Network-assisted target tracking via smart local routing

Jason M. O'Kane and Wenyuan Xu

***Abstract*—Target tracking problems have been extensively studied for both robots and sensor networks. In this paper, we consider a target tracking problem in which a sensorless tracking robot must maintain close proximity to an unpredictably moving target. To assist the robot, a network of sensor nodes, each equipped with a binary proximity sensor, is spread through the environment. This architecture has the benefit of eliminating the need for information-rich sensors on the tracker, while supplying it with nonlocal observations of the target. However, it also introduces new complications due to the mobility of the tracker and the energy limitations of the sensor nodes. To address these issues, we present algorithms that allow both the tracker and the sensor nodes to maintain partial information about the target's location. The contribution of this work is an algorithm that manages the propagation of information across this network, making message delivery decisions on-the-fly, based on each message's informative value for the tracker. We present an implementation along with simulation results. The results show that our system achieves both good tracking precision and low energy consumption in both start-up and steady state phases of the problem, and that its performance is superior to that of earlier methods for this problem.**

I. Introduction

Tracking problems for mobile robots have received substantial attention in recent years. In these problems, a robot *tracker* seeks to maintain close proximity to an unpredictable *target*. Effective target tracking algorithms have many important applications, including monitoring and security. Algorithms have been proposed to solve this problem with mobile robots under various constraints and sensor models [9], [11], [12]. However, these existing methods for robotic tracking are hampered by two primary limitations.

First, existing tracking methods generally rely on sensors onboard the robot, which by nature only provide information about the target's location when the target is nearby. This limitation is particularly problematic in cases where (a) the tracker starts with little or no knowledge of the target's location or (b) the tracker loses contact with the target during its execution. To recover from these situations using only local information is a challenging problem, requiring extensive search in the worst case [4], [6].

Second, prior work assumes that the robot has access to sensors such as visual or range sensors that are, in spite of their local nature, relatively powerful and information-rich. Such sensing capabilities add additional cost and complexity to the robot. It is desirable to design and deploy simpler robots with less sophisticated sensing hardware. Moreover, tracking with limited sensing is of independent interest for cultivating understanding of the information requirements of the target tracking task.

J. M. O'Kane and W. Xu are with the Department of Computer Science and Engineering, University of South Carolina, 301 Main St., Columbia, SC 29208, USA. e-mail: {jokane, wyxu}@cse.sc.edu.

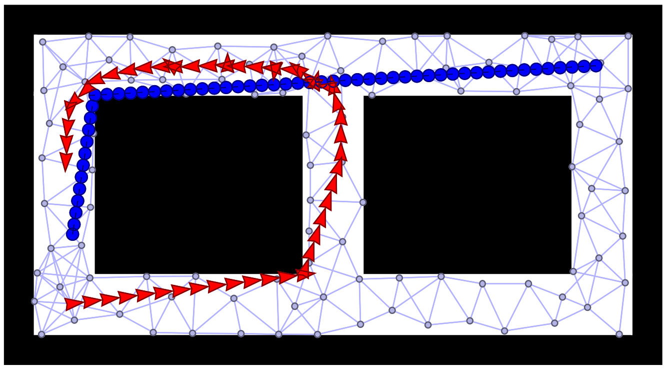

Fig. 1. A time-lapse view of an example tracking problem, in which a tracker (triangle shape) to find and maintain close proximity to a target (circle shape). A wireless sensor network deployed in the environment provides the tracker with information about the target's whereabouts. Edges connect nodes within communication range of one another.

In this paper, we propose a tracking technique that resolves these limitations by allowing the robot to utilize a wireless sensor network to assist in the tracking task. The tracking task can be divided into two parts: sensing the target and following its movements. As such, we decouple these parts and delegate the sensing task to a stationary sensor network. The mobile tracker then follows the target using only the observations made by these sensor nodes. This arrangement eliminates the need for complex sensors on the tracker, and also provides an efficient means for delivering nonlocal information to the tracker.

To emphasize the simple nature of the system, we assume the sensor nodes are equipped with binary proximity sensors that cannot sense the accurate location of the target. Instead, the sensors report only whether the target is within a given sensing range. Furthermore, we assume that these sensors experience frequent false negative errors. To utilize this coarse location information, our approach makes extensive use of the concept of *information states* [8], which explicitly encode the robot's uncertainty about the target position. Specifically, the tracking robot uses information collected from the network to synthesize a set of *possible states*, then makes greedy motions intended to reduce the size of this set, and to minimize the tracker's distance from the set of possible states.

Note, however, that wireless sensor networks are characterized by limited energy and communication resources. Therefore only lightweight, energy-efficient protocols are feasible in sensor networks. Generally, among the three main components of a sensor node (e.g., sensor(s), processor, and radio), the radio dominates the energy consumption [19]. Thus, to extend network lifetime, the network protocols should limit the total number of messages required.

The contribution of this paper is to describe and evaluate a tracking technique for a robot cooperating with a sensor network in which:

1) Nothing is known about the target's motion other than its maximum speed.
2) The tracking robot has no sensors that directly provide information about the target.
3) The sensor nodes detect only when the target is nearby, but do not provide any precise location information, and are subject to frequent, unpredictable failures.
4) Each sensor node has a limited energy budget for making transmissions.

Specifically, we demonstrate that for such systems, both tracking performance and energy-efficient network operation can be achieved simultaneously.

The remainder of this paper has the following structure. We begin by reviewing related work in Section II. Next, we formalize the tracking problem in Section III. Sections IV and V describe the strategy for controlling the tracker and the protocol to deliver sensing data to the tracker, respectively. In Section VI, we present and discuss simulation results. Concluding remarks appear in Section VII.

II. Related work

A. Robotic target tracking

Target tracking problems for mobile robots have been studied for some time. The objective for these problems is generally to maintain visibility between the target and the tracker. Algorithms are known for planning the tracker's motions using dynamic programming [9], sampling-based [12], and reactive approaches [11].

B. Sensor network target tracking

Wireless sensor networks (WSNs) have been deployed to track the positions of humans [17], moving vehicles [20], and other moving targets [16]. Sensor nodes passively collect measurements and rely on multi-hop communication to deliver data to a central data collection unit. As a result, the communication can become expensive when the network size is large. The tracking architecture proposed in this work reduces this communication cost by having a mobile tracker follow a target and collect the information from the target in its vicinity.

The use of mobile sensor networks, in which individual nodes have both sensing and motion capability, has also been proposed as a means to track moving targets [14], [21]. The primary concern in this area is to track the targets while maintaining the network connectivity. We propose a different architecture where the connectivity problem and mobility management issues are decoupled.

C. Routing in sensor networks and MANET

Other research has considered similar problems of message delivery in sensor networks [5], [18]. Such protocols work well with stationary sinks, but are unsuitable for a mobile receiver.

For mobile ad hoc networks (MANET), Ad hoc On-demand Distance Vector (AODV) [15] and Optimized Link State Routing (OLSR) [2] are two well-known routing protocols. Those protocols either impose high latency for the initial path setup or lead to a wasteful overhead of routing traffic, and therefore are not suitable for a network with a highly mobile receiver and battery-operated sensors.

D. Data aggregation in sensor networks

The efficiency of our approach stems in part from the idea of *data aggregation*, in which the sensor nodes combine and synthesize the information they receive about the target's position, rather than simply forwarding messages unchanged. This concept of in-network data aggregation has also been studied in the context of WSNs. Typically, the data aggregation algorithm, such as TAG [10], routes the aggregated values up towards the root of a pre-established routing tree with partial data aggregated at internal tree nodes. The unique idea presented in this paper is that the in-network state computation is used to assist route selection, so that messages are delivered efficiently between the sensors near the target and the ones near the tracker.

E. Cooperative robot-network systems

The idea of combining WSNs with mobile robots has been investigated as well. In particular, mobile robots are used for sensor network deployment with the goal of achieving good sensor coverage [1]. Our work complements theirs in the sense that we focus on the tracking application after the deployment is done. WSNs are also proposed to assist mobile robots to track targets [7], using the sensors that can supply precise location information to the robot. We take a different viewpoint: we design the tracking algorithms by considering issues associated with both sensor networks and mobile robots; and thus, achieve good tracking performance at reasonable operational cost for the network while using simple sensor devices and robots.

F. Authors' prior work

The authors' prior work [13] addresses a similar problem using a time-to-live (TTL) scheme to dynamically control how far information travels across the network. The algorithm works reasonably well in "steady state" scenarios, when the target and tracker are near one another, but performs poorly in the "startup phase" in which the tracker and target are separated by many hops in the network. The current paper proposes an entirely new method for the network that achieves good performance in both phases.

III. Problem statement

This section formalizes our target tracking problem and the performance criteria we use to measure its success.

A. System model

A point target moves unpredictably, but with maximum speed s_{tgt}, in a closed, bounded, polygonal, planar environment $E \subset \mathbb{R}^2$. Time is modeled as a continuous progression along the interval $[0, T]$. Let $q(t) \in E$ denote the position of the target at time $t \in [0, T]$.

A point robot called the tracker also moves in E. At time t, the position of the tracker is denoted $p(t)$. The tracker can choose its velocity vector $u(t)$. The velocity is constrained by a maximum speed s_{trk}. The tracker knows its position $p(t)$ within E. The tracker has no sensors that directly report on the position of the target; it instead must rely solely on the communications from the network, as described below. The *state* $x(t) = (p(t), q(t))$ comprises the target and tracker positions.

To assist the tracker, a network of k stationary wireless sensor nodes is distributed through E at positions $n_1, \ldots, n_k$. Each node knows its own position. To simplify the notation, we assume that the nodes are identical, with a fixed sensing range r_s and a fixed communication range r_c. Each node n_i can:

1) Possibly **detect the target** whenever $||q(t) - n_i|| \leq r_s$. This sensing is boolean: The node knows only whether or not the target has been detected, but no other information. This detection is also unreliable, in the sense that failing to detect the target does not imply that $||q(t)-n_i|| > r_s$. Such false negatives, which can occur as a result of unmodeled occlusions in the environment, noise, or other factors, are assumed to be extremely common. (We briefly discuss false positive errors in Section VII.)
2) **Broadcast a message** to all nodes n_j for which $||n_i - n_j|| \leq r_c$. We assume that the time required for each transmission is negligible compared to the physical speeds of the robots. In our algorithm, the content of these messages is a description of the information the node has accumulated. Specifically, the content of each message is a description of a *disk-based information state* denoted η. Section V-A describes disk-based information states in detail. Informally, η is a compact description of a set of possible states. Because of the broadcast nature of the wireless network, a single transmission is sufficient to send the message to all of a node's neighbors.

In addition, the tracker is equipped with network communication hardware, so that it can receive messages that are broadcast by nodes within r_c of $p(t)$. The tracker also uses this hardware to transmit a beacon that informs the wireless sensor nodes of its presence. This beacon is detected by the node at n_i whenever $||p(t) - n_i|| \leq r_c$. As with the target detection sensors, receipt of this beacon signal is subject to frequent false negative errors.

B. Evaluation criteria

To evaluate our system's success, we will use three criteria. The tracker's primary objective is to minimize the average distance between $p(t)$ and $q(t)$ throughout the system's execution:

$$P = \frac{1}{T}\int_0^T ||p(t) - q(t)|| \mathrm{d}t. \quad (1)$$

However, because the energy available to each wireless sensor node is limited by battery capacity, a secondary objective is to minimize the average number of message broadcasts made in the network per unit time. Let $C(i)$ denote the number of broadcasts made by the node at n_i between $t = 0$ and $t = T$. The system seeks to keep

$$C = \frac{1}{T}\sum_{i=1}^{k} C(i) \quad (2)$$

as small as possible. Finally, note that the system's execution can informally divided into a *startup* phase, during which the tracker works to reduce a relatively large distance between itself and the target, and a *steady state* phase, during which the tracker has moved near the target and works to maintain this proximity. As T increases, the relative emphasis placed on the startup phase by P decreases. Therefore, we also consider the *capture time* S, which measures the length of the startup phase, and is defined by

$$S = \min\{t \in [0, T] \ \mid \ ||p(t) - q(t)|| \leq \epsilon\}, \quad (3)$$

in which ϵ is a small positive constant.

These three performance criteria, P, S, and C, are at least partially in conflict with one another: Intuition suggests—and our experiments confirm—that decreases in P or S generally require corresponding increases in C. This tradeoff motivates our use of Pareto optimality concepts, treating the problem as a multi-objective optimization problem. Section VI presents and discusses these results.

IV. Controlling the tracker

This section describes the algorithms used by the tracker to respond to the data it receives from the network. Since the current state $x(t)$ is not necessarily known, the tracker faces two challenges: First, it must efficiently represent the limited knowledge it has about the target's position. Second, it must use this representation to plan its motion toward the target. Our approach overcomes these challenges using the the tracker's *information states* (I-states). In this context, the I-state is the set of possible states that are consistent with the information the tracker has received. The tracker computes its I-state, then chooses its motions as a function of this current I-state.

We first describe how to maintain the I-state (Section IV-A), then we present strategies used by the tracker to utilize this information (Section IV-B). Note that we are concerned here only with the control of the tracker; we defer to Section V our discussion of approaches to efficiently deliver target location information to that tracker.

A. Computing tracker information states

The tracker does not know the target's precise location, and instead must rely on the history of messages it has received to direct its motions. Suppose that tracker, as of time t, has received m messages, each describing an observation of the target:

$$\{(c_1, t_1), \ldots, (c_m, t_m)\},$$

with each c_i describing a circle known to contain the target at time $t_i < t$. Let $Q(t) \subseteq E$ denote the set of target positions

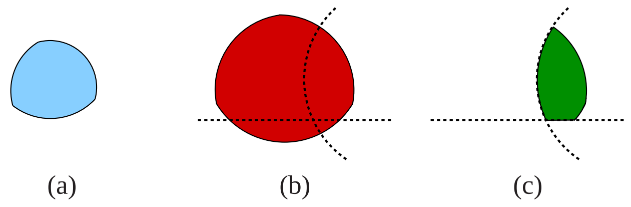

Fig. 2. Computing the tracker's information state. (a) An initial information state. (b) Expansion to account for the passage of time, and intersection with received message circles and the environment. (c) The resulting updated information state.

for which there exists a valid target trajectory consistent with that information. The tracker always knows its own position, so the tracker's I-state (that is, the set possible states) at time t is $\eta(t) = \{p(t)\} \times Q(t)$.

To compute such I-states, we perform iterative updates, maintaining the current $\eta(t)$ and updating it when time passes and when new messages are received. We start with the initial I-state $\eta(0) = \{p(0)\} \times E$. Then two kinds of updates are performed throughout the execution:

1) When time from t_1 to t_2 passes without any messages being received, we compute $\eta(t_2)$ from $\eta(t_1)$. To accomplish this we replace $p(t_1)$ with $p(t_2)$, perform a Minkowski sum of $Q(t_1)$ with a disc of radius $(t_2 - t_1)s_{tgt}$, and intersect the resulting region with E. The resulting region is retained as $Q(t_2)$.[1]
2) When a message describing a circle know to contain the target is received, the existing I-state is updated to the correct $\eta(t)$ by performing an intersection with a disk with the given center and radius r_s.

Figure 2 illustrates each of these updates.

B. Tracker strategy

We now describe how the tracker moves. Notice that, aside from knowing its own position $p(t)$ and a set of possibilities $Q(t)$ for the target's position, the tracker cannot draw any additional conclusions about the state. Given this uncertainty, the ideal position for that tracker, that minimizes average the distance to the target across all its possible positions is, by definition, the centroid of $Q(t)$. Note, however, that the centroid of $Q(t)$ may not be inside E. Based on these observations, we use the following strategy for the tracker:

Move with speed s_{trk} along the shortest path in E from $p(t)$ to the closest point in E to the centroid of $Q(t)$.

Computing this motion takes time linear in the complexity of $Q(t)$, for both the centroid and shortest path elements [3]. Each time the I-state is updated, the tracker's motion plan is also recomputed accordingly.

[1]Note that this approach may slightly overestimate the I-state when $t_2 - t_1$ is large and the boundary of E has sharp non-convex corners. This effect, which is similar to the sampling issues that arise in collision detection for path segments, can be reduced or eliminated by partitioning the time period from t_1 to t_2 into smaller segments.

V. Sensing and Data Delivery

As discussed in Section III, we consider a network of k nodes spread throughout E. Whenever a node detects the target or the tracker, it may choose to transmit a message to disseminate this information. We have already briefly argued in Section II that existing routing methods for stationary or mobile ad hoc networks are not suitable in our scenario. As a result, instead of establishing routes ahead of time, in our method the nodes send broadcast messages, but do so in a strictly controlled way to minimize energy costs.

The key insight behind our approach is the notion of *smart local routing*. We store partial information about the tracker and target positions at each sensor node. Each node uses this information to transmit messages only if they provide useful information to facilitate the target tracking. We present the details of this approach in three parts: how to compute the information state at each sensor (Section V-A); when a sensor will initiate a message (Section V-B); and what information shall be sent (Section V-C).

A. Computing node information states.

Each node maintains an I-state similar to that used by the tracker. However, the computation power and memory available to small scale sensors will, in general, be much smaller than that available to a full-fledged mobile robot. Therefore, we reduce the computational load by using a provable overestimate of the precise I-state called the *disk-based I-state*. This disk-based I-state can be stored with a small constant amount of memory and updated in constant time. Because the sensor nodes do not know the tracker's position, but could benefit from this information in assessing the relative of sending a message, the disk-based I-state contains information about both the tracker and the target locations.

Specifically, each sensor node n_i maintains a pair of disks $P_i(t)$ and $Q_i(t)$, with the invariant that

$$p(t) \in P_i(t) \text{ and } q(t) \in Q_i(t). \quad (4)$$

These two disks constitute the node's disk-based I-state. As initial values, each node assigns arbitrary centers and infinite radii to both disks. From this initial I-state, four types of updates are required in order to maintain the invariant.

1) *Target detection*. When the node detects the target, it knows that the target is within distance r_s of itself. In this case, therefore, $Q_i(t)$ is replaced the smallest disk containing $Q_i(t) \cap B(n_i, r_s)$.
2) *Tracker detection*. When the node receives the tracker's beacon signal, it knows that the tracker is within distance r_c of itself. Similar to the previous case, $P_i(t)$ is replaced the smallest disk containing $P_i(t) \cap B(n_i, r_c)$.
3) *Message receipt*. When the node receives a message from another node n_j, that message will describe the disk-based I-state of n_j. Specifically, this message contains the center coordinates and radius of both $Q_j(t)$ and $P_j(t)$. The node at n_i uses this information to refine its

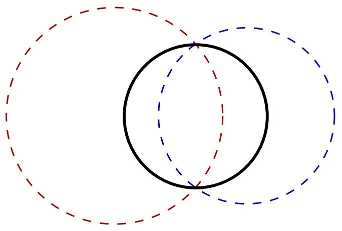

Fig. 3. Computing the smallest disk enclosing the intersection of two disks. The general case, in which the intersection in nonempty, but neither disk fully contains the other.

own knowledge, replacing $Q_i(t)$ with the smallest disk enclosing $Q_i(t) \cap Q_j(t)$ and replacing $P_i(t)$ with the smallest disk enclosing $P_i(t) \cap P_j(t)$. In this way, each node can refine its own knowledge based on messages it receives from neighboring nodes. This information sharing is the means by which knowledge is propagated across the network.

4) *Passage of time*. When time Δt passes without any of the previous three events occurring, then new disks are computed with the same centers, and having $\text{radius}(P(t+\Delta t)) = \text{radius}(P(t)+\Delta t s_{trk})$ and $\text{radius}(Q(t+\Delta t)) = \text{radius}(Q(t) + \Delta t s_{tgt})$. As with the tracker's I-state, this expansion of the disks corresponds to the unknown motions that tracker and target might have made during this time.

These update operations depend on the ability to compute the smallest disk enclosing the intersection of two other disks. In general, given two disks D_1 and D_2, assume without loss of generality that D_1 is centered at the origin, that D_2 is centered at $(b, 0)$, and that the radii of the disks are r_1 and r_2, respectively. We must consider four cases:

- If $r_2 > r_1 + b$, then D_2 contains D_1. The intersection itself is D_1 itself: $D_1 \cap D_2 = D_1$.
- If $r_1 > r_2 + b$, then D_1 contains D_2, and the intersection itself is a D_2 itself: $D_1 \cap D_2 = D_2$.
- If $r_1 + r_2 < b$, then D_1 and D_2 are disjoint, and there is no solution. This case does not occur in our setting, since $D_1 \cap D_2$ must contain either $p(t)$ or $q(t)$.
- Otherwise, D_1 and D_2 intersect in a "lens-shaped" region, as shown in Figure 3. Let $c = (r_1^2 - r_2^2 + b^2)/(2b)$. This region has vertical extrema at $(c, +\sqrt{r_1^2 - c^2})$ and $(c, -\sqrt{r_1^2 - c^2})$ and horizontal extrema at $(b - r_2, 0)$ and $(r_1, 0)$. Any disk enclosing D_1 and D_2 must contain all four of these points. This occurs with minimal radius when, choosing $a = \max(0, \min(b, c))$, the disk is centered at $(a, 0)$ and has radius $\sqrt{(a-c)^2 + r_1^2 - c^2}$.

Note that this representation consumes $O(1)$ space, and that each of its updates can be performed in $O(1)$ time. The approach is, therefore, well-suited to simple sensor nodes with limited computation power.

B. When to send a message?

The nodes' message-sending strategy is based on the disk-based I-state that each node maintains. Whenever a node receives new information, either by detecting the tracker, detecting the target, or receiving a message from another node, it must decide whether to broadcast a message of its own to propagate this information across the network. The node's I-state allows it to identify a number of situations in which it should *not* broadcast such a message to its neighbors:

1) If the new information did not result in any change the node's disk-based I-state, then the node should not broadcast its knowledge. This improves efficiency by filtering redundant messages.
2) If the node received new information about the target's position, but has already broadcast a message triggered by target information in the recent past, it should not broadcast its knowledge. This rule facilitates data aggregation, preventing the node from generating frequent messages, of which each has only limited informative value. The definition of "recent past" is governed by a parameter τ, such that no node will generate two broadcasts triggered by tracker knowledge within time τ of each other.
3) Similarly, if the node received new information about the tracker's position, but has already broadcast a message triggered by tracker information in the last τ units of time, it should not broadcast its knowledge. This rate limiting is governed by the same parameter τ, but is controlled by separate timeout counter. These "dual timeouts" are crucial to facilitate free flow of information in both directions—from the tracker toward the target, and from the target toward the tracker.
4) Finally, if the node can conclude, based on its disk-based I-state, that it is not near the shortest path between $p(t)$ and $q(t)$, then it should not broadcast its knowledge. This condition prevents the wasting of energy by transmitting data to remote portions of the environment that are not active parts of the tracking problem. Details about this condition appear below.

To implement the above constraint (4) in a precise, formal way, the node must compute the *geodesic hull* of its disk-based I-state. We define the geodesic hull of a pair of point sets A and B as the union of all shortest paths in the environment E from a point $a \in A$ to a point $b \in B$. The key insight is that if a node n_i is not near the geodesic hull formed by $A = P_i(t)$ and $B = Q_i(t)$, then n_i knows with certainty that information flowing between the tracker and target need not pass through n_i. The node n_i therefore decides not to broadcast its knowledge. See Figure 4.

So far, the constraint has been expressed in terms of "nearness" to the geodesic hull. Intuitively, if the network density is relatively high, then we expect to be able to use a relatively small threshold for nearness in condition (4), knowing that other nodes closer to the shortest path are likely to receive the message and propagate the information. Likewise, a relatively sparse network suggests the need for a larger nearness threshold, as a hedge against the possibility of a large empty space in the network preventing information

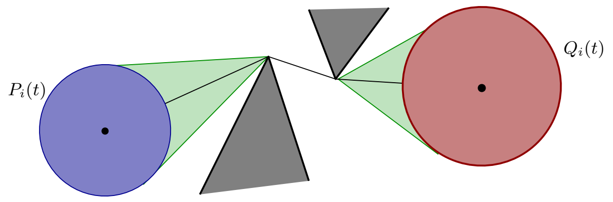

Fig. 4. The geodesic hull of two disks. If a node is far from the geodesic hull formed by the two disks in its disk-based I-state, it knows that information about the target can reach the tracker without it.

flow. Therefore, we allow nodes to broadcast messages when they, according to their disk-based I-states, could possibly be within distance $r_c - d$ of the shortest path between $p(t)$ and $q(t)$, in which d is the size of the largest open ball in E that does not contain any sensor nodes.

Therefore, every node broadcasts its knowledge only when it is within or near the geodesic hull of its disk-based I-state. As a result, information collected by nodes that see the target will propagate to the tracker without delay. Moreover, nodes that are not on or near the shortest path between target and tracker will, after receiving one of these messages, know that they need not broadcast any messages, and therefore remain silent. In this way, after an initial, one-time flooding, the disk-based I-states stored at each node enable the nodes to propagate messages only in a tight corridor around the relevant parts of the environment.

In practice, the true geodesic hull may be difficult to compute because the points $P_i(t)$ and $Q_i(t)$ may be connected by paths in many different homotopy classes. Therefore, our implementation approximates the geodesic hull by a point set consisting of all points in E that meet at least one of four criteria:

- Points in $P_i(t)$ or $Q_i(t)$.
- Points within distance $r_c - d$ of the shortest path from the center of $P_i(t)$ to the center $Q_i(t)$,
- Points inside the trapezoid formed by the four common tangent points of two particular circles. The first circle is $P_i(t)$. The second circle centered at the second vertex of the shortest path between the disk-based I-state centers and has radius $r_c - d$.
- Points inside the trapezoid formed *mutatis mutandis* from the penultimate vertex of the shortest path between centers and $Q_i(t)$.

This simplification takes advantage of the fact the information need only flow along *some* short path between the tracker and target, and not necessarily the definitive shortest path.

C. What data to transmit?

Each time a node broadcasts a message, it transmits its entire current disk-based I-state. This design is motivated by a desire to maximize the informative value of each message. This approach stands in contrast to schemes that simply forward messages without modification. The approach proposed here, in particular, indirectly facilitates *data aggregation*, by allowing sensor nodes to accumulate information in their disk-based I-states for a short period of time (via the timeouts

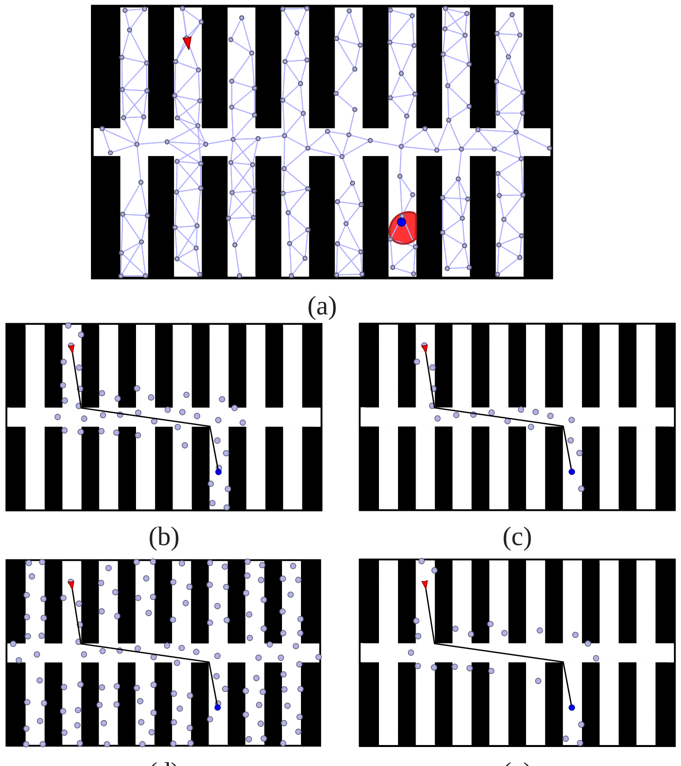

Fig. 5. Execution snapshots showing the behavior of the network nodes at one time step. (a) The complete network. The tracker's I-state is shaded. (b) Nodes that received a message at this time step. (c) Nodes that sent a message at this time step. (d) Nodes that are near the simplified geodesic hulls of their disk-based I-states. (e) Nodes that are *not* near their simplified geodesic hulls. These nodes know that they are not along the path from tracker to target, and therefore do not spend any energy sending messages.

described above) before generating a single message encapsulating the combined information from each of the messages it received.

D. Examples

Figure 1 depicts the algorithm's simulated behavior in a relatively simple environment, illustrating the motions of both the tracker and the target.

In Figure 5, we show several views of a single time step of the algorithm in a more complex environment. We observe that out of all the nodes that received the message at this time step (Figure 5b), only a portion of them will broadcast the message (Figure 5c) while the remaining nodes (Figure 5e) decide not to propagate the information any farther since they know that they are not sufficiently close to the shortest path. Note especially Figure 5d, which shows that a large fraction of the nodes cannot rule out being near the shortest path between tracker and target. For some nodes, this is because they are in fact near the shortest path. For others this is because they are far from the shortest path, and the routing algorithm had effectively prevented data from reaching them. As a result, these nodes have disk-based I-states that are large, making them unable to rule out the possibility of being near the shortest path. However, this limitation will not affect the energy efficiency of our algorithm for two reasons. First, a

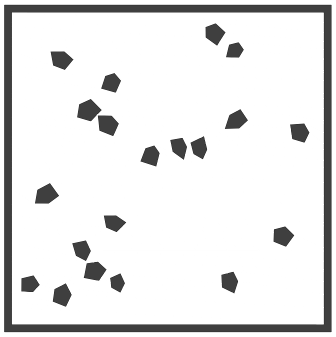

Fig. 6. The simple environment used in the experiment of Section VI-A. The obstacles are generated randomly.

collection of nodes (Figure 5e) around the shortest path which do not forward the message acts as "buffer zone." Second, even if such a message did reach a node with very large radii in its disk-based I-state, that node would immediately update its disk-based I-state, and realize that it is far away from the shortest path. This behavior is typical for our algorithm, and shows how our smart local routing prevents messages from spreading to irrelevant parts of the environment.

VI. Implementation and Evaluation

We have implemented this algorithm in simulation. In this section, we present a quantitative evaluation of its performance in comparison to an existing method and to several naïverouting protocols.

A. Comparison to TTL-based broadcast

Figure 6 shows a relatively simple environment populated by a collection of randomly-placed obstacles. To demonstrate that our approach outperforms the existing TTL-based technique [13], we performed a series of 10 trials in which the starting positions of the tracker and target and the placements of the sensor nodes were selected randomly. For each trial, we simulated the startup performance of both tracking algorithms, with 5 different parameter settings each:

- The TTL-based approach, with its timeout parameter a set to $5, 8, 11, 14,$ and 17.[2]
- The current smart local routing approach, with the minimum time interval between two successive broadcasts τ set to $3, 6, 9, 12$ and 15.

The performance of each algorithm, averaged over all trials, is shown in Figure 7. Because the start-up phase of our tracking problem has two optimality criteria, each dimension of the plot measures one of the optimality criteria. The origin of the plot represents the (unachievable) ideal of perfect tracking with no energy cost. Each data point, therefore, dominates—in the sense of faring better in both performance criteria—any other data points that are both above it and to its right. These results confirm that the smart local routing is superior to the

[2]The precise role played by this parameter is described in the original paper [13]. In this context, it is only relevant to note that the performance of that algorithm depends on a tunable parameter, denoted here as a.

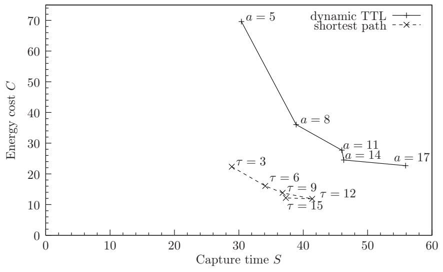

Fig. 7. Start-up performance for the environment in Figure 6.

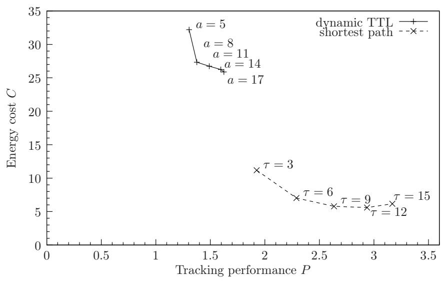

Fig. 8. Steady-state performance for the environment in Figure 6.

the dynamic TTL algorithm in the startup phase regardless of the parameter value τ, in the range that we tested.

To evaluate the performance of our algorithms in the steady-state phase of tracking, we performed ten additional trials in which the tracker and target are started near one another. We used $T = 1000$, and averaged the results over all ten trials. Figure 8 depicts the results of this experiment, showing that the dynamic TTL method maintained slightly better tracking performance at the cost of higher energy consumption.

To ensure that these results generalize to more complex environments, we repeated the same experiment in the maze-like environment depicted in Figure 9. The results, in which the improvement is even more pronounced, appear in Figures 10 and 11.

B. Comparison to existing algorithms

Finally, to ensure that our approach compares favorably to known methods, we performed a final experiment testing its performance against two basic protocols:

1) *Flooding*, in which every message is forwarded to every node in the network. This approach delivers every message to the tracker and generates very accurate tracking but also very large energy consumption.
2) *Static TTL*, in which every message is broadcast only to each node's immediate neighbors. This approach is very energy efficient, but leads to poor tracking performance because information is propagated only locally.

This experiment used the environment shown in Figure 6. The final results, which are averaged across ten trials for startup performance and ten trials for steady-state performance, appear in Figure 12.

VII. Conclusion

We presented a target tracking algorithm that uses a collaboration between a sensorless robot and a network of unreliable

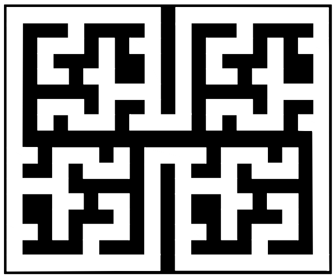

Fig. 9. A maze-like environment.

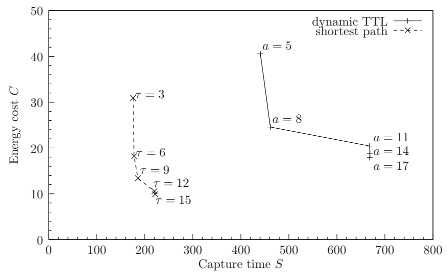

Fig. 10. Start-up performance for the environment in Figure 9.

sensor nodes. Simulations demonstrate that this algorithm has good performance in balancing energy efficiency with tracking accuracy, even in the presence of false positive sensor errors. However, a number of interesting questions remain unanswered.

Most interestingly, the algorithm we use to compute the information states would be unsuitable for systems in which the sensors are subject to false positive errors. We are currently investigating temporal filters to detect intermittent failures, and spatial filters that can detect persistent failures of individual sensors. Our preliminary results (omitted for space reasons) suggest that these techniques are quite effective.

ACKNOWLEDGMENTS

Heather O'Kane assisted with preparing the experimental results. This work is partially supported by grants from the University of South Carolina, Office of Research and Health Sciences Research Funding Program.

REFERENCES

[1] M. Batalin and G. S. Sukhatme, "The analysis of an efficient algorithm for robot coverage and exploration based on sensor network deployment," in *Proc. IEEE International Conference on Robotics and Automation*, Apr 2005, pp. 3489–3496.

[2] T. Clausen and P. Jacquet, "RFC-3626: Optimized Link State Routing Protocol (OLSR)," 2003.

[3] L. J. Guibas and J. Hershberger, "Optimal shortest path queries in a simple polygon," *Journal of Computer and Systems Sciences*, vol. 39, no. 2, pp. 126–152, 1989.

[4] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "Visibility-based pursuit-evasion in a polygonal environment," *International Journal on Computational Geometry and Applications*, vol. 9, no. 5, pp. 471–494, 1999.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proc. International conference on Mobile computing and networking*, 2000, pp. 56–67.

[6] V. Isler, S. Kannan, and S. Khanna, "Locating and capturing an evader in a polygonal environment," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2004.

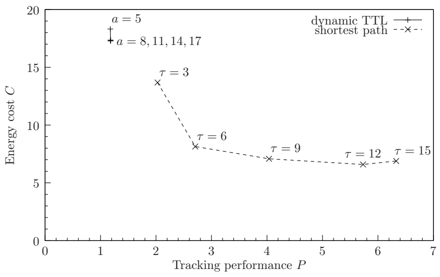

Fig. 11. Steady-state performance for the environment in Figure 9.

	Startup		**Steady state**	
	S	C	P	C
flooding	28.1	1064.0	1.2	1232.3
static TTL	59.5	21.7	1.8	25.6
dynamic TTL	30.4	69.6	1.3	32.2
smart local routing	**36.8**	**13.8**	**2.6**	**5.8**

Fig. 12. Comparison to existing message delivery schemes. Static TTL used a TTL value of 1. Dynamic TTL used $a = 5$. Smart local routing used $\tau = 9$.

[7] B. Jung and G. S. Sukhatme, "Cooperative multi-robot target tracking," in *Proc. International Symposium on Distributed Autonomous Robotic Systems*, Minneapolis, Minnesota, Jul 2006, pp. 81–90.

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

[9] 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*, 1997, pp. 731–736.

[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, 2002.

[11] 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.

[12] R. Murrieta-Cid, B. Tovar, and S. Hutchinson, "A sampling-based motion planning approach to maintain visibility of unpredictable targets," *Autonomous Robots*, vol. 19, no. 3, pp. 285–300, 2005.

[13] J. M. O'Kane and W. Xu, "Energy-efficient target tracking with a sensorless robot and a network of unreliable one-bit proximity sensors," in *Proc. IEEE International Conference on Robotics and Automation*, 2009.

[14] R. Olfati-Saber, "Distributed tracking for mobile sensor networks with information-driven mobility," in *Proc. American Control Conference*, 2007, pp. 4606–4612.

[15] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, 1997, pp. 90–100.

[16] N. Shrivastava, R. M. 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*, 2006, pp. 251–264.

[17] G. Simon, M. Maróti, A. 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*, 2004, pp. 1–12.

[18] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proc. International Conference on Embedded Networked Sensor Systems*, 2003, pp. 14–27.

[19] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proc. IEEE INFOCOM*, 2002, pp. 1567–1576.

[20] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 61–72, 2002.

[21] Y. Zou and K. Chakrabarty, "Distributed mobility management for target tracking in mobile sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 872–887, 2007.