Tracking a Moving Object
with a Binary Sensor Network

Javed Aslam*, Zack Butler†, Florin Constantin†, Valentino Crespi‡, George Cybenko§ Daniela Rus†

ABSTRACT

In this paper we examine the role of very simple and noisy sensors for the tracking problem. We propose a binary sensor model, where each sensor's value is converted reliably to one bit of information only: whether the object is moving toward the sensor or away from the sensor. We show that a network of binary sensors has geometric properties that can be used to develop a solution for tracking with binary sensors and present resulting algorithms and simulation experiments. We develop a particle filtering style algorithm for target tracking using such minimalist sensors. We present an analysis of a fundamental tracking limitation under this sensor model, and show how this limitation can be overcome through the use of a single bit of proximity information at each sensor node. Our extensive simulations show low error that decreases with sensor density.

Categories and Subject Descriptors

ACM [**C.2.1**]: Network Architecture and Design

General Terms

Algorithms, Experimentation

Keywords

Sensor Networks, Tracking, Particle Filters, Minimalism

*College of Computer and Information Science, Northeastern University. This work partially supported by NSF Career award CCR-0093131. Portions of this work were completed while the author was on faculty at the Department of Computer Science, Dartmouth College.
†Department of Computer Science, Dartmouth College
‡Department of Computer Science, California State University Los Angeles. Part of this work was developed while the author was in service at Dartmouth College.
§Thayer School of Engineering, Dartmouth College

1. INTRODUCTION

Sensor networks are systems of many small and simple devices deployed over an area in an attempt to sense and monitor events of interest or track people or objects as they move through the area. In general, the sensors used (both the sensor itself as well as any associated computing) are very simple so that their cost remains low. Different sensing modalities including temperature, sound, light and seismic vibrations may be used in such a system depending on the targets of interest.

For several of these sensing modalities, the sensor may generate as little as one bit of information at each point in time. For example, if the sensors are obtaining sound levels, instead of using the absolute sound level (which may cause confusion between loud near objects and quieter close objects), the sensor may simply report whether the sound is getting louder or quieter. Similarly for the seismic sensor, an increase or decrease in intensity can be used. In these systems, using a single bit of information allows for inexpensive sensing as well as minimal communication. This minimalist approach to extracting information from sensor networks leads to a binary model of sensor networks.

In this paper we investigate the computational power of sensor networks in the context of a tracking application by taking a minimalist approach focused on binary sensors. The binary model assumption is that each sensor network node has sensors that can detect one bit of information and broadcast this bit to a base station. We examine the scenario in which the sensor's bit is whether an object is approaching it or moving away from it. We analyze this minimalist binary sensor network in the context of a tracking application and show that it is possible to derive analytical constraints on the movement of the object and derive a tracking algorithm. We also show that a binary sensor network in which sensors have only one bit of information (whether the object they sense is approaching or moving away) will give accurate predictions about the direction of motion of the object but do not have enough information content to identify the exact object location. For many applications predicting directional information is enough—for example in tracking a flock of birds, a school of fish, or a vehicle convoy. However, it is possible to pin down the exact location by adding a second binary sensor to each node in the net. If we include a proximity sensor that allows each node to report detecting the object in its immediate neighborhood we can determine the direction and location of the moving target.

This minimalist approach to sensor networks gives us in-

sight into the information content of the tracking application, because it gleans the important resources for solving this task. By studying minimalist sensor networks we learn that the binary sensor network model with one bit gives reliable direction information for tracking, but an additional bit provided by a proximity sensor is necessary to pin down exactly the object location. Minimalist approaches to understanding the information structure of tasks have been used previously, for example in the context of robotics tasks [6].

Our tracking algorithms have the flavor of particle filtering [1] and make three assumptions. First, the sensors across a region can sense the target approaching or moving away. The range of the sensors defines the size of this region which is where the active computation of the sensor network takes place (although the sensor network may extend over a larger area). The second assumption is that the bit of information from each sensor is available in a centralized repository for processing. This assumption can be addressed by using a simple broadcast protocol in which the nodes sensing the target send their id and data bit to a base station for processing. Because the data is a single bit (rather than a complex image taken by a camera) sending this information to the base station is feasible. Our proposed approach is most practical for applications where the target's velocity is slower than the data flow in the network, so that each bit can actually be used in predictions. However, since the accuracy of our trajectory computation depends on the number of data points, the predictions are not affected by the velocity of the target relative to the speed of communication. The third assumption is that an additional sensor that supplies proximity information as a single bit is available. Such a sensor may be implemented as an IR sensor with thresholding that depends on the desired proximity range, and can also be derived from the same basic sensing element that provides the original direction bit of information.

2. RELATED WORK

Target tracking is concerned with approximating the trajectory of one or more moving objects based on some partial information, usually provided by sensors. Target tracking is necessary in various domains such as computer vision [7], sensor networks [13], tactical battlefield surveillance, air traffic control, perimeter security and first response to emergencies . A typical example is the problem of finding the trajectory of a vehicle by bearings measurement, which is a technique used by radars. Work in robotics has also considered tracking targets from moving platforms [11].

Several methods for tracking have been proposed. This includes Kalman filter approaches or discretization approaches over the configuration space. A recent method that shows great promise is particle filtering, which is a technique introduced in the field of Monte Carlo simulations. The main idea of particle filtering is to discretize the probability distribution of the object's position rather than maintaining the entire feasible position space. This is achieved by keeping multiple copies (called "particles") of the object each of which has an associated weight. With every action (usually a sensor reading) a new set of particles is created from the current one and the weights are updated. Any function about the object is then obtained as the weighted sum of the function values at each particle. The seminal paper in this domain is [9], which states the basic algorithm

and properties. Since then many papers have addressed this topic; among the most important are the variance reduction scheme [8] and the auxiliary particle filter [12]. A survey of theoretical results concerning the convergence of particle filter methods can be found in [5].

Probabilistic methods have also been used in robotics for simultaneous localization and mapping (SLAM), in which the robot attempts to track itself using the sensed position of several landmarks. For example, in [10], particle filter techniques were used for localization only when the traditional Kalman filter technique had failed. These algorithms typically assume range and bearing information between the landmarks and tracked vehicle, unlike the very simple sensors considered here.

Sensor networks face two kinds of major problems. First, efficient networking and energy-saving techniques are required. The sensors have to communicate with one another or with a "base" to transmit readings or results of the local computation. In [3], increasingly complex activation schemes are considered in an attempt to substantially improve the network's energy use with little loss in tracking quality.

Second, we should be efficient in processing the information gathered by sensors. In [2], Brooks, Ramanathan and Sayeed propose a "location-centric" approach by dynamically dividing the sensor set into geographic cells run by a manager. In the case of multiple measurements, they compare the data fusion (combine the data and then take a single decision) versus the decision fusion (take many local decisions and then combine them) approaches.

A distributed protocol for target tracking in sensor networks is developed in [4]. This algorithm organizes sensors in clusters and uses 3 sensors in the cluster toward which the target is headed to sense the target. The target's next location is predicted using the last two actual locations of the target.

Our sensor model requires sending only one bit of information to a central computer and thus the issues shown above are not of capital importance. We rather focus on geometric properties of the sensors configuration and on an algorithm for solving this tracking problem.

We are inspired by this previous work and use the particle filtering approach in the context of the binary sensor model.

3. THE BINARY SENSOR NETWORK MODEL

In the binary sensor network model, each sensor node consists of sensors that can each supply one bit of information only. In this section we assume that the sensor nodes have only one binary sensor that can detect whether the object is approaching (we will call such a sensor a *plus sensor*) or moving away (we will call such a sensor a *minus sensor*). We assume that the sensor range is such that multiple sensors can detect this information and forward the bit to a base station. We call this the active region of the sensor network. Because the data is simple and consists of one bit only, this assumption can be met through a protocol in which the active sensors forward their id and data bit. The sensor may be noisy and use thresholding and hysteresis to detect movement and compute the direction bit. The active region of the sensor network may change over time, but since we assume that only the active sensors report data, the computations are done relative to those sensors only. We assume that the base station knows the location of each

sensor. Without loss of generality, we assume from now on that all the sensors can sense the object movement over the same space.

In this section we characterize the geometry of the plus sensors and minus sensors instantaneously first, and then over time, using history information. We then relate this characterization to constraints on the trajectory of the object they sense, which will lead to the tracking algorithm developed in the next section.

The tracking problem can be formulated as follows. Suppose a set of m binary sensors $S = \{S_1, S_2, \ldots, S_m\}$ are deployed within a bounded $2D$ area. Assume now that an object U is moving inside the area along a curve Γ and let $X(t)$ be one of its parametric representations. Finally, let the sensors sample the environment at regular intervals of time, thereby producing a sequence of binary m-vectors $s \in \{-1, 1\}^m$ (with $s_i^{(j)} = +1/-1$ meaning U is approaching/going away from sensor i at time t_j). Then we would like to provide an estimate of the trajectory X of U for the given placement of the sensors.

3.1 The Instantaneous Sensor Network Geometry

Consider a single sample $s \in \{-1, 1\}^m$ of data, produced at time t. We would like to determine sufficient and necessary conditions for the location X of the target and the direction of its movement $V = X'$.

The key result reported as Theorem 2 shows that the location of the tracked object is outside the convex hull of the plus sensors and also outside the convex hull of the minus sensors. We first show an important property of the plus and minus sensors relative to the instantaneous velocity and position of the object.

LEMMA 1. *Let i and j be two arbitrary sensors located at positions S_i and S_j and providing opposite information about U at time t. Without loss of generality, let $s_i^{(t)} = +1$ and $s_j^{(t)} = -1$ (object U is decreasing its distance from sensor i and increasing its distance from sensor j). Then it must be the case that*

$$S_j \cdot V(t) < X(t) \cdot V(t) < S_i \cdot V(t) ,$$

where $\cdot$ denotes the scalar product in $\mathbf{R}^2$.

Proof. Consider the situation as depicted in Fig. 1. Since U is going away from sensor S_j then it must be that $\alpha > \pi/2$. Analogously, since U is approaching sensor S_i, it must also be that $\beta < \pi/2$. These two conditions translate into

$$(S_j - X) \cdot d\mathbf{l} < 0 \quad \text{and} \quad (S_i - X) \cdot d\mathbf{l} > 0 ,$$

or in integral form

$$\int_\Gamma (S_j - X) \cdot d\mathbf{l} \quad \text{strictly decreasing,}$$

and

$$\int_\Gamma (S_i - X) \cdot d\mathbf{l} \quad \text{strictly increasing.}$$

Replacing $d\mathbf{l} = X'(\tau)d\tau$ our conditions become

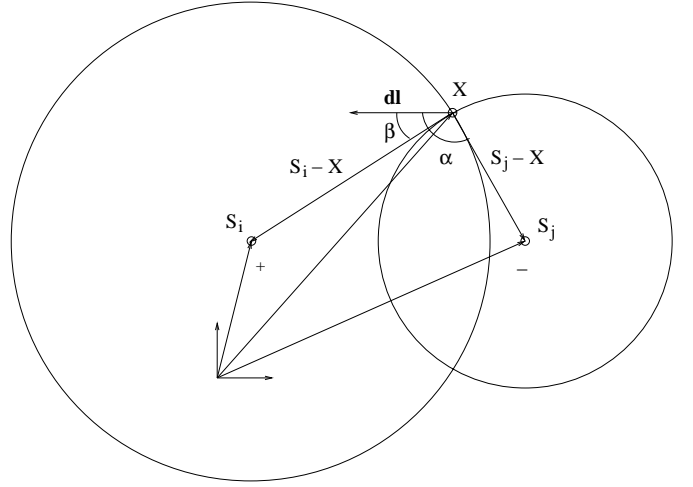$$\int_0^t (S_j - X(\tau)) \cdot X'(\tau)d\tau \quad \text{strictly decreasing,}$$

Figure 1: Necessary and sufficient conditions on X.

and

$$\int_0^t (S_i - X(\tau)) \cdot X'(\tau)d\tau \quad \text{strictly increasing.}$$

But this amounts to saying that

$$(S_i - X(t)) \cdot X'(t) > 0 \quad \text{and} \quad (S_j - X(t)) \cdot X'(t) < 0 ,$$

from which the claim follows. □

An immediate corollary of this lemma is the following condition for the feasibility of a pair (X, V):

$$\max_j \{S_j \cdot V \mid s_j = -1\} < X \cdot V < \min_i \{S_i \cdot V \mid s_i = +1\}.$$

This velocity constraint can be used to derive a useful sensor detection separation result that will result in further object trajectory constraints.

Figure 2 shows the intuition behind the constraints computed based on the sensor geometry. The current position of the object is between the convex hull of the plus sensors and the convex hull of the minus sensors and the object is heading toward the convex hull of the plus sensors. History information accumulated over time can be used to identify the direction and position of the object within this region.

Next we present the theoretical results limiting the feasible object-sensors configurations. Theorem 2 provides a coarse approximation of the location of the tracked object, namely that it has to be outside the minus sensors' and plus sensors' convex hulls.

THEOREM 2. *Let $s \in \{+1, -1\}^m$ be a sample of the sensor values at a time t. Let $A = \{S_i \mid s_i = +1\}$ and $B = \{S_j \mid s_j = -1\}$ and $C(A)$ and $C(B)$ their convex hulls. Then: $C(A) \cap C(B) = \emptyset$. Furthermore, $X(t) \notin C(A) \cup C(B)$.*

Proof. Assume by contradiction that the first part of the claim is false. Then $C(A) \cap C(B) \neq \emptyset$. This implies that there exists at least one sensor $u \in B$ whose position S_u falls inside $C(A)$. So S_u must be a convex combination of the vertices $\mathbf{a}_j$ of $C(A)$: $S_u = \sum_j \alpha_j \mathbf{a}_j$, with $\alpha_j \geq 0$, $\sum_j \alpha_j = 1$. Now, since $s_u = -1$, by Lemma 1 we must have:

$$\left(\sum_j \alpha_j \mathbf{a}_j \right) \cdot V(t) = \sum_j \alpha_j (\mathbf{a}_j \cdot V(t)) < X(t) \cdot V(t) .$$

On the other hand it must also be that

$$\sum_j \alpha_j \mathbf{a}_j \cdot V(t) \geq \sum_j \alpha_j \min_i \{\mathbf{a}_i \cdot V(t)\} \geq \mathbf{a}_{i_0} \cdot V(t) > X(t) \cdot V(t) \,,$$

which is contradictory. We denote $i_0 = \underset{i}{\operatorname{argmin}} \, \mathbf{a}_i \cdot V(t)$.

To show the second part of the claim, assume that $X(t) \in C(A)$. So, as before, $X(t)$ can be expressed as a convex combination of the vertices in $C(A)$: $X(t) = \sum_j \alpha_j \mathbf{a}_j$ and by Lemma 1 it must be

$$X(t) \cdot V(t) < \min_j \{\mathbf{a}_j \cdot V(t)\}$$

or by substituting the convex combination

$$\sum_j \alpha_j \mathbf{a}_j \cdot V(t) < \min_j \{\mathbf{a}_j \cdot V(t)\},$$

which is again contradictory. □

The approximation given by Theorem 2 can be further refined using the following result. Theorem 3 states that the plus and minus convex hulls are separated by the normal to the object's velocity.

THEOREM 3. *Let $s \in \{+1, -1\}^m$ be a sample of the sensors values at a certain time t. Let $A = \{S_i \mid s_i = +1\} \neq \emptyset$, $B = \{S_i \mid s_i = -1\} \neq \emptyset$ and $C(A)$, $C(B)$ their respective convex hulls. Then the normal $\vec{N}$ to the velocity separates $C(A)$ and $C(B)$ and V points to $C(A)$.*

Proof. We can suppose modulo a translation of the plane that the current location X of the object is $X = (0, 0)$. Let m be the slope of the velocity and let $\vec{V} = (v, m \cdot v)$ where $v \in \mathbf{R}$ and assume without loss of generality that $m \notin \{0, \infty\}$. Then the equation of the normal $\vec{N}$ is: $y = -\frac{1}{m} \cdot x$

Let $S^+ = (a^+, b^+)$ be an arbitrary "plus" sensor and $S^- = (a^-, b^-)$ an arbitrary "minus" sensor. Then we have to show that

$$\left(\frac{a^+}{m} + b^+\right) \cdot \left(\frac{a^-}{m} + b^-\right) < 0$$

i.e., any two opposite (i.e.,"plus" and "minus") sensors lie on different half-planes with respect to $\vec{N}$. What sensors report can be translated as $(S^+ - X) \cdot V > 0$ or $a^+ \cdot v + b^+ \cdot m \cdot v > 0$ and respectively $(S^- - X) \cdot V < 0$ or $a^- \cdot v + b^- \cdot m \cdot v < 0$. By multiplying these relations we get that

$$\left(a^- \cdot v + b^- \cdot m \cdot v\right) \cdot \left(a^+ \cdot v + b^+ \cdot m \cdot v\right) < 0$$

and, by factoring each parenthesis by $m \cdot v$,

$$m^2 \cdot v^2 \cdot \left(\frac{a^-}{m} + b^-\right) \cdot \left(\frac{a^+}{m} + b^+\right) < 0$$

and the claim follows. For the remaining part of the claim, note that V points to the "plus" convex hull if and only if $S_+ \cdot V > 0$ or $\left(a^+, b^+\right) \cdot (v, m \cdot v) > 0$ or further $a^+ \cdot v + b^+ \cdot m \cdot v > 0$, which is what the sensors read. □

In our model which assumes that sensors are not influenced by noise the only correct sensor reports have to respect the constraints in Theorem 2 and Theorem 3.

3.2 Linear Programming Perspective

In Section 3.1 we showed some instantaneous analytical properties of trajectories tracked with binary sensors. The proofs presented in that section are intuitive but not constructive. In this section we show how the tracking problem can be formulated constructively in an equivalent fashion using linear programming.

We wish to determine the current position of the tracked object (denoted by (x_0, y_0)) and the slope of the normal to its velocity (denoted by m_0), based on the locations of the plus and minus sensors. Unlike in classification theory, we wish here to characterize the entire feasible region not just one line (a separating hyperplane) in that region. We know that the line of slope m_0 passing through x_0 (i.e., the normal to velocity) separates the convex hulls of the "plus" and "minus" sensors. Moreover the velocity points toward the "plus" convex hull.
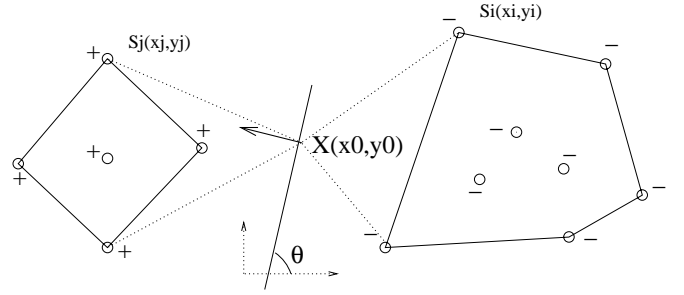

Figure 2: This figure shows the intuition behind the natural constraints on the velocity of the tracked object that are grounded in the convex hull separation result.

Let $S_i = (x_i, y_i)$ and $S_j = (x_j, y_j)$ be, respectively, sensors with information $-$, $+$. The constraints for the tracking problem can be written as:

- $-\infty < m_0 < 0$
   - $\diamond$ $y_i - y_0 \geq m_0 \cdot (x_i - x_0)$
   - $\diamond$ $y_j - y_0 \leq m_0 \cdot (x_j - x_0)$
- $m_0 = 0$
   - $\diamond$ $\max y_i \leq y_0 \leq \min y_j$
- $\infty > m_0 > 0$
   - $\diamond$ $y_i - y_0 \leq m_0 \cdot (x_i - x_0)$
   - $\diamond$ $y_j - y_0 \geq m_0 \cdot (x_j - x_0)$
- $m_0 = 0$
   - $\diamond$ $\max x_j \leq x_0 \leq \min x_i$

The above inequalities can be translated into linear inequalities by introducing a new variable $\mu_0 = m_0 \cdot x_0$. If m_0 (the slope) is given then these cases can be reduced to case $m_0 = 0$ by a rotation of angle $-\theta$ where $m_0 = \tan\theta$.

Case $m_0 = 0$ is very convenient because of its simplicity. The domain for y_0 becomes an interval, the boundaries for x_0 being given by the bounded area between the convex hulls.
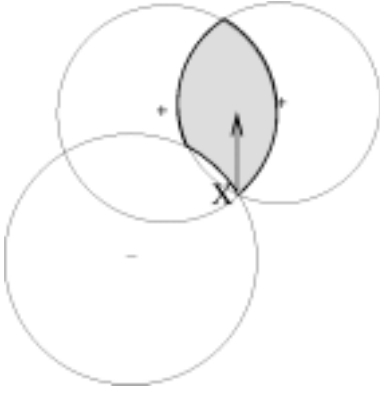
Figure 3: The geometry of the next object position given current sensor values. The future object position has to be inside the shaded area.

3.3 Incorporating History

We now extend the instantaneous characterization of the tracked object over time, using history. Consider Figure 3. Intuitively, future positions of the object have to lie inside all the circles whose center is located at a plus sensor and outside all circles whose center is located at a minus sensor, where the radius associated with each sensor S is $d(S, X)$ where X is the previous object location (by $d(A, B)$ we will denote the distance between points A and B). This observation can be formalized as follows.

PROPOSITION 4. *Let t_0 be a certain time and $t_1 > t_0$ such that sensors S^- and S^+ report $-$ and $+$ respectively at all times $t, \forall t_0 < t < t_1$. Then $\forall t_0 < t < t_1$*

$$d(X(t), S^-) \geq d(X(t_0), S^-) \tag{1}$$
$$d(X(t), S^+) \leq d(X(t_0), S^+)$$

Proof. We prove the claim only for the minus sensor. The other inequality follows by duality. Let

$$(S - X_2(t)) \cdot X_2'(t) = (S - X_1(t) - A) \cdot X_1'(t) =$$
$$= (S - X_1(t)) \cdot X_1'(t) - A \cdot X_1'(t)$$
$$= (S - X_1(t)) \cdot X_1'(t)$$

We have that $f(t_0) = 0$ and $f'(t) = 2 \cdot (X(t) - S^-) \cdot X'(t) \geq 0$ because S^- reports $-$ at any time t between t_0 and t_1, which means that f is nondecreasing. Since $f(t_0) = 0$, it also follows that $f(t) \geq 0 \, \forall t_0 \leq t \leq t_1$. □

4. TRACKING WITH A BINARY SENSOR NETWORK

Section 3 gives constraints on the movement of the targeted object. By also assuming that the object's trajectory lies inside the convex hull of all sensors, a tracking algorithm can be developed. The following subsections describe this algorithm and its limitations.

4.1 The Tracking Algorithm

We derive a solution for tracking with binary sensors using the constraints in Section 3 to obtain an algorithm with the flavor of particle filtering. The key idea of the particle

filtering method is to represent the location density function by a set of random points (or particles) which are updated based on sensor readings and to compute an estimation of the true location based on these samples and weights. Algorithm 1 is a variant of the basic particle filter algorithm. Rather than keeping an equally weighted sample set (as [9] proposes), we use the idea in [8] where each particle has its own weight. The algorithm keeps at each step a set of particles (or possible positions) with weights updated according to the probability of going from the location at time $k - 1$ (denoted by x_j^{k-1}) to the location at time k (denoted by x_j^k). This probability is approximated by $\hat{p}(y_k | x_j^k)$. The first particle set is created by drawing N independent particles outside the convex hulls of the "plus" and "minus" sensors at the time of the first sensor reading. Then, with each sensor reading, a new set of particles is created as follows:

1. a previous position is chosen according to the "old" weights

2. a possible successor is chosen for this position

3. if this successor respects acceptance criterion (which is problem-specific and will be described in Subsection 4.2), add it to the set of new particles and compute its weight.

The above sequence of steps is repeated until N new particles have been generated. The last step is to normalize the weights so they sum up to 1.

Algorithm 1 Particle Filter Algorithm

  Initialization: A set of particles $(x_j^1, w_j^1 = \frac{1}{N})$ for $j = 1, \ldots, N$
  $k = 1$
  while y_k (sensor readings) $\neq \emptyset$ (sensors still active) **do**
    $k = k + 1$
    repeat
      choose j from $(1, 2, \ldots, N) \sim (w_1^{k-1}, \ldots, w_N^{k-1})$
      take $x_j^k = \hat{f}_k(x_j^{k-1}, y_k)$
      if x_j^k respects "goodness" criterion **then**
        accept it as a new particle
      end if
    until N new particles have been generated
    for $j = 1 : N$ **do**
      $w_j^k = w_j^{k-1} * \hat{p}(y_k | x_j^k)$
    end for
    Normalize vector $(w_1^k, \ldots, w_N^k)$
  end while

4.2 Implementation

In this section we describe some of the implementation details behind Algorithm 1. The sensor readings are aggregated as the bit vector reported by the sensors at time k which is denoted by y_k. The object's movement f is approximated by taking x_j^k (the new particle) inside the area given by the following constraints:

- x_j^k has to lie outside the "minus" and "plus" convex hulls (from Theorem 2)

- x_j^k has to lie *inside* the circle of center S_+ and of radius the distance from S_+ to x_j^{k-1} (from Proposition 4),

where S_+ can be any "plus" sensor at sampling times $k-1$ and k

- x_j^k has to lie *outside* the circle of center S_- and of radius the distance from S_- to x_j^{k-1} (from Proposition 4), where S_- can be any "minus" sensor at sampling times $k-1$ and k

The probability of the movement from x_j^{k-1} to x_j^k is approximated by

$$\hat{p}(y_k|x_j^k) = p_{slope}(x_j^k, y_k) \cdot p_{position}(x_j^k, y_k)$$

where p_{slope} is the ratio of possible slopes for the new position x_j^k and $p_{position}$ is a number that quantifies the relative location of the sensors, the old (x_j^{k-1}) and new (x_j^k) positions. More formally,

$$p_{position} = c \cdot \prod_{i=1}^{NS} \rho(S_i, x_j^{k-1}, x_j^k)$$

where c is a normalization constant, NS is the number of sensors and

$$\rho(S_i, x_j^{k-1}, x_j^k) = \begin{cases} 1, & \text{if } s_i^{(k-1)} \neq s_i^{(k)} \\ 1, & \text{if } s_i^{(k-1)} = s_i^{(k)} \text{ and} \\ & S_i, x_j^{k-1} \text{and } x_j^k \text{ respect (1)} \\ \frac{d(S_i, x_j^k)}{d(S_i, x_j^{k-1})}, & \text{if } s_i^{(k-1)} = s_i^{(k)} = 1 \text{ and} \\ & threshold < \frac{d(S_i, x_j^k)}{d(S_i, x_j^{k-1})} \leq 1 \\ \frac{d(S_i, x_j^{k-1})}{d(S_i, x_j^k)}, & \text{if } s_i^{(k-1)} = s_i^{(k)} = -1 \text{ and} \\ & threshold < \frac{d(S_i, x_j^{k-1})}{d(S_i, x_j^k)} \leq 1 \end{cases}$$

The acceptance criterion for x_j^k in Algorithm 1 is $p_{position} > threshold$. A small value for $threshold$ increases the estimation error, whereas a large value for $threshold$ (i.e. close to 1) increases the number of tries for finding a new particle (and thus the running time). A typical value for $threshold$ in our simulation is 0.8.

4.3 Experiments

To evaluate our approach, we implemented Algorithm 1 in **MATLAB** and performed extensive simulations on our implementation. All trajectories are taken inside the $[0,1] \times [0,1]$ square and thus the error measurements are relative to this square. Several types of trajectories have been considered: linear trajectories, trajectories with random turns and trajectories with "mild" turns (at each sensor readings the direction of the tracked object can vary from the previous one with at most $\pi/6$). All trajectories are piecewise linear and the distance traveled by the object between sensor readings is almost constant. A typical simulation example for a linear trajectory (denoted by triangles) can be seen in Fig. 5. The distance traveled between sensor readings is $N(0.12, 0.02)$, i.e. drawn from a normal distribution with a mean of 0.12 and a standard deviation of 0.02.

In Figure 4 we describe the accuracy of our tracking algorithm. The plots show the Root Mean Square Error (RMSE) for three different layouts of sensor networks and trajectories. The two lines in each plot represent different error calculations for the same experiments, namely whether the particles are weighted in the error calculation as they are in the filtering algorithm. For these experiments, the sensors were placed in a grid for the first plot (with 16, 25, 36,..., 196, 225 sensors) and randomly for the other two (with 16, 25, 36,..., 100 sensors). The trajectories are random walks in the first two plots (with "mild" turns) and linear in the last plot. In all plots the distance traveled by the object is $N(0.12, 0.02)$. A simulation example can be seen in Fig 5. The experiments described in the first and second plots were run $N_1 = 50$ times with random trajectories generated at each run. The third experiment was run $N_2 = 50$ times on 5 different linear trajectories. In all experiments 200 particles were sampled at each sensor reading.

The data shows a decreasing trend for the estimation error as the number of sensors increases, especially in the third case, where the trajectories are linear. However the error can not be made arbitrarily small even with a large number of sensors. The reason for this effect is explained graphically in Fig. 5, where three parallel trajectories are shown, all of which are consistent with the obtained sensor readings. Theorem 7 shows that certain sets of trajectories (including trajectories on parallel lines that respect the conditions in the theorem) can not be discerned by a binary sensor, regardless of its placement. In Fig. 5, the real trajectory is denoted by triangles and the trajectories parallel to it are denoted by stars. The snapshots are taken at the time of the last sensor reading, corresponding to the last point of the trajectory. The "plus" sensors are given as squares and the "minus" sensors as circles. The dots represent the cloud of particles at each step. The second example illustrates the major limitation of our model: binary sensors can only give information about the movement direction of an object but not about its' position as it will be shown in Section 4.4. In this example the actual trajectory starts and ends at point $0.75, 0.933$ (up, at right). The direction of the estimated trajectory gets approaches the actual movement direction, but the estimated location is far from the actual location.

4.4 Model Limitation

Our simulation results suggest a natural limitation for the binary sensor model. The information provided by a binary sensor network can only be used to obtain reliable information about the motion direction of the tracked object. The results in this section show that certain pairs of trajectories are indistinguishable for any binary sensor. We also describe such pairs of trajectories by presenting a constructive method for producing them. In particular, we show that two trajectories which always have parallel velocities obeying a given constraint and are always a constant distance apart cannot be differentiated under the binary sensor model.

Suppose two points, $X(t)$ and $Y(t)$, are moving so that they are indistinguishable for *all* possible binary sensors in the plane according to our binary sensor model.

Lemma 5 shows that the velocity vectors, $X'(t)$ and $Y'(t)$, have to be parallel to each other and perpendicular to the difference vector $X(t) - Y(t)$.

LEMMA 5. *For all times, t, $X'(t) = \frac{dX(t)}{dt} = \gamma(t)Y'(t)$ for some scalar function $\gamma(t) > 0$. Moreover, $(X(t) - Y(t)) \cdot X'(t) = 0$ for all times t.*

Proof. Consider $X(t)$ and $X'(t)$. The two half spaces determined by the line going through $X(t)$ and orthogonal to $X'(t)$ partition sensors into two groups: the half space into which $X'(t)$ points contains sensors that will detect X
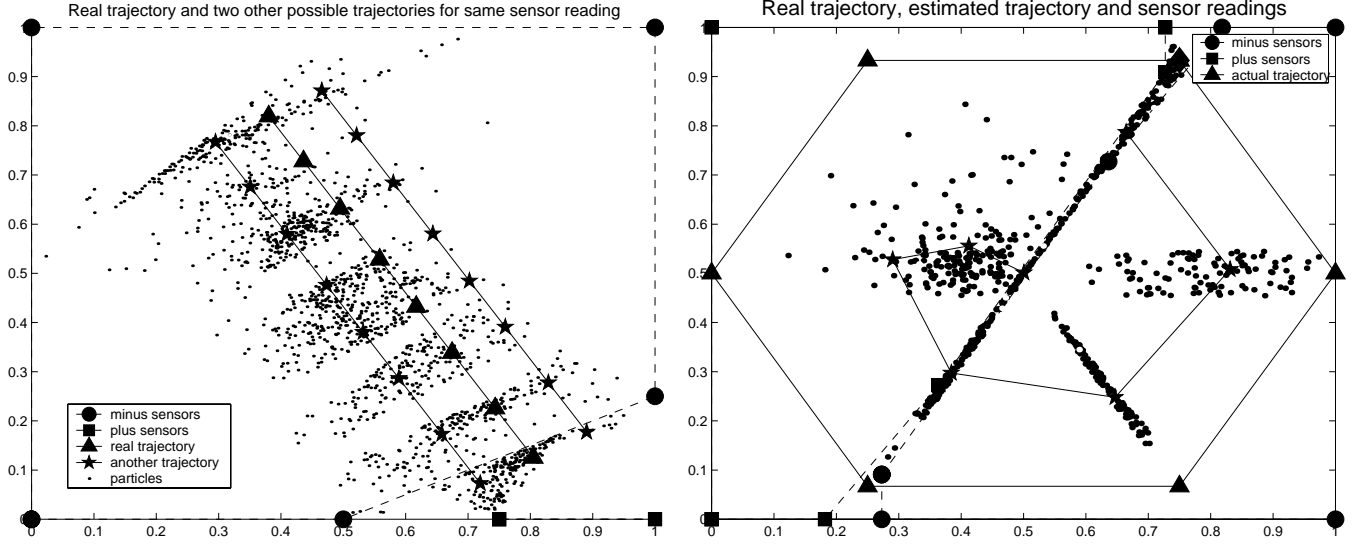
Figure 5: Simulation examples for Algorithm 1. The plus sensors are denoted by squares and the minus sensors are denoted by circles. Also, the plus and minus convex hulls are figured. In the first example the two trajectories figured by ⋆ are other possible trajectories consistent with sensor readings. In the second example the estimated trajectory is figured by ⋆.

approaching while the other half space of sensors will detect $X(t)$ moving away.

Consider the two half spaces thus partitioned by the other point, $Y(t)$ at time t as well. If the half spaces do not coincide, the region R depicted in Fig. 6 a. will contain sensors which detect X as moving away but Y as approaching at time t, or vice versa. Therefore the half spaces must coincide and so $X'(t) = \frac{dX(t)}{dt} = \gamma(t)Y'(t)$ for some scalar function $\gamma(t) > 0$. The assertion that $(X(t) - Y(t)) \cdot X'(t) = 0$ clearly follows as well. □

Lemma 6, which is a corollary of Lemma 5, shows that $X(t)$ and $Y(t)$ must be at a constant distance from each other at all times. Let $X(t) = Y(t) + a(t)$.

LEMMA 6. $||a(t)||^2 = a(t) \cdot a(t) = constant.$

Proof. By definition, $X(t) = Y(t) + a(t)$ so that

$$a(t) \cdot X(t) = a(t) \cdot (Y(t) + a(t)) = a(t) \cdot Y(t) + ||a(t)||^2.$$

Now differentiate both sides with respect to t to get (dropping the time dependence of all vectors for simplicity)

$$a' \cdot X + a \cdot X' = a' \cdot Y + a \cdot Y' + \frac{d||a||^2}{dt}.$$

Using the fact that $a \cdot X' = a \cdot Y' = 0$ from Lemma 1, we get

$$a' \cdot (X - Y) + a \cdot (X' - Y') = a' \cdot a = \frac{d||a||^2}{dt} = 2a' \cdot a.$$

Thus $\frac{d||a||^2}{dt} = a' \cdot a = 0$ at all times and so $||a||$ is a constant. □

Theorem 7 puts together the results in the precedent lemmas and shows that the necessary indistinguishability conditions are also sufficient.

THEOREM 7. *Two trajectories $X(t)$ and $Y(t)$ are indistinguishable for* all *possible binary sensors in the plane if and only if both the following conditions hold:*

- $X'(t) = \gamma(t)Y'(t)$, *where $\gamma(t) > 0$ $\forall t$ is a scalar function*

- $(X(t) - Y(t)) \cdot X'(t) = 0$ *(or $(X(t) - Y(t)) \cdot Y'(t) = 0$)*

Proof.

If $X(t)$ and $Y(t)$ are indistinguishable for *all* possible binary sensors then Lemma 5 and Lemma 6 show that the two conditions hold.

Suppose the above conditions hold. Let S be an arbitrary sensor in the plane. S reports $\mathrm{sgn}((S - X(t)) \cdot X'(t))$ for $X(t)$ and $\mathrm{sgn}((S - Y(t)) \cdot Y'(t))$ for $Y(t)$. We have

$$(S - X(t)) \cdot X'(t) = (S - Y(t) - (X(t) - Y(t))) \cdot (\gamma(t)Y'(t))$$
$$= \gamma(t)((S - Y(t)) \cdot Y'(t) - (X(t) - Y(t)) \cdot Y'(t))$$
$$= \gamma(t)(S - Y(t)) \cdot Y'(t).$$

Because $\gamma(t) > 0$ at all times t we get that

$$\mathrm{sgn}((S - X(t)) \cdot X'(t)) = \mathrm{sgn}((S - Y(t)) \cdot Y'(t))$$

which shows that $X(t)$ and $Y(t)$ are indistinguishable for sensor S. As S is arbitrarily chosen we get that the two trajectories are indistinguishable for any sensor. □

Theorem 7 implies that the two points must be moving along a path determined by a radius of some circle at all times, although the circle's radius can change over time as long as it is larger than $||a||$ and even be infinite (which is the degenerate case of moving along parallel lines). Fig. 6 b. shows the transition from a straight, parallel trajectory to following an arc of a circle. The transitions can happen smoothly since the points can come to rest at the point of transition and then start again.
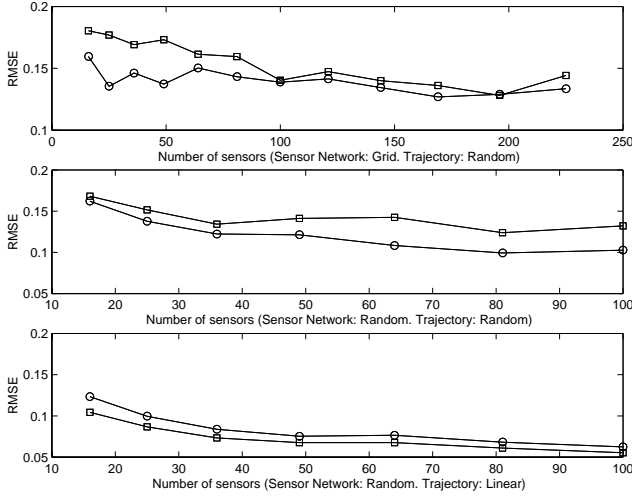
Figure 4: Root Mean Square Error (RMSE) of tracking with different sensor network layouts and number of sensors. The RMSE is based on the error of all particles at a given time. The squares in each plot denote the error based on weighting the particles equally in the error calculation while the circles denote the error when the particles are weighted in the error calculation according to their probabilities.

The following result shows that, under mild conditions, given a parametric curve we can easily identify another curve at arbitrary distance that is indistinguishable in our sensor model. Before engaging in the proof we need to recall some basic facts of the differential geometry of plane curves.

DEFINITION 1. *Let $X(t) = (x_1(t), x_2(t))$ be a twice differentiable parameterized plane regular curve. The (signed) curvature of $X(t)$ at t is given by*

$$k(t) = \frac{x_1' x_2'' - x_1'' x_2'}{\|X'\|^3} \;.$$

THEOREM 8. *Let $X(t) = (x_1(t), x_2(t))$ be a parameterized plane regular curve at least twice differentiable. Then, for all $\alpha \in \mathbf{R}^+$ such that $k(t) < 1/\alpha$, there exists a parameterized plane curve $Y(t) = (y_1(t), y_2(t))$ indistinguishable from $X(t)$, such that $\forall t \; \|X(t) - Y(t)\| = \alpha$.*

Moreover, for all such α there exist at most two indistinguishable curves $Y(t)$ from $X(t)$ such that $\|X(t) - Y(t)\| = \alpha$.

Proof. We'll first prove the existence of such a plane curve by constructing it.

Since X is regular, i.e., $X'(t) \neq 0$ for all t, the following curve is well defined:

$$Y(t) = (x_1(t), x_2(t)) + \frac{\alpha}{\|X'(t)\|}(-x_2'(t), x_1'(t)) \;.$$

We can immediately observe that $a(t) = Y(t) - X(t) = \frac{\alpha}{\|X'(t)\|}(-x_2'(t), x_1'(t))$ verifies

- $a(t) \cdot X'(t) = \frac{\alpha}{\|X'(t)\|}(-x_2'(t), x_1'(t)) \cdot (x_1'(t), x_2'(t)) = 0$, and
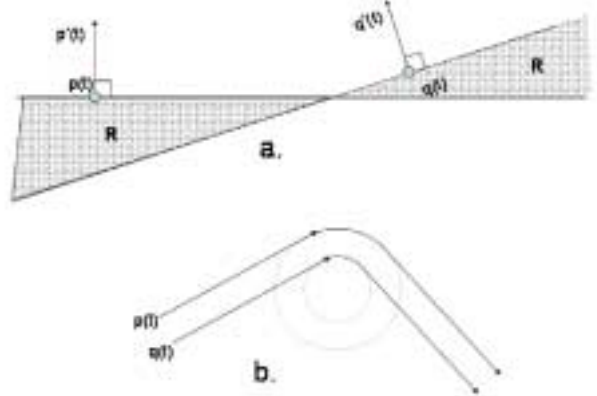
- $\forall t \; \|a(t)\| = \alpha$.

Figure 6: An illustration of the indistinguishability properties of our sensor model. Part a. shows that the two velocities $X'(t)$ and $Y'(t)$ have to be parallel and $X(t) - Y(t)$ must be perpendicular to them. Otherwise sensors in the shaded region R would give different reports for $X(t)$ and $Y(t)$. Part b. shows an example of two piecewise linear or circular trajectories that are indistinguishable by any binary sensor.

So it will be enough to show that $Y'(t) = \gamma(t) \cdot X'(t)$, for some scalar function $\gamma(t) > 0$ or, equivalently, that

1. $a(t) \cdot Y'(t) = 0$, and

2. $X'(t) \cdot Y'(t) > 0$.

In fact, condition 1 will tell us that X' and Y' lie along parallel directions, whereas, condition 2 will ensure that the two velocity vectors are not antiparallel. After dropping the dependence upon t for convenience of notation we can write:

$$Y' = (x_1', x_2') + \frac{\alpha}{\|X'\|}(-x_2'', x_1'') - \frac{\alpha}{\|X'\|^2}\frac{X'}{\|X'\|}X''(-x_2', x_1') \;.$$

Let us first show the validity of condition 1 on the orthogonality. We have:

$$a(t) \cdot Y'(t) = a(t) \cdot (X'(t) + a'(t)) = a(t) \cdot X'(t) + a(t) \cdot a'(t).$$

We already know that $a(t) \cdot X'(t) = 0$. As $\|a(t)\| = \alpha$ we get that $a(t) \cdot a'(t) = 0$. Hence, $a(t) \cdot Y'(t) = 0$.

Let us now verify the validity of condition 2 that depends upon our constraint on the curvature. Expanding $X'Y'$ we obtain:

$$
\begin{aligned}
X' \cdot Y' &= X' \cdot X' + X' \cdot \frac{\alpha}{\|X'\|}(-x_2'', x_1'') \\
&\quad - X' \cdot \frac{\alpha}{\|X'\|^3}X'X''(-x_2', x_1') \\
&= \|X'\|^2 + \frac{\alpha}{\|X'\|}X'(-x_2'', x_1'') \\
&= \|X'\|^2 - \alpha\frac{x_1'x_2'' - x_1''x_2'}{\|X'\|} \\
&= \|X'\|^2(1 - \alpha k) \;.
\end{aligned}
$$

157

And finally we can see that $X'Y' > 0$ if and only if $k < 1/\alpha$ as assumed[1].

Let's prove now that the curve constructed above is the only twice differentiable curve Y at constant distance α from X such that X and Y are indistinguishable. Let Y be a plane curve at constant distance α from X such that X and Y are indistinguishable under our sensor model. By Theorem 7 we get that $(X(t) - Y(t)) \cdot X'(t) = 0$.

Let's denote the unit normal vector to X at time t by $N_X(t)$. From the definition of $N_X(t)$ we get that $N_X(t) \cdot X'(t) = 0$. This means that the directions of vectors $X(t) - Y(t)$ and $N_X(t)$ are the same or, equivalently, that there exists a scalar function $\gamma(t)$ such that $X(t) - Y(t) = \gamma(t)N_X(t)$.

We also assumed that $||X(t) - Y(t)|| = \alpha$. Using this we get $||\gamma(t)N_X(t)|| = \alpha$ or $|\gamma(t)|||N_X(t)|| = \alpha$ or further $|\gamma(t)| = \alpha$ because $||N_X(t)|| = 1$. As X and Y are twice differentiable, $X' \neq 0$ and $\gamma(t) = (X(t) - Y(t))/||X'(t)||$ we get that $\gamma(t)$ is a constant, equal in absolute value with α. We conclude our proof with the observation that we can have at most two different curves $Y(t)$. We have exactly two curves if $k(t) < -1/\alpha$ also holds.

We may observe the following. Let $\alpha(s)$ be a curve parameterized by Arc Length and be $n(s)$ the unit vector orthogonal to $\alpha'(s)$ at s. Then, by requiring that the basis $(\alpha'(s), n(s))$ is oriented as the canonical basis (e_1, e_2) we can give a sign to the curvature by defining $\alpha''(s) = k(s) \cdot n(s)$.

Thus the sign of k provides information about whether the curve is turning towards the normal vector $n(s)$ ($k > 0$) or away from it ($k < 0$). So, we need to be careful with the interpretation of $k < 1/\alpha$ for if $k < 0$ the constraint will be always verified. However this fact means that $Y(t)$ can be at arbitrary distance from $X(t)$ only if it lies on the positive direction of the normal vector $n(s)$ (away from the direction of the turn of X). In other words, our constraint on the curvature says that the distance between the two curves must always be lower than the largest of the two rays of curvature. $\square$

5. TRACKING WITH A PROXIMITY BIT

As Theorem 7 shows, there exist pairs of trajectories that can not be distinguished by any binary sensor. We conclude that additional information is needed to disambiguate between different trajectories and to identify the exact location of the object. This can be realized by adding a second binary sensor capable of providing proximity information (such as an IR sensor) to each sensor node in the network. If the object is detected within some set range of the proximity sensor, that node broadcasts a message to the base station. The range of the proximity sensor may be different and much smaller than the range of the movement direction sensor. It is useful to set the proximity range so that the sensors are non-overlapping (this can be done by appropriate thresholding) but this is not necessary. The base station will approximate the location of the object in the region covered by all the sensors reporting object detection. For simplicity of presentation we assume for the rest of the session that the detection range can be calibrated so that at most one sensor detects the object at a time.

---

[1]The ray of curvature is by definition $R = 1/|k|$.

5.1 Algorithm and Implementation

Algorithm 2 describes the solution to tracking that uses a motion direction bit and a proximity bit in each sensor node. Algorithm 2 extends Algorithm 1 using the proximity information. When a sensor node detects the object, the ancestors of every particle which is not inside the range are shifted as far as the last time the object was spotted by proportional amounts. Note that this algorithm reduces to Algorithm 1 when no proximity sensor is triggered, so it is not necessary for the proximity sensors to cover the entire region.

Algorithm 2 Algorithm for Binary Sensors with Range

Use Algorithm 1 as basis.
if sensor S sees the object **then**
  for all accepted particles P not inside the range of S **do**
    Let P' (a new particle) be the intersection between the range of S and semi-line $(PS]$
    Let $P_1, \ldots, P_k$ be the ancestors of P since the last time the object was spotted.
    for $i = 1$ to k **do**
      $P_i = P_i - (P - P')/(k + 1)$
    end for
  end for
end if

5.2 Experiments

If we assume the sensors have the ability to report the presence of the object in their proximity, then the metric for the performance of the algorithms should be the relative error *after* the object is first spotted. Because we expect trajectories to be winding over the area covered by the sensor network we first ask how efficient the proximity sensing is at detecting the object. More specifically, this can be formulated as "After how many time steps is the object first spotted given a sensor layout?". Some simulation results are shown in Fig. 7, that show how many trajectories out of 100000 randomly generated trajectories have entered a sensor range after k steps, where k goes from 1 to 800. The total number of trajectories for each subplot is: 46111 (top, left), 83425 (top, right), 61173 (down, left) and 90235 (down, right). In each graph the remaining trajectories were not spotted at all or were spotted after more than 800 readings. The average length of a trajectory is about 146. The trajectories were generated as follows: the distance traveled between sensor readings is $N(0.02, 0.001)$ and the changes in direction are "mild" (that is, the direction can change at most $\pi/6$ between sensor readings). The results are for 25 and 100 sensors. The starting position is randomly chosen. Fig. 7(right) shows the results for a small range value (where the ranges cover less than 10% of the whole area). Fig. 7(left) shows the results for a large range value, (where the ranges cover about 70% of the whole area). The graphs suggest that the distribution of the amount of time that passes until an object is first spotted is exponential.

Two simulation examples of Algorithm 2 are shown in Fig. 9. On the first example, the object gets in the proximity range of a sensor at readings time $t = 5$ (when all particles can be seen to reset very close to the true object position) and $t = 11$ (the last reading, near the top of the plot). On the second example, the object gets in the proximity
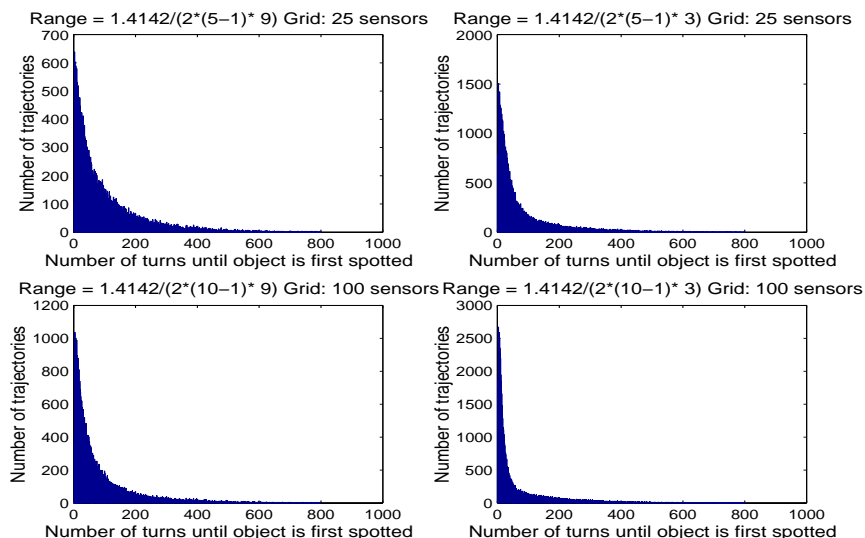
Figure 7: The graphs show on the x-axis the number of readings until the object gets first time in a sensor range and on the y-axis the number of trajectories for a given number of readings elapsed.

range of a sensor at reading time $t = 3$ (near the center of the plot). The real trajectory is denoted by triangles and the estimated trajectory is marked with a thick dashed line. The snapshot is taken at the time of the last sensor reading, corresponding to the last point of the trajectory. The "plus" sensors are given by squares and the "minus" sensors by circles. The dots represent the particles *after* the shifting step in Algorithm 2. We have repeated this simulation over 200 example trajectories computed by 16 to 64 node sensor networks (1000 runs in total). The trajectory approximated by the sensor network is very good, and has root mean square error ranging between 0.15 (for a 16 node sensor network) and 0.02 (for a 64 node sensor network). We believe that for field tracking applications involving animals, people, or cars, these are practical approximations. The tracking performance after the proximity bit was added to the model is shown the picture on the right in Figure 8. The simulation conditions are similar to the ones for the picture on the left, considering sensors placed in a grid or randomly with random or linear trajectories.

Two error models were considered and they are explained below.

Suppose $r(k)$ is the actual position of the object, $p_i(k)$ is the i-th particle generated by the algorithm and w_i its weight at reading time t out of n reading times.

First error model is the Root Mean Square Error **RMSE** (denoted by squares), which calculates at each time step the distance between the particle cloud centroid (regarded as an estimator for the actual position) and the actual position of the object. More precisely, **RMSE** calculates $\sqrt{\frac{1}{n} \sum_k E_k^2}$, where

$$E_k = ||r(k) - \sum_i w_i p_i(k)||.$$

The other error model (what we call "average error") calculates at each time step the average distance from the particles in the cloud to the true position. In other words, the

average error is equal to $\frac{1}{n} \sum_k E_k'$ where

$$E_k' = \sum_i w_i ||r(k) - p_i(k)||$$

Second error model gives a bigger error showing a significant variance within the particle cloud. The second error model is more relevant if we think of each particle instead of the particle cloud centroid as an estimator of the true position of the object. In extreme cases such as all particles being on a circle around the true position, **RMSE** can be 0 while the other error provides a better interpretation of the tracking performance.

The data shows the same decreasing trend for the estimation error as in the one-bit model, but the error is lower and has a faster decreasing rate.

6. CONCLUSIONS AND FUTURE WORK

In this paper we studied the computational power of binary sensor networks with respect to the tracking application. We take a minimalist stance and ask what can a simple binary sensor compute given two different types of sensed data. We assume that the nodes in the network have sensors that can give one bit of information only. We derive a geometric characterization for the tracking problem when the sensor nodes provide one bit of information about the object: is the object approaching or moving away from the sensor? We show that this problem setup leads to a tracking algorithm that gives good information about the direction of movement of the object but that additional information is needed to provide the exact location of the object. A proximity sensor bit can provide this information and the tracking algorithm can be extended to use this information. The resulting error in trajectory prediction is low. Thus, since broadcasting single bits over a network is feasible, and the computation performed by the base station in response to the sensor values is fast, we conclude that the binary sensor model is a practical solution to certain tracking applications.
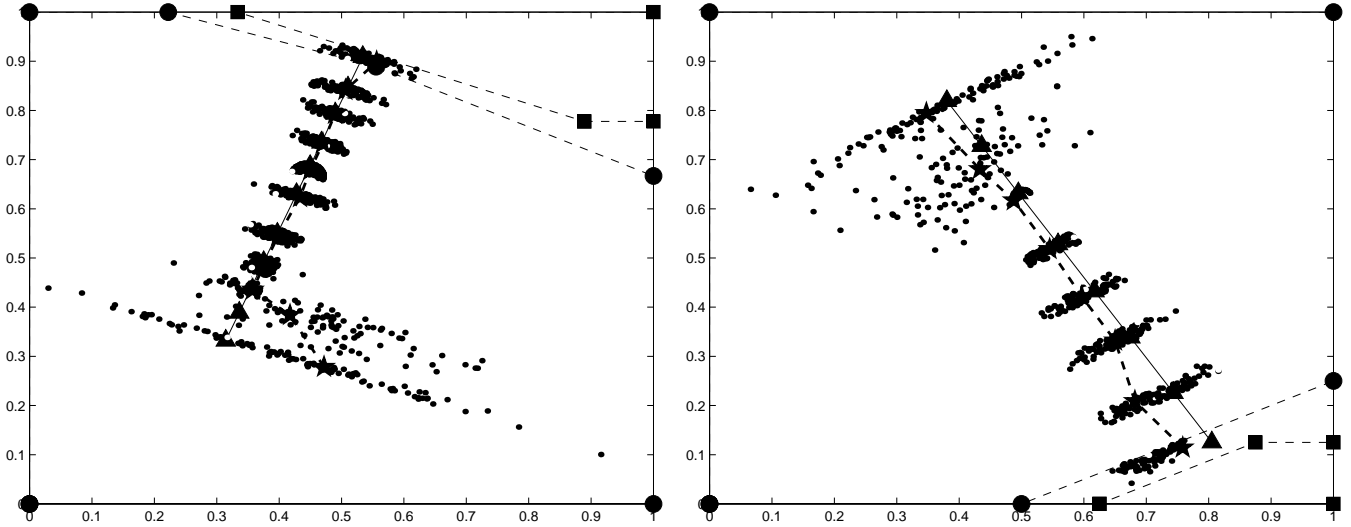
Figure 9: Simulation examples for Algorithm 2.
In the first run, the object gets in a sensor range at sampling times 5 (position $(0.3971, 0.5495)$) and 11 (position $(0.5341, 0.9091)$).
In the second run, the object gets in a sensor range only at sampling time 3 (position $(0.495, 0.632)$). The plus sensors are denoted by squares and the minus sensors are denoted by circles. The actual trajectory is denoted by triangles (the thin line) and the estimated trajectory by stars (the thick dashed line). In both runs the sensor readings shown are taken at last step and only the sensors on the boundary of the minus and plus convex hulls are shown.

Several important aspects of the binary model sensor model remain open, and we plan to consider these in our future work.

First, real world sensors are influenced by noise. We can incorporate noise in our model by adding a Gaussian variable ε to the signal strength gradient $\frac{d}{dt}S_i(t)$ at sensor S_i and then quantize it as -1, 0 or 1. A 0 report at a certain time means that the sensor's signal strength gradient is below a certain threshold and thus not reliable enough, which can also be regarded as a temporarily shutdown of the sensor. The Gaussian variable ε has zero mean, but its variance should be determined from real data reflecting sensors' characteristics.

Another way of dealing with noise is to ignore the information given by "untrustworthy" sensors. We can decide which sensors are not reliable at a certain time t by approximating a sensor's reading based on the sensors' geometry.

One possible approach is to consider a snapshot of the sensors at time t. Let V_+ be the set of plus sensors visible from the minus convex hull and V_- the set of minus sensors visible from the plus convex hull. Let G_+ and G_- be respectively their centroids. Take E_+ and respectively E_- be the points where line G_+G_- enters the plus and minus convex hulls. Finally let M be the middle point of segment E_+E_-.

We take M as a very rough approximation of the object's location and the line E_+E_- as an approximation of the object's direction. Then we can write the measure of a plus sensor's S_+ reliability as:

$$\mu(S_+) = d(S_+, M) \cdot \cos(S_+MG_+),$$

where $d(A, B)$ is the Euclidean distance between A and B.

For a minus sensor the measure is

$$\mu(S_-) = d(S_-, M) \cdot \cos(S_-MG_-).$$

The measure approximates the sensor-dependent part of the scalar product $(S - X(t)) \cdot X'(t)$, which can be written as

$$\|S - X(t)\| \cdot \|X'(t)\| \cdot \cos\left(\angle\left(S - X(t), X'(t)\right)\right)$$

A first observation is that for a sensor S_+ the angle S_+MG_+ is rarely greater than $\pi/2$. Even then, it has to be that S_+ is close to the minus convex hull and the possible directions for the object's movement are very limited. In the presence of noise one might want to discard such sensors anyway. This measure only uses the frontier sensors, i.e. the ones that are visible from the other convex hull. The non-frontier sensors do not matter if the frontier sensor reports are accurate. An interior sensor, for example, to the minus convex hull cannot report + because then the plus and minus convex hulls would be no longer disjoint, contradicting Theorem 2. Considering the non-frontier sensors too would change the centroid's position without adding extra information. Finally, the measure is symmetric for plus and minus sensors. So we can ignore (do not trust) a sensor if its measure is below a certain threshold.

Another open question is the effectiveness of tracking relative to the amount of data available.

In the paper we start with the one-bit model and then add a second bit for proximity. One would naturally be concerned about how adding extra bits influences the tracking accuracy. If k bits are available, an interesting problem is to find the best way to allocate these bits with respect to direction and proximity. If the sensor density is high and proximity is sampled often enough then direction can be inferred from those two and so velocity and proximity are
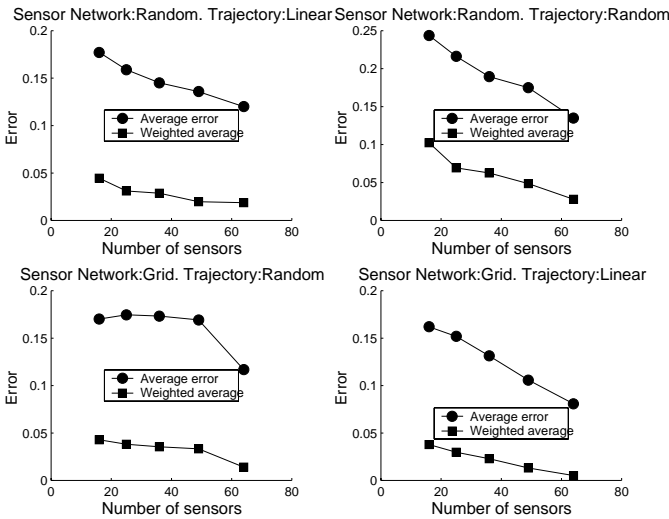
Figure 8: Tracking error for various network layouts and number of sensors for systems with a proximity bit at each sensor node. The squares represent the RMSE error based on all the particles separately, while the circles represent the average error calculated based on the weighted average of all particles.

not independent variables. This suggests that a compressing scheme could be used to send more information over the network. We thus get a new optimization problem having as parameters the number of bits used, the sensor density and the bits' allocation scheme.

A possible drawback of our method is the centralized computational structure. An approach for a decentralized solution is to have every sensor run a local particle filter using only a subset of the information read by the other sensors.

The basic idea is that at each time step t every sensor S requests information (the bits) only from the sensors that are likely to flip based on its local information. S assumes that the object moved on the same direction and traveled the same distance between times $t-2$ and $t-1$ as between times $t-1$ and t (thus the predicted position at time t is on the same line as the positions at time $t-2$ and $t-1$) and only requests information from the sensors that would flip based on this trajectory. In addition to this information, the sensor requests information from a fixed number of sensors randomly chosen. This is useful in order to have control on trajectories that are not close to linear. The remaining sensors are assumed to remain unchanged. If the sensor readings available at sensor S do not respect the necessary conditions in Theorem 2 then the sensor updates its information by requesting data from all the sensors. In the beginning each sensor is assigned a different area as possible starting location of the object. At first two time steps every sensor gets the readings from all sensors so that the starting information is accurate.

In the near future we will investigate how to implement this algorithm using our Mote network testbed and how to extend our algorithms to support multiple target tracking.

7. REFERENCES

[1] S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp, A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking, *IEEE Transactions of Signal Processing*, Vol. 50(2), 174-188, February 2002.

[2] R. R. Brooks, P. Ramanathan, and A. Sayeed, Distributed Target Tracking and Classification in Sensor Networks, *Proceedings of the IEEE*, September 2002

[3] B. Krishnamachari, Energy-Quality Tradeoffs for Target Tracking in Wireless Sensor Networks, *IPSN 2003*, 32-46.

[4] H. Yang and B. Sikdar, A Protocol for Tracking Mobile Targets using Sensor Networks, *Proceedings of IEEE Workshop on Sensor Network Protocols and Applications, 2003*.

[5] D. Crisan and A. Doucet. A survey of convergence results on particle filtering for practitioners, 2002.

[6] Bruce R. Donald, James Jennings, and Daniela Rus. Information invariants for distributed manipulation. *International Journal of Robotics Research*, 16(5):673–702, 1997.

[7] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proc. of IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 22–29, 1998.

[8] P. Clifford, J. Carpenter and P. Fearnhead. An improved particle filter for non-linear problems. In *IEE proceedings - Radar, Sonar and Navigation*, I46:2–7, 1999.

[9] D. Salmond, N. Gordon and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proc.F, Radar and signal processing*, 140(2):107–113, April 1993.

[10] Eduardo Nebot, Favio Masson, Jose Guivant, and Hugh Durrant-Whyte. Robust simultaneous localization and mapping for very large outdoor environments. In *Experimental Robotics VIII*, 200–9. Springer, 2002.

[11] Lynne E. Parker. Cooperative motion control for multi-target observation. In *Proc. of IEEE International Conf. on Intelligent Robots and Systems*, pages 1591–7, Grenoble, Sept. 1997.

[12] Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446), 1999.

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

Acknowledgments