On the value of ignorance: Balancing tracking and privacy using a two-bit sensor

Jason M. O'Kane

Department of Computer Science and Engineering, University of South Carolina
jokane@cse.sc.edu

Abstract: We consider a target tracking problem in which, in addition to sensing some information about the position of a mobile target, the tracker must also ensure that the *privacy* of that target is preserved, even in the presence of adversaries that have complete access to the tracker's sensor data. This kind of problem is important for many kinds of robot systems that involve communication systems or agents that cannot be fully trusted. In this paper, we (1) introduce a formal, quantitative definition for privacy, (2) describe algorithms that allow a robot to maintain conservative estimates of its performance in terms of tracking and privacy, (3) give strategies for the tracker to maximize its tracking performance, subject to constraints on the allowable privacy levels, and (4) present an implementation of these methods along with some experimental results.

1 Introduction

Robots must deal continually with uncertainty. The general problem of sensing and acting to reduce uncertainty is well-studied and continues to receive attention. This paper considers the related but complementary problem of sensing and acting in order to *maintain* uncertainty, rather than eliminating it. To motivate this (perhaps counterintuitive) idea, consider the following problem (inspired by [9, 10]):

> **Panda tracker problem:** A giant panda moves unpredictably through a wilderness preserve. A mobile robot tracks the panda's movements, periodically sensing partial information about the panda's whereabouts and transmitting its findings to a central base station. At the same time, poachers attempt to exploit the presence of the tracking robot — either by eavesdropping on its communications or by directly compromising the robot itself — to locate the panda. We assume, in the worst case, that the poachers have access to any information collected by the tracking robot, but they cannot control its motions. The problem is to design the tracking robot so that the base station can record coarse-grained information about the panda's movements, without allowing the poachers to obtain the fine-grained position information they need to harm the panda. See Figure 1.

In this problem, the tracking robot must collect some information about the panda's location, but cannot collect too much information without endangering the *privacy* (and therefore, the safety) of the panda. More generally, this sort of privacy concern is relevant in at least three kinds of robotics applications:

(1) Applications which require communication over an untrusted channel.
(2) Applications which require communication with an untrusted recipient.
(3) Applications in which the robot itself cannot be fully trusted.

For each of these classes of problems, it is essential to design a robot and its programming to ensure that the information it collects or transmits is not harmful to disclose. We emphasize that the goal is to ensure that privacy is preserved even when the best possible use is made of all the available information. This approach specifically precludes the possibility of getting a "free ride" by intentionally forgetting. It also allows us

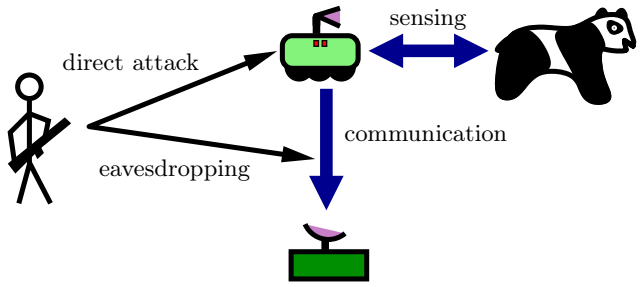

Fig. 1. Basic components of the panda tracker problem. The tracking robot needs to sense the panda's location and communicate this information to the base station, while preventing the poacher from exploiting this information to locate the panda.

to avoid explicitly considering the actions and knowledge of the poachers — we simply ensure that it is acceptable for them to know everything that the tracker knows.

In this paper, we consider one version of the panda tracker problem, in which a *target* moves in the plane unpredictably but with bounded velocity. A *tracker* monitors that target's motions using a sensor that reports very coarse information about the direction from the tracker to the target. A crucial first observation is that the ability to maintain privacy is closely related to the informative value of the robot's sensors. A tradeoff exists between strong sensors (which make tracking easy but privacy difficult) and weak sensors (which make privacy easy to maintain at the expense of tracking accuracy). We present results suggesting that a reasonable balance between tracking and privacy can be obtained using sensors that provide relatively little information.

Our approach is based on analyzing the tracker's *information states* as they change over time. In this context, the information state at a particular time is simply the set of states that are consistent with the history of actions executed and sensor readings obtained by the tracker up to that time. A distinctive feature of this work is that we maintain upper and lower bounds on the extent of the information state, without needing to compute the information states themselves.

The primary goal of this work is to investigate the role that privacy concerns can play in robotic tracking problems. After reviewing related prior work in Section 2, this paper makes several new contributions.

1. We give quantitative definitions for tracking and privacy, in terms of the tracker's information states. These definitions appear in Section 3.
2. We describe representations and algorithms for the tracker to maintain a lower bound on privacy and an upper bound on the tracking accuracy achieved throughout its execution. These algorithms, which are introduced in Section 4, require $O(1)$ storage and $O(1)$ update time, so they are well-suited to implement, for example, on extremely simple mobile sensor platforms.
3. We derive reactive strategies, in Section 5, for planning the motions of the tracker that keep privacy and tracking within acceptable bounds. Implementations of these algorithms are presented.

Concluding discussion appears in Section 6.

2 Relationship to prior work

The panda tracker problem was first proposed in the context of wireless sensor networks. Both temporal [9] and spatial [10] privacy of an observed target can be protected by careful design of the network's routing protocols. Crucial to both of these works is their application of Kerckhoffs' Principle, under which it is assumed that the adversaries have complete knowledge of the protocol used by the network. This view is paralleled by our assumption that the adversaries have access to all of the information the tracker does.

We encompass both forms of privacy by considering how position information changes over time. To our knowledge, the present work is the first to directly examine the effects of sensing and action on privacy.

Our work is also closely related to the lines of research that seek to understand tasks by solving them in spite of severe sensing limitations. Problems in manipulation [3,5], localization [4,21], navigation [11,12,15–17,22], mapping [8], and pursuit-evasion [6,7] have been solved for such systems. These examples are meant to be representative, rather than exhaustive. A common thread through much of this work is a two-step approach, based on solving the passive information state update problem before considering how to choose actions actively. In each case, the solution depends on representing and updating the robot's knowledge in ways that make this active planning manageable. This work follows the same approach.

More specifically, target tracking problems have also been studied in the literature. Much of this work is focused on maintaining visibility between the tracker and the target within a cluttered environment. Specific methods have used dynamic programming [14], sampling-based [20], and reactive [19] approaches. The novelty of our work is that we are the first to explicitly include privacy in the formulation. One closely related variation is the stealth tracking problem, in which the tracker must maintain visibility of the target, while remaining near the boundary of the target's visibility polygon to avoid possible detection [1]. Our work differs primarily because we are concerned with the privacy of the target's location, rather than that of tracker's location, and because we consider a much weaker sensor.

3 Problem statement

This section formalizes the panda tracker problem. We begin in Section 3.1 with a general formulation for which we give a quantitative definition of privacy, then in Section 3.2 apply this formulation to the specific version of the panda tracker problem solved in this paper.

3.1 General formulation

Consider a formulation with the following elements:[1]

- A division of time into a sequence of discrete, but not necessarily equal length, *stages*, numbered $k = 1, 2, \ldots$. In each stage, the robot can both sense and act.
- A *state space* X. In stage k, the relevant information about the situation in the world is modeled by a *state* $x_k \in X$.
- A *distance function* $d : X \times X \to \mathbb{R}$ for pairs of states, under which X is a metric space.
- An *initial condition* $\eta_0 \subseteq X$ representing a set of possible starting states. Unless η_0 is a singleton set, the actual starting state is unknown to the robot.
- An *observation space* Y, so that $y_k \in Y$ models the sensor information collected by the robot at stage k.
- An *observation function* $h : X \to Y$, under which $y_k = h(x_k)$. That is, h describes how the observation y_k is determined by the current state x_k.[2]
- An *action space* U. The robot chooses one action $u_k \in U$ to execute in each stage.
- A *nature action space* Θ. The nature action $\theta_k \in \Theta$ at stage k models the effects of noise, actions by other decision makers, or both.
- A *state transition function* $f : X \times U \times \Theta \to X$ that describes how the stage changes. The state x_{k+1} at stage $k + 1$ is given by $x_{k+1} = f(x_k, u_k, \theta_k)$.

From these basic ingredients, we can make two additional definitions for convenience. First, define an iterated transition function to apply multiple transitions at once:

[1] A more complete treatment of this type of formulation appears in Chapter 11 of [13].

[2] Note that, for simplicity, this formulation implicitly assumes that sensing is deterministic. We briefly discuss the implications of this assumption in Section 6. Note that this is uniquely interesting for our problem because sensor noise, with the increased uncertainty it brings, may actually be helpful.

$$f(x, u_1, \theta_1, \ldots, u_k, \theta_k) = f(\cdots f(f(x, u_1, \theta_1), u_2, \theta_2) \cdots, u_k, \theta_k). \tag{1}$$

Second, denote the *preimage* of each observation y by $H(y)$, so that

$$H(y) = \{x \in X \mid h(x) = y\}. \tag{2}$$

The robot does not necessarily know its state, but instead must rely on its sensor-action history to make its decisions. At stage k, this information consists of $y_1, \ldots, y_k$ and $u_1, \ldots, u_{k-1}$. This history can be used to determine the set of possible states at stage k. The following two definitions make this notion precise.

Definition 1. *A state $x \in X$ is* consistent with *a sensor-action history $(y_1, u_1, \ldots, y_{k-1}, u_{k-1}, y_k)$ if there exists some $x_1 \in \eta_0$ and a sequence of nature actions $\theta_1, \ldots, \theta_{k-1}$ such that*

$$x = f(x_1, u_1, \theta_1, \ldots, u_{k-1}, \theta_{k-1}) \tag{3}$$

and

$$y_j = h(f(x_1, u_1, \theta_1, \ldots, u_{j-1}, \theta_{j-1})) \tag{4}$$

for each $j = 1, \ldots, k$.

Definition 2. *The* information state *(I-state) η_k at stage k is the set of all states consistent with the robot's sensor-action history. The* information space *(I-space) $\mathcal{I}$ is the powerset of X, which contains all possible I-states.*

The intuition is that a state x is a member of an I-state η_k if and only if the robot cannot conclusively rule out x as a possible state at stage k. Such I-states are useful because the robot can always keep track of its I-state, even when there is insufficient information to determine the underlying true state. Let $F : \mathcal{I} \times U \times Y \to \mathcal{I}$ denote an *information transition function* that describes how the I-state changes over time. Given an information state η_k, and action u_k, and an observation y_{k+1}, we have $\eta_{k+1} = F(\eta_k, u_k, y_{k+1})$. We can describe this information transition in terms of f and H:

$$F(\eta_k, u_k, y_{k+1}) = \{f(x, u_k, \theta) \mid x \in \eta_k, \theta \in \Theta\} \cap H(y_{k+1}). \tag{5}$$

How are these I-states related to privacy and tracking? Informally, privacy is preserved whenever the I-state is a relatively large set; conversely good tracking depends on keeping the I-state relatively small. This intuition leads us to the following quantitative definitions of privacy and tracking.

Definition 3. *A closed ball B is a* privacy disk *for an I-state η_k if $B \subseteq \mathrm{cl}(\eta_k)$. The* privacy margin *of an I-state η_k is the largest radius over all privacy disks of η_k.*

Definition 4. *A closed ball B is a* tracking disk *for an I-state η_k if $\mathrm{cl}(\eta_k) \subseteq B$. The* tracking margin *of an I-state η_k is the smallest radius over all tracking disks of η_k.*

Figure 2 illustrates these definitions, which depend on the metric function d. The intuition is that the privacy and tracking disks form inner and outer circles around the boundary of η_k. A privacy disk is a region within which no states can be ruled out; a tracking disk is a region outside of which every state can be ruled out. The radii of these two disks provide a quantitative measures of privacy and tracking. Note that the privacy disk of maximal radius is not necessarily unique (for example if η_k is the region between two concentric circles), but the minimal tracking disk *is* unique.

One might imagine an alternative to Definitions 3 and 4 based directly on the area or volume of the I-state. Such an approach would not use separate measures for privacy and tracking, but instead quantify the robot's uncertainty by the volume of its I-states and work to ensure that this value remains within given bounds. Our decision to use Definitions 3 and 4 rather than this type of volume-based approach is motivated by several considerations. First, our definitions are more strict than the volume-based alternative — an I-state will, in general, have volume greater than or equal to that of its largest privacy disk and less than or equal to that of its smallest tracking disk. Therefore, the performance bounds we obtain are valid for the

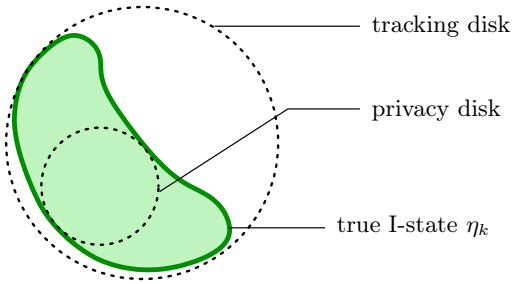

Fig. 2. An illustration of Definition 3. The boundary of the tracker's I-state must lie between the boundaries of the privacy and tracking disks.

volume-based definitions as well. Second, the fact that we are concerned only with the largest ball contained within and the smallest ball around the I-state allows us to maintain bounds on tracking and privacy that do not require computing the I-state's exact shape. Finally, for tracking in particular, the volume of the I-state may not represent the tracking accuracy very well. In extreme cases, the I-state may consist of several small regions distant from one another. Such an I-state would have small volume, but would require a large amount of travel to search completely.

3.2 The panda tracker problem

Now we can state the panda tracker problem as a specific instance of the general formulation from Section 3.1.

Problem formulation

A single target moves unpredictably in an obstacle-free plane. Its motions are continuous and are constrained by a known maximum velocity, but nothing else is known about its trajectory. A single tracker, modeled by a point with orientation in the plane, monitors the movements of the target using a low-speed, low-resolution bearing sensor that reports, in a coordinate system attached to the tracker, which quadrant contains the target. The tracker is able to move very quickly in comparison to the target. Each stage models a period of motion by both the target and the tracker, followed by a sensor reading by the tracker. Formally, we can describe this system as follows:

- The state space is $X = \mathbb{R}^2$. A single state represents the target's position relative to the tracker, in coordinate frame with the tracker's position at its origin and its positive y-axis in the direction the tracker faces. We denote the individual coordinates of a state using superscripts, so that $x = (x^{(1)}, x^{(2)})$ for each $x \in X$. Note, however, that it will sometimes be more straightforward to consider a global frame, specifying the position and orientation of the tracker and the position of the target relative to a stationary reference point.
- The distance metric d is the standard Euclidean metric over $\mathbb{R}^2$.
- The initial condition η_0 is a disk known to contain the target. This includes, as a special case in which the disk is centered at the origin, the situation where the tracker starts with an upper bound on the distance to the target.
- The action space U is the set of rigid body transformations achievable by the tracker within one stage (that is, between two consecutive sensor readings). For concreteness, we consider a robot that can translate and rotate freely with maximal velocity v_{trk} and maximal angular velocity ω_{trk}.
- The nature action space Θ is the set of translations of magnitude at most v_{tgt}, the displacement achievable by the target in a single stage.
- The state transition function $f : X \times U \times \Theta \to X$ applies the motion u_k of the tracker and the motion θ_k of the target to the current state x_k to compute the new state $x_{k+1} = f(x_k, u_k, \theta_k)$.

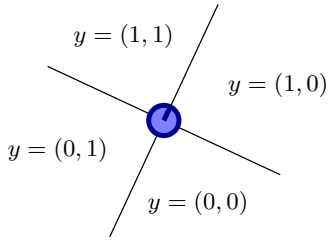

Fig. 3. Observation preimages for the panda tracker problem. The sensor reports which quadrant contains the target.

Fig. 4. An iRobot Create differential drive robot, equipped with a 4 infrared beacons and 4 infrared sensors. When deployed in pairs, such robots can track one another by sensing which quadrant contains the other robot.

- The observation space is $Y = \{0, 1\} \times \{0, 1\}$.
- The observation function $h : X \to Y$ returns the quadrant containing the target, according to

$$h(x_k) = h\left(x_k^{(1)}, x_k^{(2)}\right) = \left(\left[x_k^{(1)} \geq 0\right], \left[x_k^{(2)} \geq 0\right]\right),$$

  in which $[\,\cdot\,]$ is the indicator function that returns 1 if its argument is true and 0 otherwise. See Figure 3. Although we do not emphasize any particular hardware implementation, Figure 4 shows one possible realization of this kind of sensor.

Goal conditions

For purposes of comparison, we consider two specific, mutually independent choices for the tracker's goal:

G1: *Minimize tracking margin* — Keep the tracking margin as small as possible, without regard for the privacy margin.

G2: *Maintain privacy bound* — Keep the tracking margin as small as possible, while ensuring that the privacy margin is no smaller than a given minimum, denoted ρ.

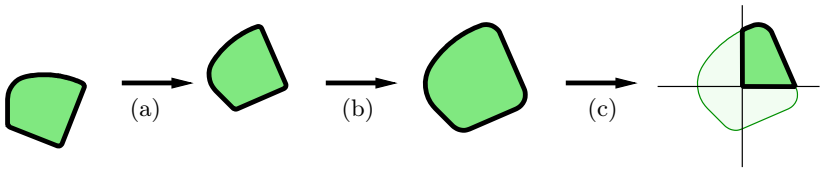

Fig. 5. Computing explicit information transitions for the panda tracker problem. (a) Rigid body transformation by u_k. (b) Minkowski sum with Θ. (c) Intersection with $H(y_{k+1})$.

In each case, we are concerned with the worst-case tracking and/or privacy margins, taken over all possible trajectories of the target. We discuss strategies to achieve these two goals in Section 5. One motivation for considering both goals is that the difference in tracking performance between G1 and G2 can be viewed, in some sense, as a measure of the amount of tracking granularity that must be sacrificed in order to maintain privacy. Section 5.3 presents experiments that quantify this tradeoff.

4 Passive updates

This section presents algorithms for maintaining a lower bound on the privacy margin and an upper bound on the tracking margin for the panda tracker problem defined in Section 3.2. These techniques are passive in the sense that we defer (to Section 5) the question of how to choose actions for the tracker, considering for now only how to estimate the performance during a given execution. We present two approaches to this problem: an optimal algorithm based on computing the I-state itself, and a second approach that is more efficient, but produces only upper and lower bounds on the tracking and privacy margins, respectively.

4.1 Updates directly from the I-state

For the first approach, consider how the tracker might maintain an explicit representation of the I-state η_k. The initial I-state η_0 is a disk. In subsequent stages, to compute η_{k+1} from η_k, we must perform three transformations on η_k (recalling that states are represented in a coordinate frame attached to the tracker):

1. Rigid body transformation by u_k, reflecting the motion of the tracker.
2. Minkowski sum of the resulting region with Θ, which is a disk of radius v_{tgt}, reflecting all possibilities for the unknown motion θ_k of the target.
3. Intersection of the resulting region with a quarterplane[3](the observation preimage $H(y_{k+1})$), reflecting the new information supplied by the sensor.

Brute force algorithms for these operations would take time linear in the complexity of boundary of η_k. In practice, the circular arcs arising in Step 2 can be approximated by polygonal chains.

With this information, the exact privacy margin can be computed by finding the largest disk inside η_k. The center of this largest disk must be equidistant from at least three distinct points on the boundary of η_k, and therefore it must be located at one of the vertices of the Voronoi diagram of the boundary of η_k. Algorithms are known to compute this Voronoi diagram in $O(n)$ time [2]. Likewise, the exact tracking margin can be found by computing, in $O(n)$ time [18], the smallest enclosing disk around η_k.

4.2 Indirect updates with limited memory

For some applications, including those in which the tracker is a very simple mobile sensor platform, computation and memory resources are at a premium. This section presents passive update algorithms appropriate

[3] We use the term quarterplane to refer to the planar intersection of two halfplanes with orthogonal boundary lines.

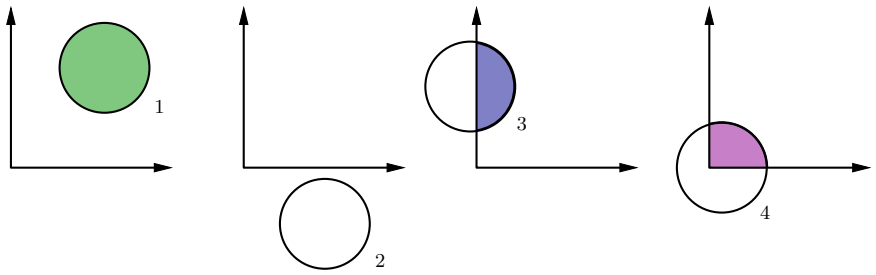

Fig. 6. Four cases that arise in updating P_k to P_{k+1} and T_k to T_{k+1}.

for such situations, which maintain estimates on the tracking and privacy margins but require only relatively small, constant amounts of time and space. The central idea is to define two sequences of disks, $P_1 \ldots, P_k$, and $T_1, \ldots, T_k$, so that, at each stage i, we have $P_i \subseteq \eta_i \subseteq T_i$. Note that these need not necessarily be the largest privacy disks or the smallest tracking disks for their respective I-states. In the initial condition, the I-state itself is a disk, so we start with $P_1 = T_1 = \eta_0$. In subsequent stages, we use P_k to compute P_{k+1} and T_k to compute T_{k+1}. The following sections describe this process. After the tracker computes P_{k+1} and T_{k+1}, it discards P_k and T_k.

Updating the privacy disk

Given a privacy disk $P_k \subseteq \eta_k$, an action u_k, and an observation y_{k+1}, we want to compute a new privacy disk $P_{k+1} \subseteq \eta_{k+1}$. Accounting for the motions of the tracker and the target is straightforward. As in Section 4.1, we perform a rigid body transformation on P_k by u_k, followed by a Minkowski sum with Θ. Since P_k is a disk, these operations can be realized by a translation of the center of P_k and an increase of its radius by v_{tgt}. Let P'_k denote the disk resulting from these operations.

It remains to consider the effects of the observation y_{k+1}. At stage k, no states within P_k can be ruled out. Therefore, we choose for P_{k+1} the largest disk inside of $P'_k \cap H(y_{k+1})$, resulting in a region in which no states in P_{k+1} can be ruled out for stage $k+1$. The problem remains to find the largest disk inside the region of intersection between a disk and a quarterplane. Depending on the position of P'_k relative to the axes of the quarterplane $H(y_{k+1})$, we have one four cases (Figure 6):

1. P'_k is fully inside $H(y_{k+1})$. The resulting intersection is P'_k itself, so no change is needed and $P_{k+1} = P'_k$.
2. P'_k is fully outside $H(y_{k+1})$. The resulting intersection is empty, so P_{k+1} is undefined. If this situation occurs, the tracker can no longer make any privacy guarantees. See Section 5 for a strategy that provably prevents this case from occurring.
3. P'_k is partially inside $H(y_{k+1})$, crossing only one of the quarterplane axes. In this case, we can ignore the other quarterplane axis and consider only the intersection between a halfplane and a disk. Use a coordinate frame in which P'_k is centered at the origin, and the relevant halfplane boundary is a vertical line $x = b$. See the left portion of Figure 7. The largest disk in this intersection region has radius $(b + \text{radius}(P'_k))/2$ and center $((b - \text{radius}(P'_k))/2, 0)$.
4. P'_k is partially inside $H(y_{k+1})$, crossing both quarterplane axes. For this case, choose a coordinate frame whose horizontal axis bisects the observation preimage quarterplane and whose vertical axis passes through the center of P_k. See the right portion of Figure 7. Let $(s, 0)$ denote the position of the quarterplane vertex and let $(0, c)$ denote the position of the center of P'_k in this frame. The largest disk in the intersection region has its center on the x-axis in this coordinate system, so let $(a, 0)$ denote the center of P_{k+1}. For a fixed a, the distance to the nearest point on the boundary of P'_k is $\text{radius}(P'_k) - \sqrt{c^2 + a^2}$. Notice also that the nearest point on each of the quarterplane axes has distance $\sqrt{2}(a - s)$ from $(a, 0)$. The resulting radius is maximized with these distances are equal, so we solve

$$\text{radius}(P'_k) - \sqrt{c^2 + a^2} = \sqrt{2}(a - s) \tag{6}$$

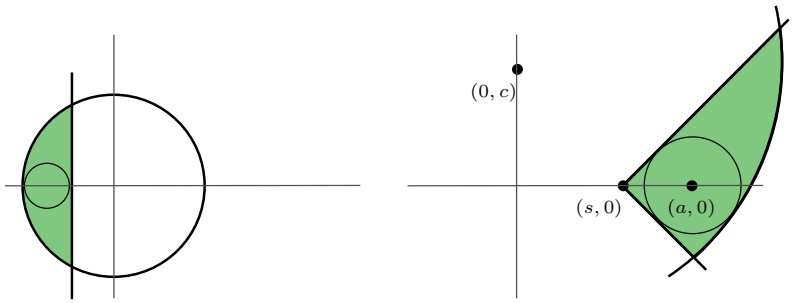

Fig. 7. Updating P'_k to P_{k+1}. The algorithm computes the largest disk inside the shaded region. [left] Case 3. The center of P'_k is at the origin, and the relevant axis of $H(y_{k+1})$ is a vertical line $x = b$. [right] Case 4. The horizontal axis bisects $H(y_{k+1})$ and the vertical axis bisects P'_k.

for a, using standard analytical techniques. The resulting a can be used to compute P_{k+1}.

Note that we are computing the *largest* privacy disk consistent with both the privacy disk retained from the previous stage and the new information obtained at stage k.

Updating the tracking disk

Computing a new tracking disk T_{k+1} from T_k, u_k, and y_{k+1} is similar to the procedure for privacy disk updates, but instead requires finding the smallest disk around the quarterplane/disk intersection region. Let T'_k denote the tracking disk T_k translated and dilated to account for the motion of tracker and target, analogous to the computation of P'_k from P_k in Section 4.2. The same four cases (Figure 6) arise:

1. T'_k is fully inside $H(y_{k+1})$. No change is needed, so $T_{k+1} = T'_k$.
2. T'_k is fully outside $H(y_{k+1})$. This never occurs, because both T'_k and $H(y_{k+1})$ contain the system's true state x_k.
3. T'_k is partially inside $H(y_{k+1})$, crossing only one of the quarterplane axes. Here, the intersection region is simply between a disk and a halfplane. As in the privacy case, use a coordinate frame in which T'_k is centered at the origin, and the relevant halfplane boundary is a vertical line $x = b$. If $b \geq 0$, no disk smaller than T'_k will suffice, so $T_{k+1} = T'_k$. If $b < 0$, then choose for T_{k+1} the smallest enclosing disk, which has center $(b, 0)$ and radius $\sqrt{\text{radius}(T'_k)^2 - b^2}$.
4. T'_k is partially inside $H(y_{k+1})$, crossing both quarterplane axes. If the intersection region contains the quarterplane vertex, the smallest enclosing disk is the circumcirle of the quarterplane vertex and the two points on both the boundary of $H(y_{k+1})$ and the quarterplane axes. If the intersection region does not contain the quarterplane vertex, then the smallest enclosing disk is T'_k itself.

See Figure 8. Using these methods, the tracker can maintain P_k and T_k throughout its execution, giving a lower bound on the privacy margin and an upper bound on the tracking margin.

5 Active tracking

In this section we build on the passive update methods from Section 4 by introducing techniques to actively control the tracker to achieve goal conditions G1 (Section 5.1) and G2 (Section 5.2). In Section 5.3 we discuss an implementation of these algorithms and present some experimental results.

Throughout this section, we use a global coordinate frame, considering the motions of the tracker and target relative to an arbitrary but fixed external reference point. Since the observation received is determined by the tracker's position and orientation, we can view the active tracking problem as a problem of deciding

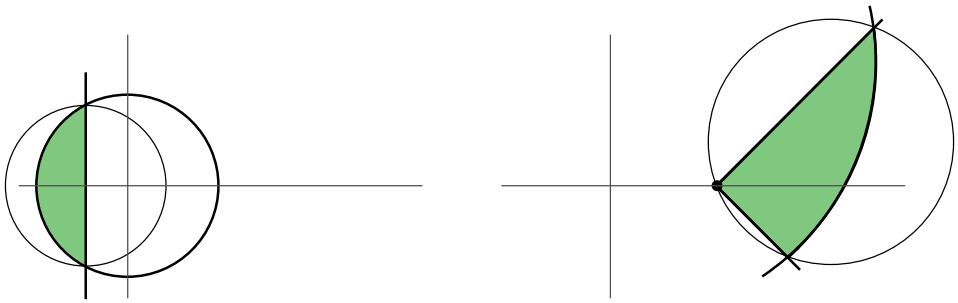

Fig. 8. The new tracking disk T_{k+1} is the smallest disk enclosing the shaded region. [left] Case 3. The center of T'_k is at the origin, and the relevant axis of $H(y_{k+1})$ is a vertical line $x = b$. [right] Case 4. The horizontal axis bisects $H(y_{k+1})$ and the vertical axis bisects P'_k.

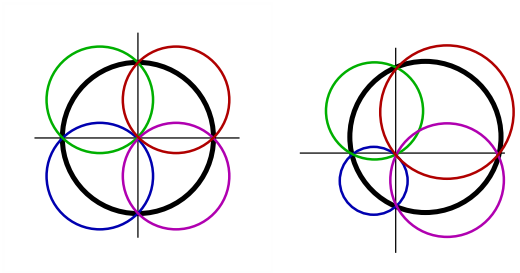

Fig. 9. [left] Positioning the tracker at the center of T_k minimizes the worst-case tracking margin at stage $k + 1$. [right] Other positions for the tracker lead to larger worst-case tracking margins. In this case, the tracker is positioned below and to the left of the center of T_k, and the worst case occurs if the target is in the top right quadrant.

where, and in what orientation, to place the tracker, thereby indirectly positioning the boundaries of the observation preimages. Throughout this section, we assume that the tracker has access at each stage to the privacy disk P_k and the tracking disk T_k, computed as described in Section 4.2. The strategies we derive are expressed by prescribing the destination of the tracker, relative to P_k and T_k. Note, however, that if the tracker has sufficient computation power, it can instead use the explicit passive update method described in Section 4.1, and replace P_k and T_k with the smallest enclosing and largest enclosed disks of η_k. In either case, we retain from Section 4 the convention that P'_k denotes a circle with the same center as P_k, with a radius larger by v_{tgt} (the translation is not needed in this coordinate frame); T'_k is defined similarly in terms of T_k.

5.1 Minimizing the tracking margin (G1)

Goal condition G1 requires the tracker to keep the tracking margin as small as possible, without regard for privacy. Thus, G1 refers to a typical target-tracking application. Our approach to achieving this goal is based on the following observation:

Lemma 1. *For a given T_k, the worst-case tracking margin for stage $k + 1$ is minimized when the tracker moves to the center of T_k.*

Proof. If the tracker positions itself at the center of T_k, the sensor preimage boundaries divide T_k into four parts that are identical up to a rotation. Regardless of the target's position, the resulting T_{k+1} will have the same radius. In contrast, if the tracker is not at the center of T_k, then for at least one of the four possible observations, the resulting T_{k+1} will be larger, thereby degrading the worst-case tracking margin. See Figure 9.

This observation leads directly to a strategy to achieve G1:

Strategy for G1 *Move to the center of T_k.*

What tracking performance does this strategy achieve? If the tracker executes this strategy, the tracking radii achieved can be expressed recursively:

$$\text{radius}(T_{k+1}) = \frac{(\text{radius}(T_k) + v_{tgt})}{\sqrt{2}}, \tag{7}$$

as described in Section 4.2. This sequence of worst-case tracking margins is monotone, and converges to $(1+\sqrt{2})v_{tgt}$. To execute this strategy, the tracker must be fast enough to move in a single stage between the centers of successive tracking disks T_k and T_{k+1}. Since center of T_k will lie on the boundary of T_{k+1}, this distance is equal to the radius of T_{k+1}. Therefore, $(1+\sqrt{2})v_{tgt}$ is the maximum useful speed for the tracker, in the sense that increasing the tracker's maximum speed beyond this value will not improve the tracking margins.

5.2 Maintaining a privacy bound (G2)

Next, we consider G2, in which the tracker wants to keep the tracking margin as small as possible, subject to the constraint that the privacy margin can become no smaller than a given minimum, denoted ρ. Our basic approach is to compute at each stage a "safe region" of destinations for the tracker, within which the privacy bound is maintained, while collecting some information. If this region is empty, the tracker instead chooses a position from which only one observation is possible, preventing any reduction of the privacy margin. The following lemma makes this idea more precise.

Lemma 2. *For given T_k and P_k, the privacy margin at stage $k+1$ will be at least ρ if, at the end of stage k, either*

(a) the radius of P'_k is at least $(1+\sqrt{2})\rho$, and the tracker is within distance $\sqrt{\text{radius}(P'_k)^2 - 2\,\text{radius}(P'_k)\rho} - \rho$, of the center of P_k, facing the center of P_k, or

(b) the tracker is farther than $\sqrt{2}\,\text{radius}(T'_k)$ from the center of T_k, oriented so that none of the observation preimage boundaries cross the interior of T'_k.

Proof. First consider condition (a). Notice that for a fixed distance from the center of P_k, the worst-case privacy is maximized when the tracker faces the center of P_k. Assuming the tracker takes this orientation, we consider (as in Section 4.2) a coordinate system in which the center of P_k lies on the vertical axis, and the horizontal axis bisects $H(y_{k+1})$. Let $(0, c)$ denote the center of P_k and let $(s, 0)$ denote the vertex of $H(y_{k+1})$. Note that because the tracker faces $(0, c)$, we have $c = s$. It remains to find the value of c such that the resulting largest enclosed disk has radius ρ. Let $(a, 0)$ denote the position of the center of the largest enclosed disk. The distance from this point to observation preimage boundary is $(a - c)/\sqrt{2}$; the distance to the boundary of P_k is $\text{radius}(P'_k) - \sqrt{c^2 + a^2}$. The radius of the largest enclosed disk is maximized when these values are equal, which occurs at $c = \sqrt{\text{radius}(P'_k)^2/2 - \text{radius}(P'_k)\rho} - \rho/\sqrt{2}$. Finally, for the tracker to generate this c (and, therefore, to maintain privacy margin ρ), it must have distance $\sqrt{2}c$ of the center of P_k. For condition (b), notice that both P'_k and T'_k will be fully contained within $H(y_{k+1})$. In this case, only one observation is possible, and neither P_{k+1} nor T_{k+1} will be affected the observation.

Lemma 2 provides two options for the tracker: To stay near the center of P_k (condition (a)), or to move far from the center T_k (condition (b)). Recalling the proof of Lemma 1, note that whenever the former option is available, it results in a smaller tracking margin in stage $k+1$. As a result, we can state the following motion strategy for the tracker:

Strategy for G2 *If $r > (1+\sqrt{2})\rho$, move to the point in the disk described in condition (a) of Lemma 2 closest to the center of T_k, facing the center of P_k. Otherwise, move distance $\sqrt{2}\,\text{radius}(T'_k)$ from the center of T_k, facing an angle of $\pi/4$ away from the line between the center of T_k and the tracker's destination.*[4]

[4] Under a different (and perhaps slightly more realistic) model in which the robot can choose whether or not to use its sensor at each stage, this second option can be replaced by simply choosing not to sense.

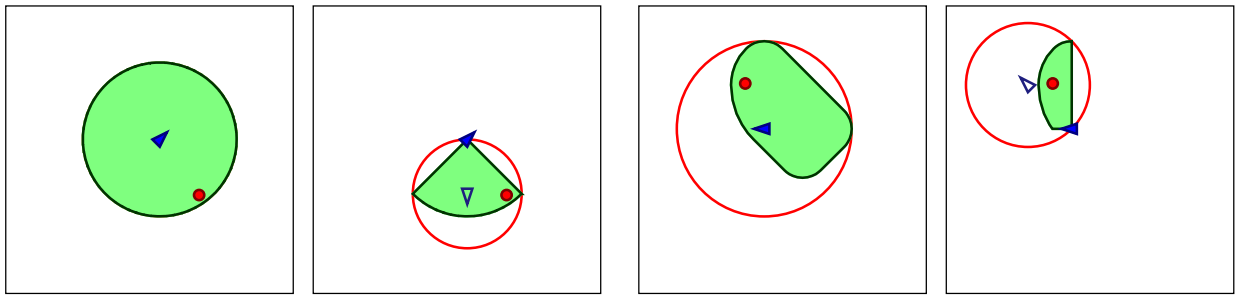

Fig. 10. Execution snapshots the Strategy for G1 described in Section 5.1. (a) The initial condition, in which the tracking disk and the true information state are identical. (b) After the first observation is received, the tracking disk encloses the information state (which the tracker does not compute). The tracker's destination is the center of this tracking disk. (c) Before sensing in stage 28 of this execution. The information state η_{28} and tracking disk T'_{28} are shown. (d) After sensing in stage 28. Notice that in this example, the tracking disk T_k is only a poor approximation of the smallest tracking disk.

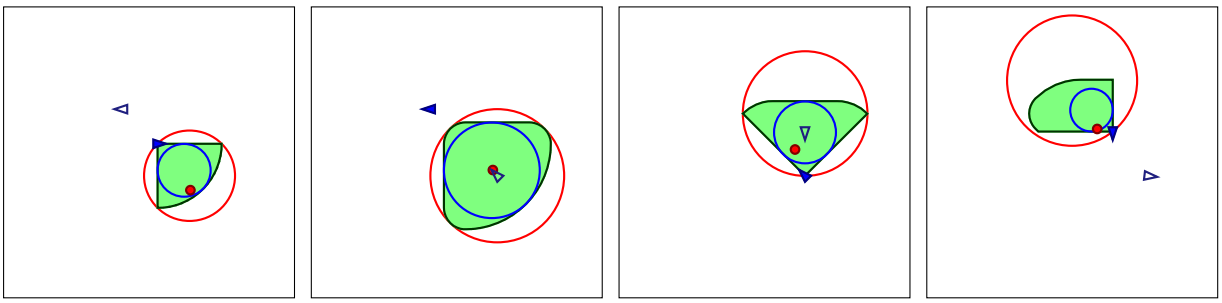

Fig. 11. Execution snapshots the Strategy for G2 described in Section 5.2. Shown left to right are the situations after sensing for $k = 1, 2, 3, 4$. Notice that the safe region is empty for $k = 1$ and $k = 4$.

In either case, the tracker's destination is straightforward to compute using circle-line intersection methods.

5.3 Simulation and experimental results

To evaluate our approach, we have implemented these algorithms in simulation. Figures 10 and 11 show several stages of example executions of G1 and G2, respectively. In these figures, the target's position is shown with a small circle. The tracker's current position and orientation are shown with a shaded triangle; the destination for the tracker is shown as an unshaded triangle. The large outer circle is the boundary of T_k, and the shaded region is the I-state η_k. Figure 11 also shows the boundary of P_k.

One natural question is to ask how much the tracking performance is degraded when the tracker's goal is G2 with nonzero ρ, compared to the "pure tracking" performance of G1.[5] That is, how much tracking accuracy must be sacrificed in order to guarantee a given level of privacy? To answer this question, we performed experiments with $v_{tgt} = 2.5$ and values of ρ varying between 0.1 and 2.5 in increments of 0.1. In each trial, the target moved through a sequence of randomly-selected destination points and we recorded the tracking radius achieved for each of these 25 values of ρ. Each trial lasted for 1,000 stages, and we averaged the results over 1,000 trials. Figure 12 summarizes the results of this experiment. The results, as one might expect, illustrate a tradeoff between tracking and privacy. When $\rho = 0.1$, the average results are similar

[5] Note, however, that G1 is *not* simply a special case of G2 in which $\rho = 0$. The former allows P_k to become an empty set, but the latter requires P_k to remain at least a disk with radius 0, that is, a single point.

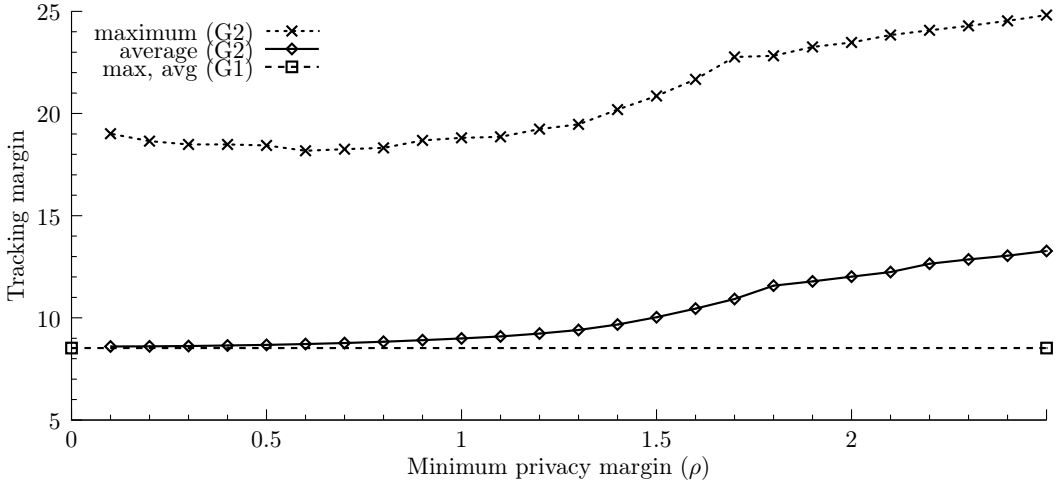

Fig. 12. Comparison of tracking radii for various tracker goals, including G1, and G2 with varying values of ρ.

to those for G1, and for larger values of ρ, the tracking radii increase roughly linearly. More substantial differences can be seen when comparing the maximum tracking radius that occurred in each trial.

We also performed experiments to assess the performance advantage gained by using the explicit updates of Section 4.1 compared to the constant time and space indirect updates of Section 4.2. We used $v_{tgt} = 1.5$, two different motion patterns for the target: a random walk and repeated circuits around a square of radius 10, and three motion strategies for the tracker: random movements, our strategy for G1, and our strategy for G2 with $\rho = 1.5$. For each of these six combinations, we performed 100 trials of 1,000 stages each, computed the ratios of tracking radii for explicit compared to implicit updates, and computed similar ratios for the privacy margins. For both tracking and privacy, we divided the smaller value by the larger one, so that a result of 1.0 would indicate optimal performance. Figure 13 shows the results. Privacy results are shown only for the cases in which the tracker's goal is G2; for the other two goals, P_k is quickly eliminated, leading to average ratios very close to 0.

Several interesting phenomena can be observed in the results. For tracking, implicit updates generate the best results when the tracker's goal is G1. This reflects the fact that, in general, the I-states achieved in this condition have the approximate shape of a right isosceles triangle, with the center of T_k on the hypotenuse. This leads to similar results for both implicit and explicit updates. Observe also that, across all tracker strategies, the approximation is superior for random motions of the target than for the square pattern. This reflects that fact that, whenever the target makes a long motion in a single direction, implicit updates repeatedly make the same kinds of over- or underestimates. The issue is especially visible for G2, in which the tracker is frequently "held back" by a need to stay within P_k.

6 Discussion and Conclusion

This paper presented an initial investigation of the role that a geometric view of privacy can play in robotic tracking problems. Unsurprisingly, many important questions have been left unanswered. This section presents some discussion and briefly reviews a few of these open problems.

6.1 Other goal conditions

It may be tempting to consider other goal conditions. Two possibilities that are somewhat analogous to the G1 and G2 that we have considered in detail are:

Tracking		Tracker motion		
		random	G1	G2
Target motion	random	0.596	0.838	0.619
	square	0.352	0.770	0.386

Privacy		Tracker motion
		G2
Target motion	random	0.789
	square	0.696

Fig. 13. Comparison of implicit versus explicit update methods for various motion strategies. In each table, a value of 1 would indicate that the performance of the implicit algorithm perfectly matches the explicit (optimal) algorithm. [top] Ratio of exact tracking margins for explicit updates to tracking margin upper bounds for implicit updates [bottom] Ratio of privacy margin lower bounds for implicit updates to exact privacy margins for explicit updates.

G3: *Maximize privacy margin* — Keep the privacy margin as large as possible, without regard for the tracking margin.

G4: *Maintain tracking bound* — Keep the privacy margin as large as possible, while ensuring that the tracking margin is no larger than a given maximum, denoted τ.

Note, however, that for G3, ideal results can be achieved simply by powering down the robot! For G4, an obvious reactive strategy would choose between a "run away" action (to allow P_k to grow unimpeded) whenever T_k is sufficiently small, and evenly divide P_k otherwise.

6.2 Sensor models

In this paper we considered only one particular kind of sensor for the tracker, one in which the preimages are quarterplanes. Alternatives we contemplated in the development of this work include sensors whose preimages are disks and their planar complements (that is, the plane with a disk deleted), halfplanes, and annuli. Each requires its own method for passively updating the tracker's information and its own unique active motion strategies. Note, however, that depending on the sensor models used, it may be challenging or impossible to design active strategies to satisfy G1 or G2. For example, if the observation preimages are a disk and its complement (such as would be the case if the sensor reported only whether the target was within a certain radius), then regardless of the tracker's strategy, the worst-case tracking margin would increase without bound, corresponding the case in which the target moves far away from the tracker throughout its execution. Similar difficulties occur if the observation preimages are halfplanes.

Recall also that the sensor models used in this work are deterministic, not allowing for sensor noise. The primary effect that sensor noise would have on our formulation is that the observation preimages would overlap, allowing the observation to be chosen in some unknown way whenever the state is in an overlap region. See Figure 14. This enlargement of the preimages can be expected to make privacy easier to maintain at the expense of increased difficulty in tracking. For the particular case of our quadrant sensor, the update algorithms would require only slight generalizations, and we expect the resulting active strategies to be quite similar as well.

6.3 Strategies that exploit explicit updates

The strategies introduced in Section 5 depend on having access to a privacy disk and a tracking disk at each stage, and can not directly make use of additional information to which the tracker may have access. One

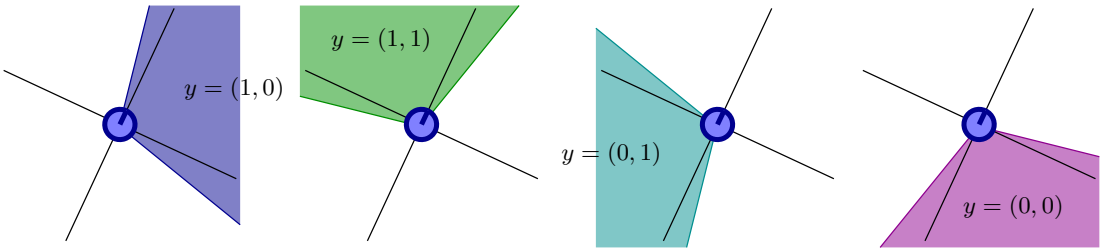

Fig. 14. Introducing sensor noise would allow the observation preimages to overlap.

obvious possibility is to use the I-state directly to choose actions, rather than using its smallest enclosing and largest enclosed disks. As an example, for G1 this approach would require the tracker to choose a position and orientation for the observation preimage boundaries so that the smallest enclosing disk of the resulting I-states is minimized.

6.4 Broader extensions

Several other extensions merit additional research, the most interesting of which include the presence of obstacles; related problems with multiple trackers, multiple targets, or both; and nontrivial mobility constraints.

Acknowledgment

I am grateful to Wenyuan Xu for introducing me to the panda tracker problem, and for helpful discussions. This work is partially supported by a grant from the University of South Carolina, Office of Research and Health Sciences Research Funding Program.

References

1. T. Bandyopadhyay, Y. Li, M. H. Ang, 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. F. Y. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. In *Proc. International Symposium on Algorithms and Computation*, pages 382–391, London, UK, 1995. Springer-Verlag.
3. M. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics and Automation*, 4(4):369–379, Aug. 1988.
4. L. Erickson, J. Knuth, J. M. O'Kane, and S. M. LaValle. Probabilistic localization with a blind robot. In *Proc. IEEE International Conference on Robotics and Automation*, 2008.
5. K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorthmica*, 10:201–225, 1993.
6. 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*, 9(5):471–494, 1999.
7. 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.
8. W. H. Huang and K. R. Beevers. Topological mapping with sensing-limited robots. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, pages 235–250, 2004.
9. P. Kamat, W. Xu, W. Trappe, and Y. Zhang. Temporal privacy in wireless sensor networks. In *Proc. IEEE International Conference on Distributed Computing Systems*, 2007.
10. P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *Proc. IEEE International Conference on Distributed Computing Systems*, 2005.
11. I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Transactions on Robotics and Automation*, 13(6):814–822, Dec. 1997.

12. I. Kamon, E. Rivlin, and E. Rimon. Range-sensor based navigation in three dimensions. In *Proc. IEEE International Conference on Robotics and Automation*, 1999.
13. S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.
14. 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.
15. A. Lazanas and J. C. Latombe. Landmark-based robot navigation. In *Proc. National Conference on Artificial Intelligence (AAAI)*, 1992.
16. V. J. Lumelsky and A. A. Stepanov. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorthmica*, 2:403–430, 1987.
17. V. J. Lumelsky and S. Tiwari. An algorithm for maze searching with azimuth input. In *Proc. IEEE International Conference on Robotics and Automation*, pages 111–116, 1994.
18. N. Meggido. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12:759–776, 1983.
19. 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.
20. 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.
21. J. M. O'Kane and S. M. LaValle. Localization with limited sensing. *IEEE Transactions on Robotics*, 23:704–716, Aug. 2007.
22. B. Tovar, R. Murrieta-Cid, and S. M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, 2007.