

A Distributed Cooperative Target Tracking with Binary Sensor Networks

Zijian Wang, Eyuphan Bulut, and Boleslaw K. Szymanski

Department of Computer Science and Center for Pervasive Computing and Networking
Rensselaer Polytechnic Institute
Troy, NY 12180 USA
{wangz, bulute, szymansk}@cs.rpi.edu

Abstract—Target tracking is a typical and important cooperative sensing application of wireless sensor networks. We study it in its most basic form, assuming the binary sensing model in which each sensor can return only 1-bit information regarding target's presence or absence within its sensing range. A novel, real-time and distributed target tracking algorithm is proposed. The algorithm reduces the uncertainty of the target location from a two-dimensional area into a one-dimensional arc and estimates the target velocity and trajectory in a distributed and asynchronous manner. Extensive simulations show that our algorithm achieves good performance by yielding highly accurate estimates of the target's location, velocity and trajectory.

Keywords—cooperative sensing; target tracking; binary sensor networks; distributed algorithms

I. INTRODUCTION

Wireless sensor networks composed of miniature devices that integrate physical sensing, data processing and communication capabilities present great opportunities for a wide range of applications [1]. Among them, target tracking is a representative and important application that usually requires a cooperative sensing processing to achieve good results [2,3]. One of the fundamental studies of target tracking focuses on networks composed of sensor nodes capable of the most elementary binary sensing that provides just one bit of information about the target: whether it is present within the sensing range or not. These so-called binary sensor networks constitute the simplest type of sensor networks that can be used for target tracking.

Under the binary sensing model, individual sensors of the network have very limited information regarding the target status, which means that it is necessary to use cooperation to accurately track the target. A number of approaches to this problem have been proposed in recent years. The algorithms presented in [4,5] first route the binary information to a central node and then the central node applies particle filters on information gathered from all sensors to update the target's track. But particle filters are expensive to compute and transmitting data from each node to a central one is very costly in terms of the energy needed for communication for any non-trivial size network. In [6], each point on the target's path is estimated by the weighted average of the detecting sensors' locations. Then, a line that fits best this point and the points on the trajectory established in the recent past is used as the target

trajectory. Kim et al. [7] improve the weight calculation for each sensor node that detected the target and use the estimated velocity to get the estimated target location. However, these two methods require time synchronization of the entire network and assume that the target moves at a constant velocity on a linear trajectory. Furthermore, they only use positions of the sensor nodes that detected the target. Actually, the absence of detection can also provide information that can be used to improve the tracking accuracy. In [8], both the presence and absence of the target within the node's sensing range are used to form local regions that the target had to pass. These regions are bounded by the intersecting arcs of the circles defined by the sensing ranges of the relevant nodes. The trajectory is estimated as a piecewise linear path with the fewest number of linear segments that traverses all the regions in the order in which the target passed them. However, the algorithm is centralized and complex to compute. It also requires a designated tracker node to fuse data. Additionally, the designated node has to accumulate information from tracking sensors to form all regions needed to compute the estimated trajectory, which means that the tracking is not real-time but delayed.

In this paper, we propose a novel distributed target tracking algorithm using binary sensor networks. Each active node computes the target's location, velocity and trajectory locally but uses cooperation to collect the sensing bits of its neighbors. Furthermore, the algorithm tracks the target in real-time, does not require time synchronization between sensor nodes and can be applied to target moving in random directions and with varied velocities.

The remainder of the paper is organized as follows. We describe the network model and our assumptions in Section II. In Section III, we introduce our distributed target tracking algorithm using cooperative binary sensor networks. Section IV presents the simulation results. Finally, we provide conclusions in section V.

II. NETWORK MODEL AND ASSUMPTIONS

The sensor network comprises N nodes placed randomly with uniform distribution over a finite, two-dimensional planar region to be monitored. Each node has a unique identifier and its sensing region forms a disk centered at the node and bounded by a circle defined by the sensing range R . The union of sensing regions of all network nodes guarantees redundant

coverage of the region to be monitored. For simplicity, we assume that the sensing range of each node is identical across the network. However, our algorithm also applies when sensing range vary from node to node. Each node generates one bit of information (“1” for target’s presence and “0” for its absence) only at the moment that the target enters or exits the sensing range of this node. Otherwise, we get no other information about the location, velocity, or other attributes of the target. A node that has not detected change in the absence or presence of the target within its sensing range does not transmit to save energy and bandwidth. Each time a new bit of information is generated, the node communicates it to its neighbors that are defined as nodes whose sensing range intersects its sensing range (depending on the relation between the sensing and communication radii, this may require more than one hop). Henceforward, we use the term neighbor in this specific sense. A node knows its location and the locations of its neighbors (possibly through communication at the network deployment stage, not discussed here). Each node has its own local timer and can time stamp sent or received messages. Additionally, we assume that the target moves with velocity that is low relative to the node’s sensing frequency. Consequently, time of discovery of the change in the target’s presence within the node’s sensing range differs negligibly little from the time of the target moves within or out of this range.

III. A COOPERATIVE TRACKING ALGORITHM

A. Basic Idea

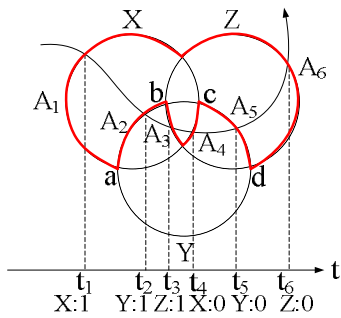


Figure 1. Illustration of the basic idea behind the algorithm

To illustrate our basic idea, we use an example from Fig. 1, which shows a target moving through an area covered by three nodes. Initially, the target is outside of the sensing ranges of all three nodes. Later, it moves within the sensing range of node X at the system time t_1 , and then sensing ranges of nodes Y at time t_2 and Z at time t_3 . Finally, it leaves sensing ranges of nodes X, Y and Z, in that sequence, at times t_4 , t_5 , t_6 . According to the model described in the previous section, each node will generate a bit “1” at the time of first sensing the target’s presence and later a bit “0” at the time of first lacking to sense its presence which correspond to the times at which the target enters and then exits sensing range of the node. Consequently, at the transition time t_j , the target must be on arc A_j which is a part of the border circle of the sensing range of the node reporting the bit information and which can be determined cooperatively from presence and absence bits of neighbors of that node. Let’s consider arc A_2 defined at time t_2 as an

example. At time t_2 , node Y senses the target presence within its sensing range for the first time, so the arc is a part of the sensing range border circle of node Y. At that time, node Y knows that the target is within the sensing range of node X, so the target must be on arc “abc”. Node Y knows also that the target is not within the sensing range of node Z, so the target can not be on arc “bcd”. Hence, node Y concludes that the target must be on arc A_2 . It is important to observe that, by using this method, the two-dimensional uncertainty of the target’s location on the plane is reduced to a one-dimensional uncertainty within the circle section. Shorter this circle section is, smaller the uncertainty becomes.

B. Tracking Algorithm

At the network deployment stage, each node initializes the status list of its neighbors to “0”’s. Each time a node receives one-bit information from a neighbor, it updates the status list. At the moment at which the node discovers the change in the target’s presence within its sensing range, it identifies the arc of its sensing range border circle that the target is crossing. The target location is estimated as the middle point of the corresponding arc and broadcast to neighbors. Two relatively accurate estimates of target location combined with the difference of local times at which those estimates were made are used for distributed computation of the target velocity. A weighted line fitting method is used to find a line, approximating a fragment of the target trajectory, that best fits the estimated target locations.

1) Initialization and information update

In the initialize procedure, each node establishes a list of its neighbors. Each element of the list stores the following information: neighbor node identifier, intersection points of the sensing circles of the node and its neighbor, an angle corresponding to the arc defined by these intersection points and one-bit information generated by the neighbor, initialized to “0”. Upon receiving one-bit information from a neighbor, the node updates the corresponding entry in the list.

2) Location estimate

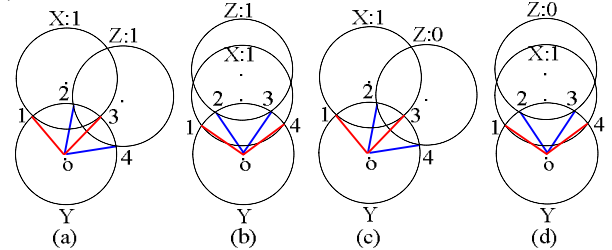


Figure 2. Instances of angle combinations

We combine all angles corresponding to arcs defined by the neighbor list to determine the arc that the target is crossing. The four instances of this process are shown in Fig. 2. If the neighbors both generated bits equal to “1”, the corresponding central angles are combined by “&” operation that returns the intersection of these two angles. As shown in Fig. 2(a), the common angle of $\angle 1o3$ and $\angle 2o4$ is $\angle 2o3$, so the node Y estimates the target location as the middle point of arc “23” when it senses that the target just moved within its sensing

range. One special instance is shown in Fig. 2(b), where the common angle is just one of the two angles. If one neighbor status is set to “1” while the other is set to “0”, the corresponding central angles are combined with “-” operation that returns the angle formed from the first angle by excluding from it the second. For example, in Fig. 2(c) $\angle_{1o3} - \angle_{2o4}$ is equal to \angle_{1o2} . In a special case shown in Fig. 2(d), the result may consist of two angles, \angle_{1o2} and \angle_{3o4} . The correct angle in this case is chosen by considering the recent estimate of the target location.

Let FA be the sought arc’s central angle initialized to 2π (the entire circle of the sensing border of a node). Let IN be the set of neighbor nodes with status set to “1” and let OUT be the set of neighbor nodes with status set to “0”. Then, the final angle whose corresponding arc is the one that the target is crossing can be expressed as:

$$FA = FA \& \underset{i \in IN}{angle_i} - \underset{j \in OUT}{angle_j} \quad (1)$$

where $angle_i$ is the central angle corresponding to neighbor i .

3) Velocity estimate

We use a distributed, asynchronous algorithm to estimate the target velocity. As shown in Fig. 3, three nodes X, Y and Z work in asynchronous time. At time t_{Y1} on node Y’s local clock, node Y senses target’s presence within its sensing range for the first time and generates a bit “1” message. The estimated location of the target is also included in this message to save energy and bandwidth. Since the elapsed time of radio transmission is negligible, node Z receives this message at time t_{Z1} on its local clock. Node Z will also receive the message from node X at time t_{Z2} . Then, node Z can use the time difference $t_{Z1}-t_{Z2}$ and the difference of locations reported in these two messages to estimate the target velocity. To estimate velocity accurately, only location estimates with relatively high accuracy are used, those are locations at the middle points of the short arcs.

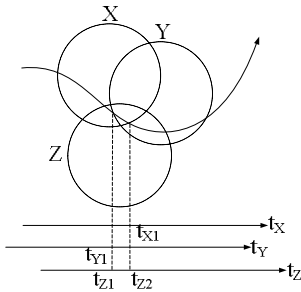


Figure 3. Velocity estimation

4) Trajectory estimate

A weighted line fitting method is used to get the target trajectory and the weight of each estimate is defined as:

$$w = |circle|/|arc| \quad (2)$$

where $|arc|$ is the length of corresponding arc whose middle point is the estimated target location and $|circle|$ is the length of the sensing range border circle. Each node finds the line (or two or more line sections if the target turns around) that best fits these weighted estimated locations. This line when expressed as $y = a \cdot x + b$ minimizes the metric Q defined as:

$$Q = \sum_{i \in E} w_i (y_i - a \cdot x_i - b)^2 \quad (3)$$

where $E = [(y_0, x_0), \dots, (y_i, x_i), \dots, (y_k, x_k)]$ is a list of the estimated target locations for which the line is fitted.

IV. SIMULATION

To evaluate our approach, we have performed extensive simulations in which N wireless nodes are randomly placed on a 800×800 unit square in a two-dimensional simulated plane to track a single mobile target. The number of nodes and the node’s sensing range are varying simulation parameters.

A. Location Estimate

The first metric that we consider is the location estimation error, measured as the ratio of the distance between the estimated and real target locations to the sensing range. Several types of trajectories have been considered: linear, circular, and a piece-wise linear trajectory with random turns.

Fig. 4 shows the typical example of estimated location points for these three kinds of trajectories. The sample network is composed of 600 nodes with sensing range of 150 units.

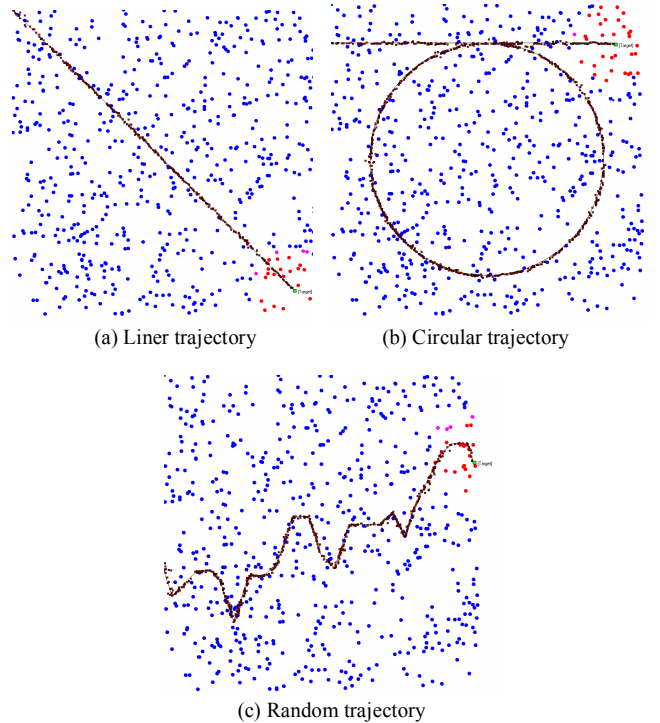
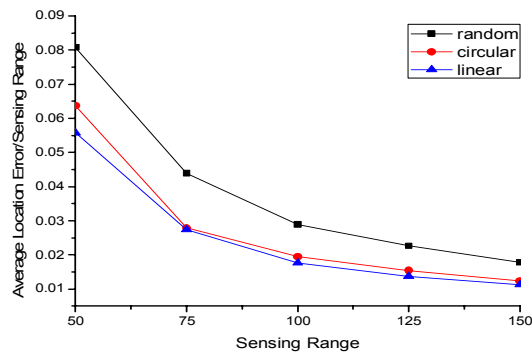
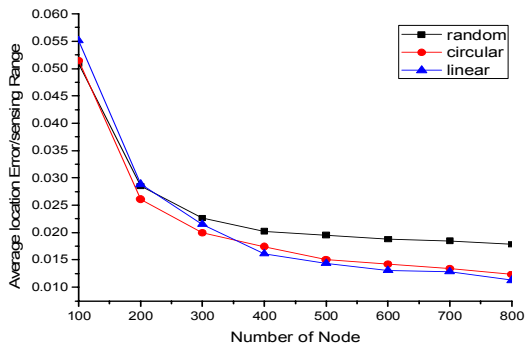


Figure 4. Examples of location estimation

When evaluating the impact of simulation parameters on the location estimation accuracy, we first kept the number of nodes fixed at 800 and varied the sensing range from 50 to 150 units. Then, we kept the sensing range fixed at 150 units and varied the number of nodes from 100 to 800. The plots in Fig. 5 show the average error in location for the configuration and the three trajectories described above. We observe that the localization error decreases inversely to the sensing range and to the number of nodes.



(a) Impact of sensing range



(b) Impact of the number of nodes

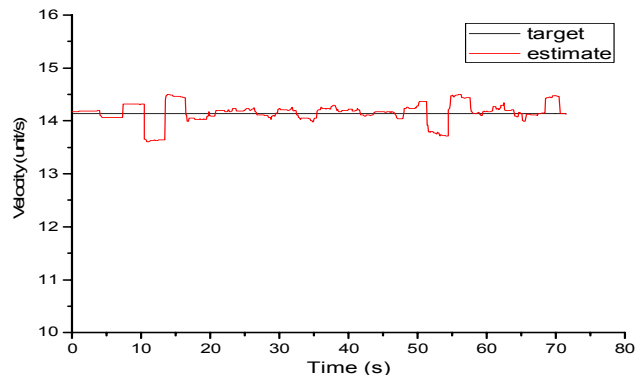
Figure 5. Location estimate accuracy

For the random trajectory, we made the target move at a relatively high velocity (up to four times higher than in the case of linear and circular trajectories). This change affects the position estimation quality in two ways. First, each node senses for target with constant frequency, so the error of sensing the target’s distance linearly increases with the target’s speed. Second, more frequent reporting of target position may create traffic congestion where the target is sensed and therefore delay sensing message delivery and processing. Because of this delay, each node may have incomplete neighbor information when it estimated the target position. Consequently, the error for random trajectory is higher than for the other two. Yet, even in this case, our location error (below 10%) with random placement of nodes is smaller than the results presented in [7] (above 15%). Likewise, in relatively sparse networks, with the total of 200-400 nodes and 100 units sensing range, our algorithm performed well. It is also very austere in sending messages. A node generates a single message only if there is a change in target’s presence within its sensing range. Hence, with moderately fast targets, it is unlikely to cause congestion delaying packet delivery. Thus,

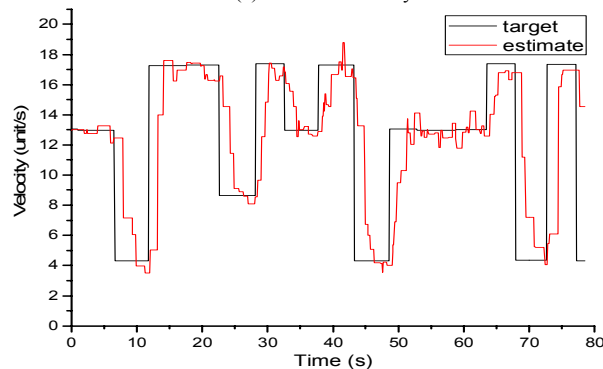
in the following, we use a relatively dense network with the total of 800 nodes and 150 unit sensing range.

B. Velocity Estimate

We tested the performance of velocity estimation under the configuration of 800 nodes with 150 unit sensing range in two scenarios in which target moves on a linear trajectory. In the first scenario, the target moves at a constant velocity. In the second scenario, the velocity of the target changes suddenly several times to a random value.



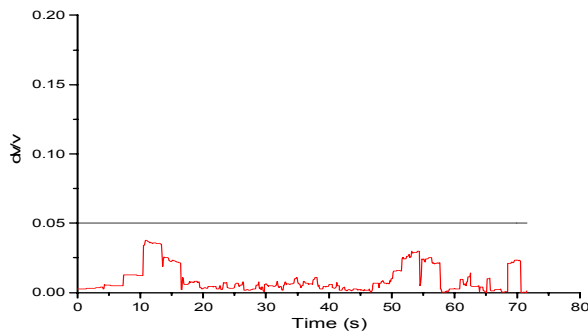
(a) Constant velocity



(b) Random velocity

Figure 6. Estimated velocity versus real velocity

Fig. 6 shows the estimated versus real velocities as a function of time in these two scenarios. Clearly, the estimated velocity is very close to the real velocity in the first scenario. These two agree also well in the second scenario, although there is some delay before the change of real velocity is reflected in its estimate.



(a) Constant velocity

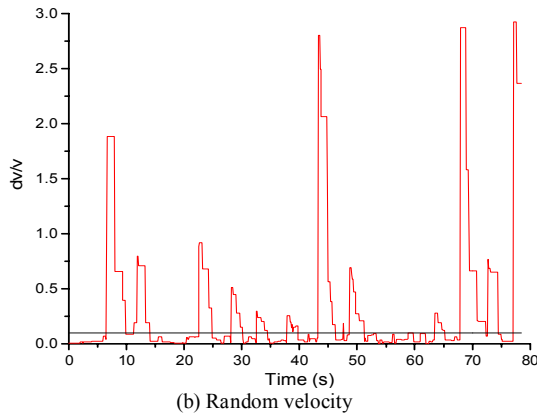


Figure 7. Velocity deviation

In Fig. 7, the plot of the deviation of the two velocities shows that the maximum deviation is always less than 5% in the first scenario and below 10% in the second scenario, excluding the brief moments immediately after the velocity change.

C. Trajectory Estimate

On all three kinds of trajectories and under the configuration of 800 nodes with 150 units sensing range, our trajectory estimate method delivers very good performance.

Fig. 8 shows the typical estimations for these trajectories. We measure the accuracy of estimated trajectory using the average difference between the estimated and real trajectories. It is calculated using the area of a polygon formed by these two trajectories divided by the length of the real target trajectory. The average accuracy are 0.085, 0.424 and 0.782, for linear, circular and piece-wise linear trajectories with random turns, respectively.

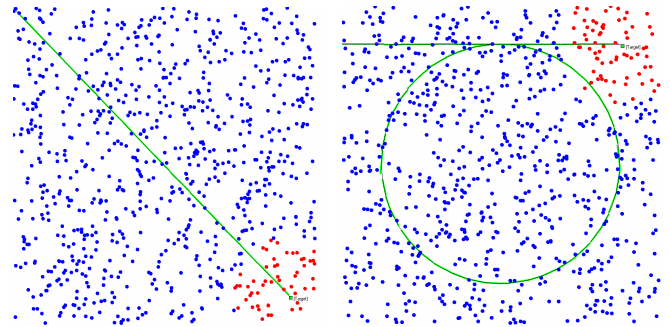
V. CONCLUSION

Target tracking is a typical and important application of sensor network usually using cooperative sensing. In this paper, we study the target tracking problem under the simple and basic binary sensor network model. We introduce a real-time distributed target tracking algorithm without time synchronization. Extensive simulations of this algorithm performed under different configurations and scenarios are reported. We observe that cooperative sensing involving just neighbor nodes can compensate for a very simple and basic sensing capabilities of the sensor network nodes. The introduced algorithm, thanks to cooperative sensing, yields very good performance estimating accurately the target location, velocity and trajectory using the binary sensor networks.

ACKNOWLEDGEMENT

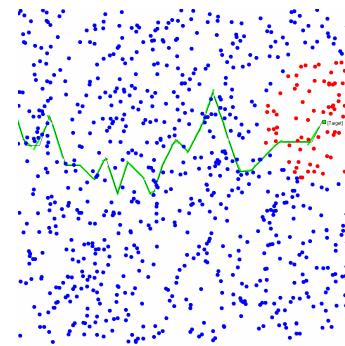
Research was sponsored by US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defence,

or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.



(a) Linear trajectory

(b) Circular trajectory



(c) Random trajectory

Figure 8. Examples of trajectory estimate

REFERENCES

- [1] C.-Y. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proc. of the IEEE*, vol. 91(8):1247–1256, 2003.
- [2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y.-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Int. J. Computer and Telecom. Networking*, 46:605–634, Dec. 2004.
- [3] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," *Proc. ACM SenSys*, 2003.
- [4] P. M. Djuric, M. Vemula, and M. F. Bugallo, "Signal processing by particle filtering for binary sensor networks," *Proc. 11th IEEE Digital Signal Processing Workshop & IEEE Signal Processing Education Workshop*, pp. 263–267, 2004.
- [5] T. Jing, S. Hichem, and R. Cedric, "Binary variational filtering for target tracking in sensor networks," *Proc. 14th IEEE/SP Workshop on Statistical Signal Processing*, pp. 685–689, 2007.
- [6] K. Mechitov, S. Sundresh, Y. Kwon, and G. Agha, "Cooperative tracking with binary-detection sensor networks," *Tech. Rep. UIUCDCS-R-2003-2379*, University of Illinois at Urbana-Champaign, Sept. 2003.
- [7] W. Kim, K. Mechitov, J.-Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," *Proc. IPSN*, 2005.
- [8] N. Shrivastava, R. Mudumbai, U. Madhow, and S. Suri, "Target tracking with binary proximity sensors: Fundamental limits, minimal descriptions, and algorithms," *Proc. ACM SenSys*, 2006.