Decentralized Particle Filter With Arbitrary State Decomposition

Tianshi Chen, *Member, IEEE*, Thomas B. Schön, *Member, IEEE*, Henrik Ohlsson, *Member, IEEE*, and Lennart Ljung, *Fellow, IEEE*

***Abstract*—In this paper, a new particle filter (PF) which we refer to as the decentralized PF (DPF) is proposed. By first decomposing the state into two parts, the DPF splits the filtering problem into two nested subproblems and then handles the two nested subproblems using PFs. The DPF has the advantage over the regular PF that the DPF can increase the level of parallelism of the PF. In particular, part of the resampling in the DPF bears a parallel structure and can thus be implemented in parallel. The parallel structure of the DPF is created by decomposing the state space, differing from the parallel structure of the distributed PFs which is created by dividing the sample space. This difference results in a couple of unique features of the DPF in contrast with the existing distributed PFs. Simulation results of two examples indicate that the DPF has a potential to achieve in a shorter execution time the same level of performance as the regular PF.**

***Index Terms*—Parallel algorithms, particle filtering, nonlinear system, state estimation.**

I. Introduction

In this paper, we study the filtering problem for the following nonlinear discrete-time system

$$\begin{aligned} \xi_{t+1} &= f_t(\xi_t, v_t) \\ y_t &= h_t(\xi_t, e_t) \end{aligned} \tag{1}$$

where t is the discrete-time index, $\xi_t \in \mathbb{R}^{n_\xi}$ is the state at time $t, y_t \in \mathbb{R}^{n_y}$ is the measurement output, $v_t \in \mathbb{R}^{n_v}$ and $e_t \in \mathbb{R}^{n_e}$ are independent noises whose known distributions are independent of t, ξ_t and y_t, and $f_t(\cdot)$ and $h_t(\cdot)$ are known functions. The filtering problem consists of recursively estimating the posterior density $p(\xi_t \mid y_{0:t})$ where $y_{0:t} \triangleq \{y_0, \ldots, y_t\}$. Analytic solutions to the filtering problem are only available for a relatively small and restricted class of systems, the most important being the Kalman filter [19] which assumes that (1) has a linear-Gaussian structure. A class of powerful numerical algorithms to the filtering problem are particle filters (PFs), which are sequential Monte Carlo methods based on particle representations of probability densities [2]. Since the seminal work [15], PFs have become an important tool in handling the nonlinear non-Gaussian filtering problem, and have found many applications in statistical signal processing, economics and engineering; see, e.g., [2], [12], [14], and [16] for recent surveys of PFs.

In this paper, a new PF, which we refer to as the decentralized PF (DPF), will be proposed. By first decomposing the state into two parts, the DPF splits the filtering problem of (1) into two nested subproblems and then handles the two nested subproblems using PFs. The DPF has the advantage over the regular PF that the DPF can increase the level of parallelism of the PF in the sense that besides the particle generation and the importance weights calculation, part of the resampling in the DPF can also be implemented in parallel. As will be seen from the DPF algorithm, there are actually two resampling steps in the DPF. The first resampling in the DPF, like the resampling in the regular PF, cannot be implemented in parallel, but the second resampling bears a parallel structure and can thus be implemented in parallel. Hence, the parallel implementation of the DPF can be used to shorten the execution time of the PF.

As pointed out in [5], the application of PFs in real-time systems is limited due to its computational complexity which is mainly caused by the resampling involved in the PF. The resampling is essential in the implementation of the PF as without resampling the variance of the importance weights will increase over time [13]. The resampling however introduces a practical problem. The resampling limits the opportunity to parallelize since all the particles must be combined, although the particle generation and the importance weights calculation of the PF can still be realized in parallel [13]. Therefore, the resampling becomes a bottleneck to shorten the execution time of the PF. Recently, some distributed resampling algorithms for parallel implementation of PFs have been proposed in [5] and [22]. The idea of the distributed resampling is to divide the sample space into several strata or groups such that the resampling can be performed independently for each stratum or group and can thus be implemented in parallel. The effect of different distributed resampling algorithms on the variance of the importance weights has been analyzed in [22]. Based on the introduced distributed resampling algorithms, a couple of distributed PFs have been further proposed in [5] and [22], such as the distributed resampling with proportional allocation PF (DRPA-PF) and the distributed resampling with nonproportional allocation PF (DRNA-PF).

The underlying idea of the DPF is different from that of the existing distributed PFs, while they all have parallel structure. The parallel structure of the DPF is created by decomposing

Manuscript received February 18, 2010; revised June 01, 2010, September 01, 2010; accepted October 23, 2010. Date of publication November 11, 2010; date of current version January 12, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Peter J. Schreier. An abridged version of this paper was presented at the 2010 IEEE Conference on Decision and Control, Atlanta, GA, December, 2010.

The authors are with the Division of Automatic Control, Department of Electrical Engineering, Linköping University, Linköping, 58183, Sweden (e-mail: tschen@isy.liu.se; schon@isy.liu.se; ohlsson@isy.liu.se; ljung@isy.liu.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2010.2091639

the state space, differing from the parallel structure of the distributed PFs which is created by dividing the sample space. This difference results in a couple of unique features of the DPF in contrast with the existing distributed PFs. First, compared to the DRPA-PF, the DPF allows a simpler scheme for particle routing and actually treats each processing element as a particle in the particle routing. Second, the DPF does not have the efficiency decrease problem of the DRPA-PF. Given a PF with parallel structure, it works most efficiently if each processing element handles the same number of particles. However, the efficiency of the DRPA-PF usually decreases, since the numbers of particles produced by each processing element are not evenly but randomly distributed among the processing elements. Third, it will be verified by two numerical examples that, the DPF has the potential to achieve in a shorter execution time the same level of performance as the bootstrap PF. In contrast, the DRNA-PF actually trades the PF performance for the speed improvement [5]. Besides, the level of parallelism of the DPF can be further increased in two ways so that the execution time of the parallel implementation of the DPF can be further shortened; the first one is to utilize any of the distributed resampling algorithms proposed in [5], [22] to perform the first resampling of the DPF, and the other is based on an extension of the DPF. As a result, the DPF is a new option for the application of PFs in real-time systems and the parallel implementation of PFs.

The rest of the paper is organized as follows. The problem formulation is given in Section II. In Section III, the DPF algorithm is first derived and then summarized, and finally some issues regarding the implementation of the DPF are discussed. In Section IV, some discussions about the DPF and its comparison with the existing distributed PFs are made. In Section V, two numerical examples are elaborated to show the efficacy of the DPF. Finally, we conclude the paper in Section VI.

II. Problem Formulation

A. Intuitive Preview

The formulas for particle filtering tend to look complex, and it may be easy to get lost in indices and update expressions. Let us therefore provide a simple and intuitive preview to get the main ideas across.

Filtering is about determining the posterior densities of the states. If the state is two-dimensional with components x and z, say, the density is a surface over the $x-z$ plane, see Fig. 1. One way to estimate the density is to fix M points in the plane in a regular grid [Fig. 1(a)] and update the values of the density according to Bayesian formulas. This is known as the point-mass filter [3], [6]. Another way is to throw M points at the plane at random [Fig. 1(b)], and let them move to important places in the plane, and update the values of the densities at the chosen points, using Bayesian formulas. This is a simplified view of what happens in the regular PF. A third way is illustrated in [Fig. 1(c)]: Let the points move to well chosen locations, but restrict them to be aligned parallel to one of the axes (the z axis in the plot). The parallel lines can move freely, as can the points on the lines, but there is a restriction of the pattern as depicted. The algorithm we develop in this paper (DPF) gives both the movements of the lines and the positions of the points on the lines, and the density values at the chosen points, by application of Bayesian formulas.

It is well known that the regular PF outperforms the point-mass filter with the same number of points, since it can concentrate them to important areas. One would thus expect that the DPF would give worse accuracy than the regular PF with the same number of points, since it is less "flexible" in the allocation of points. On the other hand, the structure might allow more efficient ways of calculating new point locations and weights. That is what we will develop and study in the following sections.

B. Problem Statement

Consider (1). Suppose that the state ξ_t can be decomposed as

$$\xi_t = \begin{bmatrix} x_t \\ z_t \end{bmatrix} \tag{2}$$

and accordingly that (1) can be decomposed as

$$\begin{aligned} x_{t+1} &= f_t^x(x_t, z_t, v_t^x) \\ z_{t+1} &= f_t^z(x_t, z_t, v_t^z) \\ y_t &= h_t(x_t, z_t, e_t) \end{aligned} \tag{3}$$

where $x_t \in \mathbb{R}^{n_x}, z_t \in \mathbb{R}^{n_z}$, and $v_t = [(v_t^x)^T \ (v_t^z)^T]^T$ with $v_t^x \in \mathbb{R}^{n_{vx}}$ and $v_t^z \in \mathbb{R}^{n_{vz}}$. In the following, it is assumed for convenience that the probability densities $p(x_0), p(z_0 \,|\, x_0)$ and for $t \geq 0, p(x_{t+1} \,|\, x_t, z_t), p(z_{t+1} \,|\, x_{t:t+1}, z_t)$ and $p(y_t \,|\, x_t, z_t)$ are known.

In this paper, we will study the filtering problem of recursively estimating the posterior density $p(z_t, x_{0:t} \,|\, y_{0:t})$. According to the following factorization:

$$p(z_t, x_{0:t} \,|\, y_{0:t}) = p(z_t \,|\, x_{0:t}, y_{0:t}) p(x_{0:t} \,|\, y_{0:t}) \tag{4}$$

where $x_{0:t} \triangleq \{x_0, \ldots, x_t\}$, the filtering problem (4) can be split into two nested subproblems:

1) recursively estimating the density $p(x_{0:t} \,|\, y_{0:t})$;
2) recursively estimating the density $p(z_t \,|\, x_{0:t}, y_{0:t})$.

That the two subproblems are nested can be seen from Fig. 2 where we have sketched the five steps used to derive the recurrence relation of the conceptual solution to the filtering problem (4). Since there is in general no analytic solution to the filtering problem (4), a numerical algorithm, i.e., the DPF is introduced to provide recursively the empirical approximations to $p(x_{0:t} \,|\, y_{0:t})$ and $p(z_t \,|\, x_{0:t}, y_{0:t})$. The DPF actually handles the two nested subproblems using PFs. Roughly speaking, the DPF solves the first subproblem using a PF with N_x particles $(x_{0:t}^{(i)}, i = 1, \ldots, N_x)$ to estimate $p(x_{0:t} \,|\, y_{0:t})$. Then the DPF handles the second subproblem using N_x PFs with N_z particles each to estimate $p(z_t \,|\, x_{0:t}^{(i)}, y_{0:t}), i = 1, \ldots, N_x$. As a result of the nestedness of the two subproblems, it will be seen later that the steps of the PF used to estimate $p(x_{0:t} \,|\, y_{0:t})$ is nested with that of the N_x PFs used to estimate $p(z_t \,|\, x_{0:t}^{(i)}, y_{0:t})$.

Remark 2.1: The idea of decomposing the state into two parts and accordingly splitting the filtering problem into two nested subproblems is not new. Actually, it has been used in the Rao-Blackwellized PF (RBPF); see, e.g., [1], [8], [9], [11], [13], and [25]. However, the RBPF imposes certain *tractable* substructure assumption on the system considered and, hence,

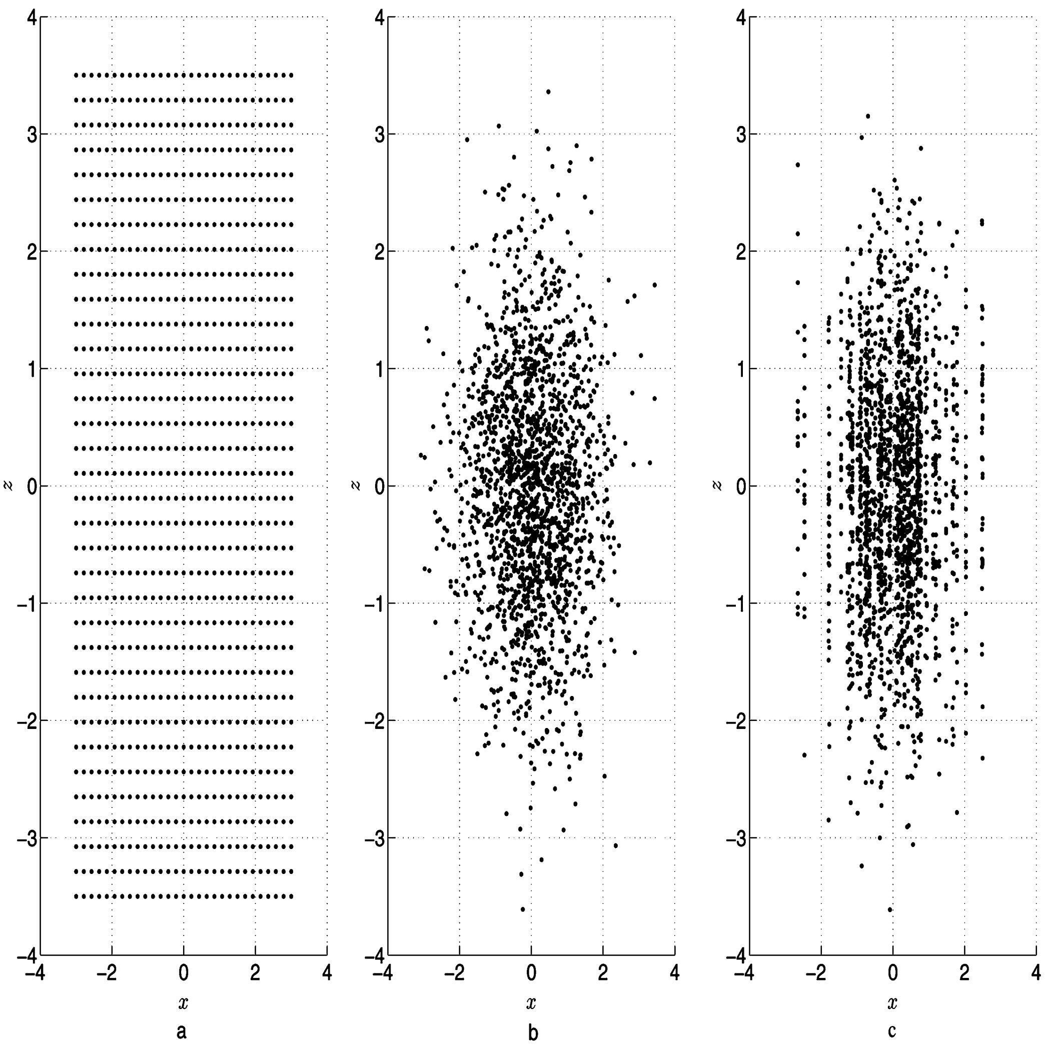

Fig. 1. Patterns for points where the posterior densities are computed/estimated. (a) A fixed regular grid used in the point-mass filter. (b) Randomly allocated points following the regular PF equations. (c) Points randomly allocated to vertical parallel lines (which themselves are randomly located) used in the DPF.

$$\begin{tikzcd}
p(x_{0:t}|y_{0:t-1}) \arrow[d] & & & p(z_t|x_{0:t},y_{0:t-1}) \arrow[dll] \arrow[dd] \\
p(x_{0:t}|y_{0:t}) \arrow[dd] & p(y_t|x_{0:t},y_{0:t-1}) \arrow[l] & p(y_t|x_t,z_t) \arrow[l] \arrow[dr] & \\
 & & & p(z_t|x_{0:t},y_{0:t}) \arrow[dll] \arrow[dd] \\
p(x_{0:t+1}|y_{0:t}) & p(x_{t+1}|x_{0:t},y_{0:t}) \arrow[l] & p(x_{t+1}|x_t,z_t) \arrow[l] \arrow[dr] & \\
 & & & p(z_t|x_{0:t+1},y_{0:t}) \arrow[d] \\
 & & p(z_{t+1}|x_{t:t+1},z_t) \arrow[r] & p(z_{t+1}|x_{0:t+1},y_{0:t})
\end{tikzcd}$$

Fig. 2. Sketch of the steps used to derive the conceptual solution to the filtering problem (4). Assume that the probability densities $p(x_0), p(z_0 \mid x_0)$ and for $t \geq 0, p(x_{t+1} \mid x_t, z_t), p(z_{t+1} \mid x_{t:t+1}, z_t)$ and $p(y_t \mid x_t, z_t)$ are known. With a slight abuse of notation let $p(x_0 \mid y_{0:-1}) = p(x_0)$ and $p(z_0 \mid x_0, y_{0:-1}) = p(z_0 \mid x_0)$. Then five steps are needed to derive the recurrence relation of the conceptual solution to the filtering problem (4). The specific relations between the different densities can be obtained by straightforward application of Bayesian formulas and thus are omitted. Instead, the notation $p_1(\,\cdot\,) \rightarrow p_2(\,\cdot\,)$ is used to indicate that the calculation of the density $p_2(\,\cdot\,)$ makes use of the knowledge of the density $p_1(\,\cdot\,)$.

solves one of the subproblem with a number of optimal filters, such as the Kalman filter [19] or the HMM filter [24]. In particular, the filtering problem (4) has been previously studied in [25] where (3) is assumed to be conditionally (on x_t) linear in z_t and subject to Gaussian noise. Due to these assumptions, the state z_t of (3) is marginalized out by using the Kalman filter. However, since there is no tractable substructure assumption made on (3) in this paper, no part of the state ξ_t is analytically tractable as was the case in [1], [8], [9], [11], [13], and [25]. $\Diamond$

In the following, let $\tilde{x} \sim p(x)$ denote that $\tilde{x}$ is a sample drawn from the density $p(x)$ of the random variable x, let $\mathcal{N}(m, \Sigma)$ denote the (multivariate) Gaussian probability density with mean vector m and covariance matrix Σ and let $\mathrm{P}(A)$ denote the probability of the event A. For convenience, for each $i = 1, \ldots, N, \alpha_i = \beta_i / \sum_{j=1}^{N} \beta_j$ is denoted by

$$\alpha_i \propto \beta_i; \quad \sum_{i=1}^{N} \alpha_i = 1 \tag{5}$$

where N is a natural number, $\alpha_i, \beta_i, i = 1, \ldots, N$, are positive real numbers, and $\alpha_i \propto \beta_i$ denotes that α_i is proportional to β_i.

III. Decentralized Particle Filter

In this section, the DPF algorithm is first derived and then summarized. Finally, some issues regarding the implementation of the DPF are discussed.

We here make a comment regarding some notations used throughout the paper. Suppose we have

$$P_N(d\xi_t) = \frac{1}{N}\sum_{i=1}^{N}\delta_{\xi_t^{(i)}}(d\xi_t) \tag{6}$$

where $\delta_{\xi_t^{(i)}}(A)$ is a Dirac measure for a given $\xi_t^{(i)}$ and a measurable set A. Then the expectation of any test function $g(\xi_t)$ with respect to $p(\xi_t)$ can be approximated by

$$\int g(\xi_t)p(\xi_t)d\xi_t \approx \int g(\xi_t)P_N(d\xi_t) = \frac{1}{N}\sum_{i=1}^{N} g\left(\xi_t^{(i)}\right). \tag{7}$$

Although the distribution $P_N(d\xi_t)$ does not have a well defined density with respect to the Lebesgue measure, it is a common convention in the particle filtering community [2], [15] to use

$$p_N(\xi_t) = \frac{1}{N}\sum_{i=1}^{N}\delta\left(\xi_t - \xi_t^{(i)}\right) \tag{8}$$

where $\delta(\,\cdot\,)$ is a Dirac delta function, as if it is an *empirical approximation of the probability density* $p(\xi_t)$ with respect to the Lebesgue measure. The notations like (8) and the corresponding terms aforementioned are, in the most rigorous mathematical sense, not correct. However, they enable one to avoid the use of measure theory which indeed simplifies the representation a lot especially when a theoretical convergence proof is not the concern.

A. Derivation of the DPF Algorithm

First, we initialize the particles $\tilde{x}_0^{(i)} \sim p(x_0), i = 1,\ldots,N_x$, and for each $\tilde{x}_0^{(i)}$, the particles $\tilde{z}_0^{(i,j)} \sim p(z_0 \mid \tilde{x}_0^{(i)}), j = 1,\ldots,N_z$. Then the derivation of the DPF algorithm will be completed in two steps by the induction principle. In the first step, we show that Inductive Assumptions 3.1 to 3.3 to be introduced below hold at $t = 1$. In the second step, assume that Inductive Assumptions 3.1 to 3.3 hold at t for $t \geq 1$, then we show that Inductive Assumptions 3.1 to 3.3 hold recursively at $t+1$.

In the following, we first introduce Inductive Assumptions 3.1 to 3.3 at t.

Assumption 3.1: The DPF produces N_x particles $x_{0:t-1}^{(i)}, i = 1,\ldots,N_x$ and an unweighted empirical approximation of $p(x_{0:t-1} \mid y_{0:t-1})$ as

$$\tilde{p}_{N_x}(x_{0:t-1} \mid y_{0:t-1}) = \frac{1}{N_x}\sum_{i=1}^{N_x}\delta\left(x_{0:t-1} - x_{0:t-1}^{(i)}\right) \tag{9}$$

and for each path $x_{0:t-1}^{(i)}, i = 1,\ldots,N_x$, the DPF produces N_z particles $\bar{z}_{t-1}^{(i,j)}, j = 1,\ldots,N_z$, and a weighted empirical approximation of $p(z_{t-1} \mid x_{0:t-1}^{(i)}, y_{0:t-1})$ as

$$p_{N_z}\left(z_{t-1} \mid x_{0:t-1}^{(i)}, y_{0:t-1}\right) = \sum_{j=1}^{N_z}\bar{q}_{t-1}^{(i,j)}\delta\left(z_{t-1} - \bar{z}_{t-1}^{(i,j)}\right) \tag{10}$$

where $\bar{q}_{t-1}^{(i,j)}$ is the importance weight and its definition will be given in the derivation.

Assumption 3.2: The particles $\tilde{x}_t^{(i)}, i = 1,\ldots,N_x$, are generated according to the proposal function $\pi(x_t \mid x_{0:t-1}^{(i)}, y_{0:t-1})$ and for each $\tilde{x}_t^{(i)}, i = 1,\ldots,N_x$, the particles $\tilde{z}_t^{(i,j)}, j = 1,\ldots,N_z$, are generated according to the proposal function $\pi(z_t \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$ where $\tilde{x}_{0:t}^{(i)} \triangleq (x_{0:t-1}^{(i)}, \tilde{x}_t^{(i)})$.

Assumption 3.3: For each $i = 1,\ldots,N_x$, an approximation $p_{N_z}(y_t \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$ of $p(y_t \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$ can be obtained as

$$p_{N_z}\left(y_t \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1}\right) = \sum_{j=1}^{N_z}\tilde{r}_t^{(i,j)}p(y_t \mid \tilde{x}_t^{(i)}, \tilde{z}_t^{(i,j)}) \Big/ \sum_{l=1}^{N_z}\tilde{r}_t^{(i,l)} \tag{11}$$

with $\tilde{r}_t^{(i,j)} = \tilde{p}_{N_z}(\tilde{z}_t^{(i,j)} \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})/\pi(\tilde{z}_t^{(i,j)} \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$, where $\tilde{p}_{N_z}(\tilde{z}_t^{(i,j)} \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$ is an approximation of $p(\tilde{z}_t^{(i,j)} \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$ and its definition will be given in the derivation.

Remark 3.1: In Inductive Assumptions 3.1 and 3.3, $\tilde{p}_{N_x}(x_{0:t-1} \mid y_{0:t-1})$ and $p_{N_z}(z_{t-1} \mid x_{0:t-1}^{(i)}, y_{0:t-1})$ are empirical approximations of $p(x_{0:t-1} \mid y_{0:t-1})$ and $p(z_{t-1} \mid x_{0:t-1}^{(i)}, y_{0:t-1})$, respectively. These approximations should of course be as close to the true underlying density as possible. This closeness is typically assessed via some test function $g_{t-1}(\,\cdot\,)$ used in the following sense:

$$I(g_{t-1}) = \int g_{t-1}(z_{t-1}, x_{0:t-1}) \times p(z_{t-1}, x_{0:t-1} \mid y_{0:t-1})dz_{t-1}dx_{0:t-1} \tag{12}$$

where $g_{t-1} : \mathbb{R}^{n_z} \times \mathbb{R}^{t\times n_x} \rightarrow \mathbb{R}$ is a test function. With the empirical approximations defined in (9) and (10), an estimate $I_{N_x,N_z}(g_{t-1})$ of $I(g_{t-1})$ is obtained as follows:

$$I_{N_x,N_z}(g_{t-1}) = \frac{1}{N_x}\sum_{i=1}^{N_x}\sum_{j=1}^{N_z}\bar{q}_{t-1}^{(i,j)}g_{t-1}\left(\bar{z}_{t-1}^{(i,j)}, x_{0:t-1}^{(i)}\right). \tag{13}$$

A convergence result formalizing the "closeness" of (13) to (12), as N_x and N_z tend to infinity, will be in line with our earlier convergence results [18] of the PF for rather arbitrary unbounded test functions. ◊

Then in the first step we should show that Inductive Assumptions 3.1 to 3.3 hold at $t = 1$. However, since the first step is exactly the same as the second step, we only consider the second step here due to the space limitation. In the second step, assume that Inductive Assumptions 3.1 to 3.3 hold at t, then we will show in seven steps that Inductive Assumptions 3.1 to 3.3 recursively hold at $t+1$.

1) *Measurement update of $x_{0:t}$ based on y_t*

By using importance sampling, this step aims to derive a weighted empirical approximation of $p(x_{0:t} \mid y_{0:t})$. Note that an unweighted empirical approximation $\tilde{p}_{N_x}(x_{0:t-1} \mid y_{0:t-1})$ of $p(x_{0:t-1} \mid y_{0:t-1})$ has been given as (9) in Inductive Assumption 3.1, then simple calculation shows that the following equation holds approximately:

$$p(x_{0:t} \mid y_{0:t}) \propto \tilde{p}_{N_x}(x_{0:t-1} \mid y_{0:t-1})p(x_t \mid x_{0:t-1}, y_{0:t-1}) \times p(y_t \mid x_{0:t}, y_{0:t-1}). \tag{14}$$

From Inductive Assumptions 3.1 and 3.2, $x_{0:t-1}^{(i)}, i = 1,\ldots,N_x$, can be regarded as samples drawn from $\tilde{p}_{N_x}(x_{0:t-1} \mid y_{0:t-1})$, and for $i = 1,\ldots,N_x$, $\tilde{x}_t^{(i)}$ is the sample drawn from $\pi(x_t \mid x_{0:t-1}^{(i)}, y_{0:t-1})$. Therefore $\tilde{p}_{N_x}(x_{0:t-1} \mid y_{0:t-1})\pi(x_t \mid x_{0:t-1}^{(i)}, y_{0:t-1})$ can be treated as the importance function. On the other hand, from (10) and

$$\begin{aligned} p(x_t \mid x_{0:t-1}, y_{0:t-1}) &= \int p(z_{t-1} \mid x_{0:t-1}, y_{0:t-1}) \\ &\quad \times p(x_t \mid x_{t-1}, z_{t-1}) dz_t \end{aligned} \tag{15}$$

an approximation $p_{N_z}(\tilde{x}_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{0:t-1})$ of $p(\tilde{x}_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{0:t-1})$ can be obtained as

$$\begin{aligned} p_{N_z}&\left(\tilde{x}_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{0:t-1}\right) \\ &= \sum_{j=1}^{N_z} \bar{q}_{t-1}^{(i,j)} p\left(\tilde{x}_t^{(i)} \mid x_{t-1}^{(i)}, \bar{z}_{t-1}^{(i,j)}\right). \end{aligned} \tag{16}$$

Moreover, an approximation $p_{N_z}(y_t \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$ of $p(y_t \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$ has been given as (11) in Inductive Assumption 3.3.

Now using the importance sampling yields a weighted empirical approximation of $p(x_{0:t} \mid y_{0:t})$ as

$$p_{N_x}(x_{0:t} \mid y_{0:t}) = \sum_{i=1}^{N_x} w_t^{(i)} \delta\left(x_{0:t} - \tilde{x}_{0:t}^{(i)}\right) \tag{17}$$

where $w_t^{(i)}$ is evaluated according to

$$\begin{aligned} w_t^{(i)} &\propto \frac{p_{N_z}\left(y_t \mid \tilde{x}_{0:t}^{(i)}, y_{0:t-1}\right) p_{N_z}\left(\tilde{x}_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{0:t-1}\right)}{\pi\left(\tilde{x}_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{0:t-1}\right)}; \\ \sum_{i=1}^{N_x} w_t^{(i)} &= 1. \end{aligned} \tag{18}$$

2) *Resampling of* $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$, $i = 1,\ldots,N_x$

By using resampling, this step aims to derive an unweighted empirical approximation of $p(x_{0:t} \mid y_{0:t})$.

Resample N_x times from the discrete distribution over $\{\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1,\ldots,N_x\}$ with probability mass $w_t^{(i)}$ associated with the element $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$ to generate samples $\{x_{0:t}^{(i)}, \bar{z}_t^{(i,1)}, r_t^{(i,1)}, \ldots, \bar{z}_t^{(i,N_z)}, r_t^{(i,N_z)}\}, i = 1,\ldots,N_x$, so that for any m

$$\begin{aligned} \mathrm{P}&\left\{\left\{x_{0:t}^{(m)}, \bar{z}_t^{(m,1)}, r_t^{(m,1)}, \ldots, \bar{z}_t^{(m,N_z)}, r_t^{(m,N_z)}\right\}\right. \\ &\left.= \left\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\right\}\right\} = w_t^{(i)} \end{aligned} \tag{19}$$

where $r_t^{(i,j)}$ can be defined as $r_t^{(i,j)} = \tilde{p}_{N_z}(\bar{z}_t^{(i,j)} \mid x_{0:t}^{(i)}, y_{0:t-1})/\pi(\bar{z}_t^{(i,j)} \mid x_{0:t}^{(i)}, y_{0:t-1})$ according to the definition of $\tilde{r}_t^{(i,j)}$ in Inductive Assumption 3.3 (see Section III-C–1 for more detailed explanation).

As a result, it follows from (17) that an unweighted empirical approximation of $p(x_{0:t} \mid y_{0:t})$ is obtained as

$$\tilde{p}_{N_x}(x_{0:t} \mid y_{0:t}) = \frac{1}{N_x} \sum_{i=1}^{N_x} \delta\left(x_{0:t} - x_{0:t}^{(i)}\right). \tag{20}$$

3) *Measurement update of z_t based on y_t*

By using importance sampling, this step aims to derive a weighted empirical approximation of $p(z_t \mid x_{0:t}^{(i)}, y_{0:t})$. Simple calculation shows that

$$p\left(z_t \mid x_{0:t}^{(i)}, y_{0:t}\right) \propto p\left(z_t \mid x_{0:t}^{(i)}, y_{0:t-1}\right) p\left(y_t \mid x_t^{(i)}, z_t\right). \tag{21}$$

From Inductive Assumption 3.2, for each $x_t^{(i)}$, $i = 1,\ldots,N_x$, the particles $\bar{z}_t^{(i,j)}, j = 1,\ldots,N_z$, are generated from the proposal function $\pi(z_t \mid x_{0:t}^{(i)}, y_{0:t-1})$. Therefore $\pi(z_t \mid x_{0:t}^{(i)}, y_{0:t-1})$ is chosen as the importance function. On the other hand, note that an approximation $\tilde{p}_{N_z}(\bar{z}_t^{(i,j)} \mid x_{0:t}^{(i)}, y_{0:t-1})$ of $p(\bar{z}_t^{(i,j)} \mid x_{0:t}^{(i)}, y_{0:t-1})$ has been given in Inductive Assumption 3.3.

Then using importance sampling and also noting the definition of $r_t^{(i,j)}$ yields, for each $i = 1,\ldots,N_x$, a weighted empirical approximation of $p(z_t \mid x_{0:t}^{(i)}, y_{0:t})$ as

$$p_{N_z}\left(z_t \mid x_{0:t}^{(i)}, y_{0:t}\right) = \sum_{j=1}^{N_z} \bar{q}_t^{(i,j)} \delta\left(z_t - \bar{z}_t^{(i,j)}\right) \tag{22}$$

where $\bar{q}_t^{(i,j)}$ is evaluated according to

$$\bar{q}_t^{(i,j)} \propto p\left(y_t \mid x_t^{(i)}, \bar{z}_t^{(i,j)}\right) r_t^{(i,j)}; \quad \sum_{j=1}^{N_z} \bar{q}_t^{(i,j)} = 1. \tag{23}$$

4) *Generation of particles $\tilde{x}_{t+1}^{(i)}, i = 1,\ldots,N_x$*

Assume that the particles $\tilde{x}_{t+1}^{(i)}, i = 1,\ldots,N_x$, are generated according to the proposal function $\pi(x_{t+1} \mid x_{0:t}^{(i)}, y_{0:t})$.

5) *Measurement update of z_t based on x_{t+1}*

By using importance sampling, this step aims to derive a weighted empirical approximation of $p(z_t \mid \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$. Simple calculation shows that

$$\begin{aligned} p&\left(z_t \mid \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) \\ &\propto p\left(z_t \mid x_{0:t}^{(i)}, y_{0:t-1}\right) p\left(y_t \mid x_t^{(i)}, z_t\right) \\ &\quad \times p\left(\tilde{x}_{t+1}^{(i)} \mid x_t^{(i)}, z_t\right). \end{aligned} \tag{24}$$

Analogously to step 3), choose $\pi(z_t \mid x_{0:t}^{(i)}, y_{0:t-1})$ as the importance function and note that $\tilde{p}_{N_z}(\bar{z}_t^{(i,j)} \mid x_{0:t}^{(i)}, y_{0:t-1})$ is an approximation of $p(\bar{z}_t^{(i,j)} \mid x_{0:t}^{(i)}, y_{0:t-1})$. Using importance sampling and also noting the definition of $r_t^{(i,j)}$ yields, for each $i = 1,\ldots,N_x$, a weighted empirical approximation of $p(z_t \mid \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ as

$$p_{N_z}\left(z_t \mid \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) = \sum_{j=1}^{N_z} q_t^{(i,j)} \delta\left(z_t - \bar{z}_t^{(i,j)}\right) \tag{25}$$

where $\tilde{x}_{0:t+1}^{(i)} \triangleq (x_{0:t}^{(i)}, \tilde{x}_{t+1}^{(i)})$ and $q_t^{(i,j)}$ is evaluated according to

$$q_t^{(i,j)} \propto p\left(y_t \,|\, x_t^{(i)}, \bar{z}_t^{(i,j)}\right) p\left(\tilde{x}_{t+1}^{(i)} \,|\, x_t^{(i)}, \bar{z}_t^{(i,j)}\right) r_t^{(i,j)};$$
$$\sum_{j=1}^{N_z} q_t^{(i,j)} = 1. \tag{26}$$

6) *Resampling of the particles* $\bar{z}_t^{(i,j)}, i = 1, \ldots, N_x, j = 1, \ldots, N_z$
By using resampling, this step aims to derive an unweighted empirical approximation of $p(z_t \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$. For each $i = 1, \ldots, N_x$, resample N_z times from the discrete distribution over $\{\bar{z}_t^{(i,j)}, j = 1, \ldots, N_z\}$ with probability mass $q_t^{(i,j)}$ associated with the element $\bar{z}_t^{(i,j)}$ to generate samples $\{z_t^{(i,j)}, j = 1, \ldots, N_z\}$, so that for any m,

$$\mathrm{P}\left\{z_t^{(i,m)} = \bar{z}_t^{(i,j)}\right\} = q_t^{(i,j)}. \tag{27}$$

As a result, it follows from (25) that for each $i = 1, \ldots, N_x$, an unweighted empirical approximation of $p(z_t \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ is as follows:

$$\tilde{p}_{N_z}\left(z_t \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) = \frac{1}{N_z} \sum_{j=1}^{N_z} \delta\left(z_t - z_t^{(i,j)}\right). \tag{28}$$

7) *Generation of particles* $\tilde{z}_{t+1}^{(i,j)}, i = 1, \ldots, N_x, j = 1, \ldots, N_z$
Assume that for each $\tilde{x}_{t+1}^{(i)}, i = 1, \ldots, N_x$, the particles $\tilde{z}_{t+1}^{(i,j)}, j = 1, \ldots, N_z$, are generated according to the proposal function $\pi(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$. By using importance sampling, we try to derive a weighted empirical approximation of $p(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$. First, choose $\pi(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ as the importance function. Then from (28) and

$$p(z_{t+1} \,|\, x_{0:t+1}, y_{0:t}) = \int p(z_t \,|\, x_{0:t+1}, y_{0:t}) p(z_{t+1} \,|\, x_{t:t+1}, z_t) dz_t \tag{29}$$

an approximation of $p(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}), i = 1, \ldots, N_x$, can be obtained as

$$\tilde{p}_{N_z}\left(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) = \frac{1}{N_z} \sum_{l=1}^{N_z} p\left(z_{t+1} \,|\, \tilde{x}_{t:t+1}^{(i)}, z_t^{(i,l)}\right). \tag{30}$$

Now using importance sampling yields that for $i = 1, \ldots, N_x$, a weighted empirical approximation of $p(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ as follows:

$$p_{N_z}\left(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) = \sum_{j=1}^{N_z} \tilde{r}_{t+1}^{(i,j)} \delta\left(z_{t+1} - \tilde{z}_{t+1}^{(i,j)}\right) \Big/ \sum_{l=1}^{N_z} \tilde{r}_{t+1}^{(i,l)} \tag{31}$$

where $\tilde{r}_{t+1}^{(i,j)}$ is evaluated according to

$$\tilde{r}_{t+1}^{(i,j)} = \tilde{p}_{N_z}\left(\tilde{z}_{t+1}^{(i,j)} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) \Big/ \pi\left(\tilde{z}_{t+1}^{(i,j)} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right). \tag{32}$$

Finally, from (31) and

$$p(y_{t+1} \,|\, x_{0:t+1}, y_{0:t}) = \int p(z_{t+1} \,|\, x_{0:t+1}, y_{0:t}) \times p(y_{t+1} \,|\, x_{t+1}, z_{t+1}) dz_{t+1} \tag{33}$$

an approximation

$$p_{N_z}(y_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$$

of

$$p(y_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$$

can be obtained as (11) with t replaced by $t+1$. In turn, it can be seen from (20), (22), step 4), step 7), and (32) that Inductive Assumptions 3.1 to 3.3 hold with t replaced by $t+1$. Hence, we have completed the derivation of the DPF by the induction principle.

B. Summary of the Filtering Algorithm

Initialization
Initialize the particles $\tilde{x}_0^{(i)} \sim p(x_0)$, $i = 1, \ldots, N_x$, and for each $\tilde{x}_0^{(i)}$, the particles $\tilde{z}_0^{(i,j)} \sim p(z_0 \,|\, \tilde{x}_0^{(i)}), j = 1, \ldots, N_z$. With a slight abuse of notation let $p(x_0) = p_{N_z}(x_0 \,|\, x_{0:-1}, y_{0:-1}) = \pi(x_0 \,|\, x_{0:-1}, y_{0:-1})$ and $p(z_0 \,|\, x_0) = \tilde{p}_{N_z}(z_0 \,|\, x_0, y_{0:-1}) = \pi(z_0 \,|\, x_0, y_{0:-1})$.
At each time instant $(t \geq 0)$

1) *Measurement update of* $x_{0:t}$ *based on* y_t
The importance weights $w_t^{(i)}, i = 1, \ldots, N_x$, are evaluated according to (18).
2) *Resampling of* $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)} \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$, $i = 1, \ldots, N_x$
According to (19), resample $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)} \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1, \ldots, N_x$, to generate samples $\{x_{0:t}^{(i)}, \bar{z}_t^{(i,1)}, r_t^{(i,1)} \ldots, \bar{z}_t^{(i,N_z)}, r_t^{(i,N_z)}\}, i = 1, \ldots, N_x$, where for $t \geq 0, \tilde{r}_t^{(i,j)}$ is defined according to (32).
3) *Measurement update of* z_t *based on* y_t
For $i = 1, \ldots, N_x$, the importance weights $\bar{q}_t^{(i,j)}$, $j = 1, \ldots, N_z$, are evaluated according to (23).
4) *Generation of particles* $\tilde{x}_{t+1}^{(i)}, i = 1, \ldots, N_x$
For each $i = 1, \ldots, N_x, \tilde{x}_{t+1}^{(i)}$ is generated according to the proposal function $\pi(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$.
5) *Measurement update of* z_t *based on* x_{t+1}
For $i = 1, \ldots, N_x$, the importance weights $q_t^{(i,j)}$, $j = 1, \ldots, N_z$, are evaluated according to (26).
6) *Resampling of the particles* $\bar{z}_t^{(i,j)}, i = 1, \ldots, N_x, j = 1, \ldots, N_z$
According to (27), for each $i = 1, \ldots, N_x$, resample the particles $\bar{z}_t^{(i,j)}, j = 1, \ldots, N_z$, to generate samples $z_t^{(i,j)}, j = 1, \ldots, N_z$.

7) *Generation of particles* $\tilde{z}_{t+1}^{(i,j)}, i = 1,\dots,N_x, j = 1,\dots,N_z$

For each $i = 1,\dots,N_x$, the particles $\tilde{z}_{t+1}^{(i,j)}, j = 1,\dots,N_z$, are generated according to the proposal function $\pi(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ where $\tilde{x}_{0:t+1}^{(i)} \triangleq (x_{0:t}^{(i)}, \tilde{x}_{t+1}^{(i)})$.

C. Implementation Issues

1) Two Resampling Steps: Unlike most of PFs in the literature, the DPF has two resampling steps, i.e., step 2) and step 6). Furthermore, the second resampling bears a parallel structure. This is because the particles $\bar{z}_t^{(i,j)}, i = 1,\dots,N_x, j = 1,\dots,N_z$, can be divided into N_x *independent* groups in terms of the index i. Therefore, the second resampling can be implemented in parallel.

In the implementation of the first resampling, it would be helpful to note the following points. For $i = 1,\dots,N_x$, $\{\tilde{r}_t^{(i,1)},\dots,\tilde{r}_t^{(i,N_z)}\}$ is associated with $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)},\dots,\tilde{z}_t^{(i,N_z)}\}$, according to the definition $\tilde{r}_t^{(i,j)} = \tilde{p}_{N_z}(\tilde{z}_t^{(i,j)} \,|\, \tilde{x}_{0:t}^{(i)}, y_{0:t-1})/\pi(\tilde{z}_t^{(i,j)} \,|\, \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$. Therefore, after resampling of $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)},\dots,\tilde{z}_t^{(i,N_z)}\}, i = 1,\dots,N_x, \{\tilde{r}_t^{(i,1)},\dots,\tilde{r}_t^{(i,N_z)}\}, i = 1,\dots,N_x$, should accordingly be resampled to obtain $\{r_t^{(i,1)},\dots, r_t^{(i,N_z)}\}, i = 1,\dots,N_x$. According to the definition of $\tilde{r}_t^{(i,j)}, r_t^{(i,j)}$ can be defined as $r_t^{(i,j)} = \tilde{p}_{N_z}(\bar{z}_t^{(i,j)} \,|\, x_{0:t}^{(i)}, y_{0:t-1})/\pi(\bar{z}_t^{(i,j)} \,|\, x_{0:t}^{(i)}, y_{0:t-1})$. As a result, $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)},\dots,\tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1,\dots,N_x$, is resampled in step 2). Moreover, since the particles $x_{0:t-1}^{(i)}, i = 1,\dots,N_x$, will not be used in the future, it is actually only necessary to resample $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)},\dots,\tilde{z}_t^{(i,N_z)}, r_t^{(i,N_z)}\}, i = 1,\dots,N_x$, to generate $\{x_t^{(i)}, \bar{z}_t^{(i,1)}, r_t^{(i,1)},\dots,\bar{z}_t^{(i,N_z)}, r_t^{(i,N_z)}\}, i = 1,\dots,N_x$.

2) Construction of the Proposal Functions: Like [15] where the "prior" is chosen as the proposal function, we try to choose $p(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$ and $p(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ as the proposal functions $\pi(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$ and $\pi(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$, respectively. Unlike [15], however, $p(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$ and $p(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ are usually unknown. Therefore, we need to construct approximations to $p(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$ and $p(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ such that the particles $\tilde{x}_{t+1}^{(i)}$ and $\tilde{z}_{t+1}^{(i,j)}$, $j = 1,\dots,N_z$, can be sampled from the approximations, respectively.

A convenient way to construct those approximations is given as follows. From (22), an approximation of $p(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$ can be obtained as

$$p_{N_z}\left(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t}\right) = \sum_{j=1}^{N_z} \bar{q}_t^{(i,j)} p\left(x_{t+1} \,|\, x_t^{(i)}, \bar{z}_t^{(i,j)}\right). \quad (34)$$

In turn, a further approximation of $p(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$ can be obtained as $\mathcal{N}(\bar{x}_{t+1}^{(i)}, \Sigma_{t+1}^{(i)})$ with $\bar{x}_{t+1}^{(i)}$ and $\Sigma_{t+1}^{(i)}$, respectively, the mean and the covariance of the discrete distribution over $\{\tilde{x}_{t+1}^{(i,j)}, j = 1,\dots,N_z\}$ with probability mass $\bar{q}_t^{(i,j)}$ associated with the element $\tilde{x}_{t+1}^{(i,j)} \sim p(x_{t+1} \,|\, x_t^{(i)}, \bar{z}_t^{(i,j)})$. Therefore, for $i = 1,\dots,N_x$, the particle $\tilde{x}_{t+1}^{(i)}$ can be generated from

$$\pi\left(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t}\right) = \mathcal{N}\left(\bar{x}_{t+1}^{(i)}, \Sigma_{t+1}^{(i)}\right). \quad (35)$$

On the other hand, from (28) and (29), an approximation $\tilde{p}_{N_z}(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ of $p(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ has already been given in (30). Then, it follows from (30) and the assumption that $p(z_{t+1} \,|\, x_{t:t+1}, z_t)$ is known that, for each $i = 1, \dots, N_x$, the particle $\tilde{z}_{t+1}^{(i,j)}, j = 1,\dots,N_z$ can be generated from

$$\pi\left(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) = \tilde{p}_{N_z}\left(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right). \quad (36)$$

Remark 3.2: From (25) and (29), another approximation of $p(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ can be obtained as

$$p_{N_z}\left(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) = \sum_{j=1}^{N_z} q_t^{(i,j)} p\left(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, \bar{z}_t^{(i,j)}, y_{0:t}\right). \quad (37)$$

This observation shows that the PF used to estimate $p(z_t \,|\, x_{0:t}^{(i)}, y_{0:t})$ is closely related to the so-called marginal PF [21]. A marginal PF would sample the particles $\tilde{z}_{t+1}^{(i,j)}$, $j = 1,\dots,N_z$, from

$$\pi\left(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}\right) = \sum_{j=1}^{N_z} q_t^{(i,j)} \pi\left(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, \bar{z}_t^{(i,j)}, y_{0:t}\right). \quad (38)$$

According to [21], sampling from (38) is precisely equivalent to first resample the particles $\bar{z}_t^{(i,j)}, i = 1,\dots,N_x, j = 1,\dots,N_z$ according to (27) and then sample the particle $\tilde{z}_{t+1}^{(i,j)}$ from $\pi(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, z_t^{(i,j)}, y_{0:t})$. If (37) is chosen as the proposal function, i.e., $\pi(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t}) = p_{N_z}(z_{t+1} \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ in the marginal PF, then sampling from (38) would be exactly the same as what we did. In addition, like the marginal PF, for each $i = 1,\dots,N_x$, the computational cost of $\tilde{r}_t^{(i,j)}, j = 1,\dots,N_z$, is $\mathcal{O}(N_z^2)$. However, this should not be a problem as a small N_z of particles are usually used to approximate $p(z_t \,|\, x_{0:t}^{(i)}, y_{0:t})$. Moreover, a couple of methods have been given in [21] to reduce this computational cost. On the other hand, if (36) is chosen as the proposal function, then $\tilde{r}_t^{(i,j)} = 1, i = 1,\dots,N_x, j = 1,\dots,N_z$. As a result, the resampling of $\{\tilde{r}_t^{(i,1)},\dots,\tilde{r}_t^{(i,N_z)}\}, i = 1,\dots,N_x$, is not needed and the computation of (11), (23) and (26) is simplified. $\Diamond$

Remark 3.3: If (3) has a special structure, then the construction of $\pi(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$ can become simpler. We mention two cases here. First, assume that the x-dynamics of (3) is independent of z, then $p(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t}) = p(x_{t+1} \,|\, x_t^{(i)})$ and thus we can choose $\pi(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t}) = p(x_{t+1} \,|\, x_t^{(i)})$. Systems with this special structure have been studied in the literature before, see, e.g., [11]. Second, assume that (3) takes the following form:

$$\begin{aligned} x_{t+1} &= f_t^x(x_t, z_t) + g_t^x(x_t) v_t^x \\ z_{t+1} &= f_t^z(x_t, z_t) + g_t^z(x_t, z_t) v_t^z \\ y_t &= h_t(x_t, z_t, e_t) \end{aligned} \quad (39)$$

where $f_t^x(\cdot), f_t^z(\cdot), g_t^x(\cdot), g_t^z(\cdot)$ and $h_t(\cdot)$ are known functions, v_t is assumed white and Gaussian distributed according to

$$v_t = \begin{bmatrix} v_t^x \\ v_t^z \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} Q_t^x & (Q_t^{zx})^T \\ Q_t^{zx} & Q_t^z \end{bmatrix}\right). \quad (40)$$

Then from (34), a further simplified approximation of $p(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t})$ can be obtained as $\mathcal{N}(\bar{x}_{t+1}^{(i)}, \Sigma_{t+1}^{(i)} + g_t^x(x_t^{(i)}) Q_t^x (g_t^x(x_t^{(i)}))^T)$ with $\bar{x}_{t+1}^{(i)}$ and $\Sigma_{t+1}^{(i)}$, respectively, the mean and the covariance of the discrete distribution $\{\tilde{x}_{t+1}^{(i,j)}, j = 1, \ldots, N_z\}$ with probability mass $\bar{q}_t^{(i,j)}$ associated with the element $\tilde{x}_{t+1}^{(i,j)} \sim p(x_{t+1} \,|\, x_t^{(i)}, \bar{z}_t^{(i,j)})$. For this special case, $\pi(x_{t+1} \,|\, x_{0:t}^{(i)}, y_{0:t}) = \mathcal{N}(\bar{x}_{t+1}^{(i)}, \Sigma_{t+1}^{(i)} + g_t^x(x_t^{(i)}) Q_t^x (g_t^x(x_t^{(i)}))^T)$. $\diamondsuit$

While we have assumed the proposal functions in the form of $\pi(x_t \,|\, x_{0:t-1}^{(i)}, y_{0:t-1})$ and $\pi(z_t \,|\, \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$, it is possible to choose proposal functions in more general forms. For example, in the importance sampling of (24), the proposal function $\pi(z_t \,|\, x_{0:t}^{(i)}, y_{0:t-1})$ can be replaced by another proposal function in the form of $\pi(z_t \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t-1})$. Moreover, this new proposal function, together with the two proposal functions $\pi(x_t \,|\, x_{0:t-1}^{(i)}, y_{0:t-1})$ and $\pi(z_t \,|\, \tilde{x}_{0:t}^{(i)}, y_{0:t-1})$ can be further made dependent on y_t. That is, in the importance sampling of (14), (21) and (24), the proposal functions in the form of $\pi(x_t \,|\, x_{0:t-1}^{(i)}, y_{0:t}), \pi(z_t \,|\, x_{0:t}^{(i)}, y_{0:t})$ and $\pi(z_t \,|\, \tilde{x}_{0:t+1}^{(i)}, y_{0:t})$ can be used, respectively. Due to the space limitation, we do not discuss this issue here and we refer the interested reader to, for example, [2], [7], [13], [23] for relevant discussions. Finally, we remaind that if these proposal functions in more general forms are used, then slight modification is needed to make to the DPF algorithm.

3) Computing the State Estimate: A common application of a PF is to compute the state estimate, i.e., the expected mean of the state. For (3), the state estimate of x_t and z_t are defined as

$$\bar{x}_t = \mathrm{E}_{p(x_t \,|\, y_{0:t})}(x_t), \bar{z}_t = \mathrm{E}_{p(z_t \,|\, y_{0:t})}(z_t). \tag{41}$$

Then the approximation of $\bar{x}_t$ and $\bar{z}_t$ can be computed in the following way for the DPF. Note from (17) that $p(x_t \,|\, y_{0:t})$ has an empirical approximation $p_{N_x}(x_t \,|\, y_{0:t}) = \sum_{i=1}^{N_x} w_t^{(i)} \delta(x_t - \tilde{x}_t^{(i)})$. Then, an approximation $\hat{x}_t$ of $\bar{x}_t$ can be calculated in the following way:

$$\hat{x}_t = \mathrm{E}_{p_{N_x}(x_t \,|\, y_{0:t})}(x_t) = \sum_{i=1}^{N_x} w_t^{(i)} \tilde{x}_t^{(i)}. \tag{42}$$

Analogously, note from (20) and (22) that $p(z_t \,|\, y_{0:t})$ has an empirical approximation $p_{N_x,N_z}(z_t \,|\, y_{0:t}) = (1/N_x) \sum_{i=1}^{N_x} \sum_{j=1}^{N_z} \bar{q}_t^{(i,j)} \delta(z_t - \bar{z}_t^{(i,j)})$. Then, an approximation $\hat{z}_t$ of $\bar{z}_t$ can be calculated as

$$\hat{z}_t = \mathrm{E}_{p_{N_x,N_z}(z_t \,|\, y_{0:t})}(z_t) = \frac{1}{N_x} \sum_{i=1}^{N_x} \sum_{j=1}^{N_z} \bar{q}_t^{(i,j)} \bar{z}_t^{(i,j)}. \tag{43}$$

IV. Discussion

In the DPF algorithm, the summation calculation in the normalizing factor (the denominator) of $w_t^{(i)}$ in (18) and the first resampling, i.e., the resampling of $\{\tilde{x}_t^{(i)}, \bar{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \bar{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1, \ldots, N_x$, are the only operations that cannot be implemented in parallel. Besides, the remaining operations including the second resampling, i.e., the resampling of the particles $\bar{z}_t^{(i,j)}, i = 1, \ldots, N_x, j = 1, \ldots, N_z$, can be divided into N_x *independent* parts in terms of the index i and can thus be implemented in parallel. In this sense, we say that the DPF has the advantage over the regular PF in that the DPF can increase the level of parallelism of the PF.

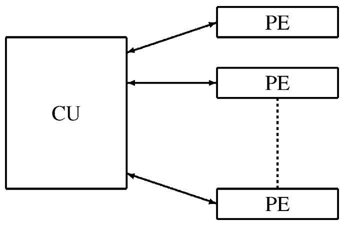

Fig. 3. The architecture of the PF with parallel structure. It consists of a central unit (CU) and a number of processing elements (PE) where the CU handles the operations that cannot be implemented in parallel and the PEs are run in parallel to deal with the operations that can be implemented in parallel.

The underlying idea of the DPF is different from that of the distributed PFs, such as the DRPA-PF and the DRNA-PF (see Section I), while they all have parallel structure. This difference results in a couple of unique features of the DPF in contrast with the distributed PFs, which identify the potential of the DPF in the application of PFs in real-time systems and the parallel implementation of PFs. In the remaining part of this section, we first briefly review the DRPA-PF and the DRNA-PF, and then we point out the unique features of the DPF. Finally, we show that there exist two ways to further increase the level of parallelism of the DPF.

Before the discussion, it should be noted that all the DPF, the DRPA-PF, and the DRNA-PF have the architecture as shown in Fig. 3.

A. Review of the DRPA-PF and the DRNA-PF [5]

Assume that the sample space contains M particles, where M is assumed to be the number of particles that is needed for the sampling importance resampling PF (SIR-PF) [13] or the bootstrap PF [15] to achieve a satisfactory performance for the filtering problem of (1). Then the sample space is divided into K disjoint strata where K is an integer and satisfies $1 \le K \le M$. Further assume that each stratum corresponds to a PE. Before resampling each PE thus has N particles where $N = M/K$ is an integer.

The sequence of operations performed by the kth PE and the CU for the DRPA-PF is shown in Fig. 4. The interresampling is performed on the CU and its function is to calculate the number of particles $N^{(k)}$ that will be produced after resampling for the kth PE. $E(N^{(k)})$ should be proportional to the weight $W^{(k)}$ of the kth PE which is defined as the sum of the weights of the particles inside the kth PE. In particular, $N^{(k)}$ is calculated using the residual systematic resampling (RSR) algorithm proposed in [4]. Once $N^{(k)}, k = 1, \ldots, K$, are known, resampling is performed inside the K PEs independently which is referred to as the intraresampling. Note that after resampling, for $k = 1, \ldots, K$, the kth PE has $N^{(k)}$ particles and $N^{(k)}$ is a random number because it depends on the overall distribution of the weights $W^{(k)}, k = 1, \ldots, K$. On the other hand, note that each PE is supposed to be responsible for processing N particles and to perform the same operations in time. Therefore after

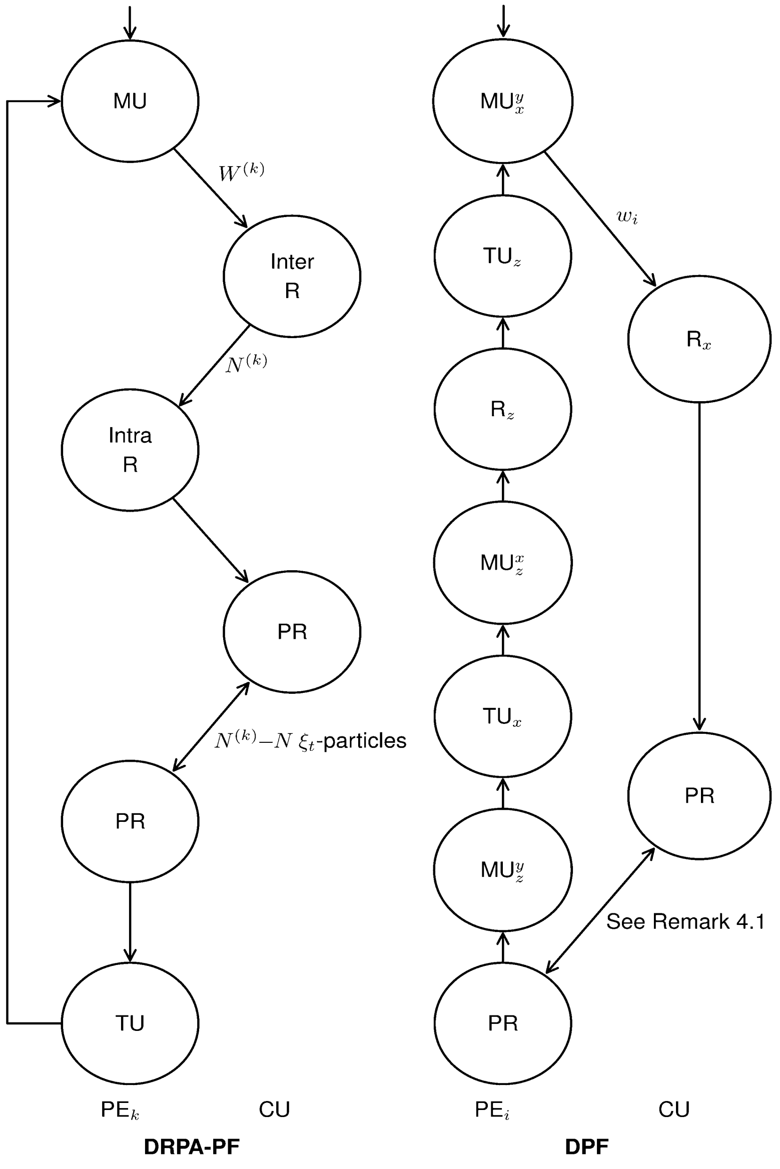

Fig. 4. Sequence of operations performed by the specified PE and the CU for the DRPA-PF and the DPF. The data transmitted between the specified PE and the CU and between different PEs are marked. For the DRPA-PF, the abbreviations are MU (measurement update of $\xi_{0:t}$ based on y_t), Inter R (inter resampling), Intra R (intra resampling), PR (particle routing), and TU (generation of particles $\xi_{t+1}^{(l)}, l = 1, \ldots, M$). For the DPF, the abbreviations are MU$_x^y$ (measurement update of $x_{0:t}$ based on y_t), R$_x$ (resampling of $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1, \ldots, N_x$), PR (particle routing), MU$_z^y$ (measurement update of z_t based on y_t), TU$_x$ (generation of particles $\tilde{x}_{t+1}^{(i)}, i = 1, \ldots, N_x$), MU$_z^x$ (measurement update of z_t based on x_{t+1}), R$_z$ (resampling of $\bar{z}_t^{(i,j)}, i = 1, \ldots, N_x, j = 1, \ldots, N_z$) and TU$_z$ (generation of particles $\tilde{z}_{t+1}^{(i,j)}, i = 1, \ldots, N_x, j = 1, \ldots, N_z$).

resampling, the exchange of particles among the PEs has to be performed such that each PE has N particles. This procedure is referred as particle routing and is conducted by the CU. According to [5], the DRPA-PF requires a complicated scheme for particle routing due to the proportional allocation rule. In order to shorten the delay caused by the complicated particle routing scheme in the DRPA-PF, the DRNA is in turn proposed in [5].

B. Unique Features of the DPF

The parallel structure of the DPF is created by decomposing the state space, differing from the parallel structure of the distributed PFs which is created by dividing the sample space. In the following, we will show that this difference results in a couple of unique features of the DPF.

Before the discussion, the sequence of operations performed by the ith PE and the CU for the DPF is shown in Fig. 4. The summation calculation in the normalizing factor of $w_t^{(i)}$ in (18) and the resampling of $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1, \ldots, N_x$, are performed on the CU. Assume that there are N_x PEs, and for each $i = 1, \ldots, N_x$, the ith PE handles the ith independent part of the remaining operations of the DPF. In particular, for each $i = 1, \ldots, N_x$, the ith PE handles the resampling of the particles $\bar{z}_t^{(i,j)}, j = 1, \ldots, N_z$. Therefore, the resampling of the particles $\bar{z}_t^{(i,j)}, i = 1, \ldots, N_x, j = 1, \ldots, N_z$, is run in parallel on the PEs.

Remark 4.1: In the particle routing of the DPF, the data transmitted between different PEs depends on the resampling result of

$\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1, \ldots, N_x$. More specifically, there will be no data transmitted through the ith PE if $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$ is selected only once in the resampling. Otherwise, the data transmitted through the ith PE will be $\{\tilde{x}_t^{(m)}, \tilde{z}_t^{(m,1)}, \tilde{r}_t^{(m,1)}, \ldots, \tilde{z}_t^{(m,N_z)}, \tilde{r}_t^{(m,N_z)}\}$ for some $m = 1, \ldots, N_x$. In particular, if $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$ is selected more than once, then $m = i$; if $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$ is not selected, then $m = 1, \ldots, N_x$ and $m \neq i$. Therefore, the data transmitted between any two PEs, say, the i_1th PE and the i_2th PE, will be either zero or $\{\tilde{x}_t^{(m)}, \tilde{z}_t^{(m,1)}, \tilde{r}_t^{(m,1)}, \ldots, \tilde{z}_t^{(m,N_z)}, \tilde{r}_t^{(m,N_z)}\}$ for either $m = i_1$ or $m = i_2$. $\Diamond$

In contrast to the DRPA-PF, the DPF allows a simpler particle routing scheme. For the DRPA-PF, since after resampling the kth PE has $N^{(k)}$ particles that is a random number, a complicate scheme has to be used for the DRPA-PF to make all K PEs has equally N particles. For the DPF, however, since after the resampling of $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$, $i = 1, \ldots, N_x$, all N_x PEs still have the same number of particles, then the DPF allows a simpler particle routing scheme and actually each PE can be treated as a single particle in the particle routing.

Given a PF with parallel structure, it works most efficiently if each PE handles the same number of particles. The efficiency of the DRPA-PF usually decreases, since the numbers of particles produced by each PE are not evenly but randomly distributed among the PEs. To be specific, note that the time used by the kth PE to produce $N^{(k)}$ particles, $k = 1, \ldots, K$, after resampling are usually not the same. This observation implies that the time used by the DRPA to produce the particles after resampling is determined by the k^*th PE that produces the largest $N^{(k^*)}$. Clearly, the more unevenly the numbers of particles produced by each PE are distributed, the more time the DRPA takes to produce the particles after resampling. Especially, in the extreme case that $N^{(k^*)} \gg N^{(k)}$ with $k = 1, \ldots, K$, and $k \neq k^*$, the efficiency of the DRPA-PF will be decreased significantly. However, for the DPF, the ith PE that handles the resampling of particles $\bar{z}_t^{(i,j)}$, $j = 1, \ldots, N_z$, produces, after resampling, the same number of particles $z_t^{(i,j)}, j = 1, \ldots, N_z$. Therefore, the DPF does not have the efficiency decrease problem of the DRPA-PF.

Besides, it will be verified by two numerical examples in the subsequent section that, the DPF has the potential to achieve in a shorter execution time the same level of performance as the bootstrap PF. However, the DRNA-PF actually trades PF performance for speed improvement [5], [22].

Remark 4.2: The ideal minimum execution time T_{ex} of the DRPA-PF and the DRNA-PF have been given in [5] and [22]. Analogously, we can also give the ideal minimum execution time of the DPF. Like [5], [22], the following assumptions are made. Consider an implementation with a pipelined processor. Assume that the execution time of the particle generation and the importance weights calculation of every particle is LT_{clk} where L is the latency due to the pipelining and T_{clk} is the clock period. Also assume that the resampling takes the same amount of time as the particle generation and the importance weights calculation. As a result, we have the ideal minimum execution time of the DPF as $T_{\text{ex}}^{\text{DPF}} = (2N_z + L + N_x + M_r + 1)T_{\text{clk}}$. Here, $2N_z$ represents the delay due to the resampling of the particles $\bar{z}_t^{(i,j)}, j = 1, \ldots, N_z$, and the corresponding particle generation and importance weights calculation, N_x represents the delay due to the resampling of $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1, \ldots, N_x, M_r$ is the delay due to the particle routing and the extra one T_{clk} is due to the particle generation and importance weight calculation of the particle $x_t^{(i)}$. $\Diamond$

C. Two Ways to Further Increase the Level of Parallelism of the DPF

The first resampling of the DPF, i.e., resampling of $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}, i = 1, \ldots, N_x$, is the major operation that cannot be implemented in parallel. If N_x is large, then this resampling will cause a large delay. In order to further increase the level of parallelism of the DPF and shorten the execution time, it is valuable to find ways to handle this problem.

Two possible ways will be given here. The first one is straightforward and is to employ any of the distributed resampling algorithms proposed in [5] and [22] to perform the first resampling of the DPF and besides, the remaining parts of the DPF stay unchanged. Nonetheless, we prefer the DRPA to the other distributed resampling algorithms, since it can produce the same result as the systematic resampling [20] according to [4].

Compared to the first way, the second way only applies to high dimensional (1) and it is based on an extension of the DPF. We have assumed above that the state ξ_t is decomposed into two parts according to (2). Actually, the DPF can be extended to handle the case where the state ξ_t is decomposed into more than two (at most n_ξ) parts. For illustration, we briefly consider the case where the state ξ_t is decomposed into three parts. The more general case can be studied using the induction principle.

For convenience, assume that the state x_t in (2) can be further decomposed into two parts, i.e.

$$x_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} \tag{44}$$

and accordingly, (3) can be further decomposed into the following form:

$$\begin{aligned} x_{1,t+1} &= f_t^{x_1}(x_{1,t}, x_{2,t}, z_t, v_t^{x_1}) \\ x_{2,t+1} &= f_t^{x_2}(x_{1,t}, x_{2,t}, z_t, v_t^{x_2}) \\ z_{t+1} &= f_t^{z}(x_{1,t}, x_{2,t}, z_t, v_t^{z}) \\ y_t &= h_t(x_{1,t}, x_{2,t}, z_t, e_t) \end{aligned} \tag{45}$$

where $x_{1,t} \in \mathbb{R}^{n_{x_1}}, x_{2,t} \in \mathbb{R}^{n_{x_2}}$, $v_t^x = [(v_t^{x_1})^T \ (v_t^{x_2})^T]^T$ with $v_t^{x_1} \in \mathbb{R}^{n_{v^{x_1}}}, v_t^{x_2} \in \mathbb{R}^{n_{v^{x_2}}}$. Assume that the probability densities $p(x_{1,0} \mid y_{-1}) = p(x_{1,0})$, $p(x_{2,0} \mid x_{1,0}, y_{-1}) = p(x_{2,0} \mid x_{1,0})$, $p(z_0 \mid x_{1,0}, x_{2,0}, y_{-1}) = p(z_0 \mid x_{1,0}, x_{2,0})$ and for $t \geq 0, p(x_{1,t+1} \mid x_{1,t}, x_{2,t}, z_t)$, $p(x_{2,t+1} \mid x_{1,t:t+1}, x_{2,t}, z_t)$, $p(z_t \mid x_{1,t:t+1}, x_{2,t:t+1}, z_t)$ and $p(y_t \mid x_{1,t}, x_{2,t}, z_t)$ are known.

The filtering problem of (45) can be split into three nested subproblems according to the following factorization:

$$\begin{aligned} p(z_t, x_{1,0:t}, x_{2,0:t} \mid y_{0:t}) &= p(z_t \mid x_{1,0:t}, x_{2,0:t}, y_{0:t}) \\ &\quad \times p(x_{2,0:t} \mid x_{1,0:t}, y_{0:t}) p(x_{1,0:t} \mid y_{0:t}) \end{aligned} \tag{46}$$

where for $i = 1, 2, x_{i,0:t} \triangleq \{x_{i,0}, \ldots, x_{i,t}\}$. It can be shown that the DPF can be extended to handle the filtering problem of (45) by using PFs to solve the three nested subproblems. Roughly speaking, a PF with N_{x_1} particles $(x_{1,0:t}^{(i)}, i = 1,\dots,N_{x_1})$ will be used to estimate $p(x_{1,0:t} \,|\, y_{0:t})$, and for each $i = 1,\dots,N_{x_1}$, a PF with N_{x_2} particles $(x_{2,0:t}^{(i,j)}, j = 1,\dots,N_{x_2})$ will be used to estimate $p(x_{2,0:t} \,|\, x_{1,0:t}^{(i)}, y_{0:t})$, and for each $i = 1,\dots,N_{x_1}$ and $j = 1,\dots,N_{x_2}$, a PF with N_z particles will be used to estimate $p(z_t \,|\, x_{1,0:t}^{(i)}, x_{2,0:t}^{(i,j)}, y_{0:t})$.

Similar to the DPF based on (4), the major operation that cannot be implemented in parallel in the DPF based on (46) is its first resampling, i.e., the resampling of N_{x_1} composite particles. If a satisfactory performance of the DPF based on (46) can be achieved with $N_{x_1} + N_{x_2} \ll N_x$, then the number of composite particles involved in the first resampling of the DPF will be reduced from N_x to N_{x_1}. Therefore, in this way the level of parallelism of the DPF is further increased. If the DPF is implemented in parallel, then the execution time of the DPF will be further decreased as well. However, it should be noted that $N_{x_1} \cdot N_{x_2}$ PEs are required to fully exploit the parallelism of the DPF based on (46). Due to the space limitation, we cannot include the extension of the DPF in this paper and instead we refer the reader to [10] for the details.

V. Numerical Examples

In this section we will test how the DPF performs on two examples. The simulations are performed using Matlab under the Linux operating system. The platform is a server consisting of eight Intel(R) Quad Xeon(R) CPUs (2.53 GHz).

A. Algorithms Tested

For the two examples, the bootstrap PF is implemented in the standard fashion, using different number of particles (M). The DPF is implemented according to Section III-B for different combinations of "x and z particles" (N_x and N_z). The DRPA-PF according to [5] is tested as well, using different number of PEs (K). The formulas of [5] has been closely followed, but the implementation is our own, and it is of course possible that it can be further trimmed. In addition, as suggested in [17], [20] systematic resampling is chosen as the resampling algorithm for all algorithms tested.

B. Performance Evaluation: Accuracy

In the tests, the performance of all algorithms are evaluated by 20000 Monte Carlo simulations. Basically, the accuracy of the state estimate is measured by the Root Mean Square Error (RMSE) between the true state and the state estimate. For example, the RMSE of $\hat{x}$ is defined as

$$\text{RMSE of } \hat{x} = \sqrt{\frac{1}{250}\sum_{t=1}^{250}\frac{1}{20000}\sum_{i=1}^{20000}\|x_t^i - \hat{x}_t^i\|^2} \tag{47}$$

where with a slight abuse of notation, x_t^i denotes the true state at time t for the ith simulation and $\hat{x}_t^i$ is the corresponding state estimate. It is also tested how well the RMSE reflects the accuracy of the estimated posterior densities (see Remark 5.1 for more details).

C. Performance Evaluation: Timing

One objective with the simulations is to assess the potential efficiency of the parallel implementation of the DPF. For that purpose, we record the following times

- T_{si}: This is the average execution time of the sequential implementation of a PF.
- T_{cp}: This is the average time used by the operations that cannot be implemented in parallel in a PF.
- T_{pi}: This is the potential execution time of parallel implementation of a PF. For the bootstrap PF with centralized resampling and the DPF, it is calculated according to $T_{\text{pi}} = T_{\text{cp}} + (T_{\text{si}} - T_{\text{cp}})/N_{\text{PE}}$ where N_{PE} is the number of processing elements. For the DPF, let $N_{\text{PE}} = N_x$. For the bootstrap PF with centralized resampling, let N_{PE} be the maximal N_x in the simulation of the corresponding example. Here, the bootstrap PF with centralized resampling means that besides the resampling, the remaining particle generation and importance weights calculation of the bootstrap PF are implemented in parallel. For the DRPA-PF, T_{pi} is calculated according to $T_{\text{pi}} = T_{\text{cp}} + T_{\text{mir}} + (T_{\text{si}} - T_{\text{cp}} - T_{\text{mir}})/N_{\text{PE}}$ where $N_{\text{PE}} = K$ and T_{mir} is the average maximal intra-resampling time for the DRPA-PF.

D. Performance Evaluation: Divergence Failures

The rate r_d is used to reveal how often a PF diverges in the 20 000 Monte Carlo simulations. The bootstrap PF and the DRPA-PF are said to diverge if their importance weights are all equal to zero in the simulation. The DPF is said to diverge if $w_t^{(i)}, i = 1,\dots,N_x$, are all equal to zero in the simulation. Once the divergence of a PF is detected, the PF will be rerun.

E. Sketch of the Simulation

For the two examples, the bootstrap PF using M particles is first implemented and its accuracy measured by the RMSE will be treated as the reference level. Then it is shown that the DPF using suitable N_x and N_z "x and z particles" can achieve the same level of accuracy. In turn, the DRPA-PF using M particles, but with different number of processing elements is also implemented. Finally, the bootstrap PF using $2M$ particles is implemented.

F. Two Dimensional Example

Consider the following two dimensional nonlinear system

$$\begin{aligned} x_{t+1} &= x_t + \frac{z_t}{1+z_t^2} + v_t^x \\ z_{t+1} &= x_t + 0.5z_t + \frac{25z_t}{1+z_t^2} + 8\cos(1.2t) + v_t^z \\ y_t &= \operatorname{atan}(x_t) + \frac{z_t^2}{20} + e_t \end{aligned} \tag{48}$$

where $[x_0 \;\; z_0]^T$ is assumed Gaussian distributed with $[x_0 \;\; z_0]^T \sim \mathcal{N}(0, I_{2\times 2}), v_t = [v_t^x \; v_t^z]^T$ and e_t are assumed white and Gaussian distributed with

$$v_t \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & 0.1 \\ 0.1 & 10 \end{bmatrix}\right), \quad \text{and } e_t \sim \mathcal{N}(0,1) \tag{49}$$

TABLE I
SIMULATION RESULT FOR SYSTEM (48) WITH (49)—"SEE SECTIONS V.B–V.D FOR EXPLANATIONS OF THE NUMBERS"

Case	RMSE of $[\hat{x}_t, \hat{z}_t]$	T_{si} (Sec)	T_{cp} (Sec)	T_{pi} (Sec)	r_{d}
Bootstrap PF, $M = 1000$	[2.0173, 2.3322]	**0.1891**	0.0313	0.0326	0.0155
DPF, $N_x = 100,\ N_z = 19$	[2.0104, 2.3497]	0.3545	**0.0168**	**0.0202**	0.0133
DPF, $N_x = 120,\ N_z = 19$	[1.9914, 2.3045]	0.3901	0.0176	0.0207	0.0175
DPF, $N_x = 110,\ N_z = 24$	[1.9907, 2.3154]	0.4127	0.0175	0.0211	0.0113
DPF, $N_x = 120,\ N_z = 24$	[1.9906, 2.3259]	0.4338	0.0179	0.0214	0.0076
DRPA-PF, $M = 1000,\ K = 40$	[2.0222, 2.3557]	0.6324	0.0671	0.0878	0.0124
DRPA-PF, $M = 1000,\ K = 25$	[2.0332, 2.4049]	0.4769	0.0565	0.0799	0.0124
Bootstrap PF, $M = 2000$	[**1.9714**, **2.2664**]	0.2579	0.0510	0.0528	**0.0059**

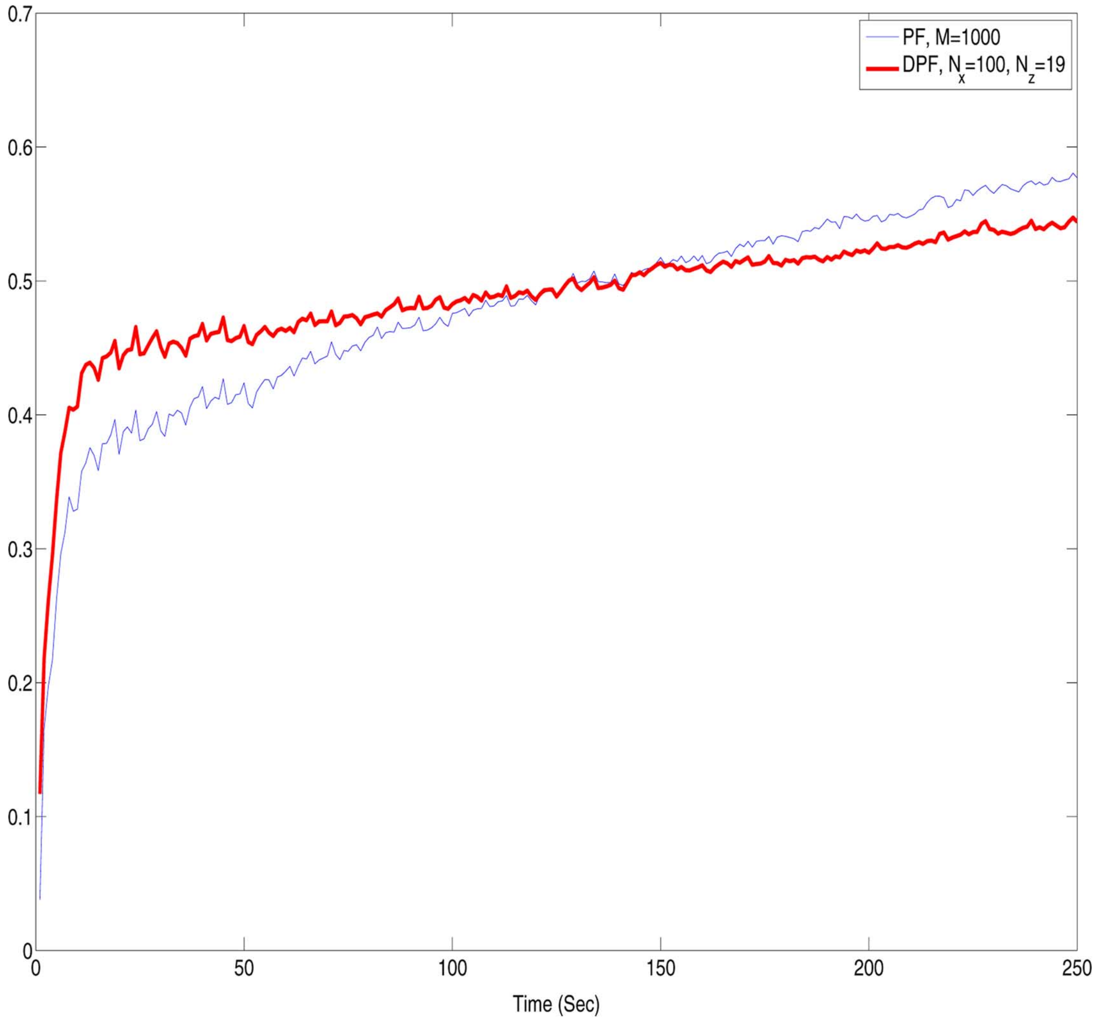

Fig. 5. The distance $\sum_{x_t} |F(x_t \mid y_{0:t}) - \tilde{F}(x_t \mid y_{0:t})|$ between the "true" posterior cumulative distribution $F(x_t \mid y_{0:t})$ and the empirical approximation of the posterior cumulative distribution $\tilde{F}(x_t \mid y_{0:t})$ obtained by using the bootstrap PF and the DPF (with $M = 1000$ and $N_x = 100, N_z = 19$ particles, respectively) as a function of time t (Thin curve: the bootstrap PF, Thick curve: the DPF). The result is an average over 20 000 simulations.

For the DPF, the proposal functions are chosen according to Remark 3.3 and (36). The simulation result over [1 250] is shown in Table I, from which it can be seen that the DPF has the potential to achieve in a shorter execution time the same level of accuracy as the bootstrap PF.

Remark 5.1: In Table I, the accuracy of the algorithms is measured entirely through the RMSE of the state estimate (47). Since the PF actually computes estimates of the posterior density $p(z_t, x_{0:t} \mid y_{0:t})$ one may discuss if this would be a more appropriate quantity to evaluate for comparison. Actually, the state estimates $\hat{x}_t$ could be quite accurate even though the estimate of the posterior density is poor. To evaluate that, we computed an accurate value of the true posterior density using the bootstrap PF with "many" (100 000) particles, and compared that with the estimates of the posterior densities using the bootstrap PF and the DPF with fewer ($M = 1000$ and $N_x = 100, N_z = 19$) particles. To avoid smoothing issues for the empirical approximations of the posterior densities, we made the comparisons for the posterior cumulative distributions (empirical approximations of the posterior cumulative distributions). The result is shown in Fig. 5. Moreover, let $\bar{x}_t^i$ denote the true mean of x_t^i, then (47) with x_t^i replaced by $\bar{x}_t^i$ is also calculated: it is 0.7239 for the

TABLE II
SIMULATION RESULT FOR (50) WITH (51)—"SEE SECTIONS V.B–V.D FOR EXPLANATIONS OF THE NUMBERS"

Case	RMSE of $[\hat{x}_{1,t}, \hat{x}_{2,t}, \hat{z}_{1,t}, \hat{z}_{2,t}]$	T_{si} (Sec)	T_{cp} (Sec)	T_{pi} (Sec)	r_{d}
Bootstrap PF, $M = 1500$	[1.1566, 1.3494, 2.0111, 2.8241]	**0.2022**	0.0316	0.0339	0.0072
DPF, $N_x = 50$, $N_z = 29$	[1.1707, 1.3678, 2.0485, 2.9383]	0.2366	**0.0145**	**0.0190**	0.0109
DPF, $N_x = 60$, $N_z = 49$	[1.1633, 1.3569, 1.9879, 2.7911]	0.3255	0.0160	0.0212	0.0039
DPF, $N_x = 75$, $N_z = 39$	[1.1610, 1.3537, **1.9794**, **2.7547**]	0.3309	0.0164	0.0206	0.0040
DRPA-PF, $M = 1500$, $K = 30$	[1.1564, 1.3490, 2.0028, 2.7894]	0.5083	0.0470	0.0669	0.0084
DRPA-PF, $M = 1500$, $K = 50$	[1.1566, 1.3495, 2.0148, 2.8302]	0.6951	0.0607	0.0780	0.0107
Bootstrap PF, $M = 3000$	[**1.1518**, **1.3419**, **1.9794**, 2.7601]	0.3105	0.0539	0.0573	**0.0021**

bootstrap PF with $M = 1000$, and 0.6851 for the DPF with N_x=100 and $N_z = 19$. From the above simulation results, we see that the DPF is at least as good as the bootstrap PF in approximating the posterior cumulative distribution. We conclude that the RMSE (47) gives a fair evaluation of the accuracy of the state estimate produced by the DPF for (48) with (49). $\diamondsuit$

G. Four Dimensional Example

Consider the following four dimensional nonlinear system

$$\begin{aligned}
x_{1,t+1} &= 0.5x_{1,t} + 8\sin(t) + v_t^{x_1}\\
x_{2,t+1} &= 0.4x_{1,t} + 0.5x_{2,t} + v_t^{x_2}\\
z_{1,t+1} &= z_{1,t} + \frac{z_{2,t}}{1+z_{2,t}^2} + v_t^{z_1}\\
z_{2,t+1} &= z_{1,t} + 0.5z_{2,t}\\
&\quad + \frac{25z_{2,t}}{1+z_{2,t}^2} + 8\cos(1.2t) + v_t^{z_2}\\
y_t &= \frac{x_{1,t}+x_{2,t}}{1+x_{1,t}^2} + \operatorname{atan}(z_{1,t}) + \frac{z_{2,t}^2}{20} + e_t \qquad (50)
\end{aligned}$$

where $x_t = [x_{1,t}\ x_{2,t}]^T$ and $z_t = [z_{1,t}\ z_{2,t}]^T$. $[x_0^T\ z_0^T]^T$ is assumed Gaussian distributed with $[x_0^T\ z_0^T]^T \sim \mathcal{N}(0, I_{4\times4})$, $v_t = [(v_t^x)^T\ (v_t^z)^T]^T$ with $v_t^x = [v_t^{x_1}\ v_t^{x_2}]^T$ and $v_t^z = [v_t^{z_1}\ v_t^{z_2}]^T$, and e_t are assumed white and Gaussian distributed with

$$v_t \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0.1\\ 0 & 0 & 0.1 & 10 \end{bmatrix}\right), \quad \text{and } e_t \sim \mathcal{N}(0,1). \qquad (51)$$

Since the "x-dynamics" does not depend on z_t, we let $\pi(x_{t+1} \mid x_{0:t}^{(i)}, y_{0:t}) = p(x_{t+1} \mid x_t^{(i)})$ and choose the other proposal function according to (36). The simulation result over [1 150] is shown in Table II, from which it can be seen that the DPF has the potential to achieve in a shorter execution time the same level of accuracy as the bootstrap PF.

H. Summary

Regarding the accuracy, comparison of the first part of the RMSE column in Tables I and II shows that with suitably chosen N_x and N_z, the DPF achieves the same level of accuracy as the bootstrap PF. On the other hand, with comparable number of particles (it is fair to compare M with $N_x(N_z+1)$) the accuracy is not much worse for the DPF than for the bootstrap PF. In fact, in Table II the DPF even performs slightly better than the PF for some of the states (no statistical significance), illustrating that allocating points as in Fig. 1(c) could actually be beneficial for some systems.

Regarding timing, comparison of the T_{si} and T_{pi} column in Tables I and II shows that the execution time of the DPF can be shortened significantly if the DPF can be implemented in parallel. Moreover, the DPF has a potential to offer better accuracy in shorter execution time. In particular, the T_{pi} of the DPF is less than that of the bootstrap PF with centralized resampling. It is valuable to note that the T_{pi} of the DPF is even smaller than the T_{cp} of the bootstrap PF with centralized resampling, which is actually the lower bound of the execution time of the bootstrap PF with centralized resampling. In addition, as discussed in Section IV.C, the T_{pi} of the DPF can be further shortened by using any one of the distributed resampling algorithms to perform the resampling of $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, \ldots, \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$, $i = 1, \ldots, N_x$. As a result, it is fair to say that the parallel implementation of the DPF has the potential to shorten the execution time of the PF.

VI. CONCLUSION

In this paper we have proposed a new way of allocating the particles in particle filtering. By first decomposing the state into two parts, the DPF splits the filtering problem into two nested subproblems and then handles the two nested subproblems using PFs. Returning to the questions in the intuitive preview in Section II.A, we have seen that the DPF can produce results of not much worse accuracy compared to the regular PF, with comparable number of particles. We have also seen that the structure gives a potential for more efficient calculations. The advantage of the DPF over the regular PF lies in that the DPF can increase the level of parallelism of the PF in the sense that part of the resampling has a parallel structure. The parallel structure of the DPF is created by decomposing the state space, differing from the parallel structure of the distributed PFs which is created by dividing the sample space. This difference results in a couple of unique features of the DPF in contrast with the existing distributed PFs. As a result, we believe that the DPF is a new option for parallel implementation of PFs and the application of PFs in real-time systems.

An interesting topic for future work is to study how to decompose the state given a high dimensional system such that the execution time of the parallel implementation of the DPF can be maximally reduced. Another interesting topic is to test the DPF in different types of parallel hardware, for example, graphical processing units (GPU) and field-programmable gate arrays (FPGA).

A further topic of future investigations is to generalize the line pattern in Fig. 1(c) to other lines and curves that may pick up useful shapes in the posterior densities to be estimated. This essentially involves a change of state variables before the DPF is applied.

Acknowledgment

The authors would like to thank the reviewers for their useful comments and constructive suggestions, and the associate editor for his time handling the paper. The first author would also like to thank Umut Orguner for his continuous support and helpful discussions.

This work was supported by the Strategic Research Center MOVIII, funded by the Swedish Foundation for Strategic Research, SSF, and CADICS, a Linnaeus center funded by the Swedish Research Council.

References

[1] C. Andrieu and A. Doucet, "Particle filtering for partially observed Gaussian state space models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 64, pp. 827–836, 2002.

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

[3] N. Bergman, "Recursive Bayesian Estimation: Navigation and Tracking Applications," Ph.D. Thesis No 579, Linköping Studies in Science and Technology, SE-581 83, Linköping, Sweden, May 1999.

[4] M. Bolić, P. Djurić, and S. Hong, "Resampling algorithms for particle filters: A computational complexity perspective," *EURASIP J. Appl. Signal Process.*, vol. 15, pp. 2267–2277, 2004.

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

[6] R. S. Bucy and K. D. Senne, "Digital synthesis on nonlinear filters," *Automatica*, vol. 7, pp. 287–298, 1971.

[7] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," *IEE Proc.—Radar, Sonar Navig.*, vol. 146, no. 1, pp. 2–7, 1999.

[8] G. Casella and C. P. Robert, "Rao-Blackwellisation of sampling schemes," *Biometrika*, vol. 83, pp. 81–94, 1996.

[9] R. Chen and J. Liu, "Mixture Kalman filters," *J. Royal Statist. Soc.: Series B (Statist. Methodol.)*, vol. 62, pp. 493–508, 2000.

[10] T. Chen, T. B. Schön, H. Ohlsson, and L. Ljung, An Extension of the Dencentralized Particle Filter with Arbitrary State Partitioning Autom. Contr. Div., Linköping Univ., Tech. Rep. No. 2930, 2010.

[11] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," in *Proc. 16th Conf. Uncertainty in Artif. Intell.*, 2000, pp. 176–183.

[12] A. Doucet, N. D. Freitas, and N. Gordonn*, Sequential Monte Carlo Methods in Practice*. New York: Springer, 2001.

[13] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statist. Comput.*, vol. 10, pp. 197–208, 2000.

[14] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *Handbook of Nonlinear Filtering*, D. Crisan and B. Rozovsky, Eds. Oxford, U.K.: Oxford Univ. Press, 2009.

[15] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Process. IEE Proc. F*, vol. 140, no. 2, pp. 107–113, 1993.

[16] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 425–437, 2002.

[17] J. D. Hol, T. B. Schön, and F. Gustafsson, "On resampling algorithms for particle filters," in *Proc. IEEE Nonlin. Statist. Signal Process. Workshop*, 2006, pp. 79–82.

[18] X. Hu, T. B. Schön, and L. Ljung, "A basic convergence result for particle filtering," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1337–1348, 2008.

[19] R. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[20] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *J. Comput. Graph. Statist.*, vol. 5, no. 1, pp. 1–25, 1996.

[21] M. Klass, N. de Freitas, and A. Doucet, "Towards practical N^2 Monte Carlo: The marginal particle filter," in *Proc. Uncertainty in Artif. Intell.*, 2005.

[22] J. Míguez, "Analysis of parallelizable resampling algorithms for particle filtering," *Signal Process.*, vol. 87, pp. 3155–3174, 2007.

[23] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *J. Amer. Statist. Assoc.*, vol. 94, no. 446, pp. 590–599, 1999.

[24] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE Acoust., Speech, Signal Process. Mag.*, pp. 4–16, 1986.

[25] T. B. Schön, F. Gustafsson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Trans. Signal Process.*, vol. 53, pp. 2279–2289, 2005.

Tianshi Chen (M'10) received the M.S. degree in 2005 from the Harbin Institute of Technology and the Ph.D. degree in December 2008 from the Chinese University of Hong Kong.

Since April 2009, he has been a Postdoctoral Researcher with the Division of Automatic Control, Department of Electrical Engineering, Linköping University, Linköping, Sweden. His research interests include nonlinear control theory and applications, system identification, and statistical signal processing.

Thomas B. Schön (M'07) was born in Sweden in 1977. He received the B.Sc. degree in business administration and economics in February 2001, the M.Sc. degree in applied physics and electrical engineering in September 2001, and the Ph.D. degree in automatic control in February 2006, all from Linköping University, Linköping, Sweden.

He has held visiting positions with the University of Cambridge, U.K., and the University of Newcastle, Australia. His research interests are mainly within the areas of signal processing, system identification and machine learning, with applications to the automotive and the aerospace industry. He is currently an Associate Professor at Linköping University.

Henrik Ohlsson (S'00–M'04) was born in Sweden in 1981. He received the M.Sc. degree in applied physics and electrical engineering in October 2006 and the Licentiate degree in automatic control in December 2008, both from Linköping University, Sweden.

He has held visiting positions with the University of Cambridge, U.K., and the University of Massachusetts, Amherst. His research interests are mainly within the areas of system identification and machine learning. He is currently a Ph.D. student at Linköping University.

Lennart Ljung (S'74–M'75–SM'83–F'85) received the Ph.D. degree in automatic control from Lund Institute of Technology in 1974.

Since 1976, he has been a Professor of Automatic Control in Linköping, Sweden, and is currently Director of the Stratetic Research Center "Modeling, Visualization and Information Integration" (MOVIII). He has held visiting positions with Stanford University, Stanford, CA, and the Massachusetts Institute of Technology (MIT), Cambridge. He has written several books on system identification and estimation.

Dr. Ljung is an IFAC Fellow and an IFAC Advisor. He is a member of the Royal Swedish Academy of Sciences (KVA), the Royal Swedish Academy of Engineering Sciences (IVA), an Honorary Member of the Hungarian Academy of Engineering, and a Foreign Associate of the U.S. National Academy of Engineering (NAE). He has received honorary doctorates from the Baltic State Technical University, St. Petersburg; Uppsala University, Sweden; the Technical University of Troyes, France; the Catholic University of Leuven, Belgium; and Helsinki University of Technology, Finland. In 2002, he received the Quazza Medal from IFAC. He received the Hendrik W. Bode Lecture Prize from the IEEE Control Systems Society in 2003 and the IEEE Control Systems Award for 2007.