Decentralization of Particle Filters Using Arbitrary State Decomposition

Tianshi Chen, Thomas B. Schön, Henrik Ohlsson and Lennart Ljung

***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 sub-problems and then handles the two nested sub-problems using PFs. The DPF has an 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 thus can 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 from a numerical example indicates that the DPF has a potential to achieve the same level of performance as the regular PF, in a shorter execution time.**

I. Introduction

In this paper, we study the filtering problem of 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 $p(\xi_t|y_{0:t})$ where $y_{0:t} \triangleq \{y_0, ..., y_t\}$. 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 [17] which assumes that system (1) has a linear-Gaussian structure. A class of powerful numerical algorithms for 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 [14], 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, 11, 13, 15] 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 system (1) into two nested sub-problems and then handles the two nested sub-problems 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 [12]. 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 [12]. 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, 19]. 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 [19]. Based on the introduced distributed resampling algorithms, a couple of distributed PFs have been further proposed in [5, 19], 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 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 suffer from the efficiency decrease

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.

T. Chen, T. B. Schön, H. Ohlsson and L. Ljung are with Division of Automatic Control, Department of Electrical Engineering, Linköping University, Linköping, Sweden {tschen,schon,ohlsson,ljung}@isy.liu.se

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 number of particles produced by each processing element is not evenly, but randomly distributed among the processing elements. Third, it will be verified by a numerical example that, the DPF has the potential to achieve the same level of performance as the bootstrap PF, in a shorter execution time. In contrast, the DRNA-PF actually trades the PF performance for the speed improvement [5]. Moreover, 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, 19] 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.

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 a 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.

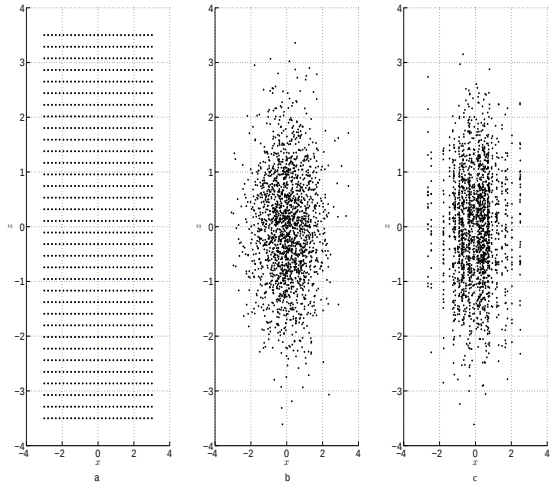

Fig. 1. Patterns for points where the posterior densities are computed/estimated. a. A fixed regular grid using in point-mass filter. b. Randomly allocated points following the regular PF equations. c. Points randomly allocated to vertical parallel lines (which themselves are randomly located).

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 system (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 system (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 $v_t^x \in \mathbb{R}^{n_{v^x}}$ and $v_t^z \in \mathbb{R}^{n_{v^z}}$. 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 $x_{0:t} \triangleq \{x_0, ..., x_t\}$, the filtering problem (4) can be split into two nested sub-problems:

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

In writing down the conceptual solution to the filtering problem (4), it is clear that the two sub-problems are nested. Since there is in general no analytic solution to the filtering problem (4), a numerical algorithm, i.e., the DPF is introduced to recursively provide 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 sub-problems using PFs. Roughly speaking, the DPF solves the first sub-problem using a PF with N_x

particles ($x_{0:t}^{(i)}$, $i = 1, ..., N_x$) to estimate $p(x_{0:t}|y_{0:t})$. Then the DPF handles the second sub-problem using N_x PFs with N_z particles each to estimate $p(z_t|x_{0:t}^{(i)}, y_{0:t})$, $i = 1, ..., N_x$. As a result of the nested nature of the two sub-problems, 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 sub-problems is not new. Actually, it has been used in the Rao-Blackwellized PF (RBPF); see, e.g, [1, 7, 8, 10, 12, 21]. However, the RBPF imposes a certain *tractable* substructure assumption on the system considered and hence solves one of the sub-problem with a number of optimal filters, such as the Kalman filter [17] or the HMM filter [20]. In particular, the filtering problem (4) has been previously studied in [21] where system (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 system (3) is marginalized out using the Kalman filter. However, since there is no tractable substructure assumption made on system (3) in this paper, no part of the state ξ_t is analytically tractable as was the case in [1, 7, 8, 10, 12, 21]. ♢

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 $\Pr(A)$ denote the probability of the event A. For convenience, for each $i = 1, ..., N$, $\alpha_i = \beta_i / \sum_{j=1}^{N} \beta_j$ is denoted by

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

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

III. Decentralized Particle filter

A. Filtering algorithm

Due to the space limitation, we have to skip the derivation and directly introduce the DPF algorithm as follows.

Initialization

Initialize the particles $\tilde{x}_0^{(i)} \sim p(x_0)$, $i = 1, ..., N_x$, and for each $\tilde{x}_0^{(i)}$, the particles $\tilde{z}_0^{(i,j)} \sim p(z_0|\tilde{x}_0^{(i)})$, $j = 1, ..., N_z$. 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 $w_t^{(i)}$, $i = 1, ..., N_x$, are evaluated according to

$$w_t^{(i)} \propto \frac{p_{N_z}(y_t|\tilde{x}_{0:t}^{(i)}, y_{0:t-1}) p_{N_z}(\tilde{x}_t^{(i)}|x_{0:t-1}^{(i)}, y_{0:t-1})}{\pi(\tilde{x}_t^{(i)}|x_{0:t-1}^{(i)}, y_{0:t-1})}; \qquad \sum_{i=1}^{N_x} w_t^{(i)} = 1 \tag{6}$$

where

$$p_{N_z}(y_t|\tilde{x}_{0:t}^{(i)}, y_{0:t-1}) = \sum_{j=1}^{N_z} \tilde{r}_t^{(i,j)} p(y_t|\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,j)}) / \sum_{l=1}^{N_z} \tilde{r}_t^{(i,l)} \tag{7}$$

$$\tilde{r}_t^{(i,j)} = \frac{\tilde{p}_{N_z}(\tilde{z}_t^{(i,j)}|\tilde{x}_{0:t}^{(i)}, y_{0:t-1})}{\pi(\tilde{z}_t^{(i,j)}|\tilde{x}_{0:t}^{(i)}, y_{0:t-1})} \tag{8}$$

and for $t \geq 1$,

$$p_{N_z}(\tilde{x}_t^{(i)}|x_{0:t-1}^{(i)}, y_{0:t-1}) = \sum_{j=1}^{N_z} \bar{q}_{t-1}^{(i,j)} p(\tilde{x}_t^{(i)}|x_{t-1}^{(i)}, \bar{z}_{t-1}^{(i,j)}) \tag{9}$$

$$\tilde{p}_{N_z}(z_t|\tilde{x}_{0:t}^{(i)}, y_{0:t-1}) = \frac{1}{N_z} \sum_{l=1}^{N_z} p(z_t|\tilde{x}_{t-1:t}^{(i)}, z_{t-1}^{(i,l)}) \tag{10}$$

2) *Resampling of* $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)} ..., \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$, $i = 1, ..., N_x$
Resample $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)} ..., \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$, $i = 1, ..., N_x$, to generate samples $\{x_{0:t}^{(i)}, \bar{z}_t^{(i,1)}, r_t^{(i,1)} ..., \bar{z}_t^{(i,N_z)}, r_t^{(i,N_z)}\}$, $i = 1, ..., N_x$, according to

$$\Pr\{\{x_{0:t}^{(m)}, \bar{z}_t^{(m,1)}, r_t^{(m,1)}, ..., \bar{z}_t^{(m,N_z)}, r_t^{(m,N_z)}\} = \{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, ..., \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}\} = w_t^{(i)} \tag{11}$$

3) *Measurement update of z_t based on y_t*
For $i = 1, ..., N_x$, the importance weights $\bar{q}_t^{(i,j)}$, $j = 1, ..., N_z$, are evaluated according to

$$\bar{q}_t^{(i,j)} \propto p(y_t|x_t^{(i)}, \bar{z}_t^{(i,j)}) r_t^{(i,j)}; \qquad \sum_{j=1}^{N_z} \bar{q}_t^{(i,j)} = 1 \tag{12}$$

4) *Generation of particles $\tilde{x}_{t+1}^{(i)}$, $i = 1, ..., N_x$*
For each $i = 1, ..., N_x$, $\tilde{x}_{t+1}^{(i)}$ 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 $i = 1, ..., N_x$, the importance weights $q_t^{(i,j)}$, $j = 1, ..., N_z$, are evaluated according to

$$q_t^{(i,j)} \propto p(y_t|x_t^{(i)}, \bar{z}_t^{(i,j)}) p(\tilde{x}_{t+1}^{(i)}|x_t^{(i)}, \bar{z}_t^{(i,j)}) r_t^{(i,j)}; \qquad \sum_{j=1}^{N_z} q_t^{(i,j)} = 1 \tag{13}$$

6) *Resampling of the particles $\bar{z}_t^{(i,j)}$, $i = 1, ..., N_x$, $j = 1, ..., N_z$*
For each $i = 1, ..., N_x$, resample the particles $\bar{z}_t^{(i,j)}$, $j = 1, ..., N_z$, to generate samples $z_t^{(i,j)}, j = 1, ..., N_z$, according to

$$\Pr\{z_t^{(i,m)} = \bar{z}_t^{(i,j)}\} = q_t^{(i,j)} \tag{14}$$

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

B. 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 $\bar{z}_t^{(i,j)}$, $i = 1, ..., N_x$, $j = 1, ..., N_z$, 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 $i = 1, ..., N_x$, $\{\tilde{r}_t^{(i,1)}, ..., \tilde{r}_t^{(i,N_z)}\}$ is associated with $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, ..., \tilde{z}_t^{(i,N_z)}\}$, 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 $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, ..., \tilde{z}_t^{(i,N_z)}\}$, $i = 1, ..., N_x$, $\{\tilde{r}_t^{(i,1)}, ..., \tilde{r}_t^{(i,N_z)}\}$, $i = 1, ..., N_x$, should accordingly be resampled to obtain $\{r_t^{(i,1)}, ..., r_t^{(i,N_z)}\}$, $i = 1, ..., N_x$. 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, $\{\tilde{x}_{0:t}^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, ..., \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$, $i = 1, ..., N_x$, is resampled in step 2). Moreover, since the particles $x_{0:t-1}^{(i)}, i = 1, ..., N_x$, will not be used in the future, it is actually only necessary to resample $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, ..., \tilde{z}_t^{(i,N_z)}, r_t^{(i,N_z)}\}$, $i = 1, ..., N_x$, to generate $\{x_t^{(i)}, \bar{z}_t^{(i,1)}, r_t^{(i,1)}, ..., \bar{z}_t^{(i,N_z)}, r_t^{(i,N_z)}\}, i = 1, ..., N_x$.

2) Construction of the proposal functions: Like [14] 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 [14], 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 $\tilde{z}_{t+1}^{(i,j)}$, $j = 1, ..., N_z$, can be sampled from the approximations, respectively.

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

$$p_{N_z}(x_{t+1}|x_{0:t}^{(i)}, y_{0:t}) = \sum_{j=1}^{N_z} \bar{q}_t^{(i,j)} p(x_{t+1}|x_t^{(i)}, \bar{z}_t^{(i,j)}) \quad (15)$$

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 covariance of the discrete distribution over $\{\tilde{x}_{t+1}^{(i,j)}, j = 1, ..., N_z\}$ 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 $i = 1, ..., N_x$, the particle $\tilde{x}_{t+1}^{(i)}$ can be generated from

$$\pi(x_{t+1}|x_{0:t}^{(i)}, y_{0:t}) = \mathcal{N}(\bar{x}_{t+1}^{(i)}, \Sigma_{t+1}^{(i)}) \quad (16)$$

On the other hand, 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 (10). Then, it follows from (10) and the assumption that $p(z_{t+1}|x_{t:t+1}, z_t)$ is known that, for each $i = 1, ..., N_x$, the particle $\tilde{z}_{t+1}^{(i,j)}$, $j = 1, ..., N_z$ can be generated from

$$\pi(z_{t+1}|\tilde{x}_{0:t+1}^{(i)}, y_{0:t}) = \tilde{p}_{N_z}(z_{t+1}|\tilde{x}_{0:t+1}^{(i)}, y_{0:t}) \quad (17)$$

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

$$\bar{x}_t = \mathrm{E}_{p(x_t|y_{0:t})}(x_t), \ \bar{z}_t = \mathrm{E}_{p(z_t|y_{0:t})}(z_t) \quad (18)$$

Then the approximation of $\bar{x}_t$ and $\bar{z}_t$ can be computed in the following way for the DPF. Note 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

$$\hat{x}_t = \mathrm{E}_{p_{N_x}(x_t|y_{0:t})}(x_t) = \sum_{i=1}^{N_x} w_t^{(i)} \tilde{x}_t^{(i)} \quad (19)$$

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

$$\hat{z}_t = \mathrm{E}_{p_{N_x,N_z}(z_t|y_{0:t})}(z_t) = \frac{1}{N_x} \sum_{i=1}^{N_x} \sum_{j=1}^{N_z} \bar{q}_t^{(i,j)} \bar{z}_t^{(i,j)} \quad (20)$$

IV. Discussion

A. 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.

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

Given a PF with parallel structure, it works most efficiently if each processing element handles the same number of particles. The efficiency of the DRPA-PF usually decreases, since the number of particles produced by each processing element is not evenly, but randomly distributed among the processing elements. To be specific, note that the time used by the kth processing element to produce $N^{(k)}$ particles, $k = 1, ..., K$, after resampling is 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 processing element that produces the largest $N^{(k^*)}$. Clearly, the more unevenly the numbers of particles produced by each processing element 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 $k = 1, ..., K$, and $k \neq k^*$, the efficiency of the DRPA-PF will be decreased significantly. However, for the DPF, the ith processing element that handles the resampling of particles $\tilde{z}_t^{(i,j)}$, $j = 1, ..., N_z$, produces, after resampling, the same number of particles $z_t^{(i,j)}$, $j = 1, ..., N_z$. Therefore, the DPF does not suffer from the efficiency decrease problem of the DRPA-PF.

Moreover, as will be verified by a numerical example in the subsequent section, the DPF has the potential to achieve the same level of performance as the bootstrap PF, in a shorter execution time. However, the DRNA-PF actually trades PF performance for speed improvement [5, 19].

B. Two ways to further increase the level of parallelism of the DPF

The first resampling of the DPF, i.e., resampling of $\{\tilde{x}_t^{(i)}, \tilde{z}_t^{(i,1)}, \tilde{r}_t^{(i,1)}, ..., \tilde{z}_t^{(i,N_z)}, \tilde{r}_t^{(i,N_z)}\}$, $i = 1, ..., N_x$, 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 one of the distributed resampling algorithms proposed in [5, 19] 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 [18] according to [4]. Compared to the first way, the second way only applies to high dimensional system (1) and it is based on an extension of the DPF. We have assumed 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 $\mathbb{R}^{n_\xi}$) parts. Due to the space limitation, we refer the reader to [9] for the details.

V. Numerical Example

In this section we will test how the DPF performs on a numerical example. 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 (2.53GHz).

A. Algorithms tested

The bootstrap PF is implemented in the standard fashion, using different number of particles (M). The DPF is implemented 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 processing elements (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 [16, 18] 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

$$\text{RMSE of } \hat{x} = \sqrt{\frac{1}{250}\sum_{t=1}^{250}\frac{1}{20000}\sum_{i=1}^{20000}||x_t^i - \hat{x}_t^i||^2} \quad (21)$$

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.

C. Performance evaluation: Timing

One objective with the simulations is to assess the potential efficiency of a 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 a 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 20000 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 computer simulation. The DPF is said to diverge if $w_t^{(i)}$, $i = 1, ..., N_x$, are all equal to zero in the computer simulation. Once the divergence of a PF is detected, the PF will be rerun.

E. Sketch of the simulation

For the example, the bootstrap PF using M particles is first implemented and its accuracy measured by the RMSE will be regarded as the reference level. Then it is shown that

TABLE I

SIMULATION RESULT FOR SYSTEM (22) WITH (23) – "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**

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

$$\begin{aligned} x_{t+1} &= x_t + \frac{z_t}{1+z_t^2} + v_t^x \\ z_{t+1} &= x_t + 0.5z_t + \frac{25z_t}{1+z_t^2} + 8\cos(1.2(t-1)) + v_t^z \\ y_t &= \text{atan}(x_t) + \frac{z_t^2}{20} + e_t \end{aligned} \quad (22)$$

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

$$v_t \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & 0.1 \\ 0.1 & 10 \end{bmatrix}\right), \text{ and } e_t \sim \mathcal{N}(0,1) \quad (23)$$

The simulation result for system (22) with (23) is shown in Table I, from which it can be seen that the DPF has the potential to achieve the same level of accuracy as the bootstrap PF in a shorter execution time.

G. Summary

Regarding the accuracy, comparison of the first part of the RMSE column in Tables I 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 PF.

Regarding timing, comparison of the T_{si} and T_{pi} column in Tables I 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. 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.

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," PhD 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 Journal on Applied Signal Processing*, vol. 15, pp. 2267–2277, 2004.

[5] ——, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions on Signal Processing*, 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] G. Casella and C. P. Robert, "Rao-Blackwellisation of sampling schemes," *Biometrika*, vol. 83, pp. 81–94, 1996.

[8] R. Chen and J. Liu, "Mixture Kalman filters," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 62, pp. 493–508, 2000.

[9] T. Chen, T. B. Schön, H. Ohlsson, and L. Ljung, "An extension of the dencentralized particle filter with arbitrary state partitioning," Automatic Control Division, Linköping University, Tech. Rep. No. 2930, 2010.

[10] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 176–183.

[11] A. Doucet, N. D. Freitas, and N. Gordonn, *Sequential Monte Carlo methods in practice*. New york: Springer, 2001.

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

[13] 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, UK: Oxford University Press, 2009.

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

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

[16] J. D. Hol, T. B. Schön, and F. Gustafsson, "On resampling algorithms for particle filters," in *IEEE Nonlinear Statistical Signal Processing Workshop*, 2006, pp. 79–82.

[17] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[18] 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.

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

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

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