

Distributed Target Tracking with Directional Binary Sensor Networks

Zijian Wang, Eyuphan Bulut, and Boleslaw K. Szymanski

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

Abstract—One of the most common and important applications of wireless sensor networks is target tracking. We study it in its most basic form, assuming the binary sensing model in which each sensor can return only information regarding target's presence or absence within its sensing range. However, unlike most of traditional approaches to binary sensing, we allow sensors to recognize not only target's range but also a sector within the circular range around it. Examples of such sensors include cameras, infrared sensors, ultrasonic sensors, etc. For simplicity, we assume that either a group of sensors are collocated in a single spot providing 360 degree coverage or a sensor has multiple antennas or camera providing such coverage. A novel, real-time and distributed target tracking algorithm with directional binary sensor networks is proposed. It is an extension of our previous work on omni-directional binary sensor networks. Using simulations, we demonstrate that this new algorithm achieves high performance and outperforms other algorithms by yielding accurate estimates of the target's location. In addition, we discuss the fundamental performance limits and improvement of the tracking performance resulting from providing direction range in addition to a distance range for the algorithm.

Keywords—target tracking; binary sensing; directional sensing; wireless sensor networks; distributed algorithms

I. INTRODUCTION

Wireless sensors networks composed of miniature devices that integrate physical sensing, data processing and communication capabilities present great opportunities for a wide range of applications. Among them, target tracking is both representative and important application that usually relies on cooperation between sensing nodes to achieve good results. The fundamental studies of target tracking often focus on networks composed of sensor nodes with the most elementary sensing capabilities that provide just binary information about the target, indicating whether it is present or absent in the sensing range of a node. These so-called binary sensor networks constitute the simplest type of sensor networks that can be used for target tracking.

A number of approaches using binary sensor networks for target tracking have been proposed in recent years. The algorithms presented in [1, 2] 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 in any non-trivial size network. In [3], each point on the target's path is computed using the weighted average of the detecting sensors' locations. Then, a line that best fits the newly estimated location and the points on the trajectory established in the recent past is used as the target trajectory. Kim et al [4] 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, these algorithms use positions of only the sensor nodes that detected the target. Actually, the absence of detection also provides the information that can be used to improve the tracking accuracy. In [5], both the presence and absence of the target within the node's sensing range are used to form local regions that the target has 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. Hence, the tracking is not real-time but delayed.

In our previous work [6], we proposed a distributed target tracking algorithm for the ideal binary sensing model. In it, 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 targets moving in random directions and with varied velocities. In [7], we presented an extension of this algorithm that made it applicable to imperfect binary sensing model while keeping all the other properties of its predecessor.

All of the algorithms mentioned above used omni-directional binary sensor networks, in which each sensor can only detect the target presence or absence within its sensing range but can not get any direction information of the target. In

this paper, we propose a novel distributed target tracking algorithm using directional binary sensor networks. Under the directional binary sensing model, each sensor node's sensing region is divided into sectors and each node can identify in which sector the target is present or absent, which gives rough direction information of the target. Although this directional binary sensing model has been used in previous works [8-9], but to the best of our knowledge, it has not been used by any of the previously published algorithms for target tracking. To establish fundamental limits of the directional binary sensing, we consider the ideal case of error-free sensing in this paper, leaving more complex analysis of impact of errors on the tracking algorithms to the future work.

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 four-sector directional binary sensor networks. We derive the fundamental performance limits for our algorithm in Section IV. Section V presents the simulation results. Finally, we provide conclusions in section VI.

II. NETWORK MODEL AND ASSUMPTIONS

The sensor network comprises N nodes placed uniformly randomly 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. Each node's sensing region is divided into sectors. An example of a four-sector directional binary sensor node is shown in Fig.1. Its entire sensing region is divided into four equal size sectors, which are numbered from 0 to 3 in the clockwise sequence order. The boundaries of sectors intersect with the sensing range at point A, B, C and D, which divide the sensing circle into four arcs AB, BC, CD and DA. The initial angle of radius "oA" with positive x-axis is selected randomly for each sensor. We will denote by s the number of sectors of each node and by $b = \text{ceiling}(\log_2(s+1))$ the number of bits needed to represent presence of target in one of the sectors or its absence from the node's sensing range ($\text{ceiling}(x)$ returns the smallest integer not less than x).

At the moment at which the target enters or exits the sensing range of a sensor node for the first time, that node will generate b bits of information, indicating in which sector the target is present or that it is absent from the sensing range. This b -bit status information is also updated at the moment at which the target exits currently visited sector and enters another sector of the same sensor node. If there is no change in b -bit status information, the node remains silent to save energy and bandwidth and to avoid collisions with other nodes' transmissions. Each time a new b -bit 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 a multi-hop transmission). 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, or through GPS devices, not discussed here).

For simplicity, we assume that the sensing range of each node is identical across the network and each node's sensing region is divided into the same number of equal size sectors. However, our algorithm also applies when sensing range, sector number and sector size vary from node to node. Additionally, we assume that the target moves with velocity that is low relative to the node's sensing frequency. As a result, time of discovery of the change in the target's presence or absence within the node's sensing range differs negligibly little from the time of the target moves within or out of this range. This assumption is reasonable because the sensing frequency for ultrasonic sensor is usually around 10^{-2} - 10^{-3} second and the sensing frequency for infrared sensor is usually above 10^{-4} second, which is much higher than frequency with which a target with a typical real-world velocity passes through the sensor ranges.

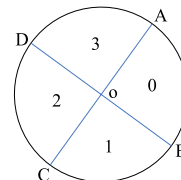


Figure 1. A four-sector directional binary sensor

III. A COOPERATIVE TRACKING ALGORITHM

A. Basic Idea

To illustrate our basic idea, we use an example from Fig.2, which shows a target moving through an area covered by two nodes whose sensing ranges are divided into four sectors. Initially, the target is outside of the sensing ranges of two nodes. Later, it moves into sector 3 of node X at the system time t_1 , crosses sector 2 of node X and enters sector 3 of node Y at time t_2 . Then, it leaves sector 1 of node X and sector 0 of node Y in that sequence, at times t_3 and t_4 , respectively. According to the model described in the previous section, each node will generate information at the time of first sensing the target's presence and later at the time of first lacking to sense its presence which corresponds to the times at which the target enters and then exits the sensing range of the node. Besides those, each node will also generate information when the target leaves one sector and enters another sector of the same node. As a result, at the transition time t_j , the target must be on arc A_j which is a part of the arc of the corresponding sector of the node reporting the information. Hence, arc A_j can be determined cooperatively from b -bit information reported by the neighbors of that node. Let's consider arc A_2 defined at time t_2 as an example. Before time t_2 , node Y will receive three messages from node X, which we can mark as "X13", "X3-2" and "X2-1". "X13" means that node X first senses the target ("1" stands for presence, "0" stands for absence) in sector 3. "X3-2" means that the target within sensing range of node X leaves sector 3 and enters sector 2 of node X. "X2-1" means that the target within sensing range of node X leaves sector 2 and enters sector 1 of node X. At time t_2 , node Y senses the target presence within its sector 3 for the first time, so the target must be on arc "abc" which corresponds to sector 3 of node Y. At that time, node Y also knows that the target is within sector 1 of node X. 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. The shorter this circle section is, the smaller the uncertainty becomes.

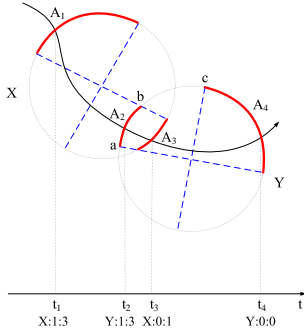


Figure 2. An illustration of the basic idea behind the algorithm

B. Tracking Algorithm

At the network deployment stage, each node initializes status lists of its neighbors for its four sectors. Each time a node receives 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 (no matter which sector the target enters or exits), it identifies the arc that the target is crossing. The target location is estimated as the middle point of the corresponding arc and broadcasted to neighbors. Our tracking algorithm includes the following steps:

1) Neighbor sector match

In the neighbor sector match procedure, each node finds out neighbor relations from all the sectors of neighbor node for each of its sector. At first, each node will calculate three types of intersection points that include: (1) intersection points of the sensing circles of the node and its neighbor, e.g. points "a" and "f" in Fig.3; (2) intersection points of the sensing circles of the node and its neighbor's sector boundaries, e.g. points "c" and "d" in Fig. 3; (3) intersection points of the sensing circles of the node and its sector boundaries that fall into sensing area of its neighbor, e.g. points "b" and "e" in Fig.3.

Then each of these intersections is sorted by the angle formed by one of the intersection points of type (1) and the intersection point itself, e.g. $\angle aob$, $\angle aoc$, $\angle aod$. After sorting, the sequence of intersection points in Fig. 3 will be "abcdef". These sorted points form a number of angles defined by each pair of points in sequence, e.g. $\angle aob$, $\angle boc$, $\angle cod$. Then, the center point of each arc corresponding to each angle is calculated and checked to see which sector it falls into. For example, the center point of the arc corresponding to $\angle boc$ falls into sectors 0 of node X and 3 of node Y, which means that sector 3 of node Y is a neighbor sector of node X's sector 0.

2) Initialization and information update

Each node first establishes neighbor lists for each of its sectors. Each element of such list stores: neighbor node identifier, neighbor sector identifier, intersection points related to this neighbor sector, an angle corresponding to the arc defined by these intersection points and b -bit status information

generated by the neighbor, initialized to "0". Upon receiving information from a neighbor, the node updates the corresponding entry in the list. Consider sector 0 of node X in Fig. 3. It has three neighbor sectors: sector 0, 1 and 3 of node Y. The intersection points "c", "d" and $\angle cod$ are related to neighbor sector 0. The intersection points "d", "e" and $\angle doe$ are related to neighbor sector 1. The intersection points "b", "c" and $\angle boc$ are related to neighbor sector 3.

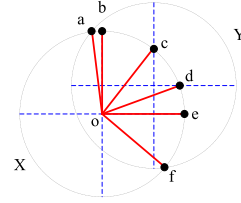


Figure 3. Neighbor list initialization

3) Location estimate

When node senses change in the status of presence or absence of target from one of its sectors, it will combine all angles in the corresponding neighbor list to determine the arc that the target is crossing. The four instances of this process are shown in Fig. 4. If sectors of both neighbors generated bits indicating target's presence, the corresponding central angles are combined by "&" operation that returns the intersection of the two angles. As shown in Fig. 4(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 sector. One special instance is shown in Fig. 4(b), where the common angle is just one of the two angles. If the status of one neighbor sector indicates presence while the other indicates absence of the target, the corresponding central angles are combined with "-" operation that returns the angle formed by excluding the second angle from the first. For example, in Fig. 4(c) $\angle 1o3 - \angle 2o4$ is equal to $\angle 1o2$. In a special case shown in Fig. 4(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.

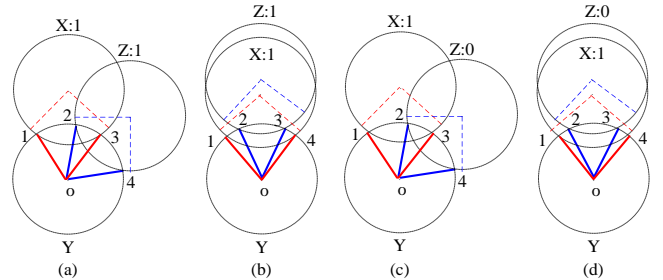


Figure 4. Instances of angle combinations

Let FA be the sought arc's central angle initialized to $2\pi/s$ (the sensing border of a sector that the target is crossing). Let IN be the set of neighbor sectors indicating target's presence while OUT be the set of neighbor sectors indicating target's absence. Then, the final angle whose corresponding arc is the one that the target is crossing can be expressed as:

$$FA = FA \& \text{angle}_i - \text{angle}_j \quad (1)$$

where angle_i is the central angle of the neighbor's sector i .

It should be noted that regardless of the relative position of any two sensor nodes and their number of sectors, the intersection situations for them must be one of the four cases in Fig. 4 because the sensor nodes are static. So our tracking algorithm can always converge and find the correct arc that the target is crossing. Clearly, the overhead of the angle combination procedure is determined by the number of sector neighbors which is defined by the network density and the number of sectors of each node.

IV. FUNDAMENTAL PERFORMANCE LIMITS ANALYSIS

In this section, we first derive the fundamental performance limits for omni-directional binary sensors and then we extend it to the directional binary sensors.

Assume that we have a domain of area A_d in which there are total of N sensors, each with a uniform sensing range (to simplify the analysis, we set this sensing range to be one unit, $R = 1$). Let's consider the specific time instance at which the target T has been just sensed by node X whose center is at point C_0 (we refer to this circle as circle C_0), as shown in Figure 5. If neighbor sensors also sense the target, they must be within a circle centered at T with radius of one unit. Thus, the probability P_k that there are k ($0 < k < N$) neighbors that also sense the target within their sensing ranges is:

$$P_k = \binom{N-1}{k} \left(\frac{\pi}{A_d} \right)^k \left(1 - \frac{\pi}{A_d} \right)^{N-k-1} \quad (2)$$

All k intersecting arcs will have one point on one side of the target and another on the other side. So the resulting average shortest length of the arc of all intersections will be the double of the length of the distance of the target from the closest intersection point on either side of the target. Then, the accuracy of our algorithm will be just half of the average shortest length of the arc. Let node Y centered at point N_0 be one of the neighbors of node X that also senses the target within the sensing range of one unit (we refer to this circle as circle N_0). These two sensing circles intersect at points P and P' . We chose the notation so that P is the closest of the two intersection points to the target, so its distance may determine the accuracy of the target position measurement. It is important to note that any node whose sensing circle also intersects with circle C_0 at point P must be on the circle centered at point P with radius of one unit (we refer to this circle as circle P). As a result, any node that also senses the target but has an intersection point with circle C_0 closer to target T than point P must fall in the shadowed area that we denote as A . If x denotes size of angle PC_0T in radians, then the length of arc PT is x . The probability $P(x)$ that the length of arc PT is less than or equal to x is $\| \text{area } A \| / \| \text{circle } T \|$, where $\| \cdot \|$ returns the area of its argument, sector QTC_0 is in circle T and sector QPC_0 is in circle P . We can get that $\| \text{area } A \| = x + \sin(x)$ and $P(x) = (x + \sin(x)) / \pi$. The probability $P_s(x)$ for the shortest arc created by k neighbor nodes sensing the target being shorter than x is defined as:

$$P_s(x) = 1 - \text{Pr ob}(y_s \geq x) = 1 - \prod_{i=1}^k \text{Pr ob}(y \geq x) = 1 - (1 - P(x))^k. \quad (3)$$

Hence the average length of the shortest arc is:

$$\delta = \sum_k P_k \int_{x=0}^{\pi} x d(1 - (1 - P(x))^k) = \sum_k P_k \delta_k \quad (4)$$

Where

$$\delta_k = \int_{x=0}^{\pi} -x d(1 - P(x))^k = \int_{x=0}^{\pi} \left(\frac{x - \sin(x)}{\pi} \right)^k dx \quad (5)$$

It is easy to see that

$$\delta_k \approx \int_{x=\pi/2}^{\pi} \left(\frac{2x - \pi}{\pi} \right)^k dx = \frac{\pi}{2(k+1)} = \frac{1}{2\rho} \quad (6)$$

where ρ is the density of the sensors in the network when $R=1$. In a general case of R being the arbitrary sensing radius of each node and d being the sensor density, we have $\rho = dR^2$ and the unit numerator in (6) is R . Using derivation patterned on [5] (omitted here for a lack of space) we can show that with randomly distributed sensors, the linear (1-D) error of our method is proportional to $O(1/(dR))$ which is the same order of magnitude as the radius of the error area reported in [5]. The reason is because we use the arc of sensing boundary instead of sensing area to estimate the target location which reduces the uncertainty dimension from two to one.

For the directional binary sensor, the upper bound of the integration in (4) and (5) will be π/s . Yet, we only consider the effect of nodes that sense the target in (4). Actually, nodes that do not sense the target also contribute to the accuracy of the algorithm by cutting the arc shorter, which is not discussed here.

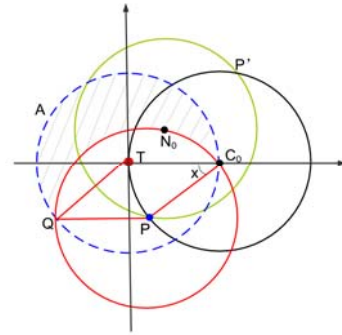


Figure 5. A configuration for calculating the shared arc length distribution

V. SIMULATION

We have designed a QT (a cross-platform application framework) based simulator that uses data exchange between multi threads to simulate the wireless communication between sensor nodes. In the simulator, we assume that there is some MAC (Media Access Control) protocol providing ideal wireless communication, which means that there are neither collisions nor data drops (simulation results from our previous works showed the collisions in MAC layer impact our tracking

algorithm performance negligibly). We chose the location estimation error, measured as the ratio of the distance between the estimated and real target locations to the sensing range R , as the basic metrics of target tracking. This metric reflects the final angle corresponding to the arc that the target is crossing in our method, and therefore is independent of sensing range, but should decrease with the increase of network density.

A. Simulation Setup

When evaluating the impact of network density on the location estimation accuracy, we kept the number of nodes fixed at 300 within a 800 by 800 area and varied the sensing range R from 50 to 150 units with increment of 25 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, including linear, circular, and a piece-wise linear trajectory with random turns. To exclude the boundary effect, all the measured trajectories are confined within the square with sides of $800 - R_{\max}$ ($R_{\max} = 150$ is the maximum sensing range) in the center of the simulated area. For the random trajectory, the length of the trajectory is proportional to the sensing range R .

B. Algorithms to be Compared

Although papers [1-5] are related to our paper, all algorithms presented in [1], [2] and [5] need a central node to gather the target information and estimate the target location. Our algorithm estimates the target information in a distributed way, so we compare its performance with the following four distributed algorithms introduced in [3] and [4]:

(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, where the weight for each node is set at $1/\sqrt{R^2 - 0.25(v \cdot t)^2}$ and v is the target velocity, while 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, where 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 previous target positions is found and the current target position is refined using this line and the target velocity.

Because algorithms (2) (3) and (4) are designed for a linear trajectory with constant velocity, our comparisons in their cases are restricted to the linear trajectory. We also compare our new algorithm with the original target tracking algorithm using traditional omni-directional binary sensor networks [6]. In comparisons, we use either four-sector or twelve-sector sensor nodes.

C. Simulation Results and Discussion

Fig. 6 shows the typical example of estimated location points for three kinds of trajectories. The sample network is composed of 300 nodes with sensing range of 150 units and four-sector directional binary sensors.

We ran each simulation setup ten times and present the averages of those runs and their confidence interval under confidence level of 95%. Fig. 7 shows the location estimate accuracy results under each of the three trajectories.

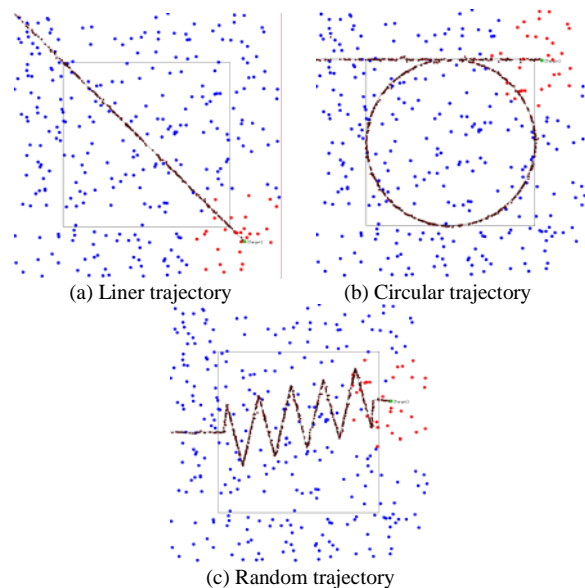


Figure 6. Examples of location estimation

In all the presented cases, the twelve-sector directional binary sensing gets the best results; the four-sector directional binary sensing outperforms our original binary sensing method, which is already the best among the (1) to (4) algorithms. The ratio of accuracy of twelve-sector directional binary sensing algorithm to our original binary sensing method is nearly 2. The ratio of accuracy of twelve-sector directional binary sensing algorithm to the best results of among (1) to (4) algorithms grows from nearly 8 for an important case of network with medium density (sensing range of 50 units) to around 16 for dense networks. Additionally, the location estimate accuracies of all three trajectories of our algorithm are close to each other, demonstrating that our algorithm works well for all kinds of trajectories. Even for a sparse network with sensing range $R = 50$, which means that there are only three or four neighbor nodes within each sensing range, the sector directional binary sensing algorithm performs well. Our tracking algorithm is very austere in sending messages. A node generates a single message each time there is a change in target's presence within the node's sensing range or the sector within which the target is located changes. Hence, the communication incurred creates little burden for the MAC layer and consumes little energy.

It is clear that when the number of sectors of each node increases, the complexity of the algorithm only increase in the neighbor initialization procedure due to the increase of intersection points. But for the angle combination procedure, the algorithm complexity remains the same because all the possible angle relations are the four instances shown in Fig. 4.

We also calculated the ratio between the increase in the number of messages produced and the increase in accuracy when we increased the number of sectors from 1 (the omni-directional binary sensor) to 4, from 4 to 12 and from 1 to 12.

These ratios reflect the tradeoff between the gain in accuracy and the cost of sending more messages. The smaller the ratio is, the more benefit we gain.

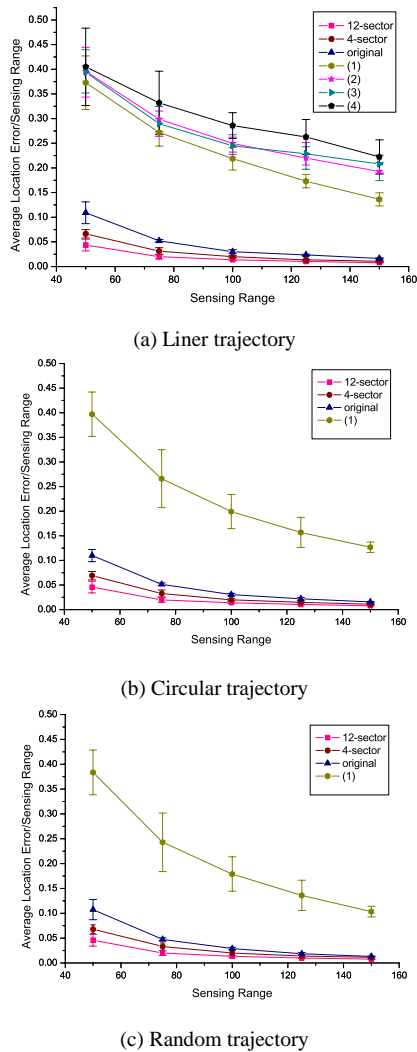


Figure 7. The location estimate accuracy

As shown in Fig. 8, the benefit is not constant. For example, if we use 4-sector binary sensor instead of omni-directional binary sensor with sensing range 125, the benefit is higher than for other sensing ranges. However, using more sectors to get better accuracy increases the chance of message collisions, which, by delaying message transmission decrease the location estimate accuracy. Therefore the number of sector used should be carefully selected in relation to the network density.

VI. CONCLUSION

In this paper, we extend our study of target tracking from the traditional omni-directional binary sensing model to directional binary sensing networks for which we introduced a real-time distributed target tracking algorithm. Extensive simulations of this algorithm performed under different configurations are reported. We observe that our new algorithm yields good performance, better than the performance achieved by our previous and other algorithms in terms of accuracy of

target's location estimation. Our target tracking method can be directly adapted to multi target tracking when the targets are sufficiently separated from each other. Even when targets get close, we should be able to distinguish each target by predicting its position through past location estimates. The details of such a design will be the subject of our future works.

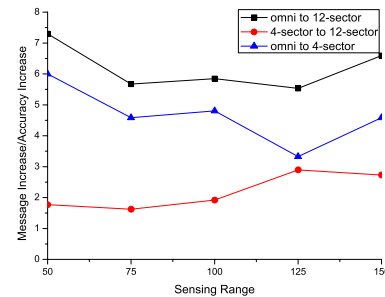


Figure 8. The ratio of message count increase to accuracy increase

ACKNOWLEDGMENT

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.

REFERENCES

- [1] P. M. Djuric, M. Vemula, and M. F. Bugallo, "Signal processing by particle filtering for binary sensor networks," Proceedings of the 2004 IEEE 11th Digital Signal Processing Workshop & IEEE Signal Processing Education Workshop, 2004, pp. 263-267.
- [2] Teng Jing, Snoussi Hichem, and Richard Cedric, "Binary variational filtering for target tracking in sensor networks," IEEE/SP 14th Workshop on Statistical Signal Processing, 2007, pp. 685-689.
- [3] 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.
- [4] W. Kim, K. Mechitov, J.-Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," In Proc. IPSN, 2005.
- [5] N. Shrivastava, R. Mudumbai, U. Madhow, and S. Suri, "Target tracking with binary proximity sensors: Fundamental limits, minimal descriptions, and algorithms," In Proc. of ACM SenSys, 2006.
- [6] Z. Wang, E. Bulut, and B. K. Szymanski, "A distributed cooperative target tracking with binary sensor networks," Proc. IEEE International Conference on Communication (ICC) Workshops, May 2008, Beijing, China, pp. 306-310.
- [7] Z. Wang, E. Bulut, and B. K. Szymanski, "Distributed Target Tracking with Imperfect Binary Sensor Networks," Proc. IEEE Global Communications Conference (GLOBECOM), December, 2008, New Orleans, USA, pp. 1-5.
- [8] Y. Cai, W. Lou, M. Li and X. Y. Li, "Target-oriented scheduling in directional sensor networks," INFOCOM 2007, Anchorage, AK, 2007, pp. 1550-1558.
- [9] J. Djughash, S. Singh, G. Kantor, and W. Zhang, "Range-only slam for robots operating cooperatively with sensor networks," Proc. IEEE Int. Conf. Robotics and Automation, 2006, pp. 2078-2084.