An Efficient Fixed-Point Implementation of Residual Resampling Scheme for High-Speed Particle Filters

Sangjin Hong, *Member, IEEE*, Miodrag Bolić, *Student Member, IEEE*, and Petar M. Djurić, *Senior Member, IEEE*

***Abstract*—A novel low-complexity residual resampling scheme for particle filters is presented. The proposed scheme uses a simple but effective "particle-tagging" method to compensate for a possible error that can be caused by finite-precision quantization in the resampling step of particle filtering. The scheme guarantees that the number of particles after resampling is always equal to the number of particles before resampling. The resulting scheme is suitable for high-speed physical realization when the number of particles is a power of two.**

***Index Terms*—Fixed-point processing, particle filters, resampling, residual resampling.**

I. Introduction

PARTICLE filters are used in nonlinear signal processing where the interest is in tracking and/or detection of random signals. Particle filters base their operations on representing relevant densities by discrete random measures composed of particles and weights and compute integrals by Monte Carlo methods [1]. More specifically, at every time instant t a random measure $\{x_{1:t}^{(m)}, w_t^m\}_{m=1}^M$ is defined, where $x_t^{(m)}$ is the mth particle of the signal at time t, $x_{1:t}^{(m)}$ is the mth trajectory of the signal, and $w_t^{(m)}$ is the weight of the mth particle (or trajectory) at time t. If now, for example, an estimate of $E(h(x_{1:t}) \mid (y_{1:t}))$ is needed, where $h(\cdot)$ is a function of $x_{1:t}$, the estimate can easily be computed using the random measure from

$$\hat{E}(h(x_{1:t}) \mid (y_{1:t})) = \sum_{m=1}^{M} w_t^{(m)} h\left(x_{1:t}^{(m)}\right). \tag{1}$$

The particle filters have three important operations: generation of new particles, computation of the particle weights, and resampling. The third operation is needed for resolving an important problem of particle filtering known as weights degeneration. More specifically, as time progresses, a few weights become very large and the remaining weights decrease in value to the point that they become negligible. The idea of resampling is to remove the trajectories that have small weights and to focus on trajectories that are dominating. Resampling is very important in particle filtering because it prevents the particle filter from weight degeneracy [2]–[4]. In the weight computation, a weight $w^{(m)}$ is computed for each particle $x^{(m)}$. The weights for all M particles are normalized for resampling where M is the number of particles used in the filtering. The objective of the resampling is to redistribute the particles according to the weights.

Standard algorithms used for resampling are different variants of stratified resampling. The two most common methods for resampling are systematic and residual resampling [1], [5]. Their algorithmic representation is given in [6]. For correct functioning of systematic resampling, it is necessary that the sum of all weights after normalization is equal to one. However, this condition is not satisfied in very large scale integration implementation due to the finite-precision effect. In residual resampling, the number of replicated particles is calculated first by truncation or rounding of the product $w^{(m)}M$. In the case of truncation, the number of particles produced after this step is in general less than M. Then, it is necessary to process the residues in order to compensate for the number of particles. In this letter, a residual resampling algorithm suitable for fixed-point implementation is considered. Residues are processed using a memory-addressing scheme and a tagging method, and this procedure guarantees the correct number of particles after resampling.

II. Proposed Resampling Scheme

We reiterate that the resampling process is an important integral part of particle filtering and that its performance crucially affects the overall particle filter performance. For a physical realization of particle filtering, it is critical to have an efficient mechanism that reduces the hardware complexity and maintains the filtering performance. In this section, we describe the operation of the proposed resampling scheme, where we assume that the number of particles is a power of two.

In the weight computations, before resampling, all the weights $w^{(m)}$ of the particles $x^{(m)}$ are normalized through addition and division, i.e., all the weights are added, and each weight is divided by the sum such that, after normalization, the sum of all weights is equal to one. However, the sum of one is only achievable with infinite precision. In hardware implementation with fixed-point number representation, these weights are quantized with K bits (excluding the sign bit), where $K = \log_2(M)$. A naive approach would quantize the value of the weight by simple truncation. Then, the integer representation of the K bits corresponds to the number of times the particle should be replicated. The simple truncation may result in a total number of replicated particles less than M. This is illustrated in Table I, where $M = 4$ and $K = 2$. The second column represents the decimal values of the particle weights, and the third column their binary representation with

Manuscript received May 22, 2003; revised September 26, 2003. This work was supported by the National Science Foundation under Award CCR-0220011. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Marcelo G. S. Bruno.

The authors are with the Department of Electrical and Computer Engineering, Stony Brook University—SUNY, Stony Brook, NY 11794-2350 USA (e-mail: snjhong@ece.sunysb.edu; mbolic@ece.sunysb.edu; djuric@ece.sunysb.edu).

Digital Object Identifier 10.1109/LSP.2004.826634

TABLE I
RESAMPLING WITH QUANTIZATION BY SIMPLE TRUNCATION

Particles	Weights	Quantized Weights	Replication Factor ($r^{(m)}$)
$x^{(1)}$	0.748	0.**10**	2
$x^{(2)}$	0.250	0.**01**	1
$x^{(3)}$	0.001	0.**00**	0
$x^{(4)}$	0.001	0.**00**	0

TABLE II
ROUNDING/TRUNCATION SCHEME AND TAGS

Three Bits	Rounding Scheme	Result	Tag Status
000	Truncate	**0**	none
001	Truncate	**0**	Tag3
010	Truncate	**0**	Tag2
011	Round	**1**	none
100	Truncate	**1**	none
101	Truncate	**1**	Tag3
110	Truncate	**1**	Tag2
111	Truncate	**1**	Tag1

two bits. The fourth column provides the replication factor, which is a decimal equivalent value of the K bits indicated in bold. According to the table, particle $x^{(1)}$ will be replicated twice; particle $x^{(2)}$ will be replicated once; and particles $x^{(3)}$ and $x^{(4)}$ will be eliminated. Because of the quantization, the sum of the resampled particles is not equal to four. In general, $R = \sum r^{(m)}$ may not be equal to M.

For solving this problem, one might consider using a conventional rounding method. When the rounding is employed, however, it is possible that the sum of all replicated particles be larger than M. It should be noted that both simple truncation and rounding methods complicate the hardware. For example, when the sum of all replicated particles is less than M, the hardware must decide which particles to additionally replicate so that the total number of particles is M. On the other hand, if the sum of all replicated particles is larger than M, the hardware must somehow choose some of the already replicated particles for removal. These two scenarios require additional iteration (i.e., scanning of all the weights for reevaluation) and selection of particles for additional replication or removal. Therefore, a modification is necessary for efficient hardware implementation.

In order to resolve the problem of not having M resampled particles, we propose that all the weights are quantized with two additional bits such that $K = \log_2(M) + 2$, excluding the sign bit. The two additional bits are used to create tags. Then, a final quantization is performed, which consists of rounding and truncation, as shown in Table II. The entries of the first column are the last three bits of the binary representation of the weight, the second column the applied rounding scheme, the third column the resulting least significant bit, and the last column the Tag status. Notice that the bit pattern **111** is not rounded, but tagged, since an adder is needed to incorporate carry propagation to the most significant bit. However, the bit pattern **011** is rounded where simple bit reversal is sufficient. The difference between Tag1 and Tag2 is to indicate that Tag1 has higher priority for replication. For simplicity in the implementation, however, these do not have to be distinguished, especially when the value of M is large (i.e., more than 64). The particles with Tag3 are used only when the total number of replicated particles is less than M.

TABLE III
RESAMPLING WITH THE PROPOSED SCHEME

Particles	Weights	Quantized Weights	Round/ Truncate	Replication Factor ($r^{(m)}$)
$x^{(1)}$	0.748	0.1011	0.**11**	3
$x^{(2)}$	0.250	0.0011	0.**01**	1
$x^{(3)}$	0.001	0.0000	0.**00**	0
$x^{(4)}$	0.001	0.0000	0.**00**	0

When $R = \sum r^{(m)} < M$, the tagged particles may be replicated once more. Note that $R+T_1+T_2+T_3 > M$ where T_1, T_2, and T_3 are the sums of particles with tags Tag1, Tag2, and Tag3, respectively. R has priority over T_1, T_2, or T_3, and therefore the tagged particles are overwritten in case $R = M$. A reason that tagging is used instead of rounding for all cases is to avoid a situation where R exceeds M such that it may be possible to exclude particles that are very important but located physically toward the end of the memory location (this will be described in the next section). The proposed scheme is illustrated in Table III on the same example as in Table I.

The modified quantization scheme ensures that the sum of all replication factors is closer to M than with the resampling based on quantization by simple truncation. In addition, only a single iteration (i.e., scanning of all the weights) is necessary, which saves computation time (i.e., cuts processing time by half) and minimizes hardware complexity.

There are three special cases, which must be considered carefully. First, there is a situation where one particle has a weight equal to 1.0 and the rest are all zero. Without any special modification, the scheme will get all the weights to zero, since it only considers the K least significant bits. To avoid this problem, the weight of 1.0 in decimal representation is represented as $1 - 2^{-(K+1)}$. Then the tagging method will guarantee that the total number of replicated particles is M. Second, it is also possible that all the weights are zero. This may happen when the estimate of the state being estimated diverges, and it is not possible to accurately compute the weights with finite precision. This situation can be detected in the weight calculation stage prior to resampling, and thus no resampling is performed. The third special case occurs when the number of particles in the resampling is greater than M due to rounding. The algorithm then produces the correct number of particles, but the number of some particles is not proportional to the respective weights.

III. LOGIC STRUCTURE

A logic structure of the proposed scheme is shown in Fig. 1. The particles and their weights are stored in a memory, and they are provided prior to resampling. The same address is used to

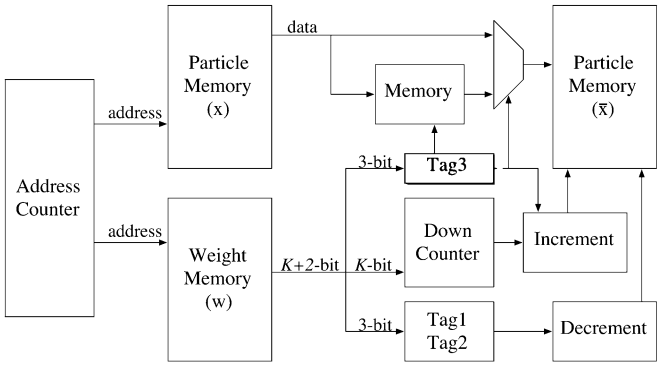

Fig. 1. Logic diagram that illustrates the structure of the proposed resampling scheme. It is assumed that the particles $x^{(m)}$ and their weights $w^{(m)}$ are provided prior to resampling. The resampled particles $\bar{x}^{(m)}$ are stored in a separate memory.

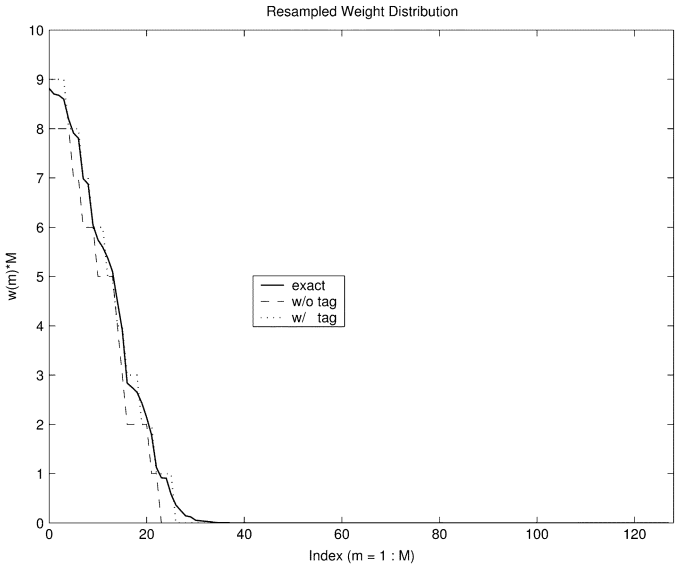

Fig. 2. Comparison of exact resampling and resampling with and without tagging. The particles are ordered according to their weights, where the first particle has the largest weight.

access the particles and weights. Each weight is read and decoded. An integer value of K bits representing r is loaded to the down-counter for particle replication. While a particle is being replicated in the other memory starting from the lowest address, the same particle, if it is tagged (Tag1 or Tag2), is also written to the same memory but starting from the highest address. This is to ensure that when we have enough replicated particles, the tagged particles can be overwritten and discarded. Thus, the total number of replicated particles is always M. However, it is still possible, as we will show in the simulation, that $R+T_1+T_2 < M$. This problem is resolved by having a small memory for storing tagged particles (Tag3) and these particles are copied to the resampled memory. This condition is checked by adding addresses of memory locating the last insertion of replicated particles and tagged particles. If the sum of these two addresses is less than M, the tagged particles (Tag3) are inserted from the starting address of particles right after R. Although we have assumed that the number of particles is a power of two, the scheme can be extended to handle an arbitrary number of particles. Moreover, parallelization is possible for high-throughput applications.

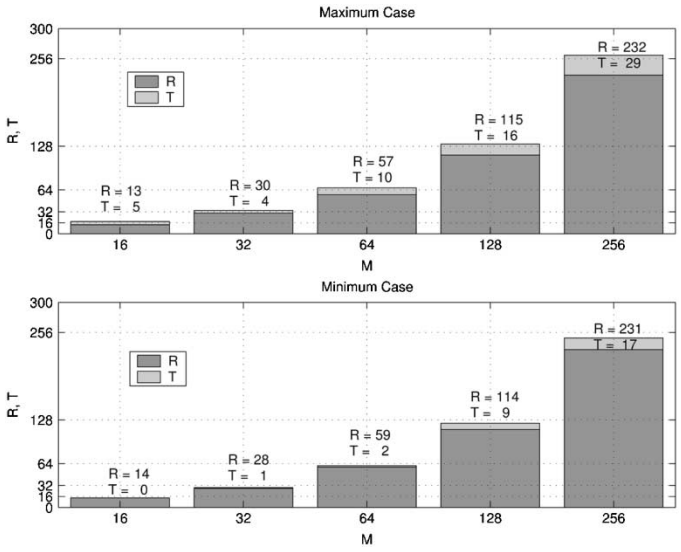

Fig. 3. Illustration of the ranges of total number of replicated and tag particles. Only T_1 and T_2 are included in T. The results are based on 100 independent resamplings.

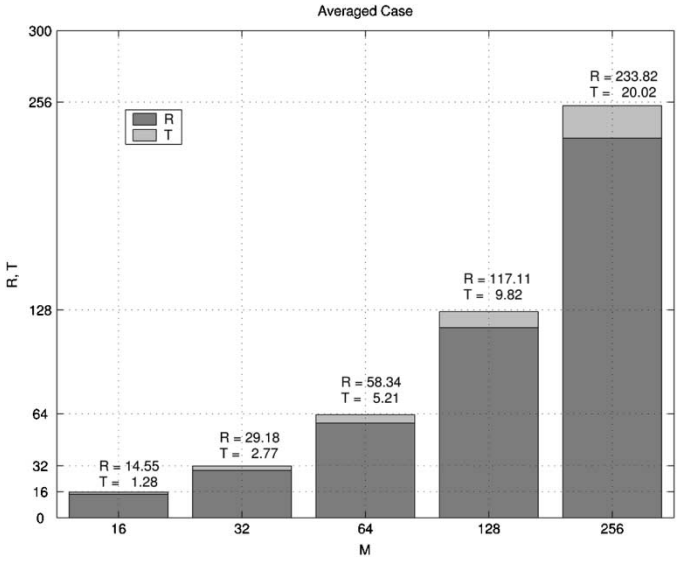

Fig. 4. Average numbers of R and T particles for various M. Only T_1 and T_2 are included in T. The results are based on 100 independent resamplings.

IV. Simulation Results

In order to illustrate the performance of the proposed resampling scheme, M weights are randomly generated, and their weight distributions after resampling are compared with that of full precision resampling. Fig. 2 illustrates the replication factors both with and without tagging for $M = 128$. The results are obtained by simulating the logic structure of the scheme, where a random set of 128 normalized weights is used to implement the resampling. As shown in the figure, the scheme with tagging is very close to that of full precision resampling, which is plotted with a solid line. The sum of the replicated particles illustrated by the resampling without tagging is less than M. Fig. 3 illustrates the range of variation of the number of replicated particles by displaying the ranges of total number of replicated plus tag particles. For example, when the number of particles is 256, the minimum and maximum numbers of $R + T$ particles are 248

and 261 (obtained from 100 independent resamplings). These numbers are important because from them one decides on the size of the Tag3 memory. We note that the range of variation is due to the quantization of the weights. As shown in the figure, it is possible that $R+T_1+T_2 > M$, and this is the reason tagged particles are not included in R. Fig. 4 shows the average number of replicated particles (including the tagged particles) for different M. For example, for 256 particles the average number of R and T particles is 233.82 and 20.02, respectively. When the total number of replicated particles exceeds 100%, the proposed scheme always guarantees that M particles are replicated because of the overwriting mechanism. When the number is below 100%, tagged (with Tag3) particles are used. This has negligible effect on the performance. In both figures, the value of R, which excludes the tagged particles, is always less than M. The total number of replicated particles will be equal to or exceed M only when tagged particles T are added.

V. Concluding Remark

In this letter, we have proposed a novel scheme for fixed-point implementation of residual resampling used in particle filters. The proposed scheme exploits a simple but effective "particle-tagging" method to compensate for possible errors that can be caused by finite-precision quantization in the resampling process. The scheme guarantees that the total number of particles after resampling is always equal to the number of particles before resampling. If the number of particles is a power of two, the resulting scheme is suitable for high-speed physical realizations.

References

[1] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.

[2] C. Berzuini, N. G. Best, W. R. Gilks, and C. Larizza, "Dynamic conditional independence models and Markov chain Monte Carlo methods," *J. Amer. Statist. Assoc.*, vol. 92, pp. 1403–1412, 1997.

[3] A. Kong, J. S. Liu, and W. H. Wong, "Sequential imputations and Bayesian missing data problems," *J. Amer. Statist. Assoc.*, vol. 89, no. 425, pp. 278–288, 1994.

[4] J. S. Liu and R. Chen, "Blind deconvolution via sequential imputations," *J. Amer. Statist. Assoc.*, vol. 90, no. 430, pp. 567–576, 1995.

[5] E. R. Beadle and P. M. Djurić, "A fast weighted Bayesian bootstrap filter for nonlinear model state estimation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 33, pp. 338–343, 1997.

[6] M. Bolić, P. M. Djurić, and S. Hong, "New resampling algorithms for particle filters," in *Proc. ICASSP*, 2003.