Energy-efficient target tracking with a sensorless robot and a network of unreliable one-bit proximity sensors

Jason M. O'Kane and Wenyuan Xu

Abstract— Existing target tracking algorithms require the tracker to have access to information-rich sensors, and may have difficulty recovering when the target is out of the tracker's sensing range. In this paper, we present a target tracking algorithm that combines an extremely simple mobile robot with a networked collection of wireless sensor nodes, each of which is equipped with an unreliable, limited-range, boolean sensor for detecting the target. The tracker maintains close proximity to the target using only information sensed by the network, and can effectively recover from temporarily losing track of the target. Our approach combines a protocol for the sensor network that conserves energy by dynamically adjusting the time-to-live for packets it transmits with a reactive strategy for the tracker based on its information state. We present an implementation along with experimental results. Our experimental results show that our system achieves both good tracking precision and low energy consumption.

I. INTRODUCTION

Tracking problems for mobile robots have received substantial attention in recent years. Informally, 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 [2], [13], [15]. However, these existing methods for robotic tracking are hampered by two primary limitations.

1. Locality: 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 [7].

2. Sensing complexity: Prior work assumes that the robot has access to sensors that are (in spite of their local nature) relatively powerful and information-rich, such as visual or range sensors. 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.

In this paper, we propose a tracking technique that resolves these limitations by allowing the robot to utilize a wireless

J. M. O'Kane and Wenyuan Xu are with the Department of Computer Science and Engineering, University of South Carolina, 301 Main St., Columbia, SC 29208, USA. {jokane, wyxu}@cse.sc.edu
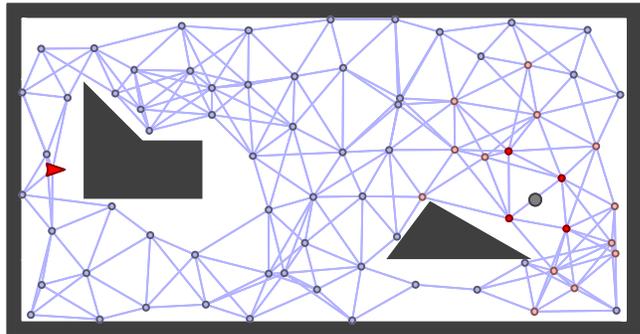
Fig. 1. An example tracking problem, in which a tracker (triangle, left side) seeks to find and maintain close proximity to a target (circle, right side). A wireless sensor network deployed in the environment provides the tracker with information about the target's whereabouts.

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 further simplify the proposed system, we assume the sensors node 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 sensor experience frequent false negative errors. To utilize this coarse location information, our approach makes extensive use of the concept of *information states* [12], 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.

Sensor networks, one key component of the proposed system, are characterized by limited energy and communication resources. 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 [22]. Thus, to extend network lifetime, the network protocols should limit the total number of messages required.

The contribution of this paper is to propose and evaluate a tracking technique for a robot cooperating with a sensor network with the following features.

- Nothing other than a maximum speed is known about the target's motion.
- The tracking robot has no sensors that directly provide information about the target.
- 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.
- Each sensor node has a limited energy budget for making transmissions.

The remainder of this paper has the following structure. We begin the paper by reviewing related work in Section II. We next formalize the tracking problem in Section III. In Section IV, we present our approach, including the protocol to deliver sensing data to the tracker, and the strategy for controlling the tracker. In Section V, we present our validation effort and discuss experimental results. We wrap up the paper by providing concluding remarks in Section VI.

II. RELATED WORK

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 [13], sampling-based [15], and reactive approaches [14]. Other approaches for tracking have employed wireless sensor networks (WSNs). WSNs have been deployed to track the positions of humans [4], moving vehicles [23], and other moving targets [11], [19].

The idea of combining WSNs with mobile robots has been investigated by Sukhatme et al. [3], [10]. In particular, mobile robots are introduced to WSNs to facilitate sensor network deployment with the goal of achieving good sensor coverage [3]. 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 [10], 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 tracing performance at reasonable operational cost for the network while using simple sensor devices and robots.

Finally, the use of mobile sensor networks, in which individual nodes have both sensing and motion capability, has been proposed as a means to track moving targets [16], [24]. The primary concern in this area is to track the targets while maintaining the network connectivity. We propose a different architecture in which the connectivity problem and mobility issues are decoupled.

III. PROBLEM STATEMENT

This section formalizes the tracking problem we consider. The system is illustrated in Figure 1.

A point target moves unpredictably, but with maximum speed s_{tgt}, in a closed, bounded, polygonal, planar environment E. The environment need not be simply connected.

Execution begins at time $t = 0$ and ends at some final time $t = T$. Let $q(t)$ denote the position of the target at time 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 $u(t)$, so that $dp/dt = u$. The velocity is constrained by a maximum speed s_{trk}. We assume that the tracker is perfectly localized within E, using either standard sensor-based localization methods [5], or GPS [18]. Therefore $p(t)$ is always known. 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 $n_1, \ldots, n_k$. The nodes localize themselves using one of the well-known sensor network localization schemes [1], [8], [17]. 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.[1] Specifically, each node n_i can:

1) Possibly detect the target whenever $||q(t) - n_i|| \le r_s$. This sensing is boolean, in the sense that 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.

2) Broadcast messages to all nodes n_j for which $||n_i - n_j|| \le r_c$. Although we assume that the time required for each transmission is negligible, these broadcasts are subject to intermittent communication failures. In our approach, the content of these messages is a description of a circle known to contain the target, along with a timestamp indicating when that information was collected. Specifically, the content of message is an ordered pair (c, t), listing the center of the circle along with its timestamp. The circles described by these messages all have radius r_s. Each node can initiate new messages and forward messages it has received.

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 $p(t)$. Finally, the tracker also uses this hardware to transmit a simple beacon signal, informing wireless sensor nodes of its presence. This beacon is detected by the node at n_i whenever $||p(t) - n_i|| \le r_c$.

A. Goal conditions

The tracker's primary objective is to minimize the average distance between $p(t)$ and $q(t)$ throughout the system's execution:

$$P = \frac{1}{T} \int_0^T ||p(t) - q(t)|| dt. \tag{1}$$

---

[1]To allow heterogeneous nodes would not require any significant changes to our approach.
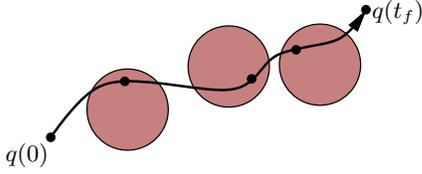
Fig. 2. A target position is consistent with a set of messages if there exists a starting target position and a feasible target trajectory that pass through each message's circle at the appropriate times reaching that target position.

Because the energy available to each wireless sensor node is often strongly limited, 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 $t = T$. We seek to minimize

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

We present experiments that explore the tradeoff between P and C in Section V.

B. Uncertainty and information states

Since the current state $x(t)$ is not necessarily known to the tracker, the challenges for the tracker are first to efficiently represent its knowledge, and second to use this representation to choose motions for the tracker. Our approach is based on the idea of computing the tracker's *information states*. In this context, the information state is the set of possible states that are consistent with the information the tracker has received. The tracker computes its information state, then uses it to choose its motions.

Formally, if the tracker has received m messages

$$\{(c_1, t_1), \ldots, (c_m, t_m)\}, \qquad (3)$$

as of some time t_f, then a target position q' is consistent with those messages if and only if there exists a continuous trajectory $q : [0, t_f] \rightarrow E$ such that $dq/dt \leq s_{tgt}$ for all $t \in (0, t_f)$, and $q(t_f) = q'$. Figure 2 illustrates this definition. Note the implicit assumption that the tracker starts with no information about the target's location. We use the notation $Q(t)$ for the set of target positions consistent with the messages received by the tracker up to time t. The tracker always knows its own position, so the information state (that is, the set of possible states) at time t is

$$\eta(t) = \{p(t)\} \times Q(t). \qquad (4)$$

Let $\mathcal{I}$ denote the space of all such information states. Because the information state is a complete picture of the knowledge available to the tracker, we describe the tracker's strategy as a function mapping information states to tracker velocities:

$$\pi : \mathcal{I} \rightarrow \{u \in \mathbb{R}^2 \mid ||u|| \leq s_{trk}\}. \qquad (5)$$

We discuss methods to maintain a representation of $\eta(t)$ in Section IV-B, and propose a greedy strategy π in Section IV-C.
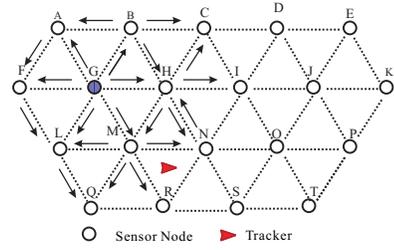

Fig. 3. A illustration of TTL-based broadcast.

IV. ALGORITHM

This section presents our tracking algorithm for the system described in Section III. We divide the presentation into three parts: (1) how to efficiently deliver target location information to the moving tracker (Section IV-A), (2) how to compute the information state of the target based on this information (Section IV-B), and (3) how to use the information state to choose motions for the tracker (Section IV-C).

A. Sensing and Data Delivery

As discussed in Section III, we consider network of k nodes spread throughout E. Whenever a node detects the target it generates a message containing its own coordinates as the circle center and the current timestamp. This initiates the message delivery process.

The design of the message delivery protocol is complicated by two factors. First, the location of the intended receiver, the tracker, is unknown in general to the network nodes. Most of the existing routing protocols for sensor networks, such as direct diffusion [9] and spanning tree based routing [21], are composed of a routing discovery phase and data delivery phase. Such protocols work well with stationary sinks, but are unsuitable for a mobile receiver. Thus, we choose broadcast as our message delivery protocol.

Second, the sensor nodes operate on batteries, so the broadcast protocol must be sensitive to this energy constraint. To accomplish this, we pretend to each message a header containing a globally unique *sequence number* and a non-negative integer *time-to-live* (TTL). We use these headers in two ways:

1) A node will only forward each message once, using the sequence number to track which messages it has forwarded already.
2) Each time a message is forwarded, its TTL is decreased by 1. Messages with TTL = 0 are discarded. As a result, the initial TTL for a node determines the maximum number of hops the message can travel on the network.

Each node forwards every message it receives as long as neither of these drop conditions is met.

Figure 3 depicts an example of sensor nodes along with the tracker. Suppose that node G detects the presence of the target. Node G generates a message with TTL = 2, a new sequence number, and its location, and sends out the message. Its neighbor M receives the message, decreases the TTL by 1 and broadcasts it. Similarly, node N receives the

```
Algorithm: Adjust_TTL
if trackerSeen() or not targetSeen() then
    messageTTL ← MIN_TTL
    timeoutCount ← 0
else
    timeoutCount ← timeoutCount +1
    if timeoutCount ≥ TTL_TIMEOUT then
        timeoutCount ← 0
        messageTTL ← 2 · messageTTL
    end
end
```

Fig. 4.   The TTL adjustment algorithm.

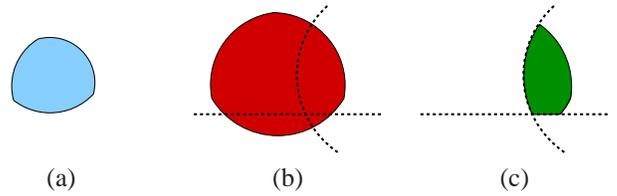(a)   (b)   (c)

Fig. 5.   Computing the 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.

message, decreases TTL, but will not forward the message since TTL = 0. In this example, the tracker is within the radio range of node M, so it successfully receives the message from G.

How can we choose the TTL value for a new message? Ideally, we want to set the TTL to the minimum hop count needed for messages to reach the tracker. Unfortunately, because the precise location of the tracker is unknown to the sensor nodes, this value is not available. Large TTL can guarantee the delivery of the message at the cost of high energy consumption, whereas small TTL requires less energy consumption, at the risk that messages may never reach the tracker. The small TTL is especially harmful when the tracker starts its execution with no knowledge of the target's position, or when it loses track of the target.

To address the issue of choosing TTL, we propose to dynamically adjust the TTL value. In the steady state, the tracker should be within the vicinity of the target. Thus, we use a small TTL value by default. When the tracker is far from the target, we dynamically adjust the TTL so that the message will reach tracker without flooding the entire network. In order to adjust TTL, we need to know whether the tracker is in the proximity of the target. One observation is that when the tracker follows the target closely, the sensor nodes that sense the target are within the vicinity of the tracker as well. These nodes should receive the beacon signal sent by the tracker. When the tracker is far from the target, these nodes will only observe the target, not the tracker.

Based on these observations, we propose the TTL adjusting algorithm shown in Figure 4, which is inspired by the TCP congestion control algorithm. Each node executes Adjust_TTL() at small, fixed time intervals. If the node senses the target and the tracker, the MIN_TTL value is used. Otherwise, the node will double its TTL after not hearing from the tracker for TTL_TIMEOUT number of time slices. We present experiments evaluating various choices for TTL_TIMEOUT in Section V.

The overall effect of this algorithm is that, when a node detects the target for a period of time without also detecting the tracker, the TTL for messages from that node is gradually increased in an attempt to ensure that the tracker receives the messages. When the tracker arrives or the target departs, the TTL is reset.

B. Computing the information state

The previous section described the operation of the sensor network, intended to deliver messages to the tracker describ-

ing circular regions that contain the target. It remains to describe how the robot can use this information to minimize its distance from the target.

First, recall the definition of the information state $\eta(t)$, which is the set of possible states at time t. Because the definition of $\eta(t)$ contains an existential quantifier over target trajectories, it is not directly useful for computing the information states. Instead we perform iterative updates, maintaining the current information state and updating it when time passes and when new messages are received. We start with the initial information 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 at $Q(t_2)$. Note that this approach may slightly overestimate the information 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.

2) When a message (c, t) is received, the existing information state is updated to the correct $\eta(t)$ by performing an intersection with a disk with center c and radius r_s.

Figure 5 illustrates each of these updates. Our implementation approximates the curved boundaries of the information states as polygonal regions, and uses the GPC General Polygon Clipper Library [20] to perform both types of updates.

C. Tracker strategy

Finally, we describe how the tracker moves. Recall that the tracker's chosen velocity $u(t)$ is a function of its information state $\eta(t) = (p(t), Q(t))$. Notice that, aside from knowing its own position 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 $Q(t)$. Note, however, that the centroid of $Q(t)$ may
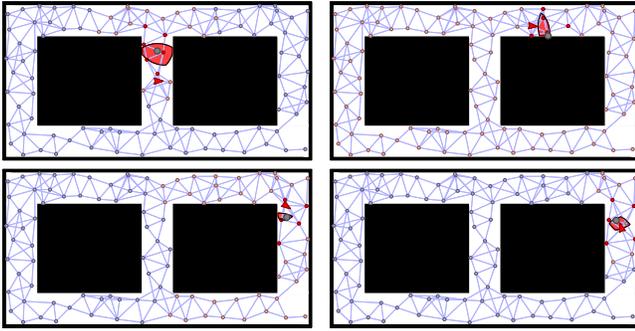
Fig. 6. A sample execution of our algorithm inside a "cinder block" shaped environment. The information state is shaded. The tracker is denoted by a triangle and the target is the grey circle.
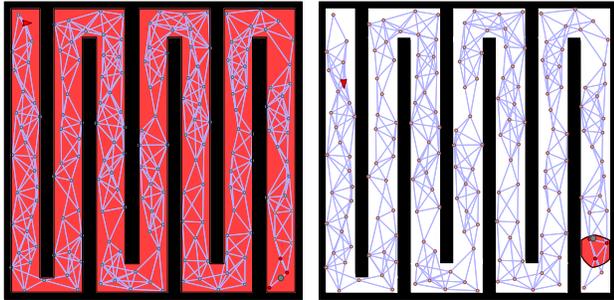

Fig. 7. [left] An initial condition with large uncertainty for the tracker. [right] The network provides information about the target location from far across the environment.

not be inside E. Based on these observations, we propose 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 $Q(t)$ 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 [6].

V. Implementation and Evaluation

A. Example executions

We have implemented this algorithm in simulation. Figure 6 illustrates its operation in a rectangular environment with two large obstacles. Several snapshots of the execution are shown, starting from an initial condition in which the tracker and the target are near each other.

Figure 7 shows a slightly more complex situation, in which the tracker and the target are initially separated by a large distance. In this example, the tracker and the target are separated by approximately 60 hops in the network. As a result, no messages reach the tracker until at least one node has experienced 6 timeouts. Note that during this time, in the absence of additional information, the tracker moves toward the centroid of E.

B. Experimental evaluation

First, to evaluate the influence of the TTL_TIMEOUT parameter, we performed a series of quantitative experiments. We varied the TTL_TIMEOUT from 1 to 20 time steps and measured both P and C, the evaluation metrics introduced in Section III-A. For each value of TTL_TIMEOUT, we

Fig. 8. The randomly-generated environment used in our quantitative experiments.

executed 5 trials in the environment shown in Figure 8, using random placements of $k = 150$ nodes. Each trial lasted $T = 500$ time steps, with the sensors collecting information once per time step. Sensor failures were modeled probabilistically. We set each sensor's failure rate set to 50%, which means sensor fails to detect the presence of the target 50% of time.

We plot the average distance P and the average number of message broadcasted C with different TTL_TIMEOUT values in Figure 9. This experiment demonstrates the expected tradeoff between tracking performance and energy-efficient network operation. For instance, larger TTL_TIMEOUT indicates that TTL will be increased less frequently, leading to longer delay for the tracker to recover from loss track of the target, but smaller C. Under these conditions, values of TTL_TIMEOUT lower than 5 generate excessive increases in energy consumption in exchange for only modest improvements tracking accuracy. Thus, we chose TTL_TIMEOUT = 5 for the rest of our experiments.

To evaluate the effectiveness of our dynamic TTL algorithm we compared it to two naive network schemes:

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 with a fixed TTL of 1. This approach is very energy efficient, but leads to poor tracking performance because messages are forwarded only locally.

In this experiment we used TTL_TIMEOUT = 5 for dynamic TTL. The results, which appear in Figure 10, are based on the same conditions as the previous experiment, and likewise are averaged across five trials for each network scheme. The results demonstrate that our dynamic TTL approach achieves, in some sense, something close to "the best of both worlds" in balancing tracking precision and energy efficiency.

VI. Conclusion

We presented a target tracking algorithm that uses a collaboration between a sensorless robot and a network of unreliable sensor nodes. Experiments demonstrate that this algorithm has good performance in balancing energy efficiency with tracking accuracy. However, a number of interesting questions remain unanswered.

First, observe that the algorithm we use to compute the tracker's information state would be unsuitable for systems in which the nodes are subject to false positive errors. We
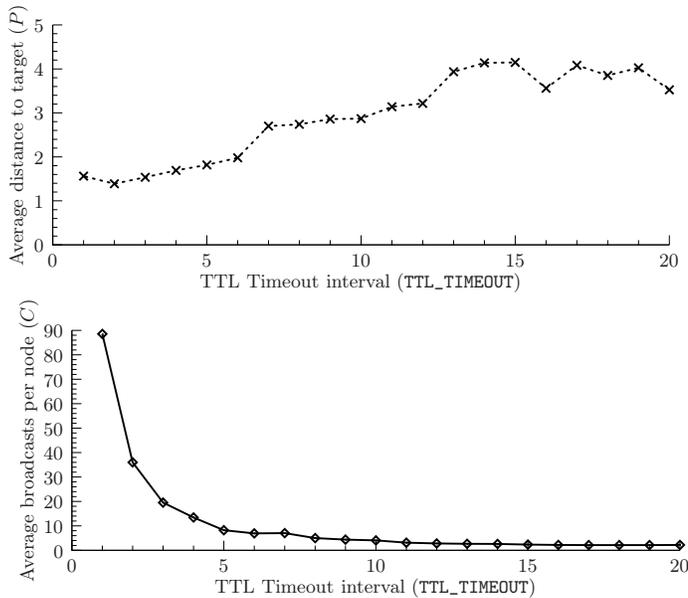
Fig. 9. Algorithm performance for varying values of TTL_TIMEOUT.

	P	C
Flooding	2.32	279.74
Dynamic TTL	2.92	7.95
Static TTL	5.57	2.06

Fig. 10. Results comparing our dynamic TTL technique to two naive network schemes. Our algorithm achieves tracking accuracy (P) comparable to flooding and energy efficiency (C) comparable to static TTL.

are currently investigating several potential extensions lift this restriction without sacrificing efficiency or requiring additional assumptions on the motion of the target.

One important avenue for potential improvement is to exploit the message content to improve the system's performance even more. In particular, each node can maintain its own information state, reflecting the information available to it from both its own observations and messages it has forwarded. Then each node can make more efficient routing decisions and reduce the overall communication requirements by combining multiple messages using geometric intersection. This idea can be extended even more by allowing nodes to send messages describing the position of the tracker, instead of only target-related messages.

ACKNOWLEDGMENTS

REFERENCES

[1] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. IEEE INFOCOM*, 2000.

[2] T. Bandyopadhyay, Y. Li, M. H. A. Jr., and D. Hsu, "Stealth tracking of an unpredictable target among obstacles," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2004, pp. 43–58.

[3] M. Batalin and G. S. Sukhatme, "Sensor coverage using mobile robots and stationary nodes," in *SPIE Conference on Scalability and Traffic Control in IP Networks II (Disaster Recovery Networks)*, Aug 2002, pp. 269–276.

[4] P. Chen, S. Oh, M. Manzo, B. Sinopoli, C. Sharp, K. Whitehouse, O. Tolle, J. Jeong, P. Dutta, J. Hui, S. Schaffert, K. Sukun, J. Taneja, B. Zhu, T. Roosta, M. Howard, D. Culler, and S. Sastry, "Instrumenting wireless sensor networks for real-time surveillance," in *Proc. IEEE International Conference on Robotics and Automation*, 2006, pp. 3128–3133.

[5] D. Fox, S. Thrun, W. Burgard, and F. Dallaert, "Particle filters for mobile robot localization," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. Berlin: Springer-Verlag, 2001, pp. 401–428.

[6] 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.

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

[8] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proc. International conference on Mobile Computing and Networking*, 2003, pp. 81–95.

[9] 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.

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

[11] W. Kim, K. Mechitov, J. Choi, and S. Ham, "On target tracking with binary proximity sensors," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005, p. 40.

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

[13] 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.

[14] 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.

[15] R. Murrieta-Cid, B. Tovar, and S. Hutchinson, "A sampling-based motion planning approach to maintain visibility of unpredictable targets," *Autonomous Robots*, pp. 285–300, 2005.

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

[17] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. International Conference on Mobile Computing and Networking*, 2000, pp. 32–43.

[18] G. Reina, A. Vargas, K. Nagatani, and K. Yoshida, "Adaptive kalman filtering for gps-based mobile robot localization," in *Proc. IEEE International Workshop on Safety, Security and Rescue Robotics*, 2007.

[19] 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.

[20] B. R. Vatti, "A generic solution to polygon clipping," *Communications of the ACM*, vol. 35, no. 7, pp. 56–63, Jul. 1992.

[21] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 14–27.

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

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

[24] 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.