A parallel resampling scheme and its application to distributed particle filtering in wireless networks

Katrin Akebute and Joaquín Miguez
Department of Signal Theory and Communications, Universidad Carlos III de Madrid.
Avenida de la Universidad 30, 28911 Leganés (Madrid), Spain. Email: {kachubegui, jmiguez}@tsc.uc3m.es

Abstract—We address the design of a particle filter (PF) that can be implemented in a distributed manner over a network of wireless sensor nodes, each of them collecting their own local data. This is a problem that has received considerable attention lately and several methods based on consensus, the transmission of likelihood information, the truncation and/or the quantization of data have been proposed. However, all existing schemes suffer from limitations related either to the amount of required communications among the nodes or the accuracy of the filter outputs. In this work we propose a novel distributed PF that is built around the distributed resampling with non-proportional allocation (DRNA) algorithm. This scheme guarantees the properness of the particle approximations produced by the filter and has been shown to be both efficient and accurate when compared with centralized PFs. The standard DRNA technique, however, places stringent demands on the communications among nodes that turn out impractical for a typical wireless sensor network (WSN). In this paper we investigate how to reduce this communication load by using (i) a random model for the spread of data over the WSN and (ii) methods that enable the out-of-sequence processing of sensor observations. A simple numerical illustration of the performance of the new algorithm compared with a centralized PF is provided.

I. INTRODUCTION

The problem of implementing particle filters (PFs) in a distributed fashion has drawn considerable attention in the past few years. Depending on the application, the goals of a distributed particle filter (DPF) can be to speed up the processing of data by sharing the computational load among several sites [1], [2] or to reduce the communication burden when the relevant data are available at different nodes [3], [4].

The proliferation of applications of wireless sensor networks (WSNs) that require both fast and local processing of observation data as brought about the proposal of several new schemes for DPFs. In [5], a fully decentralized particle filtering algorithm for cooperative blind equalization is introduced. The technique is proper, in the sense that it does not make any approximations in the computation of the importance weights of the particles. However, the scheme is practical only when the signal of interest is discrete and is demanding in terms of communications among nodes of the WSN. In [3], the communication load is reduced using quantization and parametric approximations of densities. A similar parametric approach is applied in [4] to obtain a further simplification of the communications. In [1], [2], on the other hand, methods for the parallelization of the resampling step that speed up the particle filter while guaranteeing the properness of the outputs are proposed. However, the latter techniques require that all the observations be available at every processing node, hence they can be communication intensive in a WSN. The same drawback can be observed in the algorithm of [6], that uses a random model (a collection of Markov Chains) in order to spread all the particles and weights computed by a given node over the whole network.

In this paper we propose a novel distributed PF that is built around the distributed resampling with non-proportional allocation (DRNA) algorithm of [1]. This scheme guarantees the properness of the particle approximations produced by the filter and has been shown to be both efficient and accurate when compared with centralized PFs. The standard DRNA technique, however, places stringent demands on the communications among nodes that turn out impractical for a typical wireless sensor network (WSN). In this paper we investigate how to reduce this communication load by using (i) a random model for the spread of data over the WSN and (ii) methods that enable the out-of-sequence processing [7] of sensor observations. The transmission of data over the network is carried out using Markov chain models, akin to the scheme of [6] but involving a lesser amount of messages. A simple numerical illustration of the performance of the new algorithm compared with a centralized PF is provided.

The rest of the paper is organized as follows. Section II presents a signal model for the Bayesian filtering problem and describes the (standard) centralized particle filter. The proposed DPF is introduced in Section III and some illustrative numerical results are reported in Section IV. A brief discussion of the obtained results and open issues, in Section V, concludes the paper.

II. CENTRALIZED PARTICLE FILTERING

A. Problem statement

In this paper we are concerned with the problem of Bayesian online filtering for approximating the posterior distribution of a hidden signal process given a sequence of noisy related observations.

The unobserved signal of interest, denoted \( \{x_t; t \in \mathbb{N}\} \), is modeled as a discrete-time Markov process with \textit{a priori} probability density function (pdf) \( p(x_0) \) and transition density \( p(x_t | x_{t-1}) \). The dimension of each vector \( x_t \) is \( d_x \), i.e., \( x_t \in \mathbb{R}^{d_x} \). The signal \( x_t \) is often referred to as the \textit{state} of the system of interest.

The observations are denoted as \( \{y_t; t \in \mathbb{N}\} \), \( y_t \in \mathbb{R}^{d_y} \), and they are assumed to be conditionally independent given the process \( x_t \), i.e., \( p(y_{1:t} | x_{1:t}) = \prod_{k=1}^{t} p(y_k | x_k) \), where \( y_{1:t} = \{y_1, \ldots, y_t\} \). We assume that the measurements \( y_t \) are collected by \( N \) sensors and we use \( y_{n,t} \) to denote the data collected by the \( n \)-th sensor at time \( t \), hence \( y_t = \{y_{1,t}, \ldots, y_{N,t}\} \). The observations are also conditionally independent across different sensors, i.e., \( p(y_t | x_t) = \prod_{n=1}^{N} p(y_{n,t} | x_t) \).

Our goal is to recursively estimate the posterior pdf \( p(x_{0:t} | y_{1:t}) \) and functions that stem from it, such as the filtering pdf, \( p(x_t | y_{1:t}) \), or expectations of the form

\[
I(f_t) = \mathbb{E}_{p(x_{0:t} | y_{1:t})}[f_t(x_{0:t})],
\]

where \( \mathbb{E}_{p(x)}[\cdot] \) denotes expectation with respect to the pdf in the subscript.

1Probability density functions (pdf’s) are always denoted as \( p \). The notation is argument-wise, i.e., for two different random variables \( x \) and \( y \), \( p(x) \) and \( p(y) \) are the densities of \( x \) and \( y \), respectively, possibly different. Conditional pdf’s are denoted as \( p(x | y) \).
B. Standard particle filter

The standard particle filter, also known as sampling importance resampling (SIR) algorithm or bootstrap filter [8], [9], approximates the posterior distribution with pdf $p(x_{t+1} | y_{1:t})$ by way of an empirical random walk measurement consisting of $M$ samples in the state space, denoted $x^{(m)}_{t+1}$, $m = 1, ..., M$. The algorithm can be outlined as follows:

1) **Initialization.** at $t = 0$:
   a) Draw $M$ random samples, $x^{(m)}_0$, $m = 1, ..., M$, from the prior $p(x_0)$.

2) **Recursive step.** For $t > 0$:
   a) Draw $M$ random samples $\bar{x}^{(m)}_t$, $m = 1, ..., M$, from the pdf’s $p(x_t | x^{(m)}_t)$ and denote $\bar{x}^{(m)}_t = (\bar{x}^{(m)}_t, X_{0:t-1})$.
   b) Compute importance weights: for $m = 1, ..., M$, $\bar{w}^{(m)}_t = p(y_t | \bar{x}^{(m)}_t)$. Normalize the weights as $w^{(m)}_t = \bar{w}^{(m)}_t / \sum_{m=1}^{M} \bar{w}^{(m)}_t$.
   c) Resampling: for $m = 1, ..., M$, let $x^{(m)}_t = \bar{x}^{(j)}$ with probability $w^{(j)}_t$, $j \in \{1, ..., M\}$.

The * superscript in $\bar{x}^{(m)}_t$ denotes non-normalized weights and the bar in $\bar{x}^{(m)}_t$ indicates that the particles have not yet gone through the resampling step. The latter prevents the PF from degeneracy (see, e.g., [10] for details). There are several resampling techniques that can be used in this algorithm [11] and we have assumed the conceptually simple multinomial resampling scheme.

Once we have samples and normalized weights, estimates of $x_t$, or functions of it, can be easily computed. The minimum mean squared error (MMSE) estimate of the state, for instance, can be approximated as $\hat{x}_t^{\text{MMSE}} = \sum_{m=1}^{M} w^{(m)}_t \bar{x}^{(m)}_t$. Note that estimation can also be performed after resampling, i.e., $\hat{x}_t^{\text{MMSE}} = \frac{1}{M} \sum_{m=1}^{M} x^{(m)}_t$ at the expense of a higher variance.

We refer to the standard particle filter as a centralized particle filter (CPF) in order to make explicit that it requires a central unit (CU) that collects all the observations together, generates all the particles and processes them together.

III. DISTRIBUTED PARTICLE FILTER

A. Particle sets and local resampling

We propose a distributed scheme for particle filtering that is built around the DRNA algorithm originally introduced in [1]. In the sequel we describe the method using, essentially, the notations of [2].

Assume we have $N$ nodes in the network. Each node is capable of running a separate particle filtering algorithm with $K$ particles, hence the total number of particles distributed over the network is $M = NK$. Each PF run locally in a node involves the usual steps of drawing new samples, computing weights and resample.

Before resampling, the $K$ particles assigned to node $n$ are denoted $x^{(n,k)}_{t-d:t}$, $k = 1, ..., K$, and they are assigned (unnormalized) importance weights $\bar{w}^{(n,k)}_t$, $k = 1, ..., K$. The relative importance of the $n$-th node is given by the aggregated (unnormalized) weight $W^{(n)}_t = \sum_{k=1}^{K} \bar{w}^{(n,k)}_t$. When needed, the weights can be locally normalized to yield $\tilde{w}^{(n,k)}_t = \frac{\bar{w}^{(n,k)}_t}{\sum_{k=1}^{K} \bar{w}^{(n,k)}_t}$.

Thus, the information held by the $n$-th node before resampling is summarized by the sets $\bar{X}^{(n)}_t = \{x^{(n,k)}_{t-d:t} | k = 1, ..., K\}$ and $\bar{W}^{(n)}_t = \{\bar{w}^{(n,k)}_t | k = 1, ..., K\}$. Notice that we do not assume the node to store the whole paths $\bar{x}^{(n,k)}_{t-d:t}$ but only the $d + 1$ long subpaths $\bar{x}^{(n,k)}_{t-d:t}$.

In the resampling step, for $k = 1, ..., K$ we let $x^{(n,k)}_{t-d:t} = \bar{x}^{(n,k)}_{t-d:t}$ with probability $\tilde{w}^{(n,k)}_t$, $j \in \{1, ..., K\}$, and the aggregated weight of the node remains the same, $W^{(n)}_t$. Hence, the information kept by the $n$-th node after resampling is summarized as $X^{(n)}_t = \{x^{(n,k)}_{t-d:t} | k = 1, ..., K\}$ and $W^{(n)}_t = \{\tilde{w}^{(n,k)}_t | k = 1, ..., K\}$.

B. Estimates

Assume we are interested in the estimation of $f(x^{(n,k)}_{t-d:t})$ where $f$ is some function of the states between the times $t - d$ and $t$. A local estimate can be obtained at the $n$-th node, $n \in \{1, ..., N\}$, either before resampling,

$$f(x^{(n,k)}_{t-d:t}) = \sum_{k=1}^{K} \tilde{w}^{(n,k)}_t f(x^{(n,k)}_{t-d:t}),$$  

or after resampling

$$\hat{f}(x^{(n,k)}_{t-d:t}) = \frac{1}{K} \sum_{k=1}^{K} f(x^{(n,k)}_{t-d:t}).$$

Global estimation is carried out by transmitting all the local estimates and aggregated weights, $\hat{f}(x^{(n,k)}_{t-d:t})$ and $W^{(n)}_t$, $n = 1, ..., N$, to a common node, where the estimate

$$\hat{f}(x^{(n,k)}_{t-d:t}) = \frac{1}{N} \sum_{n=1}^{N} W^{(n)}_t \hat{f}(x^{(n,k)}_{t-d:t})$$

is computed after obtaining the normalized aggregated weights $W^{(n)}_t = W^{(n)}_t / \sum_{n=1}^{N} W^{(n)}_t$. Note that the weights are used properly. Indeed, substituting (1) into (2) yields

$$\hat{f}(x^{(n,k)}_{t-d:t}) = \frac{1}{\sum_{n=1}^{N} \sum_{k=1}^{K} \tilde{w}^{(n,k)}_t} \sum_{n=1}^{N} \sum_{k=1}^{K} \tilde{w}^{(n,k)}_t f(x^{(n,k)}_{t-d:t}).$$

C. Particle exchange

In order to keep the aggregated weights balanced across the different nodes, it is necessary to exchange subsets of particles locally between pairs of neighboring nodes [2]. Let $N_n = \{i_1, i_2, ..., i_{N_n}\} \subseteq \{1, 2, ..., N\}$ denote the set of indices corresponding to the neighbors of the $n$-th node. The $n$-th node transmits a disjoint subset of $P$ particles to each of its neighbors. Specifically, let $M^{(n, i)}_t = \{x^{(n, i)}_{t-d:t}, ..., x^{(n, i)}_{t-d:t}\}$ be the particles transmitted from node $n$ to node $s \in N_n$. The indices $i_1, ..., i_P$ can be selected in any desired way (even randomly) as long as the messages $M^{(n, i)}_t$ are disjoint, i.e., $M^{(n, i)}_t \cap M^{(n, j)}_t = \emptyset$ for any $s, r \in N_n, s \neq r$.

Every node sends $P$ weighted particles to each neighbor and receives $P$ weighted particles from each neighbor. Therefore, after the particle exchange process, the overall set of particles (network-wide) is exactly the same and with the same weights. Its only assignment of particles to nodes that changes. Specifically, the information held by the $n$-th node after the particle exchange is given by

$$X^{(n)}_t = \left\{x^{(n,k)}_{t-d:t} | k = 1, ..., K\right\}$$

and

$$W^{(n)}_t = \{\tilde{w}^{(n,k)}_t | k = 1, ..., K\}.$$
of observations over the network that: (a) keeps the communication load bounded and (b) asymptotically guarantees that the observations collected at a node \( n \) become available all over the network.

Every node needs to keep track of the locally processed observations, either collected by itself or received from a different node. Specifically, at time \( t \) the node \( n \) maintains a set with the indices (node and time of origin) of each observation that has been processed up to time \( t - 1 \), denoted \( T^n_t \subseteq \{1, ..., N\} \times \{1, ..., t - 1\} \). The elements of \( T^n_t \) are pairs of the form \((s, \tau)\) where \( s \in 1, ..., N \) and \( \tau \geq t - 1 \). If an observation \((s, \tau)\) has been received and processed at node \( n \) up to time \( t - 1 \), then the observation \((s, \tau)\) has been processed at node \( n \).

The transmission of observations over the network is performed stochastically. In each time step, each observation performs a “jump” over the network, where, in order to keep communication traffic low, the number of jumps performed is less than the number of nodes in the network, i.e. \( L < N \). To each observation we attach a retransmission counter that indicates how many jumps have been needed to take the observation from the node where it was originally measured to the present node. The counter has to be transmitted together with the observation and is denoted \( c_{s,\tau} \), for the observation vector \( y_{s,\tau} \). There is an upper bound on the number of times an observation can be retransmitted, denoted \( B \). Therefore, the observation \( y_{s,\tau} \) keeps being transmitted from one node to another (neighbor) node until \( c_{s,\tau} = B \).

Let \( \mathcal{Y}^n_{L-1} \) be the set of observations at node \( n \) which are pending retransmission before we start the data exchange at time \( t \). Note that \( y_{s,\tau} \in \mathcal{Y}^n_{L-1} \) if, and only if, \( y_{s,\tau} \) was received from a neighbor node \( s \in N_n \) at time \( t - 1 \) and \( c_{s,\tau} < B \). The data exchange process at time \( t \) and node \( n \) runs as follows.

- **Initialization.** Collect the new local data at the site of node \( n \), i.e., set \( \mathcal{Y}^n_0 = \mathcal{Y}^n_{L-1} \) and \( \{y_{s,\tau}, c_{s,\tau} = 0\} \).

- For \( t = 1, ..., L \):
  1. For each \( y_{s,\tau} \in \mathcal{Y}^n_t \), choose randomly a neighbor node \( r \in N_n \) uniformly at random. Then set \( c_{s,\tau} = c_{s,\tau} + 1 \) and transmit \((y_{s,\tau}, c_{s,\tau})\) to node \( r \).
  2. Receive one pair \((y_{s,\tau}, c_{s,\tau})\) from each neighbor node \( s \in N_n \). Let \( \mathcal{Y}^n_t \subseteq \{y_{s,\tau}, c_{s,\tau}\} \).
  3. For each \( y_{s,\tau} \in \mathcal{Y}^n_t \), if \( (s, \tau) \notin T^n_{L-1} \) then \( T^n_t \leftarrow T^n_{L-1} \cup \{(s, \tau)\} \) and we update the weights in \( W^n_t \) as

\[
    w_{i}^{(n,k)} \leftarrow w_{i}^{(n,k)} \cdot p(y_{s,\tau} | x_{s}^{(n,k)}), \quad k = 1, ..., K.
\]

- Let \( \mathcal{Y}^n_t := \{y_{s,\tau} \in \mathcal{Y}^n_t : c_{s,\tau} < B\} \) be the new set of observations pending retransmission.

Some of the observations from the remaining nodes may arrive at node \( n \) out of sequence. Because of the assumed conditional independence of the observations, however, it is straightforward to show that the update equation (3) guarantees the weights to remain proper.

**E. Outline**

The proposed DPF can be summarized as follows.

1. **Initialization, at** \( t = 0 \). For each node, \( n = 1, ..., N \):
   a) Draw \( x_{0}^{(n,k)} \), for \( k = 1, ..., K \), from the prior pdf \( p(x_0) \) and assign equal weights, \( w_{0}^{(n,k)} = \frac{1}{K} \).
   b) Set \( X_0 = \{x_{0}^{(n,k)}\} \) and \( W_0 = \{w_{0}^{(n,k)}\} \).

2. **Recursive step, at every** \( t > 0 \). Given \( X_{t-1} \) and \( W_{t-1} \), \( n = 1, ..., N \):
   a) Perform the **particle exchange** as described in Section III-C to obtain the sets \( X^n_{t} \) and \( W^n_{t} \) for \( n = 1, ..., N \).
   b) **Sampling:** Draw \( x^{(n,k)}_t \), from \( p(x_t | x_{t-1}^{(n,k)}) \), for \( n = 1, ..., N \) and \( k = 1, ..., K \). Build \( X^n_{t} \), \( n = 1, ..., N \), as in Section III-A.
   c) **Weight update:** Let \( w^{(n,k)}_t = w^{(n,k)}_{t-1} p(y_{t,n} | x^{(n,k)}_t) \). Build \( W^n_{t} \), \( n = 1, ..., N \), as in Section III-A.
   d) **Resample** as in Section III-A to build the sets \( X^n_{t} \) and \( W^n_{t} \).
   e) Perform the **data exchange** as described in Section III-D, including the update of the weight sets \( W^n_{t} \).

**F. Missing data**

Using the classical theory of Markov chains, and given the adjacency matrix of the WSN, it is possible to determine the value of \( B \) that ensures that the full observation vector \( y_{1:t} \) will be received at time \( t \) by all nodes with a prescribed probability \( \pi \).

In practice, however, it may be necessary to compute estimates of the state variables when the not every node has processed all observations, i.e., when the index sets \( I^t_1, ..., I^t_l \) are possibly different. If \( I^t_i \neq I^t_j \) for any pair \( i, j \), then the assumptions of the original DRNA algorithm do not hold and the weights cannot be guaranteed to be proper.

One way to alleviate this problem is to substitute any observation \( y_{s,\tau} \) missing at node \( n \) and time \( t \) by its posterior estimate

\[
    y_{s,\tau}^n = E_p(y_{s,\tau} | y_{1:t}, (r_i) \in I^t_i) \quad \text{(4)}
\]

and update the weights accordingly. Although, obviously, the expectation in (4) cannot be computed exactly, it is straightforward to approximate it using the particles available at node \( n \).

**IV. COMPUTER SIMULATIONS**

For illustration, we consider the problem of tracking a target that moves along a 3-dimensional region and transmits a radio signal. Assume that \( N = 8 \) sensors are placed at the vertices of a cube and the length of a side of the cube is of 2 meters. Each sensor measures the received signal strength (RSS) of the signal transmitted by the target.

The state signal consists of the position and velocity of the target along each dimension, i.e., \( x_t = [r_{1,t}, r_{2,t}, r_{3,t}, \dot{r}_{1,t}, \dot{r}_{2,t}, \dot{r}_{3,t}]^T \in R^6 \), where \([r_{1,t}, r_{2,t}, r_{3,t}]^T \) denotes the target position and \([\dot{r}_{1,t}, \dot{r}_{2,t}, \dot{r}_{3,t}]^T \) denotes its velocity, both in a 3-dimensional space. The state vector has a known Gaussian prior, \( p(x_0) = N(x_0, \Sigma_0) \), and evolves with time according to the stochastic difference equation [12]:

\[
    x_t = A x_{t-1} + Q u_t, \quad \text{(5)}
\]

where \( A = \begin{bmatrix} I_3 & T_3 I_3 & 0 \end{bmatrix} \) and \( Q = \frac{1}{2} T_3^2 I_3 \begin{bmatrix} 0 I_3 \end{bmatrix} \) are known, \( I_3 \) is a \( 3 \times 3 \) identity matrix, \( T_3 = 0.1 \) is the observation period, and \( u_t \) is a Gaussian noise of zero mean and covariance matrix \( \Sigma_u = \begin{bmatrix} \sigma_u^2 I_3 & 0 I_3 \ 0 I_3 & \sigma_u^2 I_3 \end{bmatrix} \), with the variance parameters \( \sigma_u^2 \) and \( \sigma_u^2 \).

The measurement collected by the \( j \)-th sensor at time \( t \) is denoted as \( y_{j,t} \) and its relationship with the target position, \( r_t \), is described by the path-loss model [13]:

\[
    y_{j,t} = 10 \log_{10} \left( \frac{P_0}{d_{j,t}^2} + \eta \right) + e_{j,t}, \quad \text{(6)}
\]

\( ^2 \)i.e., the time interval between consecutive observations.
where $\eta = 10^{-10}$ is the minimum power a sensor is capable of detecting; $P_0 = 1$ is the transmission power; $\gamma = 3$ is the propagation parameter, $d_{j,t} = \|r_t - s_j\|$ is the distance between the position of the $j$-th sensor, $s_j$, and the target position at time $t$, $r_t$, at time $t$; and $\varepsilon_{j,t} \sim N(\varepsilon_{j,t}; 0, \sigma^2)$ is normally distributed, zero-mean noise with known variance $\sigma^2 = 1$.

We have used the dynamic model (5) to simulate random target trajectories and we have generated synthetic observations at the sensors using model (6). For comparing the algorithms, the same observations are fed into a CPF with $M = 800$ particles and the proposed DPF with $N = 8$ nodes and $K = 100$ particles per node. Each node exchanges $P = 10$ particles with each neighbor after the local resampling step and each node is connected to $N_a = 3$ other nodes. All communication links between nodes are bidirectional. For the spreading of the observations, each datum $y_{j,t}$, $j \in \{1, 2, \ldots, 8\}$, is retransmitted $L = 4$ times per algorithm step and can be retransmitted up to $B = 8$ times.

Figure 1 displays the result of a single simulation trial for the CPF and the DPF. It shows the true trajectory of the target (blue), the estimate computed using a CPF (red) and the estimated produced by the DPF (green) along the three axis of the space. It can be seen that the two PFs attain a similar output.

Figure 2 displays the absolute error (in meters) versus discrete-time attained by the CPF (solid line) and the DPF (dashed line) in the estimation of the target position. The curves have been obtained by averaging the errors in 100 independent simulations. This plot shows clearly the performance loss in which we incur by using the distributed particle filter when compared with the CPF that processes all observations together in a CU.

![Fig. 1. Tracking example in a three dimensional space. The plots show the true target trajectory (blue), the estimate computed using the CPF (red) and the estimated produced by the DPF (green). Upper: x-axis. Middle: y-axis. Lower: z-axis.]

**V. CONCLUSIONS**

We have introduced a distributed particle filtering scheme that is built around the DRNA algorithm of [1], [2] for the parallelization of the resampling steps. The new scheme can be shown to guarantee that the particle weights (and state estimators) are constructed properly when the number of allowed retransmission steps for the observations (the parameter $L$ in Section III-D) is large enough. Unlike the standard DRNA technique, that places stringent demands on the communications among nodes that turn out impractical for a typical WSN, the new method attains a trade-off between communication load and accuracy of the estimators. The performance of the resulting distributed particle filter has been illustrated by way of computer simulations for a target tracking problem in three dimensions.

**ACKNOWLEDGMENTS**

This work has been supported by Ministerio de Ciencia e Innovación (project DEIPRO, ref. TEC-2009-14504-C02-01, and program CONSOLIDAR-INGENIO 2010, ref. CSD2008-00010 COMONSENS) and a joint program of Comunidad de Madrid and Universidad Carlos III de Madrid (CCG10-UC3M/TIC-5225 ETORS).

**REFERENCES**


