Protecting an Autonomous Delivery Agent Against a Vision-Guided Adversary: Algorithms and Experimental Results

Bhaskar Vundurthy, *Member, IEEE*, and K. Sridharan, *Senior Member, IEEE*

***Abstract*—Safety considerations call for deployment of autonomous ground vehicles in defense and high risk zones for transport of goods from one point to another. Such vehicles face the threat of an *intelligent autonomous adversary* that may disrupt the transfer of material. This article investigates the challenges involved in autonomous protection of a delivery agent, via a land-based rescue agent, before interception of the delivery agent by the adversary occurs. In particular, we study how effectively an adversary equipped with a vision sensor can be handled by an autonomous rescue agent operating without vision support and relying only on wireless communication with the delivery agent. Taking capabilities and weights of the three vehicles into account, the delivery agent is assumed to be the slowest while the adversary operates at the highest speed among the three vehicles. A geometric framework based on Apollonius circles is proposed to analyze the interaction between the delivery and rescue agents. The adversary's speed and its moves (based on the direction of the delivery agent) are taken into account, along with the Apollonius circles for the rescue-delivery agent pair, to determine the possibility of capture. Regions in the plane where the delivery and rescue agents can meet, prior to a capture by the adversary, are obtained to compute safe regions for the delivery agent. Algorithms adopted by the delivery agent, rescue agent, and the adversary are described. We, then, explore the challenges in rescue of multiple delivery agents from a vision-guided adversary by introducing additional rescue agents. In particular, we study protection of k delivery agents (from an adversary) via k rescue agents. Algorithms to compute 1) multiple meeting points, one each for a delivery agent-rescue agent pair and 2) the strategy of the adversary to capture any one of the k delivery agents are presented. Experiments with multiple agents show that the delivery and rescue agents can execute their strategies using simply low-end microcontrollers without external memory.**

Manuscript received March 13, 2019; revised September 15, 2019; accepted November 20, 2019. Date of publication December 10, 2019; date of current version May 26, 2020. This work was supported by the Department of Science and Technology, Government of India through SB/S3/EECE/274/2016. Paper no. TII-19-0866. *(Corresponding author: K. Sridharan.)*

The authors are with the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai 600036, India (e-mail: bhaskarvundurthy@gmail.com; sridhara@iitm.ac.in).

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

Digital Object Identifier 10.1109/TII.2019.2958818

***Index Terms*—Autonomous delivery agent, autonomous rescue agent, Apollonius circles, defense and security applications, hardware-efficient implementation, safe regions, vision-guided adversary.**

I. Introduction

Autonomous delivery agents are often entrusted with the task of carrying goods from one point to another in defense zones and accident sites [1], [2]. These agents help reduce casualty while simultaneously gathering valuable information on the environment through sensors mounted on them.

In these scenarios, the delivery agents communicate with others regarding their current status. Goods carried by these agents may have a marker/indicator that guides other members of the team to recognize the contents on arrival at a prespecified destination. For example, an agent carrying medical equipment may be identified by a certain marker while one carrying food may have a different indicator. However, these markers can be exploited by an adversary. Hence, the delivery agents become vulnerable to predatory attacks prior to reaching the destination. These agents may not be necessarily equipped with hardware for taking (self) defensive action in the event of an attack.

In this article, we examine the effect of an autonomous rescue agent tasked with protection of the delivery agent against an attack. We take clues from nature [3] to define capabilities for the agents and the adversary. In practice, shepherds typically monitor their flock (of sheep) visually and thwart an attack by a predator taking measures based partly on the approximate distance of the predator. Similarly, a predator chooses a sheep based on sight as well as its proximity to the latter.

The work described in this article is motivated by the following questions: *Can an autonomous adversary successfully capture an autonomous delivery agent based on a vision sensor in the former? Furthermore, can an autonomous rescue agent protect a delivery agent using merely wireless communication and without explicit vision support? In addition, what challenges arise when there are multiple delivery agents?*

The advantage in not having a vision sensor on the rescue agent is reduction in hardware (and consequently power consumption). Absence of communication hardware on the adversary has similar benefits. However, limited hardware onboard presents several challenges. The rescue agent needs to gather information on the adversary through the delivery agent. Furthermore, the adversary has to react swiftly to changes in the path of the delivery agent.

We present a geometric approach to address these problems. It is based on identification of *safe regions* using the notion of Apollonius circles [4]. The regions partition the plane and help efficiently determine whether a rescue or capture would take place for *different initial locations and speeds of the agents and the adversary*. We assume each of the three entities (delivery agent, rescue agent and adversary) is small and, therefore, can be approximated by a point. Delivery agents are assumed to be the slowest (travelling at a speed denoted by v_d) in view of the cargo they carry. Rescue agents travel at a somewhat higher speed (denoted by v_r) than delivery agents. The communication between delivery and rescue agents offers the latter an advantage to minimize the time for rescue. In order to counter this advantage, we assume the adversary has the maximum speed (denoted by v_a) among the three. Section III gives a detailed description of the problems.

Coordinated motion of agents has been extensively explored in the past [5]. Furthermore, considerable work has been done on interactions between three or more autonomous vehicles, with both cooperation and conflict, in the setting. Defending a moving vehicle from an attack has been the subject of active research since the early 1960s [4], [6]. Interactions between an attacker and a defender have been studied primarily under the framework of pursuit-evasion games and enhancements [4]. To our knowledge, there does not appear to be a detailed investigation of interactions between three or more autonomous vehicles when, for instance, *one of the vehicles has a specific sensor onboard. Furthermore, study of the effect of introduction of additional vehicles into the setting has not received much attention. Section II discusses prior work in detail.*

Compared with prior works, the contributions here are as follows.

1) Development of a geometric framework for analyzing the interaction between a delivery agent, rescue agent, and a vision-equipped adversary when the delivery and rescue agents *do not have explicit vision support*.
2) Design of algorithms for determining capture/rescue of the delivery agent without repeated constructions of the geometric entities at each stage.
3) Identification of *safe regions* by a rescue agent using delivery agent's communication about the adversary.
4) Enhancement of the strategy to rescue additional delivery agents.
5) Incorporation of collision avoidance among vehicles.
6) Hardware-efficient implementation of the algorithms.

The rest of this article is organized as follows. Next section reviews prior work on interactions in autonomous systems. Section III presents the problem statement. Section IV describes the framework for protecting one delivery agent against an adversary. Section V extends the theory to multiple delivery agents. Section VI presents simulations and comparison with prior work. Experimental results are presented in Section VII while Section VIII concludes this article.

II. Relation to Prior Work

The work presented in this article involves a team (comprising a delivery agent and a rescue agent) that is initially not in the same (physical) location. A coalition takes place when a rescue happens and this requires coordinated motion. The adversary, on the other hand, introduces an element of conflict. We, therefore, begin with a review of the literature on coalition and conflict in autonomous systems.

A. Coalition and Conflict in Autonomous Systems

Coalition and coordination have been studied from different perspectives. Studies on collective motion of autonomous robots have been largely inspired by the model in [7] for motion in systems of particles. A dynamic systems approach for multiagent coordination is presented in Chen *et al.* [8]. Coalition formation has been studied from a graph-theoretic perspective in [9]. Tools from matrix theory and algebraic graph theory have been combined to analyze cooperation among multiagent systems in [10].

While prior literature focuses on deterministic systems, issues relating to uncertainty in communication for coalition in multiagent systems have been studied via a convex programming approach in [11]. Uncertainty in information for coalition and issues relating to implementation (such as sampling period) are explored via an algebraic graph theory approach in [12]. The coalition problem is studied with limited information (based on position alone) via a model predictive control strategy in [13]. An extensive review on the coalition and coordination problem is available in [5]. Recent advances on fixed-time cooperative control for multiagent systems are described in [14].

While different perspectives on various aspects of coalition of multiagent systems exist, the presence of an adversary in our setting calls for a substantially different treatment. Adversaries and their impact on security have been studied in industrial settings [15], [16]. Adversaries in multiagent systems involving autonomous vehicles have been largely studied from a game-theoretic perspective [4], [6]. Several works formulate and study various aspects of pursuit-evasion games. For instance, Pachter [17] presented an analysis of a pursuit-evasion game where the pursuer and evader move in a half-plane with the pursuer being faster than the evader. Chung *et al.* [18] reviewed pursuit-evasion in the context of mobile robotics, while a geometric approach to a pursuit-evasion game with visibility constraints is described in [19]. Pursuit-evasion with limited observations is explored in [20].

The work described in this article is somewhat similar to the three-player interactions in [3], [21], [22], and [23]. In particular, these prior works involve three "mobile" players with different roles assigned to each. However, Li *et al.* [21] and [22] assumed a defender whose task is to intercept the adversary. Scott and Leonard [3] presented a mathematical model for predation in which a predator (a bear) pursues two evaders (a mother caribou and its calf). While each of the two evaders is a prey, the evaders are allowed to choose between herding and evasion. We present a

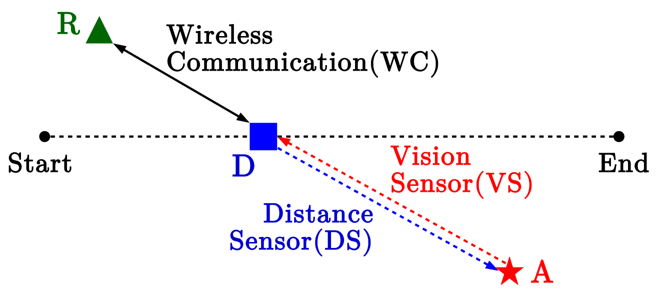

Fig. 1. Scenario involving one delivery agent (D), one rescue agent (R), and an adversary (A). The rescue agent (R) is indicated by a green triangle, adversary (A) by a red star, and delivery agent (D) by a blue square.

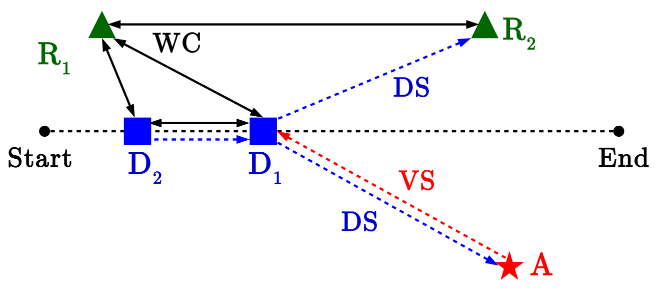

Fig. 2. Adversary's pursuit amidst two delivery and two rescue agents. WC refers to wireless communication between the delivery and rescue agents. VS corresponds to the vision sensor on the adversary while DS refers to the distance sensor on the robots.

detailed quantitative comparison of our work with the approach in [23, Sec. VI].

B. Vision-Based Target Tracking and Pursuit

The approach proposed in this article assumes an adversary with vision support. Vision-based target tracking has been well-explored in the past [24]. A target tracking strategy based on game theory has been proposed in [25].

Pursuit-evasion problems have also been explored with visibility constraints. A cell decomposition-based approach to visibility-based pursuit evasion is presented in [26]. An enhancement to [26] that considers visibility-based target tracking with a mobile observer is reported in [27]. Although visibility-based pursuit-evasion studies have been reported, a detailed geometric framework for determining rescue/capture, when the adversary is equipped only with a vision-based sensor (and has no explicit communication with the agents), does not appear to be available. Motivated by biological analogs, we study the effect (on coalition of agents) due to a (stronger) vision-guided adversary that acts independently. Furthermore, our study explores the scenario involving an arbitrary number of agents. In addition, we present experimental results.

III. Problem Description and Assumptions

We present two definitions before stating the problems addressed in this article.

Definition 1: For any two given agents, P_1 and P_2, travelling with speeds, v_1 and v_2, respectively, the **dominance region** for agent P_1 is defined as the set of points in the plane where agent P_1 reaches prior to P_2. The curve that separates the dominance regions of P_1 and P_2 is defined as the **dominance curve**.

Definition 2: The **safe region** is defined as the region where rescue of the delivery agent is feasible, prior to a capture by the adversary. This is denoted by S_h.

Problem 1: Given three autonomous vehicles, namely the delivery agent, rescue agent, and adversary (see Fig. 1), we develop a framework based on *dominance curves* for protecting the delivery agent against the adversary. Using the framework, we provide a characterization of the *safe region* for the delivery agent and identify the outcome (namely, rescue or capture) for given initial locations and speeds of the three vehicles.

In defense and other applications, it is common to have multiple delivery agents (one for supplying medicines, another for food, and so on). An opponent may be tasked with disrupting the movement of any one of these vehicles. Protecting each of the delivery agents is a challenge: Even if we deploy a rescue agent for each delivery agent, it is nontrivial to identify which rescue agent should take care of which delivery agent. The second problem can, therefore, be stated as follows.

Problem 2: Given multiple delivery agents and an equal number of rescue agents, determine the appropriate rescue agent for each of the delivery agents based on the risk of capture of each delivery agent by the adversary.

A simple case of Problem 2 with two delivery and two rescue agents is illustrated in Fig. 2. The assumptions and criteria used for solving the two problems are as follows.

1) The agents and the adversary are assumed to have discrete-time single integrator system dynamics.
2) All the vehicles (agents as well as the adversary) have distance sensors and position encoders mounted on them. Furthermore, a vision sensor is present on the adversary alone while markers are present only on the delivery agent(s). In addition, communication hardware is present only on the delivery and rescue agents. The sensors, markers, and communication hardware are used as described in items 3) and 4).
3) In the context of Problem 1, the distance sensor on the adversary allows it to detect the presence of the delivery agent/rescue agent in its vicinity. The adversary uses its vision sensor to distinguish a delivery agent (via its marker) from a rescue agent. Similarly, the distance sensor on a delivery agent allows it to detect the presence of an adversary/rescue agent in its neighborhood. To make a distinction between the adversary and the rescue agent, the delivery agent uses its wireless communication hardware and its position encoder information. The sensors and communication hardware on a rescue agent perform functions similar to the corresponding ones on the delivery agent.
4) In the context of Problem 2, communication exists between all the agents and is used as the means to distinguish the adversary from other agents (whenever the distance sensor on an agent indicates the presence of a vehicle nearby).

5) The adversary uses its distance and vision sensors to recognize rescue of a delivery agent. That is, when the distance sensor on the adversary detects two objects (and the vision sensor detects only one object with a marker) at (nearly) the same distance, the pursuit by the adversary is abandoned. Similarly, communication from the delivery agent (about capture) to the rescue agent will halt the rescue process.

IV. Protecting a Delivery Agent Against an Adversary

The geometric framework is developed by separately handling the following interactions: 1) delivery and rescue agents (see Section IV-A) and 2) delivery agent and adversary (see Section IV-B).

A. Delivery and Rescue Agent Interaction

Delivery and rescue agent interaction (denoted by D and R, respectively) is characterized by communication between the agents. Using this communication, the agents attempt to meet by determining a location (denoted by T) and proceeding to it along a straight line path. Location T can be computed by identifying the dominance region of the delivery agent, which, in turn, depends on the dominance curve of the two agents.

It is worthwhile to think of the dominance curve as the locus of all points in the plane where two agents arrive simultaneously. This notion is used in Lemma 1 to compute the dominance curve for the two agents. Note that the curve given by (1) is indeed an Apollonius circle constructed using the distance between the agents and the ratio of their speeds [23]. Our interest, however, is in computing the dominance region required for computing the meeting location (T) for the delivery agent (in Section IV-C).

Lemma 1: The dominance curve for the delivery agent [D, located at (x_d, y_d)] and rescue agent [R, located at (x_r, y_r)] moving with speeds v_d and v_r, respectively, is a circle (C_r) with center at (x_{dr}, y_{dr}) and radius r_{dr} as given by

$$(x_{dr}, y_{dr}) = \left(\frac{x_d v_r^2 - x_r v_d^2}{v_r^2 - v_d^2}, \frac{y_d v_r^2 - y_r v_d^2}{v_r^2 - v_d^2}\right)$$
$$r_{dr} = \sqrt{x_{dr}^2 + y_{dr}^2 - \frac{v_r^2(x_d^2 + y_d^2) - v_d^2(x_r^2 + y_r^2)}{v_r^2 - v_d^2}}. \tag{1}$$

Fig. 3 illustrates the dominance curve for a delivery agent D and a rescue agent R (depicted by a filled blue square and a green triangle, respectively). The curve corresponds to C_0 with center at N_0. Since the rescue agent is assumed to be moving faster than the delivery agent, the dominance curve encloses the delivery agent. Consequently, the region (in blue) contained by this curve (denoted by S_{dr}) is the dominance region of the delivery agent while the curve itself and the region (in green) external to it (denoted by $\neg S_{dr}$) is dominated by the rescue agent.

Additionally, Fig. 3 illustrates the dominance curve C_t constructed at intermediate locations of D and R (denoted by D_t and R_t) on their straight line path to T. It can be observed that C_t

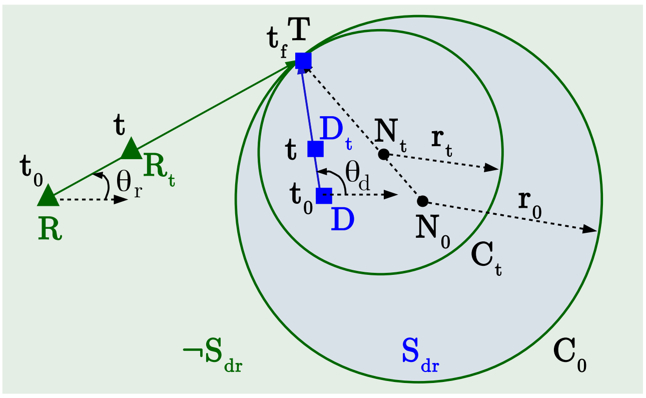

Fig. 3. Dominance curves (C_0, C_t) for initial locations (D, R) and intermediate locations (D_t, R_t) of delivery and rescue agents.

intersects (the previously computed) C_0 at T. Such a property of the dominance curves eliminates any need for recomputation of dominance regions as the agents travel to T. This is given by Theorem 1. It is based on [4].

Theorem 1: Let T be the meeting point for two agents located on the dominance curve computed with their initial locations and speeds. Subsequent dominance curves constructed with locations of agents as they move toward T remain tangential at T.

Proof: Let initial locations and speeds of delivery and rescue agents be $\{D(x_d, y_d), v_d\}$ and $\{R(x_r, y_r), v_r\}$, respectively. The center $N_0(x_0, y_0)$ and radius r_0 of the dominance curve C_0 for these initial locations can be obtained using (1). Consider any arbitrary point $T(x, y)$ on the circle C_0. The intermediate locations of D and R after time t (on their way to T) are given by $D_t(x_{dt}, y_{dt})$ and $R_t(x_{rt}, y_{rt})$ in

$$D_t(x_{dt}, y_{dt}) = (x_d + t v_d \cos(\theta_d), y_d + t v_d \sin(\theta_d))$$
$$R_t(x_{rt}, y_{rt}) = (x_r + t v_r \cos(\theta_r), y_r + t v_r \sin(\theta_r)) \tag{2}$$

where θ_d, θ_r are the heading angles from D, R to T

$$\overline{N_0 N_t} + \overline{N_t T} = \overline{N_0 N_t} + r_t = \overline{N_0 T} = r_0. \tag{3}$$

Dominance curve C_t can now be recomputed using the updated locations D_t and R_t as a circle centered at $N_t(x_t, y_t)$ with a radius of r_t using (1). It follows that the two centers (N_0 and N_t) and T are collinear as given by (3). Thus, the two circles C_t and C_r meet at T. In other words, subsequent dominance curves constructed using intermediate locations of the two agents (after a finite time), contain the point T and, thus, do not affect the position of meeting location. ■

We now examine a scenario where the rescue is not characterized by the two agents occupying exactly the same location T. Instead, we permit the rescue agent to merely come in close proximity to the delivery agent. We quantify this proximity via the notion of a *limiting distance* given by Definition 3.

Definition 3: The maximum distance between delivery and rescue agent, below which the delivery agent is considered to have been rescued is defined as the **limiting distance** of the rescue agent. This is denoted by d_r.

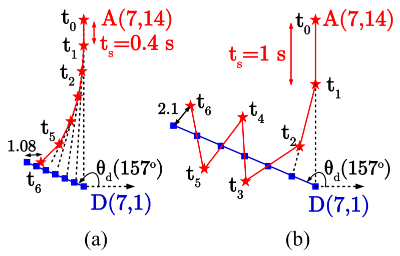

Fig. 4. Adversary's $(A(7,14))$ vision-based pursuit of delivery agent $(D(7,1))$ heading in a fixed direction $\theta_d = 157^\circ$. $v_a = 5$ m/s, $v_d = 2$ m/s. (a) $t_s = 0.4$ s. (b) $t_s = 1.0$ s.

The dominance curve is now given as the locus of points reached by the rescue agent where the separation between the two agents is d_r. The dominance curve, thus formed, turns out to be an oval, symmetric about the line joining the two agent locations, as given by

$$\frac{PD}{v_d} = \frac{PR - d_r}{v_r}. \tag{4}$$

B. Delivery Agent and Adversary Interaction

The adversary A uses its vision sensor to identify the delivery agent and begins its pursuit. It is worth noting that the direction of delivery agent (and not its position) is sufficient for pursuit. Since the adversary relies solely on the vision sensor (and not on communication), the path taken by the adversary is not necessarily a straight line. In other words, the adversary continuously reorients itself toward the delivery agent while pursuing it with a speed of v_a. These reorientations are separated by a finite time interval to facilitate processing of information from the vision sensor and to enforce the change in direction in a physical system. We define this time interval as the *sampling time* and denote it by t_s.

Remark 1: The sampling time is a notion applied only to the adversary. The numerical values of sampling time determine how often the adversary has to reorient itself. The delivery and rescue agents move purely based on communication among themselves. They take into account the best response of the adversary and, thus, need not perform reorientation (when the adversary changes direction).

The adversary first scans its accessible environment via its vision sensor and determines the direction in which the delivery agent is located. While the adversary pursues the delivery agent in the computed direction, the latter proceeds toward a rescue agent to avoid capture. This affects the relative position of the delivery agent in the image captured by the vision sensor. The adversary, then, makes the necessary correction to its heading direction and continues its pursuit until another such reorientation becomes necessary or a capture/rescue occurs.

Fig. 4 (with distances measured in meters and time in seconds) illustrates the effect of sampling time on the capture of delivery agent. In Fig. 4(a), the path taken by the adversary A (red star) in pursuit of the delivery agent D (blue square) that is heading along a fixed direction (θ_d) is shown. Reorientations at fixed intervals of $t_s = 0.4$ s can be observed in the figure for six time instants. At this point, the distance between the agent and the adversary is 1.08 m. Fig. 4(b) illustrates the same scenario for a higher sampling time, namely $t_s = 1.0$ s. As a consequence, after two time instants, the adversary repeatedly overshoots the location of the delivery agent.

Algorithm 1: Dominance Curve C_a Between A and D.

```
   Input: $(A(x_{a_0}, y_{a_0}), v_a), (D(x_{d_0}, y_{d_0}), v_d), t_s$ and $d_a$
   Output: $C_a$
1  Initialize $D_c \leftarrow \emptyset$
2  for $\theta \leftarrow 0$ to $2\pi$ do
3      $x_a \leftarrow x_{a_0}$; $y_a \leftarrow y_{a_0}$; $x_d \leftarrow x_{d_0}$; $y_d \leftarrow y_{d_0}$
4      $d \leftarrow \overline{AD}$
5      while $d > d_a$ do
6          $\phi \leftarrow \arctan(\frac{y_d - y_a}{x_d - x_a})$
7          $A(x_a, y_a) \leftarrow (x_a + t_s v_a \cos\phi, y_a + t_s v_a \sin\phi)$
8          $D(x_d, y_d) \leftarrow (x_d + t_s v_d \cos\theta, y_d + t_s v_d \sin\theta)$
9          $d \leftarrow \overline{AD}$
10     end
11     Update $D_c \leftarrow D_c \cup D(x_d, y_d)$
12 end
13 return as $C_a$, the curve obtained by linear interpolation
     of locations in $D_c$
```

The heading angle ϕ of the adversary can be computed using the positions of $A(x_a, y_a)$ and $D(x_d, y_d)$ as given by (5). Given the heading direction θ of the delivery agent, the locations of A and D, after one time instant t_s, can then be computed using (6). The path taken by A and D, for a given θ (θ_d in Fig. 4), is obtained by computing their locations using (6) and, then, utilizing these as the current locations during the next time instant

$$\phi = \arctan\left(\frac{y_d - y_a}{x_d - x_a}\right) \tag{5}$$

$$A = (x_a + t_s v_a \cos\phi, y_a + t_s v_a \sin\phi)$$
$$D = (x_d + t_s v_d \cos\theta, y_d + t_s v_d \sin\theta). \tag{6}$$

In order to define the capture of delivery agents, we revisit the notion of limiting distance. A delivery agent is considered to be captured if its distance from adversary is less than the limiting distance (denoted by d_a) of the adversary. For instance, in Fig. 4(a), if the limiting distance is 1.5 m ($d_a = 1.5$), the delivery agent is considered to be captured after six time instants (since $1.08 < d_a$). For the same value of d_a in Fig. 4(b), the closest an adversary can get to D is 2.1 m, which indicates that a capture is never possible ($2.1 > d_a$).

The dominance curve for the delivery agent (D) and adversary (A) interaction (denoted by C_a), corresponds to the locus of all points where the distance between adversary and delivery agent (denoted by $\overline{AD}$) is less than the limiting distance (indicating capture). Given initial locations ($A(x_{a_0}, y_{a_0})$, $D(x_{d_0}, y_{d_0})$) and speeds (v_a, v_d) of the two entities, Algorithm 1 details the procedure for obtaining C_a. This involves computing the curve via linear interpolation (line 13) of all the capture locations of D (stored in D_c) corresponding to all possible heading directions ($\theta \in [0, 2\pi)$) of the delivery agent (lines $2-11$). In practice, θ

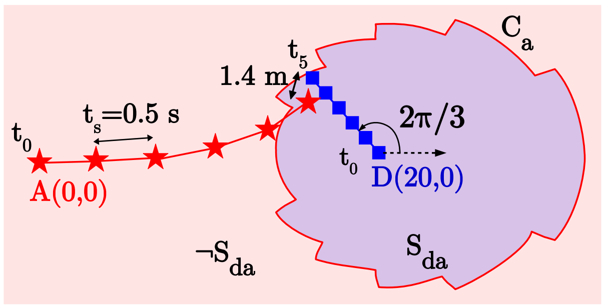

Fig. 5. Dominance curve C_a and dominance regions $(S_{da}, \neg S_{da})$ of the delivery agent and the adversary. $v_a = 3$ m/s, $v_d = 1$ m/s, $d_a = 2.5$ m.

is incremented by a fixed value that represents the minimum turning angle of the delivery agent.

Fig. 5 illustrates Algorithm 1 for two distinct locations of the delivery agent and the adversary. One instance of pursuit with heading angle $\theta = 2\pi/3$ is shown in the figure where the capture occurs after five time instants. The dominance curve, C_a, once again encloses the delivery agent, since its speed is lower than that of the adversary. The dominance region (in blue) of D, denoted by S_{da}, is the region interior to C_a, while the curve itself and the region (in red) external to it is dominated by the adversary (denoted by $\neg S_{da}$).

Note that the point of capture for a given heading angle of the delivery agent remains unchanged with subsequent recomputations of the dominance curve. Consequently, the analysis performed on initial locations of agents and adversary continue to hold until capture or rescue.

From the point of view of the delivery agent, higher values of the sampling time (for the adversary) are desirable to avoid a capture. However, the adversary attempts to quickly identify the delivery agent and reorient itself. This corresponds to a small value of the sampling time. In terms of implementation, this involves computing the circular Hough transform (CHT) based on the adversary's vision sensor output. Further details are presented in Section VII.

C. Safe Regions and Capture-Rescue Algorithms

We are now equipped with the dominance regions of the delivery agent 1) with respect to the rescue agent (S_{dr}) and 2) with respect to the adversary (S_{da}).

The algorithm adopted by the rescue agent involves computing the safe region (given by Definition 2). In other words, the set of points where the rescue agent reaches prior to or along with the delivery agent are valuable and need to be determined. However, it is to be ensured that there is no capture by adversary at these points. This observation allows us to derive a relation between the two dominance regions and the safe region. This is established via Theorem 2.

Theorem 2: Given the dominance regions of delivery agent with respect to adversary (S_{da}) and rescue agent (S_{dr}), safe region (S_h) can be computed using

$$S_h = S_{da} \setminus S_{dr}. \tag{7}$$

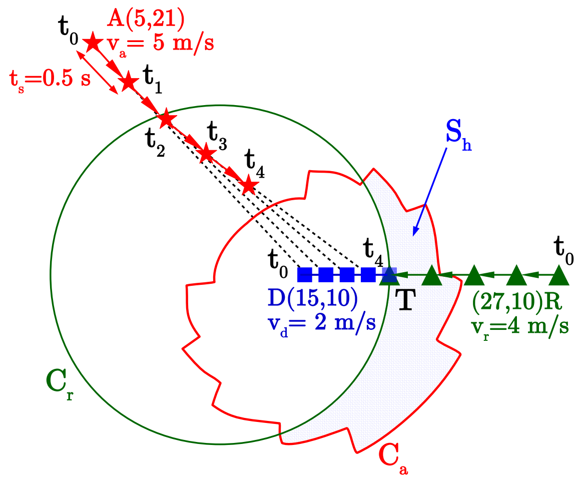

Fig. 6. Computation of safe region S_h and a successful rescue attempt.

Proof: It follows from Definition 1 that the interior of the dominance region is dominated by the entity contained in it. Since the delivery agent is the slowest, its initial location is contained in both the dominance regions. Consequently, any point inside S_{da} can be reached by the delivery agent before the adversary. Similarly, any point external to S_{dr} can be used for rescue by R. The intersection of these two regions determines the safe region as given by

$$\begin{aligned} S_h &= S_{da} \cap \neg S_{dr} \\ \Rightarrow S_h &= S_{da} \setminus S_{dr}. \end{aligned} \tag{8}$$

■

Fig. 6 illustrates computation of safe region using Theorem 2. The dominance curves C_a (shown in red) and C_r (shown in green) are computed using Algorithm 1 and (1), respectively. The corresponding dominance regions are, then, used to compute the safe region S_h (shaded in blue). It is worth noting that a part of the boundary curve C_r is included in the safe region while the entire curve C_a is excluded from it.

Before we proceed to discuss how the rescue agent operates, we will describe the algorithm adopted by the adversary. The adversary is guided purely by its vision sensor (and it does not require knowledge of the safe regions). The pursuit by the adversary is expressed via Algorithm 2. The vision sensor helps the adversary to distinguish the delivery agent from the rescue agent (line 2). The distance sensor is, then, used to compute the proximity of the delivery agent to the adversary and the rescue agent (denoted by d_1 and d_2, respectively, line 11). The pursuit (lines 5–10) continues until either capture ($d_1 \leq d_a$) or rescue ($d_2 \leq d_r$) occurs, where d_a and d_r represent the limiting distances of A and R.

The algorithm additionally handles occlusion of vision of the adversary by performing an evasive maneuver to regain its "line of sight" (lines 6–7). The higher speed of adversary becomes particularly useful in this scenario. We illustrate such a maneuver via experiments in Section VII.

We now proceed to the algorithm adopted by the rescue agent (described via Algorithm 3), which handles the tasks of

Algorithm 2: Adversary's Vision-Guided Pursuit.

```
Input: Data from vision sensor and distance sensor. The
       constants $v_a$, $d_a$, $d_r$ and $t_s$
Output: Capture or rescue of $D$
1  while $D$ is not yet detected do
2      Halt $A$ and use its vision sensor to scan for $D$
3  end
4  repeat
5      $\phi \leftarrow$ Heading direction from $A$ to $D$
6      if vision to $D$ is obstructed then
7          Translate $A$ in the direction $(\phi + \pi/2)$ with speed
           $v_a$ for time $t_s$
8      else
9          Translate $A$ in the direction $\phi$ with speed $v_a$ for
           time $t_s$
10     end
11     $d_1 \leftarrow \overline{DA}$; $d_2 \leftarrow \overline{DR}$
12 until $d_1 \leq d_a$ or $d_2 \leq d_r$
13 if $d_1 \leq d_a$ then
14     Halt $A$ and return $D$ is captured by $A$
15 else
16     Halt $A$ and return $D$ is rescued by $R$
17 end
```

Algorithm 3: Attempt to Rescue D by R.

```
Input: $(A(x_a, y_a), v_a, d_a), (R(x_r, y_r), v_r, d_r), t_s, s$ and
       $(D(x_d, y_d), v_d)$. Data from distance sensor.
Output: Capture or rescue of $D$
1  Compute $C_r$ using (4) and obtain $S_{dr}$
2  Compute $C_a$ using Algorithm 1 and obtain $S_{da}$
3  $S_h \leftarrow S_{da} \setminus S_{dr}$
4  if $S_h \neq \emptyset$ then
5      $T \leftarrow$ the point in $S_h$ that is closest to $D$
6  else
7      $T \leftarrow (v_d \times R + v_r \times D)/(v_d + v_r)$
8  end
9  Communicate $T$ to $D$ and reorient $R$ towards $T$
10 while $s = 0$ and $A$ does not obstruct $R$ do
11     Translate $R$ with speed $v_r$ until it reaches $T$
12 end
13 if $\overline{DR} \leq d_r$ then
14     Halt $R$ and return $D$ is rescued by $R$
15 else
16     Halt $R$ and return $D$ is captured by $A$
17 end
```

computing safe region S_h and determining a meeting location T. In order to minimize the distance travelled by the delivery agent prior to rescue, T is chosen as the closest point in S_h from D (line 5). However, in the absence of a safe region, capture is inevitable. In such a scenario, it is advantageous to minimize the distance between the delivery and rescue agents. Hence, T is chosen to be the point of intersection of the line segment joining the two agents and the dominance curve C_r (line 7). The algorithm also handles the case where the adversary interrupts the path of the rescue agent during its pursuit. Due to the higher speed of A, the capture of D is, then, inevitable (lines 10 and 16).

Algorithm 4: Evasive Approach of Delivery Agent D.

```
Input: $D(x_{d_0}, y_{d_0}), v_d, d_a, d_r$. Data from distance sensor.
Output: Capture or rescue of $D$
1  while Adversary is not yet detected do
2      Travel along the predetermined path for delivery
3  end
4  $\phi \leftarrow$ Heading direction from $D$ to $A$
5  $d \leftarrow$ Distance to $A$ via distance sensor
6  Update $D(x_d, y_d) \leftarrow$ current location of $D$
7  Update $A(x_a, y_a) \leftarrow (x_d + d\cos\phi, y_d + d\sin\phi)$
8  Initialize $s \leftarrow 0$ and transmit $A, D, s$ to $R$
9  Receive $T$ from $R$ and reorient $D$ towards $T$
10 repeat
11     Translate $D$ with speed $v_d$ and monitor:
12     $d_1 \leftarrow \overline{DA}$; $d_2 \leftarrow \overline{DR}$
13 until $d_1 \leq d_a$ or $d_2 \leq d_r$
14 if $d_1 \leq d_a$ then
15     Update $s \leftarrow 1$ and transmit $s$ to $R$
16     Halt $D$ and return $D$ is captured by $A$
17 else
18     Halt $D$ and return $D$ is rescued by $R$
19 end
```

We now describe the algorithm (Algorithm 4) running on the delivery agent. This consists of detection of an adversary by the delivery agent followed by transmission of its (D) location as well as that of A to R (line 8). Furthermore, a request for rescue is communicated by D via a status variable s where $s = 0$ prior to capture and $s = 1$ after capture (lines 8 and 15). Once the meeting location T is computed and transmitted by R, the delivery agent tries to evade the adversary by translating to T unless captured (lines 10–19).

Fig. 6 illustrates a pursuit (by A) and rescue (by R) scenario. The adversary is governed by Algorithm 2 and its pursuit continues for four time instants. The termination is caused by lines 12 and 16 of the algorithm when the adversary discovers that the delivery agent has been rescued ($d_2 \leq d_r$). The computation of safe region and the meeting location (T) are carried out by the rescue agent with the help of Algorithm 3. Algorithm 4 assists the delivery agent in arriving at T where it is successfully rescued.

V. Extension to Multiple Delivery Agents

In this section, we address Problem 2 (from Section III) that generalizes the pursuit to k (where $k > 1, k \in \mathbb{Z}^+$) delivery agents. We assume an equal number of rescue agents. The outcome of the interaction is considered to be capture if at least one of the delivery agents is captured by the adversary. On the other hand, the outcome is rescue when every delivery agent is rescued by an appropriate rescue agent. As before, the adversary adopts a vision-guided strategy to identify the *weakest* delivery agent and pursues it until capture or rescue. The strength of a delivery agent is determined by its distance to the adversary which leads to Lemma 2.

Lemma 2: The dominant strategy for the adversary while pursuing multiple, identical delivery agents is to choose and

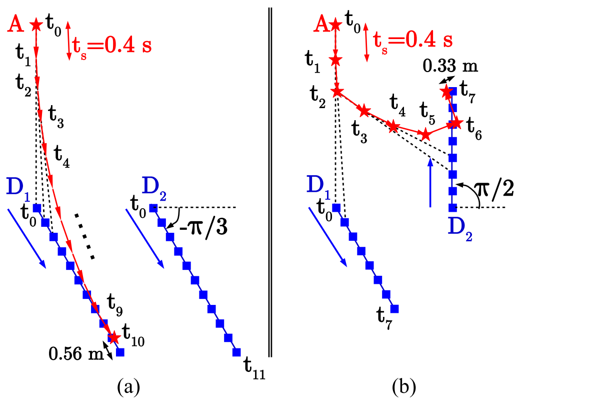

Fig. 7. Adversary's pursuit. (a) D_1 is always closer to A than D_2. Capture at t_{10}. (b) D_2 is closer to A than D_1 after $t = t_2$. Capture at t_7.

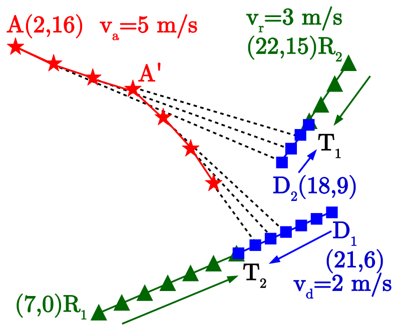

Fig. 8. D_1 and D_2 rescued by R_1 and R_2 while pursued by A.

pursue that agent which is closest to the adversary's current location.

Proof: Since the adversary adopts a vision-based pursuit, the time taken for capture is directly proportional to the distance between the adversary and delivery agent. Given that all the delivery agents travel at the same speed, the adversary pursues the delivery agent closest to its initial location. During the pursuit of a delivery agent (say D_i), if another delivery agent (say D_j, where $i, j \in \mathbb{Z}^+$) comes closer to the current location of the adversary, the time taken to capture D_j is less than the time taken to capture D_i. Thus, the adversary switches its pursuit to D_j, indicating that the adversary always pursues the delivery agent that is closest to its current location. ■

Fig. 7 illustrates Lemma 2 for two delivery agents (D_1 and D_2) and an adversary (A). In Fig. 7(a), the two agents travel in the same direction (at an angle of $-\pi/3$ to the horizontal) ensuring that D_1 remains closer to A compared to D_2 until its capture after ten time instants. However, in Fig. 7(b), the heading angles of the two delivery agents are -$\pi/3$ and $\pi/2$, respectively. It can be observed that after two time instants, D_2 is closer (than D_1) to A. The adversary switches its pursuit to D_2 and eventually captures it after seven time instants, three time instants sooner than the previous example, illustrating the ideas in Lemma 2.

We now present Algorithm 5, adopted by the adversary when there are multiple delivery agents. In this case, the adversary uses its data from vision as well as distance sensors to identify the k delivery agents, where the subscripts i and j represent an individual agent. l_j denotes the distance from A to D_j where

Algorithm 5: Pursuit of Multiple Delivery Agents by A.

```
Input: Data from vision sensor and distance sensor. The
       constants $v_a$, $d_a$, $d_r$ and $t_s$
Output: Rescue of all delivery agents or a capture
1  while no delivery agent is yet detected do
2      Halt $A$ and use its vision sensor to scan for any $D_j$
3  end
4  repeat
5      Initialize $l_j$ such that $l_j \leftarrow \infty$ if $D_j$ is rescued
6                while $l_j \leftarrow \overline{AD_j}$ otherwise
7      while $D_i$ is the closest delivery agent do
8          Execute Lines 4-12 of Algorithm 2 for $D_i$
9      end
10     if $d_1 \leq d_a$ then
11         Halt $A$ and return $A$ has captured $D_i$
12     else
13         Update $l_i \leftarrow \infty$
14     end
15 until Every $D_i$ is rescued or a capture occurs
16 Halt $A$ and return All delivery agents are rescued
```

$l_j = \infty$ if D_j is already rescued and $l_j = \overline{AD_j}$ otherwise (lines 5–6). Identification of the closest nonrescued delivery agent D_i begins the adversary's pursuit. This process continues until D_i is either captured, rescued or another agent D_j comes closer to A (lines 7–9). Fig. 8 presents an illustration where the adversary switches its pursuit from D_2 to D_1 since the former is rescued after three time instants.

Having determined the dominant strategy adopted by the adversary, we are now in a position to compute safe regions for various delivery agents and obtain their respective meeting locations for rescue. To this end, we begin with Lemma 3 that determines the safe region S_h for one delivery agent amidst multiple (k) rescue agents.

Lemma 3: Given the dominance regions of delivery agent D with respect to the k rescue agents $\{R_1, R_2, \ldots, R_k\}$ as $\{S^1_{dr}, S^2_{dr}, \ldots, S^k_{dr}\}$ and with respect to adversary (A) as (S_{da}), safe region (S_h) is computed using

$$S_h = S_{da} \setminus \left(\bigcap_{i=1}^{k} S^i_{dr} \right). \tag{9}$$

Proof: It follows from Definition 1 that the dominance regions of the delivery agent with respect to multiple rescue agents are independent of each other. Furthermore, the safe region corresponds to a zone where the delivery agent is rescued by at least one rescue agent. We, thus, have

$$S_h = \bigcup_{i=1}^{k} (S_{da} \cap \neg S^i_{dr}). \tag{10}$$

Applying distributive law to (10), we have

$$S_h = S_{da} \cap \left(\bigcup_{i=1}^{k} (\neg S^i_{dr}) \right). \tag{11}$$

Applying De-Morgan's law to (11), we have (9). ■

Having described the algorithm adopted by the adversary, we now turn our attention to the algorithm adopted by the k rescue

Algorithm 6: Attempt to Rescue all the Delivery Agents.

```
Input: $L^D = \{D_1, D_2, \cdots D_k\}, A, v_d, v_r, v_a, t_s$,
       $L^R = \{R_1, R_2, \cdots R_k\}$. Data from distance
       sensors.
Output: Rescue of all delivery agents or a capture
1  Initialize a list $T^D$ of size $k$
2  for $i \leftarrow 1$ to $k$ do
3      Define $L^{R'} = \{L_i^R, L_{i+1}^R, \cdots, L_k^R\}$
4      Compute $S_h$ for $L_i^D$ and $L^{R'}$ using Lemma 3
5      if $S_h \neq \emptyset$ then
6          $T \leftarrow$ the point in $S_h$ that is closest to $L_i^D$
7          $j \leftarrow$ index of agent in $L^{R'}$ closest to $T$
8      else
9          $j \leftarrow$ index of agent in $L^{R'}$ closest to $L_i^D$
10         $T \leftarrow (v_d \times L_j^R + v_r \times L_i^D)/(v_d + v_r)$
11     end
12     $T_i^D \leftarrow T$; Swap $L_i^R$ and $L_j^R$
13 end
14 Communicate $T^D$ to all agents
15 for $i \leftarrow 1$ to $k$ do
16     repeat
17         Translate $L_i^D$ with speed $v_d$ to $T_i^D$
18         Translate $L_i^R$ with speed $v_r$ to $T_i^D$
19     until $L_i^D$ is captured or rescued or obstructed
20     if $L_i^D$ is obstructed then
21         Transmit the current index number and receive
             the index number of obstructing agent
22         Halt the agent with higher index number
23         Goto 16
24     else
25         Halt $L_i^D$ and $L_i^R$
26     end
27 end
28 if $L_i^D$ is captured then
29     return A has captured $L_i^D$
30 else
31     return All delivery agents are rescued
32 end
```

agents (as well as the k delivery agents). Lemma 3 assists in developing such an algorithm (see Algorithm 6). The list L^D contains the delivery agents arranged in the increasing order of their distances from A. L^R is the list with all the rescue agents, which is, then, rearranged such that the delivery agent L_i^D is rescued by L_i^R at T_i^D where $i \in \{1, 2, \ldots, k\}$. For a given delivery agent L_i^D, the safe region S_h is computed via Lemma 3 (lines 3–4 of Algorithm 6) and the closest point in S_h (to L_i^D) is determined as its meeting location T_i^D (line 6). The rescue agent closest to T_i^D is assigned to rescue L_i^D (line 7) and the process is repeated for all the remaining delivery agents in the same order as L^D (lines 2–13). In the absence of a safe region, however, L_i^D is assigned a rescue agent that is closest to its location (lines 9–10). The advantage of choosing an agent closer to T_i^D over an agent closer to L_i^D (when S_h exists) is illustrated with the help of an experiment in Section VII (see Fig. 13).

Remark 2: Algorithm 6 combines the strategies adopted by the k delivery agents as well as the k rescue agents. In particular, while the computation of T is handled by the rescue agents, the delivery agents are responsible for triggering the attempt at rescue when an adversary is detected. The algorithm also handles the transfer of the delivery and rescue agents to their respective meeting locations.

Fig. 8 illustrates the rescue of two delivery agents D_1 and D_2 by two rescue agents R_1 and R_2. It can be observed that the agent D_2 is closer to the adversary resulting in $L^D = \{D_2, D_1\}$. The meeting location T_1 is first computed and the closest rescue agent R_2 is assigned to rescue D_2. The process is repeated for the second delivery and rescue agents. We, thus, have $L^R = \{R_2, R_1\}$ and $T^D = \{T_1, T_2\}$. In this scenario, safe regions exist for both the delivery agents and the meeting location is chosen from these regions. Thus, a rescue of D_1 and D_2 is ensured. Furthermore, the outcome of the interaction (in Fig. 8) will remain unaffected even if the adversary pursues D_1 before D_2.

The communication of the computed meeting locations followed by the transfer of the agents to those locations (while keeping track of the distance from A) is handled via lines 14–27 of Algorithm 6. When the paths of two or more agents are obstructed by fellow agents, the agent with a lower index number is permitted to proceed further (lines 20–23).

In order to avoid multiple computations of meeting locations, we assume that one rescue agent receives the information pertaining to various locations of the agents and the adversary and, then, uses Algorithm 6 to compute the list T^D. These meeting locations and the modified order for rescue agents L^R are, then, communicated back to all agents. The transfer of agents from their locations to their respective rescue locations is carried out by the individual agents (in parallel) until either capture or rescue takes place. Some comments on robot assignments are presented in Remark 3.

Remark 3: The assignment of rescue agents to delivery agents proposed here can be compared with the Hungarian method for multirobot task assignment [28], [29]. The Hungarian method corresponds to assigning rescue agents to delivery agents such that the total time for rescue is minimized. The Hungarian method is unbiased, however, it may not necessarily lead to rescue of multiple delivery agents. On the other hand, the proposed scheme (based on Lemma 3), places emphasis on rescue of the delivery agent with the highest risk of capture and, therefore, does better in general (with respect to the protection of the delivery agents). Examples illustrating this can be readily constructed.

VI. Simulations and Comparisons

In this section, we compare the proposed approach with the work reported in [23] via simulations. The work reported in [23] is chosen for comparison since it also involves three agents (termed predator, prey, and protector) and further 1) it has similar objectives (with an outcome of rescue/capture) and 2) has a similar flavor in terms of its approach (it pursues a geometric approach based on Apollonius circles).

We examine the effectiveness of the specific (vision) sensor on the adversary for the capture task and compare with the complete communication-based strategy in [23]. In particular, Oyler *et al.* [23] assumed communication between the agents and

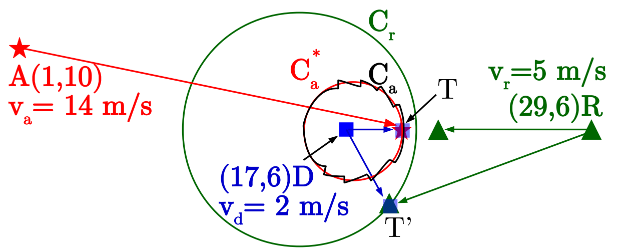

Fig. 9. Dominance curves (C_a, C_r and C_a^*) computed using Algorithm 1, (1), and [23], respectively.

TABLE I
NUMERICAL VALUE OF G IN THE PRESENCE/ABSENCE OF COMMUNICATION BETWEEN AN ADVERSARY AND THE AGENTS

	Fig. 9		Fig. 6	
	[23]	Proposed	[23]	Proposed
With communication	2.4	2.4	-7.8	-7.9
Without communication	-11.4	2.4	-16.0	-7.9

the adversary. In other words, the meeting location of the rescue and delivery agents (in addition to their current locations) is communicated to the adversary and the latter takes a straight line path to reach the meeting location. As a result, the dominance curves for both DA and DR interactions (denoted by C_a^* and C_r, respectively) result in circles, as given by (1). Fig. 9 illustrates a scenario where C_a^* (red circle) is completely enclosed in C_r (green circle) indicating definitive capture (one instance of such a capture at T is shown in Fig. 9).

In practice, however, the meeting location of rescue and delivery agents may not be known to the adversary. This can be easily exploited by the rescue agent that accomplishes successful rescue at an alternate location T', as shown in Fig. 9. The capabilities of the adversary, thus, play a major role in designing the strategy for rescue of the delivery agent. In order to facilitate quantitative comparisons of our work with the one in [23], we now introduce a distance measure, G, defined via

$$G = \begin{cases} \overline{DR}, & \text{at capture} \\ -\overline{DA}, & \text{at rescue} \end{cases}. \tag{12}$$

In (12), $\overline{DR}$ denotes the distance between D and R when a capture occurs while $\overline{DA}$ corresponds to the distance between D and A in the event of a rescue.

For a given scenario, analyzed with two adversaries adopting different strategies (vision-based or communication-based), a smaller value of G indicates a weaker adversary. Since it is desirable to study rescue strategies for a stronger adversary, the strategy (for adversary) that involves the least change in the value of G needs to be considered. Table I presents the numerical values of G in the presence and absence of communication when an adversary changes its strategy from communication (studied in [23]) to a vision-based approach (proposed scheme).

For the scenario in Fig. 9, G drops from 2.4 to -11.4, indicating a weaker adversary when a communication-based strategy (based on [23]) is adopted. Vision-based pursuit, however, strengthens the adversary removing any dependency on communication with the agents. This can be observed from Table I where G remains constant at 2.4 for the proposed scheme.

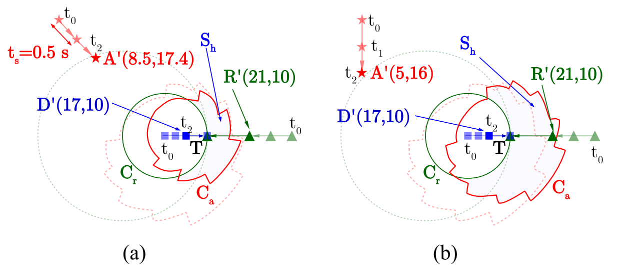

Fig. 10. Recomputation of T at successive time instants is not necessary since R incorporates the best response of A at t_0 itself. $v_a = 5$ m/s, $v_r = 4$ m/s and $v_d = 2$ m/s. (a) S_h is recomputed after two time instants. T remains unaffected. (b) A abandons its vision-based pursuit. T is still unaffected.

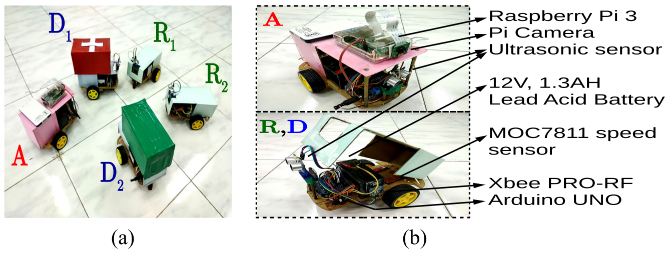

Fig. 11. Setup involving two rescue agents (R_1, R_2), two delivery agents (D_1, D_2) and one adversary (A).

Table I also presents the numerical values of G for the scenario in Fig. 6. While the straight line path adopted by the adversary using [23] gives a slightly higher value of G in the presence of communication, the adverse effect on G in the absence of communication can be observed where it drops to -16. On the other hand, with the scheme presented in this article, the value of G remains unaffected.

An additional advantage of utilizing a vision-based pursuit is that successive recomputation of dominance regions (and meeting locations) is not necessary. This is illustrated in Fig. 10(a) (for the scenario in Fig. 6) where C_a, S_h, and T are recomputed after two time instants. Note that the meeting location T obtained via Algorithm 3 remains unaffected even after two time instants. Furthermore, if the adversary deviates from its optimal strategy [discussed via Fig. 10(a)] and proceeds toward location A' [as shown in Fig. 10(b)], the recomputed meeting location T not only remains the same but the safe region considerably increases in area, giving additional advantage to rescue and delivery agents. This follows from the fact that the rescue agent utilizes the best response of adversary when computing the meeting location T.

VII. EXPERIMENTAL RESULTS

Multiple small differential drive mobile robots, fabricated locally are employed as agents and adversary [see Fig. 11(a)]. The robots measure 24 × 15 cm in area and are powered with a standard 12 V, 1.3 AH Lead acid battery. Two identical dc motors are used to drive the wheels while encoders (MOC7811

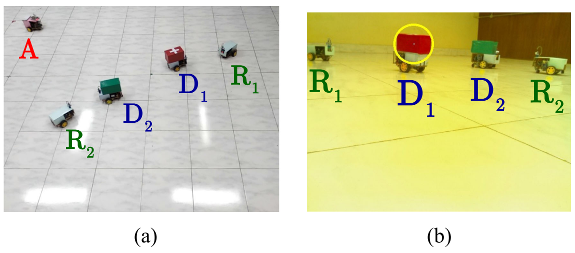

Fig. 12. Mobile robot platform showing various aspects of implementation.

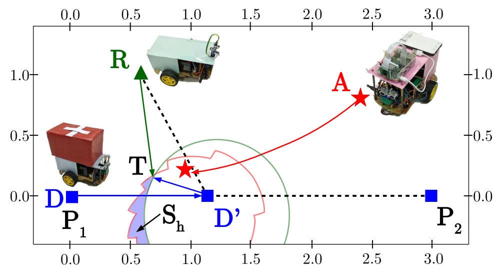

Fig. 13. Rescue of delivery agent in its safe region when the adversary and agents are located at $A(2.4, 0.8)$, $R(0.6, 1)$, and $D'(1.1, 0)$. Coordinates are in meters.

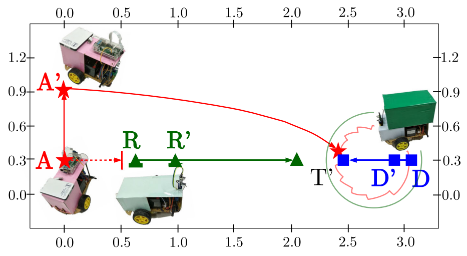

Fig. 14. Adversary handles occlusion of vision when $R(0.6, 0.3)$ is in between $A(0, 0.3)$ and $D(3.1, 0.3)$. Coordinates are in meters.

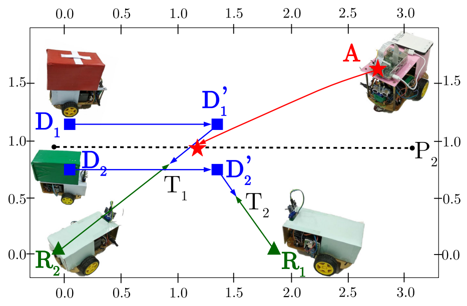

Fig. 15. Rescue of two delivery agents; Coordinates are in meters.

speed sensors) placed on each wheel are used to monitor the wheel rotations. The updation of current location is performed independently on each agent. Processing support is provided via an Arduino UNO board with an ATmega328P microcontroller. The agents as well as the adversary are equipped with ultrasonic sensors. The communication between the delivery and rescue agents is via Zigbee mesh networking protocol (and achieved through Xbee PRO RF modules). Various components utilized are shown in Fig. 11(b).

In terms of weights and speeds, the robots weigh 2.5 kg (each) with the electronics onboard. The payload carried by each delivery agent has a total weight of 5 kg (each). The maximum speed achieved by the delivery agents is approximately 0.22 m/s while the rescue agents and adversaries travel at 0.38 and 0.78 m/s, respectively.

The primary difference between the adversary and the remaining agents, in terms of the setup (hardware), is that the former employs a camera for detecting the agents while the latter utilize communication. The adversary is equipped with a Raspberry Pi 3 board with an integrated Pi camera (with only two-dimensional vision support). The red and green boxes on the delivery agents indicate their cargo. The adversary exploits these indicators by isolating the colors in the captured image and performing a CHT. Fig. 12(a) gives an overview of locations of the adversary A and the agents while Fig. 12(b) illustrates the agents as viewed by the adversary with its Pi camera. The circle computed around the closest delivery agent, using CHT, is illustrated in yellow.

The Raspberry Pi 3 on the adversary removes any dependence on a PC for processing the image acquired. Once the image is processed and the heading direction acquired, the translation of the adversary is handled completely by the microcontroller. Similarly, for the delivery and the rescue agents, the microcontroller alone implements the algorithms discussed. Thus, the experiments discussed do not require any support from a processor with external memory for implementing the proposed algorithms.

A. Summary of Experiments

Fig. 13 depicts the first experiment where a delivery agent (D) is interrupted on its path by an adversary (A). Delivery agent detects the adversary with the help of its distance sensor and communicates the same to the rescue agent. Adversary begins its pursuit as its vision sensor identifies the delivery agent at D'. In this scenario, the delivery agent is successfully rescued at T before a capture by the adversary.

In a scenario where the vision of the adversary is obstructed by one of the agents, the adversary adopts an evasive maneuver to restore its vision. Fig. 14 illustrates this experiment. Here, the higher speed of adversary allows it to capture the delivery agent although it was initially obstructed by a rescue agent.

Fig. 15 illustrates the pursuit of adversary amidst multiple agents. Ultrasonic sensors are employed to determine the closest delivery agent. The delivery agents are detected by the adversary (and vice-versa) as they arrive at D_1' and D_2'. Rescue agent R_2 employs Algorithm 6 to identify the weaker delivery agent as D_1 and assigns itself for rescue. It, then, computes (and communicates) the meeting location T_2 for R_1 and D_2. The adversary pursues until both delivery agents are rescued.

Fig. 16 presents a scenario where, *at the point of detection,* the adversary is between two delivery agents (located at D_1' and D_2'). Due to their respective positions, safe regions do not exist for either of the delivery agents. As a result, the rescue and delivery agents attempt to meet on the line segment joining their

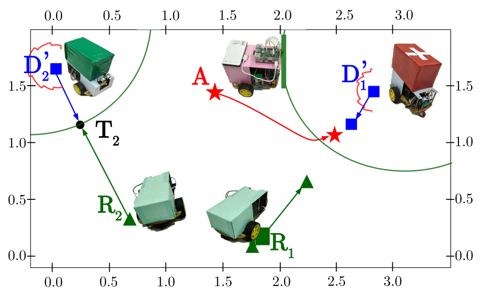

Fig. 16. Adversary between two delivery agents leading to capture of D_1; R_2 successfully rescues D_2; Coordinates are in meters.

locations. The rescue of D_1 is interrupted by A (via capture) while D_2 is successfully rescued by R_2 at T_2.

B. Discussion

In these experiments, the pursuit by the adversary has been improved by exploiting the interrupts on its Arduino UNO board to directly obtain the heading direction from Raspberry Pi 3. Such an approach brings down the sampling time of the adversary to just 0.2 s. It has been shown that the proposed algorithms allow quick computation of the meeting location, thus assisting in rescue of the delivery agent.

The meeting point does not necessarily lie on the line joining locations of the delivery and rescue agents. Furthermore, the allocation of rescue agent to a given delivery agent is not based on its proximity to the latter but is dependent on its proximity to the meeting location computed via Algorithm 6. This follows from the fact that relative distances of rescue agents from adversary are captured via the safe regions. Consequently, in Fig. 15, the weaker delivery agent D_1 is assigned to R_2 and not R_1, although R_1 is closer to D_1' than R_2.

VIII. Conclusion

In this article, we studied the problem of safeguarding a vehicle carrying food/medicines from predatory attacks. We developed algorithms to determine capture/rescue of the delivery agent based on the notion of Apollonius circle. We showed that an autonomous adversary can successfully capture the delivery agent based on only a vision sensor and without communication hardware. Furthermore, explicit vision support on the delivery or rescue agents is not required for successful rescue. Considering the possibility of multiple autonomous delivery agents in the field (one carrying food, another carrying medicines, etc.), the theory was extended to protection of k delivery agents (from an adversary) via k rescue agents. Experiments with multiple agents were also reported and they demonstrated that the autonomous delivery agents and rescue agents can execute their strategies using simply low-end microcontrollers without external memory.

The vision-based strategy is one of many approaches used by predators in nature. Others may be valuable for further study. One possibility is where the predator (adversary) can actively camouflage its movements [30] while tracking the target. In other words, we can consider an adversary that is capable of moving along a trajectory such that it appears as a stationary object to the moving target even during motion.

An alternative to the communication-based approach of the rescue agent is one involving a vision sensor to track the delivery agent. The challenges appear to be fewer here and this is due, in part, to the "same type" of hardware onboard the adversary and the rescue agent. Both the rescue agent and the adversary can pursue the delivery agent and the faster one in general can be expected to decide the outcome (capture/rescue). Furthermore, a camera onboard would increase the hardware on the rescue agent leading to increased weight and energy consumption by the autonomous rescue agent. The study can also be extended to aerial vehicles. In particular, protection of a ground vehicle by an aerial vehicle would be worth exploring, when the adversary, in particular, is airborne.

References

[1] J. Chen and M. Barnes, "Human-agent teaming for multirobot control: A review of human factors issues," *IEEE Trans. Human-Mach. Syst.*, vol. 44, no. 1, pp. 13–29, Feb. 2014.
[2] L. Weiss, "Autonomous robots in the fog of war," *IEEE Spectr.*, vol. 48, no. 8, pp. 30–57, Aug. 2011.
[3] W. Scott and N. Leonard, "Pursuit, herding and evasion: A three-agent model of caribou predation," in *Proc. Am. Control Conf.*, 2013, pp. 2978–2983.
[4] R. Isaacs, *Differential Games*. New York, NY, USA: Dover, 1965.
[5] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.
[6] R. Boyell, "Defending a moving target against missile or torpedo attack," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-12, no. 4, pp. 522–526, Jul. 1976.
[7] T. Vicsek, A. Czirok, E. Jacob, I. Choen, and O. Schochet, "Novel type of phase transitions in a system of self-driven particles," *Phys. Rev. Lett.*, vol. 75, pp. 1226–1229, 1995.
[8] G. Chen, Z. Yang, and C. Low, "Coordinating agents in shop floor environments from a dynamic systems perspective," *IEEE Trans. Ind. Informat.*, vol. 2, no. 4, pp. 269–280, Nov. 2006.
[9] B. An, Z. Shen, C. Miao, and D. Cheng, "Algorithms for transitive dependence-based coalition formation," *IEEE Trans. Ind. Informat.*, vol. 3, no. 3, pp. 234–245, Aug. 2007.
[10] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.
[11] D. Han, G. Chesi, and Y. Hung, "Robust consensus for a class of uncertain multi-agent dynamical systems," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 306–312, Feb. 2013.
[12] W. Yu, L. Zhou, X. Yu, J. Lu, and R. Lu, "Consensus in multi-agent systems with second-order dynamics and sampled data," *IEEE Trans. Ind. Informat.*, vol. 9, pp. 2137–2146, Nov. 2013.
[13] J. Zhan and X. Li, "Flocking of multi-agent systems via MPC based on position-only measurements," *IEEE Trans. Ind. Informat.*, vol. 9, pp. 377–385, Feb. 2013.
[14] Z. Zuo, Q. Han, B. Ning, X. Ge, and X. Zhang, "An overview of recent advances in fixed-time cooperative control of multiagent systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2322–2334, Jun. 2018.
[15] M. Long and C. Wu, "Energy-efficient and intrusion resilient authentication for ubiquitous access to factory floor information," *IEEE Trans. Ind. Informat.*, vol. 2, no. 1, pp. 40–47, Feb. 2006.
[16] Y. Yuan, H. Yuan, L. Guo, H. Yang, and S. Sun, "Resilient control of networked control system under DoS attacks: A unified game approach," *IEEE Trans. Ind. Informat.*, vol. 12, no. 5, pp. 1786–1794, Oct. 2016.
[17] M. Pachter, "Simple-motion pursuit-evasion in the half-plane," *Comput. Math. Appl.*, vol. 13, pp. 69–82, 1987.
[18] T. Chung, G. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics: A survey," *Auton. Robots*, vol. 31, no. 4, pp. 299–316, 2011.
[19] N. Noori and V. Isler, "Lion and man with visibility in monotone polygons," *Int. J. Rob. Res.*, vol. 33, no. 1, pp. 155–181, 2014.

[20] W. Lin, Z. Qu, and M. Simaan, “Nash strategies for pursuit-evasion differential games involving limited observations,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 51, no. 2, pp. 1347–1356, Apr. 2015.

[21] D. Li and J. Cruz, “Defending an asset: A linear quadratic game approach,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no. 2, pp. 1026–1044, Apr. 2011.

[22] E. Garcia, D. Casbeer, Z. Fuchs, and M. Pachter, “Cooperative missile guidance for active defense of air vehicles,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 2, pp. 706–721, Apr. 2018.

[23] D. Oyler, P. Kabamba, and A. Girard, “Pursuit evasion games in the presence of obstacles,” *Automatica*, vol. 65, pp. 1–11, 2016.

[24] R. Luo and C. Chang, “Multisensor fusion and integration: A review on approaches and its applications in mechatronics,” *IEEE Trans. Ind. Informat.*, vol. 8, no. 1, pp. 49–60, Feb. 2012.

[25] X. Zhou, Y. Li, B. He, and T. Bai, “GM-PHD-based multi-target visual tracking using entropy distribution and game theory,” *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1064–1076, May 2014.

[26] S. Bhattacharya and S. Hutchinson, “A cell decomposition approach to visibility-based pursuit evasion among obstacles,” *Int. J. Rob. Res.*, vol. 30, no. 14, pp. 1709–1727, 2011.

[27] R. Zou and S. Bhattacharya, “Visibility-based finite-horizon target tracking game,” *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 399–406, Jan. 2016.

[28] S. Giordani, M. Lujak, and F. Martinelli, “A distributed multi-agent production planning and scheduling framework for mobile robots,” *Comput. Ind. Eng.*, vol. 64, pp. 19–30, 2013.

[29] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, “A distributed version of the hungarian method for multirobot assignment,” *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 932–947, Aug. 2017.

[30] Y. Xu and G. Basset, “Virtual motion camouflage based phantom track generation through cooperative electronic combat air vehicles,” in *Proc. Am. Control Conf.*, 2010, pp. 5656–5661.

Bhaskar Vundurthy (S’16–M’20) received the B.E. (Hons.) degree in electronics and instrumentation engineering from the Birla Institute of Technology and Science, Pilani, India, in 2011. He, then, received the M.Tech. and Ph.D. degrees in electrical engineering from the Indian Institute of Technology Madras, Chennai, India, in 2013 and 2019, respectively.

His research interests include multiagent systems, robotics, game theory, and computational geometry.

K. Sridharan (S’84–M’96–SM’01) received the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1995.

He was an Assistant Professor with the Indian Institute of Technology (IIT) Guwahati from 1996 to 2001. Since June 2001, he has been with IIT Madras, Chennai, India, where he is presently a Professor. He was a Visiting Staff Member with Nanyang Technological University, Singapore, from 2000 to 2001 and 2006 to 2008. He has authored several papers, two Springer monographs, and holds two patents.

Prof. K. Sridharan is the recipient of the Vikram Sarabhai Research Award in 2009 and the Tan Chin Tuan Fellowship, in 2011. He was an Associate Editor of IEEE Transactions on Industrial Electronics.