

# Real-Time Ellipse Detection for Robotics Applications

Azarakhsh Keipour<sup>1</sup>, Guilherme A. S. Pereira<sup>2</sup> and Sebastian Scherer<sup>3</sup>

**Abstract**—We propose a new algorithm for real-time detection and tracking of elliptic patterns suitable for real-world robotics applications. The method fits ellipses to each contour in the image frame and rejects ellipses that do not yield a good fit. The resulting detection and tracking method is lightweight enough to be used on robots' resource-limited onboard computers, can deal with lighting variations and detect the pattern even when the view is partial. The method is tested on an example application of an autonomous UAV landing on a fast-moving vehicle to show its performance indoors, outdoors, and in simulation on a real-world robotics task. The comparison with other well-known ellipse detection methods shows that our proposed algorithm outperforms other methods with the F1 score of 0.981 on a dataset with over 1500 frames. The videos of experiments, the source codes, and the collected dataset are provided with the paper.

## I. INTRODUCTION

Real-time detection and tracking of circular and elliptic shapes using an onboard vision system are essential for many real-world (mainly robotics) applications. For example, many aerial robot' landing zones consist of an elliptical shape [1], and autonomous cars need to detect the circular road signs [2].

Detecting ellipses in images has been a topic of interest for researchers for a long time [3], [4]. In general, it is possible to classify the available ellipse detection approaches into four classes.

The first class contains voting-based algorithms, including Hough Transform (HT) [5] and the methods based on it. The HT algorithm uses a 5-dimensional parametric space for the ellipse detection task and is too slow for real-time applications. Other methods try to enhance HT by reducing the dimensionality in parametric space [6], [7], [8], performing piecewise-linear approximation for the curved segments [9] or randomizing the method [10], [11], [12]. Some of these modified HT-based methods are fast but generally less accurate and unsuitable for many robotics applications.

The second class contains optimization-based methods. Most popular methods convert the ellipse fitting problem into a least-squares optimization problem and solve the problem to find the best fit [13], [14]. These methods are generally not robust and tend to produce many false positives. However, their high processing speed is useful for quickly estimating the ellipse as the first step in other methods. Another group of optimization-based methods try to solve the non-linear optimization problem of fitting an ellipse using a genetic

algorithm [15], [16]. These algorithms perform well in noisy images with multiple ellipses, but their processing time makes them impractical for real-time applications.

The third class consists of methods based on edge linking and curve grouping [17]. These methods can detect ellipses from complex and noisy images but are generally computationally expensive and cannot be used in real-time applications.

The last class contains the methods that use an ad-hoc approach or combine the methods from the first three classes to achieve a faster and more accurate ellipse detection. A real-time method proposed by Nguyen et al. [18] detects arc-segments from the image and groups them into elliptical arcs to estimate the ellipse parameters using a least-square optimization. A method proposed by Fornaciari et al. [19] combines arc grouping with Hough Transform in a decomposed parameter space to estimate the ellipse parameters. In this way, it achieves a real-time performance comparable to slower, more robust algorithms.

Currently, in many robotics applications, the detection of the previously-known elliptic objects is done by general learning-based object detection methods [20]. However, this approach requires collection and enhancement of large amount of data for each individual object and access to powerful computing systems for training. Recently, there has been an effort to implement a learning-based method specifically for ellipse detection [21]. The method has been shown to work well for occlusions and partial views. However, it requires a powerful processor and is not yet suitable for real-time execution on onboard computers.

While many ellipse detection algorithms are proposed for computer vision tasks ([22], [23], [24]), the performance of these methods drops when used in real-world robotics tasks. Some of the challenges in these applications include:

- The algorithm should work online (with a frequency greater than 10 Hz), generally on the robot's resource-limited onboard computer.
- The elliptical object should be detected when it is far from or close to the robot.
- The shape of the ellipse is transformed by a projective distortion, which occurs when the shape is seen from different views.
- There is a wide range of possible illumination conditions (e.g., cloudy, sunny, morning, evening, indoor lighting).
- Due to reflections (e.g., from sources like sun, bulbs), the pattern may not always be seen in all the frames, even when the camera is close.
- In some frames, there may be shadows on the pattern (e.g., the shadow of the robot or trees around).

<sup>1,3</sup> Robotics Institute, Carnegie Mellon University, Pittsburgh, PA [keipour, basti]@cmu.edu

<sup>2</sup> Department of Mechanical and Aerospace Engineering, West Virginia University, WV guilherme.pereira@mail.wvu.edu

- In some frames, the pattern may be seen only partially (due to occlusion or being outside of the camera view).
- In some frames, the pattern may not be present.

Different approaches have been devised in the literature to detect and track an elliptic pattern in real applications to address the challenges above. For example, [25] uses the circular Hough-transform algorithm for the initial detection, which is a slow algorithm, and then uses other features in their pattern for the tracking. The authors of [26] have developed a Convolutional Neural Network trained with over 5,000 images collected from the pattern moving at a maximum speed of 15 km/h at various heights to select (validate) the desired ellipse from all the ellipses detected using the method in [27]. The authors of [28] use the method proposed in [19] for ellipse detection only when their robot is far from the pattern and switch to other features in their pattern for the closer frames.

It is possible to use a predictor filter along with the detector to narrow down the region of interest (ROI) and improve the detection speed of ellipses across the frames. The authors of [29] and [30] use Kalman filter to predict the direction of the ellipse movement in the frames, increase the detection speed and reduce the false positives. In many robotics applications, the combined simultaneous motions of the elliptic pattern, the robot, and the camera complicate the prediction provided by such predictors, resulting in the loss of tracking on many frames, triggering a detection on the entire frame and slowing the tracking speed.

This paper proposes a novel ellipse detection method that performs on resource-limited onboard computers in real-time. Our proposed ad-hoc method first extracts all contours from the input image and then uses a least-square method to fit an ellipse to each contour. It tests how well the estimated ellipse fits the contour and starts rejecting the contours using several criteria. The remaining contours are accepted as resulted ellipses, and if there are no contours left (e.g., due to the absence of ellipses in the frame), the algorithm returns an empty set of ellipses. The detection method is combined with a simple tracking algorithm that changes detection parameters as necessary and significantly increases the elliptical pattern detection's precision and performance. This simple tracker has been shown to be faster than using predictor filters due to the reduced tracking losses. The resulting detector and tracking method can deal with lighting variations and detect the pattern even when the view is partial. By comparing the results of our experiments on a collected dataset to the other methods tested on similar datasets we show that our method outperforms all the other real-time ellipse detection methods proposed so far.

Section II explains the developed method for detection and tracking of the elliptical targets; Section III shows an example real-time application that uses the proposed algorithm and compares the performance with some other available well-known methods. Finally, Section IV discusses how the proposed method can be further improved in the future.

---

### Algorithm 1 Proposed approach for ellipse detection

---

```

1: ▷ This function detects all the ellipses found using the
   input criteria (thresholds)
2: function DETECTELLIPSES(frame, thresholds)
3:   ▷ Initialize an empty set for the detected ellipses
4:   Detections ← ∅
5:
6:   ▷ Detect all edges in the frame
7:   edges ← DETECTEDGES(frame)
8:
9:   ▷ Extract contours from the detected edges
10:  contours ← EXTRACTCONTOURS(edges)
11:
12:  for each c ∈ contours do
13:    ▷ Reject if the contour is too small
14:    if  $|c.pixels| < minContourSize$  then continue
15:
16:    ▷ Fit an ellipse to each contour
17:    ellipse ← FITELLIPSE(c)
18:
19:    ▷ Reject ellipses with unreasonable dimensions
20:    if ellipse.largeAxis > maxAxisSize then continue
21:    if ellipse.smallAxis < minAxisSize then continue
22:    axisRatio ← ellipse.largeAxis / ellipse.smallAxis
23:    if axisRatio > maxAxisRatio then continue
24:
25:    ▷ Reject contours with small overlap with ellipse
26:    overlap ← c.pixels ∩ ellipse.pixels
27:    if  $|overlap| / |c.pixels|$  is small then continue
28:
29:    ▷ Reject ellipses with small overlap with the edges
30:    overlap ← ellipse.pixels ∩ edges.pixels
31:    if  $|overlap| / |ellipse.pixels|$  is small then continue
32:
33:    ▷ Add the detected ellipse to the set of detections
34:    Detections.Insert(ellipse)
35:  end for
36:
37:  ▷ Return the detections after all contours are processed
38:  return Detections
39: end function

```

---

## II. ELLIPSE DETECTION AND TRACKING

The idea of the proposed ellipse detection algorithm is to fit ellipses to all the contours in a frame or the region of interest and then decide if the ellipse is a good fit. With real-time ellipse fitting algorithms and fast criteria for checking the fit, the result is a real-time detection algorithm that can detect ellipses as long as the elliptic pattern's contours are (at least partially) extracted during the contour extraction. With a suitable contour extraction method, the whole detection becomes robust to the lighting and environmental changes.

The pseudocode for the proposed ellipse detector is shown in Algorithm 1. The detection function receives a frame and a set of threshold values used in the function and returns a set of detected ellipses. A step-by-step explanation of the

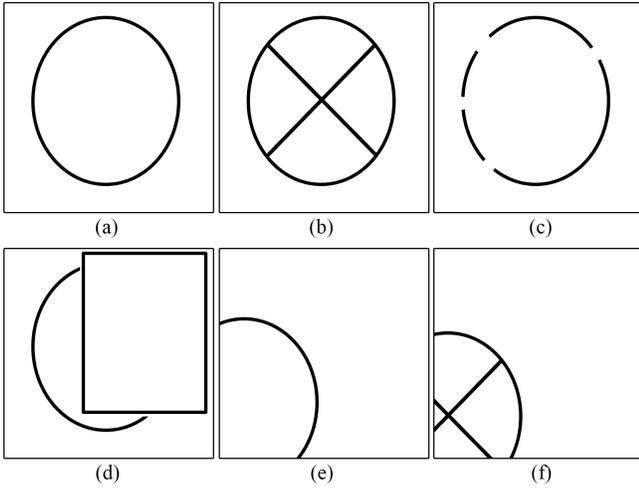


Fig. 1: Contour types of ellipses in different conditions: (a) A single contour containing a full ellipse. (b) A single contour containing a full ellipse with additional connected contours from the rest of the pattern. (c) Multiple contours, together constructing a full ellipse. (d) A single contour containing an ellipse that is partially occluded by another object. (e) A single contour containing a partially-seen ellipse. (f) A single contour containing a full ellipse with additional connected contour branches that is partially seen in the frame.

algorithm is as follows:

- 1) The first step of the algorithm is to extract edges from the input frame and create the edge image. We used the Canny edge detector [31], considering that the thresholds should be selected carefully to extract suitable edges in a large variety of conditions (e.g., illumination) while preventing the generation of too many edges. Usually, it is beneficial for smaller targets to produce more edges; this action will increase the processing time but reduces the probability of not having an edge for the elliptic target.
- 2) The resulted edges are utilized to extract contours using the algorithm proposed by Suzuki and Abe [32]. This step helps make connections between relevant edges and enables the extraction of shapes in a frame. Ideally, each contour is a collection of connected points constructing a shape's border in the edge image.
- 3) Each contour is processed individually to determine if it is a part of an ellipse or not. For robotics tasks, an ellipse can have one of the following contour types:
  - A single contour containing a full ellipse without any occlusions, broken segments or additional contours (Fig. 1(a)).
  - A single contour containing an ellipse with additional connected contours from the rest of the pattern (Fig. 1(b)).
  - Multiple contours, together constructing a full ellipse (Fig. 1(c)).
  - A single contour containing an ellipse that is partially occluded by other objects (Fig. 1(d)).
  - A single contour containing an ellipse that is partially

seen in the frame (Fig. 1(e)).

- A combination of the above contour types (Fig. 1(f)).

In order to correctly detect the above contour types, the following process is performed on each contour:

- a) If a contour has a very small number of pixels, it is ignored since it is most probably just noise.
- b) An ellipse is fit to the contour using the least-square approximation method described by Fitzgibbon and Fisher [14]. The method fits an ellipse to any input contour; therefore, the contour should be further processed to determine the actual ellipses.
- c) The resulting ellipse will be ignored if any of the axes are too small or if the ellipse's eccentricity is high (close to 1). In high eccentricity, the resulting ellipse is similar to a line and is not really an ellipse.
- d) The current contour is intersected with the perimeter of the resulted ellipse, described by its center point, minor and major axes, and rotation angle. Then, the ratio of the number of pixels in the intersection to the number of all pixels in the contour is calculated:

$$ContourOverlap = \frac{|Contour \cap Ellipse|}{|Contour|}, \quad (1)$$

where  $|\cdot|$  is used for the number of pixels. A low result means that the contour and the resulted ellipse do not fit well, and a significant portion of the contour is not lying on the fitted ellipse. In this case, the contour is ignored and not further processed.

- e) Finally, the ellipse is intersected with the edges, and the ratio of the number of pixels in the intersection to the number of all the pixels in the ellipse is calculated:

$$EllipseOverlap = \frac{|Ellipse \cap Edges|}{|Ellipse|}, \quad (2)$$

where  $|\cdot|$  is used for the number of pixels. A low result means that a significant portion of the ellipse does not correspond to any contours in the image. The reason that the ellipse is intersected with the edge image instead of only its constructing contour is that due to noise or imperfect contour detection step, sometimes an ellipse is broken into two or more contours (e.g., the cases like Fig. 1(c)). In these cases, checking the ellipse against a single contour will give a low ratio and results in false negatives. To take care of the cases similar to Figure 1(e), it is essential to count only the ellipse pixels that are actually lying in the image; otherwise, the result will be too low, and a partially viewed ellipse may get rejected. Additionally, to make the algorithm more robust to slightly imperfect ellipses, increasing the edges' thickness before intersecting them with the ellipse is beneficial.

- f) If an ellipse is not rejected in the previous steps, it represents a real ellipse in the image and is added to the set of detected ellipses.
- 4) Due to the target ellipse's thickness, there is a chance of detecting two or more concentric ellipses. Therefore, the

returned set of detected ellipses in the frame is further processed to find the concentric ellipses. All the non-concentric sets of ellipses are ignored when this happens.

The proposed algorithm can also detect the ellipses with partial occlusion or the ellipses exceeding the image boundaries. The rejection criteria can be chosen in a way to accept ellipses in such cases.

Choosing higher rejection criteria for ellipse detection generally helps to eliminate potential false positives. Therefore, it is beneficial to have higher rejection thresholds when there is no prior information about the pattern location and size in the image. However, after the first detection of the elliptical pattern, it is possible to change the initial parameters and conditions for the detection to enhance the performance and increase the detection rate. Decreasing the detection threshold values reduces the probability of losing the target ellipse in the following frames due to illumination variation, noise, occlusion, or other changes in the conditions. Additionally, if the approximate movement of the target is known, setting the region of interest (ROI) to the area where the pattern is expected to be seen will decrease both the processing time and the probability of falsely detecting other similar shapes that were rejected in the initial frame due to the higher thresholds. Furthermore, whenever the detected target is large enough to be detected in the input frame with a smaller scale, the frame can be scaled down to increase the processing speed. Performing ellipse detection on the smaller frame takes less CPU time and is much faster.

We propose these steps that can be combined with the ellipse detection algorithm to track the detected elliptical pattern in the next frames more efficiently:

- 1) Significantly decrease the *ContourOverlap* and *EllipseOverlap* threshold values. This threshold change increases the detection rate and helps the algorithm to keep tracking the target.
- 2) Determine the region of interest, which includes the current detected target and expands in all directions based on the distance from the target, the robot's relative speed and the pattern, and other available information. For example, if the distance is far and the relative linear and angular speeds are low, the target is expected to be found close to the current detection coordinates in the next frame.
- 3) Decreasing the scale of the frame by order of two (up until a set threshold) every time that the detected target is larger than a specified size and scaling the frame up again by order of two (up to the actual frame scale) every time the detected target is smaller than a chosen threshold. To make the approach more robust, ellipse detection with initial higher parameters is performed once again on a higher scale if no candidate targets are found on a lower scale. The scale change is performed to reduce the execution time, as the algorithm needs to process only a quarter number of pixels every time it scales the frame down.

Algorithms 2 and 3 show the described method in more

---

#### Algorithm 2 Proposed approach for elliptic target tracking

---

```

1: ▷ This function reads the frames from a video stream
   and performs tracking of a target
2: function TRACKTARGET(videoStream)
3:   ▷ Set frame scale, ROI and tracking status
4:   isTracking ← false
5:   scale ← 1
6:   roi ← ∅
7:
8:   while videoStream ≠ ∅ do
9:     frame ← READFRAME(videoStream)
10:    ▷ Detect the ellipse in the frame or ROI
11:    if isTracking = false then
12:      offsetTarget ← DETECTTARGET(frame, scale)
13:    else
14:      offsetTarget ← DETECTTARGET(roi, scale)
15:    end if
16:
17:    if offsetTarget = ∅ and scale > 1 then
18:      ▷ If target not found, try on higher frame scale
19:      scale ← scale ÷ 2
20:      if isTracking = false then
21:        offsetTarget ← DETECTTARGET(frame, scale)
22:      else
23:        offsetTarget ← DETECTTARGET(roi, scale)
24:      end if
25:    else
26:      ▷ If found, set the proper scale for next frame
27:      isTracking ← true
28:      if offsetTarget.size > maxTargetSize then
29:        if scale < maxScale then
30:          scale ← scale × 2
31:        end if
32:      else if offsetTarget.size < minTargetSize then
33:        if scale > 1 then
34:          scale ← scale ÷ 2
35:        end if
36:      end if
37:    end if
38:
39:    ▷ Compensate the target offset resulted from detec-
   tion in ROI. Do nothing if ROI is null
40:    target ← COMPENSATEOFFSET(offsetTarget, roi)
41:
42:    ▷ Set the tracking status and ROI for the next frame
43:    if target = ∅ then
44:      isTracking ← false
45:      roi ← ∅
46:    else
47:      isTracking ← true
48:      ▷ Expand the area around the detected target
49:      roi ← CALCULATEROI(target)
50:    end if
51:
52:    return target
53:  end while
54: end function

```

---

**Algorithm 3** Proposed approach for elliptic target detection in a given frame scale

---

```

1: ▷ This function detects the target in the input frame at
  the specified frame scale
2: function DETECTTARGET(frame, scale, isTracking)
3:
4:   ▷ Resize the frame to make the processing faster
5:   scaledFrame ← SCALEIMAGE(frame, 1 / scale)
6:
7:   ▷ Set detection parameters based on inputs
8:   thresholds ← DETERMINEPARAMS(isTracking, scale)
9:
10:  ▷ Detect ellipses using the specified thresholds
11:  detections ← DETECTELLIPSES(scaledFrame, thresh-
12:    olds)
13:
14:  ▷ Select the actual target from the detected ellipses
15:  scaledTarget ← SELECTTARGET(detections)
16:
17:  ▷ Rescale the detected target to original scale
18:  target ← SCALESIZE(scaledTarget)
19:
20:  ▷ Return the detected target (or  $\emptyset$  if not detected)
21:  return target
22: end function

```

---

detail.

### III. EXPERIMENTS AND RESULTS

#### A. Elliptic Target Dataset and Parameter Selection

We created a dataset with sequences recorded using a UAV from a stationary and moving vehicle carrying an elliptical platform at various distances, angles, and illumination conditions. The dataset contains 1,511 frames (1,378 positive and 133 negative frames) and 456 frames of a thinner version of the same pattern. The size of the frames is  $640 \times 360$ , and the ground truth for the detections is provided. The dataset can be accessed from <http://theairlab.org/landing-on-vehicle>.

The thresholds selection of the proposed ellipse detection algorithm depends on the tolerance for false positives vs. false negatives in the application. However, in practice, the detection is not too sensitive to the parameters in most cases, and they can be selected from a broad range. For our tests on the AirLab Elliptic Target Detection Dataset, we empirically chose the values shown in Table I. The GUI tool provided with the code helps with the calibration process letting the user see the parameters' effects in real-time. The ellipse detection parameters are independent of the lighting conditions. Therefore after a one-time calibration, the algorithm should detect the pattern in a wide range of weather conditions (e.g., sunny, cloudy, snowy) as long as the light is enough for the camera to capture the pattern.

In order to assess the sensitivity of the algorithm against the thresholds, Figure 2 shows the performance of the algorithm for different values of *ContourOverlap* with the

TABLE I: Ellipse Detection parameters chosen for the tests on the AirLab Elliptic Target Detection Dataset.

Parameter	Value	Justification
<i>ContourOverlap</i> for detection	0.95	To prevent False Positives.
<i>ContourOverlap</i> for tracking	0.7	To enhance target tracking.
<i>EllipseOverlap</i> for detection	0.95	To prevent False Positives.
<i>EllipseOverlap</i> for tracking	0.3	To enhance target tracking.

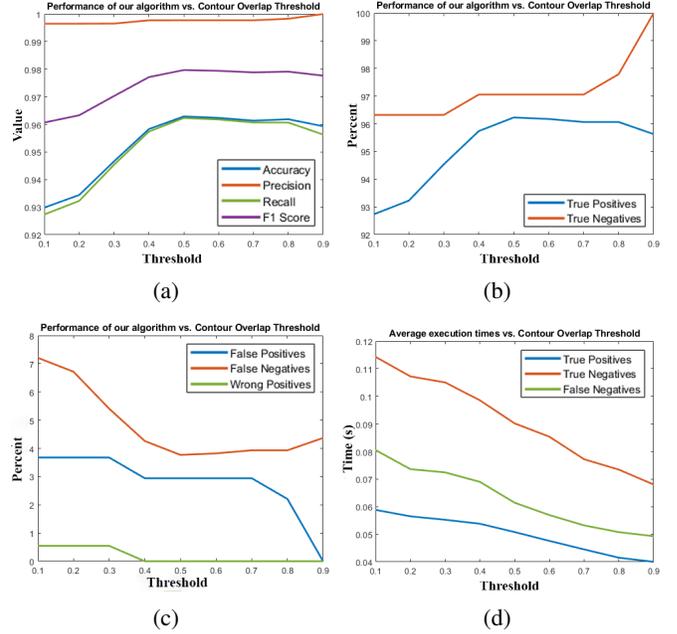


Fig. 2: Performance and execution times of our algorithm vs. contour overlap threshold (*ContourOverlap* parameter). The value of *EllipseOverlap* is fixed to 0.7 for all the experiments. Wrong Positives are the wrong detections when the target was present in the frame. (a) Accuracy, Recall, and F1 Score of the algorithm increase with the increase of *ContourOverlap* up to a point. (b) Increasing *ContourOverlap* increases the number of True Negatives while may result in fewer True Positives after some point. (c) Increasing *ContourOverlap* results in a lower number of False Positives and Wrong Detections, while it may result in an increase of the number of False Negatives after some point. (d) The execution time decreases by increasing the *ContourOverlap* parameter.

value of *EllipseOverlap* fixed to 0.7. Additionally, Figure 3 shows the performance of the algorithm for different values of *EllipseOverlap* with the value of *ContourOverlap* fixed to 0.5.

Let us define  $N_{TP}$ ,  $N_{FP}$ ,  $N_{TN}$ ,  $N_{FN}$ ,  $N_{All}$ , and  $N_{WD}$  as the number of True Positive detections, False Positives, True Negatives, False Negatives, the total number of frames and the number of frames with visible targets where the detection was wrong. The measures in the plots are defined as follows using the five outcome types above.

- Accuracy measures the ratio of all the frames that the algorithm gives the correct result (either the target is detected correctly or no target is detected in a frame without a target). It is defined as  $(N_{TP} + N_{TN})/N_{All}$ .
- Precision measures what ratio of all the target detections

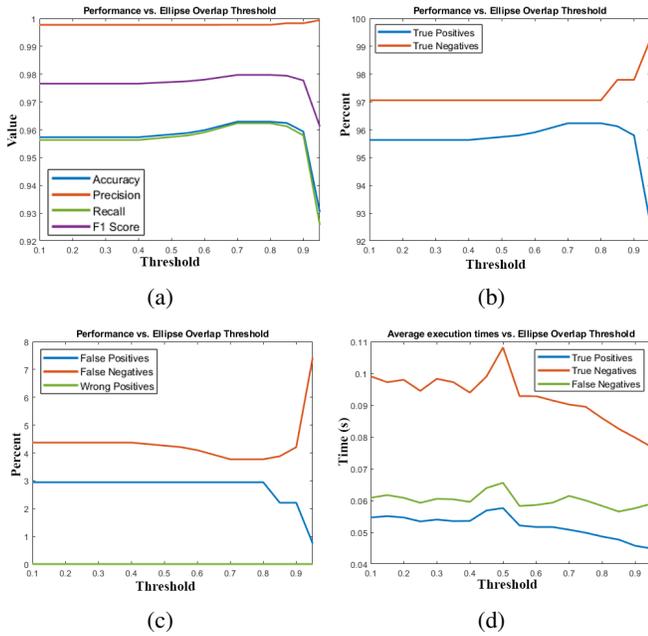


Fig. 3: Performance of our algorithm vs. ellipse overlap threshold (*EllipseOverlap* parameter). The value of *ContourOverlap* is fixed to 0.5 for all the experiments. (a) Accuracy, Recall, and F1 Score of the algorithm very slowly increase with the increase of *EllipseOverlap* up to a breaking point, where they suddenly drop. (b) Increasing *EllipseOverlap* increases the number of True Negatives while reducing the number of True Positives after a certain point. (c) Increasing *EllipseOverlap* results in a lower number of False Positives, while it may result in an increase of the number of False Negatives after some point. (d) The execution time generally decreases by increasing the *EllipseOverlap* value.

is actually correct. It is defined as  $N_{TP}/(N_{TP} + N_{FP} + N_{WD})$ .

- Recall or sensitivity measures the ratio of all the frames containing targets that are correctly detected. It is defined as  $N_{TP}/(N_{TP} + N_{FN} + N_{WD})$ .
- F1 Score is the weighted average of precision and recall and takes both false positives and false negatives into account. F1 Score is defined as  $2 \times (Recall \times Precision)/(Recall + Precision)$ .

It can be seen that increasing the value of *ContourOverlap* up to some point generally decreases the number of false results (both false positives and false negatives) and therefore increases the number of correct results and statistical measures. Although, after some point, the number of detections (true or false positives) starts dropping, and the performance slowly decreases. Additionally, the higher *ContourOverlap* results in more candidate contours being eliminated before further processing, reducing the algorithm’s execution time.

On the other hand, the value of *EllipseOverlap* has a smaller effect on performance and execution time, slightly improving the performance up to a point. If it is set too high, the algorithm starts rejecting more ellipses, causing a drop in the number of positive results, significantly decreasing the

TABLE II: Performance of the ellipse detector methods on the AirLab Elliptic Target Detection Dataset.

Method	Accuracy*	Precision*	Recall*	F1 Score*
Ours	<b>96.56%</b>	<b>99.77%</b>	<b>96.44%</b>	<b>0.981</b>
[33]	3.64%	3.64%	3.99%	0.038
[19]	88.75%	99.67%	87.66%	0.933
[35]	89.81%	99.45%	90.22%	0.965
[27]	78.36%	95.50%	80.04%	0.871

\* As defined in Section III-A.

algorithm’s performance.

The choice of other parameters in the algorithm depends on the application. For example, when the pattern is not expected to be far, setting a higher number for *minContourSize* will improve the speed by eliminating the contours that cannot belong to the elliptic pattern. On the other hand, setting this parameter too high can result in false negatives when the pattern is seen partially or consists of several partial contours. In practice, however, the method has shown to be rather insensitive to these parameters, and generally, there is no need to change the default values. In all our tests, we have used the following values:  $mnAxisSize = 5$ ,  $maxAxisSize = 700$ ,  $maxAxisRatio = 5$ ,  $minContourSize = 50$ ,  $maxScale = 100$ .

### B. Comparison With Other Methods

To compare the performance of our algorithm with other methods, we ran four other methods on the dataset introduced in Section III-A: the MATLAB implementation of a Hough Transform-inspired approach proposed by Xie & Ji [33] with a random sub-sampling inspired by the work in [34], and the C++ implementations of the methods proposed by Fornaciari et al. [19], Prasad et al. [35], and Jia et al. [27]. Table II shows the results and performance of the algorithms on our dataset.

The results show that our implemented algorithm outperforms the other methods in all the criteria. The method by Xie & Ji’s [33] was unable to perform well on the real frames of our test environment due to the relatively low resolution of our frames and the small size of the target in the frames; the few cases it could detect the elliptic target were when the target was covering a large portion of the frame. The main problem with the method by Fornaciari [19] was that it was unable to detect the elliptic targets when more than 25% of the target was outside of the frame (case (e) in Figure 1). The method by Jia et al. [27] is comparatively fast but has a high false positive rate and is unable to detect the ellipses when they are far away (small in the frame). Prasad’s method [35] has comparable performance to Fornaciari’s but is significantly slower. At the same time, our proposed algorithm was fast and still able to detect the target’s elliptic pattern in partial views or when it was small. Additionally, we should note that all the false positive cases of our algorithm on the dataset, detected elliptical drawings on the ground, which would have been rejected if the UAV’s altitude information was used.

Figure 4 shows results for the detection of the elliptical pattern in some sample frames from the dataset. The method

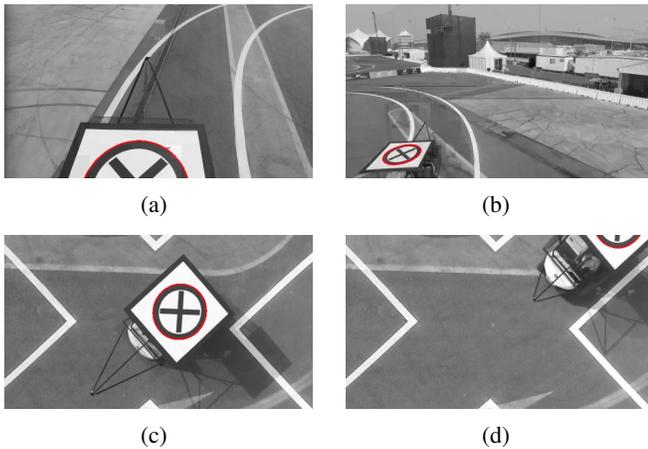


Fig. 4: Result of the proposed algorithm on sample frames. The red ellipse indicates the detected pattern on the deck of the moving vehicle.

by Fornaciari et al. is unable to detect the ellipses in Figure 4(a) and Figure 4(d).

Table III compares the execution times of the ellipse detection methods on the same dataset (using Intel Core i5-4460 CPU @ 3.20 GHz). The implementation of Xie & Ji’s method is done in MATLAB, which gives much higher execution times than C++ implementations. Therefore we excluded it from Table III. Fornaciari et al. method has similar average execution times to our algorithm. However, their approach provides slightly more consistent execution times, which can be convenient for control systems using the detection output for robot control. On the other hand, our method’s speed significantly increases (with frame processing time going down to just a few milliseconds) when the robot gets closer to the target pattern. This increase in speed especially helps the system to have a much higher detection rate when a higher processing speed is needed for the robot to approach the moving vehicle for landing.

### C. Example Application

To test the proposed ellipse detection and tracking method’s performance, it was used with a visual servoing method for an autonomous UAV landing on a circular pattern painted on top of moving platforms in indoor, outdoor, and simulated environments [36]. Figure 5 shows screenshots of the method in these different lighting conditions. Videos for these experiments and the project details can be accessed at <http://theairlab.org/landing-on-vehicle>.

## IV. FUTURE WORK AND DISCUSSION

The proposed ellipse detection and tracking algorithm has shown its performance in an example application and has outperformed other standard methods in our tests. However, the proposed method has mainly been tested on a set of elliptical patterns used in our current and prior research. Utilizing the method for other applications may require further testing and improvements. The following suggestions can further enhance the performance and increase the method’s usability in real-world robotics applications.

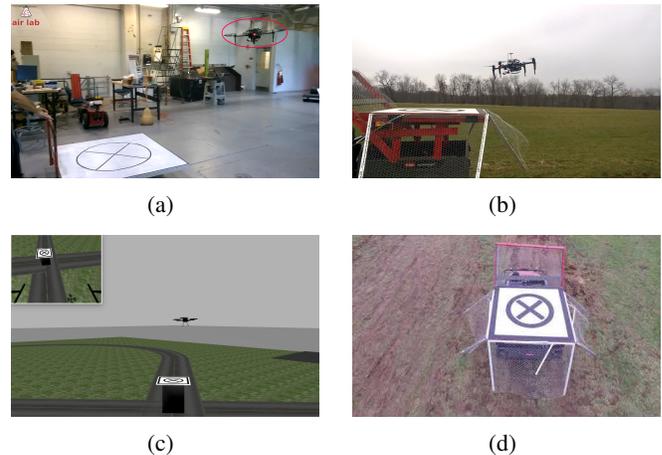


Fig. 5: Screenshots from video sequences showing our autonomous UAV landing on a pattern moving at up to 4.2 m/s speed in various lighting and environmental conditions.

The underlying algorithms used in the ellipse detection steps are the most common methods already available in the OpenCV library. The choice has been made to allow fast implementation by the potential reader and convenience. If better performance is required, the whole method’s execution speed and performance can be improved by replacing steps such as ellipse fitting with faster and better algorithms.

Finally, if the robot’s camera is not perfectly rectified, it may lose track of the elliptical target at close distances where only a small portion of the pattern is visible at such a skewed angle that it causes the circle to be seen as non-elliptical in the camera. The problem exists for any ellipse detection algorithm but can be improved using a robust tracker (such as Kernelized or Discriminative correlation filters) instead of a detector to track the target ellipse when the robot’s camera is too close to the elliptical pattern.

## ACKNOWLEDGMENT

Authors want to thank Near Earth Autonomy for their support in this project. The project was sponsored by Carnegie Mellon University Robotics Institute and Mohamed Bin Zayed International Robotics Challenge. During the realization of this work, G. Pereira was supported by UFMG and CNPq/Brazil.

## REFERENCES

- [1] S. Zhao, Z. Hu, M. Yin, K. Z. Y. Ang, P. Liu, F. Wang, X. Dong, F. Lin, B. M. Chen, and T. H. Lee, “A robust real-time vision system for autonomous cargo transfer by an unmanned helicopter,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 2, pp. 1210–1219, 2015.
- [2] A. Soetedjo and K. Yamada, “Fast and robust traffic sign detection,” in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 2005, pp. 1341–1346.
- [3] S. Tsuji and F. Matsumoto, “Detection of elliptical and linear edges by searching two parameter spaces,” in *IJCAI*, 1977, pp. 700–705.
- [4] Tsuji and Matsumoto, “Detection of ellipses by a modified hough transformation,” *IEEE Transactions on Computers*, vol. C-27, no. 8, pp. 777–781, 1978.
- [5] J. Illingworth and J. Kittler, “A survey of the hough transform,” *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0734189X88800331>

TABLE III: Execution times of ellipse detector methods on the AirLab Elliptic Target Detection Dataset.\*

Method	True Positive			True Negative			False Positive			False Negative		
	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.
Our Method	32.2	<b>64.1</b>	<b>1.5</b>	48.9	68.1	<b>6.6</b>	48.4	49.6	46.4	39.1	64.0	<b>2.5</b>
Fornaciari, et al [19]	26.5	87.3	15.5	25.4	54.2	15.3	-	-	-	<b>23.8</b>	<b>60.3</b>	14.7
Prasad, et al [35]	36.2	70.0	25.5	34.4	64.9	20.0	-	-	-	38.8	72.1	26.6
Jia, et al [27]	<b>24.4</b>	67.3	13.1	<b>21.2</b>	<b>52.7</b>	11.6	-	-	-	<b>23.8</b>	63.3	12.8

\* All times are in milliseconds.

- [6] A. Goneid, S. El-Gindi, and A. Sewisy, "A method for the hough transform detection of circles and ellipses using a 1-dimensional array," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 4, 1997, pp. 3154–3157 vol.4.
- [7] H. Yuen, J. Illingworth, and J. Kittler, "Detecting partially occluded ellipses using the hough transform," *Image and Vision Computing*, vol. 7, no. 1, pp. 31–37, 1989. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0262885689900176>
- [8] S.-C. Zhang and Z.-Q. Liu, "A robust, real-time ellipse detector," *Pattern Recognition*, vol. 38, no. 2, pp. 273–287, 2005. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320304001372>
- [9] R. K. Yip, P. K. Tam, and D. N. Leung, "Modification of hough transform for object recognition using a 2-dimensional array," *Pattern Recognition*, vol. 28, no. 11, pp. 1733–1744, 1995. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S003132039500031T>
- [10] Z. Cheng and Y. Liu, "Efficient technique for ellipse detection using restricted randomized hough transform," in *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, vol. 2, April 2004, pp. 714–718 Vol.2.
- [11] R. A. McLaughlin, "Randomized hough transform: Improved ellipse detection with comparison to electronic annexes available. see <http://www.elsevier.nl/locate/patrec.1>," *Pattern Recognition Letters*, vol. 19, no. 3, pp. 299–305, 1998. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167865598000105>
- [12] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: Randomized hough transform (rht)," *Pattern Recognition Letters*, vol. 11, no. 5, pp. 331–338, 1990. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/016786559090042Z>
- [13] S. J. Ahn, W. Rauh, and H.-J. Warnecke, "Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola," *Pattern Recognition*, vol. 34, no. 12, pp. 2283–2303, 2001. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320300001527>
- [14] A. Fitzgibbon and R. B. Fisher, "A buyer's guide to conic fitting," in *In British Machine Vision Conference*, 1995.
- [15] K.-U. Kasemir and K. Betzler, "Detecting ellipses of limited eccentricity in images with high noise levels," *Image and Vision Computing*, vol. 21, no. 2, pp. 221–227, 2003. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0262885602001555>
- [16] J. Yao, N. Kharma, and P. Grogono, "A multi-population genetic algorithm for robust and fast ellipse detection," *Pattern Analysis and Applications*, vol. 8, no. 1, pp. 149–162, Sep 2005. [Online]. Available: <https://doi.org/10.1007/s10044-005-0252-7>
- [17] F. Mai, Y. Hung, H. Zhong, and W. Sze, "A hierarchical approach for fast and robust ellipse extraction," *Pattern Recognition*, vol. 41, no. 8, pp. 2512–2524, 2008. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320308000642>
- [18] T. M. Nguyen, S. Ahuja, and Q. M. J. Wu, "A real-time ellipse detection based on edge grouping," in *2009 IEEE International Conference on Systems, Man and Cybernetics*, Oct 2009, pp. 3280–3286.
- [19] M. Fornaciari, A. Prati, and R. Cucchiara, "A fast and effective ellipse detector for embedded vision applications," *Pattern Recognition*, vol. 47, no. 11, pp. 3693 – 3708, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0031320314001976>
- [20] A. Bhattacharya, A. Gandhi, L. Merkle, R. Tiwari, K. Warrior, S. Winata, A. Saba, K. Zhang, O. Kroemer, and S. Scherer, "Carnegie mellon team tartan: Mission-level robustness with rapidly deployed autonomous aerial vehicles in the mbzirc 2020," *Field Robotics*, p. to appear, 2021. [Online]. Available: [arXiv:2107.01507](https://arxiv.org/abs/2107.01507)
- [21] W. Dong, P. Roy, C. Peng, and V. Isler, "Ellipse r-cnn: Learning to infer elliptical object from clustering and occlusion," *IEEE Transactions on Image Processing*, vol. 30, pp. 2193–2206, 2021.
- [22] Z. Wang, D. Chen, J. Gong, and C. Wang, "Fast high-precision ellipse detection method," *Pattern Recognition*, vol. 111, p. 107741, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320320305446>
- [23] Z. Liu, X. Liu, G. Duan, and J. Tan, "A real-time and precise ellipse detector via edge screening and aggregation," *Machine Vision and Applications*, vol. 31, no. 7, p. 64, Sep 2020. [Online]. Available: <https://doi.org/10.1007/s00138-020-01113-1>
- [24] C. Meng, Z. Li, X. Bai, and F. Zhou, "Arc adjacency matrix-based fast ellipse detection," *IEEE Transactions on Image Processing*, vol. 29, pp. 4406–4420, 2020.
- [25] M. Beul, M. Nieuwenhuisen, J. Quenzel, R. A. Rosu, J. Horn, D. Pavlichenko, S. Houben, and S. Behnke, "Team nimbrot at mbzirc 2017: Fast landing on a moving target and treasure hunting with a team of micro aerial vehicles," *Journal of Field Robotics*, vol. 36, no. 1, pp. 204–229, 2019. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21817>
- [26] R. Jin, H. M. Owais, D. Lin, T. Song, and Y. Yuan, "Ellipse proposal and convolutional neural network discriminant for autonomous landing marker detection," *Journal of Field Robotics*, vol. 36, no. 1, pp. 6–16, 2019. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21814>
- [27] Q. Jia, X. Fan, Z. Luo, L. Song, and T. Qiu, "A fast ellipse detector using projective invariant pruning," *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3665–3679, 2017.
- [28] Z. Li, C. Meng, F. Zhou, X. Ding, X. Wang, H. Zhang, P. Guo, and X. Meng, "Fast vision-based autonomous detection of moving cooperative target for unmanned aerial vehicle landing," *Journal of Field Robotics*, vol. 36, no. 1, pp. 34–48, 2019. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21815>
- [29] C. Meng, Z. Hu, and H. Sun, "An ellipse feature tracking method based on the kalman filter," in *2015 8th International Congress on Image and Signal Processing (CISP)*, 2015, pp. 882–887.
- [30] M. Vincze, "Robust tracking of ellipses at frame rate," *Pattern Recognition*, vol. 34, no. 2, pp. 487–498, 2001. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320399002307>
- [31] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.
- [32] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.
- [33] Y. Xie and Q. Ji, "A new efficient ellipse detection method," in *Object recognition supported by user interaction for service robots*, vol. 2, 2002, pp. 957–960 vol.2.
- [34] C. A. Basca, M. Talos, and R. Brad, "Randomized hough transform for ellipse detection with result clustering," in *EUROCON 2005 - The International Conference on "Computer as a Tool"*, vol. 2, Nov 2005, pp. 1397–1400.
- [35] D. K. Prasad, M. K. Leung, and S.-Y. Cho, "Edge curvature and convexity based ellipse detection method," *Pattern Recognition*, vol. 45, no. 9, pp. 3204–3221, 2012, best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320312000763>
- [36] A. Keipour, G. A. S. Pereira, R. Bonatti, R. Garg, P. Rastogi, G. Dubey, and S. Scherer, "Visual servoing approach for autonomous UAV landing on a moving vehicle," 2021. [Online]. Available: [arXiv:2104.01272](https://arxiv.org/abs/2104.01272)