in Proc. IEEE ICASSP 2011
Prague, Czech Republic, May 2011, pp. 3756-3759
Copyright 2011 IEEE

DISTRIBUTED GAUSSIAN PARTICLE FILTERING USING LIKELIHOOD CONSENSUS

Ondrej Hlinka[1], *Ondrej Slučiak*[1], *Franz Hlawatsch*[1], *Petar M. Djurić*[2], *and Markus Rupp*[1]

[1]Institute of Telecommunications, Vienna University of Technology, Austria (ohlinka@nt.tuwien.ac.at)
[2]Department of Electrical and Computer Engineering, Stony Brook University, NY, USA (djuric@ece.sunysb.edu)

ABSTRACT

We propose a distributed implementation of the *Gaussian particle filter* (GPF) for use in a wireless sensor network. Each sensor runs a local GPF that computes a global state estimate. The updating of the particle weights at each sensor uses the joint likelihood function, which is calculated in a distributed way, using only local communications, via the recently proposed likelihood consensus scheme. A significant reduction of the number of particles can be achieved by means of another consensus algorithm. The performance of the proposed distributed GPF is demonstrated for a target tracking problem.

Index Terms— Gaussian particle filter, distributed particle filter, likelihood consensus, target tracking, wireless sensor network.

1. INTRODUCTION

We consider sequential state estimation in a wireless sensor network. For a nonlinear/non-Gaussian state-space model, the *Gaussian particle filter* (GPF) [1] is an estimation method that is attractive in terms of complexity and performance. Here, we propose a *distributed GPF* (DGPF) algorithm that is fully decentralized: a global state estimate is available at each sensor, even though only local processing and communications are used and no fusion center is required.

In the proposed DGPF scheme, each sensor runs a *local* GPF (LGPF) that computes a *global* state estimate. The update operations at the individual LGPFs use the joint (all-sensors) likelihood function (JLF), which is provided to each sensor in a decentralized way by means of the *likelihood consensus* scheme recently introduced in [2]. A second consensus scheme can be applied to obtain a significant reduction of the number of particles employed by each LGPF and, hence, a substantial reduction of complexity. We note that consensus-based distributed particle filtering schemes were previously proposed in [3–5]. However, [3] can require a significantly larger number of consensus algorithms (one for each particle), and [4, 5] use only the local likelihood functions instead of the JLF.

This paper is organized as follows. In Section 2, we introduce the system model and review the principle of sequential Bayesian estimation. The DGPF and a reduced-complexity variant are described in Sections 3 and 4, respectively. Finally, Section 5 presents simulation results for a target tracking problem.

2. SYSTEM MODEL AND SEQUENTIAL ESTIMATION

We consider a generally nonlinear, non-Gaussian state-space model with additive Gaussian measurement noises. The random M-dimensional state vector $\mathbf{x}_n = (x_{n,1} \cdots x_{n,M})^\top$ evolves with discrete time n according to the state-transition equation

$$\mathbf{x}_n = \mathbf{g}_n(\mathbf{x}_{n-1}, \mathbf{u}_n)\,, \qquad n = 1, 2, \dots\,, \tag{1}$$

where $\mathbf{u}_n$ is white driving noise with a known probability density function (pdf) $f(\mathbf{u}_n)$. At time n, the state $\mathbf{x}_n$ is sensed by a sensor network with K sensors according to the measurement equations

$$\mathbf{z}_{n,k} = \mathbf{h}_{n,k}(\mathbf{x}_n) + \mathbf{v}_{n,k}\,, \qquad k = 1, 2, \dots, K\,. \tag{2}$$

Here, $\mathbf{z}_{n,k}$ of dimension $N_{n,k}$ is the measurement at time n and at sensor k, and $\mathbf{v}_{n,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{n,k})$ is zero-mean Gaussian measurement noise. We assume that (i) $\mathbf{v}_{n,k}$ and $\mathbf{v}_{n',k'}$ are independent unless $(n,k) = (n',k')$; (ii) the initial state $\mathbf{x}_0$ and the sequences $\mathbf{u}_n$ and $\mathbf{v}_{n,k}$ are all independent; and (iii) sensor k knows $\mathbf{g}_n(\cdot,\cdot)$, $\mathbf{h}_{n,k}(\cdot)$, and $\mathbf{Q}_{n,k}$ for all n (however, it need not know $\mathbf{h}_{n,k'}(\cdot)$ and $\mathbf{Q}_{n,k'}$ for $k' \neq k$). Hereafter, $\mathbf{z}_n \triangleq (\mathbf{z}_{n,1}^\top \cdots \mathbf{z}_{n,K}^\top)^\top$ will denote the vector containing the measurements of all sensors at time n, and $\mathbf{z}_{1:n} \triangleq (\mathbf{z}_1^\top \cdots \mathbf{z}_n^\top)^\top$ the vector of all measurements up to time n.

Equations (1) and (2) together with our statistical assumptions determine the *state-transition pdf* $f(\mathbf{x}_n|\mathbf{x}_{n-1})$, the *local* likelihood function $f(\mathbf{z}_{n,k}|\mathbf{x}_n)$, and the JLF $f(\mathbf{z}_n|\mathbf{x}_n)$. From (2) and the Gaussianity of $\mathbf{v}_{n,k}$, the local likelihood function of sensor k follows as $f(\mathbf{z}_{n,k}|\mathbf{x}_n) = C_{n,k} \exp\big(-\frac{1}{2}[\mathbf{z}_{n,k} - \mathbf{h}_{n,k}(\mathbf{x}_n)]^\top \mathbf{Q}_{n,k}^{-1} [\mathbf{z}_{n,k} - \mathbf{h}_{n,k}(\mathbf{x}_n)]\big)$, with $C_{n,k} \triangleq [(2\pi)^{N_{n,k}} \det\{\mathbf{Q}_{n,k}\}]^{-1/2}$. Due to the independence of all $\mathbf{v}_{n,k}$, we then obtain the JLF as

$$f(\mathbf{z}_n|\mathbf{x}_n) \,=\, \prod_{k=1}^{K} f(\mathbf{z}_{n,k}|\mathbf{x}_n) \,=\, C_n \exp\!\left(-\frac{1}{2}\, S_n(\mathbf{z}_n, \mathbf{x}_n)\right), \tag{3}$$

where $C_n \triangleq \prod_{k=1}^{K} C_{n,k}$ and

$$S_n(\mathbf{z}_n, \mathbf{x}_n) \,\triangleq\, \sum_{k=1}^{K} [\mathbf{z}_{n,k} - \mathbf{h}_{n,k}(\mathbf{x}_n)]^\top \mathbf{Q}_{n,k}^{-1} [\mathbf{z}_{n,k} - \mathbf{h}_{n,k}(\mathbf{x}_n)]\,. \tag{4}$$

To estimate the state $\mathbf{x}_n$ from all measurements, $\mathbf{z}_{1:n}$, we use the minimum mean-square error (MMSE) estimator [6]

$$\hat{\mathbf{x}}_n^{\text{MMSE}} \,\triangleq\, \mathrm{E}\{\mathbf{x}_n|\mathbf{z}_{1:n}\} \,=\, \int \mathbf{x}_n\, f(\mathbf{x}_n|\mathbf{z}_{1:n})\, d\mathbf{x}_n\,. \tag{5}$$

Based on the state-space model as expressed by the state-transition pdf $f(\mathbf{x}_n|\mathbf{x}_{n-1})$, the following recursion can be employed for sequentially calculating the current posterior pdf $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ from the previous posterior pdf $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$ and the JLF $f(\mathbf{z}_n|\mathbf{x}_n)$ [7]:

$$f(\mathbf{x}_n|\mathbf{z}_{1:n}) = \frac{f(\mathbf{z}_n|\mathbf{x}_n) \int f(\mathbf{x}_n|\mathbf{x}_{n-1}) f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})\, d\mathbf{x}_{n-1}}{f(\mathbf{z}_n|\mathbf{z}_{1:n-1})}\,. \tag{6}$$

A computationally feasible approximation to (5) and (6) is provided by the GPF [1]. The GPF approximates the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ by a Gaussian pdf, whose mean and covariance are calculated from a weighted particle set. In contrast to the standard particle filter [8], the GPF does not require resampling. This results in reduced complexity and allows for a parallel implementation [9].

3. DISTRIBUTED GAUSSIAN PARTICLE FILTERING

We now describe the proposed DGPF scheme. This scheme differs from the distributed particle filter in [2] mainly by the Gaussian approximation employed.

This work was supported by the FWF under Awards S10603 and S10609, by the NSF under Award CCF-1018323, and by the ONR under Award N00014-09-1-1154.

3.1. LGPF algorithm

Each sensor uses an LGPF to sequentially track the mean vector $\boldsymbol{\mu}_{n,k}$ and covariance matrix $\mathbf{C}_{n,k}$ of a Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n,k}, \mathbf{C}_{n,k})$ to the global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$. The LGPF at sensor k calculates a temporal sequence of state estimates $\hat{\mathbf{x}}_{n,k} = \boldsymbol{\mu}_{n,k}$ that are based on $\mathbf{z}_{1:n}$, i.e., the past and current measurements of *all* sensors. At a given time instant n, the LGPF at sensor k performs the following steps.

Step 1: J particles $\bar{\mathbf{x}}_{n-1,k}^{(j)}$, $j = 1, \ldots, J$ are randomly drawn from $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$.

Step 2: For each $\bar{\mathbf{x}}_{n-1,k}^{(j)}$, a new, "predicted" particle $\mathbf{x}_{n,k}^{(j)}$ is randomly drawn from $f(\mathbf{x}_n|\mathbf{x}_{n-1})\big|_{\mathbf{x}_{n-1}=\bar{\mathbf{x}}_{n-1,k}^{(j)}}$.

Step 3: An approximation $\tilde{f}(\mathbf{z}_n|\mathbf{x}_n)$ to the JLF $f(\mathbf{z}_n|\mathbf{x}_n)$ is computed by means of *likelihood consensus* (LC) as described in Section 3.2. This step requires communication with neighboring sensors.

Step 4: The weights associated with the predicted particles $\mathbf{x}_{n,k}^{(j)}$ drawn in Step 2 are calculated according to

$$w_{n,k}^{(j)} = \frac{\tilde{f}(\mathbf{z}_n|\mathbf{x}_{n,k}^{(j)})}{\sum_{j'=1}^{J} \tilde{f}(\mathbf{z}_n|\mathbf{x}_{n,k}^{(j')})}, \quad j = 1, \ldots, J. \tag{7}$$

This involves the approximate JLF $\tilde{f}(\mathbf{z}_n|\mathbf{x}_n)$ calculated in Step 3, which is evaluated at all predicted particles $\mathbf{x}_{n,k}^{(j)}$.

Step 5: From the weighted particles $\{\mathbf{x}_{n,k}^{(j)}, w_{n,k}^{(j)}\}_{j=1}^{J}$, a new mean $\boldsymbol{\mu}_{n,k}$ and a new covariance $\mathbf{C}_{n,k}$ are calculated as

$$\boldsymbol{\mu}_{n,k} = \sum_{j=1}^{J} w_{n,k}^{(j)} \mathbf{x}_{n,k}^{(j)}$$

$$\mathbf{C}_{n,k} = \sum_{j=1}^{J} w_{n,k}^{(j)} \mathbf{x}_{n,k}^{(j)} \mathbf{x}_{n,k}^{(j)\top} - \boldsymbol{\mu}_{n,k}\boldsymbol{\mu}_{n,k}^{\top}.$$

The state estimate $\hat{\mathbf{x}}_{n,k}$ (approximating $\hat{\mathbf{x}}_n^{\text{MMSE}}$ in (5)) is given by $\boldsymbol{\mu}_{n,k}$. We note that differences between the $\hat{\mathbf{x}}_{n,k}$ at different sensors k are only due to the random sampling of the particles (if they are obtained from nonsynchronized local random generators) and errors introduced by insufficiently converged LC.

Initialization: The recursive procedure defined by Steps 1–5 is initialized at time $n = 0$ by J particles $\mathbf{x}_{0,k}^{(j)}$ randomly drawn from an appropriate prior pdf, and by equal weights $w_{0,k}^{(j)} \equiv 1/J$.

3.2. Likelihood consensus

We will now review the LC scheme [2] that is used in Step 3 of Section 3.1 to provide an approximate JLF to each sensor. According to (3), calculation of the JLF $f(\mathbf{z}_n|\mathbf{x}_n)$ (up to the irrelevant factor C_n) reduces to calculation of $S_n(\mathbf{z}_n, \mathbf{x}_n)$ in (4). LC employs an iterative consensus algorithm (e.g., [10]) to perform a distributed approximate calculation of $S_n(\mathbf{z}_n, \mathbf{x}_n)$ that requires only local communications with neighboring sensors. This is based on a polynomial approximation of the sensor measurement function $\mathbf{h}_{n,k}(\mathbf{x}_n)$ in (2) (a more general formulation using an arbitrary basis expansion is also possible):

$$\mathbf{h}_{n,k}(\mathbf{x}_n) \approx \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n) \triangleq \sum_{\mathbf{r}=\mathbf{0}}^{R} \boldsymbol{\alpha}_{\mathbf{r},n,k}\, x_{n,1}^{r_1} \cdots x_{n,M}^{r_M}. \tag{8}$$

Here, $\mathbf{r} \triangleq (r_1 \cdots r_M) \in \{0, \ldots, R\}^M$; R is the degree of the multivariate polynomial $\tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)$; $\sum_{\mathbf{r}=\mathbf{0}}^{R}$ is short for $\sum_{r_1=0}^{R} \cdots \sum_{r_M=0}^{R}$ with constraint $\sum_{m=1}^{M} r_m \leq R$; and $\boldsymbol{\alpha}_{\mathbf{r},n,k} \in \mathbb{R}^{N_{n,k}}$ is the coefficient vector associated with $x_{n,1}^{r_1} \cdots x_{n,M}^{r_M}$. The calculation of $\tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)$ is performed locally at each sensor. While a simple method is a Taylor series expansion of $\mathbf{h}_{n,k}(\mathbf{x}_n)$ [2], we will here discuss an alternative approach which we observed to be more accurate (see the simulation results in Section 5).

More specifically, at time n, sensor k uses *least squares polynomial fitting* [11] based on the J data points $\{\mathbf{x}_{n,k}^{(j)}, \mathbf{h}_{n,k}(\mathbf{x}_{n,k}^{(j)})\}_{j=1}^{J}$, where the $\mathbf{x}_{n,k}^{(j)}$ are the predicted particles calculated in Step 2 of Section 3.1. To obtain an expression of the resulting coefficients $\boldsymbol{\alpha}_{\mathbf{r},n,k}$, we define the matrix $\mathbf{A}_{n,k} \in \mathbb{R}^{N_r \times N_{n,k}}$ that has the $\boldsymbol{\alpha}_{\mathbf{r},n,k}$ as its rows. Here, $N_r = \binom{R+M}{R}$ is the number of vectors $\mathbf{r} \in \{0, \ldots, R\}^M$ satisfying $\sum_{m=1}^{M} r_m \leq R$. Then [12]

$$\mathbf{A}_{n,k} = (\mathbf{X}_{n,k}^{\top}\mathbf{X}_{n,k})^{-1}\mathbf{X}_{n,k}^{\top}\mathbf{H}_{n,k}, \tag{9}$$

where the elements of $\mathbf{X}_{n,k} \in \mathbb{R}^{J \times N_r}$ equal $x_{n,1}^{r_1} \cdots x_{n,M}^{r_M}$ evaluated at $\mathbf{x}_{n,k}^{(j)}$ (each column corresponds to one of the N_r monomials $x_{n,1}^{r_1} \cdots x_{n,M}^{r_M}$ and each row corresponds to one of the J particles $\mathbf{x}_{n,k}^{(j)}$), and the rows of $\mathbf{H}_{n,k} \in \mathbb{R}^{J \times N_{n,k}}$ are the J vectors $\mathbf{h}_{n,k}(\mathbf{x}_{n,k}^{(j)}) \in \mathbb{R}^{N_{n,k}}$. We assume $J \geq N_r$, which is necessary for $\mathbf{X}_{n,k}$ to have full rank. The complexity of least squares polynomial fitting tends to be higher than that of a Taylor series expansion; however, it can be reduced by the scheme presented in Section 4.

Let $\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n) \triangleq \sum_{k=1}^{K} [\mathbf{z}_{n,k} - \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)]^{\top} \mathbf{Q}_{n,k}^{-1} [\mathbf{z}_{n,k} - \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)]$ denote the approximation to $S_n(\mathbf{z}_n, \mathbf{x}_n)$ obtained by substituting $\tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)$ for $\mathbf{h}_{n,k}(\mathbf{x}_n)$ in (4). This can be written as

$$\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n) = \sum_{k=1}^{K} \left[\sum_{\mathbf{r}=\mathbf{0}}^{2R} \beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k})\, x_{n,1}^{r_1} \cdots x_{n,M}^{r_M} \right],$$

where the $\beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k})$ depend on $\mathbf{z}_{n,k}$, $\mathbf{Q}_{n,k}$, and $\boldsymbol{\alpha}_{\mathbf{r},n,k}$ and can be calculated *locally* at sensor k (see [2] for details). We can finally express $\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)$ as a scalar-valued polynomial in $\mathbf{x}_n$, i.e.,

$$\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n) = \sum_{\mathbf{r}=\mathbf{0}}^{2R} T_{\mathbf{r},n}(\mathbf{z}_n)\, x_{n,1}^{r_1} \cdots x_{n,M}^{r_M}, \tag{10}$$

with coefficients

$$T_{\mathbf{r},n}(\mathbf{z}_n) = \sum_{k=1}^{K} \beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k}). \tag{11}$$

The vector of coefficients $\mathbf{T}_n(\mathbf{z}_n) \triangleq (T_{\mathbf{r},n}(\mathbf{z}_n))$ can be viewed as a *sufficient statistic* [6] that epitomizes the total measurement $\mathbf{z}_n$ and characterizes $\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)$ and, in turn, the approximate JLF $\tilde{f}(\mathbf{z}_n|\mathbf{x}_n) \triangleq C_n \exp\left(-\frac{1}{2}\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)\right)$. Hence, a sensor that knows $\mathbf{T}_n(\mathbf{z}_n)$ is able to evaluate $\tilde{f}(\mathbf{z}_n|\mathbf{x}_n)$ for any value of the state $\mathbf{x}_n$.

The sum over all sensors in (11) can be computed at each sensor by means of a consensus algorithm [10]. At a given time n, the resulting *LC scheme* [2] can be summarized as follows.

Step 1: Sensor k calculates the polynomial coefficients $\boldsymbol{\alpha}_{\mathbf{r},n,k}$ according to (9). Next, using the $\boldsymbol{\alpha}_{\mathbf{r},n,k}$ along with the *locally available* measurement $\mathbf{z}_{n,k}$ and noise covariance $\mathbf{Q}_{n,k}$, sensor k calculates the scalar coefficients $\beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k})$ as detailed in [2].

Step 2: The coefficients $\beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k})$ of all sensors k are added in a distributed, iterative way using a consensus algorithm. This involves communication with neighboring sensors (see [2] for details). One instance of the consensus algorithm is employed for each admissible $\mathbf{r}$; all instances are executed in parallel. After a sufficient number of consensus iterations, the $T_{\mathbf{r},n}(\mathbf{z}_n) = \sum_{k=1}^{K} \beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k})$ (see (11)) for all $\mathbf{r}$ are available at each sensor.

Step 3: Using the $T_{\mathbf{r},n}(\mathbf{z}_n)$, each sensor is now able to evaluate $\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)$ (see (10)) and, in turn, the approximate JLF $\tilde{f}(\mathbf{z}_n|\mathbf{x}_n) = C_n \exp\left(-\frac{1}{2}\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)\right)$, for any value of $\mathbf{x}_n$.

4. DGPF WITH REDUCED COMPLEXITY

We next present a reduced-complexity variant of the DGPF, referred to as DGPF-R. Each of the K LGPFs uses a reduced number $J' \triangleq J/K$ of particles (J is chosen such that J' is an integer and $J' \geq N_r$, which is necessary for $\mathbf{X}_{n,k} \in \mathbb{R}^{J' \times N_r}$ to have full rank). The sets of J' particles of all LGPFs are effectively combined such that a "virtual global GPF" with J particles is obtained. This results in LGPFs with a substantially reduced complexity, at the cost of some increase in local communications. The performance (estimation accuracy) of the DGPF-R is effectively equal to that of the DGPF.

4.1. The DGPF-R

The DGPF-R is similar to a parallel implementation of the GPF proposed in [9], which uses multiple processing units (corresponding to our sensor nodes) colocated with a central unit. However, instead of a central unit, the DGPF-R employs a consensus algorithm to combine the partial estimates (means) and partial covariances calculated at the individual sensors. Furthermore, in contrast to [9], the DGPF-R uses the approximate JLF obtained via the LC scheme.

At time instant n, the LGPF at sensor k first performs Steps 1–3 described in Section 3.1, with the difference that the number of particles is J' rather than J. The remaining steps, described in the following, are modified versions of Steps 4 and 5 of Section 3.1, as well as an additional consensus step.

Step 4: The approximate JLF $\tilde{f}(\mathbf{z}_n|\mathbf{x}_n)$, which was calculated in Step 3 using LC, is evaluated at the J' predicted particles $\mathbf{x}_{n,k}^{(j)}$, $j = 1, \ldots, J'$ drawn in Step 2. This yields the nonnormalized weights (cf. (7))

$$\tilde{w}_{n,k}^{(j)} = \tilde{f}(\mathbf{z}_n|\mathbf{x}_{n,k}^{(j)}), \quad j = 1, \ldots, J'.$$

Furthermore, the sum of these weights is computed:

$$\widetilde{W}_{n,k} = \sum_{j=1}^{J'} \tilde{w}_{n,k}^{(j)}.$$

Step 5: From the weighted particles $\{\mathbf{x}_{n,k}^{(j)}, \tilde{w}_{n,k}^{(j)}\}_{j=1}^{J'}$, a partial nonnormalized mean and a partial nonnormalized correlation are calculated as

$$\boldsymbol{\mu}'_{n,k} = \sum_{j=1}^{J'} \tilde{w}_{n,k}^{(j)} \mathbf{x}_{n,k}^{(j)}, \quad \mathbf{R}'_{n,k} = \sum_{j=1}^{J'} \tilde{w}_{n,k}^{(j)} \mathbf{x}_{n,k}^{(j)} \mathbf{x}_{n,k}^{(j)\top}.$$

Note that Steps 4 and 5 are carried out locally at sensor k.

Step 6: The partial means and correlations from all sensors are combined to obtain the global mean and covariance:

$$\boldsymbol{\mu}_n = \frac{1}{W_n} \sum_{k=1}^{K} \boldsymbol{\mu}'_{n,k}, \quad \mathbf{C}_n = \frac{1}{W_n} \sum_{k=1}^{K} \mathbf{R}'_{n,k} - \boldsymbol{\mu}_n \boldsymbol{\mu}_n^\top, \quad (12)$$

where

$$W_n = \sum_{k=1}^{K} \widetilde{W}_{n,k} \quad (13)$$

is the global sum of particle weights. The sums in (12) and (13) are computed in a distributed manner by a consensus algorithm [10]. The normalization by W_n and subtraction of $\boldsymbol{\mu}_n \boldsymbol{\mu}_n^\top$ are performed locally at each sensor after convergence of the consensus algorithm.

It is easily seen that $\boldsymbol{\mu}_n$ and $\mathbf{C}_n$ are actually the result of averaging over J particles ($J' = J/K$ particles are drawn independently at each of the K sensors). All sensors obtain identical $\boldsymbol{\mu}_n$ and $\mathbf{C}_n$, provided that the consensus algorithms are sufficiently converged. The state estimate $\hat{\mathbf{x}}_n$ (approximating $\hat{\mathbf{x}}_n^{\text{MMSE}}$ in (5)) is given by $\boldsymbol{\mu}_n$.

4.2. Comparison of DGPF and DGPF-R

If the consensus algorithms used for computing W_n, $\boldsymbol{\mu}_n$, and $\mathbf{C}_n$ are sufficiently converged, the performance (estimation accuracy) of the DGPF-R is effectively equal to that of the DGPF. In what follows, we will compare the complexity and communication requirements of the DGPF and DGPF-R. In this comparison, we do not consider the complexity and communication requirements of Step 2 of the LC (see Section 3.2), since they are identical for both schemes.

The complexities of the LGPF algorithm and of least squares polynomial fitting both depend linearly on the number of particles [9, 12]. Thus, reducing the number of particles at each sensor from J to $J' = J/K$ reduces these complexities by a factor of K. It follows that the DGPF-R is significantly less complex than the DGPF. We note that the complexity of the additional consensus algorithms required by the DGPF-R is typically negligible in comparison to the other operations performed. The additional communication requirements are determined by the speed of convergence (i.e., number of iterations) of the additional consensus algorithms, which depends mainly on the second smallest eigenvalue of the Laplacian of the communication graph [13]; they are thus application-dependent.

We finally note the following tradeoff between latency and power consumption. The reduced operation count of the DGPF-R can be exploited for reducing the processing time. Thereby, the latency of the DGPF-R is reduced relative to the DGPF, provided that the delays caused by the additional consensus algorithms are not too large. Thus, the DGPF-R may be more suitable for real-time applications; however, the power consumption is higher due to the increased communication requirements. Alternatively, if latency is not an issue, the reduced operation count of the DGPF-R can be exploited for reducing the processor clock frequency. The processing time then equals that of the DGPF; however, the processor power consumption is reduced. Thereby, the overall power consumption of the DGPF-R is reduced relative to the DGPF, provided that the additional power consumption due to the increased communication is not too large.

5. SIMULATION RESULTS

We consider a target tracking application in which the state $\mathbf{x}_n = (x_n \; y_n \; v_x \; v_y)^\top$ represents the 2D position and 2D velocity of a target in the x-y plane. The state-transition equation (1) is given by $\mathbf{x}_n = \mathbf{G}\mathbf{x}_{n-1} + \mathbf{u}_n$, $n = 1, 2, \ldots$ [1], where

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0.2 & 0 \\ 0 & 1 & 0 & 0.2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and $\mathbf{u}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_u)$ with $\mathbf{C}_u = \text{diag}(1, 1, 0.01, 0.01)$. The target emits a sound of constant amplitude $A = 40$, which is sensed by acoustic amplitude sensors [14]. The (scalar) measurement $z_{n,k}$ of sensor k is given by (cf. (2))

$$z_{n,k} = h_{n,k}(\mathbf{x}_n) + v_{n,k} = \frac{A}{\|(x_n \; y_n)^\top - \boldsymbol{\xi}_{n,k}\|} + v_{n,k}, \quad (14)$$

where $\boldsymbol{\xi}_{n,k}$ denotes the position of sensor k at time n and $v_{n,k} \sim \mathcal{N}(0, \sigma_v^2)$. The network is composed of $K = 25$ sensors that are deployed on a jittered grid within a rectangular region of size 200 m $\times$ 200 m. Each sensor communicates with other sensors within a range of 90 m.

For LC, we approximate $h_{n,k}(\mathbf{x}_n)$ in (14) by a polynomial (see (8)) of degree $R = 2$. The sums in (11) are computed by an average consensus algorithm using the Metropolis weight model [15]. As a performance benchmark, we also report the results obtained by means of exact, direct calculation of (11). The same remarks apply to the sums in (12) and (13) that are required by the DGPF-R.

We compare the DGPF, the DGPF-R, and a centralized GPF (CGPF) that processes all sensor measurements at a fusion center.

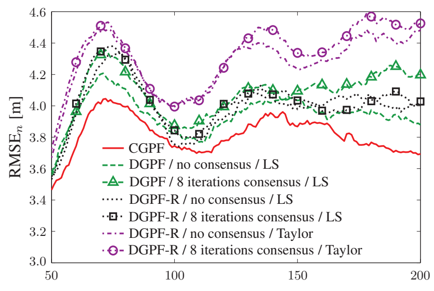

Fig. 1. RMSE$_n$ versus time n.

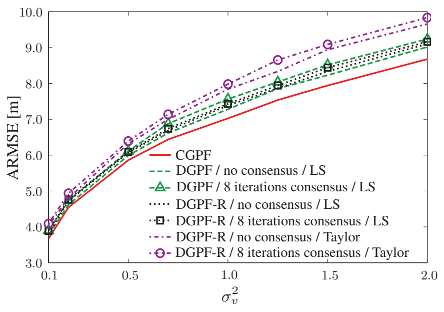

Fig. 2. ARMSE versus measurement noise variance σ_v^2.

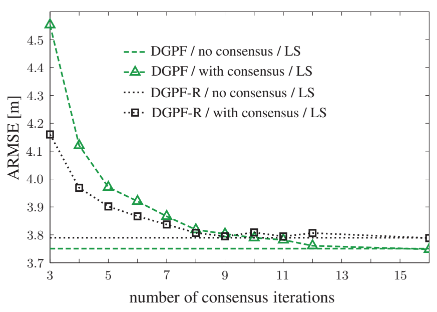

Fig. 3. ARMSE versus number of consensus iterations.

The number of particles at each sensor of the DGPF and at the fusion center of the CGPF is $J = 1000$, whereas the DGPF-R scheme employs only $J' = 40$ particles per sensor. As a performance measure, we use the root-mean-square error of the state estimate $\hat{\mathbf{x}}_{n,k}$, denoted RMSE$_n$, which is computed as the square root of the average of the squared estimation error over all sensors and over 4000 simulation runs. We also compute the *average* RMSE (ARMSE) by averaging RMSE$_n^2$ over all 200 simulated time instants n.

Fig. 1 shows the temporal evolution of RMSE$_n$ for $n = 50, \ldots, 200$, at a measurement noise variance of $\sigma_v^2 = 0.1$. The sums in (11)–(13) were computed by an average consensus algorithm in 8 iterations (denoted as "8 iterations consensus") and by exact, direct calculation ("no consensus"). We also display the performance obtained by using a Taylor approximation of $h_{n,k}(\mathbf{x}_n)$ ("Taylor") instead of least squares polynomial fitting ("LS"). In Fig. 2, we show the ARMSE versus the measurement noise variance σ_v^2.

From both figures, we observe that the performance of DGPF and DGPF-R is almost as good as that of CGPF. We also see that the least squares polynomial fitting outperforms the Taylor approximation. Furthermore, for the "no consensus" setting, using least squares polynomial fitting, DGPF performs slightly better than DGPF-R; this can be explained by the larger number of data points used by DGPF for least squares polynomial fitting. Somewhat surprisingly, DGPF-R outperforms DGPF for the "8 iterations consensus" setting, i.e., the additional consensus algorithm that is used to calculate the sums (12) and (13) results in a better performance of DGPF-R, in spite of the reduced number of particles. This can be explained as follows. The LC with only 8 consensus iterations is not completely converged, i.e., the local information is not completely diffused throughout the network and the resulting approximate JLF does not contain the complete global information. The additional consensus of DGPF-R then helps to further diffuse the local information. Of course, for the "no consensus" setting, the additional consensus does not improve the performance.

This behavior can also be observed in Fig. 3, which depicts the dependence of ARMSE on the number of consensus iterations. As long as the number of iterations is small, DGPF-R (with consensus) outperforms DGPF (with consensus). However, as the number of iterations increases, both schemes approach the respective performance of the "no consensus" setting and DGPF (slightly) outperforms DGPF-R. Note that in DGPF-R, the same number of iterations is used for each of the two consensus stages.

6. CONCLUSION

We presented a consensus-based, distributed Gaussian particle filter (GPF) for wireless sensor networks. At each sensor, a local GPF (LGPF) computes a global state estimate that reflects the past and present measurements of all sensors. An approximate joint likelihood function epitomizing all measurements is provided to each LGPF in a distributed way by means of the recently proposed likelihood consensus scheme, using only local communications between nearby sensors. We also presented a variant of the distributed GPF in which the number of particles used by each LGPF is significantly reduced while leaving the estimation accuracy effectively unchanged.

The performance of the proposed distributed GPF was assessed for a target tracking application. Our results indicate good performance even in comparison with a centralized GPF. We finally note that the proposed methods can be extended to a consensus-based, distributed implementation of a generalization of the GPF known as the *Gaussian sum particle filter* [16].

7. REFERENCES

[1] J. H. Kotecha and P. M. Djurić, "Gaussian particle filtering," *IEEE Trans. Signal Process.*, vol. 51, pp. 2592–2601, Oct. 2003.

[2] O. Hlinka, O. Slučiak, F. Hlawatsch, P. M. Djurić, and M. Rupp, "Likelihood consensus: Principles and application to distributed particle filtering," in *Proc. 44th Asilomar Conf. Sig., Syst., Comp.*, Pacific Grove, CA, Nov. 2010.

[3] S. Farahmand, S. I. Roumeliotis, and G. B. Giannakis, "Particle filter adaptation for distributed sensors via set membership," in *Proc. IEEE ICASSP-10*, Dallas, TX, pp. 3374–3377, Mar. 2010.

[4] D. Gu, "Distributed particle filter for target tracking," in *Proc. IEEE ICRA-07*, Rome, Italy, pp. 3856–3861, Apr. 2007.

[5] D. Gu, J. Sun, Z. Hu, and H. Li, "Consensus based distributed particle filter in sensor networks," in *Proc. IEEE ICIA-08*, Zhangjiajie, P. R. China, pp. 302–307, June 2008.

[6] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ: Prentice-Hall, 1993.

[7] B. Ristic and S. Arulampalam, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Boston, MA: Artech House, 2004.

[8] A. Doucet, N. De Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York, NY: Springer, 2001.

[9] M. Bolić, A. Athalye, S. Hong, and P. M. Djurić, "Study of algorithmic and architectural characteristics of Gaussian particle filters," *J. Sig. Process. Syst.*, vol. 61, pp. 205–218, Nov. 2009.

[10] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[11] F. H. Lesh, "Multi-dimensional least-squares polynomial curve fitting," *Communications of the ACM*, vol. 2, pp. 29–30, Sep. 1959.

[12] Å. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA: SIAM, 1996.

[13] M. Fiedler, "Algebraic connectivity of graphs," *Czech. Math. J.*, vol. 23, no. 2, pp. 298–305, 1973.

[14] F. Zhao and L. J. Guibas, *Wireless Sensor Networks: An Information Processing Approach*. Amsterdam, The Netherlands: Morgan Kaufmann, 2004.

[15] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. IEEE IPSN-05*, Los Angeles, CA, pp. 63–70, Apr. 2005.

[16] J. H. Kotecha and P. M. Djurić, "Gaussian sum particle filtering," *IEEE Trans. Signal Process.*, vol. 51, pp. 2602–2612, Oct. 2003.