 IEEE Signal Processing Magazine, vol. 30, no. 1, Jan. 2013, pp. 61--81

# Distributed Particle Filtering in Agent Networks: A Survey, Classification, and Comparison

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

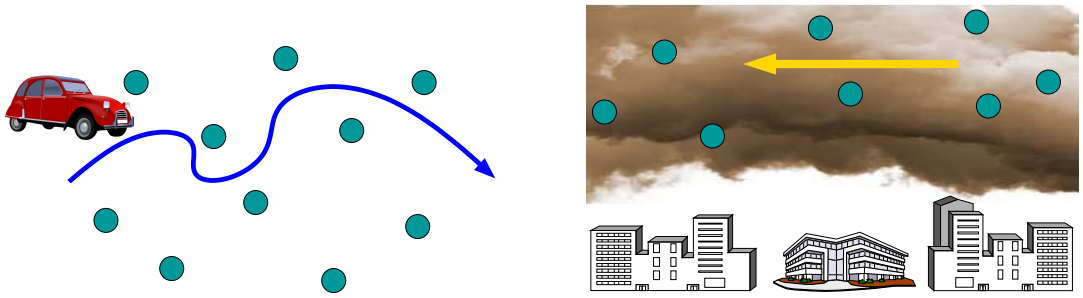
[1]Institute of Telecommunications, Vienna University of Technology, Vienna, Austria
{ondrej.hlinka, franz.hlawatsch}@nt.tuwien.ac.at

[2]Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, USA
petar.djuric@stonybrook.edu

Distributed particle filter (DPF) algorithms are sequential state estimation algorithms that are executed by a set of agents. Some or all of the agents perform local particle filtering and interact with other agents in order to calculate a global state estimate. DPF algorithms are attractive for large-scale, nonlinear, and non-Gaussian distributed estimation problems that often occur in applications involving agent networks (ANs). In this article, we present a survey, classification, and comparison of various DPF approaches and algorithms available to date. Our emphasis is on decentralized ANs that do not include a central processing or control unit.

## 1 Introduction

Particle filtering is a Monte Carlo approximation of optimal sequential Bayesian state estimation [1–4]. During the early 1990s, particle filtering turned to a filtering methodology of choice for nonlinear and non-Gaussian systems. Since then, it has captured the attention of researchers and practitioners in various fields including signal processing, control, statistics, and econometrics. In the first decade of this century, a growing interest in ANs promoted intensive research on DPFs. In many applications of ANs, the physical systems include nonlinear and non-Gaussian elements, and hence, particle filtering algorithms are a natural choice for sequential signal processing. However, for a decentralized implementation, substantial modifications and extensions of particle filtering algorithms are typically required. Because the measurements are dispersed among the agents, rather than available at one central location, diffusing the locally available information throughout



(a) (b)

Figure 1: Two examples of DPF applications in ANs: (a) target tracking, (b) chemical plume tracking. The agents are indicated by circles.

the network or a relevant part thereof is an essential component of DPF algorithms.

A standard application of DPFs in ANs is target tracking [5] (see Figure 1(a)). A noncooperative target, such as a vehicle, aircraft, person, or animal moves through an area where the AN is deployed. The target emits a signal that is sensed by the agents, and a DPF estimates (tracks) time-varying properties of the target such as its position and velocity. A second application example is the tracking of chemical plumes [5] (see Figure 1(b)). Here, micro aerial vehicles self-organize into an ad-hoc airborne AN and execute a DPF algorithm to estimate time-varying properties of the plume such as overall position, size, shape, and velocity.

## 2 Distributed Estimation in Agent Networks

Some examples of ANs are wireless sensor networks [5], sensor/actuator networks [6], robotic networks [7], networks of unmanned aerial vehicles (UAVs) [8], and networks of cameras [9]. Possible applications of ANs include environmental and agricultural monitoring [10], health-care monitoring [11], target tracking [5], pollution source localization [12], chemical plume tracking [5], and surveillance [9]. The agents may range from small, inexpensive, battery-powered sensor units equipped with limited computation and communication capabilities to resource-rich mobile robots or UAVs capable of performing complex tasks. Each agent contains (some of) the following elements: one or several sensors, communication interface, processing unit, and actuators. The on-board sensors measure physical quantities such as temperature, pressure, humidity, concentration of chemicals, distance, velocity, light intensity, vibration amplitude, or received signal power. The actuator elements allow the agents to act on the environment (e.g., turn on a water sprinkler to stop a fire) or on themselves (e.g., perform a controlled movement).

In the application context considered in this article, the agents cooperatively estimate certain

parameters (or states) of the surrounding environment based on their local measurements. They need to cooperate because their local measurements are usually insufficient for obtaining reliable estimates. This is where DPF algorithms come into play. Compared to other distributed sequential estimation algorithms, DPF algorithms typically offer superior performance in nonlinear and non-Gaussian systems.

For distributed estimation algorithms, communication aspects of the underlying AN are of central importance. These aspects concern the communication topology (which agents are connected by communication links) and the properties of the communication links (data rate, reliability, latency). The communication topology is commonly described by a communication graph, as briefly discussed in "Communication Graph of an Agent Network." For simplicity, we will usually assume the communication links to be error free.

---

### Communication Graph of an Agent Network

The communication topology of an AN can be described by a communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose vertex set (or node set) $\mathcal{V}$ comprises the agents while each edge $(k, k') \in \mathcal{E}$ in the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ indicates an undirected communication link between agents $k$ and $k'$. In the example shown in Figure 2, the vertices are depicted by circles and the edges by lines connecting these circles.

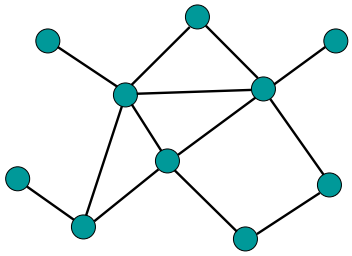


Figure 2: Example of a communication graph.

The *neighborhood* of agent $k$ is the set of agents $\mathcal{N}_k \subseteq \mathcal{V}$ defined as $\mathcal{N}_k \triangleq \{k' \in \mathcal{V} | (k, k') \in \mathcal{E}\}$, i.e., comprising those agents that are connected via a communication link to agent $k$. Usually, these neighboring agents (or briefly neighbors) are located spatially close to agent $k$. An AN is said to be *connected* if its communication graph is connected, i.e., if there exists at least one path (sequence of consecutive edges) between any two vertices in the graph. (The graph shown in Figure 2 is connected.) The *diameter* of the communication graph is the greatest distance between any two vertices. The distance between two vertices is defined as the number of edges in a shortest path connecting them. (The diameter of the graph in Figure 2 is 4.)

---

Most distributed discrete-time sequential estimation algorithms presuppose synchronization, i.e., the availability of a common clock or time base at each agent, whereas some approaches (e.g., [13]) relax this requirement. Various distributed synchronization algorithms have been proposed [14]. Furthermore, it is usually assumed that the locations of the agents are known. Distributed

algorithms for agent self-localization, especially for mobile agents, have been proposed, e.g., in [15] and [16]. Additional aspects influencing the design of distributed estimation algorithms for ANs include energy constraints (limited battery capacity of the agents), computation constraints (limited processing power and on-board memory), and communication constraints (limited transmission rate and range, intermittent connectivity, bit errors, and latency constraints). The algorithms may also have to meet application-dependent requirements related to operational lifetime, latency/reaction time, robustness to link and node failure, agent mobility, and scalability. Several of these constraints and requirements are inherently conflicting and thus lead to design tradeoffs.

## 3 Sequential Bayesian Estimation

Because DPFs perform a distributed variant of sequential Bayesian estimation, we review the principles of (centralized) sequential Bayesian estimation [2] first. Consider a time-dependent state vector $\mathbf{x}_n$, $n$ being a discrete time index, that evolves according to the *system model*

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

Here, $\mathbf{g}_n(\cdot,\cdot)$ is a known, generally nonlinear function and $\mathbf{u}_n$ is white driving noise that is independent of the past and present states and whose probability density function (pdf) is known. At time $n$, a measurement vector $\mathbf{z}_n$ is observed, which is related to $\mathbf{x}_n$ via the *measurement model*

$$\mathbf{z}_n = \mathbf{h}_n(\mathbf{x}_n, \mathbf{v}_n)\,, \qquad n = 1, 2, \ldots\,. \tag{2}$$

Here, $\mathbf{h}_n(\cdot,\cdot)$ is a known, generally nonlinear function and $\mathbf{v}_n$ is white measurement noise that is independent of the past and present states and of the driving noise and whose pdf is known. In what follows, we write $\mathbf{z}_{1:n} \triangleq (\mathbf{z}_1^\top \cdots \mathbf{z}_n^\top)^\top$ for the vector of all measurements up to time $n$. Equations (1) and (2) together with our statistical assumptions determine a probabilistic formulation of the system model by the state-transition pdf $f(\mathbf{x}_n|\mathbf{x}_{n-1})$ and of the measurement model by the likelihood function $f(\mathbf{z}_n|\mathbf{x}_n)$, both of which are allowed to be time-varying. From our statistical assumptions, it also follows that $f(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_{1:n-1}) = f(\mathbf{x}_n|\mathbf{x}_{n-1})$, i.e., the state $\mathbf{x}_n$ is conditionally independent of all past measurements, $\mathbf{z}_{1:n-1}$, given the previous state $\mathbf{x}_{n-1}$, and that $f(\mathbf{z}_n|\mathbf{x}_n, \mathbf{z}_{1:n-1}) = f(\mathbf{z}_n|\mathbf{x}_n)$, i.e., the measurement $\mathbf{z}_n$ is conditionally independent of all past measurements, $\mathbf{z}_{1:n-1}$, given the current state $\mathbf{x}_n$.

The task we consider is the estimation of the state $\mathbf{x}_n$ from all measurements up to time $n$, $\mathbf{z}_{1:n}$, in a sequential (recursive) manner that reuses previously obtained results. The Bayesian approach to sequential estimation is to calculate the posterior pdf $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ (hereafter briefly termed "posterior"). From $f(\mathbf{x}_n|\mathbf{z}_{1:n})$, one can then calculate various estimates of $\mathbf{x}_n$. For example,

the minimum mean-square error (MMSE) estimator of $\mathbf{x}_n$ is obtained as the mean of $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ [2,17]

$$\hat{\mathbf{x}}_n^{\mathrm{MMSE}} \triangleq \mathrm{E}\{\mathbf{x}_n|\mathbf{z}_{1:n}\} = \int \mathbf{x}_n f(\mathbf{x}_n|\mathbf{z}_{1:n})\, d\mathbf{x}_n\,, \qquad n = 1, 2, \ldots\,. \tag{3}$$

Based on the above model, it can be shown [2] that the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ can be calculated sequentially from the previous posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$ and the measurement vector $\mathbf{z}_n$, in two steps. In the *prediction step*, the "predicted posterior" $f(\mathbf{x}_n|\mathbf{z}_{1:n-1})$ is calculated from the previous posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$ and the state-transition pdf $f(\mathbf{x}_n|\mathbf{x}_{n-1})$ according to

$$f(\mathbf{x}_n|\mathbf{z}_{1:n-1}) = \int f(\mathbf{x}_n|\mathbf{x}_{n-1})\, f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})\, d\mathbf{x}_{n-1}\,, \qquad n = 1, 2, \ldots\,. \tag{4}$$

In the *update step*, the predicted posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n-1})$ is converted to the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ according to

$$f(\mathbf{x}_n|\mathbf{z}_{1:n}) = \frac{f(\mathbf{z}_n|\mathbf{x}_n)\, f(\mathbf{x}_n|\mathbf{z}_{1:n-1})}{f(\mathbf{z}_n|\mathbf{z}_{1:n-1})}\,, \qquad n = 1, 2, \ldots\,. \tag{5}$$

Note that this involves the likelihood function $f(\mathbf{z}_n|\mathbf{x}_n)$. The recursion for $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ established by (4) and (5) is initialized by $f(\mathbf{x}_n|\mathbf{z}_{1:n})\big|_{n=0} = f(\mathbf{x}_0)$, i.e., the prior pdf of $\mathbf{x}_0$.

A straightforward calculation of relations (3)–(5) is usually infeasible, since an analytical solution is in most cases unavailable and a numerical implementation involves the computation of multi-dimensional integrals. An important exception is the special case of linear system and measurement models with Gaussian driving and measurement noises and a Gaussian prior $f(\mathbf{x}_0)$. Here, the predicted posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n-1})$ and the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ are Gaussian. The Kalman filter (KF) relations [17,18] can then be used to update the mean vector and covariance matrix of these pdfs in a recursive manner that is consistent with the general recursion (4), (5), and the updated mean directly yields the MMSE estimate (3). For nonlinear and non-Gaussian problems, the extended KF [17,18] is a common suboptimal solution; however, the underlying approximations may lead to large errors and even divergence. Other suboptimal approaches include the Gaussian sum filter [19] and the unscented (sigma-point) KF [20]. The latter outperforms the extended KF in terms of performance and ease of implementation. Yet another class of methods is based on evaluating the pdfs on a grid in the state space [2]. Usually, the choice of an efficient grid is nontrivial, and a very large number of grid points may be required.

## 4 Particle Filtering

Particle filters (PFs) [1–4] tend to outperform the above-mentioned methods. They can handle any nonlinearity and any distributions of the driving and measurement noises, and they work well in many situations in which KF-based methods diverge. On the other hand, they are more

computationally complex than the extended KF and the unscented KF, although less complex than grid-based methods. A PF performs a Monte Carlo simulation-based approximation of optimal sequential Bayesian estimation in (3)–(5). The non-Gaussian posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ is represented by a set $\left\{\left(\mathbf{x}_n^{(j)}, w_n^{(j)}\right)\right\}_{j=1}^{J}$ of randomly drawn samples or *particles* $\mathbf{x}_n^{(j)}$ and corresponding weights $w_n^{(j)}$, which establishes a discrete approximation of the posterior, $f(\mathbf{x}_n|\mathbf{z}_{1:n}) \approx \sum_{j=1}^{J} w_n^{(j)} \delta(\mathbf{x}_n - \mathbf{x}_n^{(j)})$ (here, $\delta(\cdot)$ denotes the multidimensional Dirac delta function). Using this particle representation, one can obtain various estimates of $\mathbf{x}_n$. In particular, the MMSE estimate (3) is approximated as

$$\hat{\mathbf{x}}_n^{\mathrm{MMSE}} \approx \hat{\mathbf{x}}_n \triangleq \sum_{j=1}^{J} w_n^{(j)} \mathbf{x}_n^{(j)} . \tag{6}$$

This approximation is accurate if the number of particles $\mathbf{x}_n^{(j)}$ located in regions of significant probability mass is sufficiently large and if the weights $w_n^{(j)}$ are calculated appropriately.

At time $n$, the PF recursively updates the previous particles $\mathbf{x}_{n-1}^{(j)}$ and weights $w_{n-1}^{(j)}$ using the observation $\mathbf{z}_n$. More specifically, the particle representation of the posterior is used to approximate the prediction step (4) and update step (5). This is done by means of *importance sampling*, whereby the samples $\mathbf{x}_n^{(j)}$ are randomly drawn from a specified pdf (a so-called proposal pdf) that is different from the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$. The prediction step (4) is performed by sampling from a proposal pdf $q(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n)$, thus obtaining particles that are used to approximate the predicted posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n-1})$. The update step (5) is performed by computing the particle weights $w_n^{(j)}$ using the likelihood function $f(\mathbf{z}_n|\mathbf{x}_n)$. In "Generic Particle Filter (SIR Filter) Algorithm," we present a generic PF algorithm, also known as the sequential importance resampling (SIR) filter, which yields an approximation of the MMSE estimate $\hat{\mathbf{x}}_n^{\mathrm{MMSE}}$.

### Generic Particle Filter (SIR Filter) Algorithm

At time $n=0$, the algorithm is initialized by $J$ particles $\mathbf{x}_0^{(j)}$, $j = 1, \ldots, J$, drawn from the prior pdf $f(\mathbf{x}_0)$. The weights are initially equal, i.e., $w_0^{(j)} = 1/J$ for all $j$. At time $n \geq 1$, the following steps are performed:

1. *Prediction step:* For each previous particle $\mathbf{x}_{n-1}^{(j)}$, a new particle $\mathbf{x}_n^{(j)}$ is sampled from a suitably chosen proposal pdf $q(\mathbf{x}_n|\mathbf{x}_{n-1}^{(j)}, \mathbf{z}_n) \equiv q(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n)\big|_{\mathbf{x}_{n-1}=\mathbf{x}_{n-1}^{(j)}}$.

2. *Update step:* Nonnormalized weights associated with the particles $\mathbf{x}_n^{(j)}$ drawn in Step 1 are calculated according to

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

   The weights are then normalized according to $w_n^{(j)} = \tilde{w}_n^{(j)} / \sum_{j'=1}^{J} \tilde{w}_n^{(j')}$. The set of particles and weights $\left\{\left(\mathbf{x}_n^{(j)}, w_n^{(j)}\right)\right\}_{j=1}^{J}$ represents the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$.

3. *Calculation of estimate:* From $\left\{\left(\mathbf{x}_n^{(j)}, w_n^{(j)}\right)\right\}_{j=1}^{J}$, an approximation of the MMSE state estimate is computed according to (6), i.e., $\hat{\mathbf{x}}_n = \sum_{j=1}^{J} w_n^{(j)} \mathbf{x}_n^{(j)}$.

4. *Resampling:* The set $\left\{\left(\mathbf{x}_n^{(j)}, w_n^{(j)}\right)\right\}_{j=1}^{J}$ can be resampled if necessary (see, e.g., [2] for indications when resampling should be performed). The resampled particles are obtained by sampling with replacement from the set $\left\{\mathbf{x}_n^{(j)}\right\}_{j=1}^{J}$, where $\mathbf{x}_n^{(j)}$ is sampled with probability $w_n^{(j)}$. This produces $J$ resampled particles $\mathbf{x}_n^{(j)}$. The weights are redefined to be identical, i.e., $w_n^{(j)} = 1/J$.

---

From the SIR filter, some well-known PFs can be derived by specific choices of the proposal pdf and/or modifications of the resampling step. The bootstrap filter [1] uses the state-transition pdf $f(\mathbf{x}_n|\mathbf{x}_{n-1})$—which does not involve the measurement $\mathbf{z}_n$—as the proposal pdf and performs resampling at each time step. The auxiliary PF [2, 21] draws particles jointly with samples of an additional ("auxiliary") variable, with the goal of improving the estimation performance. In the regularized PF [2], resampled particles are drawn from a kernel-based estimate of the posterior that is derived from the original particles. Further PF variants are obtained through other modifications. The Gaussian PF [22] approximates the posterior by a Gaussian pdf that is calculated from the particles. The computations are simplified by the Gaussian approximation, while the particle-based posterior update still leads to improved performance compared to KF-based methods. In a Rao-Blackwellized PF [4], some components of the state $\mathbf{x}_n$ are integrated out analytically. This results in a reduced dimension of the state space, which allows for a reduction of the number of particles.

For good performance of a PF, the proposal pdf $q(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n)$ should be close to the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ [2, 3]. Using the state-transition pdf $f(\mathbf{x}_n|\mathbf{x}_{n-1})$ as the proposal pdf, as is done in the bootstrap filter, is simple but does not take the measurement $\mathbf{z}_n$ into account. Thus, many of the resulting particles may be located in regions with insignificant likelihood and, thus, posterior. To incorporate $\mathbf{z}_n$ into the proposal pdf and obtain particles in regions of high posterior, a process known as proposal adaptation can be used. The optimal (adapted) proposal pdf is given by $q_{\mathrm{opt}}(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n) \propto f(\mathbf{z}_n|\mathbf{x}_n) f(\mathbf{x}_n|\mathbf{x}_{n-1})$ followed by normalization [4], which is usually difficult to compute. Suboptimal adaptation methods use a Gaussian or Gaussian mixture (GM) representation of the proposal pdf, which is calculated by a simpler nonlinear filter. In particular, a PF whose proposal pdf is adapted using an unscented KF is referred to as unscented PF [20].

## 5 Distributed Sequential Bayesian Estimation

In a distributed AN setting, the measurements are dispersed among the agents. Let us consider an AN consisting of $K$ agents. At time $n$, agent $k \in \{1, \ldots, K\}$ observes a local measurement vector $\mathbf{z}_{n,k}$, which is related to the state $\mathbf{x}_n$ via the *local measurement model* (cf. (2))

$$\mathbf{z}_{n,k} = \mathbf{h}_{n,k}(\mathbf{x}_n, \mathbf{v}_{n,k}), \qquad n = 1, 2, \ldots. \tag{8}$$

Here, $\mathbf{h}_{n,k}(\cdot,\cdot)$ is a known, agent-dependent, generally nonlinear function and $\mathbf{v}_{n,k}$ is a local measurement noise that is white and independent of the past and present states and of the driving noise $\mathbf{u}_n$ in (1). The global (all-agents) measurement vector $\mathbf{z}_n$ in (2) is now given by the collection of all local measurement vectors in (8), i.e., $\mathbf{z}_n \triangleq (\mathbf{z}_{n,1}^\top \cdots \mathbf{z}_{n,K}^\top)^\top$. Equation (8) together with our statistical assumptions determines the *local likelihood function* $f(\mathbf{z}_{n,k}|\mathbf{x}_n)$ of agent $k$. In addition, most methods for distributed sequential Bayesian estimation assume that $\mathbf{v}_{n,k}$ is independent of the local measurement noises of the other agents, $\mathbf{v}_{n,k'}$ for $k' \neq k$. This implies that the *global likelihood function* $f(\mathbf{z}_n|\mathbf{x}_n)$ factorizes into the local likelihood functions, i.e.,

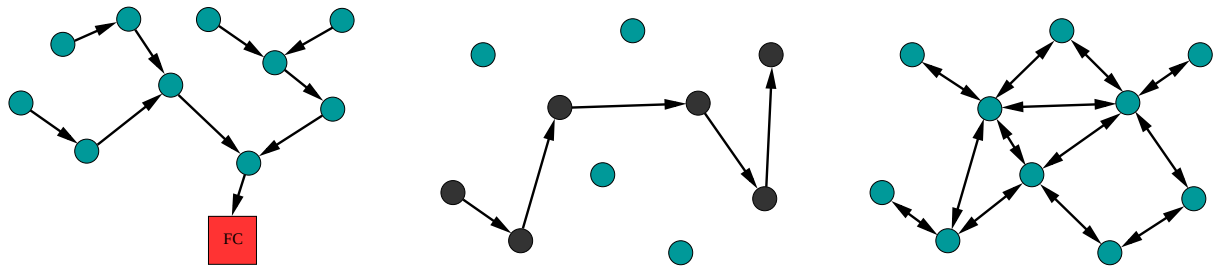
$$f(\mathbf{z}_n|\mathbf{x}_n) = \prod_{k=1}^{K} f(\mathbf{z}_{n,k}|\mathbf{x}_n)\,. \tag{9}$$

The goal of distributed sequential estimation is to estimate $\mathbf{x}_n$ in a sequential manner, based on the measurements $\mathbf{z}_{n',k}$ of all (or a relevant subset of) agents $k$ for all times $n'$ up to the current time $n$. Preferably, each agent should communicate only with neighboring agents, and the estimation results should be available at each agent, or at least at a relevant subset of agents. An important class of algorithms for distributed sequential Bayesian estimation is based on the KF or its modifications [23–29]. In the linear/Gaussian case, the "information form" of the KF—referred to as the information filter—is particularly amenable to a distributed implementation [23]. In [24] and [25], distributed information filters based on consensus algorithms are presented. A diffusion-based distributed KF is proposed in [26], and a distributed KF using gossip algorithms is developed in [27]. For nonlinear/non-Gaussian models, distributed versions of the extended KF or unscented KF are proposed in [23], [28], and [29]. A second large class of distributed algorithms for sequential Bayesian estimation is given by DPFs [30–66], which are addressed in the remainder of this article.

## 6 Distributed Particle Filtering

Because PFs tend to outperform KF-based methods for nonlinear/non-Gaussian models, DPFs are an attractive choice for distributed sequential Bayesian estimation in ANs. To obtain a distributed PF implementation, some approximations are typically required. Further approximations may be needed to reduce computation and communication requirements; their impact is studied in [67]. As a result of these approximations, DPFs usually do not perform as well as a centralized PF that has access to all the measurements. However, avoiding these approximations will usually imply excessive communication requirements, poor scalability, and practically inconvenient constraints such as the need for synchronized random generators.

The existing DPF algorithms differ in aspects such as type and amount of the data commu-




(a) (b) (c)

Figure 3: Inter-agent communications for three classes of DPFs. (a) In an FC-based DPF (here depicted using multihop communications), local information is transmitted or routed to the FC. (b) In an LA-based DPF, only a subset of agents (dark gray bullets) are activated in a sequential manner, with a corresponding sequential communication schedule. (c) In a consensus-based DPF, all agents are simultaneously active, and each one communicates with neighboring agents.

nicated between the agents, communication range, local processing, computational complexity, memory requirements, estimation accuracy, robustness, scalability, and latency. As shown in Figure 3, a basic distinction is between DPFs that employ a central processing unit—referred to as a fusion center (FC)—and those that operate in a decentralized manner. Although we review FC-based DPFs in the next section, our focus will be on decentralized DPFs.

Based on the type of the data communicated between the agents, we will discriminate two broad classes of decentralized DPFs: *statistics dissemination-based* DPFs, where processed data (representations of posteriors or likelihood functions) are exchanged between the agents [34–55,57,60–66], and *measurement dissemination-based* DPFs, where raw or quantized measurements are exchanged [56–59]. A subdivision of the statistics dissemination-based class can be based on the set of agents that perform particle filtering at any given (continuous) time, with corresponding differences regarding agent scheduling and communication topology. Accordingly, we will distinguish between *leader agent (LA)-based* DPFs, where at any given time only one agent is in charge of the processing [34–42], and *consensus-based* DPFs, where all the agents in the network process data simultaneously [43–55] (see Figure 3). A taxonomy of DPFs corresponding to these DPF classes plus certain subclasses is presented in Figure 4. We note that each of the DPF (sub)classes mentioned above can be further subdivided into DPFs where each agent calculates the posterior of the entire state vector $\mathbf{x}_n$ (the vast majority of DPFs presented in this article are of this type) and DPFs where each agent estimates only a subvector of $\mathbf{x}_n$ [64, 65] (this is advantageous for a high-dimensional state vector $\mathbf{x}_n$). However, this subclassification is not shown in Figure 4.

Our discussion of specific DPFs will be organized according to the classification depicted in Figure 4. FC-based DPFs are addressed in the next section, while the four subsequent sections

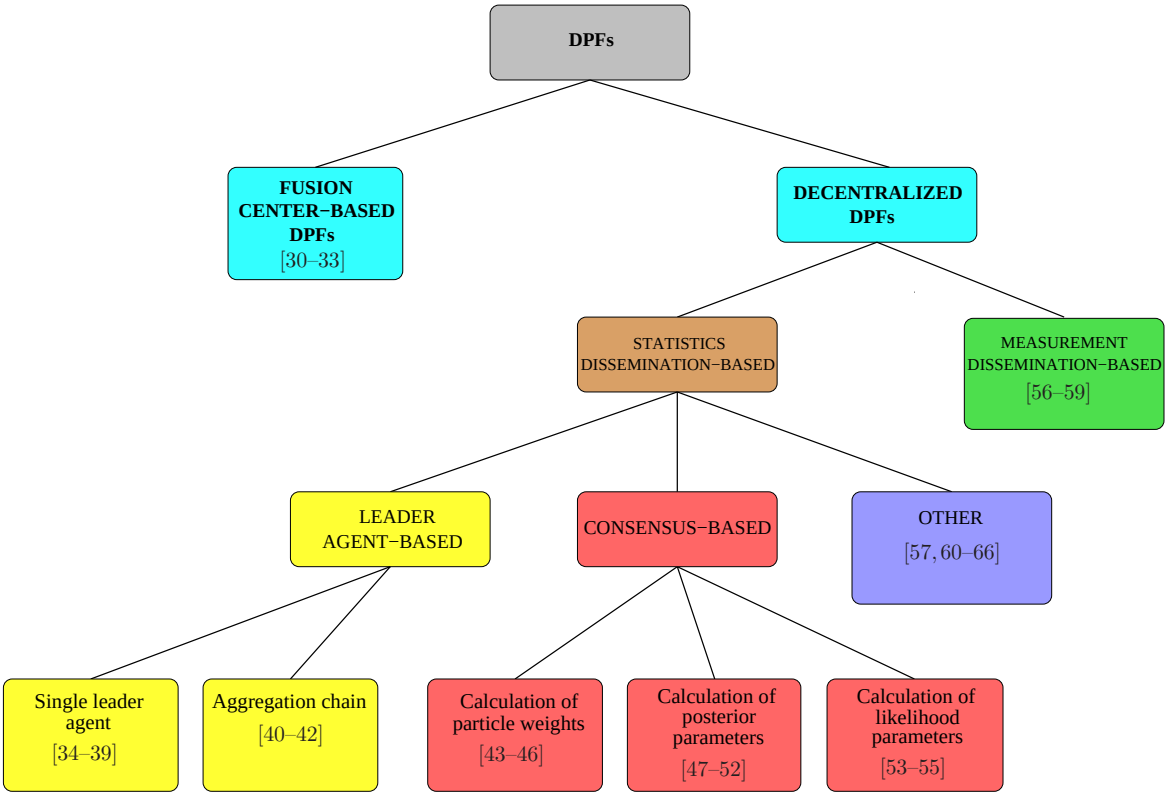



Figure 4: Taxonomy of DPFs.

are dedicated to four classes of decentralized DPFs: LA-based DPFs, consensus-based DPFs, other statistics dissemination-based DPFs, and measurement dissemination-based DPFs.

## 7 Fusion Center-based DPFs

In FC-based DPFs [30–33], each agent uses a local PF to convert its own measurement into a local posterior, which is then transmitted to an FC. The FC computes the final (global) posterior and the global state estimate. FC-based DPFs are especially interesting when the final estimate needs to be available only at a single central location, such as in monitoring applications. The communication requirements can be reduced by using approximate representations of the local posteriors transmitted from the agents to the FC, such as GM representations [30, 31] or histograms [32]. If direct communications from the agents to the FC are not feasible due to limited communication range, multihop communications can be used (see Figure 3(a)). Disadvantages of multihop communications are the necessity of rebuilding the routing tables each time the network topology changes (e.g, due to mobility of the agents) and an unevenly distributed energy consumption (since agents closer to the FC experience heavier traffic).

An extended form of FC-based DPFs uses a hierarchical communication topology in which the agents are organized into clusters and the cluster members communicate only with a privileged agent termed cluster head [33]. Thereby, intracluster communications are limited to short distances and power is saved. Each cluster head executes a local PF that converts the measurements received from the cluster members into a local posterior. The local posteriors are transmitted to the FC, where the global state estimate is calculated.

For completeness, we also mention PF methods for ANs in which raw or quantized measurements or innovations are transmitted from the agents to the FC, where they are processed by a centralized PF. Since the agents do not perform any particle filtering operations, these methods cannot be considered as DPFs. In the so-called channel-aware PFs [68], quantized local measurements are transmitted from the agents to the FC. The statistics of the channel imperfections (noise, fading) are taken into account by the PF algorithm executed at the FC. This results in an increased robustness to nonideal communication channels. The transmission of quantized measurements underlies also the PFs proposed in [69–71]. In [69], binary measurements are transmitted over a noisy channel, and an auxiliary PF or a cost-reference PF is used at the FC. In [71], a PF algorithm for processing quantized measurements is developed, and a scheme for agent selection is proposed. In order to reduce the communication requirements and energy consumption, only measurements of the most informative agents are transmitted to the FC. In [72] and [73], quantized innovations (i.e., differences between the true measurements and predicted measurements) are transmitted.

# 8 LA-based DPFs

The remainder of this article is dedicated to decentralized DPFs. Following the classification in Figure 4, we first consider the class of statistics dissemination-based DPFs, i.e., DPFs where processed data such as representations of posteriors or of likelihood functions are exchanged between the agents. We will directly discuss the three subclasses of the class of statistics dissemination-based DPFs, starting with the subclass of LA-based DPFs.

## 8.1 Basic Principles

In LA-based DPFs, information accumulates along a path formed by a sequence of adjacent agents. The agent that is currently active (the LA) performs particle filtering and stores a particle representation of the most up-to-date posterior.

One can distinguish two types of LA-based DPFs. In the single LA case [34–39], there is only a single LA during the continuous-time interval corresponding to a given discrete time step $n$, as depicted in Figure 5(a). This LA performs one PF recursion using its own measurement at time $n$

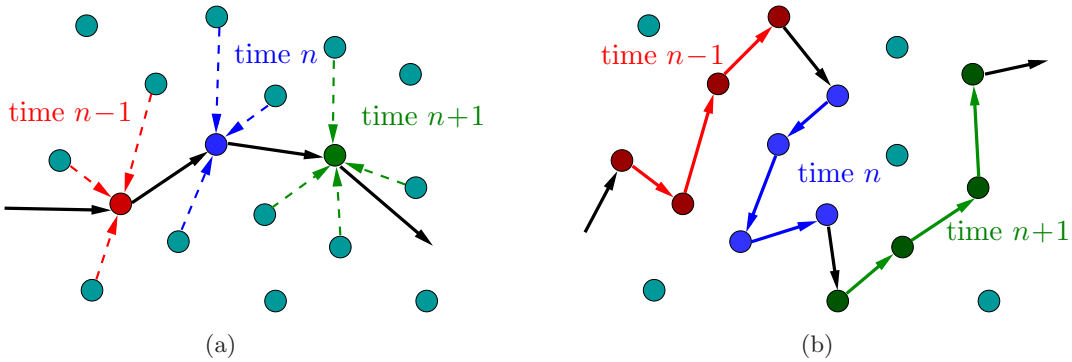



Figure 5: Two types of LA-based DPFs. (a) Single LA case: only one LA is active at each discrete time step $n$. Black arrows indicate transmissions of posteriors between consecutive LAs. Dotted colored arrows indicate transmissions of measurements from neighboring agents to the current LA, with different colors indicating different time steps. (b) AC case: multiple LAs (constituting an AC) are activated sequentially during the time interval corresponding to a discrete time step $n$. Colored arrows indicate transmissions of partial posteriors between consecutive LAs of an AC, with different colors indicating different ACs (or time steps). Black arrows indicate transmissions of posteriors from the last LA of a given AC to the first LA of the next AC.

and, possibly, the measurements of neighboring agents at time $n$. The LA then selects a new LA for time step $n+1$, which performs again one PF recursion using the measurements at time $n+1$, and so on. Unless the current LA selects itself as the new LA, the posterior is transmitted from the current LA to the new LA. Thus, since the posterior involves also past measurements, each LA performs its PF recursion based on all the information accumulated so far.

In the aggregation chain (AC) case [40–42] depicted in Figure 5(b), multiple LAs constituting an AC are activated sequentially during the time interval corresponding to a discrete time step $n$. They process only their own local measurements observed at time $n$, not any measurements from neighboring agents. A given LA of the $n$th AC receives from the previous LA a *partial posterior*, selects the next LA, and transmits to it an improved partial posterior that incorporates the own local measurement. The last LA of the $n$th AC thus obtains a posterior that reflects the measurements of all the LAs of the $n$th AC (as well as the measurements of all previous ACs). The last LA then selects the first LA of the $(n+1)$th AC and transmits to it its posterior, and the whole process is repeated. Since no measurements are actually transmitted, this scheme is particularly suited to settings with high-dimensional measurements (e.g., in camera networks). The two types of single LA DPFs and AC DPFs will be considered in more detail in later subsections.

In certain applications, most notably the tracking of multiple targets, multiple instances of an LA-based DPF can be performed in one AN: for each target, one LA-based DPF is executed by

agents located close to the target. However, if two or more targets are close to each other, there may also be only one LA-based DPF tracking more than one target.

In both the single LA case and the AC case, the current LA selects the next LA from among its neighbors, using a local on-line scheduling algorithm. Thus, the LA path through the AN is constructed dynamically and adaptively, without any central control. The selection of the LAs has a significant impact on the estimation performance and energy costs [74]. Ideally, the new LA should be the neighboring agent with the most informative measurement (possibly considering also the measurements of the neighbors of the candidate agents). However, to avoid transmissions from candidate agents, the current LA typically makes its selection without knowing the measurements of the candidate agents, based only on its prior knowledge about the candidate agents, e.g., their positions or measurement models, and its knowledge of the current posterior, which summarizes all the information about the state available so far. For the selection of the new LA based on this limited knowledge, various optimality criteria and metrics have been proposed, including entropy [38], mutual information [75], the Cramér-Rao lower bound [76, 77], and the predicted estimation error [78]. Furthermore, numerically efficient Monte Carlo computation methods for sensor selection have been described [38, 76, 77].

The handover of the (partial) posterior from one LA to the next can be a communication-intensive task, especially for a high-dimensional state vector $\mathbf{x}_n$. The communication requirements can be reduced by transmitting an approximate representation of the (partial) posterior. The choice of such a representation affects the kind of processing performed by the LAs as well as the estimation performance. Clearly, there is a tradeoff between estimation performance and savings in communication. A straightforward approach is to transmit the particles and weights representing the (partial) posterior [34]. However, this tends to result in prohibitively high communication requirements, even if an adapted proposal pdf is used to get by with a smaller number of particles and weights. Lower communication requirements are obtained by transmitting only a subset of particles with the largest weights, or by performing a more sophisticated compression of the particle representation [35]. An alternative approach, which typically results in significantly smaller communication requirements, is to transmit the parameters of a GM approximation of the (partial) posterior [38, 40, 41]. These parameters can be derived from a particle representation of the (partial) posterior by means of the expectation-maximization (EM) algorithm [79] or $k$-means clustering [41]. At the receiving agent, sampling from the GM yields a particle representation that is approximately equivalent to the particle representation at the transmitting agent.

### 8.2 Single LA DPFs

We will now provide more detailed algorithmic descriptions of LA-based DPFs. A step-by-step statement of the single LA DPF algorithm is as follows:

1. At the beginning of (the continuous-time interval corresponding to) time step $n$, the current LA $k$ receives from the previous LA—which was active at time $(n-1)$—a representation of the previous posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$, as well as measurements $\mathbf{z}_{n,k'}$ from a subset of agents $k'$ (e.g., $k' \in \mathcal{N}_k$). If the received representation of $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$ is not already a particle representation $\left\{\left(\mathbf{x}_{n-1}^{(j)}, w_{n-1}^{(j)}\right)\right\}_{j=1}^{J}$ (e.g., if it is a GM representation), the LA randomly draws particles $\mathbf{x}_{n-1}^{(j)}$ from the received representation and sets their weights to $w_{n-1}^{(j)} = 1/J$.

2. Using the particles $\mathbf{x}_{n-1}^{(j)}$ and weights $w_{n-1}^{(j)}$, the measurements $\mathbf{z}_{n,k'}$ received from the other agents, and its own measurement $\mathbf{z}_{n,k}$, the LA performs a PF recursion to calculate new particles $\mathbf{x}_n^{(j)}$ and their weights $w_n^{(j)}$. These represent the new posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$. The PF recursion is identical to Steps 1, 2, and 4 of "Generic Particle Filter (SIR Filter) Algorithm," except that the likelihood function used in Step 2, $f(\mathbf{z}_n|\mathbf{x}_n)$, is replaced by $f\left(\{\mathbf{z}_{n,k'}\}_{k'\in\mathcal{N}_k}, \mathbf{z}_{n,k} \middle| \mathbf{x}_n\right)$.

3. The LA selects the next LA, which is going to be active at time step $n+1$.

4. The LA transmits to the next LA a representation of the new posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$. This representation is either directly given by the particles $\mathbf{x}_n^{(j)}$ and weights $w_n^{(j)}$ (possibly compressed), or it is constructed from them (e.g., a GM representation).

Single LA DPFs are proposed in [34–39]. The transmission of an uncompressed particle representation of the posterior is considered in [34], while a lossless compression of the particle representation is proposed in [35] and [36]. A lossy compression scheme is proposed and analyzed in [35–37]. This scheme produces a GM approximation of the posterior; the tradeoff between estimation performance and savings in communication can be controlled adaptively by adjusting the number of Gaussian components. A theoretical analysis of the impact of posterior compression on the performance of single LA DPFs is provided in [67]. In the DPFs presented in [35–38], apart from the posterior compression, the PF operations performed by the LAs equal those of a centralized PF. On the other hand, in [39], assuming a target tracking scenario, the PF operations performed by the LAs are modified to allow for joint tracking and classification of multiple targets.

### 8.3 AC DPFs

In the following algorithmic statement of AC DPFs, the LAs constituting the AC at time step $n$ are indexed using an index-mapping function $k(i)$, which maps a local index $i \in \{1, \ldots, K_n\}$

(indicating the position of a given agent within the AC) to the global index $k$ (which is unique for each agent in the entire network). Here, $K_n$ denotes the number of agents constituting the $n$th AC. Furthermore, we denote by $k(1\colon i_0)$ the set of LAs of the AC with local indices $i \in \{1, 2, \ldots, i_0\}$.

Each LA within the $n$th AC performs the following steps, with certain differences applying to the first LA, $k(1)$, and the last LA, $k(K_n)$, as noted.

1. LA $k(i)$ receives a representation of the partial posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n-1}, \mathbf{z}_{n,k(1:i-1)})$ from the previous LA $k(i-1)$. Here, $\mathbf{z}_{n,k(1:i-1)}$ denotes the current measurements (at time $n$) of LAs $k(1), k(2), \ldots, k(i-1)$. Thus, the condition $(\mathbf{z}_{1:n-1}, \mathbf{z}_{n,k(1:i-1)})$ of $f(\mathbf{x}_n|\mathbf{z}_{1:n-1}, \mathbf{z}_{n,k(1:i-1)})$ comprises the measurements of all LAs that have been active so far: those of the LAs of all previous ACs (i.e., $\mathbf{z}_{1:n-1}$) and those of the LAs of the current AC up to LA $k(i-1)$ (i.e., $\mathbf{z}_{n,k(1:i-1)}$). If the received representation of $f(\mathbf{x}_n|\mathbf{z}_{1:n-1}, \mathbf{z}_{n,k(1:i-1)})$ is not already a particle representation $\big\{\big(\mathbf{x}_n^{(j)}, w_{n,k(i-1)}^{(j)}\big)\big\}_{j=1}^{J}$, LA $k(i)$ randomly draws particles $\mathbf{x}_n^{(j)}$ from the received representation and sets their weights to $w_{n,k(i-1)}^{(j)} = 1/J$. The first LA, $k(1)$, is different in that it receives a representation of the predicted posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n-1})$ from the last LA of the previous ($(n-1)$th) AC.

2. Using the particles $\mathbf{x}_n^{(j)}$, the weights $w_{n,k(i-1)}^{(j)}$, and its own measurement $\mathbf{z}_{n,k(i)}$, LA $k(i)$ performs a partial update step to calculate local weights $w_{n,k(i)}^{(j)} \propto f(\mathbf{z}_{n,k(i)}|\mathbf{x}_n^{(j)})$ (cf. Step 2 of "Generic Particle Filter (SIR Filter) Algorithm"). The particles $\mathbf{x}_n^{(j)}$ together with the weights $w_{n,k(i)}^{(j)}$ represent the new partial posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n-1}, \mathbf{z}_{n,k(1:i)})$. The last LA, $k(K_n)$, is different in that it performs a prediction step in addition to the partial update step. This produces predicted particles $\mathbf{x}_{n+1}^{(j)}$ representing the predicted posterior $f(\mathbf{x}_{n+1}|\mathbf{z}_{1:n})$. The predicted particles are sampled from a (possibly adapted) proposal pdf (cf. Step 1 of "Generic Particle Filter (SIR Filter) Algorithm").

3. LA $k(i)$ selects the next LA, $k(i+1)$. The last LA, $k(K_n)$, is different in that it selects the first LA of the $(n+1)$th AC. (Note that the first LA of the $(n+1)$th AC may be identical to the last LA of the $n$th AC.)

4. LA $k(i)$ transmits to the next LA $k(i+1)$ a representation of the new partial posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n-1}, \mathbf{z}_{n,k(1:i)})$, which is given by the particles $\mathbf{x}_n^{(j)}$ and weights $w_{n,k(i)}^{(j)}$ (possibly compressed) or is constructed from them (e.g., a GM representation). The last LA, $k(K_n)$, is different in that it transmits to the first LA of the $(n+1)$th AC a representation of the predicted posterior $f(\mathbf{x}_{n+1}|\mathbf{z}_{1:n})$. This transmission can be avoided if the first LA of the $(n+1)$th AC is identical to the last LA of the $n$th AC.

Thus, whereas in single LA DPFs each LA performs both the update and prediction steps of a PF recursion, in AC DPFs the LAs inside an AC perform only the update step, which serves to incorporate their local measurements into the partial posterior; only the last LA of the AC performs also the prediction step. In fact, the operation of an entire AC can be viewed as one PF recursion, with the update step implemented in a sequential manner by the LAs of the AC.

AC DPFs are proposed in [40–42]. In these schemes, the partial posteriors are transmitted between successive LAs via GM approximations, which are derived from the LA-internal particle representations by means of the EM algorithm [40, 42] or $k$-means clustering [41].

## 9 Consensus-based DPFs

With consensus-based DPFs, all agents perform particle filtering simultaneously and possess a particle representation of a posterior. Ideally, this posterior is the global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ reflecting the current and past measurements of all agents, $\mathbf{z}_{1:n}$. This goal is achieved or, at least, approached by a decentralized consensus algorithm that establishes an agreement on certain global quantities across all agents. These quantities are then used by each agent to establish a local approximate particle representation of the global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$. A short summary of consensus algorithms is given in "Consensus Algorithms." The use of consensus algorithms implies that each agent transmits certain quantities to a set of neighboring agents. This is different from the sequential communication structure used by LA-based DPFs, as visualized in Figure 3.

---

Consensus Algorithms

---

Consensus algorithms are effective tools for distributed computations [80, 81]. A special type of consensus algorithms is constituted by gossip algorithms [82, 83]. In the context of ANs, "consensus" means a global agreement on some quantity that depends on the data of all agents, and a "consensus algorithm" specifies the corresponding information exchange between neighboring agents and the computations performed by each agent. Consensus algorithms are iterative schemes that diffuse information through the network and, usually, reach a global agreement only asymptotically. The differences between the values at the individual agents after a finite number of iterations depends on the number of iterations, the size and topology of the network, and the particular consensus algorithm. The convergence of consensus algorithms is studied in [84] and, in the presence of random interagent link failures and data quantization, in [85]. Consensus algorithms are advantageous in that there is no bottleneck or single point of failure, and the algorithms are robust to changing network topologies and unreliable network conditions such as link failures.

Three consensus algorithms frequently used in DPFs are the *average consensus*, the *randomized gossip*, and the *max-consensus*. Consider a connected AN where each agent $k \in \{1, \ldots, K\}$ possesses a scalar quantity $s_k$. The goal is a distributed calculation of either the average $\frac{1}{K}\sum_{k=1}^{K} s_k$ or the maximum $\max_{k\in\{1,\ldots,K\}}\{s_k\}$, such that the result is available at each agent. In the three types of consensus algorithms, at each iteration $i$, each agent updates an "internal state" $\zeta_k$ using the internal states of a set of neighboring agents. The operations performed by agent $k$ are given by the following unified description:

1. Initialize the internal state as $\zeta_k^{(0)} = s_k$.
2. For $i = 1, 2, \ldots$, update the internal state according to $\zeta_k^{(i)} = u\big(\zeta_k^{(i-1)}, \{\zeta_{k'}^{(i-1)}\}_{k' \in \mathcal{N}_k}\big)$, where $u(\cdot)$ is a state-update function that combines the previous internal state $\zeta_k^{(i-1)}$ and the previous internal states of all neighboring agents $\{\zeta_{k'}^{(i-1)}\}_{k' \in \mathcal{N}_k}$. Then, broadcast $\zeta_k^{(i)}$ to all neighbors $k' \in \mathcal{N}_k$.

The state-update function $u(\cdot)$ depends on the type of consensus algorithm:

- *Average consensus*: $u\big(\zeta_k^{(i)}, \{\zeta_{k'}^{(i)}\}_{k' \in \mathcal{N}_k}\big) = \omega_{k,k}^{(i)} \zeta_k^{(i)} + \sum_{k' \in \mathcal{N}_k} \omega_{k,k'}^{(i)} \zeta_{k'}^{(i)}$, where the $\omega_{k,k'}^{(i)}$ are weights whose choice is discussed, e.g., in [80], [81], and [86]. Thus, the new internal state is a linear combination of the previous internal state and the previous internal states of the neighbors. For $i \to \infty$, each internal state $\zeta_k^{(i)}$ converges to the average of the $s_k$ over all agents $k$.
- *Randomized gossip*: $u\big(\zeta_k^{(i)}, \{\zeta_{k'}^{(i)}\}_{k' \in \mathcal{N}_k}\big) = \frac{1}{2}\zeta_k^{(i)} + \frac{1}{2}\zeta_{\tilde{k}}^{(i)}$, where $\tilde{k} \in \mathcal{N}_k$ is a randomly selected neighbor of agent $k$. Again, the internal states $\zeta_k^{(i)}$ converge asymptotically to the average of the $s_k$ over all agents $k$.
- *Max-consensus*: $u\big(\zeta_k^{(i)}, \{\zeta_{k'}^{(i)}\}_{k' \in \mathcal{N}_k}\big) = \max\big\{\zeta_k^{(i)}, \{\zeta_{k'}^{(i)}\}_{k' \in \mathcal{N}_k}\big\}$. Here, each internal state $\zeta_k^{(i)}$ becomes the maximum of all $s_k$ after a finite number of iterations that does not exceed the diameter of the communication graph [44]. The *min-consensus* algorithm is analogous, with obvious modifications.

---

The use of consensus algorithms in DPFs has become popular in recent years, due to the following advantages. Consensus algorithms require only local communications between neighboring agents. They do not require routing protocols or global knowledge about the network. Each agent is in possession of the global estimate (this is important in sensor/actuator and robotic ANs where the agents perform some action based on the obtained estimate). Consensus-based DPFs are robust to changes in the network topology and to link failures, and thus suitable for networks with mobile agents. They are also robust to agent failures, since the posterior is available at each agent. On the other hand, a disadvantage is the (generally) higher communication requirements of consensus-based DPFs compared to LA-based DPFs. Furthermore, the number of consensus iterations needed to diffuse local information through the AN increases with the size of the AN. This may be a problem in real-time applications.

The various consensus-based DPFs differ in the nature of the quantities computed via consensus algorithms. We consider the following types of quantities in the next three subsections: particle weights, parameters of the posterior, and parameters of the global likelihood function.

### 9.1 Consensus-based Calculation of Particle Weights

The DPFs proposed in [43–46] perform a distributed computation of global particle weights $w_n^{(j)}$ (reflecting the measurements of all agents) from local weights $w_{n,k}^{(j)}$ (each reflecting only the measurement of the respective agent $k$). For each weight $w_n^{(j)}$, $j \in \{1, \ldots, J\}$, one consensus algorithm for computing an average is executed. This approach presupposes that identical sets of particles

$\{\mathbf{x}_n^{(j)}\}_{j=1}^J$ are sampled at each agent. This, in turn, requires that the local random number generators at the agents are synchronized—such that the same pseudo-random numbers are obtained in the entire network—and that identical particle representations $\{(\mathbf{x}_{n-1}^{(j)}, w_{n-1}^{(j)})\}_{j=1}^J$ of the previous global posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$ are available at each agent.

The consensus-based calculation of the global weights is based on the factorization (9) of the global likelihood function $f(\mathbf{z}_n|\mathbf{x}_n)$. Using (9) in (7), it is seen that the $j$th global weight $w_n^{(j)}$ is proportional to the product of the local likelihood functions evaluated at $\mathbf{x}_n^{(j)}$:

$$w_n^{(j)} \propto \tilde{w}_n^{(j)} = f(\mathbf{z}_n|\mathbf{x}_n^{(j)}) = \prod_{k=1}^{K} f(\mathbf{z}_{n,k}|\mathbf{x}_n^{(j)}), \qquad j \in \{1,\ldots,J\}. \tag{10}$$

Here, we disregarded the factors $f(\mathbf{x}_n^{(j)}|\mathbf{x}_{n-1}^{(j)})$, $q(\mathbf{x}_n^{(j)}|\mathbf{x}_{n-1}^{(j)}, \mathbf{z}_n)$, and $w_{n-1}^{(j)}$ occurring in (7) because they are locally available at each agent as explained in [44] (in particular, $q(\mathbf{x}_n^{(j)}|\mathbf{x}_{n-1}^{(j)}, \mathbf{z}_n)$ can be obtained using a distributed proposal adaptation scheme). Taking the logarithm of (10) yields

$$\log(\tilde{w}_n^{(j)}) = \sum_{k=1}^{K} \log f(\mathbf{z}_{n,k}|\mathbf{x}_n^{(j)}), \qquad j \in \{1,\ldots,J\}. \tag{11}$$

This sum over all agents $k$ can be computed in a distributed way by means of a consensus algorithm. Each agent locally computes $\log f(\mathbf{z}_{n,k}|\mathbf{x}_n^{(j)})$ (by assumption, $\mathbf{x}_n^{(j)}$ is identical at all agents). Then, for each particle $\mathbf{x}_n^{(j)}$, $j \in \{1,\ldots,J\}$, one consensus algorithm is executed to calculate an approximation $\tilde{A}(\mathbf{z}_n, \mathbf{x}_n^{(j)})$ of the average $A(\mathbf{z}_n, \mathbf{x}_n^{(j)}) \triangleq \frac{1}{K}\sum_{k=1}^{K} \log f(\mathbf{z}_{n,k}|\mathbf{x}_n^{(j)})$. (Since a finite number of consensus iterations is used, only an approximation of $A(\mathbf{z}_n, \mathbf{x}_n^{(j)})$ is obtained.) Due to (11), we have $KA(\mathbf{z}_n, \mathbf{x}_n^{(j)}) = \log(\tilde{w}_n^{(j)})$ and therefore also $K\tilde{A}(\mathbf{z}_n, \mathbf{x}_n^{(j)}) \approx \log(\tilde{w}_n^{(j)})$ and further

$$\exp\big(K\,\tilde{A}(\mathbf{z}_n, \mathbf{x}_n^{(j)})\big) \approx \tilde{w}_n^{(j)}, \qquad j \in \{1,\ldots,J\}.$$

Such an approximation of $\tilde{w}_n^{(j)}$ is obtained at each agent. (Note that $K$ has to be known to each agent. This information may be provided beforehand, or a distributed algorithm for counting the number of agents may be used [87].) Next, approximations of the global weights $w_n^{(j)}$ are derived from the above approximations of the $\tilde{w}_n^{(j)}$ by normalization, based on the relation $w_n^{(j)} = \tilde{w}_n^{(j)} / \sum_{j'=1}^{J} \tilde{w}_n^{(j')}$. The resulting approximate global weights will differ slightly at different agents because only a finite number of consensus iterations has been performed. To obtain identical weights, max-consensus (or min-consensus) algorithms can be used across the agents; this produces at each agent the maximum (or minimum) of the weights of all agents. One max-consensus is used for each $w_n^{(j)}$, $j \in \{1,\ldots,J\}$. Having identical sets of weights at all agents ensures that identical particles will be sampled.

If errors caused by the consensus algorithms are neglected, this DPF scheme achieves the same estimation performance as a centralized PF. Furthermore, no parametric approximation of the

posterior is required. On the other hand, the communication requirements of these "nonparametric" DPFs are quite high, since for each particle one average-computing consensus algorithm and one max-consensus algorithm are executed at each time step $n$, and the number of particles can be as high as several thousand. To reduce the number of particles, and hence the communication requirements, the DPF presented in [44] uses an adapted proposal pdf, which is computed by a distributed scheme based on set intersections and implemented via max- and min-consensus algorithms. In [43], a distributed auxiliary PF is proposed. For distributed weight calculation, a modification of randomized gossip known as selective gossip [88] is used. Here, the communication requirements are reduced by transmitting only information about the largest weights. In [45], two algorithms for distributed weight computation—broadcast gossip [82] and belief propagation [89]—are investigated and compared. In [46], to ensure identical sets of weights at all agents, a quantization scheme combined with a min-consensus strategy is introduced as an alternative to using max-consensus algorithms.

### 9.2 Consensus-based Calculation of Posterior Parameters

In the DPFs proposed in [47–52], the global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ is approximated by a Gaussian or GM pdf. A similar parametric approximation is used for the local posteriors $f(\mathbf{x}_n|\mathbf{z}_{1:n-1}, \mathbf{z}_{n,k})$ incorporating the past measurements of all agents, $\mathbf{z}_{1:n-1}$, and the current local measurement of agent $k$, $\mathbf{z}_{n,k}$. The parameters of the global posterior are calculated from the parameters of the local posteriors in a distributed manner using consensus algorithms.

In [47] and [50–52], Gaussian representations are used. Each agent first calculates a particle representation $\left\{\left(\mathbf{x}_{n,k}^{(j)}, w_{n,k}^{(j)}\right)\right\}_{j=1}^{J}$ of the local posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n-1}, \mathbf{z}_{n,k}) \propto f(\mathbf{z}_{n,k}|\mathbf{x}_n) f(\mathbf{x}_n|\mathbf{z}_{1:n-1})$ via a PF prediction step using a local proposal pdf $q(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_{n,k})$ and a local PF update step involving the local likelihood function $f(\mathbf{z}_{n,k}|\mathbf{x}_n)$. Then, a Gaussian approximation $\mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_{n,k}, \mathbf{C}_{n,k})$ of the local posterior is calculated from the particle representation $\left\{\left(\mathbf{x}_{n,k}^{(j)}, w_{n,k}^{(j)}\right)\right\}_{j=1}^{J}$ according to

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

Finally, the local parameters $\boldsymbol{\mu}_{n,k}$ and $\mathbf{C}_{n,k}$ for $k \in \{1, \ldots, K\}$ are fused into the parameters of the Gaussian approximation $\mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_n, \mathbf{C}_n)$ of the global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ by means of a consensus-based fusion rule, and the MMSE state estimate is obtained at agent $k$ as $\hat{\mathbf{x}}_{n,k} = \boldsymbol{\mu}_n$. The DPF proposed in [47] is a distributed SIR filter that uses the state-transition pdf $f(\mathbf{x}_n|\mathbf{x}_{n-1})$ as the local proposal pdf. The global posterior parameters are calculated as the averages of the local parameters; this simple fusion rule is suboptimal but allows a straightforward application of the

average consensus algorithm. The DPF proposed in [50] is a distributed auxiliary PF that employs a fusion rule based on the product of Gaussians [90]. This rule requires a "predistortion" of each local posterior using the power $f^K(\mathbf{z}_{n,k}|\mathbf{x}_n)$ instead of the local likelihood function $f(\mathbf{z}_{n,k}|\mathbf{x}_n)$ in the local PF update step. The fusion rule employed by the DPF described in [52] is based on a rule previously proposed in [91]. The local proposal pdf is obtained at each agent by means of an unscented KF. The DPF in [51] uses a fusion rule that is based on the rule proposed in [80]. Only a single consensus iteration is executed at each time $n$; this reduces the communications and latency at the cost of a potential performance loss due to the limited diffusion of local information.

In [48] and [49], GM representations are used. In [48], an EM algorithm is executed at each agent to calculate the local GM parameters. The global GM parameters are then obtained as averages of the local GM parameters; this simple but suboptimal fusion rule (cf. [47]) is implemented by average consensus algorithms. In [49], a distributed EM scheme using consensus algorithms is proposed for deriving the global GM parameters directly from the local particle representations.

### 9.3 Consensus-based Calculation of Likelihood Parameters

The DPFs presented in the previous two subsections employ consensus algorithms to fuse approximate local posteriors (described either by particle representations or by parametric representations) into an approximate global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$. We now discuss DPFs in which consensus algorithms are used to compute the global likelihood function (GLF) $f(\mathbf{z}_n|\mathbf{x}_n)$ rather than the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ [53–55]. The GLF—or, at least, an approximation—is obtained at each agent as a function of $\mathbf{x}_n$. This allows each agent to evaluate the GLF at the particles $\mathbf{x}_n^{(j)}$, which is required for calculating the weights $w_n^{(j)}$ in the PF update step (7). Therefore, each agent is able to locally run a PF that is equivalent to a global PF because it uses the GLF and calculates a global estimate involving the all-agents measurement vector $\mathbf{z}_n$. The local PFs operate independently of each other (only the GLF is computed cooperatively), so that a synchronization of random number generators is not required. The global estimates obtained at the individual agents may differ slightly due to the nonsynchronized local random generators and errors caused by insufficiently converged consensus algorithms.

A consensus-based computation of the GLF is always possible if there exists a sufficient statistic with a special structure [53]. In fact, for any sufficient statistic $\mathbf{t}_n(\mathbf{z}_n)$, the Neyman-Fisher factorization theorem [17] states that the GLF can be written as $f(\mathbf{z}_n|\mathbf{x}_n) = f_1(\mathbf{z}_n)\, f_2\big(\mathbf{t}_n(\mathbf{z}_n), \mathbf{x}_n\big)$. Hence, $\mathbf{t}_n(\mathbf{z}_n)$ summarizes the measurement $\mathbf{z}_n$ in that an agent that knows $\mathbf{t}_n(\mathbf{z}_n)$ and the function $f_2(\cdot,\cdot)$ is able to evaluate the GLF $f(\mathbf{z}_n|\mathbf{x}_n)$ for any value of $\mathbf{x}_n$, up to an irrelevant factor $f_1(\mathbf{z}_n)$.

Suppose now that $\mathbf{t}_n(\mathbf{z}_n)$ has the following additive structure:

$$\mathbf{t}_n(\mathbf{z}_n) \;=\; \sum_{k=1}^{K} \boldsymbol{\eta}_{n,k}(\mathbf{z}_{n,k})\,, \tag{12}$$

with arbitrary functions $\boldsymbol{\eta}_{n,k}(\cdot)$, and that agent $k$ knows its own function $\boldsymbol{\eta}_{n,k}(\cdot)$ but not the functions $\boldsymbol{\eta}_{n,k'}(\cdot)$, $k' \neq k$ associated with the other agents. Based on expression (12), consensus algorithms can be used for a distributed calculation of $\mathbf{t}_n(\mathbf{z}_n)$ and, thus, of the GLF $f(\mathbf{z}_n|\mathbf{x}_n)$.

If the GLF does not have the structure considered above, a consensus-based approximate calculation of the GLF is enabled by the *likelihood consensus* (LC) proposed in [53] and [54]. The LC is based on the GLF factorization (9), which presupposes that the measurement noises $\mathbf{v}_{n,k}$ at different agents (cf. (8)) are statistically independent. Taking the logarithm of (9) yields

$$\log f(\mathbf{z}_n|\mathbf{x}_n) \;=\; \sum_{k=1}^{K} \log f(\mathbf{z}_{n,k}|\mathbf{x}_n)\,. \tag{13}$$

A direct consensus-based calculation of this sum is not possible in general because the local log-likelihood functions $\log f(\mathbf{z}_{n,k}|\mathbf{x}_n)$ depend on the unknown state $\mathbf{x}_n$. The LC therefore uses the following approximate (finite-order) basis expansions:

$$\log f(\mathbf{z}_{n,k}|\mathbf{x}_n) \;\approx\; \sum_{r=1}^{R} \alpha_{n,k,r}(\mathbf{z}_{n,k})\,\varphi_{n,r}(\mathbf{x}_n)\,, \qquad k = 1, \ldots, K\,. \tag{14}$$

Here, the expansion coefficients $\alpha_{n,k,r}(\mathbf{z}_{n,k})$ contain all agent-local information (including the local measurement $\mathbf{z}_{n,k}$); the $\varphi_{n,r}(\mathbf{x}_n)$ are fixed, agent-independent basis functions that are assumed to be known to all agents; and $R$ is the order of the basis expansion. Inserting (14) into (13) and exponentiating yields the following approximation of the GLF:

$$f(\mathbf{z}_n|\mathbf{x}_n) \;\approx\; \exp\!\left(\sum_{r=1}^{R} a_{n,r}(\mathbf{z}_n)\,\varphi_{n,r}(\mathbf{x}_n)\right), \qquad \text{with } \; a_{n,r}(\mathbf{z}_n) \;\triangleq\; \sum_{k=1}^{K} \alpha_{n,k,r}(\mathbf{z}_{n,k})\,.$$

Thus, an agent that knows the coefficients $a_{n,r}(\mathbf{z}_n)$ is able to evaluate an approximation of the GLF $f(\mathbf{z}_n|\mathbf{x}_n)$ for all values of $\mathbf{x}_n$. Furthermore, each $a_{n,r}(\mathbf{z}_n)$ is given by a sum over all agents, in which the term $\alpha_{n,k,r}(\mathbf{z}_{n,k})$ associated with the $k$th agent depends only on the local measurement $\mathbf{z}_{n,k}$. Thus, the $a_{n,r}(\mathbf{z}_n)$ can be computed in a distributed manner via consensus algorithms. Specializations of the LC to local likelihood functions belonging to the exponential family of distributions and to additive Gaussian measurement noise are described in [53].

The LC is used in [53] to implement distributed versions of the SIR filter and the Gaussian PF. A reduction of communication requirements and latency can be achieved by the use of a modified consensus algorithm with a single iteration per time step $n$ [55]. An extension of LC-based DPFs with a consensus-based distributed proposal adaptation is presented in [54].

## 10 Other Statistics Dissemination-based DPFs

Some statistics dissemination-based DPF algorithms [57, 60–66] do not fit into the classes of LA-based and consensus-based DPFs. In the DPFs presented in [60] and [61], the agents act as particles in that their positions represent candidate positions of a tracked target. Propagation of the particles is performed through activation of appropriate neighboring agents. In the DPFs proposed in [62] and [63], local posteriors computed at the individual agents are combined using channel filters. These filters track and remove the common past information—resulting from the common driving noise in the system model and from previous communications between neighboring agents—from the transmitted local posteriors so that only new information is updated at each agent. In these DPFs, it is not guaranteed that the local information is diffused through the entire network. Furthermore, a tree-structured communication topology is required. In [64] and [65], DPFs particularly suited to high-dimensional state vectors $\mathbf{x}_n$ are presented. Each agent estimates only a subvector of $\mathbf{x}_n$, which results in local PFs with reduced complexity. The information communicated between neighboring agents is given by particles and weights together with auxiliary information [64] or by local estimates and corresponding covariance matrices [65].

In [57], a DPF based on an approximate parametric representation of the GLF is proposed. The GLF is assumed to factorize as given in (9). The parametric GLF representation is constructed sequentially along a path that traverses the entire network from agent 1 to agent $K$. Agent $k$ receives from the previous agent $k-1$ a parametric approximation of the product of the local likelihood functions of agents 1 through $k-1$. Next, agent $k$ calculates a new parametric approximation incorporating its own local likelihood function as described in [57] and transmits it to agent $k+1$, which incorporates its own local likelihood function, and so on. Eventually, the last agent in the path obtains a parametric approximation of the GLF. The corresponding parameters are then transmitted back to all agents, where they are used for the update steps of local PFs. Thus, each agent obtains a global posterior and, consequently, a global estimate. Note that this is a difference from the LA-based DPFs discussed earlier.

In the DPF proposed in [66], each agent broadcasts parametric approximations of local likelihood functions to all other agents and then, based on the received parameters, computes an approximation of the global posterior using a local PF. Unlike in the DPFs presented so far, the variances of the local measurement noises do not need to be known a priori. In addition to estimating the state vector $\mathbf{x}_n$, the agents also estimate their local measurement models.

## 11 Measurement Dissemination-based DPFs

Measurement dissemination-based DPFs [56–59] are an alternative to the statistics dissemination-based DPFs presented in the previous sections. In these DPFs, each agent transmits to its neighbors its own measurement and measurements previously received from other agents. This is repeated until the measurements are disseminated through the entire network or a relevant part thereof. Each agent then acts like an FC in that it processes all the available measurements using a local PF. To reduce the communication requirements, the measurements are either quantized [57] (possibly using only a single bit [59]) or only the most relevant measurements are transmitted [56]. Because the communication requirements tend to grow with the dimension of the measurements, these DPFs are best suited to scenarios with low-dimensional measurements.

In the DPF proposed in [57], quantized measurements are transmitted from each agent to all other agents along a single communication chain that contains all the agents. The measurements are quantized locally using an adaptive quantizer that is trained using predicted particles and locally generated predictions of the measurements of all agents. Since all local random generators are synchronized, all agents generate the same predicted particles and predicted measurements. Hence, the local quantizers at each agent are identical and there is no need to transmit the quantization levels. In [59], two DPF schemes using agents equipped with directional sensors are proposed for a target tracking problem. In the first scheme, each agent broadcasts to its neighbors a binary value that indicates whether that agent detected the target. The neighbors then broadcast this information to their neighbors, etc. Once an agent knows the binary decisions of all other agents, it executes one recursion of a local PF. In the second DPF scheme, only one agent—located close to the current target position—is used at any given time $n$ to perform the particle filtering. Here, the number of binary values that is broadcast is smaller because the information need not be disseminated through the entire network. Once the agent that performs particle filtering obtains the binary decisions of all other agents, it calculates an approximation of the global posterior. When the target moves, a new agent close to the new target position is selected for particle filtering. Unlike in LA-based DPFs, where the posterior is transmitted, only the old estimate $\hat{\mathbf{x}}_{n-1}$ is transmitted to the new agent. The new agent then samples particles from $f(\mathbf{x}_n|\hat{\mathbf{x}}_{n-1})$.

## 12 Numerical Study

We present simulation results for a simple target tracking application. Our goal is to demonstrate basic principles, aspects, and effects of DPF operation and performance, but not to provide a comprehensive comparison of the performance of existing DPFs. Such a comparison is beyond

the scope of our treatment because the performance assessment of any specific DPF algorithm is strongly dependent on the application, scenario, system parameters, and performance criteria. Therefore, while the simulated DPFs reflect important characteristics of broad DPF classes, they are not identical to specific DPFs proposed in the literature.

### 12.1 A Target Tracking Example

We consider a single target that moves in the $x$-$y$ plane (cf. Figure 1(a)) and is represented by the state vector $\mathbf{x}_n \triangleq (x_n \;\; y_n \;\; \dot{x}_n \;\; \dot{y}_n)^\top$ containing the target's two-dimensional position and velocity. The state vector evolves according to (cf. (1)) $\mathbf{x}_n = \mathbf{G}\mathbf{x}_{n-1} + \mathbf{W}\mathbf{u}_n$, $n = 1, 2, \ldots$, where (cf. [22])

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{W} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \\ 1 & 0 \\ 0 & 1 \end{pmatrix};$$

furthermore, $\mathbf{u}_n \sim \mathcal{N}(\mathbf{0}_2, \sigma_u^2 \mathbf{I}_2)$ is Gaussian driving noise, with $\mathbf{u}_n$ and $\mathbf{u}_{n'}$ independent unless $n = n'$ and $\mathbf{u}_n$ independent of $\mathbf{x}_{n'}$ for all $n'$. This model is commonly used for target tracking (e.g., [22], [44], and [69]). We choose the driving noise variance as $\sigma_u^2 = 0.00035$, and the prior pdf of the target state as $f(\mathbf{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{C}_0)$ with $\boldsymbol{\mu}_0 = (4 \;\; 4 \;\; 0.05 \;\; 0.05)^\top$ and $\mathbf{C}_0 = \mathrm{diag}\{2, 2, 0.001, 0.001\}$.

The target emits an acoustic or radio signal with a known, constant transmit power $A$. At the position of agent $k$, denoted $\boldsymbol{\xi}_k$, the received power is modeled as $A/\|\boldsymbol{\rho}(\mathbf{x}_n) - \boldsymbol{\xi}_k\|^\kappa$, where $\boldsymbol{\rho}(\mathbf{x}_n) \triangleq (x_n \;\; y_n)^\top$ is the target position and $\kappa$ is the path loss exponent (see, e.g., [69]). The agent positions $\boldsymbol{\xi}_k$ and path loss exponent $\kappa$ are constant with respect to time and known. The (scalar) measurement $z_{n,k}$ acquired by agent $k$ at time $n$ is then given by

$$z_{n,k} \;=\; h_k(\mathbf{x}_n) + v_{n,k}\,, \qquad \text{with} \;\; h_k(\mathbf{x}_n) \;=\; \frac{A}{\|\boldsymbol{\rho}(\mathbf{x}_n) - \boldsymbol{\xi}_k\|^\kappa}\,, \tag{15}$$

where $v_{n,k}$ denotes measurement noise. We assume that the $v_{n,k}$ are Gaussian with zero mean and equal variance $\sigma_v^2$, i.e., $v_{n,k} \sim \mathcal{N}(0, \sigma_v^2)$; that $v_{n,k}$ and $v_{n',k'}$ are independent unless $(n,k) = (n',k')$; and that the $v_{n,k}$ are independent of $\mathbf{x}_{n'}$ and $\mathbf{u}_{n'}$ for all $n'$. This measurement model is a special case of (8). Note that $z_{n,k}$ does not depend on the velocities $\dot{x}_n$ and $\dot{y}_n$. The local likelihood function $f(z_{n,k}|\mathbf{x}_n)$ follows as $\mathcal{N}\big(h_k(\mathbf{x}_n), \sigma_v^2\big)$, with $h_k(\mathbf{x}_n)$ given by (15). Because the $v_{n,k}$ are independent, the factorization (9) holds and the GLF is obtained as

$$f(\mathbf{z}_n|\mathbf{x}_n) \;=\; \prod_{k=1}^{K} f(\mathbf{z}_{n,k}|\mathbf{x}_n) \;=\; \frac{1}{\sqrt{(2\pi\sigma_v^2)^K}} \exp\!\left(-\frac{1}{2\sigma_v^2}\sum_{k=1}^{K}[z_{n,k} - h_k(\mathbf{x}_n)]^2\right). \tag{16}$$

We choose $A = 10$, $\kappa = 2$, and $\sigma_v^2 = 0.001$.

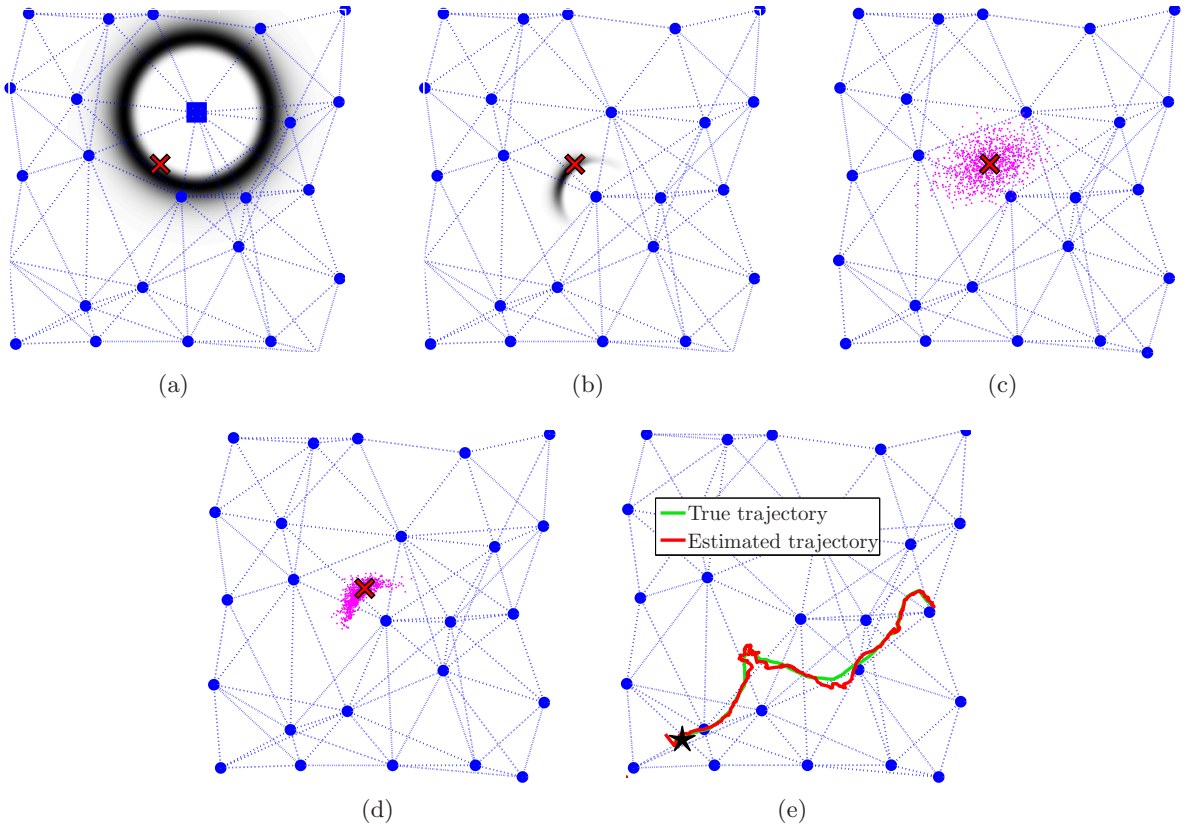



Figure 6: Example of an AN and a communication topology, along with (a) the local likelihood function of the agent indicated by the square, (b) the GLF (product of all local likelihood functions), (c) a particle representation of the predicted posterior, (d) a particle representation of the posterior (obtained by weighting the predicted particles from (c) using the GLF from (b) and performing a resampling step), and (e) a target trajectory and the corresponding estimated trajectory obtained with a centralized PF. In (a) and (b), darker shading represents higher likelihood. The cross in (a)–(d) indicates the target position. The star in (e) indicates the start point of the target trajectory.

We consider an AN consisting of $K = 25$ agents that are deployed on a jittered grid within a square of size 40m $\times$ 40m. Each agent communicates with other agents within a range of 18m. Figure 6 shows the AN and the communication topology. Examples of a local likelihood function $f(\mathbf{z}_{n,k}|\mathbf{x}_n)$ and of the GLF $f(\mathbf{z}_n|\mathbf{x}_n)$ for a randomly drawn $\mathbf{z}_n$ are shown in Figure 6(a) and (b), respectively. The local likelihood function is circularly symmetric about the agent position because the measurement function $h_k(\mathbf{x}_n)$ in (15) depends only on the distance between the target and the agent. We also see that the GLF is unimodal; this is because the GLF in (16) is the product of the circularly symmetric local likelihood functions of all agents, which have different locations due to the different agent positions. Furthermore, the nonlinearity of the $h_k(\mathbf{x}_n)$ results in a non-Gaussian

| | |
|---|---|
| CPF | centralized SIR PF |
| S-LA-DPF | single LA DPF |
| A-LA-DPF | AC DPF |
| W-C-DPF | DPF with consensus-based calculation of particle weights |
| P-C-DPF | DPF with consensus-based calculation of posterior parameters |
| L-C-DPF | DPF with consensus-based calculation of likelihood parameters |

Table 1: Abbreviations of the simulated DPFs.

posterior (shown in Figure 6(d)). In Figure 6(e), we depict a realization of the target trajectory along with a corresponding estimated trajectory that was obtained with a centralized PF.

### 12.2 Simulated DPFs and Performance Measures

We present simulation results for two LA-based DPFs and three consensus-based DPFs. More specifically, we consider a single LA DPF (S-LA-DPF) and an AC DPF (A-LA-DPF) in which the (partial) posteriors are transmitted as a particle or GM representation. The number of GM components is four unless stated otherwise. The number of agents in an AC is seven, which equals the average number of neighbors; this ensures that the S-LA-DPF and A-LA-DPF use approximately the same number of measurements at each time step $n$. The LAs are selected dynamically without central control. The current LA selects the neighboring LA closest to the current estimate of the target position as the next LA. We also consider three consensus-based DPFs that calculate particle weights (W-C-DPF), posterior parameters (P-C-DPF), or likelihood parameters (L-C-DPF) by means of average consensus algorithms with Metropolis weights [80]. We use eight consensus iterations unless stated otherwise. In the P-C-DPF, the posterior is approximated by a Gaussian whose mean and covariance are calculated using the consensus-based fusion rule of [50]. The L-C-DPF employs the LC with fourth-degree polynomial basis approximations of local likelihood functions and GLF. These DPF algorithms and the underlying techniques were described in earlier sections. As a performance benchmark, we also consider a centralized SIR PF (CPF) that has direct access to the measurements of all agents. In all DPFs, the number of particles at each agent is $J = 5000$ unless stated otherwise; the same number of particles is used in the CPF. All DPFs and the CPF employ the state-transition pdf as proposal pdf. For the reader's convenience, the abbreviations of the simulated DPFs are listed in Table 1.

As a performance measure, we use the estimated $n$-dependent root-mean-square error of the estimated target position, denoted $\mathrm{RMSE}_n$. In the LA-based DPFs, $\mathrm{RMSE}_n$ is calculated as the square root of the average of $\left\|\hat{\boldsymbol{\rho}}_n - \boldsymbol{\rho}_n\right\|^2$ over 5000 simulation runs. Here, $\boldsymbol{\rho}_n$ denotes the true

target position and $\hat{\boldsymbol{\rho}}_n$ its estimate. This estimate is obtained in the S-LA-DPF at the LA at time $n$, and in the A-LA-DPF at the last LA of the AC at time $n$. In the consensus-based DPFs, an estimate $\hat{\boldsymbol{\rho}}_{n,k}$ is obtained at each agent $k$; therefore, $\mathrm{RMSE}_n$ is calculated by averaging $\left\|\hat{\boldsymbol{\rho}}_{n,k}-\boldsymbol{\rho}_n\right\|^2$ over all agents $k = 1,\ldots,25$ in addition to the 5000 simulation runs. We furthermore calculate the (time-)averaged RMSE (ARMSE) as the square root of the average of $\mathrm{RMSE}_n^2$ over all 200 simulated time instants $n$. For the consensus-based DPFs, we also assess the error variation across the agents $k$ by the standard deviation $\sigma_{\mathrm{ARMSE}}$ of a $k$-dependent ARMSE that is obtained by averaging $\left\|\hat{\boldsymbol{\rho}}_{n,k}-\boldsymbol{\rho}_n\right\|^2$ over all time instants $n$ and simulation runs. In the calculation of $\mathrm{RMSE}_n$ and ARMSE, simulation runs corresponding to "lost tracks" are excluded. These are simulation runs where the estimation error at the final time $n=200$ exceeds 5m, i.e., half the average of the interagent distances. Removing lost tracks is important because they tend to skew the $\mathrm{RMSE}_n$ and ARMSE beyond interpretability [44,92]. However, for a complete picture of DPF performance that takes lost tracks into account, we report the percentage of lost tracks (PLT) in addition to $\mathrm{RMSE}_n$ and ARMSE.

We furthermore report the communication requirements of the DPFs and the CPF. These are defined as the total count of real numbers transmitted (over one hop between neighboring agents) during one time step $n$ within the entire network. For the CPF, we use multihop transmission of measurements from each agent to the FC, which is located in one of the corners of the network. In the S-LA-DPF, the LA transmits a representation of the posterior and its neighbors transmit their measurements. In the A-LA-DPF, each LA transmits a representation of its partial posterior. In consensus-based DPFs, each agent communicates with its neighbors by executing consensus algorithms. For the W-C-DPF, the number of consensus algorithms equals the number of particles; for the P-C-DPF, it equals the number of entries of the posterior mean vector plus the number of entries on the main diagonal and in the upper triangular part of the posterior covariance matrix; and for the L-C-DPF, it equals the number of basis functions used to approximate the GLF.

The overall computational complexity of consensus-based DPFs is generally much higher than that of LA-based DPFs and the CPF, since in the consensus-based DPFs a full-blown PF is executed at each agent. Because the computational complexity is strongly implementation dependent, we do not present a quantitative comparison of the complexities of the various DPFs.

The simulation source files and further information about the simulation setting are available online at http://www.nt.tuwien.ac.at/about-us/staff/ondrej-hlinka.

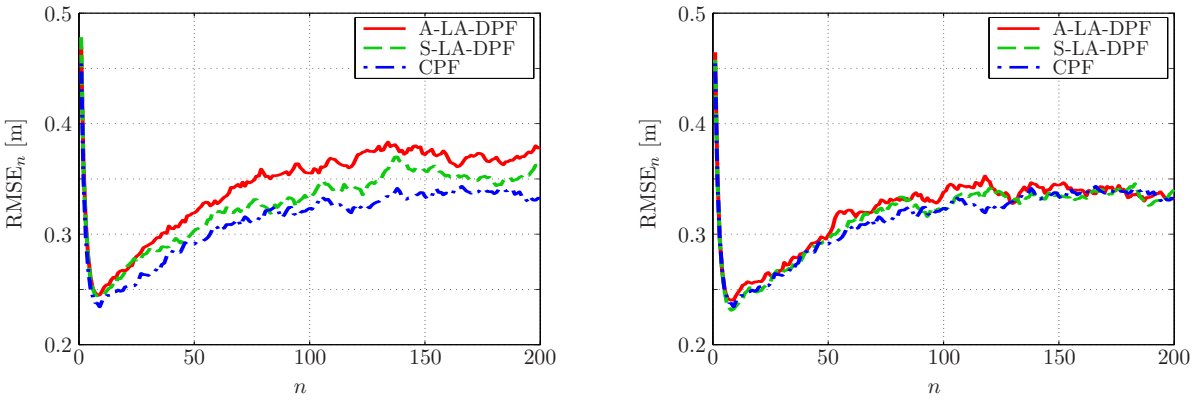



Figure 7: RMSE$_n$ versus time $n$ for the LA-based DPFs and the CPF. In the LA-based DPFs, for the transmission of (partial) posteriors between the LAs, GM representations are used in (a) and particle representations are used in (b).

### 12.3 Simulation Results: LA-based DPFs

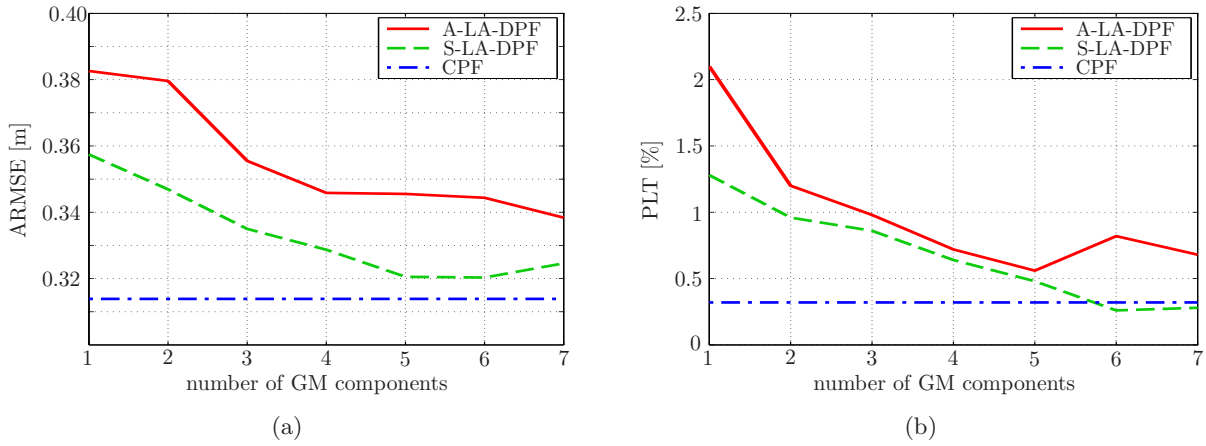
Figure 7 shows the dependence of RMSE$_n$ on $n$ for the S-LA-DPF, A-LA-DPF, and CPF. The overall shape of these RMSE$_n$ curves can be explained as follows. The initial position of the target is randomly generated from a Gaussian prior whose mean is located close to several agents. This corresponds to a high signal-to-noise ratio (SNR) of the agent measurements and, thus, results in an initial decrease of the RMSE$_n$ curve. Then, as the target randomly moves, it generally passes also through regions with lower SNRs; by averaging over all simulation runs, we thus obtain an RMSE$_n$ curve that increases but then stabilizes at a certain level. From Figure 7(a), we can see that when GM approximations of (partial) posteriors are transmitted, the S-LA-DPF outperforms the A-LA-DPF and thus performs closer to the CPF. This can be explained as follows. In the S-LA-DPF, the posterior resulting from the PF update step at the current LA is converted to a GM and transmitted to the next LA. Therefore, a single GM approximation is performed within each time step $n$. By contrast, in the A-LA-DPF, at time $n$, one GM approximation is performed each time a partial posterior is transmitted to the next LA within the AC. These multiple GM approximations cause the poorer performance of the A-LA-DPF. When particles are transmitted and thus no GM approximations are needed (see Figure 7(b)), the S-LA-DPF and A-LA-DPF perform similar and close to the CPF. Furthermore, it may be surprising that the performance of the two LA-based DPFs (which use only the measurements of a subset of agents) in Figure 7(b) is so close to that of the CPF (which uses the measurements of all agents). However, the agent scheduling in the LA-based DPFs ensures that the measurements with the highest SNR are used. Since the SNR decreases quite rapidly with a growing distance between the agent and the target,

| | ARMSE [m] | $\sigma_{\text{ARMSE}}$ [m] | PLT [%] | Communication requirements |
|---|---|---|---|---|
| CPF | 0.3139 | – | 0.32 | $\approx 64^*$ |
| S-LA-DPF (particles transmitted) | 0.3162 | – | 0.44 | $\approx 25000^*$ |
| S-LA-DPF (GMs transmitted) | 0.3287 | – | 0.64 | $\approx 65^*$ |
| A-LA-DPF (particles transmitted) | 0.3216 | – | 0.36 | 150000 |
| A-LA-DPF (GMs transmitted) | 0.3458 | – | 0.72 | 354 |
| W-C-DPF | 0.3212 | 0 | 0.38 | 1500000 |
| P-C-DPF | 0.4748 | 0.0019 | 1.44 | 2800 |
| L-C-DPF | 0.3408 | 0.1731 | 0.69 | 3000 |

Table 2: Estimation performance and communication requirements of the DPFs and the CPF.
*The exact count of real numbers transmitted during one time step depends on the specific network topology.

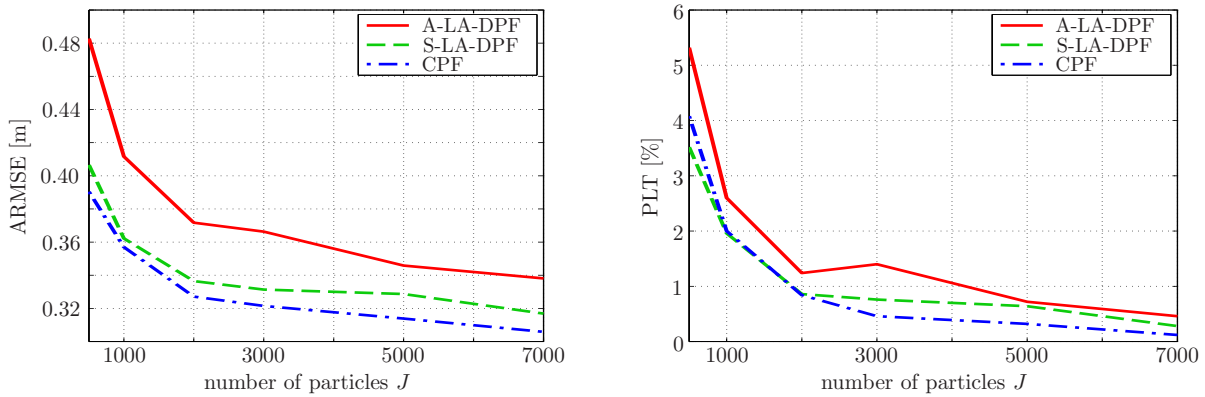
the inclusion of measurements from agents located far away from the target does not noticeably improve the performance.

In Table 2, we report the estimation performance—measured by the ARMSE, the across-agents standard deviation $\sigma_{\text{ARMSE}}$ (only relevant to consensus-based DPFs), and the PLT—and the communication requirements of the DPFs and the CPF. It can be seen that the ARMSEs of all LA-based DPFs and of the CPF are quite similar. The PLTs of the LA-based DPFs and of the CPF are all below 0.8%. The communication requirements of the S-LA-DPF are lower than those of the A-LA-DPF. This is because the measurements are only scalars; in the case of high-dimensional measurements, the opposite is true. Furthermore, for the S-LA-DPF, we assume that each agent knows the local likelihood functions of its neighbors, and therefore the numbers reported in Table 2 do not include transmission of that information. (This is not an issue for the A-LA-DPF, because each agent only uses knowledge of its own local likelihood function.) As expected, the communication requirements of both LA-based DPFs are much higher when particle representations of (partial) posteriors are transmitted than when GM representations are transmitted. In the former case, the communication requirements can be reduced by lowering the number of particles. This typically results in a performance loss, which, however, can be mitigated by using proposal adaptation. The communication requirements of the CPF are the lowest of all methods. This is again due to the low dimension of our measurements and may be very different when high-dimensional measurements have to be transmitted to the FC. Furthermore, we do not consider the overhead required for transmitting the local likelihood functions, the identities of the agents producing the measurements, and data associated with the routing protocol, and we assume that there is no need for the FC to send back information about the posterior to the agents.




(a) (b)

Figure 8: Performance of the LA-based DPFs transmitting GM representations versus the number of GM components: (a) ARMSE and (b) PLT. The performance of the CPF is shown as a reference.




(a) (b)

Figure 9: Performance of the LA-based DPFs transmitting GM representations versus the number of particles: (a) ARMSE and (b) PLT. The performance of the CPF is shown as a reference.

Figures 8 and 9 further demonstrate the performance of the LA-based DPFs transmitting GM representations. It can be seen that, as expected, the ARMSE and PLT decrease when the number of GM components increases (Figure 8) and when the number of particles increases (Figure 9).

### 12.4 Simulation Results: Consensus-based DPFs

In Figure 10, we show the dependence of $\mathrm{RMSE}_n$ on $n$ for the three consensus-based DPFs and for the CPF. It can be seen that the W-C-DPF performs almost as well as the CPF. The L-C-DPF performs second best but still very close to the CPF. The P-C-DPF, which uses a Gaussian approximation of the posteriors, exhibits a higher $\mathrm{RMSE}_n$.

From Table 2, we see that the ARMSEs of the W-C-DPF, L-C-DPF, and CPF are quite similar whereas that of the P-C-DPF is somewhat higher. For the W-C-DPF, $\sigma_{\mathrm{ARMSE}}$ is zero because

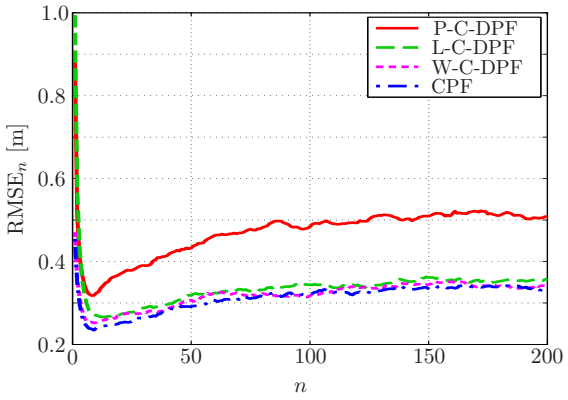
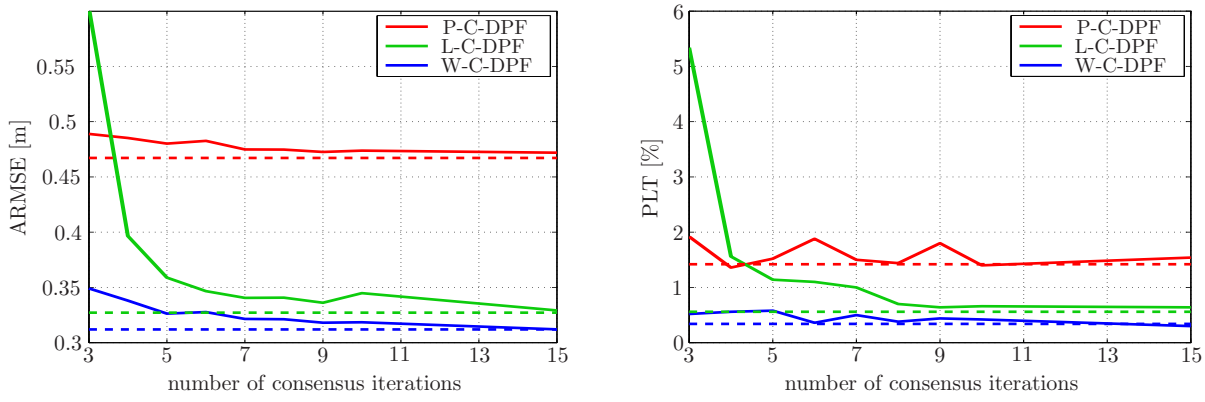



Figure 10: RMSE$_n$ versus time $n$ for the consensus-based DPFs and the CPF.

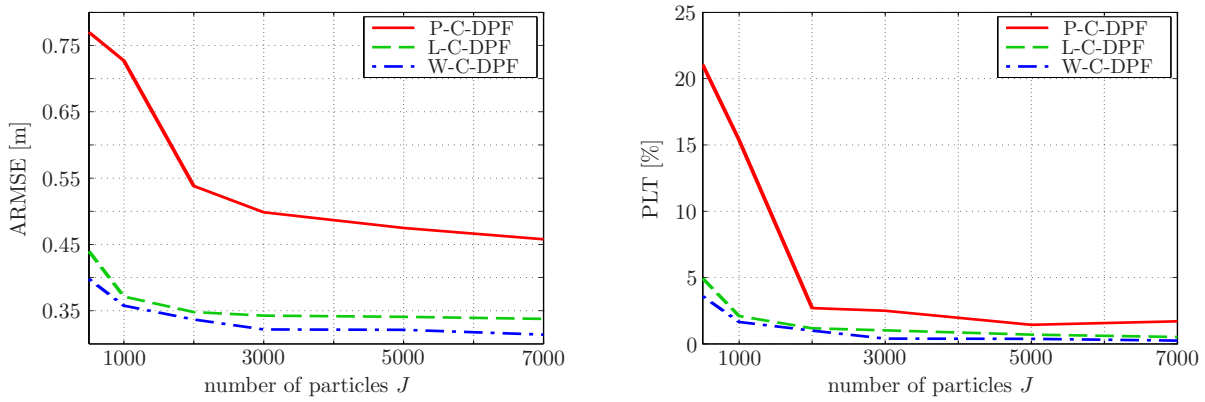
the W-C-DPF employs max-consensus algorithms that ensure identical particle weights at each agent. Furthermore, $\sigma_{\text{ARMSE}}$ is higher for the L-C-DPF than for the P-C-DPF. This is because the P-C-DPF employs a consensus step whereby Gaussian approximations of the local posteriors are combined into a global posterior, thus achieving a tighter coupling between the agents. By contrast, the local PFs of the L-C-DPF operate independently; only the GLF is computed in a distributed way. The PLT of the W-C-DPF and L-C-DPF is below 0.4 and 0.7, respectively, whereas it is 1.44 for the P-C-DPF. Thus, both in terms of ARMSE and PLT, the P-C-DPF is outperformed by the W-C-DPF and L-C-DPF. On the other hand, the communication requirements of the P-C-DPF are slightly lower than those of the L-C-DPF and much lower than those of the W-C-DPF. The high communication requirements of the W-C-DPF are due to the fact that one average and one max-consensus algorithm are needed per particle. They can be reduced by using fewer particles (the resulting performance loss can again be mitigated by proposal adaptation, as in [44]), or by employing selective gossip instead of average consensus, as in [43]. Furthermore, the communication requirements of the consensus-based DPFs are higher than those of the LA-based DPFs transmitting GM representations and also higher than those of the CPF. On the other hand, we recall that benefits derived from these increased communications are an improved robustness to agent and link failures and the availability of a particle representation of the global posterior at each agent. Finally, the communication requirements of the consensus-based DPFs (with the exception of the W-C-DPF) are much lower than those of the LA-based DPFs transmitting particle representations.

Figure 11 shows the performance of the three consensus-based DPFs versus the number of consensus iterations. We can see that, as expected, better performance is obtained for a larger number of consensus iterations; however, this improvement is small for more than about eight iterations. As a benchmark, we also consider hypothetical DPFs in which the approximate sum




(a) (b)

Figure 11: Performance of the consensus-based DPFs versus the number of consensus iterations: (a) ARMSE and (b) PLT. The dashed lines depict the performance obtained with exact sum calculation.




(a) (b)

Figure 12: Performance of the consensus-based DPFs versus the number of particles: (a) ARMSE and (b) PLT.

calculations performed by the consensus algorithms are replaced by direct, exact sum calculations; this corresponds to an infinite number of consensus iterations. Finally, Figure 12 shows that the performance of the consensus-based DPFs improves when the number of particles $J$ increases. Note, at this point, that the communication requirements of the W-C-DPF increase linearly with $J$ whereas those of the P-C-DPF and L-C-DPF are independent of $J$.

## 13 Conclusion and Outlook

DPFs are a powerful and versatile approach to decentralized state estimation in ANs, and they are especially suited to large-scale, nonlinear, and non-Gaussian systems. In this article, we provided

a survey and a classification of the DPF algorithms available to date. After a brief introduction into ANs, we set the stage for DPFs by reviewing the principles of sequential Bayesian state estimation and particle filtering. Our subsequent discussion of DPFs demonstrated the large variety of techniques that have been proposed to enable a decentralized operation of particle filtering. To make this multifaceted field more manageable and to provide a clearer distinction and appreciation of the fundamental approaches, we introduced a classification of DPF algorithms into FC-based, statistics dissemination-based, and measurement dissemination-based DPFs as well as a subdivision of statistics dissemination-based DPFs into LA-based and consensus-based DPFs. Within each (sub)class, we discussed the main features and properties of the various DPF algorithms. Finally, we presented simulation results illustrating the performance of some major DPF schemes in a target tracking application.

The development of DPF algorithms, their implementation in ANs, and their application to practical estimation tasks remain active research areas. Numerous open questions establish interesting avenues for future investigations. We briefly mention some of them as follows:

- The theoretical and experimental analysis of the performance of existing DPFs in different scenarios and applications, and of its dependence on various system and design parameters, is an important prerequisite for a successful design and deployment of DPFs. Furthermore, such an analysis can be expected to suggest algorithmic improvements.

- The nature and representation of the quantities transmitted between the agents are interesting research topics. Is it best to transmit measurements, particles, preliminary/local estimates, partial posteriors, local likelihood functions, or other quantities? How can we represent pdfs and likelihood functions in a parsimonious way? This issue is related to (distributed) source coding, though with the unconventional optimality criterion of estimation performance.

- Apart from some initial work (e.g., [13]), the issue of resilience of DPFs to synchronization errors is largely unexplored. Another issue is robustness to imperfect and intermittent interagent communication links. The channel-aware approach introduced in [68] addresses imperfect communication links in a centralized setting. Further work on this topic, especially regarding extensions to decentralized settings and more general models, is an important research direction. Some results on robustness of DPFs to intermittent communications are presented in [93].

- Going one step further, we conjecture that large performance gains can be reaped by a consistent integration of interagent communications in the design of DPFs. Ideally, techniques

for source coding/compression, modulation and channel coding (possibly of a cooperative type), multiple access and interference management, and receiver tasks (detection, decoding, synchronization, channel estimation) should be designed and optimized jointly with the DPF algorithm, using the unconventional performance criterion of estimation performance. At present, it is unknown how to achieve this goal under practical constraints on complexity.

- The development of DPFs processing multimodal measurements, such as visual and acoustic data, calls for a consistent combination of distributed particle filtering with sensor (data/information) fusion [94].

- Many DPF algorithms rely on the factorization (9) of the global likelihood function, which implies independent measurement noises. How to extend these algorithms to the case of dependent measurement noises is an open issue.

- The application of DPFs to multiple target tracking and similar problems poses challenges related to, e.g., an unknown number of targets and the association of the measurements to the targets. In this context, an interesting problem is how to combine DPF schemes with elements of random set theory [95] and data association techniques [96].

- In addition to the "primary" state estimation task (e.g., target tracking), it is often necessary to perform "secondary" estimation tasks to acquire relevant information about the AN or the environment. Examples of such secondary estimation tasks are estimation of AN topology and connectivity, agent self-localization, synchronization, estimation of statistical information and model parameters, channel estimation, and map-building (e.g., in robotics applications). Here, an interesting problem is the development of a joint estimator that consistently combines DPF-based primary estimation with secondary estimation in such a way that primary and secondary estimation support each other. For example, methods for simultaneous target tracking and agent self-localization are proposed in [15] and [16]. On a related note, it is of interest to develop DPF algorithms for scenarios with unknown or partially known prior or side information where no attempt at explicitly estimating such information can be made.

- The design of DPF schemes processing massive data—e.g., video data in camera networks—suggests a combination of distributed particle filtering with feature extraction.

- Combining DPFs with techniques from control theory is a promising approach in applications involving robotic networks, sensor/actuator networks, and networks of UAVs.

- The hardware and software implementation of DPFs is a challenging problem. In particular,

the real-time operation required by many applications places stringent constraints on the processing speed and the latency of communication links.

The facts that DPFs are formulated within a Bayesian probabilistic framework and that they use particles for a description of posteriors make them well suited to algorithmic extensions and embeddings in various directions, including those outlined above.

## Acknowledgments

We are grateful to the anonymous reviewers for numerous constructive comments and suggestions, which resulted in an improvement of this article. We acknowledge support of this work by the FWF under award S10603 within the National Research Network SISE, by the NSF under award CCF-1018323, and by the ONR under award N00014-09-1-1154.

## References

[1] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proc. F Radar Signal Process.*, vol. 140, pp. 107–113, Apr. 1993.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, pp. 174–188, Feb. 2002.

[3] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez, "Particle filtering," *IEEE Signal Process. Mag.*, vol. 20, pp. 19–38, Sep. 2003.

[4] O. Cappé, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proc. IEEE*, vol. 95, pp. 899–924, May 2007.

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

[6] A. Nayak and I. Stojmenović, *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. Hoboken, NJ: Wiley, 2010.

[7] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton, NJ: Princeton University Press, 2009.

[8] T. Shima and S. Rasmussen, *UAV Cooperative Decision and Control: Challenges and Practical Approaches*. Philadelphia, PA: SIAM, 2009.

[9] H. Aghajan and A. Cavallaro, *Multi-Camera Networks: Principles and Applications*. Burlington, MA: Academic Press, 2009.

[10] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, "Environmental wireless sensor networks," *Proc. IEEE*, vol. 98, pp. 1903–1917, Nov. 2010.

[11] J. G. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless sensor networks for healthcare," *Proc. IEEE*, vol. 98, pp. 1947–1960, Nov. 2010.

[12] T. Zhao and A. Nehorai, "Distributed sequential Bayesian estimation of a diffusive source in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 55, pp. 1511–1524, Apr. 2007.

[13] J. Beaudeau, M. F. Bugallo, and P. M. Djurić, "Target tracking with asynchronous measurements by a network of distributed mobile agents," in *Proc. IEEE ICASSP-12*, Kyoto, Japan, pp. 3857–3860, Mar. 2012.

[14] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz, "Distributed synchronization in wireless networks," *IEEE Signal Process. Mag.*, vol. 25, pp. 81–97, Sep. 2008.

[15] X. Chen, A. Edelstein, Y. Li, M. Coates, M. Rabbat, and A. Men, "Sequential Monte Carlo for simultaneous passive device-free tracking and sensor localization using received signal strength measurements," in *Proc. IPSN-11*, Chicago, IL, pp. 342–353, Apr. 2011.

[16] F. Meyer, E. Riegler, O. Hlinka, and F. Hlawatsch, "Simultaneous distributed sensor self-localization and target tracking using belief propagation and likelihood consensus," in *Proc. 46th Asilomar Conf. Sig., Syst., Comp.*, Pacific Grove, CA, Nov. 2012.

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

[18] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice Hall, 1979.

[19] D. Alspach and H. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Trans. Autom. Control*, vol. 17, pp. 439–448, Aug. 1972.

[20] R. van der Merwe, *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, OGI School of Science and Eng., Oregon Health and Science University, Hillsboro, OR, Apr. 2004.

[21] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *J. Am. Stat. Assoc.*, vol. 94, pp. 590–599, Jun. 1999.

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

[23] A. G. O. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. Boca Raton, FL: CRC Press, 1998.

[24] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. IEEE Conf. Decis. Contr.*, New Orleans, LA, pp. 5492–5498, Dec. 2007.

[25] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed Kalman filtering based on consensus strategies," *IEEE J. Sel. Areas Comm.*, vol. 26, pp. 622–633, May 2008.

[26] F. S. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering and smoothing," *IEEE Trans. Autom. Contr.*, vol. 55, pp. 2069–2084, Sep. 2010.

[27] S. Kar and J. M. F. Moura, "Gossip and distributed Kalman filtering: Weak consensus under weak detectability," *IEEE Trans. Signal Process.*, vol. 59, pp. 1766–1784, Apr. 2011.

[28] T. Vercauteren and X. Wang, "Decentralized sigma-point information filters for target tracking in collaborative sensor networks," *IEEE Trans. Signal Process.*, vol. 53, pp. 2997–3009, Aug. 2005.

[29] D. J. Lee, "Nonlinear estimation and multiple sensor fusion using unscented information filtering," *IEEE Signal Process. Letters*, vol. 15, pp. 861–864, Dec. 2008.

[30] X. Sheng and Y. H. Hu, "Distributed particle filters for wireless sensor network target tracking," in *Proc. IEEE ICASSP-05*, Philadelphia, PA, pp. 845–848, Mar. 2005.

[31] L. Zuo, K. Mehrotra, P. K. Varshney, and C. K. Mohan, "Bandwidth-efficient target tracking in distributed sensor networks using particle filters," in *Proc. FUSION-06*, Florence, Italy, Jul. 2006.

[32] Q. Cheng and P. K. Varshney, "Joint state monitoring and fault detection using distributed particle filtering," in *Proc. 41st Asilomar Conf. Sig., Syst., Comp.*, Pacific Grove, CA, pp. 715–719, Nov. 2007.

[33] M. Vemula, M. F. Bugallo, and P. M. Djurić, "Target tracking in a two-tiered hierarchical sensor network," in *Proc. IEEE ICASSP-06*, Toulouse, France, pp. 969–972, May 2006.

[34] Z. Yan, B. Zheng, and J. Cui, "Distributed particle filter for target tracking in wireless sensor network," in *Proc. EUSIPCO-06*, Florence, Italy, Sep. 2006.

[35] A. T. Ihler, J. W. Fisher III, and A. S. Willsky, "Using sample-based representations under communications constraints," Tech. Rep. 2601, Lab. Inform. and Decision Syst., Massachusetts Instit. Technol., Cambridge, MA, Dec. 2004.

[36] A. T. Ihler, *Inference in sensor networks: Graphical models and particle methods*. PhD thesis, Dept. Elect. Eng. Comput. Sci., Massachusetts Instit. Technol., Cambridge, MA, 2005.

[37] A. T. Ihler, J. W. Fisher III, and A. S. Willsky, "Particle filtering under communications constraints," in *Proc. IEEE SSP-05*, Bordeaux, France, pp. 89–94, Jul. 2005.

[38] D. Guo and X. Wang, "Dynamic sensor collaboration via sequential Monte Carlo," *IEEE J. Sel. Areas Comm.*, vol. 22, pp. 1037–1047, Aug. 2004.

[39] T. Vercauteren, D. Guo, and X. Wang, "Joint multiple target tracking and classification in collaborative sensor networks," *IEEE J. Sel. Areas Comm.*, vol. 23, pp. 714–723, Apr. 2005.

[40] 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. IPSN-05*, Los Angeles, CA, pp. 181–188, Apr. 2005.

[41] O. Hlinka, P. M. Djurić, and F. Hlawatsch, "Time-space-sequential distributed particle filtering with low-rate communications," in *Proc. 43th Asilomar Conf. Sig., Syst., Comp.*, Pacific Grove, CA, pp. 196–200, Nov. 2009.

[42] C. Nastasi and A. Cavallaro, "Distributed target tracking under realistic network conditions," in *Proc. SSPD-11*, London, UK, Sep. 2011.

[43] D. Üstebay, M. Coates, and M. Rabbat, "Distributed auxiliary particle filters using selective gossip," in *Proc. IEEE ICASSP-11*, Prague, Czech Republic, pp. 3296–3299, May 2011.

[44] S. Farahmand, S. I. Roumeliotis, and G. B. Giannakis, "Set-membership constrained particle filter: Distributed adaptation for sensor networks," *IEEE Trans. Signal Process.*, vol. 59, pp. 4122–4138, Sep. 2011.

[45] V. Savic, H. Wymeersch, and S. Zazo, "Belief consensus algorithms for fast distributed target tracking in wireless sensor networks." Available online: arXiv:1202.5261v2 [cs.DC], Aug. 2012.

[46] C. J. Bordin and M. G. S. Bruno, "Consensus-based distributed particle filtering algorithms for cooperative blind equalization in receiver networks," in *Proc. IEEE ICASSP-11*, Prague, Czech Republic, pp. 3968–3971, May 2011.

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

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

[49] D. Gu, "Distributed EM algorithm for Gaussian mixtures in sensor networks," *IEEE Trans. Neural Networks*, vol. 19, pp. 1154–1166, Jul. 2008.

[50] B. N. Oreshkin and M. J. Coates, "Asynchronous distributed particle filter via decentralized evaluation of Gaussian products," in *Proc. FUSION-10*, Edinburgh, UK, Jul. 2010.

[51] A. Simonetto, T. Keviczky, and R. Babuska, "Distributed nonlinear estimation for robot localization using weighted consensus," in *Proc. IEEE ICRA-10*, Anchorage, AK, pp. 3026–3031, May 2010.

[52] A. Mohammadi and A. Asif, "Consensus-based distributed unscented particle filter," in *Proc. IEEE SSP-11*, Nice, France, pp. 237–240, Jun. 2011.

[53] O. Hlinka, O. Slučiak, F. Hlawatsch, P. M. Djurić, and M. Rupp, "Likelihood consensus and its application to distributed particle filtering," *IEEE Trans. Signal Process.*, vol. 60, pp. 4334–4349, Aug. 2012.

[54] O. Hlinka, F. Hlawatsch, and P. M. Djurić, "Likelihood consensus-based distributed particle filtering with distributed proposal density adaptation," in *Proc. IEEE ICASSP-12*, Kyoto, Japan, pp. 3869–3872, Mar. 2012.

[55] O. Slučiak, O. Hlinka, M. Rupp, F. Hlawatsch, and P. M. Djurić, "Sequential likelihood consensus and its application to distributed particle filtering with reduced communications and latency," in *Proc. 45th Asilomar Conf. Sig., Syst., Comp.*, Pacific Grove, CA, pp. 1766–1770, Nov. 2011.

[56] M. Rosencrantz, G. Gordon, and S. Thrun, "Decentralized sensor fusion with distributed particle filters," in *Proc. UAI-03*, Acapulco, Mexico, pp. 493–500, Aug. 2003.

[57] M. J. Coates, "Distributed particle filters for sensor networks," in *Proc. IPSN-04*, Berkeley, CA, pp. 99–107, Apr. 2004.

[58] P. M. Djurić, J. Beaudeau, and M. F. Bugallo, "Non-centralized target tracking with mobile agents," in *Proc. IEEE ICASSP-11*, Prague, Czech Republic, pp. 5928–5931, May 2011.

[59] P. M. Djurić and L. Geng, "Non-centralized target tracking in networks of directional sensors," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, pp. 85–88, Mar. 2011.

[60] M. Coates and G. Ing, "Sensor network particle filters: Motes as particles," in *Proc. IEEE SSP-05*, Bordeaux, France, pp. 1152–1157, Jul. 2005.

[61] B. Jiang and B. Ravindran, "Completely distributed particle filters for target tracking in sensor networks," in *Proc. IEEE IPDPS-11*, Anchorage, AK, pp. 334–344, May 2011.

[62] L. Ong, B. Upcroft, M. Ridley, T. Bailey, S. Sukkarieh, and H. Durrant-Whyte, "Consistent methods for decentralised data fusion using particle filters," in *Proc. Int. Conf. Multisens. Fusion Integr. Intel. Sys.*, Heidelberg, Germany, pp. 85–91, Sep. 2006.

[63] L. Ong, T. Bailey, H. Durrant-Whyte, and B. Upcroft, "Decentralised particle filtering for multiple target tracking in wireless sensor networks," in *Proc. FUSION-08*, Cologne, Germany, Jun. 2008.

[64] F. Xaver, G. Matz, P. Gerstoft, and C. Mecklenbräuker, "Localization of acoustic sources using a decentralized particle filter," *EURASIP J. Wireless Commun. Networking*, vol. 2011, Sep. 2011.

[65] A. Mohammadi and A. Asif, "Distributed state estimation for large-scale nonlinear systems: A reduced order particle filter implementation," in *Proc. IEEE SSP-12*, Ann Arbor, MI, pp. 249–252, Aug. 2012.

[66] S. S. Dias and M. G. S. Bruno, "Cooperative particle filtering for emitter tracking with unknown noise variance," in *Proc. IEEE ICASSP-12*, Kyoto, Japan, pp. 2629–2632, Mar. 2012.

[67] B. N. Oreshkin and M. J. Coates, "Analysis of error propagation in particle filters with approximation," *Ann. Appl. Probab.*, vol. 21, pp. 2343–2378, Dec. 2011.

[68] O. Ozdemir, R. Niu, and P. K. Varshney, "Tracking in wireless sensor networks using particle filtering: Physical layer considerations," *IEEE Trans. Signal Process.*, vol. 57, pp. 1987–1999, May 2009.

[69] P. M. Djurić, M. Vemula, and M. F. Bugallo, "Target tracking by particle filtering in binary sensor networks," *IEEE Trans. Signal Process.*, vol. 56, pp. 2229–2238, Jun. 2008.

[70] Y. Ruan, P. Willett, A. Marrs, F. Palmieri, and S. Marano, "Practical fusion of quantized measurements via particle filtering," *IEEE Trans. Aeros. Elect. Syst.*, vol. 44, pp. 15–29, Jan. 2008.

[71] L. Zuo, R. Niu, and P. K. Varshney, "A sensor selection approach for target tracking in sensor networks with quantized measurements," in *Proc. IEEE ICASSP-08*, Las Vegas, NV, pp. 2521–2524, Apr. 2008.

[72] R. T. Sukhavasi and B. Hassibi, "Particle filtering for quantized innovations," in *Proc. IEEE ICASSP-09*, Taipei, Taiwan, pp. 2229–2232, Apr. 2009.

[73] Y. Weng, L. Xie, C. H. Tan, and G. W. Ng, "Target tracking in wireless sensor networks using particle filter with quantized innovations," in *Proc. FUSION-10*, Edinburgh, UK, Jul. 2010.

[74] J. L. Williams, J. W. Fisher III, and A. S. Willsky, "Approximate dynamic programming for communication-constrained sensor network management," *IEEE Trans. Signal Process.*, vol. 55, pp. 4300–4311, Aug. 2007.

[75] H. Wang, K. Yao, and D. Estrin, "Information-theoretic approaches for sensor selection and placement in sensor networks for target localization and tracking," *J. Comm. Netw.*, vol. 7, pp. 438–449, Dec. 2005.

[76] L. Zuo, R. Niu, and P. K. Varshney, "Posterior CRLB based sensor selection for target tracking in sensor networks," in *Proc. IEEE ICASSP-07*, Honolulu, HI, pp. 1041–1044, Apr. 2007.

[77] A. Mohammadi and A. Asif, "Decentralized sensor selection based on the distributed posterior Cramér-Rao lower bound," in *Proc. IEEE FUSION-12*, Singapore, Jul. 2012.

[78] A. S. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "Scheduling multiple sensors using particle filters in target tracking," in *Proc. IEEE SSP-03*, St. Louis, MO, pp. 549–552, Oct. 2003.

[79] C. M. Bishop, "Mixture models and EM," in *Pattern Recognition and Machine Learning*, pp. 423–459, New York, NY: Springer, 2006.

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

[81] 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.

[82] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Trans. Signal Process.*, vol. 57, pp. 2748–2761, Jul. 2009.

[83] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, pp. 1847–1864, Nov. 2010.

[84] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM J. Control Optim.*, vol. 48, pp. 33–55, Feb. 2009.

[85] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Quantized data and random link failures," *IEEE Trans. Signal Process.*, vol. 58, pp. 1383–1400, Mar. 2010.

[86] V. Schwarz and G. Matz, "Mean-square optimal weight design for average consensus," in *Proc. IEEE SPAWC-12*, Cesme, Turkey, pp. 374–378, Jun. 2012.

[87] D. Mosk-Aoyama and D. Shah, "Fast distributed algorithms for computing separable functions," *IEEE Trans. Inf. Theory*, vol. 54, pp. 2997–3007, Jul. 2008.

[88] D. Üstebay, R. Castro, and M. Rabbat, "Selective gossip," in *Proc. IEEE CAMSAP-09*, Aruba, Dutch Antilles, pp. 61–64, Dec. 2009.

[89] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, pp. 28–41, Jan. 2004.

[90] M. J. F. Gales and S. S. Airey, "Product of Gaussians for speech recognition," *Computer Speech & Language*, vol. 20, pp. 22–40, Jan. 2006.

[91] C. Y. Chong, S. Mori, and K. C. Chang, "Distributed multitarget multisensor tracking," in *Multitarget-Multisensor Tracking: Advanced Applications* (Y. Bar-Shalom, ed.), pp. 247–295, Boston, MA: Artech House, 1990.

[92] P. Willett, R. Niu, and Y. Bar-Shalom, "Integration of Bayes detection with target tracking," *IEEE Trans. Signal Process.*, vol. 49, pp. 17–29, Jan. 2001.

[93] A. Mohammadi and A. Asif, "Distributed particle filter implementation with intermittent/irregular consensus convergence." Available online: arXiv:1112.2431v2 [cs.DC], Sep. 2012.

[94] M. E. Liggins, D. L. Hall, and J. Llinas, *Handbook of Multisensor Data Fusion: Theory and Practice*. Boca Raton, FL: CRC Press, 2nd ed., 2009.

[95] R. Mahler, "Random set theory for multisource-multitarget information fusion," in *Handbook of Multisensor Data Fusion: Theory and Practice*, pp. 369–410, Boca Raton, FL: CRC Press, 2nd ed., 2009.

[96] J. Liu, M. Chu, and J. E. Reich, "Multitarget tracking in distributed sensor networks," *IEEE Signal Process. Mag.*, vol. 24, pp. 36–46, May 2007.