Distributed Particle Filtering Under Real-Time Constraints

Alper Kamil Bozkurt and Ali Taylan Cemgil
Dept. of Computer Engineering
Boğaziçi University
34342, Bebek, İstanbul, Turkey
{alper.bozkurt, taylan.cemgil}@boun.edu.tr

***Abstract*—Particle filters are powerful methods for state estimation in nonlinear/non-Gaussian dynamical systems. However due to the heavy computational requirements, they may not satisfy the real-time constraints in many applications requiring a large number of particles. By means of distributed implementation, real-time particle filtering can be achieved. However, the resampling stage in particle filters requires particle interaction which causes communication overhead. In this work, we propose a distributed resampling algorithm based on Butterfly Resampling previously described in the literature. We describe three interaction schemes (i) the complete interaction, (ii) the pairwise interaction where the nodes are constrained to communicate in pairs and (iii) the partial pairwise interaction in which only one pair is allowed to communicate. The goal is to diminish the communication cost in exchange for negligible loss of effective sample size. We conduct experiments on a cluster environment and compare our methods in terms of execution time, communication time and effective sample size. We find that the sparse interaction schemes show better performance for distributed systems and they keep the effective sample size nearly as high as the complete interaction scheme does.**

I. Introduction

Particle filters (PFs) have been very popular and widely used for nonlinear filtering problems since their introduction by Gordon et al. [1]. However, the computational cost of PFs increases linearly with the number of particles, which makes it difficult to satisfy real-time requirements for PFs with a large number of particles. A PF consists of three main stages: propagation, update and resampling. The propagation and the update stages are completely parallelizable, they do not require any communication between the nodes. However in the resampling stage, all the particles must interact with each other which causes communication overhead. Although resampling is inevitable for the stability of PFs [2], it is evidently the bottleneck in the distributed implementation of particle filters. Therefore, the main consideration is to minimize the communication between the nodes in designing distributed resampling algorithms, otherwise the communication overhead prevents PFs from speeding up.

Over the last decade, the distributed implementation of PFs has become popular as the need for fast real-time PFs has increased. Bolic et al. [3] introduce two categories of distributed resampling techniques: distributed resampling algorithms with proportional allocation (RPA) and nonproportional allocation (RNA) of particles. In the former, each node generates particles proportional to its total weight and performs a particle exchange algorithm for load balancing. In the latter case, the nodes perform a local resampling algorithm with a fixed number of particles and exchange some of their particles to decrease the weight variance among the nodes. Teulière et al. [4] propose an RPA with a particle transition scheme based on first matching. Míguez [5] introduces an unbiased local selection (LS) method based on RNA and provides an analysis on the importance weights of these methods. Balasingam et al. [6] suggest an optimal particle selection for RNAs. Demirel et al. [7] provide a parallel particle filtering library containing a couple of particle exchange algorithms. Recently, distributed particle filters with a proof of convergence have been introduced. Vergé et al. [8] propose the double bootstrap filter with a two level resampling method. Heine et al. [9] break down the resampling stage into a sequence of constrained interactions of particles, based on αSMC methodology described in Whiteley et al. [2].

In this work, we propose a distributed resampling algorithm based on Butterfly Resampling described in [9] with three interaction schemes. The first scheme is straightforward, the nodes can communicate with any other node and at the end of the resampling stage the weights of the particles become equal. The other two schemes are the main contributions of this paper. They are designed to restrict the communication pattern, and therefore they produce unequally weighted particles. In the second scheme, the nodes can communicate only in pairs and the pairs are formed in a way that the effective sample size (ESS) remains high. In the last scheme, only one pair is allowed to communicate at a time to prevent the communication cost from increasing as the number of the nodes increases. The rest of the paper is organized as follows: In Section II, we study distributed implementation of particle filters. In section III, we describe the interaction schemes with a generic distributed resampling algorithm. In Section IV, we show the simulation results and finally we draw conclusions in Section V.

II. Distributed Particle Filtering

Consider discrete time state space models having the following form:

$$x_0 \sim p(x_0), \quad x_t|x_{t-1} \sim p(x_t|x_{t-1}), \quad y_t|x_t \sim p(y_t|x_t)$$

ISBN 978-0-9928626-7-1

Here, x_t denotes the state vector and y_t denotes the measurement vector at time step t. The filtering problem is the estimation of $p(x_t|y_{1:t})$ i.e. the posterior distribution of the state vector given all the measurements so far. This PDF could be obtained recursively in two stages: prediction $p(x_t|y_{t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{t-1})dx_{t-1}$ and update $p(x_t|y_t) = p(y_t|x_t)p(x_t|y_{t-1})/p(y_t|y_{1:t-1})$.

A PF represents $p(x_t|y_{1:t})$ by a set of particles $\{(x_t^{(i)}, w_t^{(i)}) \mid i\in[N]\}$ where $[N]$ denotes the set $\{1, 2, \ldots, N\}$ and $w_t^{(i)}$ is the importance weight of the ith particle and N is the number of the particles. The particles are propagated according to the transition model and the weights are updated with the measurement. In practice, after a few iterations most of the weights are very close to zero and they require many computational effort although they have little effect on the result. This phenomenon is called weight degeneracy [10]. The particles must be resampled according to their weights when the variance of the weights exceeds a given threshold.

Now, suppose we are given a distributed system consisting of M nodes. The particles are distributed over M nodes, such that each node j has direct access only to $L = N/M$ particles,

$$\mathbf{x_t^{(j)}} := \{x_t^{(1,j)}, x_t^{(2,j)}, \ldots, x_t^{(L,j)}\}$$
$$\mathbf{w_t^{(j)}} := \{w_t^{(1,j)}, w_t^{(2,j)}, \ldots, w_t^{(L,j)}\}$$

In our implementation, all the nodes have the same role, there is no master or slave node and after each iteration the number of the particles on a node remains constant (L). The propagation and update parts can be performed in embarrassingly parallel fashion. A distributed PF can be summarized as follows:

Algorithm 1 Distributed Particle Filter

$x_0^{(i,j)} \sim p(x_0)$, $w_0^{(i,j)} \leftarrow 1/N$, for $i \in [L]$
for $t = 1, \ldots$ **do**
  $\hat{x}_t^{(i,j)} \sim p(x_t \mid x_{t-1}^{(i,j)})$, for $i \in [L]$
  $\hat{w}_t^{(i,j)} \leftarrow p(y_t \mid x_t^{(i,j)}) w_{t-1}^{(i,j)}$, for $i \in [L]$
  $\mathbf{x_t^{(j)}}, \mathbf{w_t^{(j)}} \leftarrow \text{DistributedResampler}(\hat{\mathbf{x}}_\mathbf{t}^{(\mathbf{j})}, \hat{\mathbf{w}}_\mathbf{t}^{(\mathbf{j})})$
end for

Here, j is the rank of the current node. *DistributedResampler* is a generic function which takes the particles and their weights as input parameters, performs a distributed resampling algorithm and returns the resampled particles with the new weights. We note that *DistributedResampler* does not have to perform complete resampling and therefore the new weights may not equal $1/N$.

III. Distributed Resampling

In this section, we briefly explain Butterfly Resampling and its parameters, then we describe the interaction schemes in terms of these parameters and finally we give a distributed resampling algorithm which takes the interaction schemes as parameters.

A. Butterfly Resampling

Butterfly Resampling is an instance of Augmented Resampling described in the same article [9]. Augmented Resampling is performed in K steps and in each step the particles can interact only a small number of other particles.

Now, let $\{A_k \mid k = 1, \ldots, K\}$ be a set of doubly stochastic transition matrices, each of size $N \times N$ and $(A_K, A_{K-1}, \ldots, A_1)^{i,j} = 1/N$. These matrices are representations of the conditional independence structure of the particles. The nonzero elements $A_k^{i,j} \neq 0$ denote the interaction between ith and jth particle. The new particles are sampled according to these interaction matrices by using the following procedure:

Algorithm 2 Augmented Resampling

$\hat{w}_{t,0}^{(i)} \leftarrow \hat{w}_t^{(i)}$, $\hat{x}_{t,0}^{(i)} \leftarrow \hat{x}_t^{(i)}$, for $i \in [N]$
for $k = 1, \ldots, K$ **do**
  for $i = 1, \ldots, N$ **do**
    $\hat{w}_{t,k}^{(i)} \leftarrow \sum_j A_k^{i,j} \hat{w}_{t,k-1}^{(j)}$
    $\hat{x}_{t,k}^{(i)} \sim \dfrac{\sum_j A_k^{i,j} \hat{w}_{t,k-1}^{(j)} \delta_{\hat{x}_{t,k-1}^{(i)}}}{\hat{w}_{t,k}^{(i)}}$
  end for
end for
$w_{t+1}^{(i)} \leftarrow \hat{w}_{t,K}^{(i)}$, $x_{t+1}^{(i)} \leftarrow \hat{x}_{t,K}^{(i)}$, for $i \in [N]$

The symbol δ_x denotes the Dirac (unit) delta measure located at x. There are many ways to choose the interaction matrices, for example when $K = 1$ and $A_1^{i,j} = 1/N$ for $, i, j \in [N]$, we obtain the standard complete resampling method. In Butterfly Resampling, the main idea is to break down the complete interaction matrix into well structured sparse matrices so that non-trivial limits for the moments can be established [9]. Let $N = r_1 r_2 \ldots r_K$ be the factorization of N, then A_k matrices of Butterfly Resampling are formed as follows:

$$A_k := I_{r_K} \otimes \cdots \otimes I_{r_{k+1}} \otimes U_{r_k} \otimes I_{r_{k-1}} \otimes \cdots \otimes I_{r_1}$$

where the symbol $\otimes$ denotes Kronecker product and U_{r_k} is $r_k \times r_k$ matrix which has $1/r_k$ as every entry. We can see the matrices satisfy the condition $\prod_k A_k = U_N$ and each matrix A_k ensures that the particles interact in small groups consisting of r_k particles at step k.

B. Complete Interaction Scheme (CIS)

In this method, we factorize N as $L \times M$. Resampling is completed in two steps ($K = 2$) and the steps use the interaction matrices $A_1 = I_M \otimes U_L$ and $A_2 = U_M \otimes I_L$ respectively. We can easily see that the particles in the same node fully interact with each other and there is no interaction between the particles belonging different nodes in A_1, thus the first step requires no communication. On the other hand, in the second step the particles need to interact across the nodes. However, since after the first step the particles in the same node have equal weights, the implementation of this step is not complicated. The details are explained in Section III-E.

ISBN 978-0-9928626-7-1 © EURASIP 2017

C. Pairwise Interaction Scheme (PIS)

Here, similar to the previous scheme r_1 equals L. In this case we factorize the second radix $r_2 = M$ as $2 \times 2 \times \ldots \times 2$ and the number of steps K becomes $m+1$ where $2^m{=}M$. For instance if $M = 4$ the matrices become,

$$A_2 = \frac{1}{2L}\begin{bmatrix} I_L & I_L & \cdot & \cdot \\ I_L & I_L & \cdot & \cdot \\ \cdot & \cdot & I_L & I_L \\ \cdot & \cdot & I_L & I_L \end{bmatrix}, A_3 = \frac{1}{2L}\begin{bmatrix} I_L & \cdot & I_L & \cdot \\ \cdot & I_L & \cdot & I_L \\ I_L & \cdot & I_L & \cdot \\ \cdot & I_L & \cdot & I_L \end{bmatrix}$$

The nodes are constrained to communicate in pairs. The disadvantage of this technique is that to proceed to the next step, the nodes have to wait the other nodes to finish the previous step i.e. there are implicit communication barriers between the steps. These implicit barriers significantly reduce the performance of the resampling stage. However, in most of the scenarios, the complete resampling is not needed. Partial mixing of the weights could be enough to keep ESS above a given threshold. Therefore we can stop resampling at earlier steps. In Pairwise Interaction Scheme, we perform only two steps, which corresponds to applying A_1 and one of the other 2-radix matrices. The second matrix to be used is chosen in a deterministic way in any order e.g. $A_2A_1, A_3A_1, \ldots, A_KA_1$. In this way, we get rid of the implicit barriers and we obtain a more sparse interaction scheme. Furthermore, the butterfly pattern distributes the weights over the nodes in m time steps thereby providing a good weight balance.

D. Partial Pairwise Interaction Scheme (PPIS)

The second step of PIS requires $M/2$ communicating pairs, which could still introduce communication overhead preventing scaling up. Therefore we further break down the matrices A_k, $k{>}1$ of PIS into more sparse matrices. Here we only explain how the matrix A_2 is factorized, but the method can be generalized without difficulty for the rest of the matrices. We know $A_2 = I_{M/2} \otimes U_2 \otimes I_L$ and we define

$$\hat{A}_{2,l} := \begin{bmatrix} I_{2M(l-1)} & \cdot & \cdot \\ \cdot & U_2 \otimes I_L & \cdot \\ \cdot & \cdot & I_{N-2Ml} \end{bmatrix}$$

We can see that $A_2 = \prod_l^{M/2} \hat{A}_{2,l}$. The second matrix is chosen from these sparse matrices $\{\hat{A}_{i,l} \mid i \neq 1, i \in [K], l \in [M/2]\}$. The key idea here is that only one pair is able to communicate during resampling. Even though this scheme cannot keep ESS as high as PIS does, the communication cost introduced is significantly small because each node communicates every $M/2$ time steps.

E. Distributed Resampling Algorithm

In this section, we describe a distributed resampling algorithm which applies the interaction schemes described above. The schemes essentially use two interaction matrices A_1 and B. The first matrix A_1 is fixed for all the schemes and it equals $I_M \otimes U_L$. The schemes differ in the second interaction matrix B which inherently specifies the communication pattern of the nodes. Let $\mathcal{P}_t^{(j)}$ denote the set of the nodes that jth node is able to communicate with in time step t. In CIS, $\mathcal{P}_t^{(j)}$ is the whole node set, in PIS, $\mathcal{P}_t^{(j)}$ consists of jth node and its target node, and in the last interaction scheme (PPIS), $\mathcal{P}_t^{(j)}$ is mostly formed by only jth node, but it contains also the target node every $M/2$ time steps. Applying A_1 is straightforward; each node performs a local resampling. After that, the weights of the particles belonging to the same node become identical. By exploiting this feature, the second step can be easily performed by calculating the particle duplicate numbers (offspring number) of the nodes. Let $\mathbf{L}$ denote the set $\{L^{(j)} \mid j \in \mathcal{P}_t^{(j)}\}$. Here $L^{(j)}$ is the offspring number of the jth node after the second step satisfying the condition $\sum_{j\in\mathcal{P}_t^{(j)}} L^{(j)} = \#\mathcal{P}_t^{(j)} \cdot L$ where the symbol # denotes the cardinality of a set. $\mathbf{L}$ can be calculated by using the standard resampling techniques such as multinomial and residual resampling [11].

The difference between $L^{(j)}$ and L determines whether jth node is receiver or sender. In other words, if $\hat{L}^{(j)} = L^{(j)} - L$ has a positive value, jth node has $\hat{L}^{(j)}$ surplus of particles to be sent, otherwise it means jth node receives $|\hat{L}^{(j)}|$ particles. A matching algorithm is needed to match the nodes with surplus of particles with the nodes with shortage of particles. A good matching should minimize the number of communicating pairs. Here, we describe a greedy matching algorithm, however other approaches also can be used [7]. Let $T(\tau, \nu)$ represent the number of the particles to be sent from τ to ν. The matching algorithm takes a sender τ with a particle surplus $\hat{L}^{(\tau)}$ and schedules a transmission of $min(\hat{L}^{(\tau)}, |\hat{L}^{(\nu)}|)$ particles to the first receiver ν and the values of $T(\tau, \nu)$, $\hat{L}^{(\tau)}$ and $\hat{L}^{(\nu)}$ are updated accordingly. The algorithm iterates over the receivers until $\hat{L}^{(\tau)}$ becomes zero. Once the transmission of the particles is finished, the weights of the particles are set to $\dfrac{\sum_{j\in\mathcal{P}_t^{(j)}} W^{(j)}}{\#\mathcal{P}_t^{(j)} \cdot L}$.

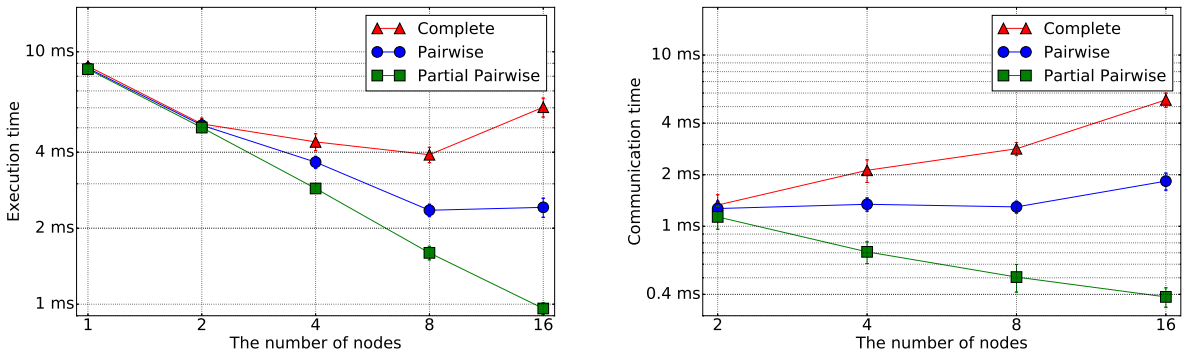
Algorithm 3 Distributed Resampling Algorithm

$\tilde{\mathbf{x}}_\mathbf{t}^{(\mathbf{j})} \leftarrow \text{ResampleLocally}(\hat{\mathbf{x}}_\mathbf{t}^{(\mathbf{j})}, \hat{\mathbf{w}}_\mathbf{t}^{(\mathbf{j})})$
$\mathbf{W} \leftarrow \text{GatherWeights}(\mathcal{P}_t^{(j)})$
$\mathbf{L} \leftarrow \text{OffspringNumbers}(\mathbf{W}, \#\mathcal{P}_t^{(j)} \cdot L)$
$\hat{L}^{(i)} = L^{(i)} - L, \quad \text{for } i \in \mathcal{P}_t^{(j)}$
$T \leftarrow \text{Match}(\hat{\mathbf{L}})$
if $\hat{L}^{(j)} > 0$ **then**
  $\text{Send}(T(j, \nu)), \quad \text{for } \nu \in \mathcal{P}_t^{(j)}$
else
  $\text{Receive}(T(\tau, j)), \quad \text{for } \tau \in \mathcal{P}_t^{(j)}$
end if
$\hat{w}_t^{(i,j)} \leftarrow \dfrac{\sum_{j\in\mathcal{P}_t^{(j)}} W^{(j)}}{\#\mathcal{P}_t^{(j)} \cdot L}, \quad \text{for } i \in [L]$

IV. Experimental Results

In this section, we present our simulation results. We compare the interaction schemes in terms of execution time, communication time and effective sample size. We implemented our methods in C++ with Open MPI [12], and we

ISBN 978-0-9928626-7-1 © EURASIP 2017



(a) Average execution time of a single time step.

(b) Average communication time of a single time step.

Fig. 1: The comparison of performances of the interaction schemes. ($N = 8192$)

used a cluster consisting of nodes having Intel(R) Xeon(R) CPU E5-2650L v3 @ 1.80GHz. We ran the simulations on 16 separate cores each belonging to a different node.

A. Lorenz System

We use the application of tracking the state of the chaotic Lorenz system to evaluate our methods. The Lorenz system is non-periodic and highly nonlinear, and small errors in its estimated state result in larger errors in a later state [13]. We use the discrete time version as the state transition model in our simulations described in [5]. The state vector is 3-dimensional $(x_{1,t}, x_{2,t}, x_{3,t})$ and it is updated over time according to the following propagation equations:

$$x_{1,t} = x_{1,t-1} - ST(x_{1,t-1} - x_{2,t-1}) + \sqrt{T}u_{1,t}$$
$$x_{2,t} = x_{2,t-1} + T(Rx_{1,t-1} - x_{2,t-1} - x_{1,t-1}x_{3,t-1}) + \sqrt{T}u_{2,t}$$
$$x_{3,t} = x_{3,t-1} - T(Bx_{3,t-1} - x_{1,t-1}x_{2,t-1}) + \sqrt{T}u_{3,t}$$

and we can only make noisy measurements on the first dimension.

$$y_t = x_{1,t} + \nu_k$$

S, R, B are the system constants and their default values are $10, 28, 8/3$ respectively. T is the time step which equals 10^{-2} time units and $\nu_t, u_{i,t}$ are zero-mean Gaussian white noise with the variance $\sigma^2 = 1$. The initial state is distributed from a Gaussian distribution with the standard mean $(-5.91652, -5.52332, 24.5723)$ and the covariance $10^{-2} \cdot I_3$.

B. Execution and Communication Time

We ran our simulations for 4096 time steps and $N = 8192$ particles. We recorded the filtering time and the amount of time spent in communication. Then we calculated the average execution time and communication time for a single time step. The results are plotted in Fig. 1. The communication time in CIS increases almost linearly with the number of nodes M. It clearly hinders the speed-up to be gained through the parallelization. The underlying reason for this problem is that the number of transmitted particles increases as M grows [14].

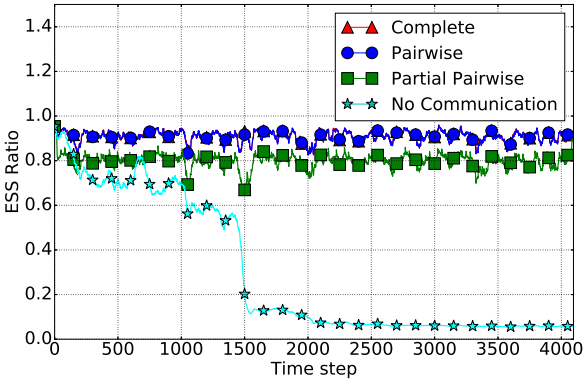


Fig. 2: Effective sample size ratio against time for $N = 8192$ and $M = 16$. The ESS curves of Complete and Pairwise Interaction Schemes overlap. (ESS ratio is obtained by ESS/N.)

Besides, there is no restriction on the communication pattern. It is possible for a node to communicate with many other nodes in a time step, which may bring about an unbalanced communication load. In PIS, the nodes are constrained to communicate in disjoint pairs, therefore the communication cost introduced is less than that in CIS. However increasing M also increases the number of communicating pairs and at some point it starts to lower the performance of the particle filter. In PPIS, there are only two communicating nodes at any time step regardless of M, that is increasing M does not impose an additional communication cost. As a result, PPIS benefits the most from linearly decreasing number of local particles (L).

C. Effective Sample Size

In Fig. 2 we show the effective sample size of the interaction schemes in each time step. The results are obtained for $M = 16$ nodes and $N = 8192$ particles. For better visualization, ESS ratio is smoothed by a 10-step moving average filter. We see that ESS curves of PIS and CIS are so close that they overlap, which confirms that the complete

ISBN 978-0-9928626-7-1 © EURASIP 2017

interaction is indeed not needed for many models including highly nonlinear models such as the Lorenz System. In PPIS, ESS is lower than those in the other schemes, and the variance of ESS is relatively high. This is the price to pay for sparse communication. Lastly, we observe that running parallel PFs without communication is not stable as stated in [2]. If the nodes do not communicate with each other, eventually one node dominates the others by holding all the weights and ESS decreases below N/M.

V. Conclusion

In this article, we propose three different interaction schemes for distributed resampling. First, we describe a simple interaction scheme where the nodes fully interact with each other and at the end of the resampling stage all the particles have equal weights. Second, we put some constraints on the communication patterns such that the nodes communicate only in pairs during resampling. In this way, the communication overhead is significantly reduced and the effective sample size is kept high. In the last scheme, we further make the interaction more sparse; only one pair is allowed to communicate during resampling. Although the effective sample size is affected negatively, this scheme is practical when the number of the nodes is high (say more than 8). The experimental results show that the communication overhead makes Complete Interaction Scheme impractical. Pairwise interaction Scheme keeps the effective sample size as high as the complete resampling does and it significantly decreases the communication time. The last method, Partial Pairwise Interaction Scheme, yields the best performance in terms of execution time, however the effective sample size is reduced noticeably and its variance is relatively high.

References

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

[2] N. Whiteley, A. Lee, and K. Heine, "On the role of interaction in sequential Monte Carlo algorithms," *Bernoulli*, vol. 22, no. 1, pp. 494–529, 02 2016. [Online]. Available: http://dx.doi.org/10.3150/14-BEJ666

[3] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2442–2450, July 2005.

[4] V. Teulière and O. Brun, "Parallelisation of the Particle Filtering Technique and Application to doppler-bearing tracking of maneuvering sources," *Parallel Comput.*, vol. 29, no. 8, pp. 1069–1090, Aug. 2003.

[5] J. Míguez, "Analysis of Parallelizable Resampling Algorithms for Particle Filtering," *Signal Process.*, vol. 87, no. 12, pp. 3155–3174, Dec. 2007.

[6] B. Balasingam, M. Bolić, P. M. Djurić, and J. Míguez, "Efficient distributed resampling for particle filters," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 3772–3775.

[7] Ö. Demirel, I. Smal, W. J. Niessen, E. Meijering, and I. F. Sbalzarini, "PPF — A parallel particle filtering library," in *IET Conference on Data Fusion Target Tracking 2014: Algorithms and Applications (DF TT 2014)*, April 2014, pp. 1–8.

[8] C. Vergé, C. Dubarry, P. Del Moral, and E. Moulines, "On Parallel Implementation of Sequential Monte Carlo Methods:The Island Particle Model," *Statistics and Computing*, vol. 25, no. 2, pp. 243–260, Mar. 2015. [Online]. Available: http://dx.doi.org/10.1007/s11222-013-9429-x

[9] K. Heine, N. Whiteley, A. T. Cemgil, and H. Guldas, "Butterfly resampling: asymptotics for particle filters with constrained interactions," vol. arXiv:1411, 2014.

[10] O. Cappe, S. J. Godsill, and E. Moulines, "An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, May 2007.

[11] R. Douc and O. Cappe, "Comparison of resampling schemes for particle filtering," in *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, Sept 2005, pp. 64–69.

[12] (2017) Open MPI: v1.10 nightly snapshot tarballs. [Online]. Available: https://www.open-mpi.org/nightly/v1.10/

[13] E. N. Lorenz, "Deterministic Nonperiodic Flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963. [Online]. Available: https://doi.org/10.1175/1520-0469(1963)020¡0130:DNF¿2.0.CO;2

[14] N. Santoro, *Design and Analysis of Distributed Algorithms (Wiley Series on Parallel and Distributed Computing)*. Wiley-Interscience, 2006.

ISBN 978-0-9928626-7-1 © EURASIP 2017