Decentralized Tracking of Indistinguishable Targets Using Low-Resolution Sensors

Jason M. O'Kane

***Abstract*— This paper addresses the problem of using a distributed team of tracking robots to maintain proximity to a collection of unpredictably moving targets. Each tracker is equipped with short range communication hardware and a low resolution sensor that can detect the presence of targets, but not their precise locations, their identities, nor even the number of targets within the sensing range. We present methods for set-membership-based filtering that exploit specific structures in the information space for this problem. We also describe an active algorithm that allows the tracker team to follow the targets in a coordinated way, in spite of the incomplete information and intermittent communication links. A simulation of this algorithm shows that the passive filtering algorithm maintains an accurate representation of each tracker's knowledge, and that the active tracking algorithm is able to leverage this information to achieve accurate, robust target tracking.**

I. Introduction

The problem of representing, managing, and eliminating uncertainty is of central interest for autonomous robots. This problem is especially challenging when the robot's sensors provide information that is too noisy or too incomplete to form a complete picture of the relevant situation in the world. In this paper, we address this issue for the specific case in which the sensors are intended to track the unpredictable movements of a collection of target agents. Figure 1 illustrates this scenario. Specifically, we show how to do so using sensors that can detect the presence of those targets, but cannot pinpoint their exact locations nor distinguish which target is being detected. We describe a method that allows a team of trackers to maintain close proximity to the targets, using only these sensors.

The basic technical machinery we use to achieve this goal is the concept of an *information state*, which explicitly represents the knowledge available to each individual tracking robot at a particular time. The intuition behind our approach is to maintain a set of possible states that are consistent with the robot's observations. Although this set-based "nondeterministic" approach is, in general, more conservative than the corresponding forms of probabilistic reasoning, it also has advantages in reduced modeling burden and in the possibility of guarantees of success. These differences are especially salient in contexts which the targets' motions may be adversarial, rather than random.

A key idea in ensuring the filter's computational performance remains tractable is the idea of *selective overapproximation*. This idea takes two forms in the complete filter. First, we limit the complexity of each portion of the representation by constraining the geometric operations to produce regions within a restricted range space (such as, for example, the set of all planar rectangles), overapproximating the "true" result if necessary. Second, we keep the overall complexity of the representation manageable by carefully merging representation elements. Our merge operation ensures that the resulting approximation remains correct, in the sense of containing the system's true state, while keeping the amount of overapproximation small.

J. M. O'Kane is with the Department of Computer Science and Engineering, University of South Carolina, 301 Main St., Columbia, SC 29208, USA. e-mail: jokane@cse.sc.edu. This is work is partially supported by DARPA and NSF.

Fig. 1. A team of tracking robots, shown as triangles, works to follow the movements of a group of unpredictable targets, shown as circles.

Our broader interest in this work is based on a desire to understand how small teams of robots can work together in predictable ways to achieve complex goals. As such, we consider means by which robots within communication range can exchange information with one another and efficiently "join" their knowledge into a single combined representation. We believe that this approach has the potential for decentralized multi-robot coordination algorithms that do not rely on information-rich sensors, nor on persistent connectivity between the robots.

After illustrating the basic idea with an example (Section II), reviewing related work (Section III), and giving a formal problem statement (Section IV), this paper makes several new contributions:

1) We define the relevant information space and characterize the set of reachable information states (Section V).
2) We describe a passive algorithm that allows each tracker to efficiently maintain a representation of its information state (Section VI) and a distributed, active algorithm for tracking the motions of the targets (Section VII).
3) We present simulation results (Section VIII).

Discussion and concluding remarks appear in Section IX.

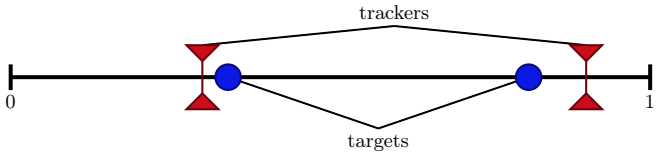

Fig. 2. Two targets moving along a unit interval, monitored by two motionless trackers.

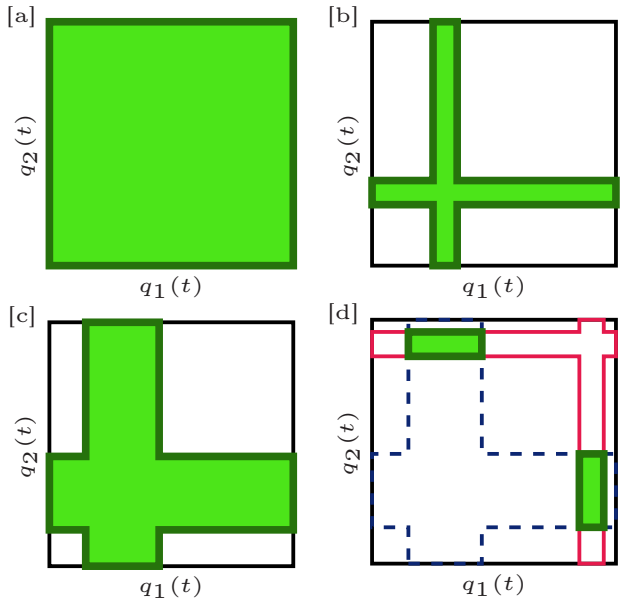

Fig. 3. Changes in the set of possible states for the scenario in Figure 2. [a] The initial condition. [b] After the first observation. [c] After the passage of a small period of time. [d] After a second observation.

II. Illustrative example

As a simple example, consider the system illustrated in Figure 2, which two point targets move, with some known maximum speed, along the unit interval $W = [0, 1]$. Denote the position of the targets at time t as $q_1(t)$ and $q_2(t)$. Two trackers monitor the movements of the targets. For simplicity, the trackers remain motionless. Also, we assume that the trackers are close enough to communicate with one another, so that observations made by one tracker are shared with the other. Each tracker has a sensor that can detect when either of the targets is within distance 0.1 of itself. The trackers' sensors may experience frequent, unpredictable false negative errors, and cannot distinguish between the two targets. Based on the observations of these trackers, what conclusions can be drawn about the positions of the targets over time?

- First, notice that, because the positions of both targets are of interest, the relevant state space is $X = W \times W$, a unit square. Before either of the trackers detects a target, no information about the targets' positions is available; any state in X could possibly be the true state. This situation is shown in Figure 3a.
- Suppose that the tracker on the left, at position 0.3, detects a target. This information allows the system to eliminate as possible states everything except those states in which $|q_1(t) - 0.3| < 0.1$ or $|q_2(t) - 0.3| < 0.1$. This "plus-shaped" region is depicted in Figure 3b. This information is shared with the rightmost tracker.
- Next, suppose some period of time elapses after this observation. The set of possible states after this time can be formed by expanding the previous result in each direction by the largest displacement that could possibly be made by the targets within the given time. See Figure 3c.
- Finally, suppose the right tracker, at position 0.9, detects a target at this time. This constrains the state to lie within the plus-shaped region shown with solid lines in Figure 3d. Combined with the knowledge retained from the previous step (dashed lines), the trackers can conclude that the target lies within the shaded region, whose boundary is a pair of rectangles.

Continuing this process of alternating observations and time intervals, we can make two comments about the sets of possible states the the system may encounter. First, the set of possible states can always be described as a finite union of rectangles. Second, the set of possible states is always symmetric about the line $q_1(t) = q_2(t)$. This is a direct consequence of the indistinguishability of the targets. In fact, these observations foreshadow the structure identified in Section V for systems with more targets, more trackers, and higher-dimensional environments.

III. Related work

Target tracking is a well-studied problem for mobile robots. A common approach is to equip the tracker with a visual sensor that provides location information whenever the target is within that sensor's field of view, with the objective of maintaining visibility of the target. Algorithms are known for planning the tracker's motions in this context using dynamic programming [8], sampling-based [11], and reactive approaches [1], [10]. Our research is also related to algorithms for pursuit-evasion, in which adversarial targets must be located within an environment [4], [5], [20]. This paper expands upon the author's prior research on target tracking [13], [14] to deal with multiple-target multiple-tracker scenarios, without the assistance of an embedded sensor network. A separate thread of research uses wireless sensor networks to monitor moving targets, typically relaying the collected data to a central sink node [16], [18], [21]. Others have expanded this work to use mobile nodes, seeking to track targets while maintaining the network connectivity [15], [22]. In many of these cases, the primary concern is to manage communication on the network. The idea of geometric reasoning for estimation by mobile sensor networks has also been explored [12].

The technical details of our work make substantial use of the notion of graph matchings, particularly for bipartite graphs. A matching on a graph $G = (V, E)$ is a set of edges $M \subseteq E$, such that no node in V is incident to more than one edge in M. In particular, to compute perfect matchings (that is, matchings in which each vertex in V is incident to *exactly*

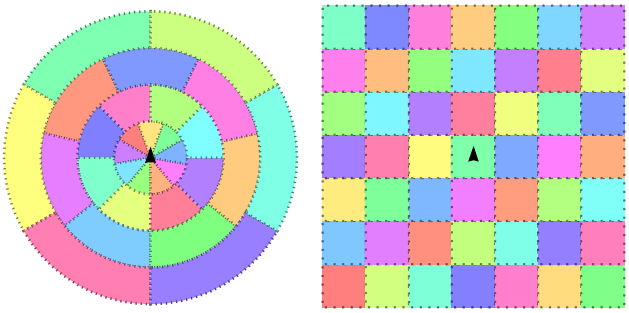

Fig. 4. Sensing preimages for trackers with [left] variable-resolution radial sensors ($r = 30$), and [right] a coarse-grained position target sensor ($r = 49$).

one edge in M), we use Edmonds's matching algorithm [2], as implemented by the Boost Graph Library [17]. To generate minimum-weight matchings on weighted bipartite graphs, we use the BlossomV algorithm [6]. Finally, one operation requires an enumeration of all perfect matchings of a bipartite graph [3], [19].

IV. Problem Statement

This section formalizes our distributed target tracking problem. The intuition is that, as the targets move through their environment, each tracking robot receives periodic observations that provide partial information about the current positions of the targets. We consider settings defined by the following elements.

- Discrete **stages** of time, indexed by consecutive integers $k = 1, 2, \ldots$.
- A planar **environment** $W \subseteq \mathbb{R}^2$.
- A collection of n identical point **targets**. We denote the position of target i at stage k as $q_k(i) \in W$. The targets move unpredictably through W, subject to a maximum movement distance s_{tgt} in each stage.
- A collection of m point **trackers**. We denote the position of tracker j at stage k as $p_k(j) \in W$. Each tracker can move distance at most s_{trk} in each stage, and knows its own position at all times. Whenever two trackers are within a known communication range, denoted r_c, they can communicate freely.
- A **target state space** $X_{tgt} = W^n$ and a **tracker state space** $X_{trk} = W^m$ in which individual elements describe the positions of all the targets or trackers respectively, and an overall **state space** $X = X_{tgt} \times X_{trk}$. The target state at stage k is denoted x_k.
- A set of r disjoint planar **sensing regions** $S_1, \ldots, S_r \subseteq \mathbb{R}^2$, representing regions relative to the position of each tracker, in which the presence of targets can be detected. See Figure 4. This includes, as a limiting case with $r = 1$, the possibility that each tracker has only a single binary sensor that only reports the presence or absence of targets.
- An **observation space** $Y = \{0, 1\}^r$. At each stage k, each tracker j receives an observation $y_k(j) \in Y$. The interpretation is that, if the l-th entry of $y_k(i)$ is 1, then at least one target is in region S_l relative to $p_j(k)$. Because of the possibility of false negative errors, if the l-th entry of $y_k(i)$ is 0, no conclusions can be drawn about the presence of targets in S_l.

Whenever there is no ambiguity, we omit the "(j)" that identifies a particular tracker. The goal is to select movements for the trackers to minimize the average distance from each target to the nearest tracker, expressed as:

$$P_k = \frac{1}{nk} \sum_{i=1}^{k} \sum_{l=1}^{n} \min_{j=1,\ldots,m} ||p_i(j) - q_i(l)||. \qquad (1)$$

We treat this objective function as our definition of the goal for a team of tracking robots.

Based on this formulation, we can make two additional definitions that will streamline the presentation.

Definition 1: The **observation preimage** $H(y, p) \subseteq X_{tgt}$ of an observation $y \in Y$ at position $p \in W$ is defined as the set of target states that could possibly generate observation y for a tracker at position p.

Definition 2: The **forward projection** $F(x) \subseteq X_{tgt}$ of a target state $x \in X_{tgt}$ is defined as the set of target states reachable from x in a single stage. The forward projection $F(S)$ of a set of target states $S \subseteq X_{tgt}$ is defined as the union of the forward projections of the target states in S. The forward projection $F(E)$ of a set of environment positions $E \subseteq W$ is the set of positions reachable within a single stage a single tracker starting in E.

V. Information States

Using its history of movements and observations, what conclusions can a tracker draw about the positions of the targets? The key idea is that each observation enables the tracker to eliminate some target states as possibilities.[1]

Definition 3: A target state $x_k \in X_{tgt}$ is **consistent** with a sequence of observations $y_1, \ldots, y_k$, and a sequence of tracker positions $p_1, \ldots, p_k$ if there exist target states $x_1, \ldots, x_{k-1} \in X_{tgt}$ such that each transition between target states is in the corresponding forward projection, $x_i \in F(x_{i-1})$, and each observation is in the corresponding observation preimage, $y_i \in H(y_i, p_i)$.

Definition 4: The **information state** (I-state) $\eta_k \subseteq X_{tgt}$ of a tracker at stage k is the set of target states consistent with the history of observations and movements of that tracker.

Informally, an I-state represents the knowledge available to each tracker at a given stage. The algorithms described in subsequent sections are concerned with efficiently maintaining such I-states, and using them to drive the trackers' decision making. However, it is helpful to a have formal characterization of the set of I-states that can actually occur in our system. This characterization can viewed as a generalization of the observations at the end of Section II. First, we establish that every reachable I-state can be expressed in a particular, well-behaved form.

[1]Details about this information state approach appear in chapters 11 and 12 of LaValle's book [7].

Definition 5: An I-state has an **ordered union-of-products** form if it can be expressed as

$$\eta_k = \bigcup_i \left(R_1^{(i)} \times \cdots \times R_n^{(i)} \right), \tag{2}$$

in which each $R_l^{(i)}$ is a subset of W. An I-state has an **unordered union-of-products** form if it can be expressed as

$$\eta_k = \bigcup_i \bigcup_{\tau \in S_n} \left(R_{\tau(1)}^{(i)} \times \cdots \times R_{\tau(n)}^{(i)} \right), \tag{3}$$

in which each $R_l^{(i)}$ is a subset of W, and S_n is the group of all permutations of n elements.

Note that the ordered union of products form still treats targets as identifiable, in spite of the intuition that the trackers have no means to distinguish one target from another. In contrast, the unordered union-of-products form does capture this notion of indistinguishability. The latter form is used by the filter described in Section VI. It remains to establish that any reachable I-state can be expressed in this form.

Lemma 1: Every reachable I-state can be expressed in ordered union-of-products form.

Proof: Use induction on the stage index k. At the initial condition, in which the tracker has no prior knowledge, we have $\eta_1 = W^n$, which is a degenerate union of products. Now suppose that the statement holds at stage k to show that it holds at stage $k+1$. Between these two stages, two changes to the I-state occur: forward projection to account for possible movements of the targets and intersection with the observation preimage $H(y_k, p_k)$ to account for the tracker's sensing. For the forward projection, we have

$$\begin{aligned} F(\eta_k) &= F\left(\bigcup_{i=1}^{l} \left[R_1^{(i)} \times \cdots \times R_n^{(i)} \right] \right) \\ &= \bigcup_{i=1}^{l} \left[F\left(R_1^{(i)}\right) \times \cdots \times F\left(R_n^{(i)}\right) \right], \end{aligned}$$

which remains a union of products. For the observation updates, consider in succession each a for which the a-th element of y_k is 1. Let S denote the result of translating the sensor region S_a into the global coordinate frame. The I-state change made by each of these partial observations is

$$\begin{aligned} \eta_k' &= F(\eta_k) \cap \left(\bigcup_{j=1}^{n} \left[W^{j-1} \times S \times W^{n-j} \right] \right) \qquad (4) \\ &= \left(\bigcup_{i=1}^{l} \left[R_1^{(i)} \times \cdots \times R_n^{(i)} \right] \right) \\ &\quad \cap \left(\bigcup_{j=1}^{n} \left[W^{j-1} \times S \times W^{n-j} \right] \right) \\ &= \bigcup_{i=1}^{l} \bigcup_{j=1}^{n} \left[R_1^{(i)} \times \cdots \times \left(R_j^{(i)} \cap S \right) \times \cdots \times R_n^{(i)} \right], \end{aligned}$$

in which the n-fold union arises from the tracker's inability to distinguish between the targets. This is an ordered union-of-products form for η_{k+1}. ■

Lemma 2: If η is a reachable I-state, $x \in \eta$ is a target state contained in that I-state, and $\tau \in S_n$ is a permutation of n items, then $\tau(x)$ is also contained in η.

Proof: Since $x \in \eta$, there exists a sequence of target states $x_1, \ldots, x_{k-1} \in X_{tgt}$ consistent with the tracker's movements and observations. Note, however, that the observations received depend only on the *number* of targets within each sensing region S_l, and not on their specific identities, so the same sequence of observations is feasible for the target state sequence $\tau(x_1), \ldots, \tau(x_{k-1})$. Note also that, for $i = 1, \ldots, k$, we have $\tau(x_i) \in F(\tau(x_{i-1}))$. Conclude from the definitions that $\tau(x) \in \eta$. ■

Lemma 3: Every reachable I-state has an unordered union of products form.

Proof: Follows directly from Lemmas 1 and 2. ■

The upshot of this reasoning is that we can represent information states, which are regions in $\mathbb{R}^{2n}$, using a only a set of 2-dimensional regions in W, without requiring any higher-dimensional geometric operations.

VI. Passive filtering

We have now laid sufficient groundwork to describe the filtering algorithm used by each tracker. The intuition is to represent the I-state's unordered union-of-products form (whose existence is guaranteed by Lemma 3) by storing the $R_j^{(i)}$ regions of Equation 3. Specifically, to store the robot's information state η_k, the robot retains a collection of **elements**. Each element is composed of an unordered collection of n **regions**, each of which is an explicit geometric description of a subset of W. An element E represents the union of all permuted Cartesian products of its associated regions. The intuition is that an element represents one "iteration" of the outer union of Equation 3, and each region within that element represents one of the $R_{\tau(j)}^{(i)}$ subsets of W in that iteration. The number of elements can vary over time, informally as a function of the degree of ambiguity in the robot's observations. The initial I-state has a single element composed of n copies of W, representing a complete lack of knowledge about the target state.

A. Single tracker updates

For a single tracker working in isolation, two kinds of operations are needed to keep such a data structure up-to-date. Algorithm 1 shows the details.

1) **Motion updates**: In each stage, the tracker updates its I-state to account for the possibility of motion by the targets, by replacing each region $R_j^{(i)}$ with its forward projection $F\left(R_j^{(i)}\right)$.
2) **Observation updates**: Upon receiving an observation y_k, the tracker must compute the intersection of its current I-state with the observation preimage $H(y_k, p_k)$. This is accomplished by iterating over the nonzero entries in y_k. In each iteration, the filter replaces each element in its I-state with n new elements, each generated by intersecting one of that element's regions with the appropriate sensing region. This process follows directly from Equation 4.

Algorithm 1 INFOSTATEUPDATE(η_k, p_k, y_k)

1: **for** each element $R_1^{(i)} \times \cdots R_n^{(i)}$ in η **do**
2: **for** each region $R_j^{(i)}$ in element $R_1^{(i)} \times \cdots R_n^{(i)}$ in η **do**
3: $R_j^{(i)} \leftarrow F(R_j^{(i)})$
4: **end for**
5: **end for**
6:
7: $\eta \leftarrow \eta_k$
8: **for** each sensing region S_l for which the l-th entry of y_k is equal to 1 **do**
9: $\eta' \leftarrow$ empty I-state, with no elements
10: **for** each element $R_1^{(i)} \times \cdots R_n^{(i)}$ in η **do**
11: **for** $j \in 1, \ldots, n$ **do**
12: insert element $R_1^{(i)} \times \cdots \times \left(R_j^{(i)} \cap S_l\right) \times \cdots \times R_n^{(i)}$ into η'
13: **end for**
14: **end for**
15: $\eta \leftarrow \eta'$
16: **end for**
17:
18: **return** η

These two update techniques are sufficient to maintain the correctness of the robot's representation. Unfortunately, the observation updates, as described, can increase the number of elements in the representation without bound. To keep the number of elements manageable, we apply a sequence of changes after each update:

- First, we **reorder regions in each element** into a canonical, lexicographic order. We know that this does not change the underlying subset of X_{tgt} represented by the data structure because of Lemma 2.
- Second, we **reorder the elements** into a canonical order. Because the union operation is commutative, again there is no change to the portion of X_{tgt} represented.
- Third, we **remove duplicate elements** (since $A \cup A = A$), and **elements representing the empty set** (since $A \cup \emptyset = A$).
- Finally, we **remove dominated elements** (since, if $B \subset A$, then $A \cup B = A$). One element E_2 dominates another element E_1 if the subset of X_{tgt} represented by E_1 is fully contained in the subset of X_{tgt} represented E_2. To test this for a pair of elements, we form a bipartite graph in which the $2n$ total nodes correspond to the regions from E_1 and E_2. Edges are defined by containment of regions from E_1 by regions from E_2. Then E_2 dominates E_1 if and only if this graph has a perfect matching.

For clarity of presentation, these operations are described as a monolithic post-processing step. However, substantial optimizations can be achieved by applying them on-the-fly. An obvious example is, in Line 12 of Algorithm 1, to immediately drop any element containing an empty region.

Even with these clean-up operations, the number of elements can still grow quickly when there is insufficient information to fully disambiguate the targets. To combat this, we cap the number of elements at a relatively small limit, denoted M. Whenever the I-state has more than M elements, we select the two most similar elements, as measured by the total pairwise overlap between regions in each element. We replace these two elements $R_1^{(a)} \times \cdots \times R_n^{(a)}$ and $R_1^{(b)} \times \cdots \times R_n^{(b)}$ with a single new element $\left(R_1^{(a)} \cup R_1^{(b)}\right) \times \cdots \times \left(R_n^{(a)} \cup R_n^{(b)}\right)$. The tracker repeats this process until at most M elements remain. Note especially that, after these reductions, the resulting representation remains a superset of the original I-state. Therefore, a tracker is able to retain the guarantee that the true tracker state is contained within its I-state. Combined with the geometric overapproximation method described below, which limits the complexity of each element, this results in an overall reduction in the complexity of the I-state.

Moreover, note that large numbers of elements are necessary only when there are many observations that cannot be definitively attributed to a single target. At the other extreme, when there is enough information to match observations to targets, the filter uses only a single element. In this sense, the filter is "ambiguity sensitive": Its run time depends on the degree of ambiguity in the information available to the tracker, spending computation time and memory only when doing so is necessary to correctly represent its uncertainty. This idea is similar in spirit to the large family of output sensitive algorithms in computational geometry, whose run time depends not only on the size of the input but also on the size of the correct output.

1) Geometric overapproximation: An important complication is that the geometric operations called for in Section VI-A, such as unions and intersections of potentially complicated regions in W, may themselves require expensive computation and storage resources. When necessary, we can simplify the computation by constraining each region to have a relatively simple geometric form. For example, depending on the shapes of the sensing regions, disks or rectangles may be reasonable choices. The geometric operations in the filter can then be modified to return the smallest such region containing the "true" exact result. Examples of these operations for regions constrained to be circular disks have previously been described by the author [13].

2) Querying the I-state: So far we have shown how to maintain a representation of the robot's I-state in unordered union of products form. One natural use for such a data structure is to answer queries of the form, "Does the I-state contain a given target state x?" This kind of query can be reduced to the question of whether any of the elements in I-state contains state x. For each element E, we again use matchings on a bipartite graph with $2n$ nodes. The two node sets are the tracker positions in x and the regions in E. Two nodes are connected by an edge if and only if the corresponding tracker position is contained in the corresponding region. A perfect matching exists on this graph

if and only if x is in the set represented by E.

B. Communication updates

The filtering technique described in Section VI-A considers only a single tracker operating on its own. However, in our distributed multiple target tracking application, we also must consider the effect of communication between the trackers on their I-states. Whenever two trackers are within communicate range, we assume that they transmit their I-states to one another. The resulting problem is: Given two up-to-date I-states $\eta^{(1)}$ and $\eta^{(2)}$, compute a single combined I-state $\eta' = \eta^{(1)} \cap \eta^{(2)}$. Representing each I-state as an unordered union-of-products, we get:

$$\begin{aligned}\eta' &= \left[\bigcup_i \bigcup_\tau \left(R^{(i)}_{\tau(1)} \times \cdots \times R^{(i)}_{\tau(n)}\right)\right] \\ &\quad \cap \left[\bigcup_j \bigcup_\rho \left(T^{(j)}_{\rho(1)} \times \cdots \times T^{(j)}_{\rho(n)}\right)\right] \\ &= \bigcup_i \bigcup_j \bigcup_\tau \left[\left(R^{(i)}_{\tau(1)} \cap T^{(j)}_1\right) \times \cdots \times \left(R^{(i)}_{\tau(n)} \cap T^{(j)}_n\right)\right]\end{aligned}$$

Note, however, that a direct application of this equation to compute η' would take $O(M^2 n!)$ time. Instead, to exploit the fact that many of the intersections will result in empty elements, we form a bipartite graph, in which each node represents a region from one of the two elements being merged, and edges connect R- and T-nodes whose intersection is non-empty. On this graph, we enumerate all the perfect matchings [19] and generate a new element by intersecting the matched regions in each such perfect matching. This computation can either be performed on both of the communicating trackers in parallel (duplicating computation, but avoiding additional communication), or a single tracker can perform the computation and transmit the result.

The result is the intersection of $\eta^{(1)}$ and $\eta^{(2)}$ and as such is a subset of both of these original I-states, but its representation may require more elements. After computing the new element set for η', we apply the same clean-up operations described above to control the size of the representation.

VII. Active tracking

The final portion of our approach is to show how I-states maintained as described in Section VI can be used for distributed target tracking. The goal here is to illustrate one possible application for I-states maintained as described in Section VI; many other choices are possible. The basic idea of the approach is that, as time passes, each tracker maintains an I-state describing its knowledge about the target state, along with $m-1$ separate subsets of W describing its knowledge about the other trackers' positions. Whenever two trackers are within communication distance of one another, they share this knowledge with one another.

The tracking algorithm, which is executed on each tracker based on this information, proceeds in three steps. First, the tracker selects a collection of m destinations for all the of trackers, including itself. Second, it finds a matching of the trackers to those destinations that minimizes an estimate of the total amount of travel needed to reach the destinations. Third, the tracker moves along the shortest path toward the destination assigned to itself in this matching. Section VII-A and Section VII-B describe the destination selection and tracker-to-destination matching processes, respectively. This three-stage process is repeated whenever a tracker updates its I-state. Although the approach relies on each tracker reasoning about the decisions of each other tracker, the method still scales reasonably well in the number of trackers, because the clustering and graph matching algorithms can be executed in time that is negligible in comparison to the time required to execute the I-state filter.

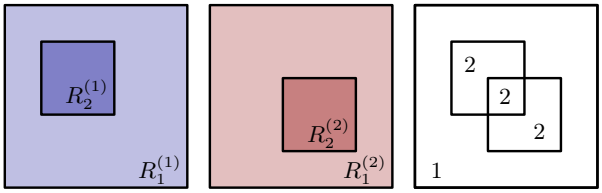

Fig. 5. Computing upper bounds on the number of targets in each partition cell, for an example I-state with 2 elements. [left] Element E_1, with two overlapping regions. [center] Element E_2, with two overlapping regions. [right] The partition of the square environment into 4 cells induced by this I-state. Each cell is labeled with the upper bound $t(c)$ on the number of targets in that cell.

The crucial features of this overall approach are that the trackers' movements are driven directly by their information states, and that if two trackers have the same information, then each will choose a different target. In this way, we can ensure that multiple trackers do not "chase" the same target for very long. If two trackers do select the same target, they will eventually come within communication distance. When this occurs, the trackers will share their knowledge with each other, and, now having the same knowledge, one the trackers will assign itself to a different destination. This works without relying on a central decision maker and in spite of a sparse communication graph that will not, in general, be a connected graph.

A. Destination selection

The intuition of the destination selection portion of the algorithm is to spread the trackers out across the environment in places that could possibly have large numbers of targets. To achieve this, the tracker computes a partition of the environment into cells, such that each cell is a maximal subset of W that does not cross any of the region boundaries in the tracker's I-state. We can derive an upper bound on the number of targets in each cell c of this partition as follows. First, for each element, we count the number of regions that cover c. Then, we select the largest such value over all elements in the I-state, as Figure 5 illustrates. We denote this upper bound as $t(c)$, and the planar area of the cell as $a(c)$.

Based on these upper bounds, we use a modified version

of the k-means clustering algorithm [9], with m cluster centers. The centroids of the partition cells are used as the observations in the input to clustering algorithm. There are two changes from the standard algorithm:

1) **Greedy initialization**: The initial cluster centers are assigned in a greedy manner. The tracker computes largest $t_{max} = \max[t(c)]$ across all partition cells c, and places a cluster center at the centroid of the largest contiguous collection of cells for which $t(c) = t_{max}$. The algorithm then decrements the $t(c)$ values for the cells in that contiguous collection, and repeats the process m times, modifying the working $t(c)$ values each time. The effect is that the cluster centers are initially spread across the environment in places with large values for $t(c)$.
2) **Weighted distance function**: The main body of the k-means algorithm works by iteratively moving the cluster center positions to the mean of the observations nearest to that center. To place greater emphasis on partition cells more likely to contain one or more targets, we assign each observation a multiplicative weight $w(c) = t(c)/a(c)^2$. These weights are included in the distance computation for moving the cluster centers.

The converged cluster centers, a set of m points in W, are used as the tracker destinations.

B. Matching destinations to trackers

Given this collection of destinations, it remains for the tracker to decide which of these it should select for itself. To do this, we use yet another form of bipartite graph matching. Specifically, the tracker builds a complete bipartite graph. The trackers themselves form one group of nodes; the destinations selected as in Section VII-A form the other group of nodes. Edges connect each node in the former set to each node in the latter set with weights equal to the largest possible distance (recall that the other tracker's positions are not fully known) from the given tracker to the given destination. Using the BlossomV algorithm [6], we find the minimum weight perfect matching on this graph. Notice that, because the input is a complete bipartite graph, a perfect matching will always exist. The intuition is to find a global assignment of trackers to potential target locations that minimizes the total distance to be traveled.

Finally, the tracker moves toward whichever destination is matched to itself in this minimum-weight matching. This destination is the one that, based on the information currently available to this tracker, appears to be the best choice for optimizing the global performance of the team.

VIII. Simulation Results

To study its performance and efficiency, we implemented this algorithm in simulation. Figure 6 shows a simple example in which two trackers monitor three targets. Its I-state regions are constrained to remain rectangles, as described in Section VI-A.1. In this example, the largest number of elements was 12, so the value of M did not play a role. The

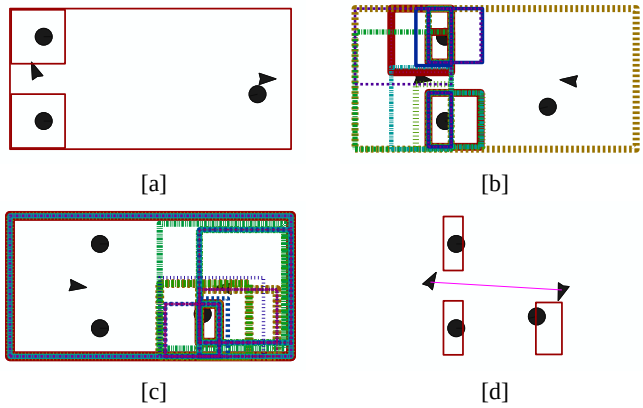

Fig. 6. Simulation snapshots with two trackers and three targets. [a] The I-state of the leftmost tracker early in the simulation. [b] Immediately before coming within communication range of the right tracker, the left tracker's I-state has 8 elements. [c] At the same stage, the right tracker's I-state has 10 elements. [d] When communication is finally possible, the trackers exchange information, eliminating most of the uncertainty and resulting in identical, single element I-states.

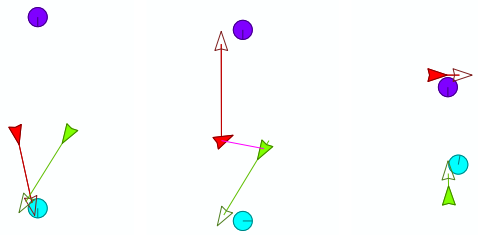

Fig. 7. Destination selection in active tracking. In this example, the sensing range is larger than the communication range. The destination toward which each tracker is moving is shown as an unshaded triangle. [left] Two trackers, each unaware of the presence of the other, chase the same target. [center] When the trackers are close enough to communicate, they merge their I-states, and one of the trackers selects a different destination. [right] A few stages later, each tracker is following a unique target.

trackers have 9 square sensing regions in a grid pattern. For the active tracking approach, Figures 7 and 8 illustrate the implicit coordination between the trackers.

For a quantitative evaluation of the success of the active tracking algorithm, we performed an experiment in the environment depicted in Figure 9. We simulated varying numbers of trackers and targets in this environment and measured the tracker team's performance P (as defined in Equation 1) across 1000 stages with $M = 10$. To eliminate any bias from a "startup phase" in which the trackers have not yet found the targets, we assigned the initial positions of the first m targets to be near the initial position of a tracker. Each target moved along shortest paths to a sequence of randomly-selected destinations. The results shown in Figure 9 are averaged over 10 trials with these parameters. They confirm the expectation that the trackers' performance generally improves with fewer targets and more trackers, and that performance degrades

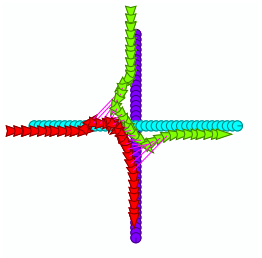

Fig. 8. When two targets cross, the trackers near them adjust their movements to ensure that both targets are tracked. Line segments connect the trackers when they are close enough to communicate. In this example, the crossing causes the trackers to seamlessly "swap" targets.

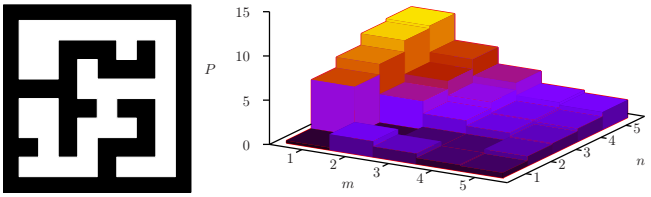

Fig. 9. [left] The environment used for our quantitative tests. [right] Results from these tests, showing tracking performance as a function of the number of trackers and the number of targets.

rapidly when the trackers outnumber the targets.

IX. Conclusion

This paper described a technique for tracking multiple mobile agents using a team of trackers, without relying on precise sensors or central decision making. More broadly, the techniques in this work may have applications in other scenarios in which sensors cannot distinguish between any of several distinct phenomena. However, several aspects of the problem are still not fully understood.

First, although the selective overapproximation scheme used to simplify the representation of the I-state does provide a guarantee of correctness, in the sense that the true state is contained in the I-state, we currently cannot characterize the amount of overapproximation required to maintain this guarantee. Strong bounds on this error would have obvious value for this problem. We are currently investigating techniques that maintain, as part of the representation, an estimate of the amount of difference from the true I-state. Moreover, it may be profitable to maintain both inner and outer limits for the I-state boundary, in a manner similar to the use of interval arithmetic in numerical analysis.

Another limitation is in the communication model, in which we have assumed that the trackers communicate freely whenever they are within range of one another. Although this assumption may be reasonable in some instances, especially because each I-state can be expressed with only a few hundred bytes, it remains an open problem use these I-state to efficiently reason about the value of communication, in order to reduce the overall bandwidth requirements.

There are also interesting extensions to allow the trackers to track unknown numbers of targets. The idea is to assume at first that only one target exists, and to extend the I-state to accommodate additional targets whenever the observations can no longer be explained by the tracker's current estimate of the number of targets.

Acknowledgements

We gratefully acknowledge support for this work from the U. S. National Science Foundation (IIS-0953503) and the Defense Advanced Research Projects Agency (N10AP20015).

References

[1] T. Bandyopadhyay, Y. P. Li, M. H. Ang Jr., and D. Hsu. Stealth tracking of an unpredictable target among obstacles. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, pages 43–58, 2004.

[2] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[3] K. Fukuda and T. Matsui. Finding all the perfect matchings in bipartite graphs. *Applied Mathematics Letters*, 7(1):15–18, 1994.

[4] L. Guilamo, B. Tovar, and S. M. LaValle. Pursuit-evasion in an unkown environment using gap navigation trees. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[5] V. Isler, D. Sun, and S. Sastry. Roadmap based pursuit-evasion and collision avoidance. In *Proc. Robotics: Science and Systems*, 2005.

[6] V. Kolmogorov. Blossom V: A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, 2009.

[7] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[8] S. M. LaValle, H. H. González-Baños, C. Becker, and J.-C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proc. IEEE International Conference on Robotics and Automation*, pages 731–736, 1997.

[9] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[10] R. Murrieta, A. Sarmiento, S. Bhattacharya, and S. A. Hutchinson. Maintaining visibility of a moving target at a fixed distance: The case of observer bounded speed. In *Proc. IEEE International Conference on Robotics and Automation*, 2004.

[11] R. Murrieta-Cid, B. Tovar, and S. Hutchinson. A sampling-based motion planning approach to maintain visibility of unpredictable targets. *Autonomous Robots*, 19(3):285–300, 2005.

[12] T. Nelson and R. Freeman. Set valued estimation in mobile sensor networks. In *Proc. IEEE Conference on Decision and Control*, 2009.

[13] J. M. O'Kane. On the value of ignorance: Balancing tracking and privacy using a two-bit sensor. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2008.

[14] J. M. O'Kane and W. Xu. Network-assisted target tracking via smart local routing. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010. To appear.

[15] R. Olfati-Saber. Distributed tracking for mobile sensor networks with information-driven mobility. In *Proc. American Control Conference*, pages 4606–4612, 2007.

[16] N. Shrivastava, R. Mudumbai U. Madhow, and S. Suri. Target tracking with binary proximity sensors: fundamental limits, minimal descriptions, and algorithms. In *Proc. International Conference on Embedded Networked Sensor Systems*, pages 251–264, 2006.

[17] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, Indianapolis, Indiana, 2001.

[18] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proc. International Conference on Embedded Networked Sensor Systems*, pages 1–12, 2004.

[19] T. Uno. A fast algorithm for enumerating bipartite perfect matchings. In *International Symposium on Algorithms and Computation*, 2001.

[20] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry. Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 2002.

[21] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, 19(2):61–72, 2002.

[22] Y. Zou and K. Chakrabarty. Distributed mobility management for target tracking in mobile sensor networks. *IEEE Transactions on Mobile Computing*, 6(8):872–887, 2007.