Distributed Target Tracking with Imperfect Binary Sensor Networks

Zijian Wang, Eyuphan Bulut, and Boleslaw K. Szymanski
Department of Computer Science and Center for Pervasive Computing and Networking, RPI, Troy, NY

Abstract—**We study target tracking with wireless sensor networks 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 in its sensing range. A novel, real-time and distributed target tracking algorithm for imperfect binary sensing model is proposed, which is an extension of our previous work on ideal binary sensing model. The algorithm estimates the target velocity and trajectory in a distributed and asynchronous manner. Extensive simulations show that our algorithm achieves high performance and outperform other algorithms by yielding accurate estimates of the target's location, velocity and trajectory.**

Index Terms—**Target Tracking, Binary Sensor Networks, Distributed Algorithms, Imperfect Sensing**

I. Introduction

TARGET tracking is a representative and important application for wireless sensor networks [1,2]. 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.

There are two kinds of binary sensing model for binary sensor networks: ideal binary sensing model and imperfect binary sensing model. In ideal binary sensing model, each node can detect exactly if the target falls in to its sensing range R (as shown in Fig. 1(a)). In real world, detection ranges often vary depending on the environmental conditions, such as the relative orientation of the target and the sensor. These factors make target detection near the boundary of the sensing range much less predictable. The above observations give rise to an imperfect binary sensing model in which the target is always detected within an inner disk of radius R_{in} but is detected only with some nonzero probability in an annulus between the inner disk and an outer disk of radius R_{out}. Targets outside the outer disk are never detected (as shown in Fig. 1(b)).

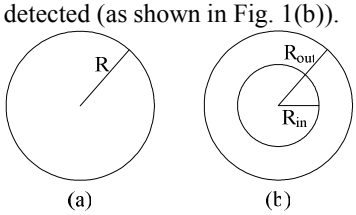


Fig. 1. Binary sensing model

A number of approaches using binary sensor networks for target tracking have been proposed in recent years. The algorithms presented in [3,4] 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 [5], 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 [6] improved the weight calculation for each sensor node that detected the target by using 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 [7], both the presence and absence of the target within the node's sensing range were 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

Manuscript received March 31, 2008.

Zijian Wang. Author is with the Department of Computer Science and the Center for Pervasive Computing and Networking, Rensselaer Polytechnic Institute, Troy, NY 12180 USA. (e-mail: wangz@cs.rpi.edu, phone: 518-944-1156).

Eyuphan Bulut and Boleslaw Szymanski. Co-authors are with the Department of Computer Science and the Center for Pervasive Computing and Networking, Rensselaer Polytechnic Institute, Troy, NY 12180 USA. (e-mail: {bulute, szymansk}@cs.rpi.edu).

compute the estimated trajectory, which means that the tracking is not real-time but delayed. In our previous work [8], we proposed a distributed target tracking algorithm for ideal binary sensing model. Each active node computes the target's location 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.

In this paper, we extend our previous work and propose a distributed target tracking algorithm that can be used for imperfect binary sensing model while keeping all the other properties of its predecessor.

The remainder of the paper is organized as follows. We describe the network model and our assumptions in Section II. In Section III, we first give a brief overview of our previous work and then introduce our distributed target tracking algorithm for imperfect binary sensing model. 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 the union of sensing regions of all network nodes guarantees redundant coverage of the region to be monitored. Each node generates one bit of information ("1" for target's presence and "0" for its absence) only at the moment when there is a change in the presence/absence status of the target. Each time a new bit of information is generated, the node communicates it to its neighbors that are defined as nodes whose sensing ranges intersect its sensing range. Each 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.

III. Distributed Target Tracking Algorithm

A. Overview of Algorithm for Ideal Binary Sensing Model

Basic Idea

To illustrate our basic idea, we use an example from Fig. 2, which shows a target moving through an area covered by three nodes with ideal binary sensing model. 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 order, at times t_4, t_5, t_6, respectively. With the ideal binary sensing model, 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 and those are the times at which the target enters and then exits sensing range of that 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 denoted as 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.

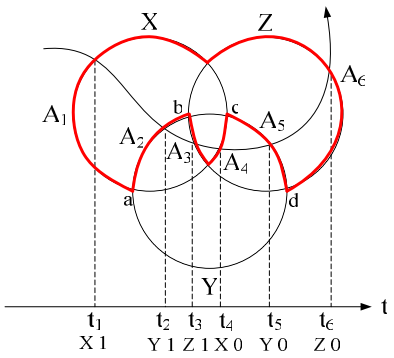


Fig. 2. Illustration of the basic idea behind the algorithm

Target Tracking Algorithm

At the network deployment stage, 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". 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.

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. 3. 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. 3(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. 3(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 angle. For example, in Fig. 3(c) $\angle 1o3$ - $\angle 2o4$ is equal to $\angle 1o2$. In a special case shown in Fig. 3(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:

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

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

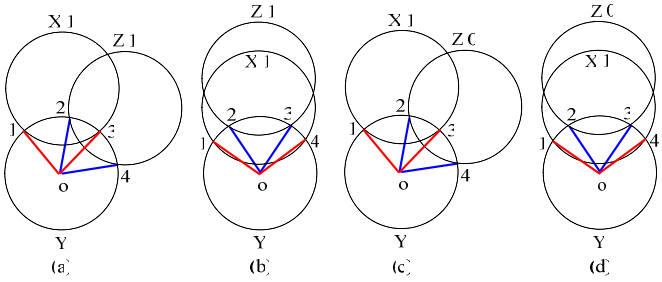


Fig. 3 Instances of angle combinations

B. *Tracking Algorithm for Imperfect Binary Sensing Model*

In order to make our algorithm robust, as in [7], we take a worst-case approach to the information provided by the imperfect binary sensing model: if a sensor output is "1", then we assume that the target is somewhere inside the large disk of radius *Rout*; if a sensor output is "0", then we assume that the target is somewhere outside the small disk of radius *Rin*.

The main influence of the imperfect binary sensing model is that we can no longer identify circular arcs that the target crosses (as was possible in the ideal binary sensing model) when there is a change in the status of the target sensed by a node. Instead, we can only identify that the target must be within the ring determined by *Rin* and *Rout*. However, we can use a thin ring section which is determined by the neighbor output to approximate the circular arcs and then estimate the position of the target. Although this will make the one-dimensional uncertainty of the target's location expand to a two-dimensional uncertainty, if the resulting ring section is short and thin, the error still will be small.

Initialization

In the initialize procedure, each node establishes a list of its neighbors and calculates the exact angle corresponding to a neighbor depending on the output and the relative position of that neighbor.

The three instances for neighbor (node Y) that outputs bit "1" are shown in Fig. 4. As described previously, if node Y outputs bit "1", we can only be sure that the target is within sensing range *Rout*. When node X senses there is a change in the status of the target, it knows that the target is within the ring determined by *Rin* and *Rout*. Depending on the relative position of node Y to node X, there would be up to two angles corresponding to node Y resulting from the intersection of *Rin* and *Rout* circle of node X and *Rout* circle of node Y. If there are two angles existing for node Y, we choose the angle that makes sure that the target must fall in this angle, for example we choose $\angle b_1ob_2$ and $\angle a_1oa_2$ in Fig. 4 (a) and (b) as the corresponding angle to neighbor Y. If there is only one angle existing for node Y, then it is the corresponding angle, as shown in Fig. 4 (c).

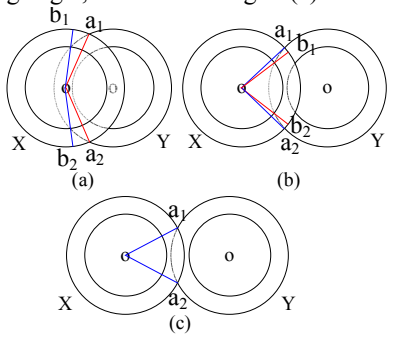


Fig. 4 Angle corresponding to neighbor's output "1"

The three instances for neighbor (node Y) that outputs bit "0" are shown in Fig. 5. As described previously, if node Y outputs bit "0", we can only know that the target is outside sensing range *Rin*. Depending on the relative position of node Y to node X, there would be up to two angles corresponding to node Y resulting from the intersection of *Rin* and *Rout* circle of node X and *Rin* circle of node Y. If there are two angles existing for node Y, we choose the angle that makes sure that the target must fall out of this angle, for example we choose $\angle a_1oa_2$ and $\angle b_1ob_2$ in Fig. 5 (a) and (b) as the corresponding angle to neighbor Y. For the instance shown in Fig. 5(c), we can not determine that the target is outside $\angle a_1oa_2$ because the target could be within $\angle a_1oa_2$ no matter what node X outputs. So, if node Y outputs bit "0", it will be considered as a neighbor of node X only if its *Rin* circle intersects with the *Rin* circle of node X.

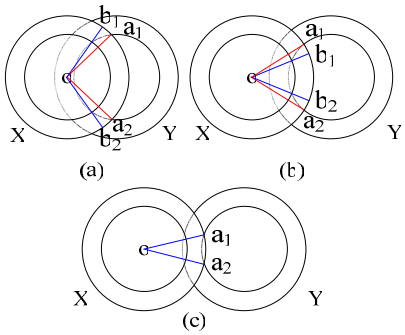


Fig. 5 Angle corresponding to neighbor's output "0"

Location Estimate

At the moment at which the node discovers the change in the target's presence, it calculates the final angle corresponding to the ring section that the target is crossing using the same angle combination method as in the ideal binary sensing model. Then, the thickness of the ring section is recalculated to make the

estimation of target position more accurate.

We calculate the intersection points of each pair of node X's neighbor that output bit "1". The intersection point that falls into the final angle and is most far away from the center of node X determines one of the boundaries of the ring section, which will make the ring section as thin as possible. A new thinner ring section is determined by this intersection point, *Rout* circle and final angle, for example ring section "abcd" shown in Fig. 6 (a). Please note that the neighbor node that outputs bit "0" contributes only to the angle combination but not to the thickness calculation. As shown in Fig. 6 (b), the recalculated ring section may exclude some area into which the target may fall, although with small probability because this area is near *Rout* circle of node X. But when the final angle is small, this area will be negligible in size. The target position is estimated as the center point of this ring section.

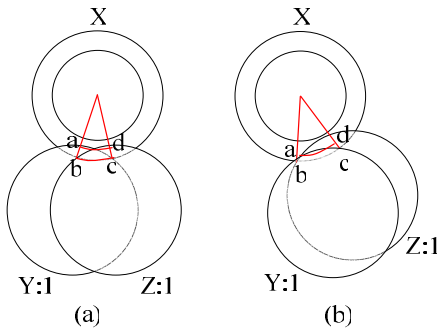


Fig. 6 Ring section thickness calculation

Velocity Estimate

We use a distributed, asynchronous algorithm to estimate the target velocity. As shown in Fig. 7, 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 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 and thin ring sections).

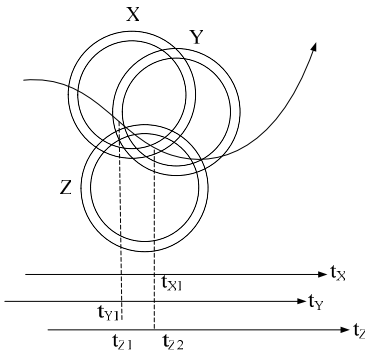


Fig. 7 Velocity estimation

Trajectory Estimate

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

$$w = \frac{1}{|\text{ring section}|/|\text{ring}|} \tag{2}$$

where $|\text{ring section}|$ is the area of corresponding ring section whose middle point is the estimated target location and $|\text{ring}|$ is the area of the ring determined by circles *Rin* and *Rout*. 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 can be expressed as $y = a \cdot x + b$ allowing us to define the corresponding minimization metric Q as:

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

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

IV. Simulation

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 R_{out}.

Simulation Setup

When evaluating the impact of network density on the location estimation accuracy, we kept the number of nodes fixed at 800 and varied the sensing range R_{out} from 40 to 150 units. The velocity of the target was adjusted proportionally to the sensing range, making it constant if measured in sensing range units.

Several types of trajectories have been considered: linear, circular, and a piece-wise linear trajectory with random turns. In order to exclude the boundary effect, all the trajectories are confined within the square area with length of 800-R_{max} in the middle, where R_{max} is the maximum sensing range (150 units) in the simulation. For the random trajectory, the length of the trajectory is proportional to the sensing range R_{out} . As in [5], we set R_{in} =0.9*R_{out}.

Detection Probability

Two kinds of detection probabilities for imperfect binary sensing models are used. The first one is a constant distribution as defined in formula (4), where d is the distance between the sensing node and the target.

$$\begin{cases} \dfrac{R_{out} - d}{R_{out} - R_{in}} & R_{in} \le d \le R_{out} \\ 1 & d \le R_{in} \\ 0 & R_{out} \le d \end{cases} \tag{4}$$

The second one is an exponential distribution defined in formula (5), where α is its exponent parameter. In order to make the detection probability approximately 0 when d=R_{out},

we let $e^{-\alpha(R_{out}-R_{in})}=0.01\%$, yielding $\alpha=\frac{\ln(0.01\%)}{R_{in}-R_{out}}$.

$$\begin{cases} e^{-\alpha(d-R_{in})} & R_{in}\le d\le R_{out} \\ 1 & d\le R_{in} \\ 0 & d\ge R_{out} \end{cases} \quad (5)$$

Algorithms to be Compared

We compare our algorithm with the following other four algorithms introduced in [5] and [6]:

(1) Equal weight: target's position is estimated as the average of the detecting sensors' positions.

(2) Distance weight: target's position is estimated as the weighted average of the detecting sensors' positions. The weight for each node is set at $1/\sqrt{R_{out}^2-0.25(v\cdot t)^2}$, where v is the target velocity and t is the time expired since the target has been detected.

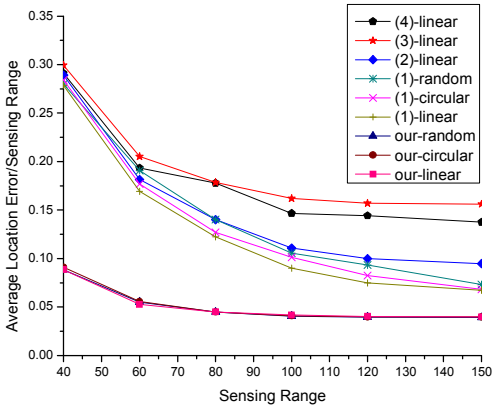
(3) Duration weight: target's position is estimated as the weighted average of the detecting sensors' positions. Given the time t that expired since the node has detected the target, the weight for each node is ln(1+t).

(4) Line fit: the initial estimate of the target position is made as in algorithm (2), and then a line that fits the history target position point is found and the current target position is refined using this line and the target velocity.

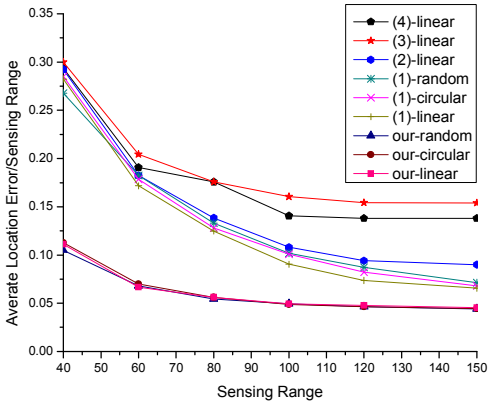
Because algorithms (2) (3) and (4) are design for linear trajectory with constant velocity, we only compare our algorithm with them for the linear trajectory.

Simulation Results and Discussion

Fig. 8 shows the location estimate accuracy results under both of the two detection probabilities. We can see that in both cases, our algorithm gets the best result compared with all four other algorithms. Additionally, the location estimate accuracies for all the three trajectories of our algorithm are nearly the same, which demonstrates that our algorithm works well for all kinds of trajectories. Even for the sparse network with sensing range R_{out} = 40, which means that there are only five neighbor nodes within each sensing range, the algorithm still performs well.



(a) The results under the first detection probability

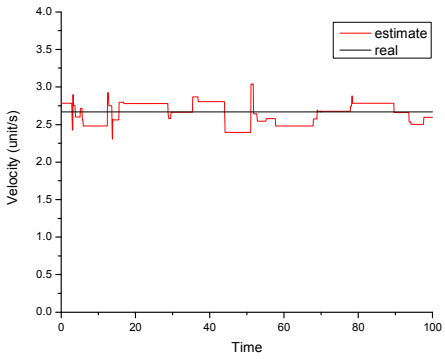


(b) The results under the second detection probability

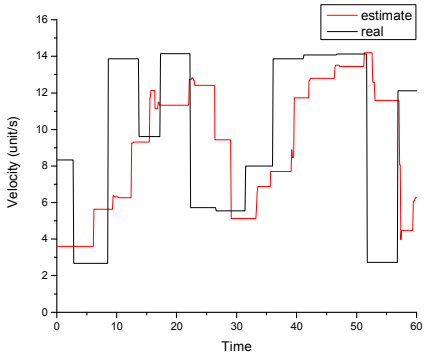
Fig. 8 The location estimate accuracy

B. Velocity Estimate

We tested the performance of velocity estimation under the configuration of 800 nodes with R_{out}=40 unit and R_{in}=0.9*R_{out} unit sensing ranges using the first detection probability in two scenarios in which target moves along a linear trajectory. In the first scenario, the target moves at a constant velocity which is R_{out}/15 unit/second. In the second scenario, the velocity of the target changes suddenly to a random value that is a multiple of R_{out}/15 unit/second several times during simulation.



(a) The results for the constant velocity



(b) Random velocity

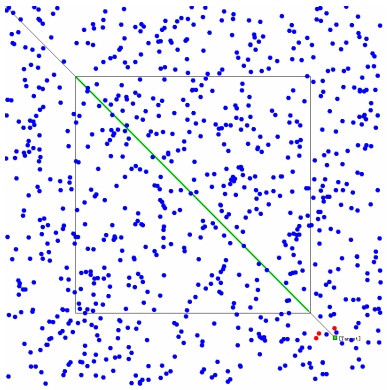
Fig. 9 The estimated velocity versus the real velocity

Fig. 9 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

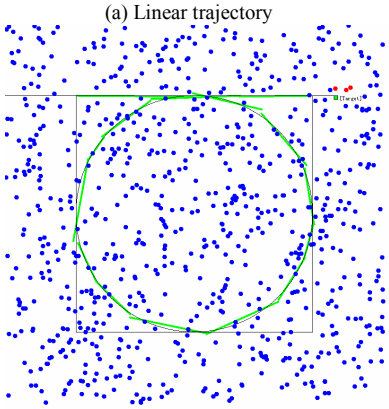
delay before the change of real velocity is reflected in its estimate so there are some large deviations in the brief moments immediately after the velocity change.

C. *Trajectory Estimate*

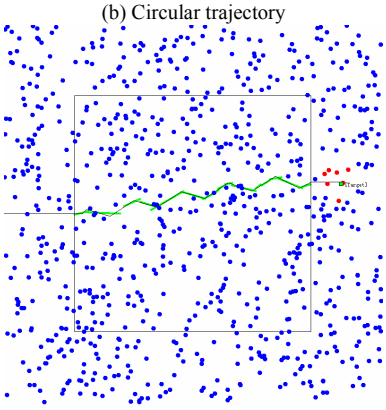
Fig. 10 shows the typical estimations for three trajectories under the configuration of 800 nodes with R_{out}=40 units and R_{in}=0.9*R_{out} using the first detection probability. We measure the accuracy of estimated trajectory using the average difference between the estimated and real trajectories. 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 accuracies are 0.287, 1.811 and 2.873, for linear, circular and piece-wise linear trajectories with random turns, respectively.



(a) Linear trajectory



(b) Circular trajectory



(c) Random trajectory

Fig. 10 Examples of trajectory estimation

V. Conclusion

Target tracking is a typical and important application of sensor network usually using cooperative sensing. In this paper, we extend our study of target tracking problem under the ideal binary sensing model and introduce a real- time distributed target tracking algorithm without time synchronization for imperfect binary sensing. Extensive simulations of this algorithm performed under different configurations and scenarios are reported. We observe that our algorithm yields good performance and outperform other algorithms by estimating accurately the target location, velocity and trajectory.

References

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, 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," The International J. of Computer and Telecom. Networking, 46:605-634, Dec. 2004.

[2] 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.

[3] 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.

[4] T. Jing, S. Hichem, and R. Cedric, "Binary variational filtering for target tracking in sensor networks," Proc. IEEE/SP 14th Workshop on Statistical Signal Processing, pp. 685-689, 2007.

[5] K. Mechitov, S. Sundresh, Y. Kwon, and G. Agha, "Cooperative tracking with binary-detection sensor networks," Technical Report UIUCDCS-R-2003-2379, University of Illinois at Urbana-Champaign, September 2003.

[6] W. Kim, K. Mechitov, J.-Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," Proc. IPSN, 2005.

[7] 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.

[8] Z. Wang, E. Bulut, and B. K. Szymanski, "A distributed cooperative target tracking with binary sensor networks," Proc. ICC2008 CoopNet Workshop, to appear.