Open access • Proceedings Article • DOI:10.1109/ICIF.2010.5712070

# Asynchronous distributed particle filter via decentralized evaluation of Gaussian products — Source link

Boris N. Oreshkin, Mark Coates

**Institutions:** McGill University

**Published on:** 26 Jul 2010 - International Conference on Information Fusion

**Topics:** Approximation algorithm, Particle filter, Robustness (computer science), Sensor fusion and Spanning tree

# Asynchronous distributed particle filter via decentralized evaluation of Gaussian products

**Boris N. Oreshkin and Mark J. Coates**
Department of Electrical and Computer Engineering
McGill university
Montreal, Quebec, Canada
boris.oreshkin@mail.mcgill.ca, mark.coates@mcgill.ca

**Abstract** − *We present a distributed particle filtering algorithm for target tracking in sensor networks. Several existing algorithms rely on the establishment and maintenance of a spanning path or tree. This is challenging in networks with dynamic topologies induced by mobile nodes and changing wireless conditions; the algorithms are vulnerable to link or node failure. More recent algorithms employ consensus algorithms to improve robustness but they adopt suboptimal fusion rules leading to a significant deterioration in performance. In our algorithm, nodes run local particle filters and then approximate their local posteriors using Gaussian approximations. A global posterior approximation is then computed using a novel gossiping approach that implements the optimal fusion rule. The resultant protocol is simple, robust and efficient. We present simulation results demonstrating a significant performance improvement over the best-performing existing algorithm.*

**Keywords:** Particle filter, Gaussian product, decentralized tracking.

## 1 Introduction

Sensor network deployment is becoming increasingly common and target tracking is an important application. When the target dynamics or observations cannot be adequately captured by a linear model, particle filtering becomes attractive. One option for implementing a particle filter in a sensor network is to perform computation at a single leader node (which may change over time) [1–3]. This approach has several drawbacks: there is a single point of failure; only one node can be queried; the leader node must have knowledge of all observation models and associated parameters (including calibration settings and possibly node locations).

An alternative approach is to implement distributed particle filters where likelihoods are calculated locally and the local information is then fused to form a global posterior. Several distributed particle filters require the establishment and maintenance of a spanning path or tree [4–8]. This can be very difficult when the mobility of nodes and changes in environmental conditions lead to frequent changes in the underlying network topology. The algorithms lack robustness, being vulnerable to node and link failures. More recently, several algorithms have been proposed that significantly improve the robustness, at the cost of an increased communication overhead, by using consensus algorithms to distribute the information [9–12]. The majority form local parametric approximations to posteriors generated by local particle filters [9, 10, 12] and then form a global parametric approximation by averaging the parameters. Unfortunately, this is a suboptimal fusion rule and leads to a significant deterioration in performance. The distributed particle filter in [11] is an exception; it exchanges particles rather than parameters, but this leads to a much higher communication overhead if the target state has relatively low dimension.

In this paper, we present a distributed particle filtering algorithm. Nodes run local particle filters and then form Gaussian approximations to their local posteriors. A global posterior approximation is then computed using a gossiping protocol. The local approximation procedure and the information exchange protocol are very simple, robust and efficient. Our algorithm is similar to that presented in [10], but we make the following important contributions: (i) we derive the optimal decentralized information fusion protocol (ii) we develop a novel robust gossiping approach to approximate the optimal fusion rule based on Gaussian product approximation. We present simulation results demonstrating a significant tracking performance improvement over the existing algorithm with similar communication and computation requirements.

### 1.1 Paper Organization

The paper is organized as follows. Section 2 presents a formal problem definition. In Sections 3 and 4 we derive the fusion protocol and describe its distributed implementation. Section 5 presents the results of numerical

experiments. Section 6 concludes the paper.

### 1.2 Related Work

There are several branches of the literature addressing the task of distributed particle filtering. The work that is most closely related to our proposed algorithm addresses the scenario when the measurements are distributed among a set of sensor nodes and multiple participating nodes perform the filtering task.

Rosencrantz et al. were among the first to address this problem in [13]. They described a query-based framework for localized information sharing. The approach can provide significant communication savings, but it is suboptimal, and there is no framework for the network-wide information fusion.

Coates proposed two algorithms for distributed particle filtering in [4]. In one algorithm, nodes maintain a common particle filter and use it to perform highly compressive quantization of their measurements. The quantized measurements are then distributed throughout the network. Ruan and WIllett proposed a similar algorithm for fusion of quantized measurements in [14], but focused on a centralized fusion architecture. Ing et al. extended the ideas in [6], improving the quantization and encoding procedure and added transmission vectorization to reduce the header transmission overhead. The requirement for synchronized particle filters introduces fragility and each node must be aware of all measurement modalities and models in the network.

The other algorithm presented in [4] assumes that the likelihoods at each node are conditionally independent given the state, so that the global likelihood can be expressed as a product of the local likelihoods. Nodes form a parametric model of the likelihood in the region of the state-space occupied by the particle set. They share these parameters along a spanning path or tree, which enables distributed computation of a parametric approximation of the global likelihood, permitting a consistent updating of particle weights.

Sheng et al. build upon this latter approach in [5]. Nodes run local particle filters and form Gaussian mixture model approximations of their local particle sets. These are shared via a fusion spanning path, allowing nodes to calculate an approximation to the global likelihood. The methods in [5] introduce resampling at each node to avoid the need for synchronized particle filters at each node. Gao et al. [7] and Song et al. [8] propose similar algorithms, but use Principal Component Analysis to update the mixture parameters more efficiently.

The algorithms in [4, 5, 7, 8] all require the establishment and maintenance of a stable spanning path (or spanning tree) over the topology of participating nodes. This can be very difficult in a challenging wireless environment with mobile nodes. The algorithms are not robust to failures of links or nodes.

There are a few distributed particle filtering algorithms that are more robust to adverse network conditions and failures [9–11]. In [9], Gu presents a distributed particle filter in which participating nodes run local particle filters and then apply the distributed expectation-maximization (EM) algorithm to calculate the parameters of a Gaussian mixture model (GMM) approximation to the global posterior. Particles and weights are then propagated locally by sampling from this global approximation.

In [10], Gu et al. use a different method to combine the local filter information. Each node forms a Gaussian approximation of its local particle set. A consensus algorithm is then applied to fuse these local approximations, constructing a Gaussian approximation to the global posterior. The fusion rule is suboptimal, with the global parameters being set to the averages of the local parameters. Liu et al. adopt a similar approach in [12] but use support vector machines to form the local approximations.

Lee and West propose an alternative robust particle filter that is based on the exchange of particles rather than Gaussian parameters [11]. The approach requires significantly more communication if the state has small dimension, but scales better as the state dimension increases.

Also related to our work, but less closely, are other distributed filtering schemes based on consensus algorithms. Rahman et al. and Olfati-Saber et al. have described distributed extended Kalman filters that employ consensus to share information [15, 16]. Although the EKF can be employed to approximate linearalizable filtering recursions, there is no guarantee of convergence or (asymptotic) consistency for such approximations.

Finally, we should note that the phrase "distributed particle filter" has also been used to describe the leader-node particle filters that appear in [1–3], where the particle filter calculations are performed at a single sensor node, but the node changes over time, often with the goal of following the target through the network. The phrase is also used to describe algorithms where the particle filtering computations are distributed among multiple nodes to improve execution speed [17–19]; the primary difference is that in such systems all nodes have access to all of the observations and there is no need to share or fuse likelihood information.

## 2 Problem Formulation

We assume that a collection of geographically distributed sensor nodes forms a set of vertices $\mathcal{V}_t = \{1, \ldots, n_t\}$ and the time-varying wireless links among these sensor nodes form a set of edges $\mathcal{E}_t$ of a sequence of graphs $G_t(\mathcal{V}_t, \mathcal{E}_t)$, $t = 0, 1, 2, \ldots$ The set of edges $\mathcal{E}_t$ is a set of unordered pairs $u, v \in \mathcal{V}$ such that at time $t$ $(u, v) \in \mathcal{E}_t$ if and only if a bidirectional wireless link between $u$ and $v$ can be established with high probability. We assume that the global connectivity structure $\mathcal{E}_t$ and the complete vertex set $\mathcal{V}_t$ is unknown to nodes

in the network, but nodes have the knowledge of their local neighbours, with whom they can establish a direct wireless link.

Every sensor node $u_t$ measures certain physical phenomenon using its measurement modality and obtains a noisy measurement $y_t^{u_t}$ at time step $t$. The information acquired by this sensor is described by its likelihood function $p(y_t^{u_t}|x_t)$. We assume that sensor nodes are unaware of the possibly heterogeneous measurement modalities and measurement noise models employed by other nodes in the network. The sensor noises are assumed conditionally independent given the state, implying that the joint likelihood can thus be factorized as follows:

$$p(y_t^{\mathcal{V}_t}|x_t) = \prod_{u_t \in \mathcal{V}_t} p(y_t^{u_t}|x_t). \quad (1)$$

Here $y_t^{\mathcal{V}_t} = \{y_t^{u_t} : u_t \in \mathcal{V}_t\}$ is the complete measurement snapshot (measurement set) acquired by the network at time $t$. We define the time history of such measurements as $y_{1:t}^{\mathcal{V}_{1:t}} = \{y_j^{\mathcal{V}_j} : 1 \leq j \leq t\}$.

We also assume that although there may be a lack of communication synchronization, measurement timing synchronism is established (via a decentralized protocol), so that local clock drift is compensated for and the expression $x_t$ has the same meaning for each sensor node. Our last assumption is that there is no centralized entity responsible for sensor coordination and that any sensor can be queried by a sink and should be able to report global tracking information at any point in time.

Our goal is to design a distributed tracking protocol that exploits the global measurement snapshot at every time step under the assumptions outlined above. It follows under these assumptions that this protocol should be decentralized, with an asynchronous communication model. The protocol can rely on the global likelihood being factorizable, but it cannot rely on transmitting raw or aggregated measurements, since measurement modalities and noise models are not known elsewhere in the sensor network.

# 3 Distributed Fusion Protocol

Under the assumptions outlined in the previous section the optimal (centralized) Bayes fusion protocol can be outlined, at every iteration, as follows:

$$p(x_t|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) = \int p(x_t|x_{t-1}) p(x_{t-1}|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) \mathrm{d}x_{t-1} \quad (2)$$

$$p(x_t|y_{1:t}^{\mathcal{V}_{1:t}}) = \frac{p(y_t^{\mathcal{V}_t}|x_t) p(x_t|y_{1:t-1}^{\mathcal{V}_{1:t-1}})}{\int p(y_t^{\mathcal{V}_t}|x_t) p(x_t|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) \mathrm{d}x_t}. \quad (3)$$

This recursion is optimal in terms of the mean squared error performance. In a distributed setting, however, nodes do not have access to the global likelihood, so they cannot locally compute (or estimate) the recursive update (3).

## 3.1 Optimal Distributed Protocol

If the numerator of (3) is calculated using a distributed algorithm, then each node can perform the integration to evaluate the denominator (at least approximately).

We are thus interested in the distributed calculation of the joint distribution

$$p(x_t, y_t^{\mathcal{V}_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) = p(x_t|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) \prod_{u_t \in \mathcal{V}_t} p(y_t^{u_t}|x_t). \quad (4)$$

If node $u_t$ runs a *local* Bayes recursion at time $t$, then the information available at this node is

$$p(x_t, y_t^{u_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) = p(x_t|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) p(y_t^{u_t}|x_t). \quad (5)$$

Let us explore distributed protocols already available for scalar distributed computations. Bauso et al. described a number of such protocols [20] that permit distributed computation of various functions of the local scalar values. One such function is the geometric mean,

$$z = \sqrt[n]{\prod_{u \in \mathcal{V}} z_u}, \quad (6)$$

where $z_u$ is the local piece of data and $\mathcal{V}$ is a set of cardinality $n$. In our case $p(x_t, y_t^{u_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}})$ is such a local piece of data. However, we cannot represent $p(x_t, y_t^{\mathcal{V}_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}})$ directly via $p(x_t, y_t^{u_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}})$ using (6). On the other hand, if we compute an alternative local value:

$$\widetilde{p}(x_t, y_t^{u_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) = p(x_t|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) p^{n_t}(y_t^{u_t}|x_t), \quad (7)$$

we can construct the following expression for (4):

$$p(x_t, y_t^{\mathcal{V}_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) = \left( \prod_{u_t \in \mathcal{V}_t} \widetilde{p}(x_t, y_t^{u_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) \right)^{1/n_t}. \quad (8)$$

This representation suggests a procedure for calculating the required joint distribution: compute locally at each node the quantities $\widetilde{p}(x_t, y_t^{u_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}})$ and then use a distributed protocol from [20] to compute the geometric mean.

## 3.2 Approximate Distributed Protocol

The exact optimal Bayes computations are only viable for a limited class of dynamic and measurement models. In a general situation, (7) will be approximated in one form or another, and this approximation may require a large number of terms (e.g. in the case of particle filter hundreds and thousands of particles). Distributed evaluation of the geometric mean of such approximations will probably involve the exchange of many values, which is undesirable in a sensor network where communication energy should be preserved.

We propose to use particle approximation to perform the local predict-update recursion calculations,

but then form a local Gaussian approximation of $\widetilde{p}(x_t, y_t^{u_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}})$. This greatly reduces the required data exchange during information fusion. The local value at node $u_t$ is approximated by $\mathcal{N}(x_t, \boldsymbol{\mu}_t^{u_t}, \mathbf{R}_t^{u_t})$, where $\boldsymbol{\mu}_t^{u_t}$ is the mean and $\mathbf{R}_t^{u_t}$ is the covariance matrix calculated to optimize a certain cost function (e.g. log-likelihood) given the particle approximation of (7).

We can thus rewrite the global Gaussian approximation, $\widehat{p}(x_t, y_t^{\mathcal{V}_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}})$, of the joint distribution as:

$$\widehat{p}(x_t, y_t^{\mathcal{V}_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) = \left( \prod_{u_t \in \mathcal{V}_t} \mathcal{N}(x_t, \boldsymbol{\mu}_t^{u_t}, \mathbf{R}_t^{u_t}) \right)^{1/n_t}. \quad (9)$$

Since the product of Gaussians is itself a Gaussian, it can be shown [21] that

$$\begin{aligned} \widehat{p}(x_t, y_t^{\mathcal{V}_t}|y_{1:t-1}^{\mathcal{V}_{1:t-1}}) &\propto \mathcal{N}(x_t, \boldsymbol{\mu}_t^{\Sigma}, \mathbf{R}_t^{\Sigma})^{1/n_t} \\ &\propto \mathcal{N}(x_t, \boldsymbol{\mu}_t^{\Sigma}, n_t \mathbf{R}_t^{\Sigma}), \end{aligned} \quad (10)$$

where the summary statistics $\boldsymbol{\mu}_t^{\Sigma}$ and $\mathbf{R}_t^{\Sigma}$ are given by:

$$\boldsymbol{\mu}_t^{\Sigma} = \mathbf{R}_t^{\Sigma} \sum_{u_t \in \mathcal{V}_t} (\mathbf{R}_t^{u_t})^{-1} \boldsymbol{\mu}_t^{u_t}, \quad (11)$$

$$\mathbf{R}_t^{\Sigma} = \left( \sum_{u_t \in \mathcal{V}_t} (\mathbf{R}_t^{u_t})^{-1} \right)^{-1}. \quad (12)$$

The information fusion step performed at every iteration of the proposed tracking algorithm thus involves calculating a Gaussian product.

Based on the above expressions and on the ideas underlying randomized gossip (RG) [22], the asynchronous version of the distributed average consensus algorithm, we propose a distributed asynchronous iteration for the asymptotic computation of the summary statistics (11) and (12). In particular, by defining $\mathbf{Q}_t^{u_t} = (\mathbf{R}_t^{u_t})^{-1}$ and $\boldsymbol{\nu}_t^{u_t} = (\mathbf{R}_t^{u_t})^{-1} \boldsymbol{\mu}_t^{u_t}$ (quantities that can be calculated locally) we can see the following:

$$n_t \mathbf{R}_t^{\Sigma} = \left( \frac{1}{n_t} \sum_{u_t \in \mathcal{V}_t} \mathbf{Q}_t^{u_t} \right)^{-1}, \quad (13)$$

$$\boldsymbol{\mu}_t^{\Sigma} = (n_t \mathbf{R}_t^{\Sigma}) \left( \frac{1}{n_t} \sum_{u_t \in \mathcal{V}_t} \boldsymbol{\nu}_t^{u_t} \right). \quad (14)$$

Thus the computation of quantities defining the Gaussian product in (10) can be accomplished in a distributed fashion using two vectorized randomized gossip iterations. The first iteration is over local inverse covariance matrices, $\mathbf{Q}_t^{u_t}$; the second iteration is over the local transformed mean vectors $\boldsymbol{\nu}_t^{u_t}$.

At every step of randomized gossip, one node wakes up (using, for example, a Poisson clock [23]) and selects a neighbour from its neigbourhood at random using a probabilistic law; these nodes exchange information and update their values. The gossip update at iteration $k$ for nodes $u_t(k), v_t(k)$ selected with the above specified rule consists of exchanging matrices $\mathbf{Q}_t^{u_t(k)}(k), \mathbf{Q}_t^{v_t(k)}(k)$ (in the inverse covariance iteration) or vectors $\boldsymbol{\nu}_t^{u_t(k)}(k), \boldsymbol{\nu}_t^{v_t(k)}(k)$ (in the transformed mean iteration) and updating their respective values to the corresponding means:

$$\begin{aligned} \mathbf{Q}_t^{u_t(k)}(k+1) &= \mathbf{Q}_t^{v_t(k)}(k+1) \\ &= (\mathbf{Q}_t^{u_t(k)}(k) + \mathbf{Q}_t^{v_t(k)}(k))/2, \end{aligned} \quad (15)$$

$$\begin{aligned} \boldsymbol{\nu}_t^{u_t(k)}(k+1) &= \boldsymbol{\nu}_t^{v_t(k)}(k+1) \\ &= (\boldsymbol{\nu}_t^{u_t(k)}(k) + \boldsymbol{\nu}_t^{v_t(k)}(k))/2. \end{aligned} \quad (16)$$

### 3.3 Network Size Estimation

Note that calculating (7) requires the knowledge of the network size, $n_t$. In this section we describe the distributed algorithm for the network size estimation based on the gossiping idea [24] (see [25] for a review of alternative techniques). We use the following algorithm based on the fact that the gossip average is equal to $1/n_t$ if only one of the nodes in the network has value one and all others are initially set to 0 [24].

Initially, every node flips a coin and assumes value 1 with probability $\rho$. Nodes initialized with value 1 create unique tokens, the rest of the nodes are initialized with zeros. Each node that assumed value 1 starts a random walk marked by a unique token. The next node in the random walk is selected uniformly at random from the neighbourhood of the current node. If $i$ and $j$ are the current and the next nodes in the random walk, and $x_i(k)$ and $x_j(k)$ are their respective values before the update, then the new values become:

$$x_i(k+1) = x_j(k+1) = \frac{x_i(k) + x_j(k)}{2}$$

Node $i$ also transmits an exponential moving average of the squared difference between the exchanged values, defined by the following recursion wih $e(0) = 1$:

$$e(k+1) = (1-\alpha)e(k) + \alpha(x_i(k) - x_j(k))^2. \quad (17)$$

Each random walk terminates when $e(k+1) \leq \gamma$, where the threshold $\gamma$ is chosen according to the required accuracy of the network size estimate.

Each node formulates a final estimate of the network size by calculating the reciprocal of the average of the current values associated with all random walks that have traversed it.

## 4 Proposed Distributed Particle Filter

In this section we discuss a concrete particle filter implementation using the fusion framework discussed in Section 3. The high level algorithm describing one iteration of the proposed distributed particle filter is shown below in Algorithm 1.

**Algorithm 1**: Distributed particle filter

At time $t$ make measurements, $y_t^{\mathcal{V}_t}$ ;
$\widehat{n}_t = \text{EstimateSize}(\alpha, \gamma, G_t(\mathcal{V}_t, \mathcal{E}_t))$ ;
**for** $u_t \in \mathcal{V}_t$ *and* $i = 1 \dots N$ **do**
  $\widetilde{\xi}_t^{u_t,(i)} \sim q_t(x_t | x_{t-1}, y_t^{u_t}, y_{1:t-1}^{\mathcal{V}_{1:t-1}})$ ;
  $\widetilde{\omega}_t^{u_t,(i)} = \frac{p_{Y_t|X_t}^{\widehat{n}_t}(y_t^{u_t}|\widetilde{\xi}_t^{u_t,(i)}) p_{X_t|X_{t-1}}(\widetilde{\xi}_t^{u_t,(i)}|\xi_{t-1}^{u_t,(i)})}{q_t(\widetilde{\xi}_t^{u_t,(i)}|\xi_{t-1}^{u_t,(i)}, y_t^{u_t}, y_{1:t-1}^{\mathcal{V}_{1:t-1}})}$ ;
  $\boldsymbol{\mu}_t^{u_t}, \mathbf{R}_t^{u_t} = \text{SaveGaussian}(\{\widetilde{\xi}_t^{u_t,(i)}, \widetilde{\omega}_t^{u_t,(i)}\}_{i=1}^N)$ ;
**endfor**
$\boldsymbol{\mu}_t^{\Sigma}, \mathbf{R}_t^{\Sigma} = \text{DoFusion}(\{\boldsymbol{\mu}_t^{u_t}, \mathbf{R}_t^{u_t}\}_{u_t \in \mathcal{V}_t})$ ;
**for** $u_t \in \mathcal{V}_t$ **do**
  $\{\xi_t^{u_t}\}_{i=1}^N = \text{Resample}(\mathcal{N}(x_t, \boldsymbol{\mu}_t^{\Sigma}, \widehat{n}_t \mathbf{R}_t^{\Sigma}))$ ;
**endfor**

We now provide a detailed description of the proposed algorithm. When a new set of measurements becomes available, the algorithm estimates the size of the network that acquired the measurements using the procedure described in Section 3.3. The local particle filter is then invoked to calculate the local Gaussian approximation $\mathcal{N}(x_t, \boldsymbol{\mu}_t^{u_t}, \mathbf{R}_t^{u_t})$. The filter samples the new set of particles, $\widetilde{\xi}_t^{u_t}$, from the proposal distribution $q_t(x_t|x_{t-1}, y_t^{u_t}, y_{1:t-1}^{\mathcal{V}_{1:t-1}})$. The likelihood $p_{Y_t|X_t}^{\widehat{n}_t}(y_t^{u_t}|x_t)$ can be peaked and it is important to concentrate particles in the regions of high likelihood. There are approaches for doing this, e.g. the auxiliary particle filter by Pitt and Shepard [26]; we employ an auxiliary particle filter in our simulations. The last local step is the construction of the Gaussian approximation $\mathcal{N}(x_t, \boldsymbol{\mu}_t^{u_t}, \mathbf{R}_t^{u_t})$ based on the sample $\widetilde{\xi}_t^{u_t}$, which we accomplish using standard maximum likelihood formulae and normalized weights $\overline{\omega}_t^{u_t,(i)} = \widetilde{\omega}_t^{u_t,(i)} / \sum_{j=1}^N \widetilde{\omega}_t^{u_t,(j)}$:

$$\boldsymbol{\mu}_t^{u_t} = \sum_{i=1}^N \overline{\omega}_t^{u_t,(i)} \widetilde{\xi}_t^{u_t,(i)} \tag{18}$$

$$\mathbf{R}_t^{u_t} = \sum_{i=1}^N \overline{\omega}_t^{u_t,(i)} (\widetilde{\xi}_t^{u_t,(i)} - \boldsymbol{\mu}_t^{u_t})(\widetilde{\xi}_t^{u_t,(i)} - \boldsymbol{\mu}_t^{u_t})^T. \tag{19}$$

The estimated Gaussian approximations at every node are used in the global information fusion step described in detail in Section 3.2. When the global measurement snapshot is diffused over the network via the gossip protocol, we finally obtain the globally updated particle set $\xi_t^{u_t}$ at every node $u_t$ by sampling $N$ particles from the global Gaussian approximation $\mathcal{N}(x_t, \boldsymbol{\mu}_t^{\Sigma}, \widehat{n}_t \mathbf{R}_t^{\Sigma})$.

It is interesting to note how the structure of the proposed algorithm differs from the fusion model proposed by Gu et al. [10]. Gu et al. [10] estimate the global summary statistics as $\boldsymbol{\mu}_t^{\Sigma} = 1/|\mathcal{V}_t| \sum_{u_t \in \mathcal{V}_t} \boldsymbol{\mu}_t^{u_t}$ and $\mathbf{R}_t^{\Sigma} = 1/|\mathcal{V}_t| \sum_{u_t \in \mathcal{V}_t} \mathbf{R}_t^{u_t}$. Instead of using an exponentiated likelihood $p^{n_t}(y_t^{u_t}|x_t)$ they use the original likelihood $p(y_t^{u_t}|x_t)$. It is difficult to justify this suboptimal fusion method, although it can perform reasonably if the local distributions are all very similar. Our algorithm approximates the optimal fusion procedure and can address the more general case when the likelihood functions $p(y_t^{u_t}|x_t)$ differ from sensor to sensor in a heterogeneous network.

# 5 Numerical Experiments

In our simulations we consider the two-dimensional scenario where $K$ sensors are distributed uniformly in a unitary square region. Only sensors within connectivity radius $\sqrt{2 \log K / K}$ of each other can communicate directly (random geometric graph model). A single target that makes a clockwise coordinated turn of radius 0.2 with a constant turn rate $\omega_t = 0.139$. It starts in y-direction with initial position $[0.25, 0.25]$ and is tracked for 50 seconds.

In our filters, the target motion is modeled by the nearly coordinated turn model [27] with known constant turn rate and unknown cartesian velocity. Therefore, the state of the target is given as $x_t = [p_t^x, p_t^y, v_t^x, v_t^y, \omega_t]^T$, where $p, v$ and $\omega$ denote the position, velocity and turn rate respectively. The dynamic model for the coordinated turn model $f_t(.)$ is

$$x_{t+1} = \begin{pmatrix} 1 & 0 & \frac{\sin(\omega_t)}{\omega_t} & \frac{\cos(\omega_t)-1}{\omega_t} & 0 \\ 0 & 1 & \frac{1-\cos(\omega_t)}{\omega_t} & \frac{\sin(\omega_t)}{\omega_t} & 0 \\ 0 & 0 & \cos(\omega_t) & -\sin(\omega_t) & 0 \\ 0 & 0 & \sin(\omega_t) & \cos(\omega_t) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} x_t + \varsigma_{t+1}$$

where $\varsigma_{t+1}$ is Gaussian process noise, $\varsigma_{t+1} \sim \mathcal{N}(0, P_\varsigma)$, $P_\varsigma = \text{diag}([0.05, 0.05, 0.04, 0.04, 0])$, sampling period is 1 second. We assume that initially, the benchmarked filters obtain an initialization $x_0$ with mean corresponding to the true position of the target and a covariance $P_0 = \text{diag}([0.01, 0.01, 0.0001, 0.0001, 0])$.

The $K$ sensors uniformly distributed in the unit square use 4 different measurement modalities: bearing, received signal strength, range and radial velocity measurements (that can be obtained e.g. from Doppler velocity measurements). The results of our simulations are averaged over 1000 Monte-Carlo trials. For every trial we randomly generated a fixed senor network configuration. In this configuration every sensor $i$ was assigned coordinates $s_i = [s_i^x, s_i^y]$ and one of the measurement modalities with equal probability for the duration of one trial. The bearing-only modality is described as

$$h_t(x_t) = \arctan[(p_t^y - s_i^y)/(p_t^x - s_i^x)]. \tag{20}$$

The received signal strength measurement is

$$h_t(x_t) = 1/[(p_t^y - s_i^y)^2 + (p_t^x - s_i^x)^2 + a], \tag{21}$$

where $a$ is set to 0.0001. The range measurement is

$$h_t(x_t) = \|p_t - s_i\|, \tag{22}$$

where $p_t - s_i$ is the target displacement vector, $v_t$ is the cartesian velocity vector, $\|\cdot\|$ is the Euclidean distance. Finally, the radial velocity measurement equation is

$$h_t(x_t) = (p_t - s_i) \cdot v_t / \|p_t - s_i\|, \tag{23}$$

where $\cdot$ is the dot product of two vectors.

The measurements from the sensors are corrupted by Gaussian noise with standard deviations respectively for bearing-only, RSS, range, and radial velocity modalities 0.175, 2, 0.14, 0.004 in the simulation scenario.

The network size estimation routine was used with the settings $\gamma = 10^{-6}$ and $\alpha = 0.95$. The probability $\rho_t$ of a node setting its value to 1 initially at time $t$ was set to $\rho_t = 1/\widehat{n}_t$ and we assumed that initially, at time $t = 0$, $\widehat{n}_0 = 5$ (assumed lower bound for the network size). The additional communication overhead that arised from the network size estimation routine is relatively small (10-20 percent of the total communication required for the fusion operation).

For example, in our experiment we have the dimensionality of exchanged data $D = (d-1) + d(d-1)/2 = 4 + 5 \times 4/2 = 10$ (mean plus covariance), and we set the number of consensus iterations during the Gaussian product calculation to be equal to $5\widehat{n}_t^2$. For the network sizes $K = 10$ and $K = 50$ we thus have the total number of values exchanged $5D\widehat{n}_t^2 \approx 5 \times 10 \times 10^2 = 5 \times 10^3$ (with our settings $\widehat{n}_t \approx n_t$) and $5D\widehat{n}_t^2 \approx 5 \times 10 \times 50^2 \approx 1.3 \times 10^5$. The average number of values exchanged during network size estimation for $K = 10$ and $K = 50$ was $\approx 10^3$ and $\approx 1.3 \times 10^4$.

The network size estimation protocol can be embedded into the Gaussian product calculation protocol. If the number of participating sensor nodes changes relatively slowly over time, there the size estimation protocol does not have to be run every time step, further reducing the communication overhead.

### 5.1 Benchmarked Filters

In the simulations, three different particle filters have been implemented:

*PFprop*: proposed particle filter discussed in Sections 3 and 4.

*PFcent*: a centralized particle filter which collects all the measurements from all the sensors and uses the posterior distribution of the PFprop filter as a proposal distribution.

*PFGu*: a distributed particle filter proposed by Gu et al. in [10].

The second filter provides us with the practical performance limit achievable by a particle filter and the third filter is our proposed filter.

The centralized filter *PFcent* is executed with a fixed number of particles equal to 10000 throughout our simulations. The two other filters are tested with $N = 500$ and $N = 1000$ particles for network sizes $K = 10, 20, 30, 40, 50$. We use the RMS position error to compare the performances of particle filters. Let $(p_t^x, p_t^y)$ and $(\hat{p}_{k,i}^x, \hat{p}_{k,i}^y)$ denote the true and estimated target positions at time step $t$ at the $i$th Monte-Carlo trial. Suppose $M$ is the number of such Monte-Carlo trials. The RMS position error at $t$ is calculated as

$$\mathrm{RMS}_t = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\hat{p}_{t,i}^x - p_t^x)^2 + (\hat{p}_{t,i}^y - p_t^y)^2} \tag{24}$$

### 5.2 Results and Discussion

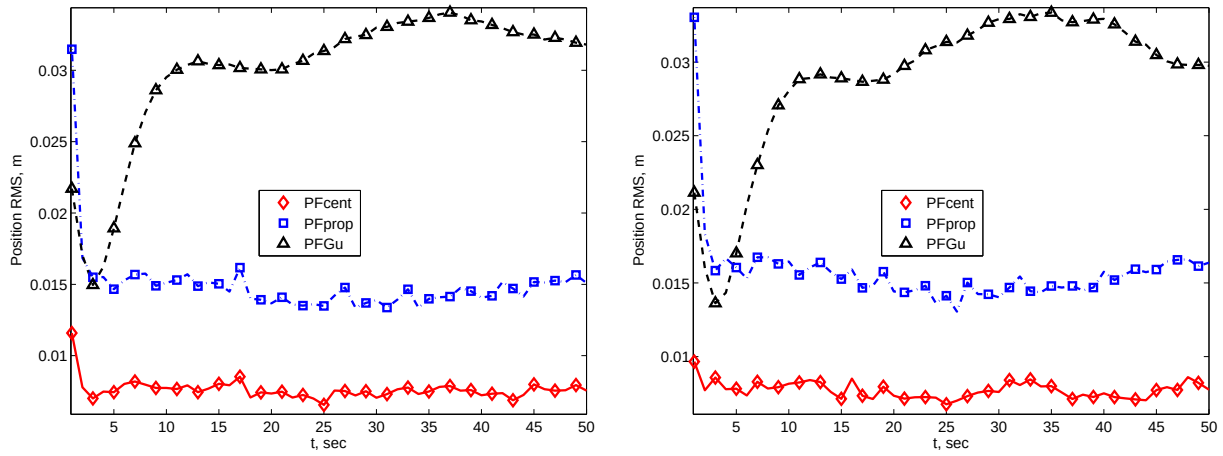
In our first experiment we fix the network size at $K = 50$ and measure the RMS tracking performance of the algorithms described in the previous section for two particle cloud sizes, $N = 500$ and $N = 1000$. The performance of the investigated algorithms is shown in Fig. 1. It is clear that the proposed algorithm, *PFprop*, has significantly better performance compared to the algorithm with the suboptimal fusion rule, *PFGu*, after the first few steps of the transient behaviour.

In our second experiment we vary the network size and measure the RMS performance at time $t = 25$ for the three filters discussed in Section 5.1. We report the results of this experiments in Fig. 2. We can see that the proposed filter that was derived from the optimal distributed tracking protocol outperforms the filter proposed by Gu et al. [10]. We can see that the proposed filter is around 2 times more accurate than the latter filter in terms of measured RMS performance. We can see that there is still space for improvement, since our algorithm is still 1.5–2 times worse than the centralized filter.

## 6 Concluding Remarks

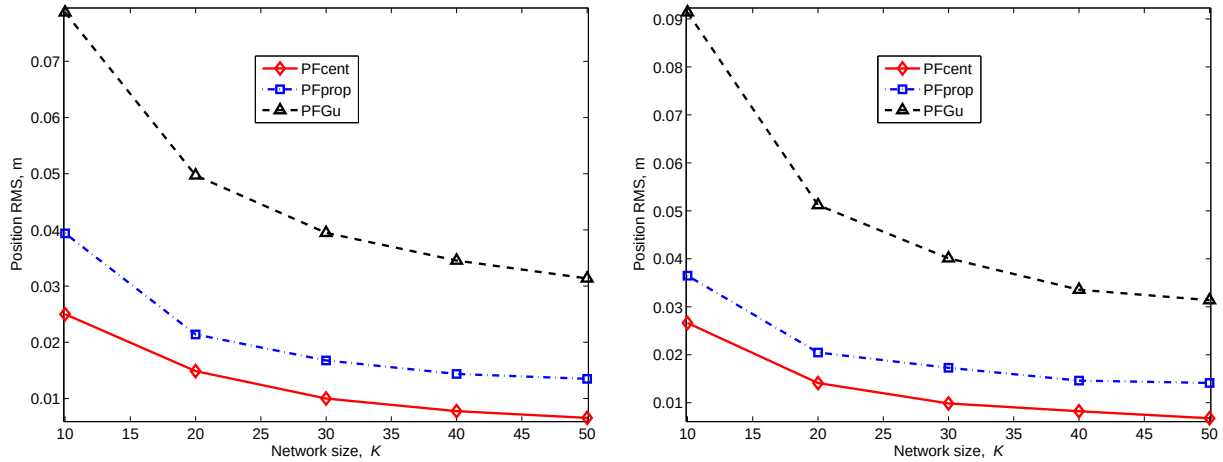
We have proposed a distributed asynchronous particle filter based on local Gaussian approximation of the posteriors generated by local particle filters and fusion via distributed calculation of a Gaussian product. We first derived an optimal distributed tracking and fusion protocol, and then developed a practical algorithm by approximating this protocol. The communication overhead of the proposed algorithm is significantly greater than some previously proposed particle filters, but there is no need for establishment and maintenance of a spanning tree or path for fusion. This makes the algorithm much more robust and more applicable for networks with dynamic topologes deployed in regions with adverse wireless conditions. In contrast to other robust distributed particle filters that are based on fusing parametric approximations, our algorithm approximates an optimal fusion rule, and our simulations demonstrate that this can lead to a significant performance improvement. The RMS tracking error achieved by our algorithm is approximately half that of the algorithm with similar communication and computation requirements.

Our simulations also indicate that there is room for improvement, since our algorithm does not achieve the optimal performance. In future we are planning to work




(a) N=500 (b) N=1000

Figure 1: RMS tracking performance for the centralized filter, proposed filter and the filter due to Gu et al. [10].




(a) N=500 (b) N=1000

Figure 2: RMS tracking performance at time $t = 25$ for varying network sizes and for the centralized filter, proposed filter and the filter due to Gu et al. [10].

on improving the performance of the proposed algorithm by using a more sophisticated local approximation scheme and developing the appropriate associated distributed fusion protocol. We will also explore the use of more advanced gossiping protocols that can reduce the communication overhead but achieve similar accuracy.

# References


[1] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative signal and information processing: an information-directed approach," *Proc. IEEE*, vol. 91, no. 8, pp. 1199–1209, Aug. 2003.

[2] A. Ihler, J. Fisher, and A. Willsky, "Particle filtering under communication constraints," in *Proc. IEEE Workshop Stat. Signal Process.*, Bordeaux, France, May 2005.

[3] Y. Huang, W. Liang, H. bin Yu, and Y. Xiao, "Target tracking based on a distributed particle filter in underwater sensor networks," *J. Wireless Communications & Mobile Computing*, vol. 8, no. 8, pp. 1023–1033, 2008.

[4] M. Coates, "Distributed particle filters for sensor networks," in *Proc. Int. Symposium Info. Process. Sens. Netw. (IPSN)*, Berkeley, CA, USA, Apr. 2004.

[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 *Proc. Int. Symp. Info. Process. Sens. Netw. (IPSN)*, Los Angeles, CA, USA, Apr. 2005.

[6] G. Ing and M. J. Coates, "Parallel particle filters for tracking in wireless sensor networks," in *Proc. IEEE Workshop Signal Process. Adv. Wireless Comm.*, New York, NY, USA, Jun. 2005.

[7] W. Gao, H. Zhao, C. Song, and J. Xu, "A new distributed particle filtering for WSN target tracking," in *Proc. Int. Conf. Signal Process. Syst.*, Singapore, May 2009.

[8] C. Song, H. Zhao, and W. Jing, "Asynchronous distributed PF algorithm for WSN target tracking," in *Proc. Int. Conf. Wireless Comm. Mobile Comp. (IWCMC)*, Leipzig, Germany, Jun. 2009.

[9] D. Gu, "Distributed particle filter for target tracking," in *Proc. IEEE Int. Conf. Robotics and Automation*, Roma, Italy, Apr. 2007.

[10] D. Gu, J. Sun, Z. Hu, and H. Li, "Consensus based distributed particle filter in sensor networks," in *Proc. Int. Conf. Info. and Automation (ICIA)*, Zhangjiajie, China, Jun. 2008.

[11] S. Lee and M. West, "Markov Chain Distributed Particle Filters (MCDPF)," in *Proc. IEEE Conf. Decision and Control*, Shanghai, China, Dec. 2009.

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

[13] M. Rosencrantz, G. Gordon, and S. Thrun, "Decentralized sensor fusion with distributed particle filters," in *Proc. Conf. Uncertainty in AI (UAI)*, Acapulco, Mexico, Aug. 2003.

[14] Y. Ruan, P. Willett, and A. Marrs, "Fusion of quantized measurements via particle filtering," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, USA, Mar. 2003.

[15] R. Rahman, M. Alanyali, and V. Saligrama, "Distributed tracking in multihop sensor networks with communication delays," *IEEE Trans. Signal Process.*, vol. 55, no. 9, pp. 4656–4668, 2007.

[16] R. Olfati-Saber and N. F. Sandell, "Distributed tracking in sensor networks with limited sensing range," in *Proc. Am. Control Conf.*, Seattle, Washington, USA, Jun. 2008.

[17] A. S. Bashi, V. P. Jilkov, X. R. Li, and H. Chen, "Distributed implementations of particle filters," in *Proc. Int. Conf. Info. Fusion*, Cairns, Queensland, Australia, Jul. 2003.

[18] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2442–2450, 2005.

[19] W. Du, J.-B. Hayet, J. Verly, and J. Piater, "Ground-target tracking in multiple cameras using collaborative particle filters and principal axis-based integration," *IPSJ Trans. Comp. Vision Appl.*, vol. 1, pp. 58–71, 2009.

[20] D. Bauso, L. Giarré, and R. Pesenti, "Non-linear protocols for optimal distributed consensus in networks of dynamic agents," *Syst. Control Letters*, vol. 55, no. 11, pp. 918–928, 2006.

[21] M. Gales and S. Airey, "Product of Gaussians for speech recognition," *Computer Speech & Language*, vol. 20, no. 1, pp. 22–40, 2006.

[22] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 2508–2530, 2006.

[23] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.

[24] M. Jelasity and A. Montresor, "Epidemic-style proactive aggregation in large overlay networks," in *Proc. Int. Conf. Distributed Computing Syst.*, Washington, DC, USA, Mar. 2004.

[25] E. L. Merrer, A.-M. Kermarrec, and L. Massoulie, "Peer to peer size estimation in large and dynamic networks: a comparative study," in *Proc. Int. Symp. High-Performance Distributed Computing*, Los Alamitos, CA, USA, Jun. 2006.

[26] M. Pitt and N. Shepard, "Filtering via simulation: auxiliary particle filters," *J. Am. Statist. Assoc.*, vol. 94, no. 446, pp. 590–599, Jun. 1999.

[27] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, 1st ed. Wiley-Interscience, June 2001.