Target Tracking with Binary Proximity Sensors

NISHEETH SHRIVASTAVA
Bell-Labs Research, India
and
RAGHURAMAN MUDUMBAI, UPAMANYU MADHOW, and SUBHASH SURI
University of California at Santa Barbara

30

We explore fundamental performance limits of tracking a target in a two-dimensional field of binary proximity sensors, and design algorithms that attain those limits while providing minimal descriptions of the estimated target trajectory. Using geometric and probabilistic analysis of an idealized model, we prove that the achievable spatial resolution in localizing a target's trajectory is of the order of $\frac{1}{\rho R}$, where R is the sensing radius and ρ is the sensor density per unit area. We provide a geometric algorithm for computing an economical (in descriptive complexity) piecewise linear path that approximates the trajectory within this fundamental limit of accuracy. We employ analogies between binary sensing and sampling theory to contend that only a "lowpass" approximation of the trajectory is attainable, and explore the implications of this observation for estimating the target's velocity. We also consider nonideal sensing, employing particle filters to average over noisy sensor observations, and geometric geometric postprocessing of the particle filter output to provide an economical piecewise linear description of the trajectory. In addition to simulation results validating our approaches for both idealized and nonideal sensing, we report on lab-scale experiments using motes with acoustic sensors.

This work was supported by the National Science Foundation under grants ANI-0220118, CCF-0431205, CNS-0520335, and CCF 0514738; the Office of Naval Research under grant NOO014-06-0066; and the Institute for Collaborative Biotechnologies through grant DAAD19-03-D-0004 from the U.S. Army Research Office.
N. Shrivastava was a PhD candidate at University of California, Santa Barbara, when this research was conducted. R. Mudumbai is now affiliated with the University of Iowa.
A preliminary version of this article was presented at the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys'06).
Authors' addresses: N. Shrivastava, Bell-Labs Research, Bangalore, India; email: nisheeths@alcatel-lucent.com; R. Mudumbai, Electrical and Computer Engineering, University of Iowa, Seamans Center 4016, Iowa City, IA 52242; email: rmudumbai@engineering.uiowa.edu; U. Madhow, Department of Electrical & Computer Engineering, University of California, Santa Barbara, CA 93106; email: {raghu,madhow}@ece.ucsb.edu; S. Suri, Department of Computer Science, University of California, Santa Barbara, CA 93106; email: suri@cs.ucsb.edu.

DOI 10.1145/1614379.1614382 http://doi.acm.org/10.1145/1614379.1614382

Categories and Subject Descriptors: H.1.1 [**Models and Principles**]: Systems and Information Theory—*Information theory, Value of information*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Sensor networks, target tracking, binary sensing, fundamental limits, distributed algorithms

ACM Reference Format:

Shrivastava, N., Mudumbai, R., Madhow, U., and Suri, S. 2009. Target tracking with binary proximity sensors. ACM Trans. Sensor Netw., 5, 4, Article 30 (November 2009), 33 pages.
DOI = 10.1145/1614379.1614382 http://doi.acm.org/10.1145/1614379.1614382

1. INTRODUCTION

We investigate the problem of target tracking using a network of binary proximity sensors: each sensor outputs a 1 when the target of interest is within its sensing range, and 0 otherwise. This simple sensing model is of both fundamental and practical interest for several reasons. First, because of the minimal assumption about the sensing capability, it provides a simple and robust abstraction for a basic tracking architecture of broad applicability, which can be enhanced in a situation-specific fashion to take advantage of additional information such as target velocity or distance, if available. Second, the communication requirements for the binary proximity model are minimal—each sensor can smooth out its noisy observations and express its output as one or more disjoint intervals of time during which the target is in its range, which can be encoded efficiently by timestamps when the output changes from 0 to 1 and vice versa. Finally, the simplicity of the model permits the derivation of intuitively attractive performance limits, which serve both to guide design of tracking algorithms and to provide lower bounds on tracking performance with more sophisticated sensors.

We begin by exploring the fundamental limit of *spatial resolution* that can be achieved in tracking a target within a two-dimensional field of binary proximity sensors. The spatial resolution measures the accuracy with which a target's trajectory can be tracked, and it is defined as the worst-case deviation between the estimated and the actual paths. We prove that the ideal achievable resolution Δ is of the order of $\frac{1}{\rho R}$, where R is the sensing range of individual sensors and ρ is the sensor density per unit area. This result articulates the common intuition that, for a fixed sensing radius, the accuracy improves linearly with an increasing sensor density. But it also shows that, for a fixed number of sensors, the accuracy improves linearly with an increase in the sensing radius, which occurs because an increase in the sensing radius leads to a finer *geometric partition* of the field. Our spatial resolution theorem helps explain empirical observations reported in prior work on tracking in binary sensor networks [Kim et al. 2005].

Next, we consider minimal representations and velocity estimation for the target's trajectory. There are infinitely many candidate trajectories consistent with the sensor observations and within the guaranteed spatial resolution of the true trajectory, and all of which are "good enough" for localization accuracy. On the other hand, the velocity estimation for the target depends crucially on the shape of the trajectory. We use an analogy between binary sensing and the

sampling theory and quantization to argue that "high-frequency" variations in the target's trajectory are invisible to the sensor field at a spatial scale smaller than the resolution Δ. Therefore, we can only hope to estimate the shape or velocity for a "lowpass" version of the trajectory.[1] We then consider piecewise linear approximations to the trajectory that can be described economically. We give sufficient conditions for the lowpass version of the true target trajectory under which such minimal representations can estimate the velocity accurately.

Our results on velocity estimation can be paraphrased as follows: velocity estimates for a segment of the trajectory approximated by a straight line are good if the segment is long enough. This motivates an Occam's razor approach for describing the trajectories in terms of piecewise linear paths in which the line segments are as long as possible, without exceeding the approximation error limit provided by our spatial resolution theorem. For idealized sensing, we develop the OccamTrack algorithm for efficiently computing such piecewise linear trajectories, and associated velocity estimates, from the sensor observations. The efficacy of the algorithm in achieving the fundamental limits on spatial resolution and velocity estimation error is demonstrated via simulations.

Next, we consider more realistic sensor models, in which the coverage areas for different sensors may be different, and not exactly known to the tracker node. For such non-ideal sensors exhibiting sensing errors, it is necessary to average the sensor observations prior to applying geometric algorithms for obtaining minimal path descriptions. In particular, directly applying the OccamTrack algorithm to noisy observations can yield poor performance. We provide a simple model for nonideal sensing, and demonstrate that a particle filter is effective in smoothing the noisy observations. We then apply a geometric postprocessing algorithm on the particle filter output to extract an economical piecewise linear description of the estimated target trajectory, with the required goodness of fit guided by the fundamental limits on spatial resolution. Simulations are used to demonstrate the effectiveness of the overall algorithm in providing accurate tracking with minimal path representations.

Finally, we carried out a lab-scale demonstration with motes equipped with acoustic sensors for a quick validation of our framework. We found that the coverage area varies significantly across sensors, and exhibits nonmonotonicity: the probability that a target is detected does not necessarily go down monotonically with distance from the sensor. We employed two approaches to deal with real-world noisy data: (i) preprocessing of the noisy sensor outputs to clean up obvious error patterns, followed by the OccamTrack algorithm, and (ii) the Particle Filter algorithm followed by geometric post-processing. Both these approaches show good tracking performance.

1.1 Related Work

Object tracking has long been an active area of research for battlefield [Ketcham et al. 2005], robotics [Rao et al. 1993], and other applications. Any sensor that generates a signal dependent on distance from a target can be used for tracking. Accordingly different sensing modalities such as radar, acoustic,

[1]A technical definition of the lowpass trajectory is given in Section 3.5.1.

ultrasonic, magnetic, seismic, video, RF and infrared [Meesookho et al. 2002], and occasionally combinations of multiple modalities [Chellappa et al. 2004], have been considered for tracking applications in both theory and practice. The range and capabilities of these sensors vary widely, and many different approaches to modeling and data processing have been investigated. For instance, ultrasonic signals carry rich information about the range of the object, whereas infrared sensors are best modeled as detectors or binary sensors [Sabatini et al. 1995]. We do not attempt to do justice to the vast literature on tracking, but briefly review closely related work.

The robustness and effectiveness of tracking using binary sensing models has been convincingly demonstrated for a large-scale sensor network in Arora et al. [2004]. The success of this project provides strong motivation for the fundamental exploration of binary sensing undertaken here. The Kim et al. [2005] consider a model identical to ours. They employ piecewise linear path approximations computed using variants of a weighted centroid algorithm, and obtain good tracking performance if the trajectory is smooth enough. Our fundamental limits provide an explanation for some of the empirical observations in Kim et al. [2005], while our algorithms provide more accurate and more economical path descriptions. Another closely related paper is Aslam et al. [2003], which considers a different sensing model, where sensors provide information as to whether a target is moving towards or away from them. While the specific results for this model are quite different from ours, the philosophy is similar in that Aslam et al. [2003] use geometric analysis to characterize fundamental limits. However, the sensing model in Aslam et al. [2003] can lead to unacceptable ambiguities in the target's trajectory (the authors offer an example of parallel trajectories that are indistinguishable without additional proximity information). In contrast, the binary proximity model considered here, despite its minimalism, is able to localize the target to within a spatial resolution $O(\frac{1}{\rho R})$. More recently, our results on the localization accuracy of binary sensors have been extended to more general types of sensors including nonideal and noisy sensors [Mudumbai and Madhow 2008]. Liu et al. [2004], present some interesting ideas using geometric duality to track moving shadows in a network of binary sensors. Although their technique is not applicable to our problem setting, their notion of cells in dual space has some resemblance to our localization patches. In more recent work [Wang et al. 2008] the idea of localizing a target to a one-dimensional arc when it enters or leaves a sensor's coverage area has also been further developed.

Since binary sensors have a finite sensing region, one important question is the possibility of lack of coverage in some parts of the network. While we do not address the coverage problem in this paper, our statistical analysis of the localization error is based on spatial Poisson processes, similar to the methods used in the literature on area and path coverage processes [Ram et al. 2007; Hall 1988].

Classical tracking is often formulated as a Kalman filtering problem, using Gaussian models for sensor measurements and the target trajectory. Distributed tracking based on Kalman filtering has recently been considered [McErlean and Narayanan 2002]. Particle filters [Doucet et al. 2000] offer an

alternative to Kalman filters in non-Gaussian setting, and have been investigated for tracking using sensor networks [Coates 2004]. Most prior work on particle filtering assumes more sensed information (with a more detailed probabilistic model) than provided by the binary sensing model of this paper. Khan et al. [2003, 2005] have used particle filtering for an insect tracking application, where the insect targets are assumed to interact according to a Markov random field. Fox et al. [2001] provide a survey of particle-filter based methods for the problem of mobile robot localization, where robots wish to determine their location, and the locations of other robots, using sensory inputs. Our contribution in this paper is to provide a particularly simple particle filtering algorithm that provides robust performance using the minimal information obtained from nonideal binary sensors.

2. THE GEOMETRY OF BINARY SENSING

In this section, we describe an idealized model for a binary sensor network, and the structure of the geometric information it provides regarding a target's location. This geometric structure forms the basis for our theoretical bounds and algorithms.

Consider a network of n sensors in a two-dimensional plane. Each sensor detects an object within its *sensing region,* and generates one bit of information (1 for presence and 0 for absence) about the target; we call this the ideal *binary sensing model*. We get no other information about the location, speed, or other attributes of the target. The information of a sensor is efficiently encoded by the transitions between its 0 and 1 bits, and so its output can be summarized by the timestamps marking these transitions.

We assume that the sensors are continuously monitoring the target and that the sensors are synchronized with each other in time to sufficient accuracy (several timing synchronization algorithms are available in the literature [Elson et al. 2002]), so that the timestamps of the transitions are known accurately. We also assume that the location of each sensor is known—the sensor locations can be recorded at the time of deployment, or can be estimated using localization techniques [Savvides et al. 2001].

Our emphasis is on discovering and attaining fundamental limits of tracking performance, and therefore we abstract away lower layer networking issues by assuming that the sensor observations are communicated to, and fused at, a *tracker node.* Given the minimal communication needs of binary proximity sensors, such a centralized architecture may well be the most attractive choice for implementation in many settings, using multihop wireless communication between the sensors and the tracker node(s). In any case, it is relatively straightforward to develop communication- and storage-efficient hierarchical distributed versions of our centralized algorithms: for example, the tracker node can be chosen dynamically based on the target's location, and it can convey its summary of the particular segment of the target's trajectory to the next tier of the hierarchy.

For simplicity, we assume that each sensor has a circular sensing region of radius R: a sensor outputs a 1 if a target falls within the *sensing disk* of radius R

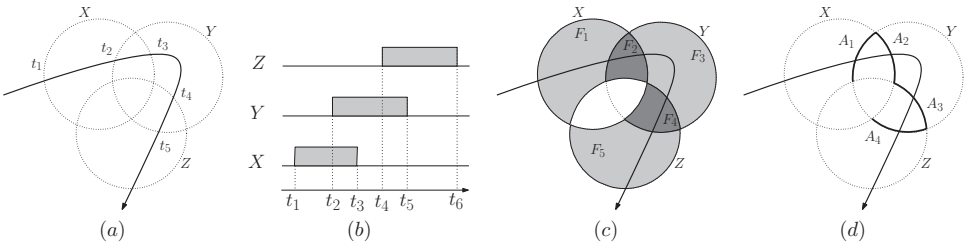


Fig. 1. A target moving through a field of three binary proximity sensors, X, Y and Z; (b) shows sensor outputs as a function of time; (c) shows the *localization patches* to which the target is localized over time intervals with constant signature; and (d) shows the *arcs* marking boundaries between patches.

centered at its location. The parameter R is termed the *sensing range.* However, our framework also applies to sensing regions of more complex shapes that could vary across sensors. We assume noiseless sensing for the time being: the sensor output is always 1 if a target is within its sensing range, and always 0 if there is no target within its sensing range, with 100% accuracy. Methods for handling noisy sensor readings are considered in later sections.

The geometry of binary sensing is best illustrated via an example. Figure 1(a) shows a target moving through an area covered by three sensors. Figure 1(b) shows the sensor outputs as a function of time. We define the *signature* of any point p in two-dimensional space as the n-bit vector of sensor readings, whose ith position represents the binary output of sensor i for a target at location p.[2] In Figure 1, if we define the signature as the bits output by sensors X, Y and Z in that order, then the target's signature evolves over time as follows: 000, 100, 110, 010, 011, 001, 000. Initially, it is outside the sensing disks of all three sensors; then it enters the disk of X, then Y, then it leaves the disk of X, enters that of Z, and so on. The time instants $\{t_j\}$ mark the transitions when the target either enters or leaves a sensor's range. Figure 1(c) shows that the target can be localized within a *localization patch* F_j during the time interval $[t_j, t_{j+1})$, which corresponds to the set of possible locations corresponding to the signature during this interval. When the target moves from a patch F_j to the next patch F_{j+1}, we note that exactly one sensor's bit changes: either the target enters the sensing disk of some sensor, or it leaves the disk of some sensor. The two patches, F_j and F_{j+1}, therefore, share a *localization arc* A_j of the disk of the sensor whose reading has flipped, as shown in Figure 1(d). A simple but important observation is that, at the transition times t_j, the two-dimensional uncertainty in the target's location is reduced to a one-dimensional uncertainty.

In general, a localization patch need not be connected and, correspondingly, the localization arc of two such patches can also have two or more pieces. (As a simple example, consider three sensing disks A, B, C, respectively, centered at points $(0, 0)$, $(1, 0)$, and $(2, 0)$, where the radius of the disks is 1.5. Then, the patch with signature $(0, 1, 0)$ has two disconnected pieces—these are the regions

[2]The notion of signature is a conceptual tool. Our algorithms do not actually use the entire bitmap for a given target location, but work with a much smaller localized version, as explained in the next section.

that are inside disk B but outside A and C.) Although disconnected patches are mainly an artifact of low sensor density, one can also create pathological examples where a patch can have two pieces even under high density. The nonconnectivity of patches, however, does not impact the tracking resolution, because our Theorem 2 ensures that even if a patch is disconnected, all of its pieces lie within the resolution bound of each other.

The preceding geometric information structure forms the basis for our results in subsequent sections. Our derivation of fundamental limits in Section 3 is based on estimation of the size of the regions F_j. The geometric algorithms for computing minimal description trajectory estimates consist of computing piecewise linear approximations that pass through the patches F_j or the arcs A_j in the order specified by the evolution of the target's signature.

3. FUNDAMENTAL LIMITS

We assume ideal sensing with sensing range R for each sensor, and an average sensor density of ρ sensors per unit area. Thus, the performance limits we derive depend only on the parameters ρ and R. We first show that the spatial resolution cannot be better than order of $\frac{1}{\rho R}$, regardless of the spatial distribution of the sensors. We then show that this resolution can be achieved using standard uniform random distributions as well as regular grids. Finally, we show that binary sensing is analogous to discrete sampling, in that it provides information only about a "lowpass" version of the target's trajectory, and discuss the implications for obtaining minimal path representations and velocity estimates.

3.1 An Upper Bound on Spatial Resolution

The *localization error* of an estimated trajectory is defined to be the maximum deviation of the estimated path from the actual path. This is just the L_∞ norm of the difference between the actual and estimated trajectories, viewed as functions of time. The *spatial resolution* of a binary sensor field is the worst-case localization error for any target trajectory through the field.

As observed in Section 2, binary sensing localizes a target to within a *patch* corresponding to a specific signature, or bit vector of sensor outputs. The spatial resolution is therefore given by the *diameter* of the largest patch induced by the binary sensing field. In the following, we argue that in *any* configuration of sensors, this diameter is *lower bounded* by $\frac{c}{\rho R}$, for an absolute constant c, which gives an *upper bound* on the achievable resolution.

THEOREM 1. *If a network of binary proximity sensors has average sensor density ρ and each sensor has sensing radius R, then the worst-case L_∞ error in localizing the target is at least $\Omega(1/\rho R)$.*

PROOF. We are interested in asymptotic behavior and so we assume that the sensor field is large relative to R, and we can ignore the boundary behavior by focusing on the portion of the field that is at least R away from the boundary. Since the average sensor density is ρ in the field, there must be a circular region of radius $2R$ that contains at most (the average number of) $N = \rho(4\pi R^2)$

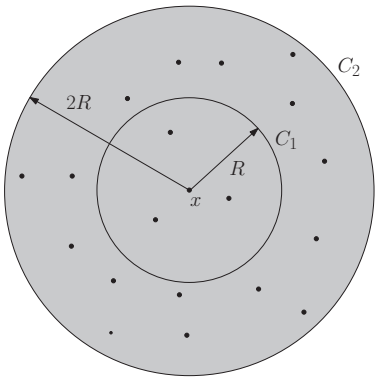


Fig. 2. Illustration for Theorem 1.

sensors in it. Let x be the center of this circle, let C_1 denote the circle of radius R centered at x, and let C_2 be the circle of radius $2R$ centered at x. (See Figure 2 for illustration.) We observe that only the sensors contained in C_2 can sense a target that lies in C_1. Since there are at most N such sensors, their sensing disks can partition the inner circle C_1 into at most $N^2 - N + 2$ "patches" [Agarwal and Sharir 2000][3]. On the other hand, the circle C_1 has area πR^2, so at least one of the patches must have area at least $c\pi R^2/N^2$, for some constant c. Plugging in the value of N, we get that some patch in C_1 must have area at least

$$\frac{c\pi R^2}{16\pi^2\rho^2R^4} \;=\; \Omega\left(\frac{1}{\rho^2R^2}\right).$$

Therefore, the *diameter* (the longest projection) of this patch is at least $\Omega(\frac{1}{\rho R})$, the square root of the area, which proves the claim. □

Theorem 1 makes no assumptions on the distribution of sensors: it only makes use of the average sensor density bound, and upper bounds the best resolution one can hope to achieve in an ideal deployment. In the next subsection, we address the complementary question: is this ideal resolution achievable, and what distributions of sensor nodes can realize this? Our investigation here is analytic, with a goal of showing that certain simple configurations of sensors lead to regions where the maximum L_∞ error matches the bound of Theorem 1. Algorithmic questions of computing compact trajectory approximations are addressed in the following section.

3.2 Achievability of Spatial Resolution Bound

The spatial resolution of Theorem 1 can be achieved (neglecting edge effects) by simply arranging the sensors in a regular grid. Since such an ideal deployment is often impossible in practice, we now show that a random Poisson distribution with density ρ also achieves the desired resolution. In the process, we also derive a sharp tail bound on the size of a localization patch.

[3]The result (without proof) can also be found at: http://mathworld.wolfram.com/PlaneDivisionbyCircles.html

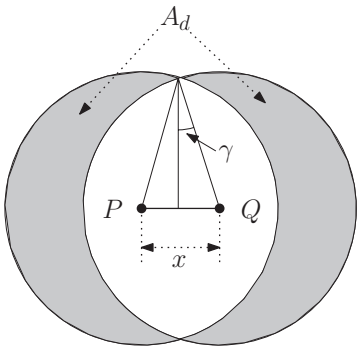


Fig. 3. Illustration for proof of Theorem 2.

Mathematically, the Poisson distribution of mean ρ means that (i) the number of sensors in a region of area A is a Poisson random variable N_A with mean ρA, and (ii) for two *nonoverlapping* regions, the corresponding numbers of sensors are independent random variables. We assume an asymptotic regime in which the probability of a point in the plane being within range of at least one sensor tends to one. For an arbitrary point P, this condition is satisfied if there is at least one sensor in a disk of radius R centered at P. Thus, $P[\text{no sensor in disk of radius } R] = e^{-\rho\pi R^2} \rightarrow 0$, which requires that $\rho R^2 \rightarrow \infty$. (In practice, values of ρR^2 of the order of 4 or more suffice to guarantee adequate coverage). The following theorem states our result.

THEOREM 2. *Consider a two-dimensional network of binary proximity sensors, distributed according to the Poisson distribution of density ρ, where each sensor has sensing radius R. Then the localization error at any point in the plane is of order $\frac{1}{\rho R}$.*

PROOF. See Figure 3 for an illustration. Consider an arbitrarily chosen point P in the plane, and an arbitrarily chosen direction of movement, starting from that point. Given the isotropic nature of the Poisson distribution, without loss of generality, this direction can be chosen as going right along the horizontal direction. Let X denote the minimum movement required in that direction before there is a change in signature (i.e., before the boundary of some sensor's disk is crossed). We wish to characterize the tail of the distribution of X.

To this end, consider a point Q that is a distance x away from P along the direction of movement, as shown in Figure 3. Any sensor detecting P (resp. Q) must lie in the disk of radius R with center at P (resp. Q). Thus, P and Q have the same signature if and only if the symmetric difference of these two disks (the shaded region in Figure 3) contains no sensor, assuming that either P or Q is detected by at least one sensor. (Under the assumption that ρR^2 is large, the last condition is met with high probability.)

Letting A_d and A_u, respectively, denote the area of the symmetric difference and the union of the two disks, it follows from the Poisson distribution that

$$P[X > x] = e^{-\rho A_d} \frac{1 - e^{-\rho(A_u - A_d)}}{1 - e^{-\rho A_u}}, \quad 0 \leq x \leq 2R.$$

(We note that $A_d \leq A_u$, with equality for $x \geq 2R$, so that $P[X > x] = 0$ for $x \geq 2R$. Thus, X is upper-bounded by $2R$.) Elementary geometric calculations yield that

$$A_d = \left(2\gamma + \sin 2\gamma\right) R^2,$$

where γ is the angle shown in Figure 3, satisfying $\sin\gamma = \frac{x}{2R}$. For our purpose, it suffices to loosely bound A_d below as

$$A_d \geq 2R^2 \sin\gamma = xR$$

(using $\gamma \geq \sin\gamma$). This implies that

$$P[X > x] \leq e^{-\rho A_d} \leq e^{-\rho R x}, \tag{1}$$

which guarantees the promised asymptotic decay with c, for $x = \frac{c}{\rho R}$. In fact, the exponent of decay is approximately twice as large as that used in our proof: this follows because the values of x, and γ, we are considering are small, and $A_d \sim 2xR$, which yields $P[X > x] \sim e^{-2\rho R x}$. □

This theorem can be interpreted in two ways.

(1) The *a*verage diameter of a localization patch under random deployment varies as $O(\frac{1}{\rho R})$.
(2) The probability that the size of a typical localization patch exceeds $\frac{c}{\rho R}$ decreases exponentially as $\exp(-c)$.

3.3 Remarks on Spatial Resolution Theorems

Theorems 1 and 2 show that the spatial resolution cannot be better than $O(\frac{1}{\rho R})$, and that this resolution can be achieved with a random (Poisson) sensor deployment. The dependence on sensor density seems to match common intuition: the more sensors we have, the better the spatial accuracy one should be able to achieve. On the other hand, the dependence on sensing radius may seem counterintuitive—because these are *binary proximity* sensors, they do not actually measure the distance to the target, and so having a large sensing radius may seem like a disadvantage. Indeed, as the sensing radius increases, we seem to get less information from an individual sensor: its 1 bit localizes the target to a larger area. Nevertheless, as our theorem shows, at the *system level*, the accuracy improves with larger sensing radius. This is a good example of the advantage of *networked* sensing, where the increase in an individual sensor's uncertainty is counter-balanced by a *quadratic* increase in the number of patches into which the sensor field is partitioned by the sensing disks. When the sensing radius is small, the sensing disks are either disjoint or overlap only a little, and there are only $O(n)$ patches. As the radius begins to grow, more disks pairwise intersect, and at sufficiently large radius, all pairs intersect, partitioning the sensor field into $\Theta(n^2)$ patches, thereby reducing the size of each patch and improving the localization accuracy. In a finite sensor field, of course, this improvement stops when the radius becomes comparable to the length of the field.

Our theorems also help explain some of the empirical results of Kim et al. [2005] for target tracking using binary proximity sensors. They found that for a fixed ρR^2 (which we can interpret as fixing the average number of sensors that can detect a target at a given position), better accuracy was achieved for the combination of "higher density and smaller radius" than "lower density and larger radius," leading them to propose that deployments with higher sensor density and smaller sensing radius are preferable. This empirical observation is a direct *consequence* of our theoretical results: for constant ρR^2, reducing the sensing radius by 1/2 corresponds to a factor of 4 increase in the density, while reducing the density by 1/2 corresponds to $\sqrt{2}$ increase in the radius. The former combination yields a higher value of ρR, which implies better spatial resolution.

3.4 Spatial Resolution in *d* Dimensions

Our theorems for maximum achievable resolution generalize quite easily to d dimensions. The achievable resolution in d dimension is $\Theta(1/(\rho R^{d-1}))$. In this section, we give detailed proofs of the d-dimensional results, both in uniform and possion distributions.

THEOREM 3. *If a network of binary proximity sensors deployed in a d-dimensional space has average sensor density ρ and each sensor has sensing radius R, then the worst-case L_∞ error in localizing the target is at least $\Omega(1/(\rho R^{d-1}))$.*

PROOF. The proof follows an argument very similar that of the 2-dimensional upper bound (Theorem 1). Since the average sensor density is ρ in the field, there must be a d-dimensional spherical region C_2 of radius $2R$ that contains at most $N = \rho\pi(2R)^d$ sensors in it. Let x be the center of this sphere, let C_1 and C_2 denote the spheres of radius R and $2R$ centered at x (similar to circles in Figure 2, only in d-dimensions). Again, the N sensors can partition C_1 into at most $O(N^d)$ d-dimensional "patches" [Agarwal and Sharir 2000][4]. Since the volume of C_1 is πR^d, at least one of the patches must have volume at least $c\pi R^d/N^d$, for some constant c. Therefore, the *diameter* (the longest projection) of this patch is at least $\Omega((c\pi R^d/N^d)^{1/d})$, the d-th root of the area, which is $\Omega(R/N)$. Plugging in the value of N, we get the diameter as $\Omega(1/(\rho R^{d-1}))$. □

Similarly we can also generalize Theorem 2 to d dimentions.

THEOREM 4. *For binary proximity sensors deployed in a d-dimensional space ($d \in \{1, 2, 3\}$.), the localization error at any point is of order $\frac{1}{\rho R^{d-1}}$.*

PROOF. We extend the proof of Theorem 2 result to arbitrary d-dimensional space. We start with the observation that any point P is detected by sensors located within a d-dimensional sphere of radius R, and is not detected by any sensors outside this sphere. Once again we consider a point Q located at a distance x away from P in some arbitrarily chosen direction. Similar to Figure 3,

[4]The result (without proof) can also be found at: http://mathworld.wolfram.com/SpaceDivisionbySpheres.html

P and Q lie in different localization regions if and only if there is at least one sensor in the region A_d, which in d-dimensional space is a *spherical* annulus.

For 1-dimensional deployment, the annulus A_d is simply two line segments of total length $A_d = 2x$. For 3-dimensions, the volume of the annulus can be shown to be

$$A_d = \frac{2\pi}{3} R^3 \sin\gamma\left(2 + \cos^2\gamma\right) \approx \pi R^2 x, \text{ if } x \ll R \tag{2}$$

Thus, in general, we have $A_d = cR^{d-1}x$, where c is independent of R and x. Thus we have, analogous to (1)

$$P[X > x] \leq e^{-\rho A_d} \approx e^{-\rho R^{d-1} x} \tag{3}$$

which shows that the size of the localization region varies as $\frac{1}{\rho R^{d-1}}$. □

3.5 Sampling and Velocity Estimation

The geometric information structure introduced in Section 2 shows that binary sensors can only localize the target to localization patches, and the resolution theorems of Section 3 show that these patches attain localization accuracy of $\Delta = O(\frac{1}{\rho R})$. Thus, as far as spatial accuracy is concerned, nothing further remains to be said. For any sequence of patch boundaries crossed by the target, there are infinitely many candidate trajectories crossing those patches in the same order, and *any one* of which is as good as another because they all lie within the achievable localization accuracy. Clearly, however, all these paths are not equally attractive as an estimate of trajectory. On grounds of "representational frugality," perhaps one would prefer a path that uses a small number of segments as opposed to the one that uses a large number of segments. A different criterion may be to choose paths that track the second important quantity of interest in target tracking: its *velocity*. It turns out that these two topics (path representation and velocity estimation) are in fact closely related, and are the focus of this section.

Our starting point is an analogy between binary sensing and analog-to-digital conversion based on sampling and quantization, which immediately suggests that only a "lowpass" version of the trajectory can be reproduced. Consider, for instance, the trajectory shown in Figure 4, which corresponds to the same sensor outputs as the trajectory of Figure 1 but includes "high-frequency" variations around a slowly varying trend. Within the spatial resolution afforded by our sensor model, these two trajectories are indistinguishable. The high-frequency trajectory of Figure 4, however, clearly has a higher velocity than the smooth trajectory of Figure 1. But, as we note below, the high-frequency component of its velocity cannot be estimated based on binary sensor readings. This suggests that, among many spatially equivalent paths, piecewise linear approximations adequately represent the output of the sensor field in terms of both spatial resolution and velocity estimation. An analysis of velocity estimation errors using such piecewise linear representations leads to the intuitively pleasing conclusion that paths that use few segments (frugal representation) are also the paths that lead to good velocity estimation! These ideas lay the

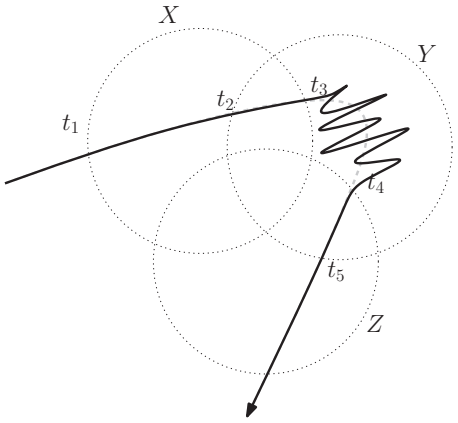


Fig. 4. A trajectory exhibiting high frequency variations that cannot be captured by binary sensors.

foundation for our algorithms (described in Section 4) that employ an Occam's razor approach to the construction of estimated trajectories.

3.5.1 *Lowpass Trajectories.* We begin with a simple but important interpretation of a binary sensor field as a device for spatial sampling. Let $\mathbf{x}(t)$ denote the two-dimensional vector specifying the true location of the target at time t. Using the notation of Section 2, we can say that $\mathbf{x}(t_j) \in A_j$, where $\{t_j\}$ are the times at which the target's signature changes, and A_j is the arc defining the boundary between the patches F_j and F_{j+1}. For a moment, assume that a *genie* or an *oracle* actually tells us the precise locations $\mathbf{x}(t_j)$, for the set of time instants $\{t_j\}$. We can now infer the following about the velocity vector $\vec{v}(t) = d\mathbf{x}/dt$:

$$\int_{t_j}^{t_{j+1}} \vec{v}(t)\, dt \;=\; \mathbf{x}(t_{j+1}) - \mathbf{x}(t_j).$$

In other words, even with the genie's aid, all that we can say about the target's trajectory during the interval $[t_j, t_{j+1})$ is that (i) the target is confined to the patch F_j, and (ii) the *average* vector velocity of the target in the patch is

$$\vec{v}_j = \frac{\mathbf{x}(t_{j+1}) - \mathbf{x}(t_j)}{t_{j+1} - t_j}$$

We denote the corresponding scalar average velocity by $v_j = ||\vec{v}_j||$.

Note that we cannot infer anything about the deviation $\vec{v}(t) - \vec{v}_j$ in the vector velocity from its average over the path, since this deviation integrates to zero in the time interval $[t_j, t_{j+1})$. This means that any high-frequency fluctuations in the path that are of small enough amplitude to stay within the patch F_j are entirely "invisible" to the binary sensor field.

Indeed, for a one-dimensional field of sensors, the sampling and quantization interpretation is immediate, without requiring invocation of a genie: the patches reduce to intervals and the arcs reduce to points. In this case, the binary sensor field is identical to a level-crossing analog-to-digital converter [Sayiner et al. 1996].

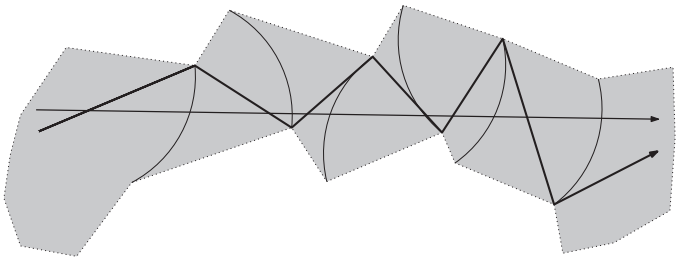


Fig. 5. Two paths in REP (the zig-zag and the straight line) can differ in length by at most a factor of 2.

Therefore, at best we can hope to reconstruct a *lowpass representation* of the target's trajectory, which we *define* as a piecewise linear approximation over spatial scale Δ, with line segments connecting the sequence of points $\mathbf{x}(t_1), \mathbf{x}(t_2), \ldots$. Other definitions that interpolate more smoothly across the arcs A_j are also possible, but the piecewise linear form has the virtue of being a minimal representation of the information obtained from the binary sensors and the genie (in particular, it preserves information in the average velocity sequence $\{\vec{v}_j\}$).

The trajectory shape and the velocity estimates for the lowpass representation serve as a benchmark for comparing the output of any algorithm based on the sensor readings. Since this benchmark is defined with the genie's help (which eliminates the spatial uncertainty at each arc A_j), it is not attainable in practice without some additional assumptions regarding the trajectory, as discussed in the next section.

3.5.2 *Velocity Estimation Error.* The set of all piecewise linear paths that visit the sequence of arcs A_j in the order given by the sensor signature sequence forms an equivalence class under the spatial resolution: all these paths are equivalent to the lowpass trajectory defined by the genie within the spatial resolution Δ. Let us call this set REP, for spatial *Resolution Equivalence Paths* class. Even considering the lowpass representation, where all fluctuations of spatial scale smaller than Δ are removed, two paths in REP can differ in length by a factor of 2: in a triangle of side length Δ, there are two possible paths, one of length Δ that follows one side, and one of length 2Δ following the other two sides. (See Figure 5.) More generally, one path can be a straight line, and the other can zig-zag taking 2Δ long detours for each segment of length Δ covered along the straight line.

In the absence of any other information, we simply have no way to decide which among the many candidate paths in the equivalence class REP offers the best approximation to the true path. The only way to decrease this uncertainty is to *assume* additional conditions that help shrink the spread of the path lengths in the equivalence class. In the following, we identify simple and natural *technical conditions* under which all the paths in the equivalence class have roughly the same length, and therefore any choice is guaranteed to give a good approximation. In particular, just as the accuracy of spatial resolution is controlled by size of the localization patches, the accuracy of the velocity

estimation is controlled by the *variance in the path lengths of the equivalence class*.

We consider minimal representations of the trajectory in terms of piecewise linear approximations with line segments spanning several patches, and ask when velocity estimations computed using such a representation are accurate. That is, we seek conditions under which the entire class of equivalent paths provides a good approximation to the genie-aided average scalar velocity function, which is a piecewise constant sequence taking value v_j over the time interval $[t_j, t_{j+1})$.

We first relate the relative error in velocity estimation to the relative spread in path lengths in the equivalence class REP. Suppose that the estimated trajectory is of length L between arcs A_k and A_{k+m}. Assuming that the scalar velocity is constant over this path segment, it can be estimated as the length divided by the time to go between arcs A_k and A_{k+m}: $v = L/(t_{k+m} - t_k)$. Suppose the true trajectory between A_k and A_{k+m} has length $L + \delta L$. Then, our velocity estimate error is $\delta v = \delta L/(t_{k+m} - t_k)$. We therefore obtain that

$$\frac{\delta v}{v} = \frac{\delta L}{L} \tag{4}$$

That is, by considering the relative variation $\frac{\delta v}{v}$ in velocity rather than the absolute variation δv, we are able to remove dependence on time scaling. The results we derive depend, therefore, only on the spatial variations of the path shape and its velocity, and not on time scale.

The main consequence of Equation (4) is that, if L is large enough and the permissible variation δL (constrained both by the sensor readings and the assumptions we make about the true trajectory) is small enough, then we can obtain accurate velocity estimates. For example, in order for a velocity estimate to be accurate to within 10%, we need to be able to guarantee that $\delta L \leq 0.1L$. If we assume that the scalar velocity is constant over large enough path sections, then we may be able to accurately estimate the velocity *as long as the variations in path lengths consistent with the sensor readings can be controlled.* Controlling the path length fluctuations is the same as bounding the path length spread in our equivalence class REP.

The following theorem characterizes the intrinsic ambiguity (caused by the spatial resolution Δ) in velocity estimation based on straight line approximations, arguing the the relative spread in path lengths is small if the line segment is long enough.

THEOREM 5. *Suppose a portion of the trajectory is approximated by a straight line segment of length L to within spatial resolution Δ. Then, the maximum variation in the velocity estimate due to the choice of different candidate straight line approximations is at most*

$$\frac{\delta v}{v} \leq 2\left(\frac{\Delta}{L}\right)^2.$$

Furthermore, this also bounds the relative velocity error if the true trajectory is well approximated as a straight line over the segment under consideration.

PROOF. By our spatial resolution theorem, the true trajectory is (approximately) a straight line that must lie within Δ of the straight line approximation s we are considering. That is, it must lie in a rectangle of width 2Δ with s as its long axis. The maximum deviation in length from the approximation s is if the true trajectory is the diagonal of this rectangle, whose length is $L+\delta L = 2\sqrt{(L/2)^2+\Delta^2}$, which yields $\delta L \approx 2\frac{\Delta^2}{L}$ for $\Delta \ll L$. Clearly, this bound also applies for deviation of the true trajectory from the straight line approximation being considered, as long as the true trajectory is well approximated as a straight line over the current segment. We now apply Equation (4) to obtain the desired result. □

Theorem 5 implies that, if we want to control the relative velocity error to less than ε using a piecewise linear approximation, then the length of each line segment must be at least

$$L \geq L_0 = \frac{\sqrt{2}\Delta}{\sqrt{\epsilon}}. \tag{5}$$

As an example, to achieve error at most 10%, segments of length 5Δ suffice; error of 5% requires segments of length $\approx 6.32\Delta$. Put another way, if on average each linear approximation segment spans α localization arcs, then the average relative velocity error is $dv/v \leq 2/\alpha^2$. Our simulation results show that even for fairly complex (synthetic) trajectories, a piecewise linear approximation works well, with α at least 10 on average.

Note that, for trajectories that "wiggle" while staying within Δ of a (long enough) straight line, Theorem 5 can be interpreted as guaranteeing accuracy in estimation of the *projection* of the velocity along the straight line. On the other hand, if a trajectory *curves* sharply, piecewise linear approximations to within Δ of the trajectory must necessarily use shorter line segments, making the velocity estimation error worse. But this is unavoidable because over short spans, the relative difference between two linear segments is larger, as implied by Theorem 5: in the extreme case, where $L \approx \Delta$, we are back to the factor of two error discussed at the beginning of Section 3.5.2.

4. TRACKING ALGORITHMS

The theoretical considerations of the previous section motivate an Occam's razor approach to tracking. Among all the candidate paths meeting the spatial resolution bound, a piecewise linear approximation that uses a minimal number of segments has the advantage of compact representation as well as accurate velocity estimation. Using the notation of Section 2, we know that the target is constrained to lie in region F_j during the time interval $[t_j, t_{j+1}]$, where $\{t_j\}$ are the time instants at which there are changes in the bit vector of sensor outputs. We formally define a localization region F_j, corresponding to an interval $[t_j, t_{j+1})$, as follows, dropping the subscript j for convenience. Let I be the subset of sensors whose binary output is 1 during the relevant interval, and let Z be the remaining sensors whose binary output is 0 during this interval.

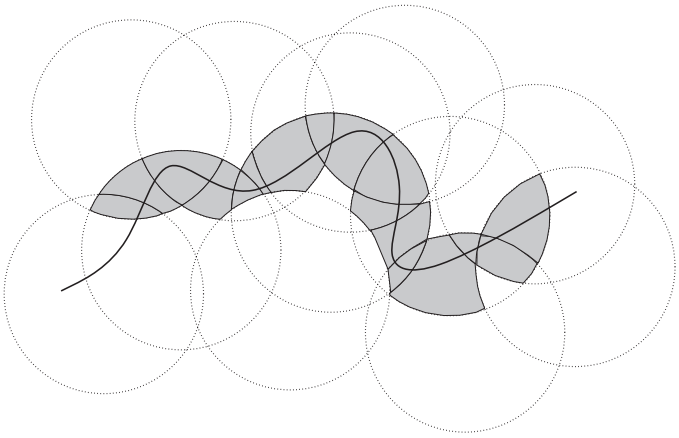


Fig. 6. The shaded band shows the regions $\{F_i\}$ to which the trajectory is localized by the sensor outputs.

Then the region F of the plane to which the target can be localized during this interval is given as:

$$F = \bigcap_{i \in I} D_i - \bigcup_{i \in Z} D_i,$$

where D_i is the sensing disk of radius R centered at sensor i. Note that it is not necessary to consider the entire set Z in order to determine F: it suffices to consider only those sensors whose disks can intersect with any disk in I. Thus, in our implementation, it is necessary only to maintain, for each sensor s, a *neighbor list* of all other sensors whose sensing disks intersect with the disk of s.

Figure 6 shows an example trajectory and the band consisting of the regions. *Any* trajectory that traverses these regions in the order specified by the sensing outputs is consistent with the target's true trajectory, within the accuracy bounds of the model. Among all these possible trajectories, the Occam's razor approach prefers the one that is the simplest. For instance, if all the regions could be traversed by a single line, then a linear trajectory has the simplest descriptive complexity, within the theoretical accuracy of tracking. Generalizing this, a *piecewise linear* trajectory with the fewest number of linear segments that traverses all the sensed regions in order is the trajectory of minimal complexity. In the following section, we describe a geometric algorithm, OccamTrack, for computing such a trajectory. Our computational model assumes that a tracker node collects the output from the sensor nodes, and runs the algorithm to compute the trajectory. The algorithm, however, can also be implemented in a distributed fashion by exchanging data among the neighboring nodes.

4.1 The OccamTrack Algorithm

Algorithm 1 describes the OccamTrack at a pseudo-code level. S is the set of all the sensors, and the algorithm operates in discrete time steps T, which are simply the instants at which one of the sensor's binary state changes. At

Algorithm 1 OccamTrack(S)

1: $T \leftarrow \{s.start, s.end : \forall s \in S\}$;
2: sort(T);
3: **for all** $t \in T$ **do**
4: $I \leftarrow \{s : t \in [s.start, s.end]\}$;
5: $Z \leftarrow \{s : s \in I.nbrlist \wedge t \notin [s.start, s.end]\}$;
6: $F \leftarrow \bigcap_{i \in I} D_i - \bigcup_{i \in Z} D_i$;
7: $B \leftarrow B \cup F$;
8: **end for**
9: $L \leftarrow$ MinSegPath(B);

each of these discrete time steps t, the algorithm determines the sets I and Z, and computes the region F localizing the target. The *time-ordered* sequence of these regions F is the *spatial band* B that contains the target's trajectory. The function MinSegPath then computes a minimum piecewise linear path traversing the band.

In the pseudo-code for MinSegPath, the function FindArcs determines the ordered sequence of localization arcs corresponding to the localization band B.[5] The function FindLine either determines that a subsequence of arcs $q_i, q_{i+1}, \ldots, q_j$ cannot be "stabbed" (in the given order) by a single line, or finds such a stabbing line. The algorithm MinSegPath uses this function in a greedy fashion to find the longest prefix of arcs that can be approximated by a single line segment, removes those arcs, and then iterates on the remaining sequence. There are only a finite number of combinatorially distinct candidate lines one needs to test to decide if a sequence of arcs can be stabbed by a line. In particular, it suffices to test the lines formed by pairs of endpoints of arcs, or lines that are tangent to some arc.[6] Figure 7 shows the minimal description path for the example of Figure 6.

We now briefly describe how we estimate the velocity of the target using the piecewise linear path L generated by OccamTrack. For each segment $\ell \in L$, we know the time instances of its first and last arc intersections, and hence the velocity can be estimated as the length of ℓ divided by the time difference. If the length satisfies the Equation (5), we will get a good estimate. This will generate the velocity for each segment in the target path. Notice that if the path was traveling along roughly a straight line, but with highly variable speed, the estimated velocity would be simply an average of the velocity and would not capture the variations in velocity too well.

[5]While a localization arc can have multiple pieces in a pathological case, the union of all its subarcs is still within the resolution bound (cf. Theorem 2). Thus, for the purpose of minimal path representation, we can safely "interpolate" all the disconnected pieces of the arc and still remain within the tolerable error. However, to keep our implementation simple, we chose to ignore such pathological contingencies, and opted to simply ignore an arc if it were found to be disconnected.

[6]In computational geometry, several theoretically more efficient methods [Guibas et al. 1991] are known for these stabbing problems, but they are complicated to implement and involve significant overhead in data structures. We chose to implement our algorithm because it is simple, compact, works fast in practice.

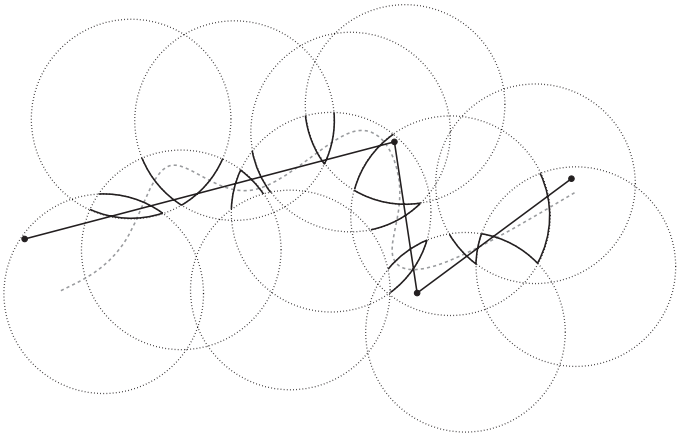


Fig. 7. The path computed by OccamTrack has 3 line segments. The sequence of arcs delineating the regions of band B are shown in thick lines.

If we are willing to sacrifice description minimality, we can generate better velocity estimates as follows. Consider the segments in the piecewise linear path generated by MinSegPath, and notice that each segment ℓ has a specific sequence of arcs $\{A_i, A_{i+1}, \ldots, A_j\}$, that it intersects. These intersection points, $\{p_i, p_{i+1}, \ldots, p_j\}$, and time instances of arc-crossing gives an estimate of location snapshots of the target in time. We can use this information and consider a window $\{p_i, p_{i+1}, \ldots p_k\}$, such that the length of segment in this window is greater than L_0 (see Equation (5)). We can now estimate the *local* velocity of this window using the length and the crossing times of p_i and p_k; this velocity estimate will have an error of at most ε. We can now *slide* the window of points by one and again repeat the same procedure. This will capture the variable velocity better, but would be more memory intensive since in the worst case we could be storing one estimate per arc.

4.2 Analysis of OccamTrack

By construction, the piecewise linear path computed by OccamTrack intersects the regions of the band B in the same order as given by the binary sensor's output. This follows because MinSegPath constrains the path to visit the boundary arcs of consecutive regions in order. This, however, does not mean that the true trajectory and the piecewise linear path visits the *same sequence* of regions. The linear shortcuts found by OccamTrack can visit additional regions. This can happen if the linear segment crosses over a *nonconvex* vertex of one of the regions. For example, in Figure 8, the piecewise linear path cuts the dashed arc and goes outside the shaded region, which is not visited by the original trajectory.

The important point to note, however, is that the maximum distance between the true trajectory and the computed path at any instant (the L_∞ error) is still bounded by $\Delta = O(1/\rho R)$, because the path computed by OccamTrack does lie entirely within the union of the *convex hulls* of the F regions in the band

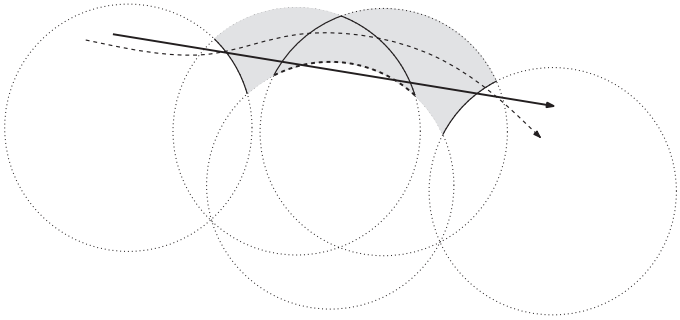


Fig. 8. Illustration to show that the reconstructed path may not visit the same sequence of arcs as the original trajectory.

B. Since the diameter of the convex hull of any F region is bounded by Δ, the error guarantee follows. Notice that in Figure 8, the piecewise linear path always stays within the convex hull of the patch with the dashed arc.

We can also prove that the MinSegPath finds a path that stabs the sequence of arcs that the target's track intersected and has at most twice the number of segments of the minimum description path. The reason for this 2-approximation is as follows. The MinSegPath greedily attempts to find the longest possible sequence of arcs that can be intersected by a single line segment. However, the segments generated by this method are *floating*, that is, they do not necessarily meet at the arcs. For example, if the output contains one line segment ℓ for arcs $\{A_1, A_2, \ldots, A_k\}$, and the next ℓ' for $A_{k+1}, A_{k+2}, \ldots, A_m\}$, then ℓ will end at A_k but ℓ' will start at A_{k+1}, leaving a gap between the two arcs. We extend ℓ and ℓ' to get a *continuous* piecewise linear path. Further, it may be that the extensions of ℓ and ℓ' do not intersect. Whenever this occurs, we use another segment to join them, thus requiring at most twice the minimum number of segments.

The aim of MinSegPath is to find the minimal description path that stabs the given sequence of arcs intersected by the trajectory; by construction it should also give a linear approximation to the trajectory with minimum number of segments. Unfortunately, this cannot be guaranteed because MinSegPath is forced to visit the arcs A_j that the target trajectory visits. On the other hand, it could be that the linear approximation with minimum number of segments of the trajectory does not even visit the same arcs. In fact, in the worst case one can imagine a situation that the target trajectory curves in such a way that it alternately touches the top and the bottom endpoints of the arcs A_j in the given sequence. Thus, any linear approximation of such a trajectory that is forced to visit the same arc sequence will require as many segments as the number of arcs.

However, one can still argue the following type of near-optimality: if a portion of the target trajectory can be approximated by a line segment with maximum L_∞ error Δ, then MinSegPath run as a postprocessing cleanup on the output of OccamTrack can approximate that part of the trajectory with a line segment with maximum error 2Δ. Thus, the piecewise linear path produced by OccamTrack, possibly with a second cleanup phase (similar to the geometric

Algorithm 2 MINSEGPATH(B)

1: $A \leftarrow$ FINDARCS(B);
2: $i \leftarrow 1$;
3: **for all** $j \in 1, 2, \ldots, m$ **do**
4: **if** $\neg$ FINDLINE($A_i, A_{i+1}, \ldots, A_j$) **then**
5: $L \leftarrow L \cup$ FINDLINE($A_i, A_{i+1}, \ldots, A_{j-1}$);
6: $i \leftarrow j$;
7: **end if**
8: **end for**

postprocessing presented in Section 4.3.2), can also estimate the velocity with the same order of error as given by the fundamental limit.

Computationally, the algorithm OCCAMTRACK is highly efficient because its time complexity depends only on the number of sensor boundaries that the trajectory crosses (that is, the number of arcs). Computing the sequence of arcs is relatively straightforward and can be done in constant time per arc; analysis of the MINSEGPATH algorithm is slightly more complex. As mentioned in previous section, to compute the line through a given sequence of m arcs, it suffices to test the lines formed by pairs of endpoints of arcs, $O(m^2)$ in total. For every line, we need to check if the line intersects the given sequence of arcs, which can be done in at most $O(m)$ time per line. Hence the total time complexity of MINSEGPATH is $O(m^3)$. The algorithm needs to store the candidate paths for the current sequence of arcs, hence the memory required is $O(m^2)$. Hence we get the following result.

THEOREM 6. *If there are m arcs in the sequence, then the worst-case time complexity of* OCCAMTRACK *is* $O(m^3)$ *and it uses* $O(m^2)$ *memory.*

Notice that Algorithm 2 is actually a simplified version, which does not remember the test results for previous candidate lines, and has to retest all pair of endpoints for the current sequence (in procedure FINDLINE) while inserting each new arc. This requires requires slightly more computation $O(m^4)$ but works with only $O(m)$ memory.

4.3 Robust Tracking with Nonideal Sensors

The OCCAMTRACK algorithm assumes ideal binary sensing. In practice, sensing is imperfect and noisy: a sensor could detect an object outside its nominal range, or it may fail to detect an object inside its range. We illustrate our approach to such non-idealities using a sensing model in which the target is always detected within an inner disk of radius R_i, called the *detection region*, and is detected with some nonzero probability in an annulus between the inner disk and an outer disk of radius R_o, called *uncertain region*. Targets outside the outer disk are never detected. This is similar to the "quasi unit disk graph" model proposed in Kuhn and Zollinger [2003] for ad-hoc networks, and is illustrated in Figure 9. Despite its simplicity, such a model is of fairly broad applicability, since it arises naturally if sensors integrate noisy samples over a reasonable time scale to make binary decisions regarding target presence or absence.

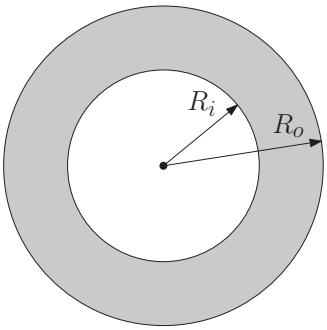


Fig. 9. The non-ideal sensing model.

The main implication of the model in Figure 9 for the OccamTrack algorithm is that we can no longer identify circular arcs corresponding to an object entering and leaving a sensor's detection range. While we can employ OccamTrack algorithm directly by approximating the sensing region as a disk of some radius R, where $R_i \leq R \leq R_o$, simulations show that the performance can be poor. We therefore consider an alternative approach, in which we employ a particle filtering algorithm to handle nonidealities. While this produces a good approximation of the true trajectory, it is not amenable to an economical description. We therefore employ a geometric post-processing algorithm to obtain a minimal representation for the output of the particle filtering algorithm. While particle filtering is a well established technique, the main novelty of the algorithm presented here is the way in which it exploits the constraints of the sensing model for a simple and efficient implementation.

In order to illustrate robustness to nonideal sensing, we take a worst-case approach to the information provided by the nonideal sensing model in Figure 9, assuming the maximal uncertainty consistent with the sensor readings. If a sensor output is 1, then we assume that the target is somewhere inside the large disk of radius R_o centered at the sensor. If a sensor output is 0, then we assume that the target is somewhere outside the small disk of radius R_i centered at the sensor. A localization patch F at any time instant is given by intersecting all such areas, just as before.

4.3.1 *Particle Filtering Algorithm.* We now sketch the particle filtering algorithm; a more detailed description and software implementation is available from the authors upon request. At any time n, we have K particles (or candidate trajectories), with the current location for the kth particle denoted by $x_k[n]$. At the next time instant $n+1$, suppose that the localization patch is F. Choose m candidates for $x_k[n+1]$ uniformly at random from F. We now have mK candidate trajectories. Pick the K particles with the best cost functions to get the set $\{x_k[n+1], k = 1, \ldots, K\}$, where the cost function is to be specified shortly. Repeat until the end of the time interval of interest. The final output is simply the particle (trajectory) with the best cost function.

Notice that the localization patch is actually: $F = \cap_{i \in I} D_i - \cup_{j \in Z} N_i$, where I, Z are the set of detecting and nondetecting sensors, D_i is the disk of radius

R_o around the detecting sensor i and N_j is the disk of radius R_i around the nondetecting sensor j.

Particle filters are a special case of the so-called *sequential Monte Carlo* methods, where the candidate paths ("particles") at each time instant n are created iteratively, from the candidate paths from the previous time instant $n-1$ along with the inputs at time n. Note that as we increase the number of particles K, we expect to get better results, however computational complexity limits the value of K in practice. The sampling time interval is chosen to be much shorter compared to the time difference between two localization patches, so as to generate a sufficiently rich set of candidates.

It remains to specify the cost function. We chose an additive cost function that penalizes changes in the vector velocity, in keeping with our restriction to lowpass trajectories. Once a candidate $x_k[n+1]$ is chosen from the current localization patch, the increment in position $x_k[n+1]-x_k[n]$ is an instantaneous estimate of the velocity vector at time n. The corresponding increment in the cost function is the norm squared of the difference between the velocity vector estimates at time n and $n-1$. This is given by

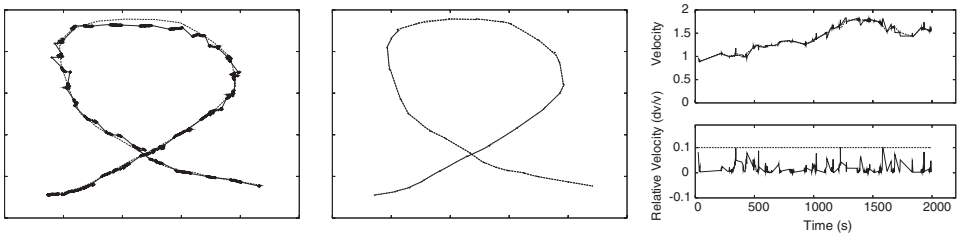
$$\begin{aligned} c_k[n] &= ||(x_k[n+1]-x_k[n])-(x_k[n]-x_k[n-1])||^2 \\ &= ||x_k[n+1]+x_k[n-1]-2x_k[n]||^2. \end{aligned}$$

The net cost function for a candidate trajectory up to time n is simply the sum of these incremental costs: $\sum_{m=1}^{n} c_k[n]$.

4.3.2 *Geometric Postprocessing.* The particle filtering algorithm described above gives a robust estimate of the trajectory consistent with the sensor observations, but it provides no guarantees of a "clean" or minimal description. This suggests the possibility of applying the geometric approach of Section 4.1 to the particle filter estimate to generate a more economical description. We now present a *geometric cleanup* algorithm, FitLine, that converts Particle Filter estimate into a piecewise linear approximation with a small number of line segments. Algorithm 3 above describes the FitLine at a pseudo-code level. It takes the ordered list of location samples p generated by Particle-Filter, and finds the piecewise linear path L. The algorithm proceeds in a greedy fashion: it tries to fit the longest sequence of points using a single line, until the error in that sequence is less than Δ. If adding the next point exceeds the desired error, it stores the last line and start afresh with the next point. The function LineSegment(Q) returns a line segment that is within distance Δ of the sequence

Algorithm 3 FitLine(p)

```
1: Q ← φ;
2: for all i ∈ 1, 2, ... , |p| do
3:    if Error(Q ∪ p_i) > Δ then
4:       L ← L ∪ LineSegment(Q);
5:       Q ← φ;
6:    end if
7:    Q ← Q ∪ p_i;
8: end for
```



(*a*) Weighted-Centroid (*b*) OccamTrack (*c*) Velocity estimate by OccamTrack

Fig. 10. Quality of trajectories produced by the weighted-centroid algorithm of Kim et al. [2005] and OccamTrack. Figure (c) shows the results of velocity estimation by OccamTrack.

of points Q, and Error$Q$ returns the values maximum distance of any point in Q with the best fitting line. We omit the details of procedures to compute the error (Error) and to find the best fitting line (LineSegment), the interested readers can find a detailed discussion in Agarwal et al. [2005].

5. SIMULATION RESULTS

We carried out extensive simulation tests to evaluate the performance of all our algorithms, under both ideal and nonideal sensing models. The code for OccamTrack was written in C and C++, the code for Particle-Filter was written in Matlab, and the experiments were performed on an AMD Athlon 1.8 GHz PC with 350 MB RAM. We first discuss our results for the ideal sensing model.

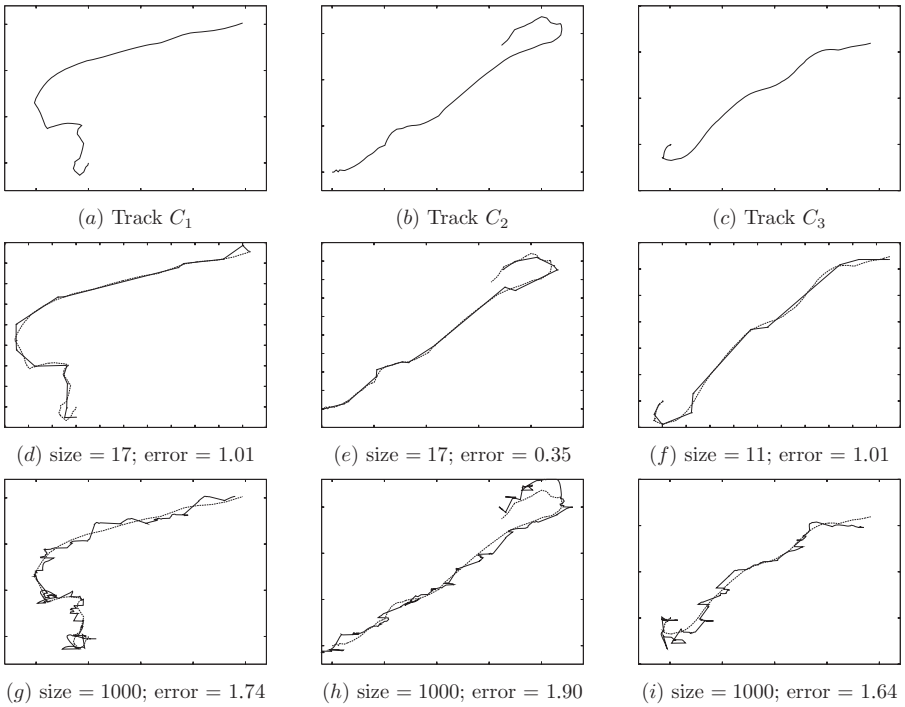
5.1 OccamTrack with Ideal Sensing

Our general experimental setup simulated a 1000×1000 unit field, containing 900 sensors in a regular 30×30 grid. The sensing range for each sensor was set to 100 units. When evaluating the scaling effects of the sensor parameters, we kept the field size and one parameter fixed, while the other parameter (radius or density) was varied.

We used *geometric random walks* to generate a variety of trajectories. Each walk consists of 10 to 50 steps, where each step chooses a random direction and walks in that direction for some length, before making the next turn. Each trajectory has the same total length, and we generated 50 such trajectories randomly.

5.1.1 *Quality of Trajectory Approximation.* On all 50 random walk trajectories, OccamTrack delivers excellent performance. Figure 10(b) is a typical example, where the true trajectory is virtually indistinguishable from the approximation computed by OccamTrack.

We also ran the weighted-centroid algorithm of Kim et al. [2005] on these trajectories. In our comparison, we used the *adaptive path-based* version of their algorithm, which is claimed to be well suited for complex and nonlinear trajectories. For ease of reference, however, we still refer to this algorithm as the *weighted-centroid* scheme. We ran this algorithm with inner and outer radii both equal to the ideal radius 100.



(a) Track C_1 (b) Track C_2 (c) Track C_3

(d) size = 17; error = 1.01 (e) size = 17; error = 0.35 (f) size = 11; error = 1.01

(g) size = 1000; error = 1.74 (h) size = 1000; error = 1.90 (i) size = 1000; error = 1.64

Fig. 11. Outputs of the tracking algorithms. Figure shows input trajectories (top row), path estimate by OccamTrack (middle row) and Weighted-Centroid algorithm (bottom row).

Figure 10(a) shows the output of the weighted-centroid method, and is typical of its performance on all our random walk trajectories. Figure 11 shows results of OccamTrack and weighted-centroid method on three sample trajectories. The weighted-centroid algorithm is sample based, and it used 1000 vertices to approximate each of the trajectories. By contrast, the OccamTrack used between 20 and 70 vertices. Despite this frugal representation, the maximum localization error for OccamTrack was *always* smaller than the weighted-centroid, on average by 30%, and in some cases by a *factor of five*. Due to its highly efficient structure, OccamTrack is also 300 times faster than weighted-centroid. In all cases, our algorithm took less than 10 milliseconds, while weighted-centroid took between 2 and 20 seconds.

5.1.2 *Velocity Estimation Performance.* In our random walk trajectories, we also varied the scalar velocity randomly at each turn, and then used OccamTrack to estimate the scalar velocity along the trajectory. For each linear segment in the piecewise linear path computed by OccamTrack, we used the first and the last localization arc to determine the time spent on that segment; recall that sensor outputs tell us the exact times for each arc. We estimate the scalar velocity for this segment by dividing the length of the segment by this time. With a goal of estimating the velocity within $\varepsilon = 0.1$, namely, 10%, we estimated the average velocity only over path segments of length at least $L = \sqrt{2}\Delta/\sqrt{\epsilon}$, as given by Equation (5).

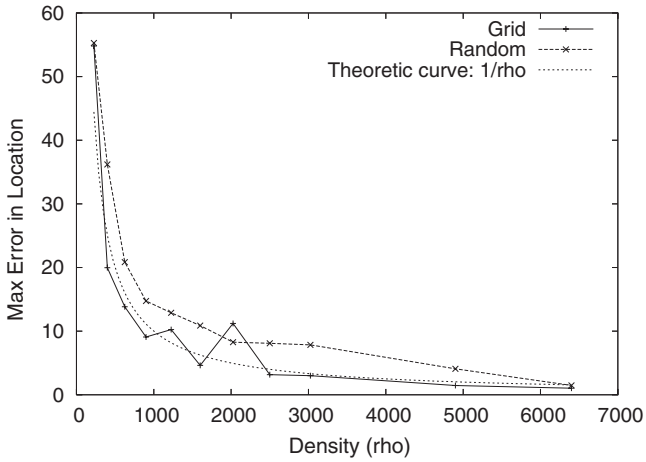


Fig. 12. Spatial resolution vs. sensor density.

In Figure 10(c), we show the results of estimating the velocity for the sample trajectory of (b). The top figure shows the overlay of both the true and the estimated velocities along the trajectory, and one can see that the two agree very well. In the bottom figure, we plot the relative error in the velocity to highlight deviation. The figure shows that the maximum deviation is always less than 10%, as predicted by theory.

The results were very similar for all 50 trajectories. In particular, on average a segment of OccamTrack's trajectory spanned about 15 patches, meaning that an average line segment in the approximation has length $L = 15\Delta$, meaning that the velocity estimates are good, as explained by Theorem 5.

In the following two experiments, we evaluated the localization accuracy of OccamTrack with varying ρ and R over many random trajectories, to see how it compares to the theoretical predictions of our theorems.

5.1.3 *Spatial Resolution as a Function of Density.* In this experiment, we measured the maximum error in localizing the target's trajectory for a varying values of the sensor density. We kept the size of the field and the sensing radius R fixed, and then varied the number of sensors in the field from $n = 100$ to $n = 6400$. (Since the area of the field is 10^6, this corresponds to variation in density from 10^{-4} to 6.4×10^{-3}.) We tried both the regular grid arrangement of the sensors, as well as the random placement.

By the spatial resolution theorem, the localization error should decrease inversely with the density. Figure 12 shows that the measured error follows closely the theoretical curve of $1/\rho$, both for the grid as well as the random placement. In each case, the reported error is the maximum error for the trajectory, averaged over 50 random walk trajectories. (The average error for each trajectory is much smaller.)

5.1.4 *Spatial Resolution as a Function of Sensing Range.* In this experiment, we kept the density constant at 900 nodes in the field, and varied the

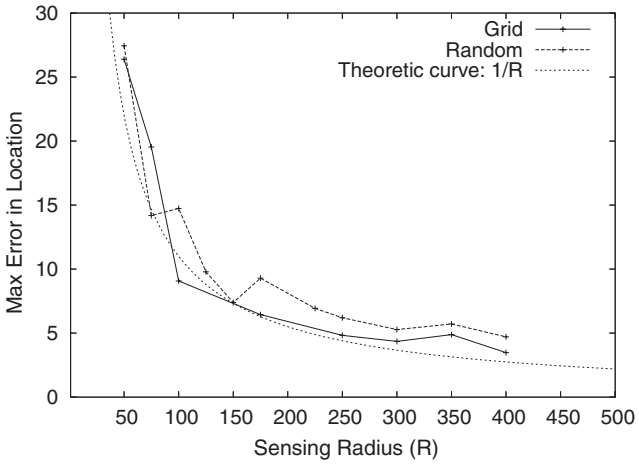


Fig. 13. Spatial resolution vs. sensing radius.

sensing radius from 50 to 400 units. Figure 13 shows the maximum error, averaged over 50 random walk trajectories, for various values of the sensing range. By the spatial resolution theorem, the localization error should decrease inversely with the sensing range, and again the measured values closely follow the theoretical curve of $1/R$.

5.1.5 *Velocity Accuracy as a Function of Density.* In this experiment, we measured the accuracy of velocity estimates of the target's trajectory for different values of the sensor density. We again kept the size of the field and the sensing radius R fixed, and varied the number of sensors in the field (hence density) from $n = 100$ to $n = 6400$. For both grid and random placement, the velocity error decreases inversely as the density is increased (see Figure 14). Since we kept both velocity of the trajectory (v) and the minimum segment length for which velocity is estimated (L_0) as constant, the velocity accuracy should vary roughly quadratically as the spatial resolution (Theorem 5). The measured values follow the curve of $1/\rho^2$, experimentally validating this relationship.

5.2 Tracking with Nonideal Sensing

We now describe the results of our experiments with non-ideal sensors. Our model of nonideal sensors is the one shown in Figure 9, where in the region between distance R_i and R_o, the target is detected with probability $\frac{1}{2}$. An imperfect detection is problematic for the ideal geometric algorithm OccamTrack because it relies on contiguous time intervals during which the target is inside the range. We used a simple hysteresis process to mitigate the affect of erratic detection: to signal the beginning of a detection interval, we require the sensor to output a 1 bit for 3 consecutive time samples; similarly, to signal the end of a detection interval, we require the sensor to output a 0 bit for 3 consecutive time samples.

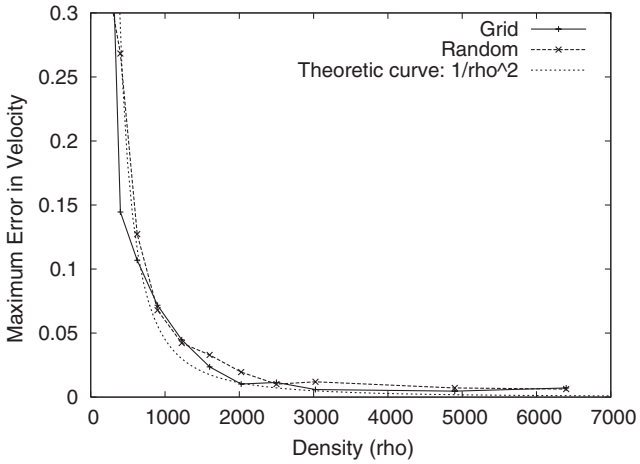
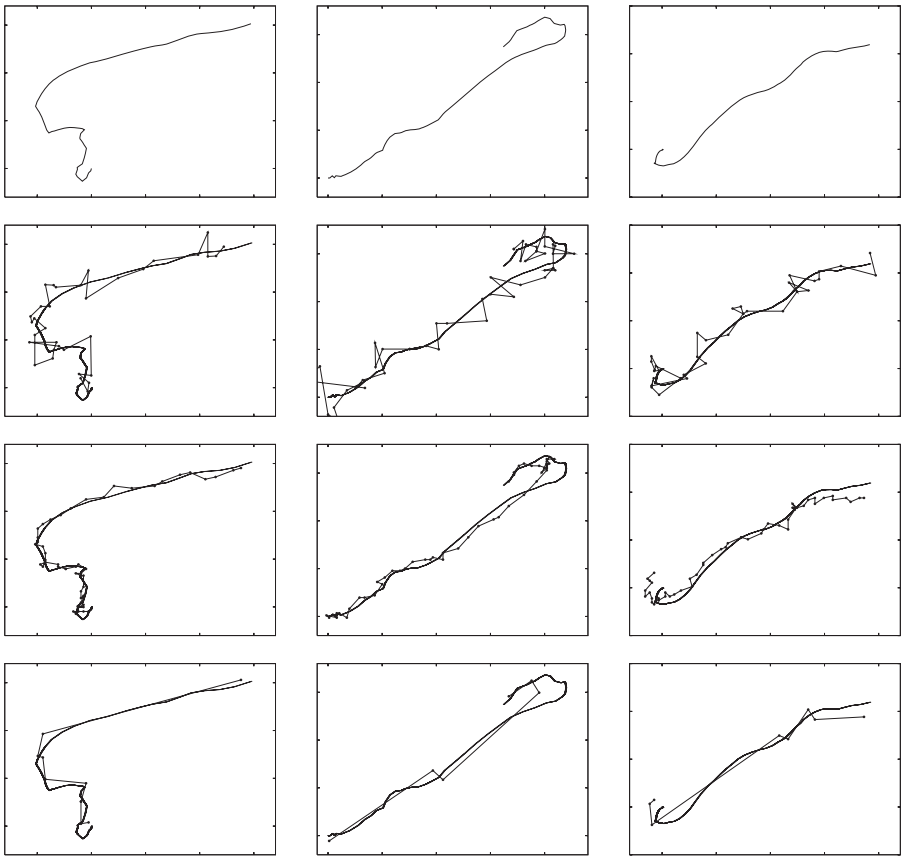


Fig. 14. Accuracy of velocity estimation vs. sensor density.

We generated a variety of geometric trajectories, simulated the sensor outputs using our non-ideal sensing model, and ran OccamTrack, Particle-Filter, and Particle-Filter with geometric post-processing, which we call Particle Filter + Geometric. A sample trajectory, along with the outputs of the three algorithms, is shown in Figure 15.

As expected, the ideal algorithm OccamTrack performs poorly when the data are imperfect: such data lead to gaps in the sequence of localization patches and infeasible localization arcs. In our implementation, we simply *ignored* these geometric inconsistencies, and just computed the piecewise linear paths using the rest of the arcs. Of course, in the worst case, poor data can completely break OccamTrack, but we found that the algorithm recovers rather well from these bad situations and produces acceptable trajectories, although not nearly as good as in the ideal case. In fact, compared to Particle-Filter, the output of OccamTrack looks significantly worse: it has significantly more pronounced turns and twists. Particle-Filter seems much better at dealing with noisy data, but its drawback is that, like any sample-b ased scheme, it produces trajectories with many vertices. This is where our combination of Particle-Filter with geometric postprocessing achieves the best of both worlds: it combines the robustness of Particle-Filter with the economic paths of the ideal OccamTrack.

Figure 15 shows the output trajectories generated by the three techniques, and Table I are the statistics of the output. As a sample result, for track C_1, Particle-Filter + Geometric uses 10 segments and has maximum error of 1.73, compared to 51 segments required for the Particle-Filter for the error of 1.19. We simulated this experiment over several trajectories, using the non-ideal sensing, and observed the same trend. On a typical input, the maximum localization error using Particle-Filter + Geometric was comparable to the basic particle filter algorithm, but in the worst-case it was almost 50% higher. On the other hand, the path description computed by Particle-Filter + Geometric was at least a *factor of* 5 smaller.



(*a*) Track C_1 (*b*) Track C_2 (*c*) Track C_3

Fig. 15. Outputs of the tracking algorithms on nonideal input. Figure shows input tracks (top row), OccamTrack (second row), Particle-Filter (third row), and Particle-Filter + Geometric (bottom row).

6. MOTE EXPERIMENTS

Finally, we set up a small lab-scale experiment using acoustic sensors to evaluate the performance of our algorithms. The setup consisted of 16 MICA2 motes arranged in a 4×4 grid with 30 centimeter separation, as shown in Figure 16. The motes were equipped with a MTS310 sensor board, which has an acoustic sensor and a tone detector circuit. (The tone detector can detect acoustic signals in a specific frequency range.) We adjusted the gain of the sound sensor so that the detection range for each sensor is about 45 cm. The target is also a MICA2 equipped with MIB310, which generated the acoustic signal using its on-board *beeper*. The target is then moved through the network in a path (shown as the dotted trajectory in Figure 17).

We first performed some experiments with a stationary target to determine the detection characteristics of the motes' tone-detector. The readings from the motes turned out to be highly non-ideal. Not only did the motes make frequent

Table I. Comparison of Target Tracking Algorithms

Track	PARTICLE-FILTER		PARTICLE-FILTER + Geometric		OCCAMTRACK	
	Size	Error	Size	Error	Size	Error
$C1$	51	1.19	10	1.73	33	2.36
$C2$	51	1.89	6	1.79	47	3.54
$C3$	51	1.73	8	1.49	34	1.67
$C4$	51	1.34	14	1.97	35	1.55
$C5$	51	1.00	6	1.71	35	1.77
$C6$	51	2.26	24	1.89	49	2.14
$C7$	51	1.30	22	2.13	45	2.17
$C8$	51	1.10	8	1.17	33	1.67

Fig. 16. The setup for our acoustic motes experiment.

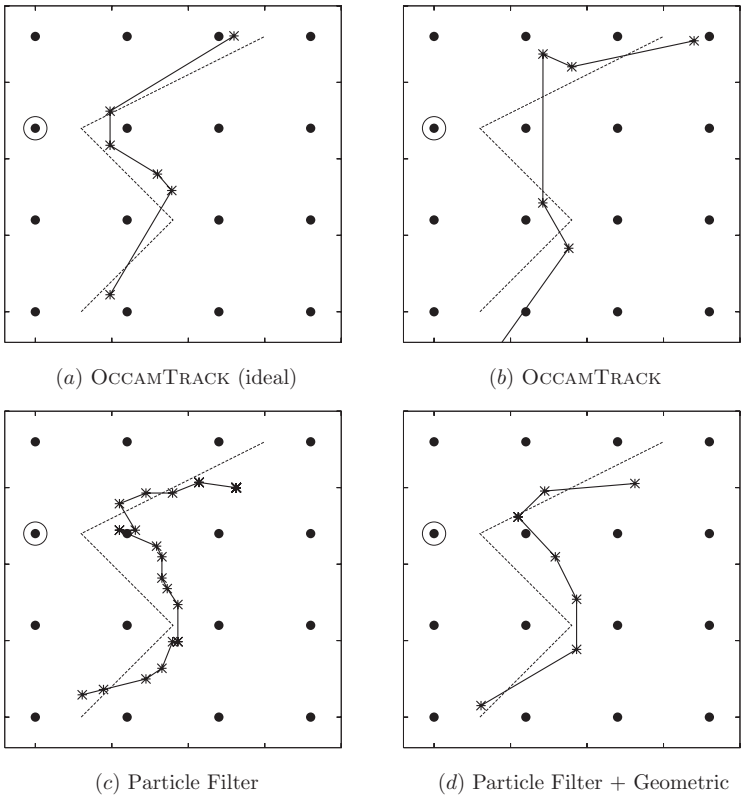
detection errors, but the probability of detecting a target was not a monotonic function of the distance from the sensor, as shown in Figure 18. While this detection behavior is difficult to model, it also means that this experiment is a good test for the robustness of our tracking algorithms.

The results of our experiment are shown in Figure 17. The detection readings we collection from these experiments showed a lot of non-ideal behavior. The most extreme being that one of the sensors, shown as double circle in the figure, failed to detect the target *entirely*, even though the target came very close to it.

On the whole, however, even in presence of such extreme failures, the results are very encouraging. All three algorithms were able to give a reasonable estimate of the target track. Figure 17(a) shows the reference output for OCCAMTRACK, assuming ideal sensing—that is, assuming the faulty sensor had also detected correctly, this is the trajectory OCCAMTRACK would produce. The other three figures show the outputs using the actual measurements from the acoustic sensors. (The actual path of the target is shown as a dotted line, while the estimated trajectories are shown in bold lines.) As expected, the trajectory quality of OCCAMTRACK suffers the most because of the failed sensor. The PARTICLE-FILTER does better, and again the combined algorithm preserves the robustness of PARTICLE-FILTER and approaches the minimal path description of OCCAMTRACK.

7. CLOSING REMARKS

We have identified the fundamental limits of tracking performance possible with binary proximity sensors, and have provided algorithms that approach



(*a*) OccamTrack (ideal) (*b*) OccamTrack

(*c*) Particle Filter (*d*) Particle Filter + Geometric

Fig. 17. The output trajectories for the experiment using acoustic sensors.

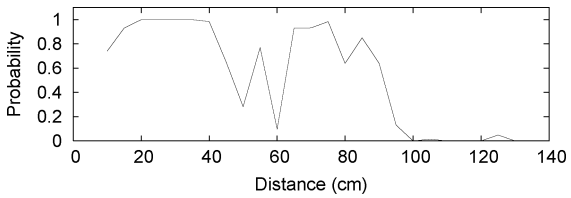


Fig. 18. Probability of target detection with distance for an acoustic sensor.

these limits. The results show that the binary proximity model, despite its minimalism, does indeed provide enough information to achieve respectable tracking accuracy, assuming that the product of the sensing radius and sensor density is large enough. Thus, this model can serve as a building block for broadly applicable and reusable tracking architectures.

The promising results obtained here and in Kim et al. [2005], as well as the success of the large-scale deployment in Arora et al. [2004], motivate a more intense investigation of tracking architectures based on the binary proximity model.

We would also like to develop minimal modifications of the basic tracking architecture to incorporate additional information (e.g., velocity, distance) if

available. The particle filtering framework appears to be a promising means for achieving this. Recent work indicates that a particle filtering approach can also be used to track multiple targets [Singh et al. 2007] using binary sensors. In addition to extensions and implementation optimization of this framework, an interesting question is whether it is possible to embed Occam's razor criteria in the particle filtering algorithm, rather than using geometric post-processing to obtain economical path descriptions.

Important issues for further investigation include the effect of uncertainty in sensor location, and the effect of imperfect time synchronization across sensors. For example, it would be interesting to explore whether it is possible to develop an iterative estimation framework for joint sensor localization, synchronization and target tracking.

ACKNOWLEDGMENTS

We are grateful to Kim et al. [2005] for giving us access to their target tracking code. We also appreciate the helpful comments provided by the anonymous reviewers.

REFERENCES

AGARWAL, P. AND SHARIR, M. 2000. Arrangements and their applications. In *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia Eds. North-Holland, Amsterdam, Netherlands, Chapter 2, 49–119.

AGARWAL, P. K., HAR-PELED, S., MUSTAFA, N. H., AND WANG, Y. 2005. Near-linear time approximation algorithms for curve simplification. *Algorithmica 42*, 203–219.

ARORA, A., DUTTA, P., BAPAT, S., KULATHUMANI, V., ZHANG, H., NAIK, V., MITTAL, V., CAO, H., DEMIRBAS, M., GOUDA, M., CHOI, Y., HERMAN, T., KULKARNI, S., ARUMUGAM, U., NESTERENKO, M., VORA, A., AND MIYASHITA, M. 2004. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Comput. Netw. 46,* 5, 605–634.

ASLAM, J., BUTLER, Z., CONSTANTIN, F., CRESPI, V., CYBENKO, G., AND RUS, D. 2003. Tracking a moving object with a binary sensor network. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*.

CHELLAPPA, R., QIAN, G., AND ZHENG, Q. 2004. Vehicle detection and tracking using acoustic and video sensors. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*. Vol. 3. iii–793–6.

COATES, M. 2004. Distributed particle filters for sensor networks. In *Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN)*. 99–107.

DOUCET, A., GODSILL, S., AND ANDRIEU, C. 2000. On sequential Monte Carlo sampling methods for bayesian filtering. *Stat. Comput. 10,* 3, 197–208.

ELSON, J., GIROD, L., AND ESTRIN, D. 2002. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev. 36*, 147–163.

FOX, D., THRUN, S., BURGARD, W., AND DELLAERT, F. 2001. Particle filters for mobile robot localization. In *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon Eds. Springer, Berlin.

GUIBAS, L., HERSHBERGER, J., MITCHELL, J., AND SNOEYINK, J. 1991. Approximating polygons and subdivisions with minimum link paths. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC)*.

HALL, P. 1988. *Introduction to the Theory of Coverage Processes*. John Wiley and Sons.

KETCHAM, S., MORAN, M., LACOMBE, J., GREENFIELD, R., AND ANDERSON, T. 2005. Seismic source model for moving vehicles. *IEEE Trans. on Geosci. Rem. Sens. 43,* 2, 248–256.

KHAN, Z., BALCH, T., AND DELLAERT, F. 2003. Efficient particle filter-based tracking of multiple interacting targets using an MRF-based motion model. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. 254–259.

KHAN, Z., BALCH, T., AND DELLAERT, F. 2005. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Trans. Patt. Anal. Mach. Intell. 27,* 11, 1805–1819.

KIM, W., MECHITOV, K., CHOI, J.-Y., AND HAM, S. 2005. On target tracking with binary proximity sensors. In *Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN)*.

KUHN, F. AND ZOLLINGER, A. 2003. Ad-hoc networks beyond unit disk graphs. In *Proceedings of the International Conference on Mobile Computing and Networking*.

LIU, J., CHEUNG, P., GUIBAS, L., AND ZHAO, F. 2004. Apply geometric duality to energy efficient non-local phenomenon awareness using sensor networks. *IEEE Wirel. Commun. Mag. 11*, 62–68.

MCERLEAN, D. AND NARAYANAN, S. 2002. Distributed detection and tracking in sensor networks. In *Proceedings of the 36th Asilomar Conference on Signals, Systems and Computers*. Vol. 2. 1174–1178.

MEESOOKHO, C., NARAYANAN, S., AND RAGHAVENDRA, C. S. 2002. Collaborative classification applications in sensor networks. In *Proceedings of the Workshop on Sensor Array and Multichannel Signal Processing*. 370–374.

MUDUMBAI, R. AND MADHOW, U. 2008. Information theoretic bounds for sensor network localization. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*.

RAM, S. S., MANJUNATH, D., IYER, S. K., AND YOGESHWARAN, D. 2007. On the path coverage properties of random sensor networks. *IEEE Trans. Mob. Comput. 6,* 5, 446–458.

RAO, B. S. Y., DURRANT-WHYTE, H. F., AND SHEEN, J. A. 1993. A fully decentralized multi-sensor system for tracking and surveillance. *Int. J. Rob. Res. 12,* 1, 20–44.

SABATINI, A. M., GENOVESE, V., GUGLIELMELLI, E., MANTUANO, A., RATTI, G., AND DARIO, P. 1995. A low-cost, composite sensor array combining ultrasonic and infrared proximity sensors. In *Proceedings of the International Conference on Intelligent Robots and Systems*. Vol. 3. 120–126.

SAVVIDES, A., HAN, C.-C., AND STRIVASTAVA, M. B. 2001. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)* . 166–179.

SAYINER, N., SORENSEN, H., AND VISWANATHAN, T. 1996. A level-crossing sampling scheme for a/d conversion. *IEEE Trans. Circ. Syst. 43,* 4, 335–339.

SINGH, J., MADHOW, U., KUMAR, R., SURI, S., AND CAGLEY, R. 2007. Tracking multiple targets using binary proximity sensors. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN'07)*. ACM, 529–538.

WANG, Z., BULUT, E., AND SZYMANSKI, B. K. 2008. A distributed cooperative target tracking with binary sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC) Workshops*, 306–310.

Received November 2007; revised July 2008; accepted November 2008