Research Article

Distributed Particle Flow Filter for Target Tracking in Wireless Sensor Networks

Junjie Wang , Lingling Zhao , and Xiaohong Su

School of Computer, Science and Technology, Harbin Institute of Technology, Harbin, China

Correspondence should be addressed to Lingling Zhao; zhaoll@hit.edu.cn

Received 28 July 2017; Revised 12 December 2017; Accepted 8 January 2018; Published 11 April 2018

Academic Editor: Hana Vaisocherova - Lisalova

Copyright © 2018 Junjie Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose, in this paper, a fully distributed tracking algorithm based on particle flow filter over sensor networks based on the max-consensus. The presented distributed particle flow filter is particularly suitable for the sensor network with limited sensing range and consists of two phases: the estimation phase and consensus phase. The local estimation results are obtained via particle flow filter in the estimation phase; then the sensor nodes agree on the best estimation based on max-consensus protocol in the consensus phase. Numerical simulations and comparisons with other distributed target tracking algorithms are carried out to show the effectiveness and feasibility of our approach.

1. Introduction

Distributed target tracking focuses on using a group of sensors to collect and process information about environment status. Compared with the central target tracking, the distributed target tracking has the following characteristics: scalability, flexibility, robustness, and fault tolerance. Due to these characteristics, the distributed target tracking has played an important role in many applications such as pedestrian tracking [1], biology [2], and environmental monitoring [3].

The distributed target tracking algorithms can be classified into three types: fusion center (FC) based, leader agent (LA) based, and consensus based [4]. In the FC-based approaches, each sensor node uses the local measurement to estimate the local states by filtering algorithms and then transmits the local estimation to a single FC, where a global estimate is calculated based on all the local estimates. In a LA-based filter, only a subset of sensors are activated in a special manner and the information about the target is accumulated along a path formed by selected sensors. While in a consensus-based filter, all sensor nodes are simultaneously active and process the local data to get local posterior; then

each one communicates with its neighboring agents [4] to agree with global posterior using consensus algorithms.

The Kalman filter is an optimal target tracking algorithm in the linear Gaussian situation. Recently, the distributed Kalman filter (DKF) [5] for the track-to-track fusion has been proposed as an optimal solution. However, the track-to-track DKF needs to handle the multiple information paths. Another well-known distributed Kalman filter is the Kalman consensus filter (KCF) [6] based on the consensus fusion algorithm which can avoid addressing the multiple information paths. The KCF performs well when all the nodes can get the measurement of a target. In the realistic scenario with limited sensing capability of sensors, some nodes become naive about the target state at some time instants [7]; the performance of the KCF will deteriorate as each node weighs its neighbors' estimates in an equal manner. To overcome this issue, the generalized Kalman consensus filter (GKCF) [7] was proposed utilizing the weighted averaging consensus. A node selection [8, 9] strategy was proposed to select the best estimate and propagate it to all the nodes rather than fusing all the sensor-estimated results. The distributed Kalman filter with node selection [9] selects the most accurate estimate to propagate through the network

rather than fusing all the sensor-estimated results. The distributed Kalman filters have the desirable property of computational simplicity in linear systems, but it is still needed to develop distributed target tracking algorithms for nonlinear non-Gaussian systems.

The distributed particle filters (DPFs) [4, 10–12] have been proposed to track a target in a wireless sensor network for the nonlinear non-Gaussian system. In [11], a distributed particle filter computes the product of likelihood function over the network using iterative average consensus. A kind of information-weighted consensus-based distributed particle filter [12] can avoid the divergence of the consensus error introduced by the naive nodes, but it performs in a low convergence rate. In the distributed particle filter proposed in [13, 14], the local posterior probability density function (PDF) is approximated by a Gaussian distribution. Then the local PDF parameters are fused into the global posterior PDF's parameters by average consensus. The work in [15] approximates the local likelihood functions by a Gaussian function and builds the global likelihood through exchange of information with neighboring nodes.

The DPFs based on averaging consensus mentioned above have two defects. The first drawback is that they require a large number of particles or samples for a given level of accuracy. To address this drawback of DPFs, we adapt another nonlinear filter, namely, particle flow filter proposed by Daum and Huang [16–19] to estimate the target state. The particle flow filter can achieve a good performance with fewer particles compared with the particle filter, especially in the high-dimensional state space. The principle behind the particle flow filter is to sample a set of particles from the prior distribution and use a stochastic method to move them such that they are then distributed according to the posterior. In other words, particles are migrated smoothly using a particle flow derived from a log homotopy relating the prior and the posterior [20]. Compared with the particle filter, the particle flow filter can yield a significant reduction of the number of particles especially in the high-dimensional case. Another issue of DPFs based on averaged consensus is that they are not suitable for the scenario which there exist some naive nodes in the wireless sensor network (WSN). In this scenario, it may happen that only a minority of sensors have measurements. Therefore, the WSN involves many unreliable sensors in the fusion step which may cause a divergent error. This is because of the fact that the average consensus algorithm gives all the nodes equal weights; even the naive nodes get less information about the target.

In this paper, inspired by the particle flow filter framework in [17] and the fusion rule in [21], we propose a distributed particle flow filter (DPFF) algorithm for WSN. We approximate local posterior as a Gaussian distribution and fuse the local posterior via a max-consensus protocol. To address the challenge in limited energy and sensor range of sensor nodes, the particle flow filter [17] is utilized to approximate local posterior. Also, the proposed DPFF seeks consensus on the best local posterior, rather than on the average of local posterior. To the best of our knowledge, particle flow filter has not been yet investigated in WSNs.

FIGURE 1: Abstract representation of the distributed tracking scenario. Four nodes (s_a , s_b , s_c , and s_d) are randomly placed in the tracking environment. Each node has a local sensing area (dotted circles around the node). The target trajectory is represented by the dashed line. The communication topology is denoted by the solid black line between nodes.

The rest of the paper is organized as follows. Section 2 introduces the background of the sensor network model and consensus theory. The details of DPFF are presented in Section 3. Section 4 evaluates the DPFF performance via two numerical examples. Finally, Section 5 gives the conclusion of this paper and the possible future work.

2. Background

2.1. Network Model. We consider a single-target tracking in the WSN which consists of N_c sensors with restricted monitoring area and communication ranges. Assume that the WSN can be modeled as an undirected connected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, in which each vertex $c \in \mathcal{C}$ represents a sensor node and each edge $\{i, j\} \in \mathcal{E}$ denotes the link between different nodes. If an edge $\{i, j\} \in \mathcal{E}$ denotes that the node j can get information from the node i and vice versa, the set of neighbors of node i is denoted as $\mathcal{N}_i = \{j \in \mathcal{C} : \{j, i\} \in \mathcal{E}\}$.

Figure 1 shows a network with four nodes. The distributed WSN has no central unit, and, thus, the sensors locally process their measurements. We restrict the monitoring region of each sensor node defined as an area within a dotted circle of radius ρ , in the sense that a sensor may not detect the target over the tracking period. Also, we assume that each sensor can only directly communicate with its neighbors in a certain communication range.

2.2. Target Model. The state of a target is represented by a vector $\mathbf{x}$. For a maneuvering target, the state vector $\mathbf{x}$ contains the information about the position and velocity. The dynamic transition of a target is modeled as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{w}_k, \quad (1)$$

where f is the state transition function and $\mathbf{w}_k$ is the processing noise.

At time k , each sensor can only obtain measurement when a target appears in its sensing area. The measurement function of sensor i is

$$\mathbf{z}_k^i = h_k^i(\mathbf{x}_k) + \mathbf{v}_k^i, \quad (2)$$

where h_k^i denotes the observation function for the i th sensor node and $\mathbf{v}_k^i$ stands for the measurement noise. The measurements of nodes are assumed independent over the network.

2.3. Max-Consensus. The max-consensus [21] is a well-known distributed algorithm which makes all the sensor nodes agree on the maximum of the value of their initial state through finite iterations. In the max-consensus algorithm, each sensor node initializes its state value as $u_i(0) = u_i$ and iteratively communicates with its neighbors $\mathcal{N}_i$ based on the update rule as follows to update its state:

$$u_i(t+1) = \max_{j \in \mathcal{N}_i \cup i} (u_j(t)), \quad (3)$$

where u_i is the state value of node i and t is the iterative step index.

According to [21], we define the max-consensus as follows:

Definition 1 (max-consensus). Consider a WSN with $\mathcal{C}$ nodes, connected over an undirected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$. Each node has a state variable $u_i(0)$, $i \in \mathcal{C}$. The discrete time max-consensus protocol is defined as

$$u_i(\mathbf{x}) = u_j(k) = \max\{u_1(0), \dots, u_n(0)\} \quad \forall k \geq K, \quad \forall i, j \in \mathcal{C}. \quad (4)$$

If (4) holds for all $u_0 \in \mathbb{R}^n$, strong max-consensus is achieved. If (4) only holds for a subset of initial states, weak max-consensus is achieved [22].

2.4. Average Consensus-Based Distributed Particle Filter in WSNs. The average consensus-based distributed particle filter consists two steps: local particle filter and average consensus filter. The local particle filter uses the local observation to get the local estimation. The output of the local particle filter is the local posteriori approximated as a Gaussian distribution $\mathcal{N}(\cdot | m_k^{i,\text{local}}, \mathbf{P}_k^{i,\text{local}})$. For the sake of using outputs of local filter among networks more effectively, each node maintains an average consensus filter. The aim of the average consensus filter is to fuse these local posteriori between neighbor nodes. According to [13], node i at time k , based on the local measurement $\mathbf{z}_k^i$, runs a local particle filter to obtain the parameters of local posterior $m_k^{i,\text{local}}, \mathbf{P}_k^{i,\text{local}}$ as follows:

$$\begin{aligned} m_k^{i,\text{local}} &= \sum_{j=1}^M w_k^{ij} \mathbf{x}_k^{ij}, \\ \mathbf{P}_k^{i,\text{local}} &= \sum_{j=1}^M w_k^{ij} \left(\mathbf{x}_k^{ij} - m_k^{i,\text{local}} \right) \left(\mathbf{x}_k^{ij} - m_k^{i,\text{local}} \right)^T, \end{aligned} \quad (5)$$

where w_k^{ij} and $\mathbf{x}_k^{ij}$ denote the weights and particles of the local particle filter, respectively. Then the average consensus algorithm is run between linked neighbor nodes as the following equation to obtain the global estimation results:

$$\mathbf{y}_{i,l+1} = \mathbf{y}_{i,l} + \epsilon \left[\sum_{j \in \mathcal{N}_i} \left(\mathbf{y}_{j,l} - \mathbf{j}_{i,l} \right) + \left(\mathbf{u}_{i,l} - \mathbf{y}_{i,l} \right) \right], \quad (6)$$

where ϵ is the updating rate. $\mathbf{y}_{i,l}$ can be represented as a parameter of local posterior such as $m_k^{i,\text{local}}$ or $\mathbf{P}_k^{i,\text{local}}$.

3. Distributed Particle Flow Filter

The average consensus-based distributed particle filter is robust to time-varying network topologies [4]. In the average-based distributed particle filter, each node runs a local particle filter to estimate the target state. This method inherits the drawbacks of the particle filter, namely, the curse of dimensionality, and particle degeneracy in the highly informative scenario. The other drawback of the particle filter is its requirement to maintain a large number of particles to attain good performance, leading to the waste of the sensors' energy.

Motivated by some desirable properties of the particle flow filter (see for example, [23, 24]) such as sufficient accuracy and low computational complexity, uniqueness of the solutions, we propose a consensus-based distributed particle flow filter (DPFF) algorithm, which consists of two main phases: *estimation* and *consensus*. According to the sensing range of sensors, the estimation phase can be divided into two branches. If a target is measured by the node i at time k , its estimation is carried out via a particle flow filter. On the contrary, the node can not detect the target at time k ; only the prediction part of the particle flow filter is run. Then each sensor computes its perception confidence value $\gamma_i(k)$ (refer to (15)) based on the estimated posterior covariance matrix or prior covariance matrix. In the consensus phase, a max-consensus algorithm is utilized to make all the sensors agree on the best estimated sensor.

Note that our algorithm requires the synchronization of clocks over the sensor networks. We now give the details of the two phases of DPFF at time k and assumed that each node obtains the best estimation results $(\bar{\mu}(k-1), \bar{\mathbf{P}}(k-1))$ over the WSN at time $k-1$.

3.1. Estimation Phase. In the DPFF, the posterior distribution is approximated by a particle set $\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_p}$ which is sampled from Gaussian distribution $\mathcal{N}(\cdot, \bar{\mu}(k-1), \bar{\mathbf{P}}(k-1))$ with same weights $w_{k-1}^i = 1/N_p$. Then all particles are transmitted to $\{\mathbf{x}_k^i\}_{i=1}^{N_p}$ at time step k by the dynamic model. Therefore, the prior distribution can be represented by particles $\{\mathbf{x}_k^i\}_{i=1}^{N_p}$.

The overall process of the estimation phase for each node is outlined in Figure 2. As a target may move in or out of the sensing area of the node i , if the sensor i has detected the target in its sensing range, then its estimation is carried out by the particle flow filter (Algorithm 1). The

FIGURE 2: The block diagram of the estimation phase for each node.

local output of each node is the local posterior approximated as a Gaussian distribution. On the contrary, in the case that the measurement is not available to the node i , then only the prediction of the particle flow filter will be executed (lines 1–3 of Algorithm 1).

In the following, the particle flow filter is presented in details. To avoid confusion, we will omit particle state indices. Particle flow filter is used to guide particles by the current measurement so that they can more accurately approximate the posterior distribution. Let $h(\mathbf{x}_k) = p(y_k | \mathbf{x}_k)$ and $g(\mathbf{x}_k) = p(\mathbf{x}_k | y_{1:k-1})$ denote the likelihood and prior functions, respectively. The log homotopy is defined as

$$\log(p(\mathbf{x}, \lambda)) = \log g(\mathbf{x}_k) + \lambda \log(h(\mathbf{x})) - \log(K(\lambda)), \quad (7)$$

where λ is a real number that varies from 0 to 1. $K(\lambda)$ is the normalization constant. When $\lambda = 0$, we obtain the predicted density function $g(\mathbf{x}_k)$, and when $\lambda = 1$, we get the posterior distribution. Suppose that the flow of particles is guided by the Bayes' rule according to

$$d\mathbf{x} = f(\mathbf{x}, \lambda) + dw. \quad (8)$$

Combined by (7), $f(\mathbf{x}, \lambda)$ can be computed by the Fokker-Planck equation

$$\frac{\partial p(\mathbf{x}, \lambda)}{\partial \lambda} = -\nabla \cdot [f(\mathbf{x}, \lambda), p(\mathbf{x}, \lambda)] + \frac{1}{2} \text{tr} \left[Q \frac{\partial^2 p(\mathbf{x}, \lambda)}{\partial \mathbf{x}^2} \right], \quad (9)$$

where Q is the covariance of the process noise. For simplicity, it is assumed that $Q = 0$. According to (7) and (9), the following equation can be derived:

$$\log h(\mathbf{x}) + [\nabla \log p(\mathbf{x}, \lambda)]^T \cdot f(\mathbf{x}, \lambda) = -\nabla f(\mathbf{x}, \lambda). \quad (10)$$

Assuming that $g(\mathbf{x}), h(\mathbf{x})$ are Gaussian PDFs, then a closed-form solution termed as exact flow filter [25] can be derived as

$$\frac{d\mathbf{x}}{d\lambda} = A(\lambda)\mathbf{x} + b(\lambda), \quad (11)$$

where

$$A = -\frac{1}{2} \mathbf{P} \mathbf{H}^T (\lambda \mathbf{H} \mathbf{P} \mathbf{H}^T + R)^{-1} \mathbf{H}, \quad (12)$$

$$b = (I + 2\lambda A) [(I + \lambda A) \mathbf{P} \mathbf{H}^T R^{-1} z + A \bar{\mathbf{x}}]. \quad (13)$$

Let $\bar{\mathbf{x}}$ and $\mathbf{P}$ represent the predicted mean vector and the prior covariance matrix, respectively. $\mathbf{H}$ denotes the measurement function matrix and R is the covariance of the measurement noise. For nonlinear models, the measurement function matrix $\mathbf{H}$ can be obtained by linearization of the measurement model. For more details on the implementation and analysis of the exact flow filter, please refer to [20, 25]. We summarize the exact flow filter in Algorithm 1.

It is important to note that there exist several different realizations for the particle flow filter, such as nonzero diffusion particle flow filter [26], incompressible flow filter [16], and exact Daum and Huang (EDH) filter [18]. In some cases, the Gromov's method was explained in [27–29], which would improve the particle flow filter performance. The performance of the particle flow filter also is influenced by the discretization of pseudotime λ . As the particle flow filter is described by an ordinary differential equation, a suitable discretization is essential to capture the flow dynamics [30]. In this paper, we set a sequence of discrete steps with uniform step size $\Delta\lambda = 0.1$.

In summary, when the sensor node received the measurement, the node can obtain the estimation result $\{\hat{\mu}_i(k), \hat{\mathbf{P}}_i(k)\}$ according to the particle flow filter (Algorithm 1). On the other hand, if the target is not detected by the i th sensor node, the target state is estimated according to the particles $\{\hat{\mathbf{x}}_k^{ij}\}_{j=1}^{N_p}$ from prediction. Then the estimated state and error covariance matrix can be computed as

$$\hat{\mu}_i(k) = \frac{1}{N_p} \sum_{j=1}^{N_p} \hat{\mathbf{x}}_k^{ij}, \quad (14)$$

$$\hat{\mathbf{P}}_i(k) = \frac{1}{N_p} \sum_{j=1}^{N_p} \left(\hat{\mathbf{x}}_k^{ij} - \hat{\mu}_i(k) \right) \left(\hat{\mathbf{x}}_k^{ij} - \hat{\mu}_i(k) \right)^T.$$

In order to measure the quality of the estimated target state of each node, the perception confidence value $\gamma_i(k)$

Input: $\bar{\mu}_i(k-1), \bar{\mathbf{P}}_i(k-1)$ **Output:** $\hat{\mu}_i(k), \hat{\mathbf{P}}_i(k)$

- 1: redraw particles $\{\mathbf{x}_i^j(k)\}_{j=1}^{N_p}$ from the Gaussian distribution which $\bar{\mu}_i(k-1), \bar{\mathbf{P}}_i(k-1)$ are the mean and covariance, respectively.
- 2: **for** $j = 1, \dots, N_p$ **do**
- 3: propagate particles $\mathbf{x}_i^j(k) = f_k(\mathbf{x}_i^j(k)) + \mathbf{v}_k$
- 4: compute $\mathbf{P}_i^-(k) = \text{cov}(\mathbf{x}(k))$
- 5: **for** $j = 1, \dots, N_\lambda$ **do**
- 6: Calculate the mean particles value $\bar{\mathbf{x}}_k$
- 7: set $\lambda = j\Delta\lambda$
- 8: linearize the measurement function at $\bar{\mathbf{x}}_k$ to get $\mathbf{H}_{\mathbf{x}_k}$
- 9: Calculate A and b from (12) and (13), respectively, using $\mathbf{P}_i^-(k), \bar{\mathbf{x}}_k, \mathbf{H}_{\mathbf{x}_k}$;
- 10: **for** $j = 1, \dots, N_p$ **do**
- 11: migrate particles $\mathbf{x}_i^j(k) = \mathbf{x}_i^j(k) + \Delta\lambda(A\mathbf{x}_i^j(k) + b)$
- 12: Compute the mean value and error matrix $\hat{\mu}_i(k), \hat{\mathbf{P}}_i(k)$ of the particles by (14)

ALGORITHM 1: Particle flow filter for node i at time k .

needs to be calculated at the end of the estimation phase. If the target is detected by the sensor i , $\gamma_i(k)$ is calculated based on the posterior error covariance matrix $\hat{\mathbf{P}}_i(k)$; otherwise, $\gamma_i(k)$ is calculated based on the prior error covariance matrix $\bar{\mathbf{P}}_i(k)$. The specific calculation formula of $\gamma_i(k)$ is as follows

$$\gamma_i(k) = \frac{1}{\text{trace}(\hat{\mathbf{P}}_i(k))}, \quad (15)$$

where $\text{trace}(\cdot)$ is the matrix trace operator. It is clear that $\gamma_i(k)$ grows with the reliability of the estimation performed by sensor node i at time k [22].

At the end of the estimation phase, each sensor node will obtain the value of perception confidence value $\gamma_i(k)$, the local estimate $\hat{\mu}_i(k)$, and the local error covariance matrix $\hat{\mathbf{P}}_i(k)$. These values can be employed to reach consensus in the consensus phase.

3.2. Consensus Phase. The aim of the consensus phase is to select the best estimation over the sensor networks and propagate the selected estimation with correlation information (error covariance matrix). It is noted that, in order to reduce the communication cost, we exchange the error covariance matrix and state estimate rather than the whole particle set. Therefore, at the next time step, each sensor node needs to redraw particles based on the best state estimate and error covariance matrix. The max-consensus for node i at time k is reported in Algorithm 2.

Algorithm 2 works as follows: node i obtains the values from the estimation phase, $\gamma_i(k), \hat{\mu}_i(k), \hat{\mathbf{P}}_i(k)$. And then sensor node i initializes its variables $\Gamma_i(0), \mathbf{U}_i(0), \prod_i(0)$ with $\gamma_i(k), \hat{\mu}_i(k), \hat{\mathbf{P}}_i(k)$, respectively. After initializing all variables, node i exchanges the variables $\Gamma_i(t-1), \mathbf{U}_i(t-1), \prod_i(t-1)$ with its neighbors (lines 5-6 of Algorithm 2). Then node i selects the max perception confidence value which corresponds to the best estimate from its neighbors (lines 7-8 of

Input: $\gamma_i(k), \hat{\mu}_i(k), \hat{\mathbf{P}}_i(k)$ **Output:** $\bar{\mu}_i(k), \bar{\mathbf{P}}_i(k)$

- 1: $\Gamma_i(0) = \gamma_i(k)$
- 2: $\mathbf{U}_i(0) = \hat{\mu}_i(k)$
- 3: $\prod_i(0) = \hat{\mathbf{P}}_i(k)$
- 4: **for** $t = 1, \dots, D$ **do**
- 5: send $\Gamma_i(t-1), \mathbf{U}_i(t-1), \prod_i(t-1)$
- 6: receive the information with connected sensors sets $\mathcal{N}_i$ to obtain $\Gamma_j(t-1), \mathbf{U}_j(t-1), \prod_j(t-1), \forall j \in \mathcal{N}_i \cup i$
- 7: $\Gamma_i(t) = \max_{j \in \mathcal{N}_i \cup i} \{\Gamma_j(t-1)\}$
- 8: $\alpha = \arg \max_{j \in \mathcal{N}_i \cup i} \{\Gamma_j(t-1)\}$
- 9: $\mathbf{U}_i(t) = \mathbf{U}_\alpha(t-1)$
- 10: $\prod_i(t) = \prod_\alpha(t-1)$
- 11: $\bar{\mu}_i(k) = \mathbf{U}_i(D)$
- 12: $\bar{\mathbf{P}}_i(k) = \prod_i(D)$

ALGORITHM 2: Max-consensus for node i at time k .

Algorithm 2). The node i will replace its estimation results to the corresponding estimate and covariance estimate matrices $\mathbf{U}_i(t)$ and $\prod_i(t)$ (lines 9 and 10, resp.). At the end of the phase, each sensor will agree with the best state estimate of the target $\bar{\mu}_i(k)$ with related error covariance matrix $\bar{\mathbf{P}}_i(k)$ (lines 12-13 of Algorithm 2). These two variables will be used in the next time $k+1$ of the estimation phase, in order to let the particle flow filter of each node start from the best estimation results and therefore to improve the algorithm's performance.

It can be proved [21] that the node converges during D steps, where D is the diameter of the sensor graph.

3.3. Convergence Analysis. In the DPFF algorithm, each node will converge within finite discrete steps through the max-consensus algorithm. The convergence of the DPFF is proved as follows:

Assuming that each node runs the DPFF algorithm, after consensus phases at time k , node i and node j hold the equation

$$\bar{\mu}_i(k) = \bar{\mu}_j(k) = \bar{\mu}_{\max}(k), \quad \forall i, j \in \mathcal{C}, \quad (16)$$

where $\mathcal{C}$ is the set of nodes and $\bar{\mu}_{\max}(k)$ is estimated mean value corresponding to the max perception confidence value $\gamma(k)$.

In each iteration of Algorithm 2, line 7 is to select the max $\gamma(k)$ from the neighbor nodes which is the update rule for the max-consensus algorithm. The max-consensus algorithm is guaranteed to converge in a finite number of iterations. Therefore, in D steps, Algorithm 2 will be guaranteed that

$$\Gamma_i(n) = \Gamma_j(n) = \max_{l \in \mathcal{J}} \gamma_l(k), \quad \forall i, j \in \mathcal{C}. \quad (17)$$

In each iteration, the variables $\mathbf{U}_i$ and $\prod_i$ store the corresponding mean value and error covariance matrix, respectively. Thus, at the end of iterations, (16) is workable. The convergence issues are discussed in [9] in detail.

Remark 1. In this paper, we approximate the posterior distribution with a Gaussian distribution. In the non-Gaussian system, the posterior distribution can be represented by the mixture of multiple Gaussian distributions. So, the DPFF can not be applied directly. A new particle flow filter called Gaussian mixture particle flow [31] can be employed to cater for the non-Gaussian situation. When the posterior distribution is represented by the Gaussian mixture model (GMM) $\sum_{j=1}^N (\mathcal{N} \cdot |m_k^j, \mathbf{P}_k^j|)$, the corresponding $\gamma(k)$ is calculated by the following:

$$\gamma(k) = \frac{1}{\sum_{j=1}^N \text{trace}(\mathbf{P}_k^j)}. \quad (18)$$

4. Experiments

In this section, we evaluate the performance of the proposed DPFF algorithm in the simulated environment and compare it with other approaches: the centralized particle filter (CPF) where we use the performance of the CPF as the base performance, the distributed particle filter based on average consensus [15] (DPF-AV), and the information weight average consensus-based distributed particle filter (DPF-WAV) [12].

4.1. Example 1: Grid Network. Consider such a simulation that a target moves in a 300 m $\times$ 500 m area with 15 sensors. The sensors have overlapped monitored regions. The monitored region of each sensor node is assumed to be a circle region of 75 m radius whose center is at the sensor's location. A sensor has a measurement of the target only if the target appears in the sensor's sensing area. Figure 3 illustrates the sensors and network connectivity. Connections between the sensors are shown as grey dashed lines. Each sensor can only communicate directly with its neighboring sensors whose distance to it is less than the communication range. The state vector of the target is represented as $\mathbf{x}_k = [\mathbf{x}_k, \mathbf{y}_k, \mathbf{v}_{x,k}, \mathbf{v}_{y,k}, \mathbf{w}_k]$,

FIGURE 3: Object trajectory, sensor node, and connectivity considered in the experiments. The connectivity among the nodes is shown using the grey dashed lines between nodes.

where $\mathbf{x}_k, \mathbf{y}_k$ and $\mathbf{v}_{x,k}, \mathbf{v}_{y,k}$ represent the target position and velocity, respectively. $\mathbf{w}_k$ denotes the turn rate of the target. In this case, a nearly coordinated turn model with the known constant turn rate $\mathbf{w}_k$ and the unknown velocity $\mathbf{v}_k$ is considered. This model is able to account for the motion of complicated maneuverable targets. The nonlinear scenario is used in [32], and the motion of target is modeled according to

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}(\mathbf{w}_k) \mathbf{x}_k + \mathbf{G} \mathbf{w}_k, \\ \mathbf{w}_{k+1} &= \mathbf{w}_k + \Delta \mathbf{u}_k, \end{aligned} \quad (19)$$

where

$$\begin{aligned} \mathbf{F}(\mathbf{w}) &= \begin{bmatrix} 1 & \frac{\sin \omega \Delta}{\omega} & 0 & -\frac{1 - \cos \omega \Delta}{\omega} \\ 0 & \cos \omega \Delta & 0 & -\sin \omega \Delta \\ 0 & \frac{1 - \cos \omega \Delta}{\omega} & 1 & \frac{\sin \omega \Delta}{\omega} \\ 0 & \sin \omega \Delta & 0 & \cos \omega \Delta \end{bmatrix}, \\ \mathbf{G}(\mathbf{w}) &= \begin{bmatrix} \frac{\Delta^2}{2} & 0 \\ \Delta & 0 \\ 0 & \frac{\Delta^2}{2} \\ 0 & \Delta \end{bmatrix}. \end{aligned} \quad (20)$$

$\mathbf{w}_k \sim \mathcal{N}(\cdot; 0, \sigma_w^2 \mathbf{I})$, $\mathbf{u}_k \sim \mathcal{N}(\cdot; 0, \sigma_u^2 \mathbf{I})$, $\Delta = 1$ s, $\sigma_w = 15$ m/s², and $\sigma_u = \pi/180$ rad/s. The observation is a noisy bearing and range vector given by

FIGURE 4: One example of estimated trajectory using DPFF (black curve), CPF (green curve), DPF-AV (cyan curve), and DPF-WAV (blue curve). The red curve indicates the ground truth.

$$\mathbf{z}_k = \begin{bmatrix} \arctan\left(\frac{p_{x,k}}{p_{y,k}}\right) \\ \sqrt{p_{x,k}^2 + p_{y,k}^2} \end{bmatrix} + \varepsilon_k, \quad (21)$$

where $\varepsilon_k \sim \mathcal{N}(\cdot; 0, R_k)$, with $R_k = \text{diag}([\sigma_\theta^2, \sigma_r^2]^T)$ and $\sigma_\theta = 2\pi/180$, $\sigma_r = 10$. p_x, p_y are the distance from the target to the sensor.

The number of particles for each node is to set 500 in the DPF-WAV and DPF-AV algorithms, while the DPFF algorithm only uses 50 particles in each node, while the CPF algorithm collects the entire available measurements from all sensor nodes and uses 500 particles to estimate.

The error metric that we have computed is the root mean square error (RMSE) between the true and estimated target positions at each time instant k from all sensors. Let (p_k^x, p_k^y) and $(\hat{p}_k^x, \hat{p}_k^y)$ denote the true and estimated target positions, respectively, at time k . The RMSE value at time k is calculated over a number of Monte Carlo (N_{MC}) runs according to

$$\text{RMSE} = \sqrt{\frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} (p_k^x - \hat{p}_k^x)^2 + (p_k^y - \hat{p}_k^y)^2}, \quad (22)$$

where $N_{MC} = 500$ is the number of Monte Carlo runs.

A sample of the estimated target track of DPFF is shown in Figure 4, which also shows the estimated track from the CPF, DPF-WAV, and DPF-AV algorithms. It is obvious that the DPF-AV lost the track while other algorithms work well when tracking the target.

Figure 5 shows the temporal evolution of RMSE. It is observed that the DPF-AV algorithm performs worst in these methods, while results of CPF, DPFF, and DPF-WAV are

FIGURE 5: Root mean square error (RMSE) versus time.

FIGURE 6: Root mean square error (RMSE) versus time.

fairly close to each other. The performance of DPFF and DPF-WAV is almost as good as the CPF and better than DPF-AV. Particularly, there is serious divergence with DPF-AV. For clear comparison of the proposed algorithm with DPF-WAV, we plot only DPF-WAV, CPF, and DPFF in Figure 6. Figure 6 illustrates the relationship of RMSE with timestamp of the DPF-WAV, CPF, and DPFF algorithms. As the DPF-WAV has a sharp fluctuation, it is obvious that the DPFF is better than the DPF-WAV and closer to the CPF. Especially, the DPF-WAV needs more particles than the DPFF and usually requires a significant amount of computing resources.

Moreover, we also computed the averaged RMSE (ARMSE) for all methods. The ARMSE is given by

TABLE 1: Average results over 500 runs with different methods.

Number	Methods	Average RMSE	Std RMSE	Average runtime (sec)
500	CPF	6.33	4.43	0.3919
	DPF-WAV	9.16	4.35	0.2851
	DPF-AV	126.62	27.51	0.2705
1000	CPF	5.98	4.03	0.7815
	DPF-WAV	8.97	4.12	0.3334
	DPF-AV	124.12	21.48	0.3102
50	DPFF	6.32	4.76	0.0672

FIGURE 7: Large-scale sparse WSN.

$$\text{ARMSE} = \sqrt{\frac{1}{TN_{MC}} \sum_{i=1}^{N_{MC}} \sum_{j=1}^T (p_k^x - \hat{p}_k^x)^2 + (p_k^y - \hat{p}_k^y)^2}. \quad (23)$$

Table 1 lists the ARMSE, standard deviation of RMSE values, and the average execution time per node over 500 simulation trials. The first column in Table 1 is the number of particles for different algorithms. It can be seen that the proposed DPFF only with 50 particles gets better estimate results than the DPF-WAV and DPF-AV, almost equal to the CPF with one-tenth the number of particles. For the average execution time per node, the DPFF is less than a quarter of the DPF-WAV and DPF-AV. In summary, the DPFF has higher accuracy and less execution time.

4.2. Example 2: Random Network. In the second scenario, we consider a large-scale sparse WSN as shown in Figure 7. That is a good benchmark since most of the WSN is spread over a large scale in the real world. Figure 7 shows the large scale sparse WSN in our simulation. There exists 100 sensor nodes and positions of sensors are randomly placed. The communication and sensing ranges of each sensor node are 20 m and 10 m, respectively. In Figure 7, the red line denotes true trajectory of the target. The state dynamic function and measurement function are similar to (19) and (21) whereas

FIGURE 8: One example of estimated trajectory using DPFF (black curve), CPF (green curve), DPF-AV (cyan curve), and DPF-WAV (blue curve). The red curve indicates the ground truth in the large-scale sparse WSN.

FIGURE 9: RMSE versus time.

the parameter of process noise $\sigma_w = 1 \text{ m/s}^2$ and the parameters of measurement noise $\sigma_\theta = \pi/180$, $\sigma_r = 2$. Then we compared the CPF, DPF-WAV, and DPF-AV with the DPFF as well. In this simulation, the DPFF maintains 50 particles per node for each time step while CPF, DPF-WAV, and DPF-AV use 1000 particles.

Figure 8 illustrates the estimated tracks of all algorithms. It can be seen that the estimated trajectory of DPF-AV (cyan curve) is far away from the target trajectory (red curve). Compared with the DPF-WAV (blue curve), the estimated result (black curve) of the DPFF is more close to the target trajectory. The RMSE values of four algorithms versus time

TABLE 2: Average results over 500 runs with different methods for random network.

Number	Methods	Average RMSE	Std RMSE	Average runtime (sec)
1000	CPF	0.19	0.13	5.7027
	DPF-WAV	1.29	1.73	0.9928
	DPF-AV	14.84	9.29	2.6869
50	DPFF	0.97	0.43	0.1295

are shown in Figure 9. A remarkable fact is that the DPF-AV clearly diverge after 60 time steps. We note from Figure 9 that the RMSE of DPF-WAV is fluctuating with time depending on the number of nodes which detected the target. In addition, to provide an overall indication of the comparative performance of the different methods, Table 2 gives the average and standard deviation RMSE and the average runtime over four methods. In Table 2, it can be seen that the average runtime per node of DPFF is 0.1295 seconds, which accounts for approximately one-tenth of the DPF-WAV (0.9928 s). The most time-consuming algorithm is the CPF, which runs a particle filter in one central node based on all the measurements. In the end, the proposed algorithm significantly outperforms the alternative algorithms in this experiment.

As a whole, the DPFF algorithm achieves a steady tracking with high accuracy with a few particles.

5. Conclusions

We presented a distributed particle flow filter algorithm for wireless sensor networks. At each sensor, a local particle flow filter computes a local state estimate that only depends on the local measurement. Then a perception confidence value is calculated from the particle flow filter. A max-consensus is used to make all the nodes agree on the best estimate of the target position. In the proposed distributed particle flow filter, each node just communicates with its neighboring sensor nodes and does not require any routing protocols. We applied the proposed distributed particle flow filter in two target tracking scenarios and demonstrated experimentally that its performance is better than the distributed averaged-based particle filter; moreover, it needs less computation time and samples. An extension of the distributed particle flow filter to multiple target tracking in the WSN remains a potential topic for the future.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC; Grant no. 61305013).

References

- [1] H. Wang, H. Lenz, A. Szabo, J. Bamberger, and U. D. Hanebeck, "Wlan-based pedestrian tracking using particle filters and low-cost MEMS sensors," in *2007 4th Workshop on Positioning, Navigation and Communication*, pp. 1–7, Hannover, Germany, 2007, IEEE.
- [2] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications - WSNA '02 ACM*, pp. 88–97, Atlanta, GA, USA, 2002, ACM.
- [3] S. Santini, B. Ostermaier, and A. Vitaletti, "First experiences using wireless sensor networks for noise pollution monitoring," in *Proceedings of the Workshop on Real-World Wireless Sensor Networks - REALWSN '08*, pp. 61–65, Glasgow, Scotland, 2008, ACM.
- [4] O. Hlinka, F. Hlawatsch, and P. M. Djuric, "Distributed particle filtering in agent networks: a survey, classification, and comparison," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 61–81, 2013.
- [5] F. Govaers and W. Koch, "An exact solution to track-to-track-fusion at arbitrary communication rates," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2718–2729, 2012.
- [6] R. Olfati-Saber, "Kalman-consensus filter : optimality, stability, and performance," in *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 7036–7042, Shanghai, China, 2009, IEEE.
- [7] A. T. Kamal, C. Ding, B. Song, J. A. Farrell, and A. Roy-Chowdhury, "A generalized Kalman consensus filter for wide-area video networks," in *IEEE Conference on Decision and Control and European Control Conference*, pp. 7863–7869, Orlando, FL, USA, 2011, IEEE.
- [8] W. Yang and H. Shi, "Sensor selection schemes for consensus based distributed estimation over energy constrained wireless sensor networks," *Neurocomputing*, vol. 87, pp. 132–137, 2012.
- [9] D. Di Paola, A. Petitti, and A. Rizzo, "Distributed Kalman filtering via node selection in heterogeneous sensor networks," *International Journal of Systems Science*, vol. 46, no. 14, pp. 2572–2583, 2015.
- [10] J. Read, K. Achutegui, and J. Míguez, "A distributed particle filter for nonlinear tracking in wireless sensor networks," *Signal Processing*, vol. 98, pp. 121–134, 2014.
- [11] O. Hlinka, O. Sluciak, F. Hlawatsch, P. M. Djuric, and M. Rupp, "Likelihood consensus and its application to distributed particle filtering," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4334–4349, 2012.
- [12] W. Tang, G. Zhang, J. Zeng, and Y. Yue, "Information weighted consensus-based distributed particle filter for large-scale sparse wireless sensor networks," *IET Communications*, vol. 8, no. 17, pp. 3113–3121, 2014.
- [13] D. Gu, J. Sun, Z. Hu, and H. Li, "Consensus based distributed particle filter in sensor networks," in *2008 International Conference on Information and Automation*, pp. 302–307, Changsha, China, 2008, IEEE.
- [14] A. Mohammadi and A. Asif, "Consensus-based distributed unscented particle filter," in *2011 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 237–240, Nice, France, 2011, IEEE.

- [15] T. Ghirmai, "Distributed particle filter for target tracking: with reduced sensor communications," *Sensors*, vol. 16, no. 9, p. 1454, 2016.
- [16] F. Daum and J. Huang, "Nonlinear filters with log-homotopy," in *Optical Engineering+ Applications*, p. 669918, San Diego, CA, USA, 2007, International Society for Optics and Photonics.
- [17] F. Daum and J. Huang, "Particle flow for nonlinear filters with log-homotopy," in *SPIE Defense and Security Symposium*, p. 696918, Orlando, FL, USA, 2008, International Society for Optics and Photonics.
- [18] F. Daum and J. Huang, "Nonlinear filters with particle flow induced by log-homotopy," in *SPIE Defense, Security, and Sensing*, p. 733603, Orlando, FL, USA, 2009, International Society for Optics and Photonics.
- [19] F. Daum and J. Huang, "Generalized particle flow for nonlinear filters," in *SPIE Defense, Security, and Sensing*, p. 769801, Orlando, FL, USA, 2010, International Society for Optics and Photonics.
- [20] T. Ding and M. Coates, "Implementation of the Daum-Huang exact-flow particle filter," in *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 257–260, Ann Arbor, MI, USA, 2012.
- [21] B. M. Nejad, S. A. Attia, and J. Raisch, "Max-consensus in a max-plus algebraic setting: the case of fixed communication topologies," in *2009 XXII International Symposium on Information, Communication and Automation Technologies*, pp. 1–7, Bosnia, Serbia, 2009, IEEE.
- [22] A. Petitti, D. Di Paola, A. Rizzo, and G. Ciciirelli, "Consensus-based distributed estimation for target tracking in heterogeneous sensor networks," in *IEEE Conference on Decision and Control and European Control Conference*, pp. 6648–6653, Orlando, FL, USA, 2011, IEEE.
- [23] Y. Li and M. Coates, "Particle filtering with invertible particle flow," 2016, <http://arxiv.org/abs/1607.08799>.
- [24] Y. Li, L. Zhao, and M. Coates, "Particle flow for particle filtering," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3979–3983, Shanghai, China, 2016, IEEE.
- [25] F. Daum, J. Huang, and A. Noushin, "Exact particle flow for nonlinear filters," in *SPIE Defense, Security, and Sensing*, p. 769704, Orlando, FL, USA, 2010, International Society for Optics and Photonics.
- [26] F. Daum and J. Huang, "Particle flow with non-zero diffusion for nonlinear filters," in *Proceedings of SPIE: Signal Processing, Sensor Fusion and Target Recognition XXII*, p. 87450P, Baltimore, MD, USA, 2013.
- [27] F. Daum, J. Huang, and A. Noushin, "Gromov's method for Bayesian stochastic particle flow: a simple exact formula for Q," in *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 540–545, Baden-Baden, Germany, 2016, IEEE.
- [28] F. Daum, A. Noushin, and J. Huang, "Numerical experiments for Gromov's stochastic particle flow filters," in *SPIE Defense+ Security*, p. 102000J, Anaheim, CA, USA, 2017, International Society for Optics and Photonics.
- [29] F. Daum, J. Huang, and A. Noushin, "Generalized Gromov method for stochastic particle flow filters," in *SPIE Defense+ Security*, p. 102000I, Anaheim, CA, USA, 2017, International Society for Optics and Photonics.
- [30] M. A. Khan and M. Ulmke, "Improvements in the implementation of log-homotopy based particle flow filters," in *2015 18th International Conference on Information Fusion (Fusion)*, pp. 74–81, Washington, DC, USA, July 2015.
- [31] M. A. Khan, M. Ulmke, and W. Koch, "A log homotopy based particle flow solution for mixture of Gaussian prior densities," in *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 546–551, Baden-Baden, Germany, Sept 2016.
- [32] B.-T. Vo, B.-N. Vo, and A. Cantoni, "The cardinality balanced multi-target multi-Bernoulli filter and its implementations," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 409–423, 2009.