism, actuator, human–robot interaction, perception and grasp, and industrial automation.