EFFICIENT DISTRIBUTED RESAMPLING FOR PARTICLE FILTERS

Balakumar Balasingam,∗ Miodrag Bolić,∗ Petar M. Djurić,† and Joaquin Míguez‡

∗School of Information Technology and Engineering, University of Ottawa (Canada)
†Department of Electrical and Computer Engineering, Stony Brook University (USA)
‡Department of Signal Theory and Communications, Universidad Carlos III de Madrid (Spain)
e-mail: {balasing,mbolic}@site.uottawa.ca, djuric@ece.sunysb.edu, joaquin.miguez@uc3m.es

ABSTRACT

In particle filtering, resampling is the only step that cannot be fully parallelized. Recently, we have proposed algorithms for distributed resampling implemented on architectures with concurrent processing elements (PEs). The objective of distributed resampling is to reduce the communication among the PEs while not compromising the performance of the particle filter. An additional objective for implementation is to reduce the communication among the PEs. In this paper, we report an improved version of the distributed resampling algorithm that optimally selects the particles for communication between the PEs of the distributed scheme. Computer simulations are provided that demonstrate the improved performance of the proposed algorithm.

Index Terms—Sequential Monte-Carlo methods, particle filters, distributed resampling.

1. INTRODUCTION

Particle filters provide a feasible solution for non-linear non-Gaussian filtering problems [1]. Given noisy observations, particle filters recursively approximate the distribution of a state-space system in the form of discrete random measures. The particle filtering algorithm is composed of three steps: sampling, computation of importance weights, and resampling. The first two steps can readily be implemented in parallel. However, the resampling step requires joint processing of all the particles.

One approach to bridge the gap between particle filtering theory and its use in practice is to reduce the execution time of the algorithms by implementing them using multiple concurrent processing elements (PEs) and the obvious challenge in accomplishing this is the parallelization of the resampling step. In [2], several types of distributed resampling (DR) algorithms were proposed and discussed. Out of these, DR with non-proportional allocation (DNA) offers a large degree of parallelism while constraining the communication to only adjacent PEs. Some theoretical analysis of DNA was later presented in [3].

We point out that the objective of DR is to reduce the execution time of the particle filter by distributing the particles among N PEs. If the PEs were let to run as separate filters, each of them would most likely have poor performance due to inadequate number of particles. Furthermore, after some time, the PEs would have N different posterior distributions that would be significantly different from the true posterior of the target. The fusion of the individual distributions would not solve the problem.

This difficulty can be alleviated by exchanging some particles among the PEs. Indeed, the DNA scheme in [2] randomly selects L particles from the (n − 1)th PE and replaces them by L randomly chosen particles from the nth PE [2]. The objective of this exchange is to allow for mixing of the posterior distributions among the adjacent PEs, so as to preserve the diversity of the particle populations.

In this paper, we propose a statistically principled way to achieve this mixing. Specifically, we argue that we should exchange the L particles that best represent the K particles in a PE. Then, we formally show, by using the Kullback-Leibler divergence, that these particles are the ones with largest weights. We present computer simulations to numerically verify that this strategy outperforms the ones based on random selection of the particles.

The rest of the paper is organized as follows. Background material on the standard particle filter is given in Section 2. The proposed distributed scheme is presented in Section 3 and numerical results are given in Section 4. Section 5 is devoted to the conclusions.

2. PARTICLE FILTERING

Particle filters are aimed to (recursively) approximate the posterior distribution of the state variables of a dynamic system given a sequence of related observations. Although the methodology can be applied with more complex systems, here we assume a state-space model with a first-order Markov state process and conditionally independent observations for simplicity. Then, we present the standard particle filtering algorithm.

2.1. State-space models

Many signal processing problems can be represented by the state-space random model

\[ x_0 \sim p(x_0), \]
\[ x_t \sim p(x_t|x_{t-1}), \]
\[ y_t \sim p(y_t|x_t), \quad t = 1, 2, \ldots, \]

where the \( d_x \times 1 \) vector \( x_t \in \mathbb{R}^{d_x} \) is the system state at time instant \( t \) and the \( d_y \times 1 \) vector \( y_t \in \mathbb{R}^{d_y} \) represents observations. The prior pdf of the state at time \( t = 0 \) is denoted \( p(x_0) \); the state process is

\[ p \] denotes probability density functions (pdfs) throughout the paper. It is an argument-wise notation, hence for two random variables \( x \) and \( y, p(x) \) is the pdf of \( x \) and \( p(y) \) is the pdf of \( y \), possibly different. The conditional density of \( x \) given \( y \) is denoted as \( p(x|y) \).
first-order Markov and evolves with time according to the transition density \( p(x_t|x_{t-1}) \), \( t \geq 1 \), and the conditional density of the observations, \( p(y_t|x_t) \), is often referred to as the likelihood of \( x_t \). These three densities, \( p(x_0) \), \( p(x_t|x_{t-1}) \), and \( p(y_t|x_t) \), completely define the model, and all their parameters are assumed known.

The goal of particle filters is to approximate integrals with respect to (w.r.t.) the sequence of filtering pdf’s \( p(x_t|y_{1:t}) \), \( t \geq 1 \), by way of summations. For an integrable real function \( f \), we denote \( \int_{y_{1:t}} f(x_t)p(x_t|y_{1:t})\,dx_t \).

2.2. Standard particle filter

The standard particle filtering algorithm, also known as a bootstrap filter [1], can be outlined as follows.

1. **Initialization.** Draw \( M \) random samples, or particles, \( x_0^{(m)} \), \( m = 1, \ldots, M \), from the prior \( p(x_0) \). Assign them equal weights, \( w_0^{(m)} = 1/M \), and let \( \mathcal{X}_0 = \{x_0^{(m)}, w_0^{(m)}\}_{m=1}^M \).

2. **Recursive step.** Given \( \mathcal{X}_{t-1} \), the following steps:

   a. Draw \( \tilde{x}_t^{(m)} \) from \( p(x_t|x_{t-1}^{(m)}) \), \( m = 1, \ldots, M \).
   
   b. Set weights \( w_t^{(m)*} = w_t^{(m)} p(y_t|x_t^{(m)}) \), \( m = 1, \ldots, M \).

   c. Normalize them as \( w_t^{(m)} = \frac{w_t^{(m)*}}{\sum_{l=1}^M w_t^{(l)*}} \).

   d. For \( m = 1, \ldots, M \), if \( \frac{1}{M} \in N \),

      - then resample: set \( \tilde{x}_t^{(m)} = \tilde{x}_t^{(k)} \) and \( w_t^{(m)} = 1/M \) with probability \( \frac{w_t^{(k)}}{\sum_{l=1}^M w_t^{(l)}} \),
      - else set \( \tilde{x}_t^{(m)} = \tilde{x}_t^{(m)} \) and \( w_t^{(m)} = w_t^{(m)*} \).

3. **Distributed resampling**

3.1. Computational scheme

The resampling step of the CPF algorithms requires that all the particles and weights be stored together and be accessed by a single processor. On the contrary, in this paper we assume a structure with \( N \) independent PEs and one control unit (CU), similar to the scheme of [2],[3]. The \( N \) PEs operate concurrently and they can be synchronized after each particle filtering step.

Let \( M \) be the total number of particles. Each PE is responsible for generating and propagating over time \( K = M/N \) (\( K \in N \)) distinct particles, together with their non-normalized weights. To be specific, we denote the particle set of the \( n \)th PE as \( \mathcal{X}_t^n = \{x_t^{(n,k)}, \bar{w}_t^{(n,k)}\}_{k=1}^K \), where \( x_t^{(n,1)}, \ldots, x_t^{(n,K)} \) are the particles, \( \bar{w}_t^{(n,1)}, \ldots, \bar{w}_t^{(n,K)} \) are the locally normalized weights (i.e., \( \sum_{k=1}^K \bar{w}_t^{(n,k)} = 1 \) for every \( n \)) and \( W_t^n = \sum_{k=1}^K \bar{w}_t^{(n,k)} \) is the sum of the non-normalized weights of the particles.

The processors are connected using an interconnection network. Even though any type of network can be used, we limit the presentation in this paper to a ring configuration, meaning that a) the PEs can communicate locally to neighboring processors and b) the exchange of messages among PEs can be performed concurrently. Similar assumptions with deterministic routing can be introduced, e.g., for a mesh network. For a ring network, the \( n \)th PE, \( n = 1, \ldots, N-1 \), can transmit data to the \((n+1)\)th PE. The \( N \)th element receives data from the \((N-1)\)th processor but transmits to PE number 1. The communication link can support the communication of \( L \) particles with their non-normalized weights, i.e., the message from the \( n \)th to the \((n+1)\)th PE has the form \( M_t^n = \{\tilde{x}_t^{(n,i)}, \bar{w}_t^{(n,i)*}\}_{i=1}^L \), where \( \bar{w}_t^{(n,i)*} = W_t^n \bar{w}_t^{(n,i)} \) and \( i_1 < \ldots < i_L \in \{1, \ldots, K\} \).

The PEs can also send data to the CU. Specifically, given a function \( f \) that can be completely evaluated at the PEs, every processor can compute a local estimate of the form

\[
(f, p_t^{(n,K)}) = \sum_{k=1}^K \bar{w}_t^{(n,k)} f(x_t^{(n,k)})
\]

and transmit the pair \( \{f, p_t^{(n,K)}\}, W_t^n, W_t^{(n)*}\} \) to the CU. Then, the CU can combine the local estimates to construct a global estimate

\[
(f, p_t^{N*K}) = \sum_{n=1}^N W_t^n (f, p_t^{(n,K)}),
\]

where \( W_t^n = W_t^{(n)*} / \sum_{n=1}^N W_t^{(n)*} \) is the normalized aggregated weight of the \( n \)th PE.

Communication from the CU to the PEs is also necessary. In hardware implementations, the CU manages the PEs through a set of control signals. In particular, the CU can command the PEs to simultaneously a) generate their \( K \) particles and start running, b) compute local estimates and dump them to the CU, c) perform resampling and d) normalize the aggregated weights. Note that:

- The computation of the \((f, p_t^{n,K})\) estimates in the \( N \) PEs does not need to be performed at any step. The local estimates can be requested by the CU at any time.

- For numerical stability, the aggregated weights \( W_t^{(n)*} = \sum_{n=1}^N W_t^{(n)*} \), \( n = 1, \ldots, N \), which are stored at the PEs, must be normalized after a number of steps. This can be done asynchronously with the regular operation of the PEs, e.g., using delayed normalization constants [3, Section 2.2].

3.2. Algorithms

The DRNA class of methods was originally proposed in [2]. It relies on a) the local resampling of particles within each PE and b) the exchange of particles among the PEs. The exchange was introduced to prevent impoverishment of the local particle sets \( \mathcal{X}_t^n \) and the overall particle set \( \mathcal{X}_t = \cup_{h=1}^N \mathcal{X}_t^h \). While many standard techniques
can be used for local resampling, the optimization of the particles to
be exchanged has not been studied so far (e.g., they were selected randomly in [2],[3]).

Here we investigate distributed particle filters (DPFs) that employ DNA. Their general form is outlined below. We reiterate that we assume $N$ PEs that process $K$ particles each, with $M = NK$ the total number of particles. All PEs can operate in parallel and resampling is carried out every $t_0$ steps.

1. **Initialization.** For $n = 1, \ldots, N$,
   (a) draw $x_0^{(n,k)}$ from $p(x_0)$ and set $w_0^{(n,k)} = \frac{1}{K}$, for all $k = 1, \ldots, K$;
   (b) set $W_0^{(n)*} = 1$ and $\lambda_0^{(n)} = \{x_0^{(n,k)}, w_0^{(n,k)}\}_{k=1}^K$.

2. **Recursive step.** Assume $\lambda_{n-1}^{(n)}$, $n=1,\ldots, N$, are available. For $n = 1, \ldots, N$,
   (a) draw $x_t^{(n,k)}$ from $p(x_t|x_{t-1}^{(n,k)})$,
   (b) compute $w_t^{(n,k)*} = \frac{W_{t-1}^{(n,k)} p(y_t|x_t^{(n,k)})}{\sum_{k=1}^K w_{t-1}^{(n,k)} p(y_t|x_t^{(n,k)})}$, $k = 1, \ldots, K$;
   (c) set $W_t^{(n)*} = \sum_{k=1}^K w_t^{(n,k)*}$ and $w_t^{(n,k)} = \frac{w_t^{(n,k)*}}{W_t^{(n)*}}$,
   (d) if $\tau \in \mathbb{N}$,
     • then exchange particles and resample,
     • else set $x_t^{(n,k)} = x_t^{(n,k)}$ and $\lambda_t^{(n)} = \{x_t^{(n,k)}, w_t^{(n,k)}\}_{k=1}^K$.

Various procedures can be applied for the exchange and resampling step. The aim is to select a particle subset that represents the empirical distribution defined by $\lambda_t^{(n)}$ and then transmit it to the next PE in the ring, in order to enhance the diversity of all the subsets, $\lambda_1^{(n)}, \ldots, \lambda_N^{(n)}$. The scheme proposed in [3] was simple. First, local resampling is carried out, hence, for $k = 1, \ldots, K$,

$$x_t^{(n,k)} = \bar{x}_t^{(n,i)}$$ with probability $w_t^{(n,i)}$, \quad $i \in \{1, \ldots, K\}$,

and the normalized weights are updated as $w_t^{(n,k)} = 1/K$, $k = 1, \ldots, K$. Then, the first $L$ particles (and non-normalized weights) in the $n$th PE are synchronously transmitted to substitute the first $L$ particles in the $(n+1)$th PE, i.e., we simultaneously carry out the substitutions

$$\{x_t^{(n+1,i)}, w_t^{(n+1,i)*}\}_i \leftarrow \{x_t^{(n,i)}, w_t^{(n,i)*}\}_i$$ \quad for $n = 1, \ldots, N - 1$ and (4)

$$\{x_t^{(1,i)}, w_t^{(1,i)*}\}_i \leftarrow \{x_t^{(N,i)}, w_t^{(N,i)*}\}_i,$$

where the unnormalized weights are $w_t^{(n,k)} = W_t^{(n)*}/K$.

Here, we propose to select the particles for exchange before resampling, in such a way that the divergence between the empirical distributions generated by a particle subset of size $L$, $\{\bar{x}_t^{(n,l)}, w_t^{(n,l)*}\}_{l=1}^L$, and the complete particle population $\bar{X}_t^{(n)} = \{x_t^{(n,k)}, w_t^{(n,k)}\}_{k=1}^K$ is minimized. In particular, given the multi-index $\{i_t^{(1)}, \ldots, i_t^{(L)}\} \subset \{1, \ldots, K\}$, define the probability mass functions $^2$ (pmf’s)

$$\pi_t^{n,L}(x) = \frac{1}{L} \sum_{l=1}^L I_{x_t^{(n,l)}}(x) w_t^{(n,l)*}$$

and

$$\pi_t^{n,K}(x) = \sum_{k=1}^K I_{x_t^{(n,k)}}(x) w_t^{(n,k)}$$,

where $I_{x}(x)$ is the indicator function that yields $I_{x}(x) = 1$ if $x = y$ and $I_{x}(x) = 0$ if $x \neq y$. Under the common assumption $0 \log 0 = 0$, the Kullback-Leibler divergence [7] between $\pi_t^{n,L}$ and $\pi_t^{n,K}$ is

$$D(\pi_t^{n,L}, \pi_t^{n,K}) = - \log \left( \frac{\sum_{l=1}^L w_t^{(n,l)}}{\sum_{k=1}^K w_t^{(n,k)}\} \right).$$

Obviously, the minimization of $D(\pi_t^{n,L}, \pi_t^{n,K})$ is equivalent to the maximization of $\sum_{l=1}^L \pi_t^{n,L}(x)$. Therefore, the optimal multi-index for the $n$th processor,

$$\{i_t^{(n)}, \ldots, i_t^{(L)}\} = \arg \min_{\{i_t^{(1)}, \ldots, i_t^{(L)}\} \in \{1, \ldots, K\}^L} D(\pi_t^{n,L}, \pi_t^{n,K})$$ \quad (5)

and building the message $M_t^{n} = \{x_t^{(n,i_t^{(1)}), w_t^{(n,i_t^{(1)})*}}\}_{l=1}^L$, $n = 1, \ldots, N$, to be transmitted to the next PE in the ring. To summarize, we perform the following steps:

1. Compute $M_t^{n} = \{x_t^{(n,i_t^{(1)}), w_t^{(n,i_t^{(1)})*}}\}_{l=1}^L, 1 \leq n \leq N$.
2. Simultaneously substitute

$$\{x_t^{(n+1,i_t^{(1)}), w_t^{(n+1,i_t^{(1)})*}}\}_{l=1}^L \leftarrow M_t^{n}, 1 \leq n < N,$$

$$\{x_t^{(1,i_t^{(1)}), w_t^{(1,i_t^{(1)})*}}\}_{l=1}^L \leftarrow M_t^{N}.$$ 3. Once the exchange is completed, update the aggregated and normalized weights at every PE, i.e., recalculate $W_t^{(n)*} = \sum_{k=1}^K w_t^{(n,k)*}$ and $w_t^{(n,k)} = w_t^{(n,k)*}/W_t^{(n)*}$. Then perform local resampling concurrently at each PE. For $n = 1, \ldots, N$ and $k = 1, \ldots, K$, set

$$x_t^{(n,k)} = x_t^{(n,i_t^{(1)})}$$ with probability $w_t^{(n,i_t^{(1)})}$, \quad $i_t^{(1)} \in \{1, \ldots, K\}$,

$$w_t^{(n,k)*} = W_t^{(n)*}/K$$ and $w_t^{(n,k)} = 1/K$.

4. Finally, update the particle sets, i.e.,

$$\lambda_t^{(n)} = \{x_t^{(n,k)}, w_t^{(n,k)}\}_{k=1}^K, 1 \leq n \leq N.$$ 2Recall that the weights $w_t^{(n,1)}, \ldots, w_t^{(n,K)}$ are normalized, i.e.,

$$\sum_{k=1}^K w_t^{(n,k)} = 1.$$

We refer to the DPF algorithm using the exchange in (4) as DPF1, while the new scheme based on (5) is referred to as DPF2 hereafter.

4. **NUMERICAL RESULTS**

For illustration, we use the problem of tracking a target that moves over a two-dimensional region and transmits a radio signal. Two
sensors receive that signal and measure its angle of arrival (this problem is commonly known in the literature as bearings-only tracking). To be specific, let $x_t = [x_{1,t}, x_{2,t}, x_{3,t}, x_{4,t}]^\top$ denote the state of the target at time $t$, which includes its horizontal and vertical coordinates, $x_{1,t}$ and $x_{2,t}$, respectively, as well as its velocity components, $x_{3,t}$ and $x_{4,t}$, along the corresponding axes. The state vector is initialized with a Gaussian prior, $p(x_0) = N(x_0, \Sigma_0)$, and then evolves with time according to the dynamical model

$$x_t = \Phi x_{t-1} + \Gamma u_t,$$

where $\Phi = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $\Gamma = \begin{bmatrix} 0.5 & 0 \\ 1 & 0 \end{bmatrix}$ are known constant matrices and $u_t = [u_{x,t}, u_{y,t}]^\top$ is a sequence of independent and identically distributed (i.i.d.) Gaussian noise vectors with zero mean and covariance matrix $\sigma_u^2 I_2$ ($I_m$ being the $m \times m$ identity matrix).

The sensors were placed at the positions $s_1 = [1, 1]^\top$ and $s_2 = [1, -1]^\top$, and we assumed that one measurement from each sensor was available at each time step $t = 1, 2, \ldots$. In particular, the vector of observed angles $y_t = [y_{1,t}, y_{2,t}]^\top$ was modeled as

$$y_t = H(x_t) + v_t,$$

where $H(x_t) = \begin{bmatrix} \tan^{-1}(\frac{y_{1,t}}{x_{1,t}}) \\ \tan^{-1}(\frac{y_{2,t}}{x_{2,t}}) \end{bmatrix}$ and $v_t = [v_{x,t}, v_{y,t}]^\top$ is a vector of i.i.d. Gaussian noise vectors with zero mean and covariance matrix $\sigma_v^2 I_2$.

We ran 1000 independent simulations of the state-space model given by (7) and (8) and, for each run, applied the CPF, DPF1 and DPF2 algorithms. The overall number of particles was $M = 250$ and, for the DPFs, we split them into $K = 10$ processors with $K = 10$ particles each. Resampling was carried out every $T_r = 2$ steps and $L = 2$ particles were exchanged.

The upper plot in Fig. 1 compares the three filters when used to track the full state vector $x_t$. In particular, if $\hat{x}_t(j)$ is the estimate produced in the $j$th simulation run, the empirical mean square error is

$$MSE = \frac{1}{N} \sum_{j=1}^{N} \sum_{t=1}^{T} \|\hat{x}_t(j) - x_t\|^2,$$

where $x_t$ is the true state vector and $\| \cdot \|$ denotes the Euclidean norm. We observe that the MSE using the new DPF2 algorithm is clearly lower than that of the original DPF1 technique. The lower plot of Fig. 1 compares the (empirical) effective sample size $M_{\text{eff}}$ obtained with each of the three algorithms. At time $t$, this quantity is computed as

$$M_{\text{eff}} = \frac{\sum_{n=1}^{N} \sum_{k=1}^{K} \left( \frac{w_t(n,k)^*}{W_t(n)^*} \right)^2}{\sum_{n=1}^{N} \sum_{k=1}^{K} \left( \frac{w_t(n,k)^*}{W_t(n)^*} \right)} - 1,$$

for the two DPFs, and $M_{\text{eff}} = \left( \sum_{m=1}^{M} w_t(m)^2 \right)^{-1}$ for the CPF. It is apparent that $M_{\text{eff}} \leq M$ and, intuitively, it can be interpreted as the size of an i.i.d. collection of samples from the true pdf of $p(x_t|y_{1:t})$ that would produce approximations as good as those obtained with the weighted particles. The simulations show that the DPF2 algorithm achieves values of $M_{\text{eff}}$ that are consistently higher than those of the DPF1 technique. The results for the CPF are displayed for reference.

5. CONCLUSIONS

We have proposed a novel scheme for distributed resampling implemented over a number of processing elements. The key for good performance of the new scheme is that the adjacent processing elements exchange a small portion of particles that are optimal according to a statistical criterion. We also provided details for computing desired integrals under the posterior distribution of the state. Computer simulations based on the new scheme have shown improved performance.

6. REFERENCES


