Completely Distributed Particle Filters for Target Tracking in Sensor Networks

Bo Jiang, Binoy Ravindran
Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24061
Email: {bjiang,binoy}@vt.edu

***Abstract*—Particle filters (or PFs) are widely used for the tracking problem in dynamic systems. Despite their remarkable tracking performance and flexibility, PFs require intensive computation and communication, which are strictly constrained in wireless sensor networks (or WSNs). Thus, distributed particle filters (or DPFs) have been studied to distribute the computational workload onto multiple nodes while minimizing the communication among them. However, weight normalization and resampling in generic PFs cause significant challenges in the distributed implementation. Few existing efforts on DPF could be implemented in a completely distributed manner. In this paper, we design a completely distributed particle filter (or CDPF) for target tracking in sensor networks, and further improve it with neighborhood estimation toward minimizing the communication cost. First, we describe the particle maintenance and propagation mechanism, by which particles are maintained on different sensor nodes and propagated along the target trajectory. Then, we design the CDPF algorithm by adjusting the order of PFs' four steps and leveraging the data aggregation during particle propagation. Finally, we develop a neighborhood estimation method to replace the measurement broadcasting and the calculation of likelihood functions. With this approximate estimation, the communication cost of DPFs can be minimized. Our experimental evaluations show that although CDPF incurs about 50% more estimation error than semi-distributed particle filter (or SDPF), its communication cost is lower than that of SDPF by as much as 90%.**

***Keywords*-Distributed particle filter; wireless sensor network; target tracking; Bayesian estimation; neighborhood estimation**

I. Introduction

Wireless sensor networks, which consist of a large number of multi-functional and low cost sensor nodes, have been extensively studied for collecting data from a geographical region of interest. A typical characteristic of WSNs is that their resources—including the power supply, computing and communication capabilities of sensor nodes—are strictly constrained [1]. Thus, resource constraints have to be considered carefully in the design of algorithms/protocols for WSNs.

Among the diverse application domains of WSNs, target tracking is one of the most fundamental types of application, which studies the dynamic state estimation problem by modeling the state space as a stochastic process that evolves over time [2].

For dynamic state estimation such as tracking problems, particle filters are one of the most widely used Bayesian estimation methods that approximate the optimal solution [3]. Particle filters are sequential Monte Carlo methods that estimate nonlinear and/or non-Gaussian dynamic processes. The posterior probability density function (or pdf) of Bayesian estimation is represented with discrete samples (or particles) with associated weights. Then in each iteration, PFs draw particles from a proposal distribution (or importance density), assign them with corresponding weights, normalize the weights, possibly resample, and finally make the estimation based on these weighted particles.

Despite their attractive tracking performance and flexibility [4], the application of PFs in WSNs is challenging due to the limited resources of WSNs. This challenge is mainly introduced by the centralized computation manner of generic PFs: weight normalization and resampling require collection of data from multiple nodes to a single computational center (either a cluster head [5], [6] or a global transceiver/sink node [7], [8]). Specifically, centralized particle filters (or CPFs) introduce the following problems: 1) collecting data consumes significant energy; 2) convergecast communication introduces a long delay, as the computational center has to receive messages in a sequential order; and 3) centralized implementation is vulnerable as a single point of failure [9].

Distributed particle filters [10] were studied as a response to these problems, in particular, to offload the computation from the central unit as well as to reduce convergecast communication [11]. However, few existing efforts on DPF were implemented in a completely distributed manner [7], because the aggregation of weights is inevitable. For efficiently transmitting the particle data to the computational center, there exist several DPF efforts that focus on reducing the communication cost by compressing messages, such as Gaussian mixture approximation [5], non-parametric particle compression based on support vector machine [9], and adaptive encoding [10], [12]. But like CPFs, these DPF efforts all share a common problem: the number of messages remain unchanged. Usually, compressing the number of messages is more efficient for saving energy than compressing the data contained in each message, especially in duty-cycled WSNs where nodes need to wake up from the sleep state for transmitting data [13], irrespective of how much data they need to transmit. Therefore, only when PFs are implemented in a completely distributed manner, the communication cost

can be minimized and the energy efficiency of WSNs can be enhanced significantly.

Toward this, we need to develop a method to aggregate the particle weights without any extra communication other than necessary particle propagation and/or those needed for sharing of measurements. This may be achieved by maintaining particles on different nodes and propagating it along the target trajectory. Based on the overhearing effect [14], nodes may receive all the propagated particles, thereby obtaining the aggregation as a side product of particle propagation.

Another important feature of WSNs that we may use for this purpose is that the local status in a WSN is relatively stable in the short term. The local status may include, but is not limited to, node positions, the topology, and the detection capability of neighbor nodes. Based on this stable local status, it is possible for a node to estimate the working status of its neighbor nodes thus the contributions they may make toward target estimation. We fully leverage this feature to approximate the contributions of neighbor nodes and further reduce the communication cost.

In this paper, we design a completely distributed particle filter for target tracking in WSNs, called CDPF, so as to minimize the communication cost. First, we develop a mechanism for maintaining particles on sensor nodes and propagating them along the target trajectory. Then we design CDPF by adjusting the order of PFs' four steps and leveraging the data aggregation during particle propagation. Finally, we introduce a neighborhood estimation method so that each node may replace the likelihood functions with approximate, estimated contributions of its neighbors, and eliminate the communication cost of measurement broadcasting. We compared CDPF with CPF and SDPF, the latter of which is a state-of-the-art effort that considers the distributed implementation of PFs from the perspective of network architecture and protocol of WSNs. Our experimental simulation studies show that, compared with SDPF, CDPF reduces the communication cost by 90%, with about 50% of the tracking error increment as the cost. Like most existing literature on Bayesian estimation and particle filters, we study the tracking problem in a possibly continuous dynamic system using the discrete-time approach [3].

The primary contribution of this paper is that, we provide a completely distributed implementation of generic PFs, which minimizes the communication cost. This makes it possible to fully leverage the advantages of PFs for WSNs. To the best of our knowledge, this is the first ever implementation of a completely distributed PF, without any special efforts for the centralization-demanding operations, such as weight aggregation.

The rest of the paper is organized as follows. In Section II, we introduce the motivations and describe our models. We discuss the mechanism of particle maintenance and propagation in Section III. In Section IV, we present the CDPF algorithm. Then in Section V, we introduce the neighborhood estimation method and its effect on reducing the communication cost. We report our experimental evaluation results in Section VI. Related work is summarized in Section VII, and we conclude the paper in Section VIII.

II. Preliminaries

In this section, we start from reviewing generic particle filters and the algorithm of its centralized implementation. Then we introduce our motivations based on the analysis of the communication overhead and finally describe our models.

A. Generic Particle Filter—Centralized Implementation

The tracking problem is usually formulated as a dynamic system in the state space $\{\mathbf{x}_k,\ k \in \mathbb{N}\}$, where k is the index of discrete time. This dynamic system includes a state transition model and a measurement (or observation[1]) model [3]:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \\ \mathbf{z}_k &= \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \end{aligned} \tag{1}$$

where $\mathbf{f}_k$ and $\mathbf{h}_k$ are possibly nonlinear functions, $\{\mathbf{v}_{k-1},\ k \in \mathbb{N}\}$ and $\{\mathbf{n}_k,\ k \in \mathbb{N}\}$ are i.i.d. process noise and measurement noise sequences respectively, and $\{\mathbf{z}_k,\ k \in \mathbb{N}\}$ is a sequence of observations at time k.

Assuming that states $\{\mathbf{x}_k,\ k \in \mathbb{N}\}$ follow a first order Markov process and observations depend only on the states, the posterior probability density $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ (i.e., the degree-of-belief in the state $\mathbf{x}_k$) can be recursively estimated in two steps—prediction and update:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1} \tag{2}$$

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \tag{3}$$

For this tracking problem, particle filters are one of the approximate approaches when the analytic solution is intractable. Particle filters represent the posterior pdf with a number of particles with associated weights, and make estimations based on these weighted particles. A generic particle filter, i.e. sequential importance sampling (SIS) algorithm, runs in an iterative manner. After the initialization step that draws N_s particles for the first time $t = 0$, each iteration consists of the following four steps [9]:

1) Prediction—draw N_s particles for time k from an importance density $q(\mathbf{x}_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k)$;

2) Update—assign and normalize a weight w^i_k for each particle;

3) Resampling (optional)—eliminate particles with low importance weights and multiply those with high importance weights so as to reduce the degeneracy effect;

[1]The terms "measurement" and "observation" will be used alternatively whenever there is no ambiguity.

4) Estimation—calculate the estimation $\hat{\mathbf{x}_k}$ based on the weighted particles.

Here $N_s \in \mathbb{N}$ is the number of particles, $i = 1, \ldots, N_s$ is the index of particles, w_k^i is the weight of particle $\mathbf{x}_k^i$ at time k, and $\hat{\mathbf{x}_k}$ is the estimated $\mathbf{x}_k$.

Sampling importance resampling (or SIR) filters are derived from generic particle filters by choosing the prior density $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ as the importance density $q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k)$, and resampling in every iteration [3].

B. Motivations

As discussed in Section I, few existing efforts on DPF were implemented in a completely distributed manner, because the weights of particles need to be transmitted to and aggregated somewhere. Next we discuss the potential improvements on reducing the communication cost if we implement a completely distributed particle filter.

It was shown in [10] that the communication workload of centralized particle filters at each iteration is $\sum_{i=1}^{N} D_m H_i$, where N is the number of sensor nodes with measurements, D_m is the data amount of a measurement message and H_i is the number of hops that D_m data needs to be propagated to the computational center. Then, we have the communication complexity of CPFs as $O(ND_mH_{max})$, where $H_{max} = \max_{i \le N} H_i$.

Coates elaborated the achievable compression on the disseminated raw data of particles, i.e., compressing D_m either by training parametric models or with adaptive encoding in [10]. Like the communication cost of CPFs, the achievable communication cost of DPF is $O(NPH_{max})$, where P is the data amount of each compressed measurement message. This result only provides a possibility of reducing the total data amount of communication if $P \ll D$. In fact, the number of communication messages is equal to or even higher than that of CPFs (due to the backward parameter exchange). Thus the efficiency of DPF completely depends on the compression efficiency on the raw data.

In [7], Coates and Ing developed a semi-distributed particle filter, named SDPF, in which particles are maintained on different sensor nodes and weight aggregation is completed on a global transceiver. The communication of SDPF consists of three parts: particle propagation, measurement sharing and weight aggregation/dissemination. First, the particles maintained on different sensor nodes are propagated in the predicted direction of the target at each iteration. As each sensor node that maintains particles needs to broadcast a message containing both particles and their weights to its neighbors within one hop, the communication cost is $\sum_{i=1}^{N_n} N_i(D_p + D_w) = (D_p + D_w)\sum_{i=1}^{N_n} N_i = N_s(D_p + D_w)$. Here N_n is the number of sensor nodes that are maintaining a subset of particles, N_i $(N_i \ge 1)$ is the number of particles maintained on sensor node i, thus $\sum_{i=1}^{N_n} N_i = N_s$. Moreover, we denote the data amount for each particle D_p (the subscript p represents "particle"), and the data amount of a particle's weight D_w (the subscript w represents "weight"). Secondly, the measurements of neighbor nodes are shared locally among N_n nodes. Then the communication cost will be $\sum_{i=1}^{N_n} D_m = N_n D_m \dashv N_s D_m$, where we mean "bounded by" with $\dashv$. Thirdly, at each iteration, each sensor node that maintains particles needs to transmit the weights of particles on it to a global transceiver, which is assumed to be one hop away from every node in the network. After a three-way query-response handshaking, the transceiver sends the calculated total weight back to active nodes. The communication cost during the whole aggregation process is $\sum_{i=1}^{N_n} N_i D_w + 2 = N_s D_w + 2$, where 2 comes from the two broadcast messages from the transceiver. Therefore, the total communication cost of SDPF is $N_s(D_p + D_w + D_m + D_w) + 2 \approx N_s(D_p + D_m + 2D_w)$.

Table I
ANALYZED COMMUNICATION COSTS OF VARIOUS PFS

Particle filter methods	Communication costs
CPF	NDH_{max}
DPF	NPH_{max}
SDPF	$N_s(D_p + D_m + 2D_w)$
CDPF	$N_s(D_p + D_m + D_w)$

Unlike DPFs and SDPF, a completely distributed particle filter does not have to collect the particle weights for the aggregation. Based on the calculation for SDPF, the communication cost of such a PF like CDPF that we present in this paper will approximately be $N_s(D_p + D_m + D_w)$.

We compare the analyzed communication costs of four PFs in Table I. Obviously, SDPF and CDPF may significantly reduce the communication cost of CPFs and DPFs by constraining the communication within one hop. Except for this, CDPF eliminates the communication for weight aggregation completely, thereby achieves the minimal communication cost.

C. Models

1) Network Model: We consider a sensor network with a two-dimensional plane, where sensor nodes are randomly deployed and their static positions are known *a priori* via GPS [15] or using algorithmic strategies such as [16].

2) Sensor Model: For the communication, we adopt the protocol model introduced in [17], where both transmission and interference depend only on the Euclidean distance between nodes. Among the typical detection models (including instant detection, sampling detection, energy detection [18], and probabilistic detection [19]), we consider the instant detection model, i.e., a sensor node detects a target when the target's trajectory intersects the node's sensing area. In addition, we assume that the sensing radius of nodes (in which nodes can detect an event) is no greater than half of the communication radius (in which nodes can communicate

with each other). This is a reasonable assumption, as the radio of a node is usually much more powerful than its sensing devices. For example, the radio range of MICA2 is up to 150 m [20], which is very difficult to reach for most sensing devices.

3) Dynamic System Model: For the algorithm discussion, we do not make any specific assumptions for the dynamic system. The model used in the simulation will be introduced in Section VI.

III. Particle Maintenance and Propagation

For the sake of clarify, we first explicitly interpret the term "distributed" in DPFs. In the existing literature, it was defined in several different ways. For example in [10], "distributed" means that the aggregation of particles and their weights is completed in a distributed manner on different sensor nodes. Other operations, such as the calculation of factorized likelihood functions, the training of parametric models and measurement quantization, all serve for this purpose. Unlike [10], the term "distributed" in [7] was interpreted as meaning that disjoint subsets of particles are maintained on different sensor nodes.

We define "distributed" in CDPF following the interpretation of [7], i.e., particles on nodes. Its advantages include: 1) the computational workload may be distributed onto different sensor nodes; 2) particles are easy to manage and propagate; and 3) particles can be combined or divided based on node positions.

Next, we introduce the maintenance and propagation mechanism of particles.

A. Particle Maintenance

Like [7], we constrain particles to locate on sensor nodes only, thus each particle will automatically have the position of its "host" node that maintains it. This may increase the estimation error of PFs. But if nodes are deployed densely enough or an error bounded by the sensing radius is tolerable, the error increment will be less important.

We do not distinguish the different particles on the same node. Hence multiple particles on a single node may be combined to one particle, with the total weights of original particles as its weight. On the contrary, a single particle may also be divided into multiple ones during the propagation, which will be detailed in Section III-B. Though the number of particles N_s may vary when being combined and divided, this variable N_s is controllable. This is because that N_s corresponds to the number of sensor nodes that maintain particles and participate into filtering, while these nodes are always around the target trajectory thus will be bounded when given a certain deployment density.

B. Particle Propagation

At the initialization step, each node that first detects an intruding target is given a particle with a certain weight.

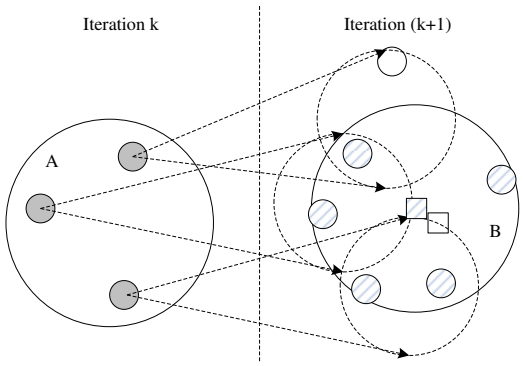

Figure 1. Particle propagation

This particle weight may be configured as a constant, or adaptively determined according to the received signal strength. In the following tracking process, these particles will be propagated along with the moving target.

Figure 1 shows the propagation of particles at iteration k. In the figure, small circles represent sensor nodes, squares mean the target positions, and dotted lines and circles signify the prediction and the direction of particle propagation. For the discussion convenience, we call the dotted circles "predicted areas". The other symbols in the figure will be introduced in Sections IV and V, where they are discussed.

Nodes always propagate particles towards the predicted target position, so that the particles in the current iteration can be reused in the next iteration, with their weights updated. A dynamic clustering mechanism as in [21] may be used for this purpose: a node broadcasts the particles on it to all of its neighbors, but only those that are highly likely to detect the target record the particles (i.e., nodes in predicted areas). We leverage the linear probability model in [21] to decide which neighbors should record the particles. If there are more than one node in the predicted area, a single particle will be divided into multiple ones, so will its weight. The weight is divided based on the following rule: 1) the total weight of divided particles is equal to the original particle's weight; and 2) the ratio of any pair of divided particles' weights is equal to the ratio of their host nodes' probabilities in the linear probability model.

Particle propagation from multiple source nodes may also overlap on neighbor nodes, e.g., nodes in the intersection of two predicted areas in Figure 1. In this case, particles from different source nodes will be combined into one on the receiving node.

It is possible that some nodes receive and record particles, but they are not able to detect the target at the next iteration, e.g., the blank node in Figure 1. Then, the weight update of particles on it will depend on the likelihood function. If the likelihood function shows zero or almost zero density, this node may drop the particle on it and stop broadcasting.

It is also possible that a node that does not receive any

propagated particles detects the target, e.g., the node outside of any predicted areas. Then a new particle will be created as in the initialization step.

C. Node Scheduling

Sensor nodes need to be scheduled for maintaining and propagating particles. In a duty-cycled WSN [13], nodes around the predicted target position may be in the sleep state when a target is approaching. Then, it needs to be proactively awakened so as to receive the propagated particles. We leverage TDSS sleep scheduling algorithm presented in [21]: nodes around the predicted target position are awakened to prepare for the approaching target, and the energy consumption could be reduced simultaneously.

IV. CDPF Design

Based on the mechanism of particle maintenance and propagation, we design the CDPF algorithm in this section. First, we partition the update step and adjust the order of filtering steps in CPFs. Then the CDPF algorithm will be specified.

A. Algorithm Design

Among the four steps of generic PFs, the prediction step may depend only on individual particles (e.g., by choosing the prior distribution as the importance density). Thus, every sensor node that maintains a subset of particles may complete the prediction step independently. However, all the other three steps (i.e., update, resampling and estimation) require collection of all the measurements and particle weights. To design CDPF, we need to develop a method to achieve the same objective without any extra communications.

Particle propagation designed in Section III provides us a feasible approach. Based on the overhearing effect, it is possible that every node in any of the predicted areas hears the particles from all the broadcasting nodes. Since we assume that the sensing radius of nodes is no greater than half of the communication radius, this goal is achievable as long as the propagation does not reach too far (i.e., the time interval of the dynamic system is not very long). Then every node that receives particles will receive all the particles, so that every node may obtain the total weight.

However, one problem of this approach is that the obtained total weight is for the previous iteration instead of the current one. Therefore, we have to adjust the order of four steps to produce a working scheme. Figure 2 shows the steps of both CPF and CDPF. We partition the "update" step into three sub-steps: 1) the "likelihood" step shares the measurements locally and calculates the likelihood functions; 2) the "assign weight" step assigns weights to particles; and 3) the "normalization" step calculates the total weight and normalizes the assigned weight. The dotted curves in Figure 2(a) signifies the reorder direction: we move

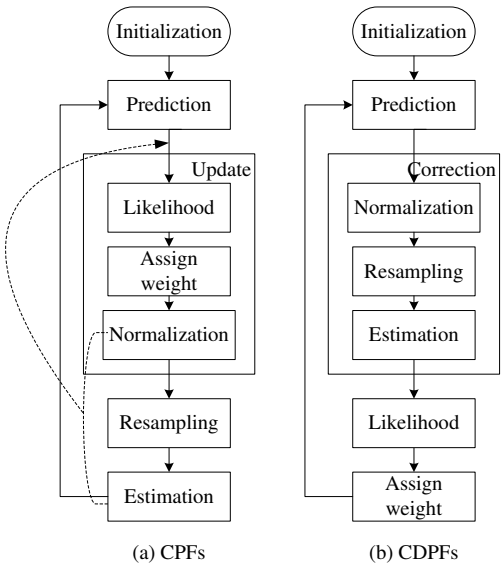

Figure 2. Steps of CPF and CDPF

normalization, resampling, and estimation steps forwards and insert them after the prediction step. The reordered steps for CDPF are shown in Figure 2(b).

After reordering the steps, we form normalization, resampling and estimation into a new step, named "correction". The correction step normalizes the updated weights, resamples and calculates the estimated target position for the previous iteration. Obviously, the correction step depends on the total weight that is aggregated by overhearing during particle propagation. This is the reason that we insert it after the prediction step.

Then the working process of CDPF will be:

1) Prediction—predict the target motion and propagate particles towards that direction.

2) Correction—normalize the propagated weights based on the total weight, resample, estimate the target position for the previous iteration, and possibly report it to sink nodes.

3) Likelihood—share the measurements at the current iteration locally and calculate the likelihood functions.

4) Assign weight—assign or update weights to particles based on the likelihood functions.

To update the weights, the measurement of each node should be shared locally with other nodes. Thus in the likelihood step, each node will receive the broadcast measurements from all of the neighbor nodes that are maintaining particles. In fact, this is also an approach to calculate the total weight without extra communication cost. However, we do not take this approach, because the broadcasting communication in this step can be eliminated, so that the communication cost can be reduced further. We will discuss the details in Section V.

In Figure 1, we drew two squares, meaning two possible positions of the target. Based on the correction step, we

now explain their difference. We use the blank square to represent the real position of the target, which is why the blank node cannot detect it. After the correction step, the estimated target position for the previous iteration will be obtained. Then we use the slashed square to represent the "predicted" target position based on this estimation. In fact, this cannot be called "prediction" any longer, because the current iteration has started. We use this term simply to show our calculation method. Then this predicted position will be an approximation to the real position, and we will use it to estimate the neighbor nodes' contributions and finally eliminate the likelihood function calculation in Section V.

B. Algorithm Details

Based on our previous design, we now detail an iteration of the CDPF algorithm in Algorithm 1.

Algorithm 1 CDPF algorithm at iteration $k+1$

1: Draw N_s samples for iteration $k+1$ from an importance density $q(\mathbf{x}_{k+1}|\mathbf{x}_k^i, \mathbf{z}_{k+1})$, i.e., propagate particles from iteration k to $k+1$;
2: Calculate the total weight by overhearing;
3: Normalize the received weights;
4: Resampling;
5: Make the estimation for iteration k;
6: Broadcast/receive measurements;
7: Calculate the likelihood function;
8: Assign/update a weight to the particle;

V. Improving CDPF: Neighborhood Estimation

As discussed in Section I, the local status in a WSN (including node positions, the topology and the detection capability of neighbor nodes) is relatively stable in the short term. Based on this feature, a sensor node may estimate the working status of its neighbor nodes thereby the contributions they may make to the target estimation. In this section, we develop an approximate estimation method to improve CDPF by further reducing the communication cost. We first discuss a prerequisite for the neighborhood estimation, which answers how a node may obtain information about its neighbors. Then, we introduce the estimation method and present the improved CDPF algorithm. Finally, we discuss the potential overhead that this estimation may introduce, and potential factors that may impact the estimation result.

A. Prerequisite

A prerequisite of neighborhood estimation is that local knowledge can be easily shared among the neighbor nodes. Basically whatever a node knows, its one-hop neighbors may easily know it via local direct communication. In many existing literature, short message exchange was commonly used for sharing local knowledge among neighbors, e.g., for updating routing information [22] or for maintaining synchronization [23]. Therefore we may reasonably assume that every sensor node knows all the detailed information about its one-hop neighbors, especially their positions.

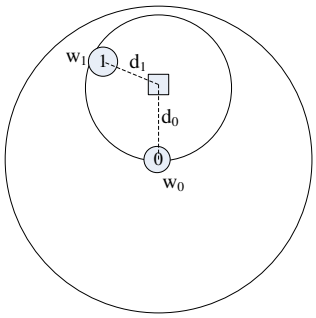

Figure 3. Neighborhood estimation

B. Estimation Method

Figure 3 shows a local topology including two sensor nodes (shown as small circles) and the predicted position of a target (the square). The large circle with a communication radius represents the one-hop neighbor area of node 0, and the middle circle with a sensing radius signifies the area in which nodes may detect the target. We define this middle circle as an estimation area:

Definition 1 (Estimation Area): In a two-dimensional plane, we define the estimation area as the circular area that is centered at a target's predicted position and has the sensing radius as its radius.

In Figure 1, the two large solid circles (labeled as A and B) are right the estimation areas of two iterations. Since we assume that the sensing radius of nodes is no greater than half of the communication radius, an estimation area will never exceed the scope of the large circle, i.e., the communication range of any sensor node within it. This means that the information of all the nodes that may detect the target thereby participate into particle filtering can be shared with each other.

In Figure 3, we let d_0 and d_1 denote the distances of two nodes from the predicted target position, and c_0 and c_1 denote the contributions of nodes 0 and 1 respectively. Based on the prerequisite, the positions of nodes 0 and 1 are known to each other, so is the predicted target position. Thus, both nodes may easily calculate d_0 and d_1.

We set the contribution of a node for a specific target inverse proportional to its distance from the target. Then, the weighted distance of any nodes from the target in the estimation area will be constant. We argue that this model is reasonable, as the closer a node is to the target, the more contribution it will make for estimating the target feature, i.e., the more information users may obtain from it. In fact, this is also intuitive in terms of the idea of PFs: when

particles are maintained on sensor nodes, the closer a node is to the target, the more weight the particle maintained on it should have. Therefore, we setup the following proportion equation:

$$c_0 d_0 = c_1 d_1 = \epsilon \tag{4}$$

where ϵ is a constant.

By assuming that $c_0 = 1$, node 0 will obtain the relative contribution of its neighbor node 1 as $c_1 = \frac{d_0}{d_1}$. Similarly, node 1 will also obtain the relative contribution of its neighbor node 0 as $c_0 = \frac{d_1}{d_0}$ when assuming $c_1 = 1$. From now on, we only discuss the estimation process of node 0. According to the symmetry, each node in the local area may complete the same estimation process. The only difference would be the values of these relative contributions.

Since node 0 may obtain such an estimation for each of its one-hop neighbors, we assume that all the estimated contributions form a set $\{c_0, c_1, \ldots, c_m\}$, where m is the number of its one-hop neighbors. Then the normalized contributions will be $\{c_0/C, c_1/C, \ldots, c_m/C\}$, where $C = 1 + \sum_{i=1}^{m} c_i$. We define the estimated neighbor contributions as the following:

Definition 2 (Estimated Neighbor Contributions): Within an estimation area, the contributions of neighbor nodes that are estimated by node 0 *are defined as:*

$$\{c_0, c_1, \ldots, c_m\} = \{\frac{1}{d_0 \cdot D}, \frac{1}{d_1 \cdot D}, \ldots, \frac{1}{d_m \cdot D}\}$$

where c_0 *represents the contribution of node* 0*,* c_i $(1 \leq i \leq m)$ *are the contributions of* m *other neighbor nodes in the estimation area,* d_i $(0 \leq i \leq m)$ *are the distances of each node from the predicted target position, and* $D = \sum_{i=0}^{m} \frac{1}{d_i}$.

Based on this definition, we may easily prove the following two propositions are true:

1) The estimated neighbor contributions are normalized; and

2) When the shared node positions and the predicted target position are consistent on all the nodes, so will the contributions estimated by all the nodes.

Theorem 1: The estimated neighbor contributions are normalized.

Proof: First, the total contribution from Definition 2 is equal to 1:

$$\sum_{i=0}^{m} c_i = \sum_{i=0}^{m} \frac{1}{d_i \cdot D} = \frac{1}{D} \sum_{i=0}^{m} \frac{1}{d_i} = \frac{1}{D} \cdot D = 1$$

Secondly, the ratio of any two contributions follows the model in Equation 4.

Hence, all the defined contributions are normalized. ■

Theorem 2: When the shared node positions and the predicted target position are consistent on all the nodes, so will the contributions estimated by all the nodes.

Proof: To prove this proposition, we only need to prove that a node's contribution estimated by itself is equal to that estimated by any other node in the estimation area. Without loss of generality, we evaluate the contribution of node 0 estimated by itself and node 1.

According to Definition 2, node 0's contribution estimated by itself is $\frac{1}{d_0 \cdot D}$. At the same time, its contribution is estimated by node 1 as $\frac{1}{d_0 \cdot D}$. Since the shared node positions and the predicted target position are consistent on all the nodes, either d_0 or D will be consistent in both results. Therefore, the two results are identical. ■

C. Improved CDPF

The result of this neighborhood estimation can replace the measurement sharing and likelihood function calculation, i.e., the likelihood step in Figure 2(b) or steps 6 and 7 in Algorithm 1. The detailed method is:

1) Each node in the estimation area estimates the contributions of itself as well as its neighbors.

2) Based on Definition 2, each node updates the particle weight as $w_{k+1} = w_k \cdot c_0$.

We name this improved version CDPF-NE, where the suffix "NE" represents neighborhood estimation. In this way, c_0 replaces the likelihood function (in case that the proposal function is chosen as the prior). Therefore broadcasting for measurement sharing could be completely eliminated. The analyzed communication cost of CDPF in Table I will then become $N_s(D_p + D_w)$, i.e., the only communication cost left is for particle propagation. Based on the architecture of "particles on nodes", this communication cost is already the minimum.

D. Discussion

First, we discuss the frequency of this estimation and its potential overhead. From the definitions above, we may observe that the local status used for neighborhood estimation mainly involves with node positions, the predicted target position, and the working status of neighbor nodes. First, the node positions never change in a static WSN. Even in a mobile WSN, nodes rarely move fast, either. Secondly, the predicted target position of CDPF is calculated by each individual node based on consistent data. So it is also consistent within the estimation area. Finally, the working status of neighbor nodes is subject to change. However, as long as the change can be anticipated, the estimation still can work correctly. For example, duty cycling is widely used [13] to reduce the energy consumption during idle listening, which is a major source of energy waste [24], thereby improve the network lifetime. With duty cycling, nodes are put into sleep states for most of the time, and only

awakened periodically. In certain cases, the sleep pattern of nodes may also be explicitly scheduled, via proactive wake-up [21], [25] for instance. No matter what sleep pattern is taken, the working status can still be anticipated as long as the pattern is certain.

Based on these conditions, we may hence exchange the local status of neighbor nodes and execute the neighborhood estimation at a low frequency, e.g., once per day, once per week or even longer. This will introduce little communication overhead, but gain much improvement on the communication efficiency for target tracking, especially in a WSN where target intrusion events are not rare.

Then we discuss the potential factors that may impact the estimation. According to the previous analysis, the most significant impacts are those uncertain factors, e.g, a random sleep pattern, unexpected node failure, mobile sensor nodes at a high speed, or overloaded nodes due to network congestion. These uncertain factors will propose more requirements on time synchronization. The level of synchronization is dependent on the impact level of these factors. For a real deployment with any of these uncertain features, CDPF-NE needs to be applied carefully. In addition, the estimation depends on the predicted target position. Thus, a wide prior distribution may result in a large error on the estimation result.

VI. Evaluation

We evaluated CDPF and CDPF-NE in Matlab and compared them with CPF and SDPF. This section reports our evaluation results using the communication cost as the overhead criterion, and root mean squared error (or RMSE) as the estimation correctness criterion.

A. Simulation Environment

The sensor network includes $2,000 - 16,000$ nodes in a two-dimensional plane, which are randomly deployed in a $200m \times 200m$ area. Thus, the node density is $5 - 40$ $nodes/100m^2$. The sensing radius of nodes is set as 10 m, and the communication radius is set as 30 m. A target crosses the surveillance field from the start point $(0, 100)$ with a constant speed 3 m/s. At each time step of 1 s, the target turns a random angle bounded by $[-15^o, +15^o]$.

We study the bearings-only tracking problem [26] in the simulation:

$$\begin{aligned} \mathbf{x}_k &= \Phi \mathbf{x}_{k-1} + \Gamma \mathbf{v}_{k-1} \\ z_k &= \arctan \tfrac{y_k}{x_k} + n_k \end{aligned} \tag{5}$$

where $\mathbf{x}_k = (\mathbf{s}_k, \mathbf{v}_k)^T = (x_k, y_k, x'_k, y'_k)^T$, z_k is the observed bearing, and

$$\Phi = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \Gamma = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

In addition, $\mathbf{v}_{k-1} = (v_x, v_y)^T_{k-1}$ and n_k are zero mean Gaussian white noises, and the variances of which are respectively $\sigma_v^2 = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$ and σ_n^2.

The detailed parameter configurations for the dynamic system above are as follows. The time step of CDPF is 5 s. The standard deviations of noises are $\sigma_x = \sigma_y = \sigma_n = 0.05$. The simulation includes 50 steps. For CPF, we adopt the number of particles $N_s = 1000$.

For all the four algorithms simulated in the experiments, i.e., CPF, SDPF, CDPF and CDPF-NE, we adopt SIR filters [3] as the basis: we use the prior distribution as the importance density, and execute the resampling step at every iteration.

B. Experimental Results

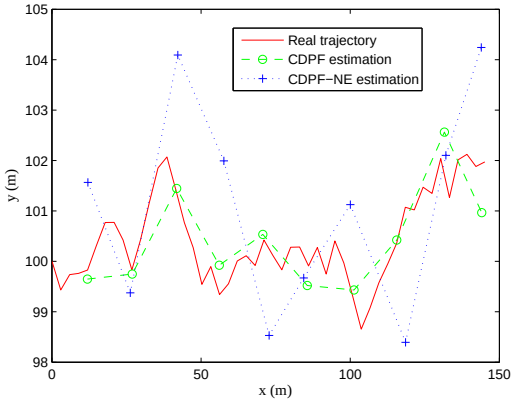

Figure 4. Estimation example

First in Figure 4, we show an estimation example including CDPF and CDPF-NE, when the node density is 20 $nodes/100m^2$. The real trajectory of the target is shown in a solid curve, which was simulated based on the target model. We may observe that the estimation error of CDPF-NE is a little greater than CDPF, as CDPF-NE replaces the measurement sharing with neighborhood estimation. However, the error of up to 3 m is still tolerable given the node density of 5 $m^2/node$.

Then we examine the communication costs of four algorithms in various node densities. Based on the dynamic model of the bearings-only tracking problem, we assume that a particle includes four integers, and either a measurement or a weight includes one integer only. On a 32-bit platform, we have $D_p = 16$, $D_m = 4$ and $D_w = 4$, all in bytes.

In Figure 5, the communication costs of all the four algorithms increase as the node density increases. This is because that the number of sensor nodes that detect the target and report the measurement increases. We observe that CDPF and CDPF-NE reduce the communication cost significantly, in which CDPF-NE achieves the minimal communication

overhead. Compared with SDPF, their reduction on the communication cost reaches up to 90%. If compared with CPF, they can also reduce the communication by about 70%. Except for their communication reduction efforts, another reason for this is that multiple particles on a single node can be combined into one, thus the data amount for propagation decreases significantly.

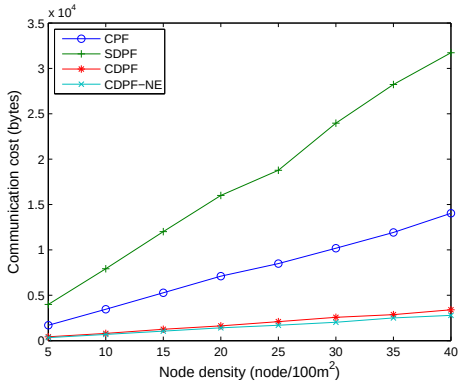

Figure 5. Communication cost

A counterintuitive observation is that the communication cost of SDPF is higher than that of CPF. This is caused by the network scale: in the configured network environment, any node can propagate the particle data to the sink node in the center of the network within four hops at the most. Thus, the hop count factor in the communication cost of CPF is not dominant. On the contrary, the eight particles on each node that detects the target increase SDPF's communication workload significantly. Therefore the two curves show a reverse relation. If the surveillance field is large enough so that the hop count factor dominates the communication cost, two curves are supposed to reverse their positions.

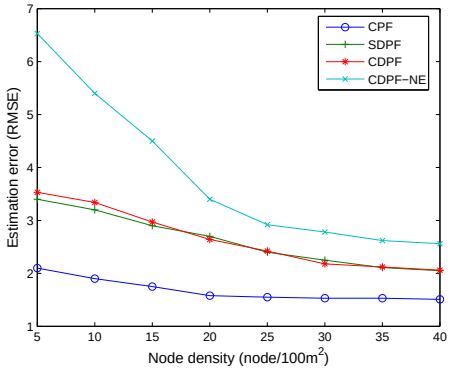

Figure 6. Estimation error

Finally, we present the result of the estimation error. In Figure 6, CDPF shows a similar RMSE to SDPF, as their operations on measurement sharing and particle propagation are similar. CDPF-NE shows the greatest estimation error, which is about 100% to 30% more than SDPF, as it simulates the likelihood function by estimating the contribution of neighbor nodes. However, the estimation error of CDPF-NE decreases faster than others, because the error difference will become less remarkable as the node deployment reaches a certain level of density. As long as a reasonably big estimation error can be tolerated, CDPF-NE would be the most efficient choice.

VII. Related Work

Bayesian estimation methods estimate the states in a dynamic system in an iterative manner, by incorporating new measures to filter the prior distribution to the posterior one. When certain constrains (including Gaussian process and measurement noises, and linear state transition and measurement functions) hold, Kalman filter [27] serves as the optimal solution by minimizing the estimated error covariance. If otherwise the dynamic system is nonlinear and/or non-Gaussian, which is usually true in real applications, particle filters [3] are usually used to approximate the optimal solutions.

Given the high computation/communication cost, it is often hard to apply PFs to WSNs. Many research efforts were conducted to either reduce the number of particles or compress the data amount of communication. In [28], the author applied KLD-sampling to adapt the number of particles dynamically, so that the estimation error is bounded at a given probability. Kwak *et. al.* introduced a heuristic algorithm based on a back-propagation neural network to adapt the sample size in [29]. However, these efforts still worked on centralized PFs, and did not consider a distributed implementation.

Compared with CPFs, the research for DPFs is much less mature. One of the most important reasons for this is that PFs are easy to be defined for centralized architectures, but difficult to be extended to distributed systems [30].

[10] is a widely cited literature about DPFs, in which Coates presented the achievable compression on particles either by training parametric models or with adaptive encoding. The compressed data, instead of the raw data, is propagated and aggregated throughout the network. This compressed data may be either parameters trained from a certain parametric model of the factorized likelihood, or quantized data by encoding the measurements. This work was the first one to complete the data aggregation step by step along with the propagation. Ing and Coates further improved the idea of adaptive encoding with Huffman tree in [12].

Although the training of a parametric model was proposed in [10], the author did not present a specific parametric model. Sheng *et. al.* provided one, i.e., Gaussian mixture model (or GMM), in [5]. The distributed algorithms are run over a set of uncorrelated sensor cliques, which are

dynamically constructed according to the moving trajectories of the target. With GMM, the measurement data may be compressed and aggregated efficiently.

Unlike [5], Liu *et. al.* introduced a non-parametric method named support vector machine (or SVM) in [9]. The raw data can also be compressed to reduce the communication cost.

All these DPF literature focused on completing the aggregation of particles in a distributed manner on different sensor nodes, and reducing the communication cost by compressing the raw data. These features will introduce the following two problems: 1) the aggregation of particle weights will experience a long delay, so will each iteration of PFs; and 2) though the total data amount is compressed, the number of communicated messages may remain or even increase. In addition, [10] strives to keep the computation result of each iteration consistent across the network, which is often unnecessary.

In [7], the authors presented a semi-distributed particle filter, which is the first to maintain particles on different sensor nodes. However, weight aggregation is still dependent on a global transceiver. Such kind of global transceiver, which is assumed to be able to communicate with all the nodes in the network directly, is usually hard to implement in real deployments. On the contrary, our CDPF algorithm removes all the weight aggregation operations and implements PFs in a completely distributed way, thereby minimizes the communication cost. Moreover, we remove many unnecessary assumptions in [7], such as binary proximity sensors, the optical communication and reflective devices.

Except for these DPF literature, Huang *et. al.* studied target tracking using DPFs in [6], which was an efforts of applying DPFs in specific scenarios. On the contrary, our work studies DPF methods instead of their applications.

VIII. Conclusion

Based on our analysis in Section II-B and the experimental evaluation in Section VI, we observe that compared with SDPF, CDPF can reduce the communication cost by 90%, with about 50% of the tracking error increment as the cost. This shows that the communication reduction effort of CDPF is significant. The application of CDPF's improved version is subject to several conditions, e.g., static nodes and stable working status of nodes. In many deployments, these conditions are easy to satisfy. Therefore, both CDPF and CDPF-NE can be widely utilized.

Potential future work directions include:

1) Evaluate CDPF's tolerance to uncertain factors. This would allow us to understand the application scope of CDPF and help with the configuration of WSN deployments.

2) Apply CDPF's idea to more PF branches. Except for generic PFs, there are many derivative efforts to solve related problems introduced by PFs, e.g., degeneracy problem, sample impoverishment. The idea of a completely distributed implementation may be applied in these areas to reduce communication costs.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 38, no. 4, pp. 393–422, 2002.

[2] M. Ding and X. Cheng, "Fault tolerant target tracking in sensor networks," in *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2009, pp. 125–134.

[3] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2001.

[4] A. Doucet, N. De Freitas, and N. Gordon, Eds., *Sequential Monte Carlo methods in practice*. New York, USA: Springer, 2001.

[5] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with gmm approximation for multiple targets localization and tracking in wireless sensor network," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*. Piscataway, NJ, USA: IEEE Press, 2005, p. 24.

[6] Y. Huang, W. Liang, H.-b. Yu, and Y. Xiao, "Target tracking based on a distributed particle filter in underwater sensor networks," *Wirel. Commun. Mob. Comput.*, vol. 8, no. 8, pp. 1023–1033, 2008.

[7] M. Coates and G. Ing, "Sensor network particle filters: motes as particles," in *IEEE Workshop on Statistical Signal Processing*, 2005, pp. 1152–1157.

[8] X. Wang, J. Ma, S. Wang, and D. Bi, "Distributed energy optimization for target tracking in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 73–86, 2010.

[9] H.-Q. Liu, H.-C. So, F. K. W. Chan, and K. W. K. Lui, "Distributed particle filter for target tracking in sensor networks," *Progress In Electromagnetics Research*, vol. 11, pp. 171–182, 2009.

[10] M. Coates, "Distributed particle filters for sensor networks," in *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*. New York, NY, USA: ACM, 2004, pp. 99–107.

[11] N.-L. Lai, C.-T. King, and C.-H. Lin, "On maximizing the throughput of convergecast in wireless sensor networks," in *GPC'08: Proceedings of the 3rd international conference on Advances in grid and pervasive computing*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 396–408.

[12] G. Ing and M. J. Coates, "Parallel particle filters for tracking in wireless sensor networks," in *Proceedings of IEEE 6th Workshop on Signal Processing Advances in Wireless Communications*, 2005, pp. 935 – 939.

[13] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, 2007, pp. 321–334.

[14] P. Basu and J. Redi, "Effect of overhearing transmissions on energy efficiency in dense sensor networks," *Third International Symposium on Information Processing in Sensor Networks (IPSN), 2004.*, pp. 196–204, 2004.

[15] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, August 2001.

[16] R. Stoleru, J. A. Stankovic, and S. H. Son, "Robust node localization for wireless sensor networks," in *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, 2007, pp. 48–52.

[17] P. Gupta and P. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 388–404, 2000.

[18] L. Lazos, R. Poovendran, and J. A. Ritcey, "Probabilistic detection of mobile targets in heterogeneous sensor networks," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2007, pp. 519–528.

[19] J. Lin, W. Xiao, F. L. Lewis, and L. Xie, "Energy-efficient distributed adaptive multisensor scheduling for target tracking in wireless sensor networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 6, pp. 1886–1896, 2008.

[20] CrossBow, "Mica2 data sheet," http://www.xbow.com.

[21] B. Jiang, K. Han, B. Ravindran, and H. Cho, "Energy efficient sleep scheduling based on moving directions in target tracking sensor network," in *IPDPS*, 2008, pp. 1–10.

[22] Y. M. Lu and V. W. S. Wong, "An energy-efficient multipath routing protocol for wireless sensor networks: Research articles," *Int. J. Commun. Syst.*, vol. 20, no. 7, pp. 747–766, 2007.

[23] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *IEEE Infocom*, vol. 3, 2002, pp. 1567–1576.

[24] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, March 2005, pp. 2470–2481.

[25] J. Fuemmeler and V. Veeravalli, "Smart sleeping policies for energy efficient tracking in sensor networks," *Signal Processing, IEEE Transactions on*, vol. 56, no. 5, pp. 2091–2101, May 2008.

[26] W. R. Gilks and C. Berzuini, "Following a moving target-monte carlo inference for dynamic bayesian models," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 63, no. 1, pp. 127–146, 2001.

[27] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *Decision and Control, 2007 46th IEEE Conference on*, Dec. 2007, pp. 5492–5498.

[28] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1003, 2003.

[29] N. Kwak, I.-K. Kim, H.-C. Lee, and B.-H. Lee, "Adaptive prior boosting technique for the efficient sample size in fast-slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 630–635.

[30] M. Rosencrantz, G. Gordon, and S. Thrun, "Decentralized sensor fusion with distributed particle filters," in *Proceedings of Uncertainty in Artificial Intelligence Acapulco*, 2003.