HIERARCHICAL RESAMPLING ARCHITECTURE FOR DISTRIBUTED PARTICLE FILTERS

Ning Zheng, Yun Pan, and Xiaolang Yan

Institute of VLSI Design
Zhejiang University, China
{zhengning, panyun, yan}@vlsi.zju.edu.cn

Ruohong Huan

College of Computer Science and Technology
Zhejiang University of Technology, China
huanrh@zjut.edu.cn

ABSTRACT

In this paper, a hierarchical resampling (HR) architecture has been presented for distributed particle filters (PFs). The proposed architectures decomposes the resampling step into two hierarchies, of which the first one, called intermediate resampling, is conducted consecutively among processing elements (PEs) the moment new particles and their weights are generated by each PE, and the second one, named unitary resampling, is performed sequentially after the whole intermediate resampling procedure and shared by all PEs. Compared with traditional distributed architectures, the HR architecture eliminates the particle redistribution step, and has such advantages as short execution time, high memory efficiency and well scalability.

Index Terms—Architecture, distributed particle filters, hierarchical resampling, latency, high memory efficiency

1. INTRODUCTION

Particle filters (PFs) have achieved, since their original introduction [1], great popularity as a powerful approach in addressing nonlinear and/or non-Gaussian dynamic systems modeled in state-space [2–6]. It is a Monte Carlo simulation and Bayesian estimation based methodology, where the basic idea is a recursive approximation of relative posterior probability distributions with a set of discrete random samples.

Of the three steps involved in general PFs, i.e. sampling, weight calculation and resampling, the last one requires a joint processing of all particles due to its sequential nature, and thus affects significantly the execution time. Various architectures have been explored for centralized resampling [7–10], though, it remains a bottleneck to implement resampling efficiently for distributed PFs. Therefore, development of a resampling architecture suitable for distributed PFs becomes the key to accelerate further the filter processing, and has gained considerable interest in the literature [11–13].

Despite the improvement in speed, current distributed implementations of PFs suffer from the following potential major drawbacks:

1) Particles need to be redistributed after resampling in each processing element (PE), which is non-deterministic and accordingly limits the hardware scalability.

2) The whole realization demands too much hardware cost, especially the memory resource.

3) The accuracy of estimation is no longer guaranteed due to the trade-off between the performance and the complexity.

In this paper, we propose a hierarchical resampling (HR) scheme and the corresponding architecture for distributed PFs. In HR, the resampling step is decomposed into two hierarchies. The first one, called intermediate resampling, is conducted continuously among PEs the moment new particles and relevant weights are generated by each PE; the second, named unitary resampling, is performed after the whole intermediate resampling procedure and shared by all PEs. HR has similar resampling result to standard resampling in statistics, yet it results in a hardware architecture for distributed PFs with a couple of encouraging features: first, resampling is decomposing into two more granular hierarchies, both of which can be realized in a pipeline fashion; second, redistribution of particles is eliminated, which leads to a simple control scheme and a well architecture scalability; third, unitary resampling has fewer weights to handle and can be shared by all PEs, which reduces significantly the execution time as well as the hardware cost.

The remainder of the paper is organized as follows. Section 2 reviews briefly the main existing distributed resampling schemes, followed by the HR scheme and its corresponding architecture in Section 3. Section 4 analyzes the performance of the architecture in terms of execution time, memory utilization, as well as scalability, and Section 5 concludes the paper.

2. DISTRIBUTED RESAMPLING SCHEMES

The most challenging issue in distributed PFs is how to redistribute particles among PEs after resampling. Such redistribution results from the random distribution of particle weights, and is necessary to balance the workload of each PE in the next recursion. Current distributed resampling schemes mainly include two strategies: resampling with proportional allocation (RPA) and resampling with non-proportional allocation (RNA) [11].

RPA is such a method that the number of particles each PE replicates is firstly calculated, known as inter-resampling, and then resampling is performed in parallel inside each PE, known as intra-resampling. The main problem of RPA is the unpredictability in execution time of particle redistribution after intra-resampling due to the randomicity of particle numbers in each PE. Besides, the sizes of memories must be designed for the worst case for each PE, which increases the total memory requirement sharply. The concept of RPA is shown in Fig.1 (a), where $K = 4$ PEs and $N = 200$ particles are employed, and a central unit (CU) is used for system control. In this case, 30 surplus particles need to be transferred from PE_0, and 20 from PE_3.

In RNA, on the other hand, the same number of particles is

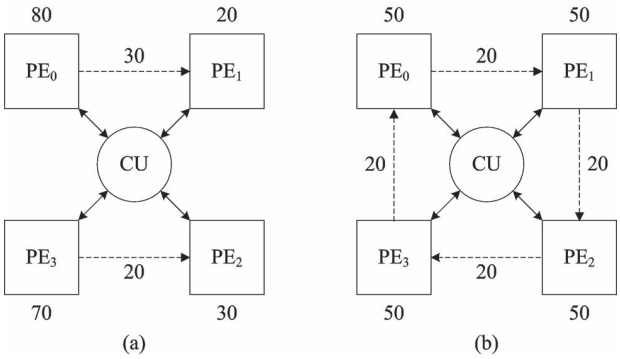

Fig.1. Concepts of (a) RPA and (b) RNA.

resampled in each PE, and then partial particles are exchanged in a deterministic way among neighboring PEs. The amount of particles sent between PEs is fixed and defined in advance. In spite of the simplicity, RNA has a inherent problem that the performance of estimation can no longer be guaranteed since particles have various weights both inside and among PEs after redistribution. The concept of RNA is shown in Fig.1 (b), where 50 particles are resampled in all PEs and 20 particles are transferred between adjacent PEs.

3. HIERARCHICAL RESAMPLING

In this section, a hierarchical resampling method is introduced. Differing from common distributed resampling schemes that start resampling after all particles and weights are generated, HR decomposes the resampling step into two hierarchies and starts as soon as the first batch of particles and their weights are prepared in PEs.

3.1. Hierarchical Resampling Scheme

Consider a distributed implementation of PFs with N particles and K PEs. The sampling step and weight calculation step are parallelized in PEs. When K new particles and weights are generated simultaneously, a resampling process is conducted with the resampled particles transferring to each PE; meanwhile the sum of the weights is stored as the collective weight for each resampled particle. The above operation is repeated N/K times till all N particles are processed and N/K collective weights are preserved. This procedure constitutes the first hierarchy of HR and is called intermediate resampling as it is performed on the basis of only a part of particles and weights each time. Afterwards, a unitary resampling step, the second hierarchy of HR, is started which resamples particles in each PE according to the collective weights. Since particles in the same storage positions of each PE are in fact resampled ones with equal weights, the unitary resampling can then be shared among all PEs. An overall description of distributed PFs with the HR scheme is given by Algorithm 1.

In standard resampling, e.g. systematic resampling, a particle x_t^i, representing the i^{th} particle generated at time index t, is resampled with a replication factor (replicated times) r_t^i that is proportional to its weight w_t^i. A mathematical representation can be expressed as

$$E(r_t^i) = \frac{Nw_t^i}{\sum_{j=1}^{N} w_t^j}, \quad i = 1,2,...,N \tag{1}$$

Algorithm 1: Distributed PFs with hierarchical resampling

- **for** $i = 1 : N/K$
  - **do in parallel** in K PEs
    - Sample a new particle $x_t^{i,j}$ from the proposal distribution
      $$x_t^{i,j} \sim q(x_t \left| x_{t-1}^{i,j}, z_t \right.), j = 1,2,...,K$$
    - Calculate the corresponding weight $w_t^{i,j}$:
      $$w_t^{i,j} = \frac{p(z_t \left| x_t^{i,j} \right.) p(x_t^{i,j} \left| x_{t-1}^{i,j} \right.)}{q(x_t^{i,j} \left| x_{t-1}^{j,j}, z_t \right.)}$$
  - **end**
  - Calculate the collective weight:
    $$W_t^i = \sum_{j=1}^{K} w_t^{i,j}$$
  - Resample according to $\{ x_t^{i,j}, w_t^{i,j} \}_{j=1}^{K}$
  - **for** $j = 1 : K$
    - Send the j^{th} resampled particle back to PE_j
  - **end for**
- **end for**
- Resample particles according to the N/K collective weights for each PE

where $E(\cdot)$ is the expectation function. In HR, a particle $x_t^{i,j}$, standing for the i^{th} particle generated by PE_j at time index t, is first resampled by intermediate resampling which gives its intermediate replication factor $r'^{i,j}_t$ as

$$E(r'^{i,j}_t) = \frac{Kw_t^{i,j}}{W_t^i}, \quad i = 1,2,...,\frac{N}{K}, \quad j = 1,2,...,K \tag{2}$$

where

$$W_t^i = \sum_{m=1}^{K} w_t^{i,m}$$

denotes the collective weight, and then resampled by unitary resampling which determines the expectation of its final replication factor $r_t^{i,j}$ as

$$\begin{aligned} E(r_t^{i,j}) &= E(r'^{i,j}_t) \cdot \frac{N}{K} \cdot \frac{W_t^i}{\sum_{m=1}^{N/K} W_t^m} \\ &= \frac{Kw_t^{i,j}}{W_t^i} \cdot \frac{N}{K} \cdot \frac{W_t^i}{\sum_{m=1}^{N/K} W_t^m} \\ &= \frac{Nw_t^{i,j}}{\sum_{m=1}^{N/K} \sum_{n=1}^{K} w_t^{m,n}}, \quad i = 1,2,...,\frac{N}{K}, \quad j = 1,2,...,K \end{aligned} \tag{3}$$

It can be seen that for a specific particle, the expectation of its replication factor in HR is identical to that in standard resampling. Indeed, a particle weight is utilized here in such two respects as follows: it is first used directly for intermediate resampling of K particles, and then employed indirectly for unitary resampling of N/K particles in each PE. This ensures a lossless performance in HR in statistics.

3.2. Hierarchical Resampling Architecture

The overall architecture of HR based distributed PFs is shown in Fig.2, where K PEs are connected to the HR unit. For illustration,

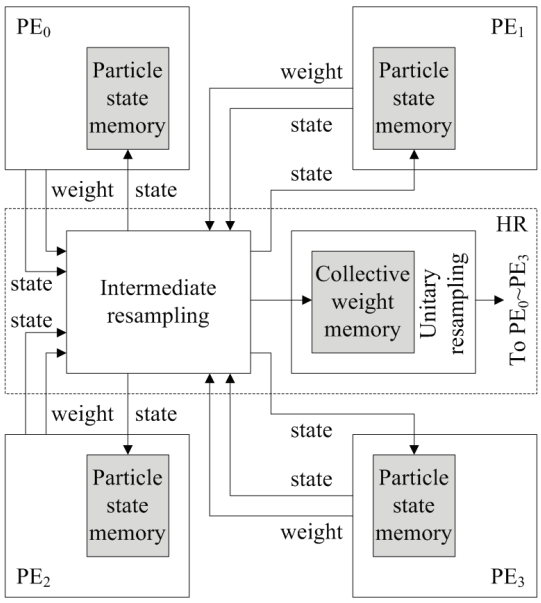

Fig.2. Architecture of HR based distributed PFs.

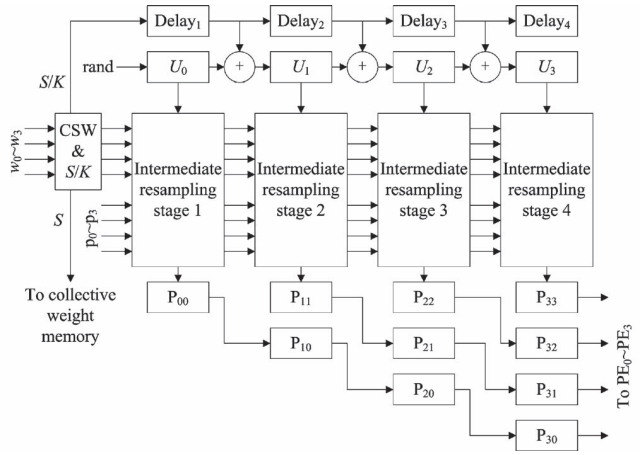

Fig.3. Architecture of intermediate resampling.

$K = 4$. Particles propagated per cycle in each PE, denote as p_0~p_3, are resampled in the intermediate resampling unit with a pipeline fashion according to their weights w_0~w_3 based on the concept of systematic resampling. Resampled particles are then sent back to the particle state memory of each PE; meanwhile the weight sum is stored in the collective weight memory in the unitary resampling unit. When N/K collective weights are prepared, unitary resampling starts and performs in a way the same as that in the centralized implementation. The final resampling outputs are shared by all PEs and depend on the architecture used for unitary resampling, e.g. the outputs are particle indexes and replication factors if the residual systematic resampling (RSR) based architecture [7] is used.

Fig.3 shows the architecture of intermediate resampling according with Fig.2. Before resampling, all temporary sum of weights (TSW), i.e. the sum of the first k ($1 \le k \le 4$) weights, are calculated. At the same time, the weight mean, denoted by S/K where S is the sum of all weights, is provided to update the temporary value of the resampling function U. In each intermediate resampling stage, as is shown in Fig.4, all TSWs are compared with the current U_i ($0 \le i \le$

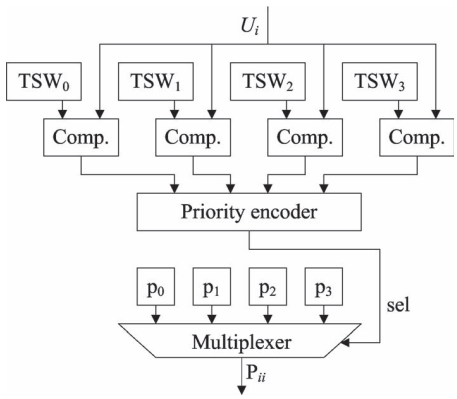

Fig.4. Architecture of one intermediate resampling stage.

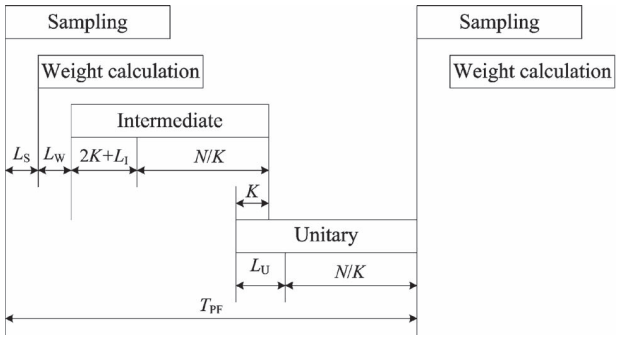

Fig.5. Timing of the HR based architecture for distributed PFs.

3) and one resampled particle P_{ii} is then selected by the subscript of the first TSW that is no less than U_i. After K stages, each resampled particle is transferred to the corresponding PE.

PE carries out the sampling and weight calculation steps, of which the architecture is cooperating with that of unitary resampling, see [7].

4. PERFORMANCE ANALYSIS

The performance of the proposed architecture is analyzed in this section from the aspects of execution time, memory utilization and scalability. Besides, the chaotic Lorenz system model (CLS) [14] is simulated as a case study.

The timing of the HR based architecture is illustrated in Fig. 5, where L_S, L_W, L_I and L_U represent the startup latencies of sampling, weight calculation, intermediate resampling, and unitary resampling, respectively. As the first resampling hierarchy is pipelined with the sampling and weight calculation steps, the execution time of one whole filter iteration is much close to that of a centralized architecture with N/K particles. If we use L to refer to all startup latencies which are limited and can be neglected when N/K is large enough, the total cycle time of the proposed architecture can be given as $2N/K + K + L$. Notice that unitary resampling is started the moment all collective weights are stored, which is K cycles before intermediate resampling is finished. Since K is generally much smaller than N/K, the processing time can be considered to scale with the number of PEs. The execution time of the RPA based and RNA based architectures is also summarized, as is shown in Table 1. For a fair comparison, the sampling step and the weight calculation step are all assumed to be pipelined, and the resampling (unitary

Table 1. Comparison of different architectures for distributed PFs

Arch.	Execution time	Memory utilization
RPA	$2N/K + K + \log_2 K + L + L_{\mathrm{RPA}}$	$N(KP + W + 2\log_2 N)$
RNA	$2N/K + L + L_{\mathrm{RNA}}$	$N(P + W + 2\log_2(N/K))$
HR	$2N/K + K + L$	$NP + N/K(W + 2\log_2(N/K))$

resampling in HR) step is realized with RSR based architecture. Also, L is used as the total startup latencies for simplicity. In the RPA based architecture, L_{RPA} is the variable time for the particle redistribution, which may become dominant in a high level of parallelism. In the RNA based architecture, L_{RNA} denotes the delay of the local particle exchange among PEs. The proposed architecture eliminates the redistribution of particle, thus performs with a definite period which is less than that of the other two architectures.

The memory utilizations of the three architectures are also included in Table 1, where the symbols, P and W, stand for the word lengths used to represent particle states and weights separately, and the logarithmic values denote the widths of replication factors and particle indexes. Since the unitary resampling in HR is shared by all PEs, the memory used for resampling in the proposed architecture is only $1/K$ of the other two, leading to a minimum utilization of the three. On the other hand, the RPA based architecture needs a size of the particle state memory K times of that in the other two, increasing the memory requirement tremendously. This is because the state memory size in RPA must be designed for the worst case due to the non-deterministic redistribution of particles after resampling.

Furthermore, the proposed architecture features well scalability, profiting from a convenient reuse of PE and the unitary resampling unit, as well as the regular structure of the intermediate resampling unit.

Fig.6 shows the simulation results of the three schemes with 8 PEs when the CLS model is applied. The mean of the root mean square error (RMSE) is used as the criterion of evaluation. As can be expected, the proposed scheme performs close to RPA and better than RNA.

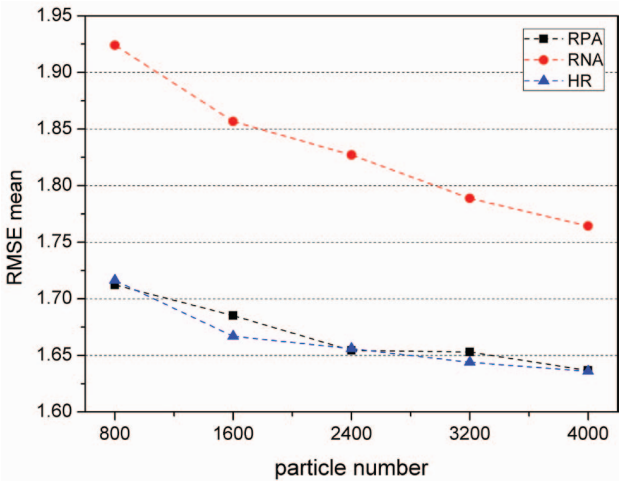

Fig.6. Comparison of RMSE mean of different schemes.

5. CONCLUSION

Presented in this paper is a hierarchical architecture for distributed PFs. By decomposing resampling into two hierarchies, the proposed architecture removes the tricky particle redistribution procedure, and simplifies the complex control logic. With high performance and no accuracy loss, it surpassed the conventional RPA based and RNA based alternatives in terms of short execution time, high memory efficiency and well scalability.

6. REFERENCE

[1] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Proc. Inst. Elect. Eng., F,* vol. 140, no. 2, pp. 107–113, 1993.

[2] A. Doucet and X. -D. Wang, "Monte Carlo methods for signal processing: a review in the statistical signal processing context," *IEEE Signal Process. Mag*, vol. 22, no. 6, pp. 152–170, 2005.

[3] P.M. Djuric, J.H. Kotecba, J. Zhang, Y. Huang, T. Gbirmai, M.F. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Process. Mag*, vol. 20, no. 5, pp. 19–38, Sep. 2003.

[4] S. Saha, N. K. Bambha, and S. S. Bhattacharyya, "Design and implementation of embedded computer vision system based on particle filters," *Comput. Vision Image Understanding*, vol. 114, no. 11, pp. 1203–1214, 2010.

[5] F. Gustafsson, F. Gunnarsson, N. Fergman, U. Rofssell, J. Jamsson, R. Karlsson, and P. -J. Nordlund, "Particle Filters for Positioning, Navigation and Tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 425–437, 2002.

[6] H. Zhou, and S. Sakane, "Sensor planning for mobile robot localization – a hierarchical approach using a Bayesian network and a particle filter," *IEEE Trans. Rob.*, vol. 24, no. 2, pp. 481–487, 2008.

[7] A. Athalye, M. Bolic, S. Hong, and P. M. Djuric, "Generic hardware architectures for sampling and resampling in particle filters," *EURASIP J. Appl. Signal Process.*, vol. 17, pp. 2888–2902, 2005.

[8] S. Hong, M. Bolic, and P. M. Djuric, "An efficient fixed-point implementation of residual resampling scheme for high-speed particle filters," *IEEE Signal Process. Lett.*, vol. 11, no. 5, pp. 482–485, 2004.

[9] S. -H. Hong, Z. -G. Shi, J. -M.. Chen, and K. -S. Chen, "A low-power memory-efficient resampling architecture for particle filters," *Circuit Syst Signal Process.*, vol. 29, no. 1, pp. 155–167, 2010.

[10] N. Zheng, Y. Pan, X. Yan, and R. Huan, "Local weight mean comparison scheme and architecture for high-speed particle filters", *Electron. Lett.*, vol. 47, no. 2, pp. 142–144, 2011.

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

[12] S. Hong, M. Bolic, and P. M. Djuric, "High-throughput scalable parallel resampling mechanism for effective redistribution of particles," *IEEE Trans. Signal Process.*, vol. 54, no. 3, pp. 1144–1155, 2006.

[13] A. C. Sankaranarayanan, A. Srivastava, and R. Chellappa, "Algorithmic and architectural optimizations for computationally efficient particle filtering," *IEEE Trans. Image Process.*, vol. 17, no. 5, pp. 737–747, 2008.

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