SIAM J. CONTROL OPTIM.
Vol. 48, No. 1, pp. 292–320

A GEOMETRIC OPTIMIZATION APPROACH TO DETECTING AND INTERCEPTING DYNAMIC TARGETS USING A MOBILE SENSOR NETWORK*

SILVIA FERRARI†, RAFAEL FIERRO‡, BRENT PERTEET§, CHENGHUI CAI†, AND KELLI BAUMGARTNER†

Abstract. A methodology is developed to deploy a mobile sensor network for the purpose of detecting and capturing mobile targets in the plane. The sensing-pursuit problem considered in this paper is analogous to the Marco Polo game, in which a pursuer Marco must capture multiple mobile targets that are sensed intermittently, and with very limited information. The competing objectives exhibited by this problem arise in a number of surveillance and monitoring applications. In this paper, the mobile sensor network consists of a set of robotic sensors that must track and capture mobile targets based on the information obtained through cooperative detections. When these detections form a satisfactory target track, a mobile sensor is switched to pursuit mode and deployed to capture the target in minimum time. Since the sensors are installed on robotic platforms and have limited range, the geometry of the platforms and of the sensors' fields-of-view play a key role in obstacle avoidance and target detection. A new cell-decomposition approach is presented to determine the probability of detection and the cost of operating the sensors from the geometric properties of the network and its workspace. The correctness and complexity of the algorithm are analyzed, proving that the termination time is a function of the network parameters and of the number of required detections.

Key words. mobile sensor networks, pursuit-evasion games, coverage, tracking, detection

AMS subject classifications. 49N75, 46N10, 74P20, 93E10

DOI. 10.1137/07067934X

1. Introduction.

The proliferation of reliable low-cost sensors and autonomous vehicles is producing advanced surveillance systems comprised of robotic sensors with a high degree of functionality and reconfigurability. These mobile sensor networks can play a critical role in several application domains, such as landmine detection and identification [37, 14]; monitoring of endangered species [25]; monitoring of urban environments, manufacturing plants, and civil infrastructure; high-confidence medical devices; and intruder and target detection systems. These networks are expected to operate cooperatively and reliably in cluttered dynamic environments with little human intervention. Coordinating such large heterogeneous sensor networks is challenging and requires the development of novel methods of communication, motion control and planning, computation, proactive estimation and sensing, and power management.

*Received by the editors January 6, 2007; accepted for publication (in revised form) December 1, 2008; published electronically February 11, 2009.

http://www.siam.org/journals/sicon/48-1/67934.html

†Laboratory for Intelligent Systems and Controls, Department of Mechanical Engineering & Materials Science, Duke University, Durham, NC 27708-0300 (sferrari@duke.edu, cc88@duke.edu, kacb@alumni.duke.edu). The work of the first author was supported in part by the Office of Naval Research (Code 321), and by NSF grant ECS CAREER #0448906.

‡MARHES Lab, Department of Electrical & Computer Engineering, University of New Mexico, Albuquerque, NM 87131-0001 (rfierro@ece.unm.edu). The work of this author was supported by NSF grants ECCS CAREER #0811347, IIS #0812338, CNS #0709329, by DOE NNSA grant DE-FG52-04NA25590 (through the UNM Manufacturing Engineering Program), and by the U.S. Army Research Office under grant DAAD19-03-1-0142 (through the University of Oklahoma).

§Department of Electrical & Computer Engineering, Oklahoma State University, Stillwater, OK 74078-5032 (brent.perteet@gmail.com).

One paradigm common to many sensing applications consists of one or more sensors installed on robotic platforms that must move through an environment to obtain measurements from multiple targets. Most of the research relating sensor measurements to robot motion planning has focused on the effects that the uncertainty in the geometric models of the environment has on the motion strategies of the robot [27, 42, 41, 35, 33]. Hence, considerable progress has been made toward integrating sensor measurements in topological maps [46], and on planning strategies based on only partial or nondeterministic knowledge of the workspace [30, 32]. Coordination of robotic networks and sensor planning approaches have received considerable attention in recent years [51, 11, 3]. One line of research has investigated the extension of motion planning techniques to the problem of sensor placement for achieving coverage of unstructured environments [1, 9] or of a desired visibility space [30, 22]. Obstacle-avoidance motion planners have been effectively modified in [40, 8] to plan the path of mobile sensors for the detection and classification of stationary targets in an obstacle-populated environment. Probabilistic pursuit-evasion strategies to detect and capture intelligent evaders in obstacle-populated environments are described in [48]. In [24] the authors show that a pursuer can detect an arbitrarily fast evader in a polygonal environment using a randomized strategy. It is shown that one evader is guaranteed to be captured by two pursuers in finite time, by solving a *lion and man* problem and assuming that at least one pursuer is as fast as the evader [24].

In this paper, we develop a cell-decomposition methodology to optimize the probability of detection of a mobile sensor network, based on the geometry of the workspace and of the robotic sensors. Cell-decomposition algorithms have previously been employed to represent the obstacle-free configuration of a robot for the purpose of obstacle avoidance [29]. We present a framework for obtaining a decomposition in which observation cells are used to represent sensor configurations that intersect the targets while avoiding polygonal obstacles. The simple philosophy behind this approach is that while the geometry of the robot must not intersect that of an obstacle to avoid collision, the geometry of the sensor's field-of-view (or visibility region) must intersect that of a target to enable a detection. Then, the tracking information is used to determine the probability of detection in the observation cells.

At any given time, the pursuers must also detect new targets, for which there is no available track information. The monitoring of a workspace by means of multiple sensors is typically referred to as coverage. Coverage control for mobile sensors has been treated in [11] using Voronoi diagrams to achieve uniform sensing performance over an area-of-interest. Another well-known coverage problem is the art-gallery or line-of-sight visibility problem, in which multiple sensors are placed such that the targets are in the line-of-sight of at least one sensor in the network [36, 45, 47]. In this paper, we consider a track-coverage formulation in which multiple sensors are deployed to cooperatively detect moving targets traversing the area-of-interest [17]. By this formulation, the probability of detection of undetected targets is obtained for every cell in the decomposition. Then, the control policies that optimize a trade-off of multiple sensing objectives are obtained by searching the robot configuration graph and by performing inner-loop trajectory generation and tracking. The path obtained from the configuration graph is one that maximizes the overall probability of detection and minimizes the distance traveled by the pursuer to detect or capture the targets.

The remainder of the paper is organized as follows. The sensing-pursuit problem is formulated in section 2. The geometric approach used to control the network of robotic sensors to detect and pursue moving targets is presented in section 3. The correctness and performance analysis of the algorithm is presented in section 4. The

simulation results obtained are described in section 5.

2. Problem statement and assumptions. We consider a pursuit-evasion game in which N pursuers comprised of robotic sensors attempt to detect and pursue M moving targets. The game takes place in a square area-of-interest $\mathcal{S} \subset \mathbb{R}^2$, with boundary $\partial\mathcal{S}$ and dimensions $L \times L$. $\mathcal{S}$ is populated by n fixed and convex obstacles $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\} \subset \mathcal{S}$. The geometry of the ith pursuer is assumed to be a convex polygon denoted by $\mathcal{A}_i$, with a configuration q_i that specifies its position and orientation with respect to a fixed Cartesian frame $\mathcal{F}_\mathcal{S}$.

The dynamics of the pursuers can be approximated using the nonholonomic unicycle model,

$$\begin{aligned} \dot{x}_p^i &= v_p^i \cos\theta_p^i, \\ \dot{y}_p^i &= v_p^i \sin\theta_p^i, \\ \dot{\theta}_p^i &= \omega_p^i, \end{aligned} \tag{2.1}$$

where $q_i = (x_p^i,\ y_p^i,\ \theta_p^i) \in SE(2)$ and $p_i = [x_p^i \quad y_p^i]^T \in \mathbb{R}^2$ is the position vector of pursuer i (referred to as its centroid). The input to pursuer i is $u_p^i = [v_p^i \quad \omega_p^i]^T$, and $u_p \in \mathcal{U} \subset \mathbb{R}^2$. The set of all pursuers is denoted by $\mathcal{P}$, and $I_\mathcal{P}$ is the index set of $\mathcal{P}$. The set of all targets in $\mathcal{S}$ is denoted by $\mathcal{T}$, where $I_\mathcal{T}$ is the index set of $\mathcal{T}$. The model of the targets is given by

$$\begin{aligned} \dot{x}_\tau^j &= c_{x_\tau}^j, \\ \dot{y}_\tau^j &= c_{y_\tau}^j, \end{aligned} \tag{2.2}$$

where $\tau_j = [x_\tau^j \quad y_\tau^j]^T \in \mathbb{R}^2$ is the position vector of target j and $c_{x_\tau}^j$ and $c_{y_\tau}^j$ are constants. In other words, targets are assumed to move along straight lines

$$y_\tau^j(t) = \frac{c_{y_\tau}^j}{c_{x_\tau}^j} x_\tau^j(t) + y_\tau^j(0) - \frac{c_{y_\tau}^j}{c_{x_\tau}^j} x_\tau^j(0), \tag{2.3}$$

where $\tau_j(0) = (x_\tau^j(0),\ y_\tau^j(0)) \in \partial\mathcal{S}$, and they remain in $\mathcal{S}$ at all $t > 0$. Exceptions to this rule are maneuvers used to avoid an obstacle or another target. The heading angle of a target j is denoted by θ_τ^j; thus $\theta_\tau^j := \arctan(c_{y_\tau}^j, c_{x_\tau}^j)$. The maximum translational speed $V_{p,\tau_{\max}}$ of all sensors and targets is known, and $V_{p_{\max}} > V_{\tau_{\max}}$ [24]. While sensors can move with any speed in $[0, V_{p_{\max}}]$, it is assumed that the speed of every target is uniformly distributed in $[V_{\tau_{\min}}, V_{\tau_{\max}}]$, with $V_{\tau_{\min}} > 0$.

In the sensing problem, the paths of the targets are represented by rays or half-lines, denoted by $\mathcal{R}_\theta^j$, that are unknown a priori. The sensors installed on the robotic platforms are assumed to be isotropic or omnidirectional, and therefore their field-of-view is represented by a disk $\mathcal{D}_i = \mathcal{D}(p_i, r_i) \in \mathcal{S}$ with radius r_i and centered at p_i. The sensor i installed on the robot $\mathcal{A}_i$ has the ability to *detect* the jth target when $\mathcal{D}_i \cap \mathcal{R}_\theta^j \neq \varnothing$. The measurements obtained from each detection can be associated with a particular target using a data-association algorithm (such as [39, 12, 21]), but they may be subject to errors and false alarms. At any time t, the set of detections associated with a target j is denoted by Z_j^t, which symbolizes all measurements of the target positions τ_j obtained since the onset of the game t_0; e.g., $Z_j^t = \{z_j(t_1), z_j(t_2), \ldots, z_j(t_l)\}$. Since the sensors produce few individual observations for each moving target (e.g., due to their limited range) and are subject to frequent false alarms, the approach known as *track-before-detect* [50] is used, in which a set

of k spatially distributed sensor detections are used to estimate the *target track*, $\mathcal{R}_\theta^j$, from Z_j^t, before declaring a positive detection. Every track may be updated every time a new measurement becomes available from the target. Once a target track has been formed from at least k sensor detections that are obtained at different moments in time, an upper-level controller declares the target positively detected and deploys a pursuer to capture it. The inputs to the pursuers take into account the information available from all targets, $Z^t = \{Z_j^t \mid j \in I_\mathcal{T}\}$, in order to optimize their sensing and pursuit performance.

Let e_{ji} be the Euclidean distance from the jth target position, τ_j, to the closest pursuer; i.e., $e_{ji} = \min d(\tau_j, p_i)\ \forall i \in I_\mathcal{P}$. Then the pursuer i is said to *capture* the target j when $e_{ji} < \varepsilon$. The threshold value ε is called the *capture threshold* for an interval Δ_c called the *capture timeframe*. We are interested in applications where the sensor's field-of-view is much larger than the robot geometry, and hence a robot can sense a target without necessarily being close enough to capture it. Once a target is captured, it becomes inactive and is removed from the set $\mathcal{T}$; thus the game terminates when $\mathcal{T} = \varnothing$. In this game, no communication between targets and sensors takes place, but the sensors may obtain position information about the targets when they enter their fields-of-view. Based on the previous discussion, the sensing-pursuit problem can be stated as follows.

PROBLEM 2.1. *Given a set $\mathcal{P}$ of N pursuers and a set $\mathcal{T}$ of M targets moving within a specified game area $\mathcal{S}$, find a set of policies $u_p^i = c^i(q_i, Z^t) \in \mathcal{U}\ \forall i \in I_\mathcal{P}$ which maximizes the total sensing reward and minimizes the total time required to capture targets in $\mathcal{T}$ that have been positively detected.*

To complete the formulation of Problem 2.1, we define the sensing reward in terms of the probability of detection, as explained in section 3. Also, sensors and targets are modeled as hybrid systems consisting of continuous dynamics along with several discrete states [19]. Figure 2.1 shows a hierarchical state diagram for the various modes of operation. Sensors operate in one of two modes, *detection* or *pursuit*, depending on whether their primary objective is to detect targets or to capture them. Also, we assume that sensors have sufficient processing capabilities to determine the time and position of a detection event from their raw measurements.

In this problem, target tracks are classified based on the following definitions.

DEFINITION 2.2. *An unobserved track is the path of a target j for which there are no detections at the present time, t; thus $Z_j^t = \emptyset$.*

DEFINITION 2.3. *A partially observed track is the path of a target that is estimated from $1 < l < k$ individual sensor detections obtained up to the present time, t; i.e., $Z_j^t = \{z_j(t_1), \ldots, z_j(t_l)\}$.*

DEFINITION 2.4. *A fully observed track is the path of a target that is estimated from at least $k > 2$ individual sensor detections obtained up to the present time, t; i.e., $Z_j^t = \{z_j(t_1), \ldots, z_j(t_m)\}$, where $m \geq k$.*

The parameter k is chosen by the user based on the reliability of the sensor detections and on the cost associated with deploying a pursuer to capture the target. For instance, in [50] it was found that, from a geometric point of view, $k = 3$ is a convenient number of detections for estimating a track in the absence of false alarms. However, in certain surveillance applications the cost associated with capturing a target is very high, and therefore a higher number of detections may be required. Only after a track is fully observed is the target considered to be *positively detected*. Then, the estimated track is used by an upper-level controller to decide which pursuer to deploy and switch to pursuit mode, and to compute a pursuit strategy online (as described in section 3.3) that takes into account the kinematic constraints of the mobile sensors. The time that elapses between sensor i becoming a pursuer and

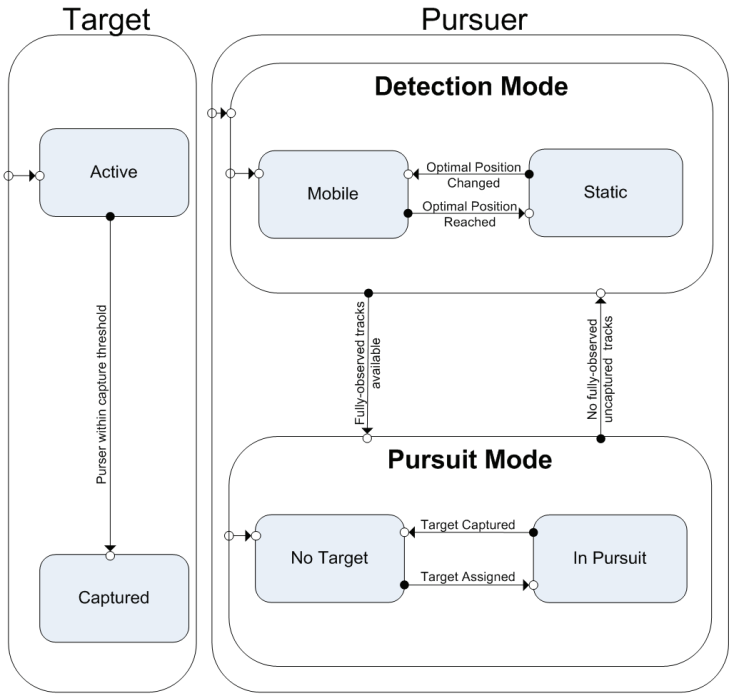

FIG. 2.1. *A finite state diagram which models the sensor (pursuer) and target as hierarchical hybrid systems with various discrete states of operation.*

intercepting target j is called the *capture time* and is denoted by $t_{c_i^j}$.

The objectives of the sensors in detection mode are to (i) avoid obstacles, (ii) maximize the probability of cooperatively detecting unobserved tracks, and (iii) maximize the probability of detecting p partially observed tracks $\{\mathcal{R}_\theta^1, \ldots, \mathcal{R}_\theta^p\}$. The objectives of a sensor i in pursuit mode are to (1) avoid obstacles and (2) minimize the time $t_{c_i^j}$ required to capture a positively detected target j, based on its fully observed track $\mathcal{R}_\theta^j$ and the pursuer's position at the time of deployment. Since in practice robotic sensors are subject to kinematic (e.g., nonholonomic), dynamic, and input constraints, we have designed and implemented a simple yet effective pursuit strategy that considers the nonholonomic constraints of the mobile sensor agents used in the simulations reported in section 5.

The following section describes a methodology for planning the motions of the pursuers, in order to meet all of the above objectives.

3. Methodology. The methodology described in this section computes policies for pursuers in detection or pursuit mode that must meet multiple sensing and motion objectives. At the onset of the game, all N pursuers are placed simultaneously into $\mathcal{S}$ in detection mode. A new game round is initiated when a new partially observed or fully observed track is obtained from the latest measurement set Z^t. At every new round of the game, one pursuer in $\mathcal{P}$ is deployed and, possibly, switched to either detection or pursuit mode. Since the pursuers can perform measurements only within their fields-of-view and are installed on robotic platforms, the problems of planning the sensor measurements and the platform paths are inevitably coupled.

The primary purpose for planning the motion of the pursuers in detection mode is to obtain measurements from the targets. However, since the target tracks may be unknown (unobserved) or uncertain (partially observed), the pursuers' motion cannot be planned using classical motion planning objectives, such as minimizing distance and reaching a final configuration [29]. In fact, the positions and fields-of-view of all pursuers must be taken into account to plan the motion of a robotic sensor in a cooperative network. Thus, at every round, a pursuer's trajectory is computed based on cooperative sensing *or* pursuit objectives and, subsequently, implemented by a trajectory tracking controller that is designed based on the unicycle model (2.1).

The sensor trajectory is obtained by modifying the classical motion planning approach known as cell decomposition [29]. Let $\mathcal{C}_{free}$ denote the robotic sensors' configuration space that is free of obstacles. A cell is defined as a closed and bounded subset of $\mathcal{C}_{free}$ within which a robotic sensor path can be easily generated, and is classified based on the following properties.

DEFINITION 3.1. *A void cell is a convex polygon $\kappa \subset \mathcal{C}_{free}$ with the property that for every configuration $q_i \in \kappa$ the sensor i has zero probability of detecting a partially observed target.*

In order to account for the geometries and dynamics of the pursuers and the targets, we also introduce the following definition.

DEFINITION 3.2. *An observation cell is a convex polygon $\underline{\kappa} \subset \mathcal{C}_{free}$ with the property that for every configuration $q_i \in \kappa$ the sensor i has a nonzero probability of detecting a partially observed target.*

Void and observation cells are determined such that an obstacle-free pursuer path can be easily computed between any two configurations inside each cell. Furthermore, two cells are said to be *adjacent* if they share a common boundary and, therefore, the pursuer can move between them without colliding with the obstacles. Typically, all cells are computed such that they do not overlap. In section 3.2, a method for obtaining these cells for the system in Problem 2.1 is presented. Subsequently, they are used to obtain the following graph.

DEFINITION 3.3. *A connectivity graph, $\mathcal{G}$, is an undirected graph where the nodes represent either an observation cell or a void cell, and two nodes in $\mathcal{G}$ are connected by an arc if and only if the corresponding cells are adjacent.*

The purpose of deploying pursuers in detection mode is to detect unobserved and partially observed target tracks. Thus, the sensing objectives are expressed in terms of a reward function that represents the improvement in the overall probability of detection that would be obtained by moving from a configuration $q_i \in \kappa_l$ to a configuration in an adjacent cell, $q_i \in \kappa_\imath$,

$$R(\kappa_l, \kappa_\imath) = P_{\mathcal{R}}(\kappa_\imath) + \Delta P^k_{\mathcal{S}}(\kappa_l, \kappa_\imath), \tag{3.1}$$

where $\Delta P^k_{\mathcal{S}}$ is the gain in the probability of cooperatively detecting unobserved tracks and $P_{\mathcal{R}}$ is the probability of detecting a target with a partially observed track. These probability density functions are obtained using the methodology described in sections 3.1 and 3.2, respectively.

At the onset of the game, $Z^0 = \varnothing$, and all targets are unobserved, with $P_{\mathcal{R}} = 0$ for any cell in $\mathcal{G}$. Thus, all N pursuers in $\mathcal{P}$ are placed simultaneously in $\mathcal{S}$ by maximizing their probability of cooperatively detecting unobserved tracks at least twice, i.e., $k = 2$, such that they may be declared partially observed. Since the sensors are omnidirectional, the orientation does not influence the region covered by each field-of-view, and the pursuers are placed by determining their initial positions

$\mathcal{X}_0 = \{p_1(t_0), \ldots, p_N(t_0)\}$ from the following optimization problem:

$$\mathcal{X}_0^* = \arg\max_{\mathcal{X}} \mathcal{P}_{\mathcal{S}}^2(\mathcal{X}) \tag{3.2}$$

with $0 \leq x_p^i \leq L$ and $0 \leq y_p^i \leq L$ $\forall i \in I_{\mathcal{P}}$. The above optimization amounts to a nonlinear program that can be solved by sequential quadratic programming [7, 5]. After all sensors are placed at $\mathcal{X}_0^*$, with some orientation θ_p^i, their initial configurations, $q_1(t_0), \ldots, q_N(t_0)$, are known and the game begins. At every game round, a pursuer is deployed in detection or pursuit mode, depending on whether the new track is partially or fully observed, respectively. If the pursuer is switched to and deployed in pursuit mode, then its obstacle-free trajectory is computed by the method in section 3.3. If the pursuer is deployed in detection mode, its obstacle-free trajectory is computed from the sequence of cells, or *channel*, that maximizes its total reward, i.e.,

$$\mu^* \equiv \{\kappa_0, \ldots, \kappa_f\}^* = \arg\max_{\mu} \sum_{(\kappa_l, \kappa_i) \in \mu} R(\kappa_l, \kappa_i), \tag{3.3}$$

where κ_f is chosen as the observation cell with the highest cumulative probability in $\mathcal{G}$, i.e., $\kappa_f = \arg\max_{\underline{\kappa}_i}(P_{\mathcal{R}}(\underline{\kappa}_i) + P_{\mathcal{S}}^k(\underline{\kappa}_i))$. In order to efficiently compute the optimal channel, μ^*, the value of the reward function (3.1) is attached to every arc in $\mathcal{G}$. Since the detection probabilities may vary slightly within each cell, they are computed in reference to the geometric centroid $\bar{q}_i$ of every cell κ_i. Then, the optimal channel μ^* is computed from $\mathcal{G}$ using the A* graph searching algorithm [29], and it is mapped into a set of waypoints that are used by a trajectory generator and trajectory tracking controller to determine the pursuer policy $u_p^i = c^i(q_i, Z^t)$.

If all sensors have the same geometry, the same connectivity-graph structure (i.e., the nodes and arcs of $\mathcal{G}$) can be utilized for all sensors. Otherwise, a different connectivity graph $\mathcal{G}_i$ may be employed for each geometry $\mathcal{A}_i$. At every round, the arc labels and the initial and final cells, κ_0 and κ_f, vary based on the latest measurements and on the sensor that is being deployed. Therefore, the A* algorithm must be run at every round of the game (section 4). In the following subsections, the probabilities of detection of unobserved and partially observed tracks that are used to define the reward function (3.1) are derived using a geometric approach.

3.1. Probability of detection for unobserved tracks. As shown in the previous section, the observation cells in the connectivity graph represent subsets of configurations that enable measurements from partially observed tracks. Also, at any given time, the network of pursuers must detect unobserved tracks of targets that have just entered the search area in $\mathcal{S}$ that have been previously missed. Since the targets are always in motion, maximizing area coverage or other coverage formulations may not lead to effective cooperative detections. It was recently shown in [4, 49] that the quality of service of an omnidirectional sensor network performing cooperative detections of moving targets, referred to as track coverage, can be assessed without any prior knowledge of the target tracks, and depends only on the geometry of the sensors and of the search area.

In this section, track coverage is formulated using geometric transversal theory (see [23] for a comprehensive review).

DEFINITION 3.4. *A family of k convex sets in $\mathbb{R}^c$ is said to have a d-*transversal *if it is intersected by a common d-dimensional flat (or translate of a linear subspace).*

When $d = 1$ and $c = 2$, the transversal is said to be a *line stabber* of the family of convex sets. Therefore, a track detected by k sensors is a stabber of their fields-of-view, e.g., of a family $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ in $\mathbb{R}^2$. It can be shown [17] that the family of

stabbers with y-intercept b_y of a disk $\mathcal{D}_i(p_i, r_i)$ in $\mathbb{R}^2$ can be represented by the cone generated by the unit vectors

$$\hat{h}_i(b_y) = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i \\ \sin\alpha_i & \cos\alpha_i \end{bmatrix} \frac{v_i}{\|v_i\|} = \mathrm{Q}_i^+ \hat{v}_i \tag{3.4}$$

and

$$\hat{l}_i(b_y) = \begin{bmatrix} \cos\alpha_i & \sin\alpha_i \\ -\sin\alpha_i & \cos\alpha_i \end{bmatrix} \frac{v_i}{\|v_i\|} = \mathrm{Q}_i^- \hat{v}_i, \tag{3.5}$$

where $v_i \equiv p_i - [0 \quad b_y]^T$. This cone, denoted by $K(\mathcal{D}_i, b_y) = \mathit{cone}(\hat{l}_i, \hat{h}_i)$, is referred to as the *coverage cone* of $\mathcal{D}_i$, with origin b_y. For notational simplicity, we omit the dependency on b_y and write the above unit vectors as $\hat{l}_i = \hat{l}_i(b_y)$ and $\hat{h}_i = \hat{h}_i(b_y)$. The angle α_i denotes half the opening angle of $K(\mathcal{D}_i, b_y)$, and its sine function can be computed from the sensor position p_i:

$$\sin\alpha_i = \frac{r_i}{\|v_i\|} = \frac{r_i}{\sqrt{(x_p^i)^2 + (y_p^i - b_y)^2}}. \tag{3.6}$$

The above unit vectors are also used to determine the line stabbers of families of k nontranslate disks. We order all unit vectors in $\mathbb{R}^2$ based on the orientation of the frame $\mathcal{F}_\mathcal{S}$ (i.e., counterclockwise). Two vectors $u_i, u_j \in \mathbb{R}^2$ are said to be ordered according to the orientation of a reference frame $\mathcal{F}_\mathcal{S}$ as $u_i \prec u_j$ if, when these vectors are translated such that their origins coincide and u_i is rotated through the smallest possible angle to meet u_j, this orientation is in the same direction as the orientation of $\mathcal{F}_\mathcal{S}$ [15]. Then, the family of stabbers with y-intercept b_y can be obtained for a family of disks, as shown by the following result.

PROPOSITION 3.5. *The set of all stabbers of a family of disks $D_k = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$, through b_y, is contained by the finitely generated cone*

$$K_k(D_k, b_y) = \mathit{cone}(\hat{l}^*, \hat{h}^*), \tag{3.7}$$

where

$$(\hat{l}^*, \hat{h}^*) = (\hat{l}_\imath, \hat{h}_\jmath), \quad \textit{where } \hat{l}_\imath \succeq \hat{l}_i,\ \hat{h}_\jmath \preceq \hat{h}_j,\ \hat{l}_\imath \prec \hat{h}_\jmath, \quad \forall i, j \in I_{D_k} \tag{3.8}$$

and I_{D_k} denotes the index set of D_k. If $\hat{l}_\imath \succeq \hat{h}_\jmath$, then $K_k(D_k, b_y) = \varnothing$.

A proof is provided in Appendix A. Since $K_k(D_k, b_y)$ represents the set of tracks detected by a family of k sensors, it is referred to as the *k-coverage cone*. The opening angle of this k-coverage cone obtained by the cross product,

$$\psi = \sin^{-1} ||\hat{l}^* \times \hat{h}^*|| = H(\det[\hat{l}^* \quad \hat{h}^*]^T) \sin^{-1}(\det[\hat{l}^* \quad \hat{h}^*]^T), \tag{3.9}$$

is a Lebesgue measure over the set of line stabbers of D and is used below to obtain the probability of detection of unobserved tracks. The Heaviside function $H(\cdot)$ guarantees that if $\hat{l}^* \succ \hat{h}^*$, the opening angle of the coverage cone is equal to zero.

We restrict our attention to tracks that traverse $\mathcal{S}$ and thus intersect two of its sides. Place $\mathcal{F}_\mathcal{S}$ along two sides of $\mathcal{S}$, and a second reference frame, $\mathcal{F}'_\mathcal{S}$, along the remaining sides, as shown in Figure 3.1. Since both frames have the same orientation, Proposition 3.5 can be applied to stabbers with any intercept, namely b_y, b_x, $b_{y'}$, and $b_{x'}$ (see Figure 3.1). The opening angles of the corresponding k-coverage cones are

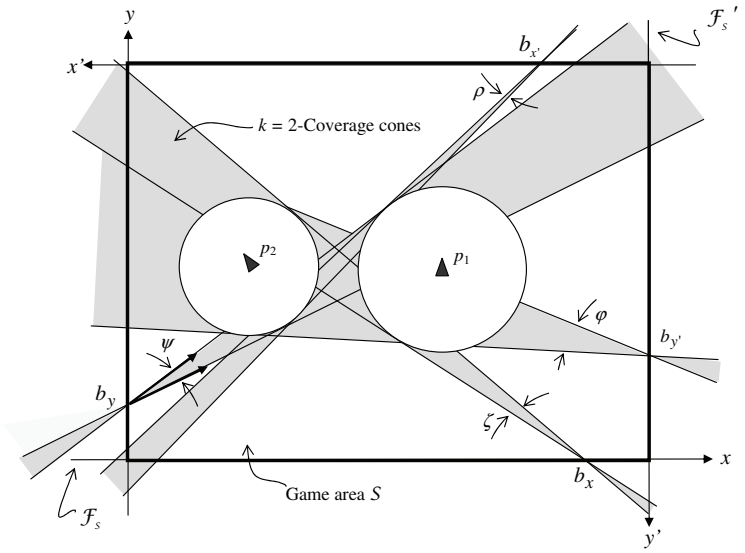

FIG. 3.1. *Coverage cone definition illustrated for two sensors with fields-of-view centered at p_1 and p_2, and a rectangular area-of-interest $\mathcal{S}$, with perimeter $\partial\mathcal{S}$ shown in bold.*

denoted by ψ, ζ, φ, and ρ, respectively, and are illustrated in Figure 3.1 for a family of $N = 2$ sensors and $k = 2$ required detections. We assume that prior to obtaining detections in $\mathcal{S}$, the probability that a target enters $\mathcal{S}$ through any intercept $b \in \partial\mathcal{S}$ and with a heading $\theta_\tau \in (-\pi/2, \ +\pi/2)$ is uniformly distributed over all of their possible values. The set of tracks traversing $\mathcal{S}$ and intersecting at least k disks is approximated by the union of the k-coverage cones over a set of intercept values that are obtained by discretizing $\partial\mathcal{S}$ using a constant interval δb. Then, the following result can be obtained for a family of N disks representing the fields-of-view of the sensor network.

THEOREM 3.6. *The probability of detection of unobserved tracks for a set $\mathcal{P}$ of N pursuers with fields-of-view $\mathcal{D}_1, \ldots, \mathcal{D}_N$, in a square game area $\mathcal{S}$ of dimensions $L \times L$, is a multivariate probability density function of the sensors' positions $\mathcal{X} = \{p_1, \ldots, p_N\}$ given by a Lebesgue measure on this union,*

$$P_{\mathcal{S}}^{k}(\mathcal{X}) = \frac{\delta b}{4\pi L} \sum_{\ell=1}^{L/\delta b} \sum_{j=1}^{m} (-1)^{j+1} \sum_{1 \le i_1 < \cdots < i_j \le m} [\psi(D_p{}^{i_{1,j}}, b_y^\ell) + \varphi(D_p{}^{i_{1,j}}, b_{y'}^\ell)]$$
$$+ \frac{\delta b}{4\pi L} \sum_{\ell=0}^{(L/\delta b - 1)} \sum_{j=1}^{m} (-1)^{j+1} \sum_{1 \le i_1 < \cdots < i_j \le m} [\zeta(D_p{}^{i_{1,j}}, b_x^\ell) + \rho(D_p{}^{i_{1,j}}, b_{x'}^\ell)]$$
$$\text{with} \quad m = \frac{N!}{(N-k)!k!}, \qquad D_p{}^{i_{1,j}} \equiv \{D_k^{i_1} \cup \cdots \cup D_k^{i_j}\}, \tag{3.10}$$

where the summation $\sum_{1 \le i_1 < \cdots < i_j \le m}$ is a sum over all the $[m!/(m-j)!\, j!]$ distinct integer j-tuples $(i_1, \ldots, i_j)$ satisfying $1 \le i_1 < \cdots < i_j \le m$, $D_k^{i_l}$ denotes the i_lth k-subset of D, and $D_p{}^{i_{1,j}}$ is a p-subset of D, with $k \le p \le n$.

A proof of this theorem is provided in Appendix B.

By letting $\delta b \to 0$, the Lebesgue measure (3.10) approaches the measure over

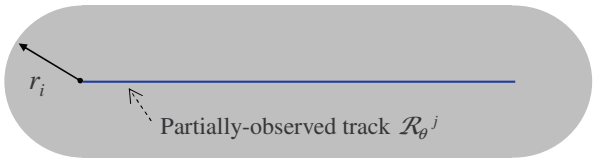

FIG. 3.2. *C-target grown isotropically from a partially observed track $\mathcal{R}_\theta^j$, based on the ith sensor range r_i.*

the entire set of tracks that traverse $\mathcal{S}$ [4]. In practice, the value δb is chosen by the user based on a trade-off between accuracy and computation time. Also, if a sensor moves to a cell κ_l, the new network configuration is approximated by $\mathcal{X}_l = \{p_1, \ldots, p_i \subset \bar{q}_l, \ldots, p_N\}$, letting the center of the sensors' field-of-view, p_i, coincide with the centroid $\bar{q}_l$ of κ_l. Thus, the gain in probability of detection for unobserved tracks that is associated with moving between two nodes $\kappa_l \to \kappa_i$ in $\mathcal{G}$ is

$$\Delta P_{\mathcal{S}}^k(\kappa_l, \kappa_i) \equiv P_{\mathcal{S}}^k(\mathcal{X}_i) - P_{\mathcal{S}}^k(\mathcal{X}_l). \tag{3.11}$$

The gain $\Delta P_{\mathcal{S}}^k$ is negative when the above change in configuration leads to a decreased probability of detection of unobserved tracks. However, since sensors in detection mode are moved according to (3.3) and both $P_{\mathcal{R}}$ and $P_{\mathcal{S}}^k$ pertain to the same set of targets $\mathcal{T}$, the overall probability of detection (3.1) increases at every round of the game.

3.2. Probability of detection for partially observed tracks. The partially observed tracks are viewed as an opportunity for obtaining additional measurements before investing in the costly resources needed to capture a target. In order to account for the geometry of the sensor field-of-view $\mathcal{D}_i$, the platform $\mathcal{A}_i$, and the target track $\mathcal{R}_\theta^j$, we present an approach motivated by cell-decomposition algorithms [29]. The simple philosophy behind this approach is that, in sensor planning problems, targets can be viewed as the dual of obstacles in classic robot motion planning. While in classic robot motion planning the geometry of the robot must avoid intersecting that of any obstacle, in sensor planning the geometry of the sensor's field-of-view must intersect that of the targets in order to enable sensor measurements.

Let $\mathcal{F}_{\mathcal{A}_i}$ denote a moving Cartesian frame embedded in $\mathcal{A}_i$. The configuration q_i specifies the position and orientation of $\mathcal{F}_{\mathcal{A}_i}$ with respect to the inertial frame $\mathcal{F}_{\mathcal{S}}$. If we assume that $\mathcal{D}_i$ and $\mathcal{A}_i$ are both rigid, then q_i also specifies the position of every point in $\mathcal{D}_i$ (or $\mathcal{A}_i$) relative to $\mathcal{F}_{\mathcal{S}}$. Using the latest estimate of a partially observed track, it is possible to identify the subset of $\mathcal{S}$ in which the sensors may obtain target measurements.

DEFINITION 3.7 (C-target). *The target track $\mathcal{R}_\theta^j$ in $\mathcal{S}$ maps in the ith sensor configuration space $\mathcal{C}$ to the C-target region $\mathcal{CR}_j = \{q_i \in \mathcal{C} \mid \mathcal{D}_i \cap \mathcal{R}_j \neq \varnothing,\ i \in I_{\mathcal{P}},\ j \in I_{\mathcal{T}}\}$.*

The boundary of a C-target is the curve followed by the origin of $\mathcal{F}_{\mathcal{A}_i}$ when $\mathcal{D}_i$ slides in contact with the boundary of $\mathcal{R}_\theta^j$. With the assumed robot and sensor geometries, the C-target boundaries are obtained by growing $\mathcal{R}_\theta^j$ isotropically by the radius r_i within $\mathcal{S}$, and they have the pill shape shown in Figure 3.2. C-obstacles are similarly defined [29] and are used together with the C-targets introduced above to obtain the connectivity graph $\mathcal{G}$ at every round.

Let $\mathcal{CO}_k$ denote the C-obstacle obtained from the kth obstacle in the game area,

$\mathcal{O}_k \subset \mathcal{S}$. In obstacle-avoidance algorithms, the obstacle-free configuration space,

$$\mathcal{C}^i_{free} = \mathcal{C} \setminus \bigcup_{k=1}^{n} \mathcal{CO}_k = \left\{ q_i \in \mathcal{C} \mid \mathcal{A}_i(q_i) \cap \left(\bigcup_{k=1}^{n} \mathcal{O}_k \right) = \varnothing \right\}, \tag{3.12}$$

is decomposed into a finite set of cells, $\{\kappa_1, \ldots, \kappa_f\}$, within which a path free of obstacles can be easily generated. In order to obtain a decomposition that includes observation cells (Definition 3.2), we present the following method:

(I) Decompose the configuration space that is void of any C-obstacles or C-targets and is defined as

$$\begin{aligned} \mathcal{C}^i_{void} &= \mathcal{C} \setminus \left\{ \bigcup_{j=1}^{n} \mathcal{CO}_k \cap \bigcup_{i=1}^{p} \mathcal{CR}_j \right\} \\ &= \left\{ q_i \in \mathcal{C} \mid \mathcal{A}_i(q_i) \cap \left(\bigcup_{k=1}^{n} \mathcal{O}_k \right) = \varnothing, \ \mathcal{D}_i(q_i) \cap \left(\bigcup_{j=1}^{p} \mathcal{R}_j \right) = \varnothing \right\}. \end{aligned} \tag{3.13}$$

(II) Decompose each obstacle-free C-target,

$$\mathcal{CR}_j \setminus \bigcup_{j=1}^{n} \mathcal{CO}_k, \qquad j = 1, \ldots, p, \tag{3.14}$$

thereby obtaining the set of observation cells.

(III) Construct a connectivity graph $\mathcal{G}$ using the void and observation cells obtained in (I) and (II), respectively.

When the C-targets are grown isotropically by a disk (Figure 3.2), the decomposition may involve generalized polygons [29]. A sweeping-line algorithm can be used to decompose a nonconvex generalized polygon with ν vertices into $O(\nu)$ convex generalized polygons in $O(\nu \log \nu)$ time (see section 5.1 in [29]). Alternatively, the pill-shaped C-targets can be approximated by a convex polygon, obtaining the running time presented in section 4. An illustrative example of workspace and corresponding cell decomposition is shown in Figure 3.3. The connectivity graph constructed using this cell decomposition is illustrated in Figure 3.4, where the observation cells are shown in grey and the void cells are white. Each node in the connectivity graph corresponds to one polygonal cell in Figure 3.3, where the cells are numbered from left to right and from top to bottom.

The probability density function $P_{\mathcal{R}}$, used to compute the reward (3.1), is obtained as follows. Suppose that $\underline{\kappa}_l$ is one of the observation cells that are obtained from the decomposition of the jth C-target: $\underline{\kappa}_l \subset \mathcal{CR}_j$. Then, the sensing benefit of visiting the lth cell in $\mathcal{G}$ is the probability of detecting the target j,

$$P_{\mathcal{R}}(\underline{\kappa}_l \subset \mathcal{CR}_j) = \Pr\{D_{ji} = 1 \mid e_{ji} \leq r_i\}, \tag{3.15}$$

where D_{ji} represents the event that the ith sensor reports a detection when the jth target comes within its detection range. In this paper, $P_{\mathcal{R}}$ is assumed to be uniform over $\mathcal{CR}_j$ for simplicity, and when a cell is void $P_{\mathcal{R}}(\kappa_l) = 0$ since the sensor field-of-view will not intersect any of the p partially observed tracks. In general, $P_{\mathcal{R}}$ can be estimated from knowledge of the measurement process and can be made dependent on time and on the distance from the target [18].

The next section presents an effective control methodology by which sensors in pursuit mode capture and intercept targets whose tracks have been fully observed and thus have been declared positively detected.

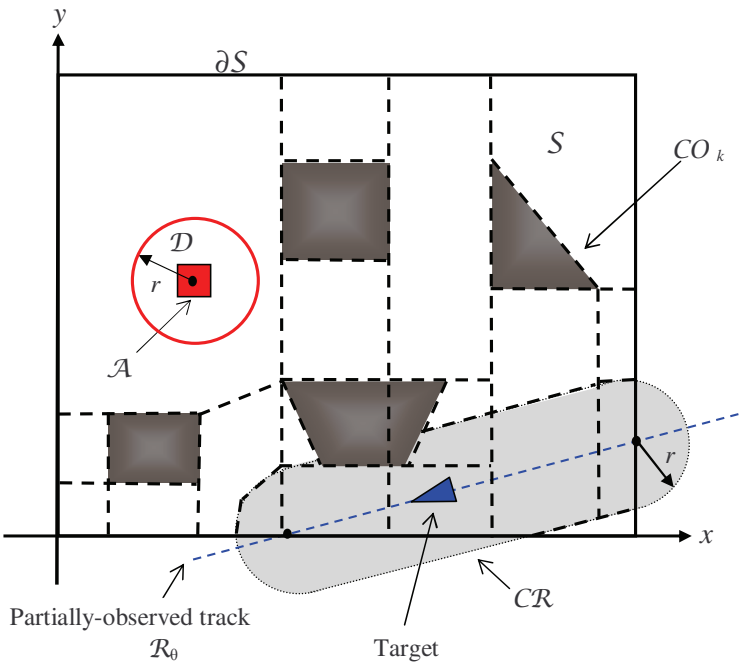

FIG. 3.3. *Example of cell decomposition (dashed lines) for a workspace with four C-obstacles (darkly shaded polygons) and one C-target $\mathcal{CR}$ (lightly shaded region) corresponding to $2 < k$ detections. One sensor with range r and field-of-view $\mathcal{D}$ is installed on a robot with a square platform geometry $\mathcal{A}$.*

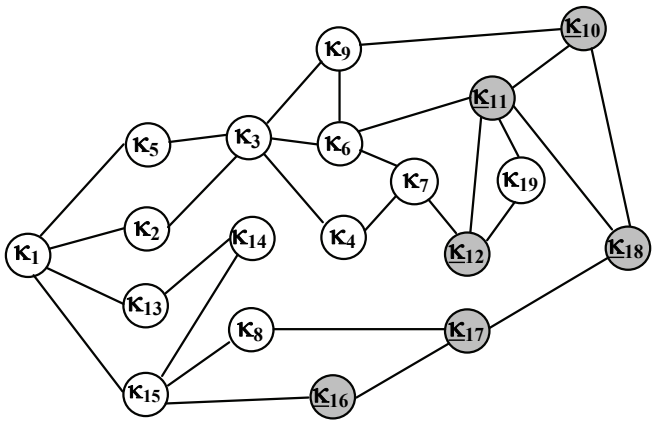

FIG. 3.4. *Connectivity graph obtained from the cell decomposition in Figure* 3.3*, where the cells in the decomposition are numbered from left to right and from top to bottom, and the observation cells are shown in grey.*

3.3. Pursuit strategy. Once a new target is positively detected, a sensor is switched to pursuit mode and deployed to capture it. A geometric approach motivated by the behavior of the potential field controller, described in [10], is used to drive the

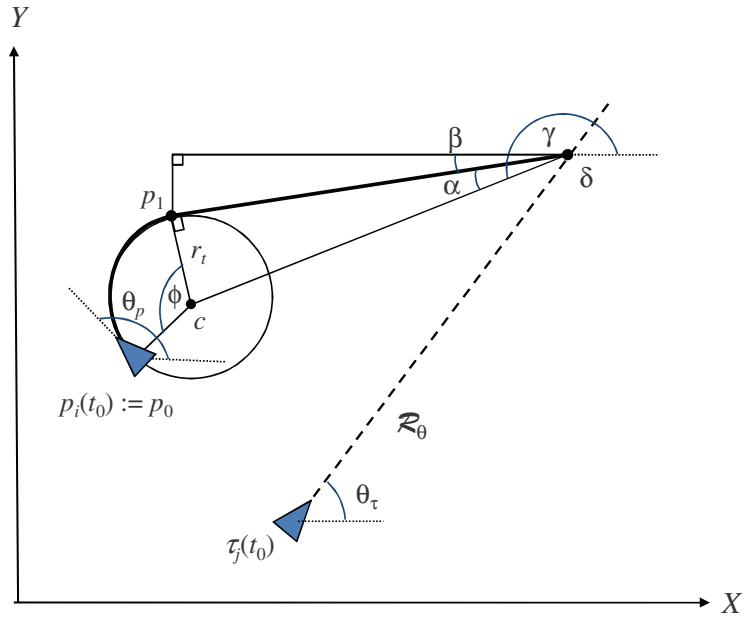

FIG. 3.5. *Control strategy to capture a target.*

nonholonomic robot sensor in pursuit mode to a goal waypoint, $\delta \in \mathbb{R}^2$, calculating both the point and time at which a pursuer will intercept a target moving in a known straight line, $\mathcal{R}^j_\theta$, with constant velocity, V_τ. This strategy, depicted in Figure 3.5, is based upon the geometry of the problem and takes into account the kinetic constraints of the pursuer but not the presence of the obstacles. Therefore, here it is combined with the cell-decomposition methodology presented in section 3.2.

First, the interception point δ is calculated by determining the time required by both the pursuer and target to reach δ. The pursuit initial time t_0 can be assumed to be the time at which the last detection $z_j(t_k)$ became available from target j, and $p_i(t_0)$ and $\tau_j(t_0)$ denote the initial positions of the pursuer i and target j, respectively. The interception point δ and the time to interception $t_{c_i^j}$ are defined as

$$\delta = \begin{bmatrix} x^j_\tau(t_0) + t_{c_i^j} V_\tau \cos\theta^j_\tau \\ y^j_\tau(t_0) + t_{c_i^j} V_\tau \sin\theta^j_\tau \end{bmatrix}, \qquad t_{c_i^j} = \frac{r_t\phi + \|c-\delta\|\cos\alpha}{V_{p_{\max}}}, \tag{3.16}$$

where the distance traveled by the pursuer is the distance along the arc p_0p_1 plus the straight line distance between p_1 and δ. The arc radius is the same as the turn radius of the pursuer and is defined as $r_t = \frac{V_{p_{\max}}}{\omega_p}$, where $V_{p_{\max}}$ and ω_p are the maximum speed and angular velocity, respectively, of the pursuer. There are two possible circles corresponding to a right or a left turn of the pursuer. The center points, c_R and c_L, of the circles defined by the turn radius are calculated as

$$c_R = \begin{bmatrix} p_{0x} + r_t\cos(\theta_p - \frac{\pi}{2}) \\ p_{0y} + r_t\sin(\theta_p - \frac{\pi}{2}) \end{bmatrix}, \qquad c_L = \begin{bmatrix} p_{0x} + r_t\cos(\theta_p + \frac{\pi}{2}) \\ p_{0y} + r_t\sin(\theta_p + \frac{\pi}{2}) \end{bmatrix}.$$

The center point lying closest to the interception point is chosen as

$$c = \begin{cases} c_R & \text{if } \|c_R - \delta\| \le \|c_L - \delta\|, \\ c_L & \text{if } \|c_R - \delta\| > \|c_L - \delta\|. \end{cases}$$

The other parameters for calculating the interception time are calculated as

$$\alpha = \arcsin\left(\frac{r_t}{\|c-\delta\|}\right), \qquad \gamma = \arctan\left[c_y - \delta_y, c_x - \delta_x\right],$$

$$\beta = \begin{cases} \gamma - \alpha & \text{if } c = c_R, \\ \gamma + \alpha & \text{if } c = c_L, \end{cases} \qquad p_1 = \begin{bmatrix} \delta_x + \|c-\delta\|\cos\alpha\cos\beta \\ \delta_y + \|c-\delta\|\cos\alpha\sin\beta \end{bmatrix},$$

$$\phi = \left|\arctan(p_{1y} - c_y, p_{1x} - c_x) - \arctan(p_{0y} - c_y, p_{0x} - c_x)\right|,$$

where $\delta = [\delta_x\ \delta_y]^T$. The time to interception $t_{c_i^j}$ and the interception point δ in (3.16) are computed numerically by Newton's method [34].

In order to find an obstacle-free shortest path between $p_i(t_0)$ and δ, the connectivity graph $\mathcal{G}$ obtained in section 3.2 is modified by changing the arc labels to reflect the Euclidean distance between any two nodes $\kappa_l \to \kappa_\iota$ in $\mathcal{G}$:

$$d(\kappa_l, \kappa_\iota) \equiv \max ||\mathcal{A}(\bar{q}_\iota) - \mathcal{A}(\bar{q}_l)||. \tag{3.17}$$

The channel μ_p^* of shortest overall distance between $\kappa_0 \ni q_i(t_0)$ and $\kappa_f \ni \delta$ (assuming a zero heading at δ) can be determined by the graph searching algorithm A* [29]. Subsequently, μ_p^* is mapped into a set of waypoints in $\mathbb{R}^2_+$ that are used by an inner-loop trajectory generator and trajectory tracking controller designed for the unicycle model (2.1).

4. Performance and complexity analysis. Previous work on the correctness and complexity of pursuit-evasion games has focused on graphs, in which one or more pursuers attempt to capture one target by moving between adjacent nodes in a graph (see [24, 2, 28, 38] for a comprehensive review). In these problem formulations, the sensing ability and fields-of-view of the pursuers are not taken into account, and the pursuit strategies consist of randomized searches on the graph, because the pursuers cannot see the evader until the latter is caught. Also, only one evader who may be restricted or unrestricted to the graph is considered during each game. By computing the connectivity graph by the methodology in section 3.2, these results could potentially be extended to the pursuit-evasion game in Problem 2.1. For example, if the strategy in [2] is implemented for one pursuer and one evader ($N = M = 1$), then the pursuer captures the evader on an n-node cycle with probability at least $\Omega(1/\log(n_{\mathcal{G}}))$, and the game ends in $O(n_{\mathcal{G}}\ \log(\text{diam}(\mathcal{G})))$ time, where $n_{\mathcal{G}}$ is the number of nodes in $\mathcal{G}$ and $\text{diam}(\mathcal{G})$ is the diameter of the graph. However, by not taking into account the sensing ability of the pursuers and the knowledge of fully observed tracks (i.e., the presence of observation cells), these strategies are not very effective at capturing multiple evaders in large game areas. In these applications, $n_{\mathcal{G}}$ and $\text{diam}(\mathcal{G})$ are very large. Therefore, the probability of capturing the evaders can be very small and the game end time $O(Mn_{\mathcal{G}}\log(\text{diam}(\mathcal{G}))$ can be very large.

The correctness and game end time for the strategy presented in section 3 are analyzed by assuming that the time required to maneuver around obstacles or to turn (ϕ/ω_p) are negligibly small compared to the duration of the game. Let $(\bar{\cdot})$ denote the expected value (or mean), and $\lfloor\cdot\rfloor$ denote the floor function. Then, the performance of the sensor network depends on the dimension of the game area (L), the number of sensors (N), the number of required detections ($k > 2$), and the field-of-view radius

(r_i), which here is assumed constant $(r_i = r\ \forall i)$ for simplicity, as summarized by the following result.

THEOREM 4.1. *The pursuit-evasion game in Problem* 2.1 *is guaranteed to terminate, provided that*

$$N \geq N_{\min} = \frac{1}{2}\left[\left\lfloor \frac{2L}{r} \right\rfloor + k - 1 + \left| \left\lfloor \frac{2L}{r} \right\rfloor - k + 3 \right|\right], \tag{4.1}$$

and requires a time

$$t_f \leq T_u = \frac{(\sqrt{2}L - 2r)}{V_{\tau_{\min}}} + \left[\left\lfloor \frac{(k-2)M}{N} \right\rfloor + 1\right] \frac{(\sqrt{2}L - r)}{\bar{V}_p} \tag{4.2}$$

$$+ \frac{r}{(V_{p_{\max}}^2 - \bar{V}_\tau^2)} + \left\lfloor \frac{M}{N} \right\rfloor \frac{\left(\bar{V}_\tau + \sqrt{2V_{p_{\max}}^2 - \bar{V}_\tau^2}\right)}{(V_{p_{\max}}^2 - \bar{V}_\tau^2)^2} L \tag{4.3}$$

to capture all M targets in $\mathcal{T}$. If the network contains at least

$$N_\ell = \frac{1}{2}\left[\ell \left\lfloor \frac{2L}{r} \right\rfloor - 4\ell(\ell-1) + (k-2)M + \left| \ell \left\lfloor \frac{2L}{r} \right\rfloor - 4\ell(\ell-1) - (k-2)M \right|\right] \tag{4.4}$$

sensors, with $\ell = 1, \ldots, \lfloor L/4r \rfloor$, then all targets in $\mathcal{T}$ can be captured in a time

$$t_f \leq T_\ell = \frac{1}{V_{\tau_{\min}}}\left\{ \frac{\sqrt{2}}{2}L - 2\sqrt{2}r(\ell-1) + \left| 2r[1 + \sqrt{2}(\ell-1)] - \frac{\sqrt{2}}{2}L \right| \right\}$$

$$+ \frac{(\sqrt{2}L - r)}{\bar{V}_p} + \frac{r}{(V_{p_{\max}} - \bar{V}_\tau)}, \tag{4.5}$$

and the game terminates in $t_f \leq T_\ell \leq T_u$, where $T_\ell = T_u$ when $\ell = 1$ and $k = 3$.

A proof is provided in Appendix C. Based on the above result, $N_{\min}$ is the minimum number of sensors needed to guarantee that the game will end in less than T_u time. But, if more sensors can be utilized, then N can be increased according to (4.4) to decrease the maximum time required to end the game, as shown by (4.5). The performance of the algorithm also depends on the choice of reward function (3.1) through the parameter k, which, together with the parameters N, L, and r, specifies the definition of the probability density function (3.10).

In Problem 2.1, a round is defined as the deployment of one sensor in either detection or pursuit mode, and it is initiated based on the measurement set Z^t when a new target track becomes either partially observed or fully observed. Thus, the computational complexity of the algorithm in section 3 is assessed based on the calculations required by each round. Let $n_{e_\mathcal{O}}$ denote the number of edges required to describe all n obstacles in $\mathcal{S}$, and $n_{e_\mathcal{R}} = 2p$ denote the number of edges required to describe all p tracks that have been partially observed up to the present time. Then, if $n_{\underline{\kappa}}$ and $n_{\mathcal{G}_a}$ are the number of observation cells and the number of arcs in $\mathcal{G}$, respectively, and $n_\delta \equiv L/\delta b$, the following result is obtained.

THEOREM 4.2. *In every round of the pursuit-evasion game in Problem* 2.1*, the running time required to deploy a sensor in detection mode is*

$$\Gamma_d = O((n_{e_\mathcal{O}} + n_{e_\mathcal{R}}) \log(n_{e_\mathcal{O}} + n_{e_\mathcal{R}}) + n_{e_\mathcal{R}} n_{e_\mathcal{O}} \log n_{e_\mathcal{O}}) + O(n_{\underline{\kappa}} n_\delta m(k + \log m)) + O(n_\mathcal{G}^2 + n_{\mathcal{G}_a}), \tag{4.6}$$

where $m = \binom{N}{k}$ *is given by the binomial coefficient and the running time required to deploy a sensor in pursuit mode is*

$$\Gamma_p = O((n_{\mathcal{G}} + n_{\mathcal{G}_a}) \log_2 n_{\mathcal{G}}). \tag{4.7}$$

A proof is provided in Appendix D. Clearly, depending on the characteristics of the robotic sensors and of the workspace, $\mathcal{S}$, only one of the three terms in (4.6) will dominate over the others, providing the overall running time complexity of the detection round. As an example, when the leading time complexity is that of the reward function (3.1), the deployment of a sensor in detection mode in a problem with $N = 50$, $k = 3$, $r = 5$ km, and $L = 100$ km took 0.797 sec on a Pentium 4 CPU 3.06 GHz computer. On the same computer, when the leading time complexity is that of A*, the deployment of a sensor in pursuit mode took between 0.078 and 0.2350 sec in a connectivity graph with $n_{\mathcal{G}} = 340$ and $n_{\mathcal{G}_a} = 338$, and between 1.672 and 60.125 sec in a graph with $n_{\mathcal{G}} = 9{,}590$ and $n_{\mathcal{G}_a} = 32{,}687$.

5. Simulation results. In order to validate the methodology developed in section 3, a MATLAB simulator has been developed. We integrate the information-driven sensor planning and pursuit strategies described in previous sections into several simulation scenarios.

5.1. Scenario 1: Multiple static sensors and one pursuer. In many surveillance applications static sensor networks can be used with a few motion-enabled sensors. Static sensors are placed to optimally cover a given area [17]. If a target (evader) is detected, then a mobile sensor can be sent to investigate or capture the target. This scenario is a special case of the pursuit-evasion problem addressed in this work. The simulation results are depicted in Figure 5.1. This scenario includes obstacles and the use of the reward function (3.1).

5.2. Scenario 2: Multiple mobile sensors and targets. This simulation scenario extends the first by considering the same environment but with multiple targets. Before the simulation scenario begins, five sensors with platforms measuring 0.25 m square are placed in the 10m-by-10m environment to maximize the probability of detecting tracks with $k = 2$, since we require this number of detections to form a partially observed track. Obstacle and coverage maps are generated for each sensor corresponding to the placement in each cell. Figure 5.2 shows the initial environment and the five sensors—one with sensing radius 1.5 m, one with sensing radius 1.25 m, and three with sensing radii of 1 m. Initially, all sensors are in *detection* mode, and each is a candidate to switch to the *pursuit* mode when target tracks become fully observed.

In this scenario, two targets enter the environment at different locations and headings and with different velocities. As they move along their trajectories, they are detected by the sensors (Figure 5.3). The sensors remain motionless since each target has been detected only once. After the second detection of a target, the network hypothesizes the target track based on previous detections and deploys the sensor which receives the highest reward (or lowest cost) as obtained by the A* graph search algorithm to move to obtain an additional detection of the target (Figure 5.4). When the second target becomes partially detected, the same track hypothesis and sensor deployment occurs. At the point that the first target's track becomes fully observed (see Figure 5.5), the network again evaluates the reward (distance) and deploys the best sensor to pursue the target. The same pursuit is performed when the second

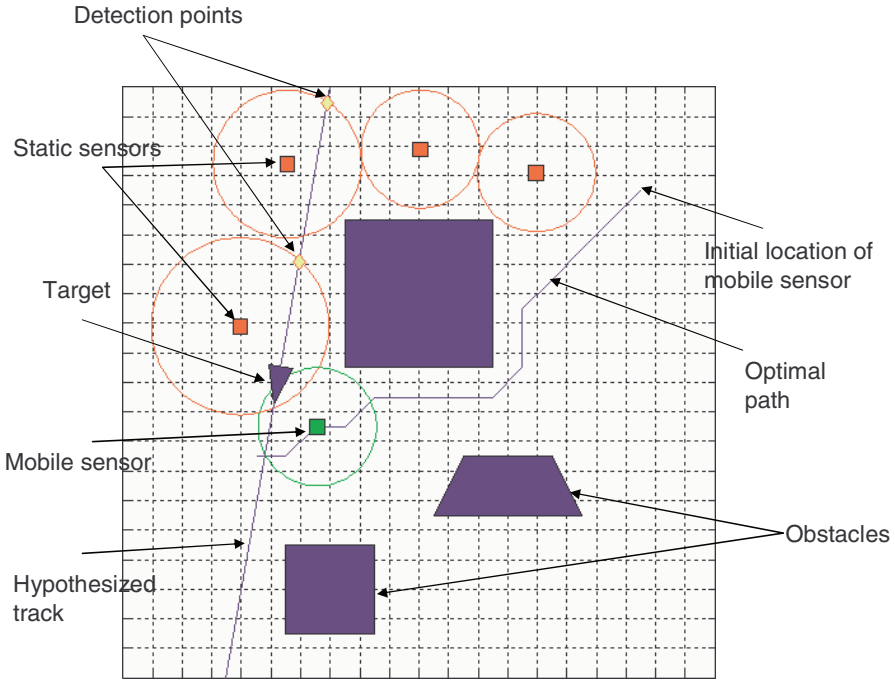

FIG. 5.1. *Static multiple detectors and one pursuer.*

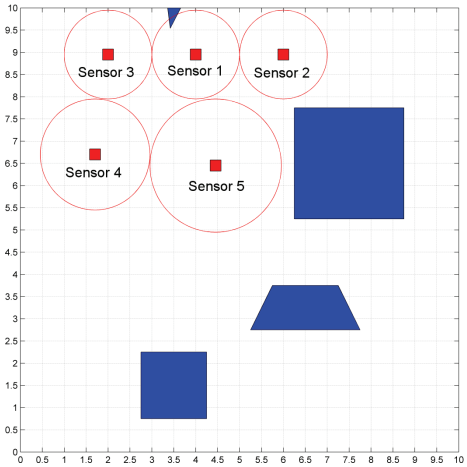

FIG. 5.2. *Initial sensor placement.*

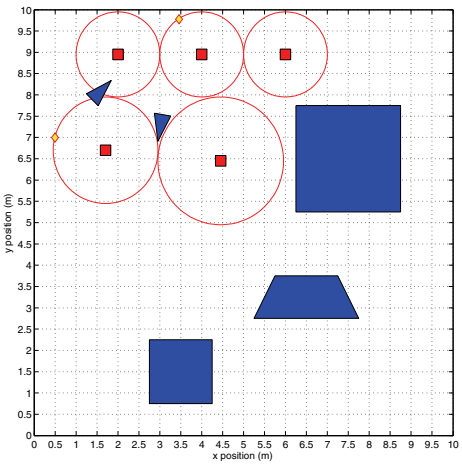

FIG. 5.3. *Two targets each detected once.*

target is fully observed, as shown in Figure 5.6. The state of the network following capture of all known targets is depicted in Figure 5.7. The network is rearranged to maximize area coverage at the next recalculation interval. Table 5.1 summarizes the chronology of the main events which occur during the simulation. Algorithm 1 illustrates how the simulation scenario has been implemented.

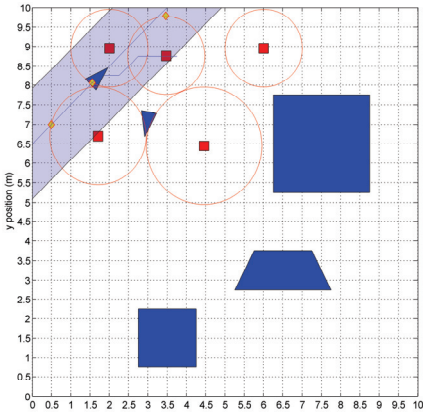

FIG. 5.4. *Target* 1 *is partially observed with its hypothesized track, and Sensor* 1 *is deployed to obtain additional observations.*

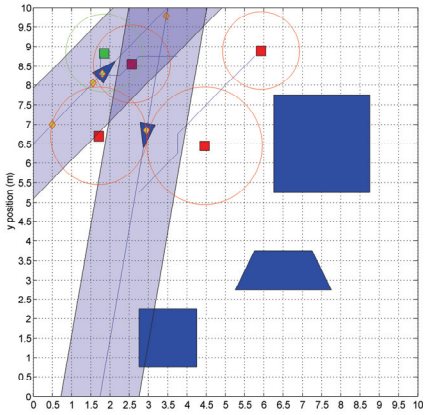

FIG. 5.5. *Target* 1 *is fully observed and Sensor* 3 *is deployed to pursue it while Target* 2 *becomes partially observed, and Sensor* 1 *is deployed to obtain an additional observation.*

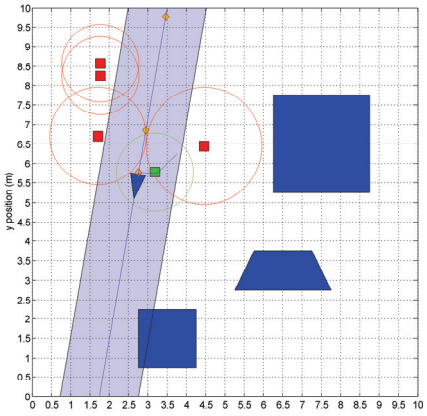

FIG. 5.6. *Target* 1 *has been captured. Target* 2 *is fully observed and is pursued by Sensor* 1.

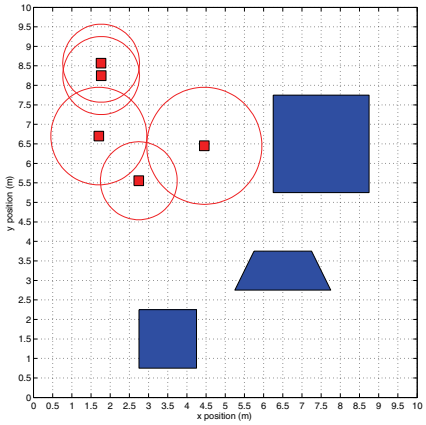

FIG. 5.7. *Final sensor arrangement after both targets are captured.*

TABLE 5.1
Simulation events of Scenario 2.

Event	Time (s)	Position (m)	Sensor	Target
Detect	0.40	(3.46,9.78)	1	2
Detect	1.70	(0.49,6.99)	4	1
Detect	5.45	(1.56,8.06)	3	1
Deploy	5.45	(1.75,8.25)	1	1
Detect	6.30	(1.80,8.30)	1	1
Pursue	6.30	-	3	1
Detect	6.35	(2.94,6.85)	4	2
Deploy	6.35	(2.75,5.25)	2	2
Capture	6.60	(1.76,8.57)	3	1
Detect	8.55	(2.75,5.77)	2	2
Pursue	8.55	-	2	2
Capture	9.40	(2.75,5.55)	2	2

Algorithm 1. Scenario 2 Algorithm.

```
Perform initial optimal sensor placement
Decompose environment into C_free and C_obstacle cells
for all Sensors do
   Calculate obstacle map
   Calculate coverage map
end for
while Game not over do
   for all Sensors in pursuit do
      if Pursued target beneath capture threshold then
         Remove target
         End pursuit
      end if
   end for
   if Detection then
      if Target detections = 2 then
         Hypothesize target track
         Calculate observation cells
         for all Sensors that have not detected this target do
            Calculate path and reward to investigate target
         end for
         Deploy the sensor with the greatest reward
      else if Target detections = 3 then
         for all Sensors not in pursuit do
            Calculate path and reward to pursue target
         end for
         Deploy the sensor with the greatest reward
      end if
   end if
   if Sensor update interval then
      for all Sensors do
         Calculate coverage map
      end for
      Deploy next sensor to maximize coverage
   end if
end while
```

6. Conclusions. This paper presents a novel framework for developing sensor control policies in systems involving multiple robotic platforms that seek to detect and intercept multiple mobile targets. Multiple objectives, such as the probability of detecting unobserved tracks, for which little or no information is available a priori, obstacle avoidance, and the probability of detection associated with partially observed targets are approached using a geometric approach. The path leading to the optimal trade-off between these objectives is obtained through the A* graph searching algorithm and is passed to a control strategy that accounts for the actual pursuers' dynamics. By adopting a track-before-detect approach, a target is declared positively detected once a satisfactory number of detections k may be used to form a consistent track. Subsequently, a heuristic rule switches one of the mobile sensors from *detection* mode to *pursuit* mode, and the track is readily available to compute an optimal pursuit

strategy. By maximizing the same reward function, the remaining sensors in *detection* mode are reconfigured such that the probability of detecting the remaining targets is again optimized. The progressive simulation scenarios presented validate the developed methodology. The future work of this approach will include fully implementing the methodology on a multivehicle testbed [13]. Additionally, the approach will be extended to reflect a more general pursuit-evasion game by considering intermittent communication among pursuers and targets, intermittent estimation, and intelligent evaders that are not restricted to moving in straight lines.

Appendix A. Proof of Proposition 3.5. This proof considers a family of $k = 3$ nontranslates $D_k = \{\mathcal{D}_i, \mathcal{D}_j, \mathcal{D}_l\}$ with index set $I_{D_k} = \{i, j, l\}$. The results can be extended to higher k by induction. The coverage cone $K(\mathcal{D}_\ell, b_y)$ contains the set of all rays that intersect $\mathcal{D}_\ell$ in $\mathbb{R}^2_+$, where $\ell \in I_{D_k}$. Then, the set of tracks intersecting all circles in the family D_k is given by the following intersection:

$$K_k(D_k, b_y) = \bigcap_{\ell \in I_{D_k}} K(\mathcal{D}_\ell, b_y) = K(\mathcal{D}_i, b_y) \cap K(\mathcal{D}_j, b_y) \cap K(\mathcal{D}_l, b_y). \tag{A.1}$$

From the properties of cones [6, p. 70], the intersection of a collection of cones is also a cone, and thus $K_k(D_k, b_y)$ is a cone. A vector z representing a ray $\mathcal{R}_\theta$ with the same slope and origin lies in a cone K if and only if $\mathcal{R}_\theta$ lies in K, since any point on $\mathcal{R}_\theta$ can be written as cz, with $c > 0$.

Consider a ray $\mathcal{R}_\theta \in K(\mathcal{D}_\ell, b_y)$, where $K(\mathcal{D}_\ell, b_y) = cone(\hat{l}_\ell, \hat{h}_\ell)$ and thus can be represented by a vector $z_\ell = c_1\hat{l}_\ell + c_2\hat{h}_\ell$ with constants $c_1, c_2 > 0$. Then, $z_\ell \in K(\mathcal{D}_\ell, b_y)$ and, by the properties of vector sum, $\hat{l}_\ell \prec z_\ell \prec \hat{h}_\ell$. Next, consider a cone $K^* = cone(\hat{l}^*, \hat{h}^*)$ that is finitely generated by two unit vectors $\hat{h}^* = \hat{h}_\jmath$ and $\hat{l}^* = \hat{l}_\imath$ with $\jmath, \imath \in I_{D_k}$, and assume $\hat{l}_\imath \prec \hat{h}_\jmath$. By the properties of finitely generated cones [6], any vector $z^* = b_1\hat{l}^* + b_2\hat{h}^*$ with constants $b_1, b_2 > 0$ must lie in K^*. It follows that a ray $\mathcal{R}^*_\theta$ with the same slope and origin as z^* must also lie in K^*, since any point on $\mathcal{R}^*_\theta$ can be written as $cz*$, with $c > 0$. Since z^* is a positive combination of $\hat{l}^*$ and $\hat{h}^*$, it also follows that $\hat{l}^* \prec z^* \prec \hat{h}^*$.

According to Proposition 3.5, choose $\hat{h}^* = \hat{h}_\jmath \preceq \hat{h}_\ell$ and $\hat{l}^* = \hat{l}_\imath \succeq \hat{l}_\ell \ \forall \ell \in I_{D_k}$. Suppose that the unit vectors of D_k can be ordered as $\hat{h}_l \prec \hat{h}_j \prec \hat{h}_i$ and $\hat{l}_i \prec \hat{l}_l \prec \hat{l}_j$. Then, the unit vectors and z^* can be ordered as follows:

$$\hat{l}_\ell \preceq \hat{l}_j = \hat{l}^* \prec z^* \prec \hat{h}^* = \hat{h}_l \preceq \hat{h}_\ell \qquad \forall \ell \in \{i, j, l\} = I_{D_k} \tag{A.2}$$

or, more explicitly,

$$\hat{l}_i \prec \hat{l}_l \prec \hat{l}_j = \hat{l}^* \prec z^* \prec \hat{h}^* = \hat{h}_l \prec \hat{h}_j \prec \hat{h}_i. \tag{A.3}$$

Since the above order also implies $\hat{l}_\ell \prec z^* \prec \hat{h}_\ell \ \forall \ell \in I_{D_k}$, then $z^*, \mathcal{R}^*_\theta \in K(\mathcal{D}_\ell, b_y)$ $\forall \ell \in I_{D_k}$. Thus, from (A.1), $z^*, \mathcal{R}^*_\theta \in K_k(D_k, b_y) = K^* = cone(\hat{l}^*, \hat{h}^*)$, provided that $\hat{h}^*$ and $\hat{l}^*$ are chosen subject to (A.2).

So far it has been assumed that $\hat{l}_\imath \prec \hat{h}_\jmath$. If the unit vectors of D_k are such that $\hat{l}_\imath \succ \hat{h}_\jmath$, then there are no vectors that can satisfy the order $\hat{l}_\imath = \hat{l}^* \prec z^* \prec \hat{h}^* = \hat{h}_\jmath$, and $K_k(D_k, b_y) = K^* = \varnothing$.

Appendix B. Proof of Theorem 3.6. The set of all tracks through a y-intercept b_y that are detected by at least k sensors in $D = \{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$ is the union

of the k-coverage cones of all k-subsets of D,

$$\mathcal{K}_k(D, b_y) = \bigcup_{j=1}^{m} K_k(D_k^j, b_y), \quad m = \begin{pmatrix} N \\ k \end{pmatrix}. \tag{B.1}$$

D_k^j denotes the jth k-subset of D, and the number m of possible k-subsets is given by the binomial coefficient N *choose* k. Since $\mathcal{K}_k(D, b_y)$ is a union of possibly disjoint cones, it may not be a cone [6]. Nevertheless, the same Lebesgue measure defined for a cone, μ on $[0,\ \pi]$, can be applied to it using the principle of inclusion-exclusion [43]

$$\begin{aligned} \mu(\mathcal{K}_k(D, b_y)) &= \mu\left(\bigcup_{j=1}^{m} K_k(D_k^j, b_y)\right) \\ &= \sum_{j=1}^{m} (-1)^{j+1} \sum_{1 \le i_1 < \cdots < i_j \le m} \mu(K_k(D_k^{i_1}, b_y) \cap \cdots \cap K_k(D_k^{i_j}, b_y)), \end{aligned} \tag{B.2}$$

where $m = \binom{N}{k} = \frac{N!}{(N-k)!\ k!}$ and $\sum_{1 \le i_1 < \cdots < i_j \le m}$ is a sum over all the $[m!/(m-j)!\ j!]$ distinct integer j-tuples $(i_1, \ldots, i_j)$ satisfying $1 \le i_1 < \cdots < i_j \le m$. Also, $\mu(\cdot)$ denotes a measure on the set. Since the right-hand side of (B.2) is an intersection of cones, it also is a cone on which we can impose μ. Moreover, it represents the set of tracks through b_y that intersect all sensors in $D_p^{i_1,j} = \{D_k^{i_1} \cup \cdots \cup D_k^{i_j}\}$. Based on the properties of k-subsets, $D_p^{i_1,j}$ must contain $k \le p \le n$ elements of D and, thus, is a p-subset of D. Based on the properties of k-coverage cones (Proposition 3.5), the set of line transversals of $D_p^{i_1,j}$ through b_y can be represented by the p-coverage cone $K_p(D_p^{i_1,j}, b_y)$. Thus, (B.2) can be written as

$$\begin{aligned} \mu(\mathcal{K}_k(D, b_y)) &= \sum_{j=1}^{m} (-1)^{j+1} \sum_{1 \le i_1 < \cdots < i_j \le m} \mu(K_p(D_p^{i_1,j}, b_y)) \\ &= \sum_{j=1}^{m} (-1)^{j+1} \sum_{1 \le i_1 < \cdots < i_j \le m} \psi(D_p^{i_1,j}, b_y), \end{aligned} \tag{B.3}$$

by Proposition 3.5, where p is the number of elements in the union of j k-subsets of D, and the opening angles $\psi(D_p^{i_1,j}, b_y)$ in the above summation are given by (3.9).

Now, let $\mathcal{R}_\theta(b_y)$ denote a ray with y-intercept b_y and heading angle $\theta_\tau \in (-\pi/2,\ +\pi/2)$. Suppose $\mu(\mathcal{K}_k(D, b_y)) = \pi$; then $\mathcal{R}_\theta(b_y)$ will be detected by k pursuers with probability one. Assuming that all headings are equally likely, if $0 \le \mu(\mathcal{K}_k(D, b_y)) \le \pi$, then $\mathcal{R}_\theta(b_y)$ will be detected by k pursuers with probability $\mu(\mathcal{K}_k(D, b_y))/\pi$. Let $P(b_y)$ denote the prior probability that a target enters $\mathcal{S}$ at b_y. Assuming that all y-intercepts are equally likely, $P(b_y) = \delta b/(L + \delta b)$. Since the two events are independent, the probability that an unobserved track is $\mathcal{R}_\theta^j(b_y)$ and is detected by k pursuers is given by the product of the individual probabilities

$$\Pr\{D_{jk} = 1, \mathcal{R}_\theta^j(b_y)\} = \frac{\delta b}{(L + \delta b)} \frac{\mu(\mathcal{K}_k(D, b_y))}{\pi}, \tag{B.4}$$

where D_{jk} denotes the event that the jth target traversing $\mathcal{S}$ along an unobserved track is detected by at least k pursuers. Since the k-coverage cones with different

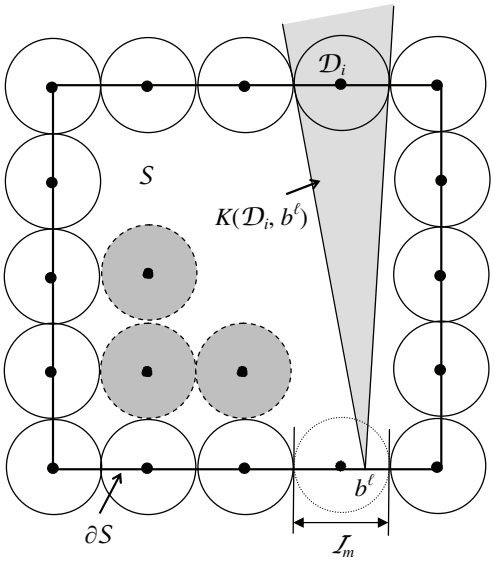

FIG. B.1. *Examples of total coverage configuration for $k = 2$ (sensors on $\partial\mathcal{S}$), sensors packed by the triangular number (grey disks), and coverage cone $K_2 < \pi$ when $N < 2L/r$.*

y-intercepts are disjoint, the probability that a target's point of entry in $\mathcal{S}$ is the ℓth intercept $b_y^\ell \in \partial\mathcal{S}$ and that it is detected by k pursuers is

(B.5)
$$\Pr\{D_{jk} = 1, \mathcal{R}_\theta^j(b_y^\ell \in \partial\mathcal{S})\} = \sum_{\ell=0}^{L/\delta b} \frac{\delta b}{\pi(L+\delta b)} \sum_{j=1}^{m} (-1)^{j+1} \sum_{1 \le i_1 < \cdots < i_j \le m} \psi(D_p^{i_1,j}, b_y^\ell).$$

The probability that $D_{jk} = 1$ and that the target's point of entry is on one of the other axes can be obtained by the same approach, using the opening angles of the corresponding coverage cones. The set of tracks that traverse $\mathcal{S}$ and are detected by at least k pursuers is given by the probability of the union of intersecting sets $\mathcal{K}_k(D, b_y)$, $\mathcal{K}_k(D, b_x)$, $\mathcal{K}_k(D, b_{y'})$, and $\mathcal{K}_k(D, b_{x'})$ that are obtained by applying (B.1) to intercepts on the y, x, y', and x' axes, respectively. The final probability density function $P_{\mathcal{S}}^k(\mathcal{X})$ in (3.10) is obtained by observing that every track in this union intersects two sides of $\mathcal{S}$, and that the indices in the second summation must be shifted in order to consider intercepts at the corners only once.

Consider now the case of a *total coverage configuration*, denoted by $\mathcal{X}_{tot}^k$, that detects all tracks in $\mathcal{S}$ at least k times. We want to show that for $\mathcal{X}_{tot}^k$ the probability density $P_{\mathcal{S}}^k$ in (3.10) is equal to one (its upper bound). In large sensor networks total coverage may be obtained by concentric configurations placed on and around $\partial\mathcal{S}$, as shown by the example in Figure B.1 with $k = 2$. The Lebesgue measure (or opening angle) of a finitely generated cone $K(\mathcal{D}_i, b^\ell) \in \mathbb{R}_+^2$, with origin b^ℓ, is bounded between 0 and π, and it is equal to π only when $b^\ell \in \mathcal{D}_i$. The Lebesgue measure on a union of cones, $\mu(\mathcal{K}_k(D, b^\ell))$, attains its upper bound π when all tracks through b^ℓ intersect k disks in D, and it is independent of k and N because all k-coverage cones are finitely generated. Then, the probability of detection for $\mathcal{X}_{tot}^k$ is obtained from (3.10) by letting $\mu(\mathcal{K}_k(D, b_y^\ell)) = \mu(\mathcal{K}_k(D, b_x^\ell)) = \mu(\mathcal{K}_k(D, b_{y'}^\ell)) = \mu(\mathcal{K}_k(D, b_{x'}^\ell)) = \pi \ \forall \ell$; i.e.,

$$P_{\mathcal{S}}^{k}(\mathcal{X}_{tot}^{k}) = \frac{\delta b}{4\pi L}\sum_{\ell=1}^{L/\delta b}(\pi+\pi) + \frac{\delta b}{4\pi L}\sum_{\ell=0}^{(L/\delta b-1)}(\pi+\pi)$$
$$= \frac{\delta b}{4\pi L}\left[\frac{L}{\delta b}2\pi + \frac{L}{\delta b}2\pi\right] = 1.$$

Appendix C. Performance analysis. Since every target remains in $\mathcal{S}$ and maintains $\mathcal{R}_\theta^j$ for the duration of the game and $V_{p_{\max}} > V_{\tau_{\max}}$, every target $j \in \mathcal{T}$ can be captured by a pursuer $j \in \mathcal{P}$ in time $t_{c_i^j}$, using the pursuit strategy in section 3.3, provided that $\mathcal{R}_\theta^j$ is fully observed. If $N \geq \lfloor 2L/r \rfloor$, an initial network configuration $\mathcal{X}_0^*$, defined in (3.2), with $\mathcal{P}_{\mathcal{S}}^2(\mathcal{X}_0^*) = 1$ can be obtained by solving a nonlinear program in which (3.10) is maximized with respect to $\mathcal{X}$, subject to $0 \leq x_p^i \leq L$ and $0 \leq y_p^i \leq L$ $\forall i \in I_{\mathcal{P}}$ [4]. As shown in Appendix B, in this total coverage configuration the network obtains at least two detections per target with probability one. After two detections are obtained, $\mathcal{R}_\theta^j$ is estimated and declared partially observed. Then, during every subsequent round, one sensor is moved from a cell κ_l to a cell $\kappa_\imath$ ($\kappa_l \to \kappa_\imath$), only if the overall probability of detection of the network, $(P_{\mathcal{R}}(\kappa_\imath) + \Delta P_{\mathcal{S}}^k(\kappa_l, \kappa_\imath))$, is increased by the change in configuration ($\mathcal{X}_l \to \mathcal{X}_\imath$). Since the probability density functions $P_{\mathcal{R}}$ and $P_{\mathcal{S}}^k$ are defined over the same set of targets $\mathcal{T}$ and the same search area $\mathcal{S}$, the probability of obtaining the $M(k-2)$ additional sensor detections required to declare all M target tracks fully observed increases at every round, provided that there are at least $(k-2)$ sensors in the network to obtain at least $(k-2)$ distinct detections per target. After a track $\mathcal{R}_\theta^j$ is fully observed, target j can always be captured by any sensor in the network, because $V_{p_{\max}} > V_{\tau_{\max}}$. Since, in the worst case, one sensor can be used to pursue every target sequentially, the game is guaranteed to terminate if $N \geq N_{\min}$, where

(C.1)
$$N_{\min} = \max\left\{\left\lfloor\frac{2L}{r}\right\rfloor + 1, k-2, 1\right\} = \frac{1}{2}\left[\left\lfloor\frac{2L}{r}\right\rfloor + 1 + k - 2 + \left|\left\lfloor\frac{2L}{r}\right\rfloor + 1 - k + 2\right|\right],$$

which simplifies to (4.1).

If $N < N_{\min}$, it can be shown by contradiction that the game cannot be guaranteed to terminate because there is a subset of tracks that may never be fully observed. From (C.1), there are two possible cases, namely, $N_{\min} = \lfloor 2L/r \rfloor + 1$ or $N_{\min} = k-2$, depending on the problem's parameters. In the first case, from the properties of the floor function, if $N < \lfloor 2L/r \rfloor + 1$, then $\lfloor N \rfloor < \lfloor 2L/r + 1 \rfloor$, and it follows that $N < 2L/r$, because N is an integer and $2L/r$ is a rational number (with the notable exception that if $2L/r$ is an integer, then $N_{\min}$ should be decreased by one). It also follows that $2Nr < 4L$, and thus for any $\mathcal{X}_0^*$ there exists at least one interval $\mathcal{I}_m = \{b \mid b \in \partial\mathcal{S},\ b \cap \mathcal{D}_i = \varnothing\ \forall i \in I_D\}$, as illustrated in Figure B.1, where if the dotted disk is removed, $N < 2L/r$. From Appendices A and B, the Lebesgue measure (or opening angle) μ of a coverage cone $K(\mathcal{D}_i, b)$ attains the upper bound π if and only if $b \in \mathcal{D}_i$. From (A.1), any $(k=2)$-coverage cone $K_2(D_2^j, b)$ is the intersection of two coverage cones, e.g., $K(\mathcal{D}_i, b)$ and $K(\mathcal{D}_l, b)$, where $D_2^j = \{\mathcal{D}_i, \mathcal{D}_l\} \subset D$, and thus $\mu(K_2(D_2^j, b)) \leq \min\{\mu(K(\mathcal{D}_i, b)), \mu(K(\mathcal{D}_l, b))\}$. It follows that for any intercept value $b^\ell \in \mathcal{I}_m$, $\mu(K(\mathcal{D}_i, b^\ell)) < \pi\ \forall i \in I_D$, and thus $\mu(K_2(D_2^j, b^\ell)) < \pi\ \forall j$, where K_2 is the $(k=2)$-coverage cone for any 2-subset of D, as defined in (D.1). Thus, by Proposition 3.5, the set of tracks in the complement set $\mathcal{K}_m(b^\ell) = \mathcal{S} \setminus K_2(D_2^j, b^\ell)$, comprised of a union of cones, is detected by at most one sensor in $\mathcal{X}_0^*$ (Figure B.1).

Now, let $M = 1$ and assume that the target has a track $\mathcal{R}_\theta^m(b^\ell) \in \mathcal{K}_m(b^\ell)$, through an intercept $b^\ell \in \mathcal{I}_m$ (Figure B.1). Then, the target can be detected by at most one sensor in $\mathcal{X}_0^*$ and remains unobserved by definition. Since there are no other targets in $\mathcal{S}$, at every subsequent round in the game all sensors remain in detection mode, and all nodes in $\mathcal{G}$ remain void cells such that $P_{\mathcal{R}}(\kappa_\iota) = 0\ \forall \kappa_\iota \in \mathcal{G}$. From (3.2), $\mathcal{X}_0^*$ is the configuration with the maximum value of $P_{\mathcal{S}}^2$. Thus, by its choice of κ_f for sensors in detection mode (section 3), the algorithm holds the sensors stationary at $\mathcal{X}_0^*$ at every round, and the game never terminates, because for this configuration $\mathcal{R}_\theta^m(b^\ell)$ always is unobserved. In the second case, $N_{\min} = k - 2 > \lfloor 2L/r \rfloor + 1$. Therefore, when $N < N_{\min}$, there may be a sufficient number of sensors in the network to obtain at least two detections per target and partially observe all tracks. However, after the tracks have been partially observed, there are not enough sensors to obtain the additional $k - 2$ independent detections required to declare any track fully observed. Thus, all sensors remain in detection mode, and the game never terminates because the targets are never captured. It can be concluded that if $N < N_{\min}$, the game cannot be guaranteed to terminate.

Based on the problem formulation in section 2, the time required to terminate the game T depends on the time required to obtain two detections per unobserved track, ΔT_d; the time required to obtain $1 < l < k$ additional detections per partially observed track, ΔT_o; and the time required to pursue every target after its track has been fully observed, ΔT_c. Although the method in section 3 allows different targets to be simultaneously detected and pursued, the worst-case scenario is one where there is no overlap between these time periods, i.e., $T = \Delta T_d + \Delta T_o + \Delta T_c$.

After all targets are in $\mathcal{S}$, the maximum time required by a network with $N_{\min}$ sensors, positioned at $\mathcal{X}_0^*$, to obtain at least two detections per target is the time required by the slowest target to travel the longest distance in $\mathcal{S}$ between two disks $\mathcal{D}_i$ and $\mathcal{D}_j$ with p_i and p_j at opposite corners, i.e., $\Delta T_d(N_{\min}) = (\sqrt{2}L - 2r)/V_{\tau_{\min}}$. Consider now a network with $N \geq N_{\min}$ sensors. Before obtaining any detections, it can be easily shown that the configuration that maximizes the distance between any two sensors is one where they are packed concentrically in $\mathcal{S}$ (e.g., the grey disks in Figure B.1). By the definition of a triangular number, this configuration can be achieved by increasing the number of sensors according to

$$N = \ell \left\lfloor \frac{2L}{r} \right\rfloor - 8\mathrm{T}_{(\ell-1)} = \ell \left\lfloor \frac{2L}{r} \right\rfloor - 4\ell(\ell-1), \quad \ell = 1, \ldots, \left\lfloor \frac{L}{4r} \right\rfloor, \tag{C.2}$$

where ℓ is an integer and $\lfloor L/4r \rfloor$ is the maximum value of ℓ that allows one to pack N sensors in $\mathcal{S}$. T_n denotes the triangular number of n and is used to represent the decrease in the number of sensors that can be placed at the corners as N increases (see Figure B.1). Then, the maximum time required to obtain two distinct detections is the time required by the slowest target to travel the maximum distance between any two sensors, that is,

$$\begin{aligned}
\Delta T_d(N) &= \frac{1}{V_{\tau_{\min}}} \max\left\{2r, \sqrt{2}L - 2r[2\sqrt{2}(\ell-1)+1]\right\} \\
&= \frac{1}{V_{\tau_{\min}}} \frac{1}{2} \left\{2r + \sqrt{2}L - 2r[2\sqrt{2}(\ell-1)+1] - \left|2r - \sqrt{2}L + 2r[2\sqrt{2}(\ell-1)+1]\right|\right\} \\
&= \frac{1}{V_{\tau_{\min}}} \left\{\frac{\sqrt{2}}{2}L - 2\sqrt{2}r(\ell-1) + \left|2r[1+\sqrt{2}(\ell-1)] - \frac{\sqrt{2}}{2}L\right|\right\}.
\end{aligned} \tag{C.3}$$

After all targets have been partially observed, $(k-2)$ additional detections per target are required to declare them fully observed (where it is assumed that $k > 2$). If there are $N_{\min}$ sensors in the network, this can always be accomplished by repeatedly moving the sensors until all M target tracks are fully observed. Since the maximum distance that must be traveled by any of the sensors to obtain an additional detection is $(\sqrt{2}L - r)$, the time required to fully observe all M target tracks is

$$\Delta T_o(N_{\min}) = \left[\left\lfloor \frac{(k-2)M}{N_{\min}} \right\rfloor + 1\right] \frac{(\sqrt{2}L - r)}{\bar{V}_p}. \tag{C.4}$$

However, if at least $(k-2)M$ sensors are available, they can all be moved at once to each obtain one additional detection, and, assuming that the running time to reconfigure them is negligibly small, all M target tracks are fully observed after a time $(\sqrt{2}L - r)/\bar{V}_p$.

Suppose that there are $\lfloor 2L/r \rfloor \leq N \geq M$ sensors available to pursue the targets after they have been fully observed. Then, the maximum distance that must be traveled to capture target j by switching the nearest sensor to pursuit mode is the distance between the sensor that has obtained the last kth detection from j and the interception point δ. For the purpose of analysis, one can place an inertial frame of reference at $\tau_j(t_k)$ and align it with the target track such that $x_\tau^j(t_k) = y_\tau^j(t_k) = \theta_\tau^j = 0$. Then, at time t_k the sensor must be within a distance r from the target position, with a heading θ_p. From (3.16), assuming that the turn time is negligibly small, the capture time t_c must satisfy the equality

$$\begin{aligned}(V_{p_{\max}} t_c)^2 &= r^2 \cos^2\theta_p + (\bar{V}_\tau t_c)^2 - 2t_c r \bar{V}_\tau \cos\theta_p + r^2 \sin^2\theta_p \\ &= (\bar{V}_\tau t_c)^2 - 2t_c r \bar{V}_\tau \cos\theta_p + r^2 \end{aligned} \tag{C.5}$$

for any θ_p, since the sensors are assumed to travel at their maximum speed when in pursuit mode (section 3.3). Differentiating (C.5) with respect to θ_p and setting the result equal to zero, it can be shown that the heading with maximum capture time is $\theta_p^* = \pi$ with respect to the target track. Thus, the maximum capture time can be obtained from

$$t_c^* = \frac{1}{V_{p_{\max}}} \left\| \begin{array}{c} (r\cos\theta_p^* - t_c^* \bar{V}_\tau) \\ r \sin\theta_p^* \end{array} \right\| \tag{C.6}$$

by letting $\theta_p^* = \pi$ and solving for $t_c^* = r/(V_{p_{\max}} - \bar{V}_\tau)$. Since all of the M sensors that obtained the last kth detection from the M targets can be deployed simultaneously to pursue them, the maximum time required to capture them is $\Delta T_c(N) = r/(V_{p_{\max}} - \bar{V}_\tau)$.

If there are only $N_{\min}$ sensors in the network, they need to be deployed $(\lfloor M/N_{\min} \rfloor + 1)$ times in order to capture all M targets that have been fully observed. The first time, the $N_{\min}$ sensors that obtained the last kth detection from $N_{\min}$ targets are deployed in $r/(V_{p_{\max}} - \bar{V}_\tau)$ maximum time, as shown in the previous paragraph. However, when the sensors are subsequently redeployed, they could be anywhere in $\mathcal{S}$, since they reenter the game after having captured other targets. It can be shown by solving a simple constrained-optimization problem (omitted here for brevity) that for a sensor and a target located anywhere in $\mathcal{S}$ the maximum capture time is $L[\bar{V}_\tau + (2V_{p_{\max}}^2 - \bar{V}_\tau^2)^{1/2}]/(V_{p_{\max}}^2 - \bar{V}_\tau^2)$. Thus, it follows that the maximum capture time with $N_{\min}$ sensors is

$$\Delta T_c(N_{\min}) = \frac{r}{(V_{p_{\max}}^2 - \bar{V}_\tau^2)} + \left\lfloor \frac{M}{N_{\min}} \right\rfloor \frac{\left(\bar{V}_\tau + \sqrt{2V_{p_{\max}}^2 - \bar{V}_\tau^2}\right)}{(V_{p_{\max}}^2 - \bar{V}_\tau^2)^2} L. \tag{C.7}$$

From the above analysis, the minimum number of sensors $N_{\min}$, obtained by the maximum of all minima in (4.1), is required to terminate the game in a maximum time T_u in (4.2). Also, the maximum time required to end the game can be reduced to T_ℓ, in (4.5), by increasing the number of sensors to the maximum of all maxima,

$$N_\ell = \max\left\{\ell\left\lfloor\frac{2L}{r}\right\rfloor - 4\ell(\ell-1), (k-2)M, M\right\}, \quad \ell = 1,\ldots,\left\lfloor\frac{L}{4r}\right\rfloor, \tag{C.8}$$

which simplifies to (4.4), since $k > 2$.

Appendix D. Complexity analysis. During every detection round, the main computations required by the methodology in section 3 are the cell-decomposition procedure to obtain $\mathcal{G}$, the computation of the sensing reward (3.1) for every observation cell in $\mathcal{G}$, and the search for the optimal channel (3.3), by the A* algorithm. Therefore, we first analyze the complexity of the cell-decomposition procedure. Since the sensor field-of-view $\mathcal{D}_i$ is a disk, the decomposition may involve generalized polygons. To avoid this case, $\mathcal{D}_i$ can be approximated by a regular octagon $\hat{\mathcal{D}}_i$ tightly contained in $\mathcal{D}_i$. For simplicity, θ_p^i is assumed fixed. The obstacle-free configuration space $\mathcal{C}_{free}^i$ for the ith pursuer $\mathcal{A}_i$ can be decomposed via the method in section 3.2. The position $p_i = (x_p^i,\ y_p^i)$ of $\mathcal{A}_i$ is abbreviated here by $(x,\ y)$, and $\mathcal{C}$ denotes the robot configuration space, which can be assumed to be a rectangle or a union of rectangles. Let $\kappa = [x_\kappa, x'_\kappa] \times [y_\kappa, y'_\kappa]$ denote a rectangle in $\mathbb{R}^2$. From the problem formulation, $\mathcal{A}_i$, $\hat{\mathcal{D}}_i$, and $\mathcal{O}_k$, $k = 1,\ldots,n$, are assumed to be convex polygons, and targets $\mathcal{R}_j$, $j = 1,\ldots,l$, are rays in the workspace $\mathcal{S}$; then it follows that $\mathcal{CO}_k$, $k = 1,\ldots,n$, and thus $\mathcal{CR}_j$, $j = 1,\ldots,l$, all are convex [29]. Let $n_{e_\mathcal{A}}$ denote the number of edges describing $\mathcal{A}_i$, which is assumed to be a fixed constant. The number of edges defining sensor $\hat{\mathcal{D}}_i$ is eight. The running time to compute the two-dimensional C-obstacle $\mathcal{CO}_k$, for all $k = 1,\ldots,n$, in $\mathcal{S}$ is $O(n_{e_\mathcal{O}})$. The running time to compute $\mathcal{CR}_j$, for all $j = 1,\ldots,l$, in $\mathcal{S}$ is $O(n_{e_\mathcal{R}})$ [29, 31]. The decomposition of the complement of $\mathcal{CO}_k$, $k = 1,\ldots,n$, and $\mathcal{CR}_j$, $j = 1,\ldots,l$, into convex cells is a vertical decomposition in two dimensions [29, 31]. Thus, the complexity of the decomposition of $\mathcal{C}_{void}^i$, in step (I), is $O((n_{e_\mathcal{O}} + n_{e_\mathcal{R}})\log(n_{e_\mathcal{O}} + n_{e_\mathcal{R}}))$. In step (II), the decomposition of $\mathcal{CR}_j \setminus \bigcup_{j=1}^n \mathcal{CO}_k$, for each $j = 1,\ldots,p$, can be also implemented via vertical decomposition in two dimensions [29, 31] with complexity $O(n_{e_\mathcal{O}} \log n_{e_\mathcal{O}})$. Thus, the complexity of step (II) is $O(n_{e_\mathcal{R}} n_{e_\mathcal{O}} \log n_{e_\mathcal{O}})$. Therefore, the overall complexity of the cell-decomposition method is $O((n_{e_\mathcal{O}} + n_{e_\mathcal{R}})\log(n_{e_\mathcal{O}} + n_{e_\mathcal{R}})) + O(n_{e_\mathcal{R}} n_{e_\mathcal{O}} \log n_{e_\mathcal{O}}) = O((n_{e_\mathcal{O}}+n_{e_\mathcal{R}})\log(n_{e_\mathcal{O}}+n_{e_\mathcal{R}})+n_{e_\mathcal{R}} n_{e_\mathcal{O}} \log n_{e_\mathcal{O}})$, whereas the complexity of a decomposition involving only obstacles is $O(n_{e_\mathcal{O}} \log(n_{e_\mathcal{O}}))$ [29, 31]. By using the *approximate-and-decompose* method in [52], the cell decomposition in section 3.2 can be run in $O((n_{e_\mathcal{O}} + n_{e_\mathcal{R}})^2)$ [8].

Next, we analyze the complexity of the probability of cooperative detections (3.10), which is clearly the most expensive computation in the sensing reward (3.1). Let $b_y = \ell \cdot \delta b$, where $\ell = 0,\ldots,L/\delta b$. Consider the complexity of computing (B.5) and denote it by Γ_y^k. Then, the time complexity of computing (3.10) is $O(4\Gamma_y^k)$. All tracks $\mathcal{R}_\alpha(b_y)$ detected by a set of k sensors, D_k, are contained by the k-coverage cone of D_k, and those detected by at least k sensors in a set of n sensors, D, are given by the union in (B.1). By computing the geometric union of the convex cones $K_k(D_k^j, b_y), j = 1,\ldots,m$, K_k can equivalently be expressed as a union of $\wp$ disjoint

convex cones,

$$\mathcal{K}_k(D, b_y) = \bigcup_{\jmath=1}^{\wp} cone(\hat{l}_\jmath^*, \hat{h}_\jmath^*), \tag{D.1}$$

where $cone(\hat{l}_\jmath^*, \hat{h}_\jmath^*) \bigcap cone(\hat{l}_\imath^*, \hat{h}_\imath^*) = \varnothing \, \forall \jmath \neq \imath$, where $\imath, \jmath = 1, \ldots, \wp$, and $\hat{l}_\jmath^* \prec \hat{l}_\jmath^* \prec \hat{l}_\imath^* \, \forall \jmath < \imath$. Clearly, $\wp \leq m$, and $\hat{l}_\jmath^* \in \{\hat{l}_j^* | j = 1, \ldots, m\}$ and $\hat{h}_\jmath^* \in \{\hat{h}_j^* | j = 1, \ldots, m\}, \jmath = 1, \ldots, \wp$. Therefore, (B.5) can be obtained by two steps: (i) obtain $K_k(\mathcal{D}, b_y)$ in (D.1), and (ii) obtain (B.5) from the following equations:

$$\Pr\{D_{jk} = 1, \mathcal{R}_\theta^j(b_y^\ell \in \partial\mathcal{S})\} = \sum_{\ell=0}^{(L/\delta b - 1)} \sum_{\jmath=1}^{\wp} \sin^{-1} ||\hat{l}_\jmath^* \times \hat{h}_\jmath^*||. \tag{D.2}$$

The computation of the inner summation above can be performed in two steps. First, for every D_k^j, $j = 1, \ldots, m$, $\forall i \in I_{D_k^j}$, compute $\sin\alpha_i$ in (3.6) and $\cos\alpha_i = (1 - \sin{\alpha_i}^2)^{1/2}$, then compute $\hat{h}_i$ in (3.4) and $\hat{l}_i$ in (3.5); choose optimal $\hat{l}_j^*$ and $\hat{h}_j^*$ so that $\hat{l}_j^* = \sup\{\hat{l}_i | i \in I_{D_k^j}\}$ and $\hat{h}_j^* = \inf\{\hat{h}_j | \, j \in I_{D_k^j}\}$. Then, $\mathcal{K}_k(D, b_y)$ is obtained in the form of (D.1); then compute the measure in (D.2).

Assume that the running time to compute the elementary function $\sin\alpha_i$ in (3.6) is a constant. The complexity of its inverse $\sin^{-1}\alpha_i$ is also a constant, since all elementary functions are analytic and hence invertible by means of Newton's method. The time to compute all $\sin\alpha_i$ and $\cos\alpha_i$, $i \in I_{D_k^j}$, is $O(k) + O(k)$, i.e., $O(k)$. Then, $\hat{h}_i$ and $\hat{l}_i$ can be obtained by simple multiplication and addition operations. Since there is no need to order $\{\hat{l}_i | i \in I_{\mathcal{D}_k^j}\}$ and $\{\hat{h}_j | \, j \in I_{\mathcal{D}_k^j}\}$, $\hat{l}_j^*$ and $\hat{h}_j^*$ can be obtained in linear running time $O(k)$. The complexity to generate m convex cones $cone(\hat{l}_j^*, \hat{h}_j^*) \; \forall j = 1, \ldots, m$ is $O(mk)$. The computation of the unions of finite convex cones is exactly similar to the computation of the union of finite closed intervals in $\mathbb{R}^1$. This class of problems is well known as Klee's measure problem [26]. In 1977, Klee considered the following problem: given a collection of m intervals in the real line, compute the length of their union; he then presented an algorithm [26] to solve this problem with computational complexity (or "running time") $O(m \log m)$. Fredman and Weide showed that Klee's algorithm, based on sorting the intervals, was optimal [20]. Therefore, the complexity of the second step is $O(m \log m)$, and $\Gamma_y^k = n_\delta(O(mk) + O(m \log m)) = O(n_\delta m(k + \log m))$, where $n_\delta = L/\delta b$ and m is the binomial coefficient N choose k. Finally, the time complexity of (3.10) is $4 \cdot \Gamma_y^k$ for every observation cell, and thus the required to compute (3.1) for all $n_{\underline{\kappa}}$ observation cells in $\mathcal{G}$ is $O(n_{\underline{\kappa}} n_\delta m(k + \log m))$.

Finally, $\mathcal{G}$ is searched for the optimal channel (3.3) using the A* algorithm. The time complexity of A* depends on the heuristic function and, therefore, on the cost or reward function attached to the arcs of $\mathcal{G}$. For the reward (3.1), the easiest choice of heuristic function is $h(x) = 0 \; \forall x$. Then, the A* reduces to Dijkstra's algorithm [16], for which the running time is $O(n_\mathcal{G}^2 + n_{\mathcal{G}_a})$. Therefore, the time complexity of a detection round is given by the leading term in (4.6), which depends on the characteristics of the workspace, robotic sensors, and targets. When the game round consists of deploying a sensor in pursuit mode, the optimal channel μ_p^* is computed by first determining the interception point and time, δ and $t_{c_i^j}$, and then using the A* algorithm to find the shortest path from $p_i(t_0)$ to δ in $\mathcal{G}$. δ and $t_{c_i^j}$ are computed by solving four nonlinear equations in four variables, using Newton's method [34].

Since the gradient evaluation equals 4^2 component function evaluations, the time complexity of one Newton's method iteration is $O(4^3)$. Although the number of iterations required is unknown, Newton's method is known to converge locally in linear, or even superlinear, time. Therefore, the time complexity for computing the pursuit strategy is that of the A*. Since, in this case, the cost attached to the arcs of $\mathcal{G}$ is the Euclidean distance (3.17), the heuristic function $h(x)$ can be chosen as the straight-line distance, and the complexity can be reduced to $O((n_{\mathcal{G}_a} + n_{\mathcal{G}}) \log_2 n_{\mathcal{G}})$ [44].

REFERENCES

[1] E. U. Acar, *Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods*, Internat. J. Robotic Res., 22 (2003), pp. 7–8.

[2] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking, *Randomized pursuit-evasion in graphs*, Combin. Probab. Comput., 12 (2003), pp. 225–244.

[3] A. Arsie and E. Frazzoli, *Efficient routing of multiple vehicles with no explicit communications*, Internat. J. Robust Nonlinear Control, 18 (2007), pp. 154–164.

[4] K. C. Baumgartner and S. Ferrari, *A geometric transversal approach to analyzing track coverage in sensor networks*, IEEE Trans. Comput., 57 (2008), pp. 1113–1128.

[5] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, Wiley Interscience, Hoboken, NJ, 2006.

[6] D. P. Bertsekas, *Convex Analysis and Optimization*, Athena Scientific, Belmont, MA, 2003.

[7] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 2007.

[8] C. Cai and S. Ferrari, *Information-driven sensor path planning by approximate cell decomposition*, IEEE Trans. Syst. Man Cyber. Part B, 39 (2009), pp. 1–18.

[9] H. Choset, *Coverage for robotics: A survey of recent results*, Ann. Math. Artif. Intell., 31 (2001), pp. 113–126.

[10] J. Clark and R. Fierro, *Mobile robotic sensors for perimeter detection and tracking*, ISA Trans., 46 (2007), pp. 3–13.

[11] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, *Coverage control for mobile sensing networks*, IEEE Trans. Robotics Automat., 20 (2004), pp. 243–255.

[12] I. J. Cox and M. L. Miller, *On finding ranked assignments with application to multitarget tracking and motion correspondence*, IEEE Trans. Aerospace Electronic Systems, 31 (1995), pp. 486–489.

[13] D. Cruz, J. McClintock, B. Perteet, O. Orqueda, Y. Cao, and R. Fierro, *Decentralized cooperative control: A multivehicle platform for research in networked embedded systems*, IEEE Control Syst. Mag., 27 (2007), pp. 58–78.

[14] R. V. Dam, *Soil effects on thermal signatures of buried nonmetallic landmines*, in Detection and Remediation Technologies for Mines and Minelike Targets VIII, Proc. SPIE 5089, SPIE, Bellingham, WA, 2003, pp. 1210–1218.

[15] H. F. Davis and A. D. Snider, *Vector Analysis*, Wm. C. Brown, Dubuque, IA 1987.

[16] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[17] S. Ferrari, *Track coverage in sensor networks*, in Proceedings of the IEEE American Control Conference, Minneapolis, MN, 2006, IEEE Press, Piscataway, NJ, pp. 2053–2059.

[18] S. Ferrari and A. Vaghi, *Demining sensor modeling and feature-level fusion by Bayesian networks*, IEEE Sensors, 6 (2006), pp. 471–483.

[19] R. Fierro, A. Das, J. Spletzer, J. Esposito, V. Kumar, J. P. Ostrowski, G. Pappas, C. J. Taylor, Y. Hur, R. Alur, I. Lee, G. Grudic, and J. Southall, *A framework and architecture for multi-robot coordination*, Internat. J. Robotic Res., 21 (2002), pp. 977–995.

[20] M. L. Fredman and B. Weide, *On the complexity of computing the measure of* $\cup[a_i, b_i]$, Comm. ACM, 21 (1978), pp. 540–544.

[21] A. Gad, F. Majdi, and M. Farooq, *A comparison of data association techniques for target tracking in clutter*, in Proceedings of the Fifth IEEE International Conference on Information Fusion, 2002, IEEE Press, Piscataway, NJ, vol. 2, pp. 1126–1133.

[22] A. Ganguli, J. Cortés, and F. Bullo, *Maximizing visibility in nonconvex polygons: Nonsmooth analysis and gradient algorithm design*, SIAM J. Control Optim., 45 (2006), pp. 1657–1679.

[23] J. E. Goodman, R. Pollack, and R. Wenger, *Geometric transversal theory*, in New Trends in Discrete and Computational Geometry, J. Pach, ed., Springer-Verlag, New York, Berlin, 1991, pp. 163–198.

[24] V. Isler, S. Kannan, and S. Khanna, *Randomized pursuit-evasion in a polygonal environment*, IEEE Trans. Robotics, 21 (2005), pp. 875–884.
[25] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, *Energy efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet*, in Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), 2002, ACM, New York, pp. 96–108.
[26] V. Klee, *Can the measure of* $\cup[a_i, b_i]$ *be computed in less than* $O(n \log n)$ *steps?*, Amer. Math. Monthly, 84 (1977), pp. 284–285.
[27] S. Koenig, C. Tovey, and Y. Smirnov, *Performance bounds for planning in unknown terrain*, Artificial Intelligence, 147 (2003), pp. 253–279.
[28] A. S. LaPaugh, *Recontamination does not help search a graph*, J. ACM, 40 (1993), pp. 224–245.
[29] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.
[30] J. C. Latombe, A. Lazanas, and S. Shekhar, *Robot motion planning with uncertainty in control and sensing*, Artificial Intelligence, 52 (1991), pp. 1–47.
[31] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, London, 2006.
[32] A. Lazanas and J. C. Latombe, *Motion planning with uncertainty—A landmark approach*, Artificial Intelligence, 76 (1995), pp. 287–317.
[33] J. Leonard, H. Durrant-Whyte, and I. Cox, *Dynamic map building for an autonomous mobile robot*, Internat. J. Robotic Res., 11 (1992), pp. 286–298.
[34] J. J. Moré and M. Y. Cosnard, *Numerical solution of nonlinear equations*, ACM Trans. Math. Software, 5 (1999), pp. 64–85.
[35] G. Oriolo, G. Ulivi, and M. Venditelli, *Real-time map building and navigation for autonomous robots in unknown environments*, IEEE Trans. Syst. Man Cyber., 28 (1995), pp. 316–333.
[36] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, London, 1987.
[37] J. Paik, *Image processing-based mine detection techniques using multiple sensors: A review*, Subsurface Sensing Technol. Appl., 3 (2002), pp. 203–252.
[38] T. D. Parsons, *Pursuit-evasion in a graph*, in Theory and Applications of Graphs, Y. Alavi and D. R. Lick, eds., Lecture Notes in Math. 642, Springer-Verlag, Berlin, 1978, pp. 426–441.
[39] A. B. Poore and N. Rijavec, *A numerical study of some data association problems arising in multitarget tracking*, Comput. Optim. Appl., 3 (1994), pp. 27–57.
[40] M. Qian and S. Ferrari, *Probabilistic deployment for multiple sensor systems*, in Proceedings of the 12th SPIE Symposium on Smart Structures and Materials: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems, vol. 5765, San Diego, 2005, pp. 85–96.
[41] N. Rao, *Robot navigation in unknown generalized polygonal terrains using vision sensors*, IEEE Trans. Syst. Man Cyber., 25 (1995), pp. 947–962.
[42] N. Rao, S. Hareti, W. Shi, and S. Iyengar, *Robot Navigation in Unknown Terrains: Introductory Survey of Non-heuristic Algorithms*, Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, Oak Ridge, TN, 1993.
[43] S. M. Ross, *Introduction to Stochastic Dynamic Programming*, Academic Press, Orlando, FL, 1983.
[44] S. Russell and P. Norvig, *Artificial Intelligence, A Modern Approach*, Prentice–Hall, Upper Saddle River, NJ, 2003.
[45] T. Shermer, *Recent results in art galleries*, Proc. IEEE, 80 (1992), pp. 1384–1399.
[46] S. Thurn, *Learning metric-topological maps for indoor mobile robot navigation*, Artificial Intelligence, 99 (1998), pp. 21–71.
[47] J. Urritia, *Art gallery and illumination problems*, in Handbook on Computational Geometry, J. Sack and J. Urritia, eds., Elsevier Science, New York, 1992, pp. 387–434.
[48] R. Vidal, O. Shakernia, J. Kim, D. Shim, and S. Sastry, *Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation*, IEEE Trans. Robotics Automat., 18 (2002), pp. 662–669.
[49] T. A. Wettergren, *Performance of search via track-before-detect for distributed sensor networks*, IEEE Trans. Aerospace Electronic Systems, 44 (2008), pp. 314–325.
[50] T. A. Wettergren, R. L. Streit, and J. R. Short, *Tracking with distributed sets of proximity sensors using geometric invariants*, IEEE Trans. Aerospace Electronic Systems, 40 (2004), pp. 1366–1374.
[51] P. Yang, R. Freeman, and K. Lynch, *Distributed cooperative active sensing using consensus filters*, in Proceedings of the IEEE International Conference on Robotics and Automation, Rome, 2007, IEEE Press, Piscataway, NJ, pp. 405–410.
[52] D. Zhu and J.-C. Latombe, *New heuristic algorithms for efficient hierarchical path planning*, IEEE Trans. Robotics Automat., 7 (1991), pp. 9–20.