

# OBJECT TRACKING BY SCATTERED DIRECTIONAL SENSORS

*Kurt Plarre and P. R. Kumar*

Coordinated Science Laboratory  
University of Illinois at Urbana-Champaign  
1308 W. Main St., Urbana, Illinois, 61801  
{plarre, prkumar}@control.csl.uiuc.edu

## ABSTRACT

We explore the application of directional sensors in a wireless sensor network to track objects moving at constant velocity. We model the field of vision of each directional sensor as a line, and the data are the times at which sensors detect objects crossing their lines. The network is initially deployed by scattering nodes, and the locations and directions in which the sensor point are also unknown a priori, in addition to the trajectory of the objects.

The estimation problem involves the solution of a highly nonconvex optimization problem. However we develop a novel three phase algorithm to solve the problem. It first chooses a coordinate basis adapted to the motions of the first two objects, and then localizes the sensors with respect to that basis, then refines it as new objects arrive, and finally transforms the basis if the GPS positions of six of the deployed nodes are known. All computations are performed in-network.

## 1. INTRODUCTION

Sensor networks present a rich and challenging environment for the design of application algorithms (e.g. [1]). The challenges faced by an engineer designing algorithms for sensor networks are novel in that, while traditional algorithms are designed to handle large problems in powerful systems, algorithms for sensor networks must be designed to solve small and medium sized problems with scarce computational power and memory resources. As an added difficulty, algorithms must be designed taking into account issues such as power consumption, communications, and node failures (e.g., [2]).

From a different point of view, sensor networks integrate many tasks that were traditionally separated: Sensing,

computation, communication, and actuation. To optimize the use of resources, algorithms for sensor networks need to seamlessly integrate these tasks. Thus, design of application algorithms must draw on ideas from filtering and estimation theory, information theory, computational complexity, and control theory.

In order to understand the different difficulties that might arise in sensor networks, it is useful to design classes of abstract problems that involve many of the issues mentioned. In [3] we have studied the problem of detecting a static heat source with temperature sensors, that is a canonical representative of problems using omnidirectional sensors to localize a stationary object. We introduced a two phase optimization algorithm that solves the nonconvex optimization problem involved. Local estimates formed by each sensor are fed into a distributed algorithm to accurately localize the heat source. We also introduced the notion of “increasing correctness” (see Section 5.1).

In this paper we study a somewhat opposite problem which is representative of scenarios where highly directional sensors are deployed to track moving objects. We address the problem of tracking of moving objects by a network of directional sensors scattered in a domain: A certain region is monitored by such a network of directional sensors whose positions and orientations themselves are initially unknown. The region is crossed sporadically by objects assumed to be moving at constant velocity at least within a bounded domain of interest. The task of the network is to detect each object and determine its movement trajectory.

We model the “field of vision” of sensors as lines. A sensing node detects an object when it crosses its corresponding line. The data available to the sensors are thus the times at which sensors detect each object. The locations of the sensors and the directions in which sensors point are unknown a priori. The sensor directions are estimated as part of the problem. Section 1 presents a more detailed description of the setup.

The above described estimation problem involves the minimization of a nonconvex function. To overcome this difficulty, we devise a novel three phase optimization algo-

---

This material is based upon work partially supported by NSF under Contract Nos. NSF ANI 02-21357 and CCR-0325716, USARO under Contract Nos. DAAD19-00-1-0466 and DAAD19-01010-465, DARPA/AFOSR under Contract No. F49620-02-1-0325, DARPA under Contract Nos. N00014-0-1-1-0576 and F33615-0-1-C-1905, and AFOSR under Contract No. F49620-02-1-0217.

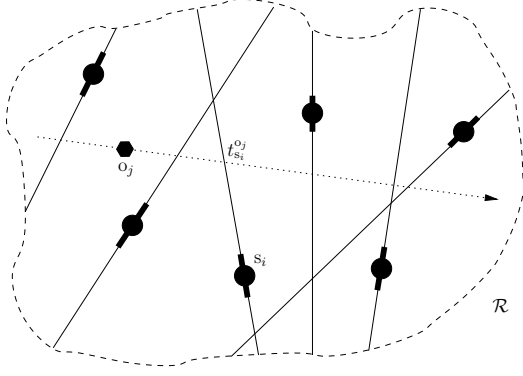


Fig. 1. Problem setup, and naming conventions.

rithm. In the first phase, the motion of the first four objects, and the sensor directions of six sensors are estimated. This problem can be solved in closed form (see Section 4.1). During the second phase, the parameter estimates are updated, every time a new object is detected. The third phase is optional and consists of transforming the estimated parameters from an “internal” representation to the “real world” coordinate system, if the locations of six sensors are known somehow, e.g. through GPS.

To prolong the lifetime of the network, we make use of the “sleep state,” a low power state in which sensors consume a fraction of the power when “awake.” After the first four particles have been detected, most sensors go to sleep. Only a few “sentinel nodes” remain awake. When a sentinel sensor detects a new object, it wakes up all other nodes (see Section 4.2).

## 2. PROBLEM DESCRIPTION

A network of directional sensors  $\{s_1, s_2, \dots\}$  monitors a certain region  $\mathcal{R}$  of the plane. Sporadically, objects moving at constant velocity cross  $\mathcal{R}$ . The size of each object is assumed negligible. A sensor detects an object when it crosses its “line of sight,” which is modeled simply as a line. See Figure 1.

Sensors are initially deployed by scattering them in a domain. Their locations and the directions of their sensors are unknown a priori. We do not assume any distribution on the locations of the sensors, or the directions of their lines of sight. The goal of the system is to describe the orientation of the sensors and the motions of all the objects in some coordinate system. Later we will show that if at least six sensors know their exact locations (for example, using GPS) then it is additionally possible to obtain the absolute correct positions with respect to an externally specified coordinate system.

It is important to mention that there are degenerate situ-

ations in which the proposed algorithm cannot be used, for example, if all sensor lines are perfectly parallel. We disregard such situations, and design the algorithm for a generic “good” case. Also, only one object can be handled by algorithm, at a time.

The equation of the line of sight of sensor  $s_i$  is

$$\frac{x_{s_i}}{a_{s_i}} + \frac{y_{s_i}}{b_{s_i}} = 1 \quad \text{or} \quad \bar{a}_{s_i} x_{s_i} + \bar{b}_{s_i} y_{s_i} = 1.$$

The movement of object  $o_j$  will be described by

$$\begin{aligned} x_{o_j}(t) &= v_{o_j}^x t + x_{o_j}^0, \\ y_{o_j}(t) &= v_{o_j}^y t + y_{o_j}^0. \end{aligned}$$

The time at which sensor  $s_i$  detects object  $o_j$  is

$$t_{s_i}^{o_j} = \frac{1 - \bar{a}_{s_i} x_{o_j}^0 - \bar{b}_{s_i} y_{o_j}^0}{\bar{a}_{s_i} v_{o_j}^x + \bar{b}_{s_i} v_{o_j}^y} + \nu_{s_i}^{o_j},$$

where  $\nu_{s_i}^{o_j}$  zero mean noise, and  $\nu_{s_i}^{o_j}$  is independent of  $\nu_{s_k}^{o_l}$  for  $(s_i, o_j) \neq (s_k, o_l)$ .

To each  $t_{s_i}^{o_j}$  we associate a random variable

$$\tau_{s_i}^{o_j} := (\bar{a}_{s_i} v_{o_j}^x + \bar{b}_{s_i} v_{o_j}^y) t_{s_i}^{o_j} + (\bar{a}_{s_i} x_{o_j}^0 + \bar{b}_{s_i} y_{o_j}^0) - 1.$$

The estimation of the object motion and sensor direction parameters will be based on the minimization of the cost function

$$J = \sum_{i,j} (\tau_{s_i}^{o_j})^2, \quad (1)$$

where, for simplicity, the arguments of  $J$ , which are the unknown parameters, are not shown explicitly. To show the difficulty of minimizing this sum of squares polynomial, we give the expanded form of  $J$ , for three sensors and two objects:

$$\begin{aligned} J = & \left[ (\bar{a}_{s_1} v_{o_1}^x + \bar{b}_{s_1} v_{o_1}^y) t_{s_1}^{o_1} + (\bar{a}_{s_1} x_{o_1}^0 + \bar{b}_{s_1} y_{o_1}^0) - 1 \right]^2 + \\ & \left[ (\bar{a}_{s_1} v_{o_2}^x + \bar{b}_{s_1} v_{o_2}^y) t_{s_1}^{o_2} + (\bar{a}_{s_1} x_{o_2}^0 + \bar{b}_{s_1} y_{o_2}^0) - 1 \right]^2 + \\ & \left[ (\bar{a}_{s_2} v_{o_1}^x + \bar{b}_{s_2} v_{o_1}^y) t_{s_2}^{o_1} + (\bar{a}_{s_2} x_{o_1}^0 + \bar{b}_{s_2} y_{o_1}^0) - 1 \right]^2 + \\ & \left[ (\bar{a}_{s_2} v_{o_2}^x + \bar{b}_{s_2} v_{o_2}^y) t_{s_2}^{o_2} + (\bar{a}_{s_2} x_{o_2}^0 + \bar{b}_{s_2} y_{o_2}^0) - 1 \right]^2 + \\ & \left[ (\bar{a}_{s_3} v_{o_1}^x + \bar{b}_{s_3} v_{o_1}^y) t_{s_3}^{o_1} + (\bar{a}_{s_3} x_{o_1}^0 + \bar{b}_{s_3} y_{o_1}^0) - 1 \right]^2 + \\ & \left[ (\bar{a}_{s_3} v_{o_2}^x + \bar{b}_{s_3} v_{o_2}^y) t_{s_3}^{o_2} + (\bar{a}_{s_3} x_{o_2}^0 + \bar{b}_{s_3} y_{o_2}^0) - 1 \right]^2. \end{aligned} \quad (2)$$

Note that (1) is a nonconvex function of

$$\{(\bar{a}_{s_i}, \bar{b}_{s_i}, v_{o_j}^x, x_{o_j}^0, v_{o_j}^y, y_{o_j}^0); 1 \leq i \leq 3, 1 \leq j \leq 2\},$$

the sensor and object parameters. To determine the minimum of this cost we devise a novel two phase algorithm (the third phase of the algorithm is optional, and corresponds to the final coordinate transformation). Details are given in Sections 4.1 and 4.2.

Section 6.1 compares the results of using the algorithm presented in Section 4 versus a local improvement algorithm started at random points.

### 3. THE MAIN IDEAS

It is evident that finding the global minimum of a nonconvex cost function, such as (1), directly, is a very difficult task. We thus divide the process into two phases. In the first phase, we use the data obtained from only  $m$  sensors and  $n$  objects, where  $m$  and  $n$  are chosen in such a way that (1) can be set exactly to zero, independent of the noise. Solving the resulting equation provides an initial estimate of the parameters. In the second phase, as new data is incorporated to the problem, the sensor and object parameter estimates are refined, using a local improvement algorithm.

We note that since we do not know the “real world” coordinate system we must choose a “custom” system in which to state the equations and thus localize the sensor rotations, and the motions of the objects. Later on we will use the locations of six sensors, if known, to transform the so obtained parameters to the correct representation. This can be done at any point of the algorithm.

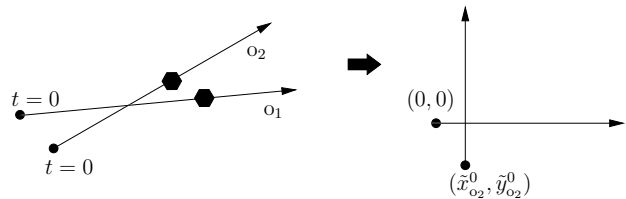
Since we are free to choose our coordinate system, we will choose it in such a way that it simplifies the expressions. In fact, if the coordinate system is not carefully chosen, the resulting equations cannot be solved in closed form. We thus have the task of finding the right coordinate system in which to write the equations, and then finding a procedure to solve them.

We choose the “adaptive” coordinate system in the following way:

1. The motion of the first object will be used to fix the “horizontal” axis, with its position at time  $t = 0$  as the origin, and speed normalized to 1. As we will see in Section 4, this will fix all parameters of  $o_1$  in the custom system.
2. The motion of the second object will be used to fix the “vertical” axis, its speed also normalized to 1. However, since its position at time  $t = 0$  is unknown, two parameters corresponding to  $o_2$  will be undetermined (Section 4).

To determine the number of sensors, and object measurements needed to find the initial estimates, i.e.,  $n$  and  $m$ , we reason in the following way:

1. Each remaining object,  $o_j$ , used in the first phase will add four unknown parameters to the problem:  $v_{o_j}^x$ ,  $x_{o_j}^0$ ,  $v_{o_j}^y$ , and  $y_{o_j}^0$ .
2. Each sensor  $s_i$  included in this phase will add two unknown parameters to define its “line.”
3. On the other hand, the number of data variables obtained from the detection of the first  $n$  objects by  $m$  sensors is  $nm$ .



**Fig. 2.** Custom coordinate system obtained from the trajectories of the first two objects.

Considering that we need at least the same number of data variables as the number of unknown parameters to solve the equations, we need:

$$nm \geq 4(n - 2) + 2 + 2m,$$

which is satisfied by  $m = 6$ , and  $n = 3$ . We thus need at least six sensors and three objects to initialize the system. But we will see in Section 4.1 that the resulting equation is quadratic, and we will need the data from a fourth object to resolve the sign of the root.

### 4. ALGORITHM

In this section we present the estimation algorithm. The organization of the associated computations in-network is discussed in Section 5.

#### 4.1. First phase

Before estimating the sensor directions and object trajectories it is important to fix the coordinate system in which to represent these geometrical objects. With no a priori information, it is impossible to know the “real world” coordinate system. We thus use an intermediate “adaptive” coordinate system given by the trajectories of the first two objects. Figure 2 illustrates this, and the definition of the parameters involved. Later, optionally, if the locations of six sensors are known by some other means, for example, GPS, then they are used to transform the estimates to the correct system.

The first object is used to fix the horizontal axis. The point on the plane at which  $o_1$  was at time  $t = 0$ , represents the origin of the coordinate system. The direction of motion determines the axis, and the scale is given by assuming that the speed of  $o_1$  is 1.

The second object fixes the vertical axis. The direction of motion of  $o_2$  determines the axis, while the scale is given by assuming that its speed is 1. The point at which  $o_2$  was at time  $t = 0$  is unknown. We call this point  $(\tilde{x}_{o_2}^0, \tilde{y}_{o_2}^0)$ . These parameters are unknown even with respect to the adaptive basis and must be estimated as part of the problem.

In our “custom” coordinate system, we know that the line corresponding to sensor  $s_i$  passes through the points

$(t_{s_i}^{o_1}, 0)$  and  $(\tilde{x}_{o_2}^0, \tilde{y}_{o_2}^0 + t_{s_i}^{o_2})$ . Thus, the equation for  $s_i$  in this system is determined as

$$\frac{\tilde{y}_{s_i}}{\tilde{x}_{s_i} - t_{s_i}^{o_1}} = \frac{\tilde{y}_{o_2}^0 + t_{s_i}^{o_2}}{\tilde{x}_{o_2}^0 - t_{s_i}^{o_1}}. \quad (3)$$

Thus, subject only to  $(\tilde{x}_{o_2}^0, \tilde{y}_{o_2}^0)$  being unknown, each sensor line is determined.

Now we turn to the second object. Reordering (3) we obtain

$$(\tilde{x}_{o_2}^0 - t_{s_i}^{o_1})\tilde{y}_{s_i} = (\tilde{y}_{o_2}^0 + t_{s_i}^{o_2})\tilde{x}_{s_i} - t_{s_i}^{o_2}\tilde{y}_{o_2}^0 - t_{s_i}^{o_1}t_{s_i}^{o_2}. \quad (4)$$

Consider now the third object  $o_3$ . Assume that the equation for  $o_3$  in our coordinate system is

$$\begin{aligned} \tilde{x}_{o_3}(t) &= \tilde{v}_{o_3}^x t + \tilde{x}_{o_3}^0, \\ \tilde{y}_{o_3}(t) &= \tilde{v}_{o_3}^y t + \tilde{y}_{o_3}^0. \end{aligned}$$

We know  $o_3$  is detected by sensor  $s_i$  at time  $t_{s_i}^{o_3}$ . Combining this information with (4), we obtain

$$\begin{aligned} (\tilde{x}_{o_2}^0 - t_{s_i}^{o_1})(\tilde{y}_{o_3}^0 + \tilde{v}_{o_3}^y t_{s_i}^{o_3}) = \\ (\tilde{y}_{o_2}^0 + t_{s_i}^{o_2})(\tilde{x}_{o_3}^0 + \tilde{v}_{o_3}^x t_{s_i}^{o_3}) - t_{s_i}^{o_1}\tilde{y}_{o_2}^0 - t_{s_i}^{o_1}t_{s_i}^{o_2}. \end{aligned} \quad (5)$$

Let  $\mathbf{M}$  be a matrix such that its  $i$ -th row is

$$\begin{aligned} \mathbf{M}_{i,*} := & [\tilde{x}_{o_2}^0 - t_{s_i}^{o_1}, t_{s_i}^{o_3}(\tilde{x}_{o_2}^0 - t_{s_i}^{o_2}), -(\tilde{y}_{o_2}^0 + t_{s_i}^{o_2}), \\ & -t_{s_i}^{o_3}(\tilde{y}_{o_2}^0 + t_{s_i}^{o_2}), (t_{s_i}^{o_1}\tilde{y}_{o_2}^0 + t_{s_i}^{o_1}t_{s_i}^{o_2})]. \end{aligned} \quad (6)$$

Likewise, let  $\mathbf{v} := [\tilde{y}_{o_3}^0, \tilde{v}_{o_3}^y, \tilde{x}_{o_3}^0, \tilde{v}_{o_3}^x, 1]^T$ . Then, from (5), we can write the linear system

$$\mathbf{M}\mathbf{v} = 0.$$

For this system to have a nontrivial solution,  $\mathbf{M}$  must be column rank deficient. Let us rewrite  $\mathbf{M}$  in term of its columns. For this, let us first define

$$\begin{aligned} \bar{e} &:= [1, 1, \dots, 1]^T, \\ \bar{T}_{o_1} &:= [t_{s_1}^{o_1}, t_{s_2}^{o_1}, \dots, t_{s_m}^{o_1}]^T, \\ \bar{T}_{o_2} &:= [t_{s_1}^{o_2}, t_{s_2}^{o_2}, \dots, t_{s_m}^{o_2}]^T, \\ \bar{T}_{o_3} &:= [t_{s_1}^{o_3}, t_{s_2}^{o_3}, \dots, t_{s_m}^{o_3}]^T, \\ \bar{T}_{o_1}^{o_2} &:= [t_{s_1}^{o_1}t_{s_1}^{o_2}, t_{s_2}^{o_1}t_{s_2}^{o_2}, \dots, t_{s_m}^{o_1}t_{s_m}^{o_2}]^T, \\ \bar{T}_{o_1}^{o_3} &:= [t_{s_1}^{o_1}t_{s_1}^{o_3}, t_{s_2}^{o_1}t_{s_2}^{o_3}, \dots, t_{s_m}^{o_1}t_{s_m}^{o_3}]^T, \\ \bar{T}_{o_2}^{o_3} &:= [t_{s_1}^{o_2}t_{s_1}^{o_3}, t_{s_2}^{o_2}t_{s_2}^{o_3}, \dots, t_{s_m}^{o_2}t_{s_m}^{o_3}]^T. \end{aligned}$$

With these definitions we can write  $\mathbf{M}$  as

$$\begin{aligned} \mathbf{M} = & [\tilde{x}_{o_2}^0 \bar{e} - \bar{T}_{o_1}, \tilde{x}_{o_2}^0 \bar{T}_{o_3} - \bar{T}_{o_1}^{o_3}, -\tilde{y}_{o_2}^0 \bar{e} - \bar{T}_{o_2}, \\ & -\tilde{y}_{o_2}^0 \bar{T}_{o_3} - \bar{T}_{o_2}^{o_3}, \tilde{y}_{o_2}^0 \bar{T}_{o_1} + \bar{T}_{o_1}^{o_2}] \end{aligned} \quad (7)$$

Since  $\mathbf{M}$  is column rank deficient, there exist real numbers  $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ , such that

$$\begin{aligned} \alpha_1(\tilde{x}_{o_2}^0 \bar{e} - \bar{T}_{o_1}) + \alpha_2(\tilde{x}_{o_2}^0 \bar{T}_{o_3} - \bar{T}_{o_1}^{o_3}) + \\ \alpha_3(-\tilde{y}_{o_2}^0 \bar{e} - \bar{T}_{o_2}) + \alpha_4(-\tilde{y}_{o_2}^0 \bar{T}_{o_3} - \bar{T}_{o_2}^{o_3}) + \\ \alpha_5(\tilde{y}_{o_2}^0 \bar{T}_{o_1} + \bar{T}_{o_1}^{o_2}) = 0. \end{aligned} \quad (8)$$

Collecting terms, and defining

$$\begin{aligned} \bar{\mathbf{M}} &:= [\bar{e}, \bar{T}_{o_1}, \bar{T}_{o_3}, \bar{T}_{o_2}, \bar{T}_{o_1}^{o_3}, \bar{T}_{o_2}^{o_3}, \bar{T}_{o_1}^{o_2}], \\ \bar{\mathbf{v}} &:= [\alpha_1 \tilde{x}_{o_2}^0 - \alpha_3 \tilde{y}_{o_2}^0, \alpha_5 \tilde{y}_{o_2}^0 - \alpha_1, \alpha_2 \tilde{x}_{o_2}^0 - \alpha_4 \tilde{y}_{o_2}^0, \\ & -\alpha_3, -\alpha_2, -\alpha_2, -\alpha_4, \alpha_5]^T, \end{aligned}$$

we can rewrite (8) as

$$\bar{\mathbf{M}}\bar{\mathbf{v}} = 0. \quad (9)$$

Let  $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]^T$  be the solution to (9), with  $\alpha_5 = 1$ . Then

$$\begin{aligned} \alpha_1 \tilde{x}_{o_2}^0 - \alpha_3 \tilde{y}_{o_2}^0 &= \theta_1, \\ \alpha_5 \tilde{y}_{o_2}^0 - \alpha_1 &= \theta_2, \\ \alpha_2 \tilde{x}_{o_2}^0 - \alpha_4 \tilde{y}_{o_2}^0 &= \theta_3, \\ -\alpha_3 &= \theta_4, \\ -\alpha_2 &= \theta_5, \\ -\alpha_4 &= \theta_6, \\ \alpha_5 &= 1. \end{aligned}$$

Solving this nonlinear system one obtains

$$\begin{aligned} \tilde{x}_{o_2}^0 = & \frac{\alpha_5 \theta_3 + \alpha_4 \theta_2 + \alpha_2 \alpha_3 \pm \\ & \sqrt{(\alpha_5 \theta_3 + \alpha_4 \theta_2 + \alpha_2 \alpha_3)^2 + 4\alpha_2 \alpha_5 (\alpha_4 \theta_1 - \alpha_3 \theta_3)}}{2\alpha_2 \alpha_5}. \end{aligned} \quad (10)$$

To resolve the sign in (10) we make use of the data provided by the fourth object  $o_4$ . We simply choose the sign that best explains the detection times  $t_{s_i}^{o_4}$ .

Once the value of  $\tilde{x}_{o_2}^0$  is known, the rest of the parameters can be easily computed.

## 4.2. Second phase

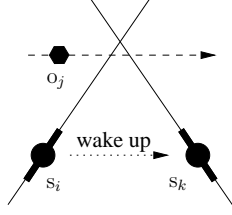
Once the parameters for the first four particles and six sensors have been estimated, most sensors go to sleep. A few sentinel sensors stay awake and sensing. When a sentinel sensor detects an object, it wakes up the complete sensor network. All sensors then wait for the object and register the time at which they detect it. It is important to note that some sensors will not detect a given object. This is shown in Figure 3, in which  $s_i$  wakes up  $s_k$  too late for it to detect  $o_j$ .

Each sensor has at most one detection time for the new object. To form an estimate of the trajectory of this object, at least four measurements are necessary. To gather this information, sensors share their measurements (if they have one), and collect measurements from other nodes. This process is completely asynchronous and unsupervised.

The obtained data are used to refine the estimates of all parameters.

To organize the computations, for each node  $s_i$  we define a matrix  $\Omega^{s_i}$ , such that

$$\Omega_{k,l}^{s_i} := \begin{cases} 1 & \text{if } s_i \text{ knows } t_{s_k}^{o_l}, \\ 0 & \text{otherwise.} \end{cases}$$



**Fig. 3.** Some sensors do not detect some of the objects. In this case  $s_k$  wakes up too late to detect  $o_j$ .

	$s_1$	$s_2$	$s_3$	$s_4$	$s_5$	$s_6$	$s_7$	$s_8$	$s_9$
$o_1$	1	1	1	1	1	1	1		
$o_2$	1	1	1	1	1	1			1
$o_3$	1	1	1	1	1	1	1		
$o_4$	1	1	1	1	1	1			
$o_5$		1		1		1	1		1
$o_6$					1	1		1	

**Fig. 4.** Example of a matrix  $\Omega^{s_i}$  indicating the measurements known to  $s_i$ .

An example matrix  $\Omega^{s_i}$  is shown in Figure 4.

For each  $s_k$ , and object  $o_l$ , let  $\mathcal{O}_{s_k}^{s_i}$  and  $\mathcal{S}_{o_l}^{s_i}$  be defined as

$$\begin{aligned} \mathcal{O}_{s_k}^{s_i} &:= \{l | \Omega_{k,l}^{s_i} = 1\}, \\ \mathcal{S}_{o_l}^{s_i} &:= \{k | \Omega_{k,l}^{s_i} = 1\}. \end{aligned}$$

The cost at  $s_i$  is then given by

$$J_{s_i} = \sum_k \sum_{l \in \mathcal{O}_{s_k}^{s_i}} [\bar{a}_{s_k}^{s_i} v_{o_l}^{x,s_i} t_{s_k}^{o_l} + \bar{a}_{s_k}^{s_i} x_{o_l}^{0,s_i} + \bar{b}_{s_k}^{s_i} v_{o_l}^{y,s_i} t_{s_k}^{o_l} + \bar{b}_{s_k}^{s_i} y_{o_l}^{0,s_i} - 1]^2. \quad (11)$$

where  $\bar{a}_{s_k}^{s_i}$ ,  $\bar{b}_{s_k}^{s_i}$ ,  $v_{o_l}^{x,s_i}$ ,  $x_{o_l}^{0,s_i}$ ,  $v_{o_l}^{y,s_i}$ , and  $y_{o_l}^{0,s_i}$  are the estimated parameters at  $s_i$ . We use Newton's method to minimize  $J_{s_i}$ .

Let us first define

$$\begin{aligned} A_{s_k,o_l}^{s_i} &:= v_{o_l}^{x,s_i} t_{s_k}^{o_l} + x_{o_l}^{0,s_i}, \\ B_{s_k,o_l}^{s_i} &:= v_{o_l}^{y,s_i} t_{s_k}^{o_l} + y_{o_l}^{0,s_i}, \\ C_{o_k,o_l}^{s_i} &:= \bar{a}_{s_k}^{s_i}, \\ D_{o_k,o_l}^{s_i} &:= \bar{a}_{s_k}^{s_i} t_{s_k}^{o_l}, \\ E_{o_k,o_l}^{s_i} &:= \bar{b}_{s_k}^{s_i}, \\ F_{o_k,o_l}^{s_i} &:= \bar{b}_{s_k}^{s_i} t_{s_k}^{o_l}. \end{aligned}$$

With these definitions we can rewrite (11) as

$$\begin{aligned} J_{s_i} &= \sum_k \sum_{l \in \mathcal{O}_{s_k}^{s_i}} [A_{s_k,o_l}^{s_i} \bar{a}_{s_k}^{s_i} + B_{s_k,o_l}^{s_i} \bar{b}_{s_k}^{s_i} - 1]^2 \\ &= \sum_l \sum_{k \in \mathcal{S}_{o_l}^{s_i}} [D_{o_k,o_l}^{s_i} v_{o_l}^{x,s_i} + C_{o_k,o_l}^{s_i} x_{o_l}^{0,s_i} + F_{o_k,o_l}^{s_i} v_{o_l}^{y,s_i} + E_{o_k,o_l}^{s_i} y_{o_l}^{0,s_i} - 1]^2. \end{aligned} \quad (12)$$

To simplify the expressions, let us also define

$$\begin{aligned} \mathbf{v}_{s_k}^{s_i} &:= [\bar{a}_{s_k}^{s_i}, \bar{b}_{s_k}^{s_i}]^T, \\ \mathbf{g}_{s_k}^{s_i} &:= \begin{bmatrix} \sum_{l \in \mathcal{O}_{s_k}^{s_i}} (A_{s_k,o_l}^{s_i} \bar{a}_{s_k}^{s_i} + B_{s_k,o_l}^{s_i} \bar{b}_{s_k}^{s_i} - 1) A_{s_k,o_l}^{s_i}, \\ \sum_{l \in \mathcal{O}_{s_k}^{s_i}} (A_{s_k,o_l}^{s_i} \bar{a}_{s_k}^{s_i} + B_{s_k,o_l}^{s_i} \bar{b}_{s_k}^{s_i} - 1) B_{s_k,o_l}^{s_i} \end{bmatrix}^T, \\ \mathbf{H}_{s_k}^{s_i} &:= \begin{bmatrix} \sum_{l \in \mathcal{O}_{s_k}^{s_i}} (A_{s_k,o_l}^{s_i})^2 & \sum_{l \in \mathcal{O}_{s_k}^{s_i}} A_{s_k,o_l}^{s_i} B_{s_k,o_l}^{s_i} \\ \sum_{l \in \mathcal{O}_{s_k}^{s_i}} B_{s_k,o_l}^{s_i} A_{s_k,o_l}^{s_i} & \sum_{l \in \mathcal{O}_{s_k}^{s_i}} (B_{s_k,o_l}^{s_i})^2 \end{bmatrix}. \end{aligned}$$

Applying Newton's method to (11) with respect to  $\bar{a}_{s_k}$  and  $\bar{b}_{s_k}$ , we obtain the recursion

$$\mathbf{v}_{s_k}^{s_i} \leftarrow \mathbf{v}_{s_k}^{s_i} - (\mathbf{H}_{s_k}^{s_i})^{-1} \mathbf{g}_{s_k}^{s_i}.$$

Similar expressions are obtained by applying Newton's method to (11), with respect to  $v_{o_l}^{x,s_i}$ ,  $x_{o_l}^{0,s_i}$ ,  $v_{o_l}^{y,s_i}$ , and  $y_{o_l}^{0,s_i}$ .

## 5. DISTRIBUTED IMPLEMENTATION

After obtaining the initial estimates for the first four particles and six sensors, each sensor waits for more data to arrive. With the collected data,  $s_i$  fills out the matrix  $\Omega^{s_i}$ , and performs Newton's method on the parameters for which there is enough data. In other words  $s_i$  moves through the rows of  $\Omega^{s_i}$ , updating the parameter estimates for the objects  $o_l$  for which there are at least four values of  $t_{s_k}^{o_l}$  known. Likewise,  $s_i$  goes through the columns of  $\Omega^{s_i}$ , updating the parameters of the line of sight for the sensors  $s_k$  for which there are at least two known values of  $t_{s_k}^{o_l}$ .

As each new datum arrives, it can be easily incorporated into the algorithm by incorporating it into the update rules for the sensor and object parameters, as a new term in the definition of the gradient and Hessian of the cost function.

It is important to note that  $s_i$  critically needs the initial estimates found in Section 4.1 to begin the refinement process. If no estimates are known, the refinement process cannot be started. The comparison with random initialization in Section 6.1 illustrates this.

### 5.1. Organization

To organize the computations we look for inspiration to graphical models (e.g., [4, 5]), and divide the operations into computation of messages and beliefs. A message computed by one node  $s_i$  is physically transmitted to another node  $s_j$ . The belief at  $s_i$  is the estimate of the result of the computation, given the messages that  $s_i$  has received. This separation simplifies the in-network organization, because then

the specification of the algorithm only requires the specification of the update rules for the messages and beliefs.

We will design the algorithms in such a way that the computations satisfy the “increasing correctness” property (see [3]), which means that at each time, each node will have an estimate of the solution of the computation which is correct for the data that has been fed into the algorithm up to that time.

## 5.2. Message passing

As mentioned before, a message passing algorithm is specified by giving update rules for the messages and the beliefs. It is remarkable that in most cases the update rules for messages and beliefs are very similar. In a general sense, the update rules specify how the information conveyed by the messages received by a node is combined with the local information known by the sensor, to form outgoing messages or beliefs.

The message from node  $s_i$  to  $s_j$  will be denoted by  $\mathbf{m}_{s_i,s_j}$ . To distinguish messages for different algorithms we will use superscripts (e.g.,  $\mathbf{m}_{s_i,s_j}^{\text{opt}}$ ). The belief at node  $s_i$  will be denoted  $\mathbf{b}_{s_i}$ , or  $\mathbf{b}_{s_i}^{\text{opt}}$ . The local information at node  $s_i$  will be denoted  $\mathbf{D}_{s_i}$ , or  $\mathbf{D}_{s_i}^{\text{opt}}$ . The update rules will take the general form:

$$\begin{aligned}\mathbf{m}_{s_i,s_j} &\leftarrow f\left(\{\mathbf{m}_{s_k,s_i}\}_{k \in \mathbf{N}_{i,j}^m}, \mathbf{D}_{s_i}\right), \\ \mathbf{b}_{s_i} &\leftarrow g\left(\{\mathbf{m}_{s_k,s_i}\}_{k \in \mathbf{N}_i^b}, \mathbf{D}_{s_i}\right),\end{aligned}$$

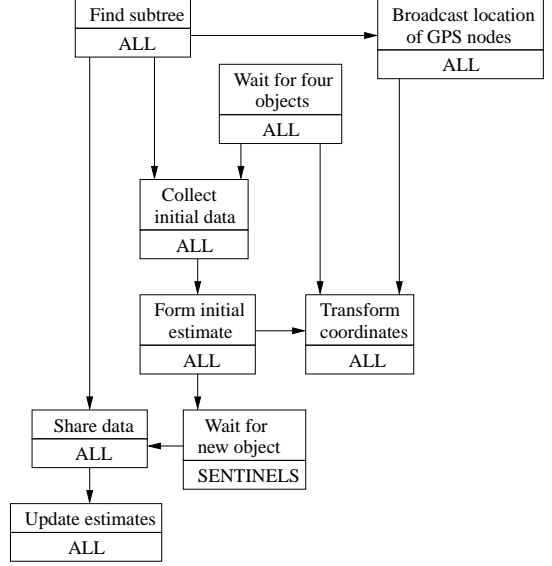
where  $\mathbf{N}_{i,j}^m, \mathbf{N}_{i,j}^b \subset \mathbf{N}_{s_i}$ , and  $\mathbf{N}_{s_i}$  denotes the set of neighbors of  $s_i$  in the connectivity graph of the network. In general, we will use  $\mathbf{N}_{i,j}^m := \mathbf{N}_{s_i}^T \setminus \{j\}$ , and  $\mathbf{N}_i^b := \mathbf{N}_{s_i}^T$ . Here  $\mathbf{N}_{i,j}^T$  denotes the set of neighbors of  $s_i$  in a spanning subtree of the connectivity graph. The update rules are defined by the functions  $f$  and  $g$ .

Let us consider, as an example, the problem of computing the number of nodes in the network. We can achieve this with the following update rules for messages and beliefs:

$$\begin{aligned}\mathbf{m}_{s_i,s_j}^{\text{count}} &\leftarrow \sum_{k \in \mathbf{N}_{i,j}^m} \mathbf{m}_{s_k,s_i}^{\text{count}} + 1, \\ \mathbf{b}_{s_i}^{\text{count}} &\leftarrow \sum_{k \in \mathbf{N}_i^b} \mathbf{m}_{s_k,s_i}^{\text{count}} + 1.\end{aligned}$$

It is easy to see that after a finite number of steps, the belief at each node will be exactly the number of nodes in the network. At each time the belief at node  $s_i$  is the correct count of nodes in a neighborhood of  $s_i$ . This neighborhood grows with time. This algorithm is increasingly correct.

A second example is an implementation of the distributed Bellman-Ford algorithm. In this case we have that  $\mathbf{m}_{s_i,s_j}^{\text{BF}} =$



**Fig. 5.** Dependence diagram indicating tasks and sensors that must perform them.

$(\mathbf{d}_{s_i,s_j}, \mathbf{p}_{s_i,s_j})$ , and  $\mathbf{b}_{s_i}^{\text{BF}} = (\mathbf{d}_{s_i}, \mathbf{p}_{s_i})$  with update rules:

$$\begin{aligned}\mathbf{d}_{s_i,s_j} &\leftarrow \min_{k \in \mathbf{N}_{i,j}^m} \{\mathbf{d}_{s_k,s_i}\} + 1, \\ \mathbf{p}_{s_i,s_j} &\leftarrow \operatorname{argmin}_{k \in \mathbf{N}_{i,j}^m} \{\mathbf{d}_{s_k,s_i}\}, \\ \mathbf{d}_{s_i} &\leftarrow \min_{k \in \mathbf{N}_i^b} \{\mathbf{d}_{s_k,s_i}\} + 1, \\ \mathbf{p}_{s_i} &\leftarrow \operatorname{argmin}_{k \in \mathbf{N}_i^b} \{\mathbf{d}_{s_k,s_i}\}.\end{aligned}$$

Here  $\mathbf{d}_{s_i,s_j}$  is the estimated minimum number of hops to the source that  $s_i$  sends to  $s_j$ , and  $\mathbf{p}_{s_i,s_j}$  the current parent of  $s_i$ .

## 5.3. Distributed algorithm

The in-network organization of the algorithm amounts to answering the question: Who does what, and when? To simplify the design of the algorithm, we divide it into *tasks*. Each task is performed by a subset of the sensors. Some tasks can be performed independently, while others are dependent. We use a dependency diagram to explicitly show the dependencies (Figure 5).

To indicate which sensors participate in each task, we divide the set of sensors into subsets. Specifically, in our case, we distinguish four meaningful subsets, namely, all sensors, which we denote by “ALL,” sensors with GPS, which we call “GPS,” sentinel sensors, which we denote by “SENTINELS,” and finally the root node of the spanning subtree (“ROOT”). The tasks into which our algorithm can be divided, and the sensors that participate are:

1. Find spanning subtree of connectivity graph Participating sensors: ALL. Behavior of ROOT node is different from other nodes.
2. Wait for four objects to cross  $\mathcal{R}$  Participating sensors: ALL.
3. Collect data from other sensors. Participating sensors: ALL.
4. Compute initial parameter estimates in custom coordinate system. Participating sensors: ALL.
5. Broadcast location of GPS sensors Participating sensors: ALL. Behavior of GPS sensors is different from other sensors.
6. Compute coordinate transformation. ALL sensors.
7. Wait for new particles to arrive. SENTINEL sensors.
8. Share data. ALL.
9. Update estimates. ALL.

We will present the pseudocode for some of the tasks. The code associated with a message passing process can be very long. To avoid confusing code, we create an “instruction” that will allow us to define messages and beliefs, without having to explicitly write out the details. We call this instruction “message.”

The syntax for this instruction is:

```
message(index1, index2; set)
  {running_update_rule;}
  {final_update_rule;}
```

Note that we have omitted the specification of the name of the message. This instruction works as a “for loop,” in which the `running_update_rule` is executed for each `index1` in `set`, and `index2` in `set`, except `index1`. The argument `set` denotes a set of indices.

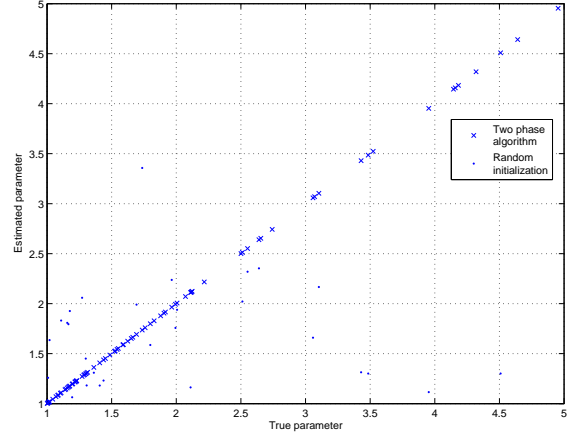
The rule `final_update_rule` is executed for each `index1` in `set`.

The belief is computed using the same update rules.

For example, the pseudocode for a simple data sharing task is

```
times= %set of detection times
data=empty_set;
message(j,k; Ntree){
  data(j)=union(data(j),data(k));
}{
  data(j)=union(data(j),times);
}
```

This code simply collects the incoming data sets, adds to it the local measurements, and produces an output data set to be transmitted.



**Fig. 6.** Comparison of two phase algorithm, to local improvement with random initialization.

## 6. SIMULATIONS

To study the behavior of the proposed estimation algorithm, we have performed simulations. Only the numerical algorithms, not the underlying communication protocols (MAC, routing) were simulated.

### 6.1. Optimization algorithm

In this section we compare the performance of the proposed two phase optimization algorithm to a local improvement algorithm started at random positions. The improvement algorithm used is the same as used in the refinement phase in Section 4.2. Thus this comparison illustrates the contribution of phase one of the algorithm which provides the initial estimate. The comparison of the algorithms is based on the value of  $\bar{a}_{s_1}$ .

The algorithm was executed a hundred times with different setups (values of  $a$ ,  $b$ ,  $v^x$ ,  $x^0$ ,  $v^y$ , and  $y^0$ ). In each case the estimated value of  $\bar{a}_{s_1}$  was recorded after five objects (the first four, and one more) had been detected. The parameters were also estimated using the same refinement algorithm presented in Section 4.2, but initialized with a random guess. For each run, ten different initializations were used. No noise was considered.

Figure 6 compares the true and estimated values of  $\bar{a}_{s_1}$ . The values of the estimated parameters are plotted versus the true values. Crosses show the parameters estimated using the two phase algorithm, while dots show the values estimated using random initialization. It can be seen that estimating the correct parameter values is a difficult problem that the two phase algorithm successfully overcomes.

## 6.2. Complete algorithm

The setup of the simulations of the algorithm is as follows:

1. One hundred sensors are located on a “Gaussian perturbed grid. That is, the location of each sensor  $s_i$  is  $(\bar{x}_{s_i}, \bar{y}_{s_i})$ , where  $\bar{x}_{s_i}$ , and  $\bar{y}_{s_i}$  are obtained as integers on a regular squared grid, plus a Gaussian random variable with variance 0.09. The angle of each sensor line is chosen uniformly in  $[0, 2\pi]$ .
2. Each sensor has exactly six neighbors in the network.
3. Object speeds and origin  $(v^x, x^0, v^y, y^0)$  are chosen uniformly in  $[0, 1]$ .
4. All sensors detect all objects. In other words, in this case, we do not simulate the effect of sleeping nodes.
5. After four objects have been detected, each node collects the measurements from its neighbors and forms initial parameter estimates. Every time a new object is detected, three steps of the simple data sharing message passing algorithm described in Section 5.3 are executed. Each node then incorporates the new data into the refinement algorithm described in Section 4.2.

To study how the robustness of the algorithm is affected by the number of detected objects, in Figure 7 we show the simulation of the detection of 20 objects (after the first four), with the first  $q$  measured with zero noise, and the remaining  $20 - q$ , with noise variance  $\nu_{s_i, o_j} = 0.1$ . This is repeated for  $q = 1$  to  $q = 20$ . The percentage error in  $\bar{a}_{s_{50}}$  is shown.

One can see that errors are small as long as the noise is zero. In fact, since the initial estimates in this case are exact, the parameter estimates are exact, for the first  $q + 4$  objects. When noise is added to the measurements, the errors increase. When the noise is added later in time, the increase in error is smaller, compared to adding noise earlier, again illustrating the value of phase one. Thus, as can be expected, the robustness of the algorithm increases with the number of objects detected.

## 7. CONCLUSIONS

We have studied the application of directional sensors to the detection and tracking of moving objects. The estimation of the object trajectory is a nonconvex optimization problem. To overcome the difficulty of local minima, a two phase optimization algorithm was designed. To organize the computations in-network, we use ideas from graphical models.

The importance of the first phase of the algorithm was illustrated by comparing it to the performance of a local improvement algorithm initialized randomly. In Figure 6, we

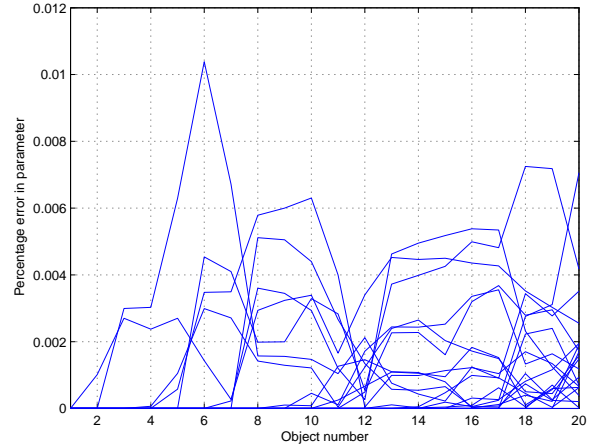


Fig. 7. Behavior of percentage error in  $\bar{a}_{s_{50}}$ .

can see that without the use of the initial solution, the performance of the algorithm is disastrous.

We also discussed the possibility of using message passing type algorithms to simplify the organization of computations in-network.

## 8. REFERENCES

- [1] A. Giridhar and P. R. Kumar, “Data fusion over sensor networks: Computing and communicating functions of measurements,” *To appear in Journal on Selected Areas in Communications*, December 2003.
- [2] M. Rabbat and R. Nowak, “Distributed optimization in sensor networks,” in *Third international conference on information processing in sensor networks (IPSN 2004)*, Berkeley, CA, 2004.
- [3] Kurt Plarre and P. R. Kumar, “Increasingly correct message algorithms for heat source detection in sensor networks,” in *Proceedings of The First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON 2004)*, Santa Clara, CA, 2004.
- [4] Michael Jordan and Christopher Bishop, *An introduction to graphical models*, Preprint, October 2001.
- [5] Judea Pearl, *Probabilistic reasoning in intelligent systems. Networks of plausible inference*, Morgan Kaufmann, 1988.