Performance Analysis of Resampling Algorithms of Parallel/Distributed Particle Filters

XUDONG ZHANG, LIANG ZHAO, WEI ZHONG, AND FENG GU

ABSTRACT
Particle filters have been widely used in various fields due to their advantages in dealing with non-linear and non-Gaussian systems. A large number of particles are needed to guarantee the convergence of particle filters for the state estimation, especially for large-scale complex systems. Therefore, parallel/distributed particle filters were adopted to improve the performance, in which different paradigms of resampling were proposed, including the centralized resampling, the decentralized resampling, and the hybrid resampling. To ease their adoptions, we analyze time consumptions and speedup factors of parallel/distributed particle filters with various resampling algorithms, state sizes, system complexities, numbers of processing units, and model dimensions in this study. The experimental results indicate that the decentralized resampling achieves the highest speedup factors due to the local transfer of particles, the centralized resampling always has the lowest speedup factors because of the global transfer of particles, and the hybrid resampling attains the speedup factors between. Moreover, we define the complexity-state ratio, as the ratio between the system complexity and the system state size to study how it impacts the speedup factor. The experiments show that the higher complexity-state ratio results in the increase of the speedup factors. This is one of the earliest attempts to analyze and compare the performance of parallel/distributed particle filters with different resampling algorithms. The analysis can provide potential solutions for further performance improvements and guide the appropriate selection of the resampling algorithm for parallel/distributed particle filters.

INDEX TERMS Parallel/distributed particle filters, resampling, performance analysis, state size, system complexity.

I. INTRODUCTION
The filtering problem is a series of state estimation problems from observations with potential noises in signal processing and related fields. An effective approach to solve the filtering problem is to provide the best estimates of the states. Some filtering algorithms have been introduced to handle the filtering problems, including the Kalman filtering algorithm and particle filtering algorithms [1]–[4]. The filtering problems usually exist in non-linear and/or non-Gaussian systems [3], [5], [6]. Kalman filter can only estimate the states for the systems with linear noises [1], [2]. Particle filters use Bayesian inferences and stochastic sampling techniques to estimate the posteriors of the states for all filtering problems [7]–[10]. The particle filtering algorithm uses a set of particles to represent the posteriors of the states. The outputs of particle filters are the estimates of states of interest, which can be iteratively calculated from the expectation of the posteriors. Unlike the Kalman filters, the particle filters are able to handle the system with non-linear and/or non-Gaussian noises [4], [11], [12]. Particle filters have been widely used in a variety of fields, such as positioning, navigating, visual tracking, modeling, and simulation [13]–[18], due to the above-mentioned advantages.

The basic mechanism of particle filters is the sequential importance sampling and resampling (SISR). The particle filtering algorithm uses a set of particles to represent the posteriors of the states. The outputs of particle filters are the estimates of states of interest, which can be iteratively calculated from the expectation of the posteriors. Unlike the Kalman filters, the particle filters are able to handle the system with non-linear and/or non-Gaussian noises [4], [11], [12]. Particle filters have been widely used in a variety of fields, such as positioning, navigating, visual tracking, modeling, and simulation [13]–[18], due to the above-mentioned advantages.

The basic mechanism of particle filters is the sequential importance sampling and resampling (SISR). The particle filtering algorithm executes the sampling step and the resampling step to calculate the expectation of the state at
each time step. In each sampling stage, particle filters calculate the weight of each particle according to one or more observation(s) depending on the number of input observation channels. For each particle, the weight is proportional to the difference between the observation(s) and the candidate represented by that particle. A particle has higher weight if it is more similar to the observation(s). Then, the weights are normalized for particle selection in the following resampling step. Generally, the particles with higher weights will be copied more times to generate more offspring particles. The particles with lower weights will generate fewer offspring particles or be discarded. The total number of particles is constant at the end of the resampling steps. The expectation of posteriors from all particles is the estimated state. The offspring particles will be the inputs of the sampling step in the next time step. The sampling step and the resampling step alternatively execute until the observations are unavailable. Sequential particle filters iteratively calculate the posteriors of states with a single processing unit (PU). The high computation cost is one of the challenges to apply sequential particle filters, especially for the systems with high dimensions and a large number of state variables [8], [19]. An effective approach to address this issue is to use multiple PUs to handle the same number of particles, which refers to the parallel/distributed particle filters [20], [21].

In parallel/distributed particle filters, the sampling procedures are the same as those in sequential particle filters due to their local nature, but the resampling procedures are more complicated due to their global nature. The centralized resampling is one type of resampling algorithms in parallel/distributed particle filters, which needs a central unit (CU) to execute the resampling procedures. In the centralized resampling, the transfers of weights and particles increase the communication cost and lower the speedup factor. Although some efficient particle routing algorithms [20] were introduced to improve the performance of the centralized resampling, it is still suffering from the high communication cost [22].

One strategy to improve the performance of the resampling is the application of a decentralized resampling algorithm [20]. The decentralized resampling lowers the communication cost by handling the particles on each PU independently. However, the decentralized resampling decreases the estimate accuracy due to the lack of global information of particles. In order to improve the performance without lowering the estimate accuracy, some types of hybrid resampling techniques were proposed [8], [19], [23], [24]. The hybrid resampling technique mainly executes the decentralized resampling to guarantee the performance and occasionally invokes the centralized resampling to ensure the estimation accuracy. The convergence of parallel/distributed particle filters with the centralized resampling, the decentralized resampling, and the hybrid resampling has been explored [25], [26].

In addition to different paradigms of the resampling procedures, the number of PUs, the state size, the system complexity, and the model dimension also impact the performance of parallel/distributed particle filters. The state size influences both the communication time and the computation time. The system complexity primarily impacts the computation time. The ratio between the system complexity and the state size is another important factor affecting the performance of parallel/distributed particle filters. The model dimension mainly impacts the communication time. However, the systematic analysis of the impacts of different resampling strategies, state sizes, system complexities, and model dimensions on the performance of parallel/distributed particle filters is lacking. Therefore, in this work, we systematically analyze the performance of parallel/distributed particle filters to provide guidelines for their selections and point out the directions for future possible improvements. The contributions of this work have the following folds. Firstly, we provide a comprehensive analysis for different parallel/distributed particle filters including those with the centralized resampling, the decentralized resampling, and the hybrid resampling. Secondly, we propose a ratio of the system complexity to the state size, namely the complexity-state ratio to measure how different systems including computation-intensive systems and communication-intensive systems impact the performance of parallel/distributed particle filters. Third, we use a concrete example to show how different parameters affect the performance of parallel/distributed particle filters with different resampling algorithms. This work provides important guidelines for the choice of parameters in parallel/distributed particle filters when choosing the resampling algorithms for various applications.

The rest of the paper is organized as follows. Section 2 introduces the related work on sequential particle filters, parallel/distributed particle filters, and their time complexity analysis. Section 3 presents the basics of the centralized resampling, the decentralized resampling, and the hybrid resampling. Section 4 shows the performance analysis of particle filters with the centralized resampling, the decentralized resampling, and the hybrid resampling algorithms. Section 5 presents the experimental results. Section 6 provides discussions on related issues. Section 7 concludes the work and points out future research directions.

II. RELATED WORK
Chopin (2002) used sequential particle filters, which combined the importance sampling and Monte Carlo methods to explore a sequence of multiple distributions of interest [27]. The experimental results showed that such particle filters were able to offer an efficient estimation tool in the static analysis, in which the preliminary exploration of partial posteriors made it possible to save computing time. Salmond et al. (2001) proposed a sequential particle filter based on the Bayesian track-before-detect technique [28]. The proposed particle filter provided sample-based approximations to the distributions of the states directly from pixel array data [28]. The particle filter was also capable to provide a measure of the probability that a target was present.
Khan et al. (2004) presented a sequential particle filter based on a Markov chain and Monte Carlo method, to deal with interacting targets, which were influenced by the proximity and/or behavior of other targets [29]. The experiments indicated that incorporating a Markov random field (MRF) to simulate interactions was equivalent to adding an interaction factor to the weights in particle filters. Nummiaro et al. (2002) developed a color-based sequential particle filter, which integrated the color distributions into a particle filtering algorithm [18]. The color distributions were applied because they are robust to partial occlusion, rotation, and scale-invariant. The experimental results indicated the proposed particle filter had advantages in tracking tasks compared with the mean shift tracking algorithm. Nattapol (2018) proposed to use a multiple model particle filter (MMPF) combining with a seismic wavelet model for the estimation of the seismic events under noisy environments [30]. The experimental results showed that the proposed MMPF can provide excellent seismic event estimates under noisy environments. Han et al. (2015) proposed an adaptive fission particle filter (AFPF) to improve the particle quality in handling the seismic signals [31]. In AFPF, all the particles were processed by a fission procedure to maintain the particle diversity. As a result, the effective seismic information represented by particles reproduces the true signal more reliably. Nattapol and Kosin (2019) introduced an adaptive resampling scheme in a particle filtering algorithm for modal frequency identification and dispersion curve estimation from a time-frequency representation of ocean acoustics signal [32]. The experimental results indicated the proposed adaptive resampling algorithm can improve the accuracy of the modal estimates as well as the dispersion curves of the signal. However, the sequential particle filters running on a single PU usually suffer from heavy computational workloads when handling the problems in large-scale systems and/or with a large number of particles.

In order to improve the performance of sequential particle filters, the parallel computing technique was introduced. As a result, the workload is divided and assigned to multiple PUs to lower the computational time. The parallel/distributed particle filters with the centralized resampling have been widely studied and applied. Medeiros et al. (2008) studied the implementation of parallel computing techniques on a color-based particle filter, which had been successfully applied in the tracking problem of non-rigid objects [33]. The main focus of the work was the parallel computation of the particles’ weights. The experimental results showed the proposed parallel particle filter performed faster in a single instruction multiple data (SIMD) processor than that in a standard desktop computer. Ing et al. (2005) proposed to use parallel particle filters in wireless sensor networks to track the moving objects [16]. The proposed parallel/distributed particle filters running on multiple PUs quantified the vectors of measured data. The experimental results indicated that the proposed method significantly reduced the energy expenditure of computation and data transmission. Sheng et al. (2005) presented parallel/distributed particle filters running on a set of uncorrelated sensor groups in a target localization problem [21]. The experimental results indicated that the proposed distributed particle filters with the Gaussian mixture model (GMM) provided ideal localization and tracking performance. However, the parallel/distributed particle filters with the centralized resampling need transfers of particles, particle weights, and particle routing information between PUs, which produces additional communication costs and lowers the speedup factors in parallel computing.

The decentralized resampling algorithm was proposed to lower the high communication cost in centralized resampling. Bolic et al. (2005) developed a decentralized resampling algorithm for efficiently distribute particle filters, which was able to improve the scalabilities of filtering architectures [20]. The proposed decentralized resampling algorithm was used for estimating the bearings-only tracking applications on a field-programmable gate array (FPGA). The experimental results showed the advantages of the proposed algorithm in performance with different numbers of particles and different levels of parallelism. Huang et al. (2008) proposed to use the parallel/distributed particle filter with a decentralized resampling algorithm to track the moving targets in cluster-based underwater sensor networks (USNs) [34]. Evaluation metrics included tracking performance, communication cost, energy cost, and tracking response time. The experimental results indicated the decentralized resampling allowed the distributed particle filter to achieve a reduction of communication cost, energy cost, and tracking response time. Chen et al. (2010) presented a novel decentralized particle filter by decomposing the state into two parts and handling those two nested subproblems using particle filters [35]. As a result, the proposed decentralized particle filter was more flexible in increasing the level of parallelism and achieved a shorter execution time in particle filters. Undoubtedly, the decentralized resampling improves the performance of parallel/distributed particle filters by decreasing the communication between PUs. However, the estimating accuracy of the particle filter with the decentralized resampling is relatively low due to the lack of particle diversity, compared with the centralized resampling [23].

The hybrid resampling combining the centralized resampling and the decentralized resampling is an effective approach to balance the performance and the estimate accuracy. Bai et al. (2015) proposed a hybrid resampling algorithm with a constant interval between centralized samplings [19]. The hybrid resampling algorithm executed the centralized resampling every a certain number of steps and the decentralized resampling for the rest steps. Compared with the centralized resampling or the decentralized resampling, the hybrid particle resampling is able to achieve better speedups without losing the accuracy in parallel/distributed particle filters. Zhang et al. (2017) further reduced the communication cost by avoiding the unnecessary centralized resampling steps if the system was well converged [23].

Even though various types of particle filters have been introduced, their performance analysis is lacking. In the
limited literature, we are aware of, Zhang et al. (2018) studied the performance of parallel/distributed particle filters with the centralized resampling [22]. In the work, they analyzed the performance of three different routing policies including the random routing policy, the minimal transfer routing policy, and the maximal balance routing policy. The experimental results showed that the minimal transfer routing policy achieved the lowest communication time and the maximal balance routing policy yielded better estimation accuracy. However, the decentralized resampling and the hybrid resampling have not been analyzed. In addition, they didn’t consider the impact of the ratio between the communication and the computation on the performance of parallel/distributed particle filters. Therefore, in this work, we propose to systematically analyze the performance including the communication time, the computation time, and the speedup factor of parallel/distributed particle filters with different resampling algorithms and provide guidelines for their wide adoptions.

III. RESAMPLING ALGORITHMS OF PARALLEL/DISTRIBUTED PARTICLE FILTERS

In parallel/distributed particle filters, the sampling procedure is independently executed on each PU. The main difference between parallel/distributed particle filters is the resampling procedure. The resampling can be classified as the centralized resampling, the decentralized resampling, and the hybrid resampling.

In the centralized resampling, a CU initially collects the weights from each PU and subsequently calculates the particle routing schedules based on the received weights and a specific routing policy, such as the random particle routing policy, the minimal transfer particle routing policy, and the maximal particle routing policy [19]. The routing schedule includes the indices of particles, the corresponding numbers needed to be copied, and the indices of the source PUs and the destination PUs for particle transfers. After that, the schedules are copied and sent to all PUs for the following particle transfers. The flowchart of parallel/distributed particle filters with the centralized resampling is shown in Figure 1. The sampling and the centralized resampling will be iteratively executed unless the observations are unavailable. The centralized resampling has the same procedures as the sequential particle filters. As a result, the centralized resampling guarantees the accuracy, but suffers from the high communication cost.

Unlike the centralized resampling, the decentralized resampling executes the particle resampling independently on each PU and exchanges a small portion of particles between neighboring PUs at each time step to improve the diversity. However, the good particles cannot be fully transferred. Apparently, the high cost communication in the centralized resampling is reduced in decentralized resampling. Therefore the decentralized resampling improves the performance, but decreases the accuracy. The flowchart of parallel/distributed particle filters with the decentralized resampling is shown in Figure 2. In the figure, the decentralized resampling exchanges 10% particles between neighboring PUs for improving the particle diversity and system convergence.

The hybrid resampling combines the centralized resampling and the decentralized resampling to improve the performance without losing the estimate accuracy. The hybrid
resampling mainly executes the decentralized resampling and occasionally invokes the centralized resampling according to specific strategies. The flowchart of parallel/distributed particle filters with the hybrid resampling is shown in Figure 3. In the figure, \( t \) represents the time step and \( k \) represents that the hybrid resampling executes the centralized resampling every \( k \) time steps. All types of hybrid resampling algorithms can be classified as a hybrid resampling algorithm with a parameter \( \alpha \), which is the ratio between the steps of the centralized resampling and the steps of the decentralized resampling.

### IV. PERFORMANCE ANALYSIS OF PARALLEL/DISTRIBUTED PARTICLE FILTERS

#### A. PARALLEL/DISTRIBUTED PARTICLE FILTERS

To analyze the performance of parallel/distributed particle filters, we define the speedup factor in Equation (1).

\[
f = \frac{t_s}{t_p} = \frac{t_s}{t_{comp} + t_{comm}}
\]

In the equation, \( f \) is the speedup factor, \( t_s \) and \( t_p \) are the time consumption for sequential particle filters and parallel/distributed particle filters, respectively; \( t_p \) consists of \( t_{comp} \) and \( t_{comm} \), which are the computation time and the communication time, respectively. For the convenience, we assume that the initialization time \( t_{startup} \) for each communication is a constant and it takes the same amount of time \( t_{data} \) to transfer one data item between PUs. Therefore, it takes \( t_{startup} + n t_{data} \) to transfer \( n \) data items between two PUs. To calculate the communication time for each step, we need to figure out the number of communications and the transferred number of data items. To analyze the computation time, we define the system complexity \( C \) as the number of operations from the current time step to the next time step. The number of operations is used to represent the time consumption for calculating the speedup factors. Although the same operations might consume different time due to the allocation of system resources in reality, we assume they have the same time consumption to qualitatively show the relationship between speedup factors and the complexity-state ratios. We assume that \( N \) particles will be evenly distributed on \( p \) PUs for parallel/distributed particle filters.

#### B. PERFORMANCE ANALYSIS OF PARALLEL/DISTRIBUTED PARTICLE FILTERS WITH THE CENTRALIZED RESAMPLING

In the centralized resampling, the CU coordinates all other PUs for the weight collection, the particle routing calculation, and the particle routing information transfer. The detailed steps and the computation or communication time for each time step are listed in Table 1. In the table, there are 6 steps including 3 computation steps and 3 communication steps. Step 1 is the evolutionary step from the current time step to the next time step and its computation time is reduced to \( 1/p \) of sequential particle filters due to the usage of multiple PUs. Step 2 is the weight calculation and the computation time is also decreased to \( 1/p \) of sequential particle filters. Step 3 is the communication that PUs send the weights to the CU, and therefore, \( p \) communications happen and the total transferred data items are \( N \). Step 4 is the particle routing calculation and its time complexity is \( N \) because the CU needs to handle \( N \) pieces of weights. Step 5 sends the particle routing policy to PUs for the particle exchange in Step 6, in which both are communication time \( t_{comm2} \) and \( t_{comm3} \). They vary by different particle routing policies.

The computational time complexity of parallel/distributed particle filters is \( O(N) \) for non-complex systems since Step 4 dominates the computational time. For a large-scale complex system, the computational time complexity is \( O(CN/p) \).

### TABLE 1. Performance analysis of parallel/distributed particle filters with the centralized resampling.

| Step | Time Complexity |
|------|----------------|
| 1. Calculate the states of \( N/p \) particles on each PU. | \( t_{comp1} = CN/p \) |
| 2. Calculate the weights of \( N/p \) particles on each PU. | \( t_{comp2} = N/p \) |
| 3. Send the weights from each PU to CU. | \( t_{comm1} = p t_{startup} + N t_{data} \) |
| 4. Compute the particle routing information in CU. | \( t_{comp3} = N \) |
| 5. Send the routing information to PU. | \( t_{comm2} \) |
| 6. Particle exchange. | \( t_{comm3} \) |
due to the significance of the system speedup complexity. Therefore, the corresponding computational speedup factor is $p$. The communication time consumptions in Step 5 and Step 6 are different according to various particle routing policies. In this study, we consider the minimal transfer particle routing policy for the centralized resampling in this work, in which PUs with the most extra particles send particles to those need the most particles, to reduce the number of transfers. The corresponding time consumptions $t_{\text{comm}}^2$ and $t_{\text{comm}}^3$ are shown as Equation (2) and Equation (3), respectively. In those equations, $\lceil N/p^2 \rceil (p - 1)$ is the largest number of transfer times between PUs.

\begin{align}
    t_{\text{comm}}^2 & \leq \left\lceil N/p^2 \right\rceil (p - 1)(t_{\text{startup}} + 4t_{\text{data}}) \quad (2) \\
    t_{\text{comm}}^3 & \leq \left\lceil N/p^2 \right\rceil (p - 1)(t_{\text{startup}} + St_{\text{data}}) \quad (3)
\end{align}

To consider the impact of ratio of the communication and the computation, we define a complexity-state ratio $r$ as $C/S$, in which $C$ is the system complexity influencing the computation time and $S$ is the state size affecting the communication time. We aim to use this ratio to provide the feature of the system, for example, computation-intensive or communication-intensive. $N$ is usually larger than $p^2$ and then the speedup factor can be simplified as shown in Equation (4). Equation (4) is derived by substituting the $t_{\text{comm}}^2$ and $t_{\text{comm}}^3$ in Equation (1) with Equation (2) and Equation (3). We omit the constant term $t_{\text{startup}}$ and assume $\lceil N/p^2 \rceil (p - 1)S \geq NS/p$ to reduce the equation, since $N$ is the number of particles, which is much larger than $p$. The equation indicates that if $r$ is a small value, which means the state size $S$ is much larger than $C$, $f \simeq rp/t_{\text{data}}$. If the state size $S$ is much smaller than $C$, the equation will be $f \equiv p$. Hence, the model can achieve the speedup up to $p$ for a computation-intensive system.

\begin{equation}
    f = \frac{CN}{CN/p + [N + \lceil N/p^2 \rceil (p - 1)S]t_{\text{data}}} \simeq \frac{CN}{C/p + (S/p)t_{\text{data}}} = \frac{C/S}{C/S}\frac{t_{\text{data}}}{Sp + t_{\text{data}}} = \frac{rp}{r + t_{\text{data}}} \quad (4)
\end{equation}

**C. PERFORMANCE ANALYSIS OF PARALLEL/DISTRIBUTED PARTICLE FILTERS WITH THE DECENTRALIZED RESAMPLING**

The decentralized resampling executes the sampling and resampling independently on each PU. Hence, the computational time complexity of parallel/distributed particle filters with the decentralized resampling is $t_{\text{comp}} = O(CN/p + N/p) = O(CN/p)$. The decentralized resampling only exchanges a portion of particles between neighboring PUs in each time step. For example, any PU with an even index $i$ sends a portion of particles to a PU with an index $(i + 1)\% M$ in the first round, where $M$ represents the number of PUs and it is an even number. In the second round, any PU with an odd index $i$ sends the same portion of particles to a PU with an index $(i + 1)\% M$. Based on that scenario, the particle transfer can be efficiently completed in two rounds. Therefore, the computational speedup factor is $p$. The communication time consumption $t_{\text{comm}} = 2(t_{\text{startup}} + N\beta St_{\text{data}}/p)$, where $\beta$ is the percentage of particles need to be sent to the neighboring PUs. After the simplification, the speedup factor is shown in Equation (5).

\begin{equation}
    f = \frac{CN}{CN/p + 2N\beta St_{\text{data}}/p} \simeq \frac{Cp/S}{C/S + 2\beta t_{\text{data}}} = \frac{rp}{r + 2\beta t_{\text{data}}} \quad (5)
\end{equation}

**D. PERFORMANCE ANALYSIS OF PARALLEL/DISTRIBUTED PARTICLE FILTERS WITH THE HYBRID RESAMPLING**

The hybrid resampling combines both the centralized resampling and the decentralized resampling. We assume the ratio between the time steps of the centralized resampling and the total number of time steps is $\alpha$, where $0 \leq \alpha \leq 1$, and we use the minimal transfer particle routing policy in the centralized resampling, the speedup factor is shown in Equation (6). In the equation, if $\alpha = 0$, the speedup factor will be the same as that in Equation (5), and if $\alpha = 1$, the speedup factor will be the same as that in Equation (4).

\begin{equation}
    f = \frac{CN}{CN/p + A + B/rp} \simeq \frac{Cp/S}{C/S + [\alpha + 2(1 - \alpha)\beta]t_{\text{data}}} = \frac{r + [\alpha + 2(1 - \alpha)\beta]t_{\text{data}}}{r + \alpha t_{\text{data}}} = A = \alpha [N + \lceil N/p^2 \rceil (p - 1)S]t_{\text{data}}, \quad B = 2(1 - \alpha)N\beta St_{\text{data}}/p \quad (6)
\end{equation}

In the analysis, the complexity-state ratio impacts the speedup factors, which is also shown in Figure 4. In the figure, the horizontal axis is the complexity-state ratio and the vertical axis is the speedup factors. The curves show how the speedup factors change with the increase of the complexity-state ratio. From the figure, we see that the increase of the complexity-state ratio indicating more computationally intensive systems achieves higher speedups, no matter what resampling algorithms are used. Therefore, parallel/distributed particle filters have better performances in computationally intensive systems.

**V. EXPERIMENTS AND RESULTS**

**A. EXPERIMENTAL DESIGN**

To evaluate how such parameters as the state size, the system complexity, the complexity-state ratio, the number of PUs, the model dimension, and the resampling algorithm impact on the performance of parallel/distributed particle filters, we run parallel/distributed particle filters with various resampling algorithms on an object tracking task as an example to compare how they perform and provide the guidelines for the choices of parameters and algorithms in different scenarios. In this example, a flying bird is tracked from a piece of video with 414 frames as shown in Figure 5. The bird is cropped with a rectangle $(300 \times 150$ pixels) in the first frame as the observation. The coordinates of the bird in each frame are estimated by using parallel/distributed particle
filters with 320 particles. Each particle contains a vector with the dimension equal to the state size. In the vector, each entry stores estimated coordinates corresponding to a sub-posterior. The sub-posterior is proportional to the distance between the observation (cropped in the first frame) and the candidate represented by the coordinates in the entry. The distance is calculated by using the Gaussian Kernel on pixel-wise. For a particle, the posterior is the expectation of all sub-posteriors of coordinates stored in the vector. A particle with a large state size contains more entries and thus has more communication costs when it is transferred between PUs. The number of operations related to computational costs in particle filters is represented by the system complexity. In order to study the influence of the system complexity on the performance, particle filters execute the calculation steps (as shown in Table 1) multiple times to increase the system complexity. For example, if the system complexity is two, particle filters execute the calculation steps twice. In the decentralized resampling, a certain percentage of particles on each PU are randomly selected and transferred to their neighboring PUs for improving the particle diversity and the system convergence. Too few or too many transferred particles between neighboring PUs will impact the system convergence. However, a large number of the transferred particles between PUs will significantly increase the communication cost, and thus decrease the performance of parallel/distributed particle filters. In this work, we transfer 10% of the particles on each PU to their neighboring PUs for the performance comparisons of different resampling algorithms. However, it can be optimized for the enhanced accuracy and performance of parallel/distributed particle filters. We will investigate this issue in our later work. In the hybrid resampling, the ratio between the centralized resampling steps and the decentralized resampling steps is 0.2. All of the particle filtering algorithms are written with C++ and Message Passing Interface (MPI) library. The computational platform is a cluster with 32 PUs (Intel Xeon(R) CPU E5-2643 3.30GHz processor, 4GB memory per PU) and a distributed memory architecture.

B. EXPERIMENTAL RESULTS

We measure and compare the communication time, the computation time, the total time consumption, and the speedup factors of parallel/distributed particle filters with the centralized resampling (the minimal transfer routing policy), the decentralized resampling, and the hybrid resampling. The speedup factors are obtained by calculating the ratios between the time consumption of sequential particle filters and that of parallel/distributed particle filters on the same tracking task, as shown in Equation (1). In Equation (1), the increase of the communication time, $t_{\text{comm}}$, can lower the speedup factor by increasing the total time consumption, $t_p$, which is equal to the sum of the communication time, $t_{\text{comm}}$, and the computation time, $t_{\text{comp}}$. The communication cost is decided by the resampling algorithm because the communication cost is only produced in the resampling procedures. However, the sampling procedures independently execute on individual PUs, and no communications exist between PUs.

1) SYSTEM COMPLEXITIES ON THE PERFORMANCE OF PARALLEL/DISTRIBUTED PARTICLE FILTERS

The system complexity influences the computation time and the speedup factor. A complicated system leads to more computation costs during the evolution of the system compared with a simple system. Figure 6 shows the relationship between the system complexity and the communication time, the computation time, the total time, and the speedup factor. In the figure, the horizontal axis is the system complexity and the vertical axis represents the computation time, the communication time, the total time, and the speedup factor. In the figure, the horizontal axis is the system complexity and the vertical axis represents the computation time, the communication time, the total time, and the speedup factor, respectively. Centralized, Decentralized, and Hybrid indicate parallel/distributed particle filters with the centralized resampling, the decentralized resampling, and the hybrid resampling, respectively. The solid line and the dashed line specify the results from 4 PUs and 16 PUs, respectively. From Figure 6(a), (b), and (c) we see that the communication time with 4 PUs is larger than that with 16 PUs due to more
transfers between PUs. Also, parallel/distributed particle filters with the hybrid resampling consume more communication time, computation time, and total time than those with the decentralized resampling and less communication time, computation time, and total time than those with the centralized resampling. This is because the centralized resampling enhances both the computation time and the communication time due to the calculation of the transfer schedules and the following transfer of schedules and particles between PUs.

In Figure 6(a), we also notice that the communication time is nearly the same with different system complexities for any resampling algorithms because the communication time is decided by the number of data transferred between PUs, including the weights, the particle transfer schedules, and the particles. Increasing the system complexity does not influence the communication, but increases the computation time and total time consumption, as shown in Figure 6(b) and (c). Figure 6(d) shows the relationship between the system complexity and the speedup factor. From the figure, we know that more PUs lead to a higher speedup factor, and with the increase of the system complexity the speedup factors slightly rise. This is because a high system complexity allows the particle filters to focus more on the computation than the communication. Parallel/Distributed particle filters with the decentralized resampling achieve the highest speedup factor and those with the centralized resampling have the lowest speedup factors under the same settings. The differences of speedup factors become larger when the number of PUs increases. When the system complexity increases, the computation time increases. Although the total time consumption increases, the ratio of computation time to the communication time also increases, and thus the speedup factor increases, as indicated in Equation (4), (5), and (6).

2) STATE SIZES ON THE PERFORMANCE OF PARALLEL/DISTRIBUTED PARTICLE FILTERS

The state size determines the communication time in parallel/distributed particle filters. A large state size allows particle filters to consume more time on the data transmission, thus increasing the communication time and decreasing the performance of parallel/distributed particle filters. Figure 7 shows the relationship between the state size and the computation time, the communication time, the total time, and the speedup factor. We also run the applications with 4 PUs (solid lines) and 16 PUs (dashed lines), respectively. In the figures, the horizontal axis is the state size and the vertical axis...
represents the communication time, the computation time, the total time, and the speedup factor, respectively. Centralized, Decentralized, and Hybrid indicate parallel/distributed particle filters with the centralized resampling, the decentralized resampling, and the hybrid resampling, respectively. Figure 7(a), (b), and (c) indicate that a larger number of PUs lead to less communication time, computation time, and total time with the same parameter settings. With the increase of the stat size, the communication time, the computation time, and the total time grow because the state size impacts both the computation and the communication. However, when the state size reaches 25, the communication time increases more significantly, leading to the same observation in the total time. Figure 7(d) displays how the speedup factor is influenced by the state size. From the figure, we notice a similar trend that a larger number of PUs achieve a higher speedup factor and parallel/distributed particle filters with the hybrid resampling perform between those with the centralized resampling and those with the decentralized resampling.

### 3) COMPLEXITY-STATE RATIOS ON THE PERFORMANCE OF PARALLEL/DISTRIBUTED PARTICLE FILTERS

The state size influences both the communication time and the computation time, and the system complexity impacts the computation time. Both the state size and the system complexity greatly impact the performance of parallel/distributed particle filters. Therefore, their combined effect needs to be considered. Based on this idea, we propose to measure the complexity-state ratio and check how it influences the performance of parallel/distributed particle filters. The values of the complexity-state ratio and the corresponding combinations of the system complexities and the state sizes are shown in Table 2. For the same complexity-state ratio, we check 3 different combinations of the system complexities and the state sizes, which are randomly generated, as shown in each column in Table 2, and use the average measurements in Figure 8. Figure 8 shows the relationship between the complexity-state ratio and the communication time.
FIGURE 8. The relationship between the complexity-state ratio and the communication time, the computation time, the total time, and the speedup factor.

The computation time, the total time, and the speedup factor. In the figure, we use the same notations as before. The horizontal axis is the complexity-state ratio and the vertical axis represents the communication time, the computation time, the total time, and the speedup factor, respectively. Likewise, the solid lines and the dashed lines denote the data from the parallel/distributed particle filters with 4 PUs and 16 PUs, respectively.

The figure indicates that all the communication time, the computation time, and the total time decrease when the number of PUs increases. The parallel/distributed particle filters with the decentralized resampling consume the least communication, computation, and total time, and the parallel/distributed particle filters with the centralized resampling consume the most communication, computation, and total time. In Figure 8(a), when the complexity state ratio is 1, we get the minimal average state size, 5, and the communication time is minimal. When the complexity state ratio is 4, we get the maximal average state size, $20/3$, and the communication time is maximal. The phenomenon indicates the trend of communication time follows the change of the average state sizes.

Besides the algorithms used in parallel/distributed filtering, the computation cost is also decided by the state size and the system complexity. A particle with a larger state size needs more time to transfer and compute because that particle contains more data items. In Figure 8(b), the change of the computation time is consistent with that of the product of the state size and the system complexity. The maximal computation time and minimal computation time are obtained when the average products of the state size and the system complexity are $58/3$ and $712/3$, corresponding to the complexity-state ratios 0.5 and 8, respectively.

Increasing the state size results in more computation and communication time. But increasing the system complexity only leads to more computation time. Figure 8(a) and Figure 8(b) indicate the computation time is much larger than the communication time for any complexity-state ratio. As the sum of communication time and computation time, the total time consumption has similar results as the computation time, as shown in Figure 8(c).

Figure 8(d) shows the relationship between the complexity-state ratios and the speedup factors. We observe that the
speedup factor slightly increases with the increase of the complexity-state ratios, which indicates more computations lead to a higher speedup factor, thus computationally intensive systems can achieve better speedups.

4) NUMBERS OF PUs ON THE PERFORMANCE OF PARALLEL/DISTRIBUTED PARTICLE FILTERS

It is obvious that the number of PUs impacts the performance of parallel/distributed particle filters. The previous experiments only use 4 PUs or 16 PUs for all the algorithms and applications. Therefore, it is necessary that we compare the time consumptions and the speedup factors with more numbers of PUs to check how PUs impact the communication time, the computation time, the total time, and the speedup factor for parallel/distributed particle filters with different resampling algorithms. Figure 9 shows the relationship between the number of PUs and the communication time, the computation time, the total time, and the speedup factor when the state size and system complexity are fixed. We use the same notation that Centralized, Decentralized, and Hybrid represent parallel/distributed particle filters with the centralized resampling, the decentralized resampling, and the hybrid resampling, respectively. The horizontal axis is the number of PUs and the vertical axis represents the communication time, the computation time, the total time, and the speedup factor, respectively.

From the figure, we know that all the communication time, the computation time, and the total time decrease with the increase in the number of PUs. Especially in the beginning, all the time significantly decrease. When the number of PUs reaches 16, all the time slightly decreases. This is because there exists a tradeoff between the computation and the communication. Also, the increasing ratio of the number of PUs gradually decreases, when the number of PUs increases.

In Figure 9(d), we have similar observations that the speedup factor significantly increases in the beginning and the trend slows down later. This reflects the same tradeoff between the communication and the computation in parallel/distributed particle filters. Also, the same trend occurs that parallel/distributed particle filters with the decentralized resampling have the best performance.

5) DIMENSIONS OF THE MODEL SPACE ON THE PERFORMANCE OF PARALLEL/DISTRIBUTED PARTICLE FILTERS

The dimension of the model state is another important factor to impact the performance of parallel/distributed particle filters.
filters. Thus, we evaluate the influence of the dimension of the model space on the performance of parallel/distributed particle filters, including one dimension (1D), two dimensions (2D), and three dimensions (3D). Figure 10 shows the relationship between the dimension of the model space and the communication time, the computation time, the total time, and the speedup factor with 4 PUs and 16 PUs, when the state size and the system complexity are fixed. The state size and the system complexity are set to be 1. The blue, red, and green colors represent the parallel/distributed particle filters with the decentralized resampling, the centralized resampling, and the hybrid resampling, respectively. The horizontal axis is the dimension of the model space and the vertical axis represents the communication time, the computation time, the total time, and the speedup factor, respectively.

The communication time reflects the communication cost in parallel/distributed particle filters, which includes the cost of transferring the particle weights, the particle routing schedules, and the particles. The change of the model dimension only influences the dimension of particles. For example, a 2D particle contains the data corresponding to the horizontal coordinate and the vertical coordinate in a 2D model. It is obvious that transferring more data consumes more time during particle transfers. Thus, the communication time increases if the dimension increases for all resampling algorithms, as shown in Figure 10(a).

The computation cost is mainly generated during the weight calculation for each pixel in sampling processes. According to the settings, the number of pixels is the same (3,000 pixels) for the observations in 1D, 2D, and 3D models. Thus, the computation time is similar for the parallel/distributed particle filters with the same resampling algorithm and the number of PUs, as shown in Fig 10(b). Fig 10(c) shows the total time consumption. As the sum of the communication time and the computation time, the total time consumption follows the same trend as shown in Fig 10(a). Fig 10(d) shows the relationship between the speedup factor and the model dimension. For parallel/distributed particle filters using the same resampling algorithm and the same number of particles, the speedup factor decreases if the dimension of the model increases because a higher dimension leads to the increase of the total time. As a result, the speedup factor decreases. This phenomenon is also reflected from Equation (1).
VI. DISCUSSIONS
Parallel/Distributed particle filters can be used in any applications of filtering problems and the data assimilation without the assumption of the system properties, such as non-linear and/or non-Gaussian systems. One of the challenges is handling a large number of particles with large state sizes is computationally expensive, especially for large-scale systems. Also, multiple processing units cannot enhance the performance if the system is communication-intensive because the processing units have to focus on the communication in such systems. In this study, we analyze and evaluate the performance of parallel/distributed particle filters with different resampling algorithms including the centralized resampling, the decentralized resampling, and the hybrid resampling algorithm. The defined measure of the complexity-state ratio and other parameters are evaluated for their impacts on parallel/distributed particle filters to facilitate their choices.

The work provides the guidelines on how to choose different resampling paradigms when applying parallel/distributed particle filters for various applications and systems. Besides the observations from the experimental results, we provide the discussions on the related issues below.

In this work, we primarily focus on the performance analysis of parallel/distributed particle filters with different resampling algorithms, which include different strategies to parallelize the global resampling procedure. All resampling algorithms generate communication costs to prevent better performance. The centralized resampling in the parallel/distributed particle filters performs the same procedure as the resampling in the sequential particle filters, thus to achieve better prediction accuracy but reduced performance. The decentralized resampling improves the performance by executing the local resampling on each PU but suffers from low prediction accuracy. The hybrid resampling aims to improve performance without losing the prediction accuracy. In summary, the system convergence of parallel/distributed particle filters with the centralized resampling is the highest because particles with high weights are distributed to other PUs in the resampling process. The system convergence of parallel/distributed particle filters with the decentralized resampling is the lowest because the particles only transfer locally between neighboring PUs. As a result, the particle lacks diversity and the corresponding system convergence is low. By using a hybrid resampling scheme with the same settings, parallel/distributed particle filters have the system convergence between. The number of particles and noises also impact the prediction accuracy of particle filters. However, in this study, we don’t convey the prediction accuracy for different resampling algorithms in parallel/distributed particle filters although it is another important factor in algorithm choices.

In the decentralized resampling, the neighboring PUs exchange a portion of particles after the local resampling in each time step. The number of particles transferred between PUs greatly impacts the performance of the decentralized resampling. In this work, we transfer 10% of the particles between neighboring PUs in each time step. The selection strategy of the transferred particles is also an important factor influencing the performance. We use random selection in this work and don’t investigate how it impacts the prediction accuracy of parallel/distributed particle filters with the decentralized resampling. This is a topic deserving to be studied and we will explore this issue in our future work. Similarly, in the hybrid resampling, we use $\alpha$ as the ratio between the centralized steps and the decentralized steps. The used percentage of transferred particles and the value of $\alpha$ play an important role in both the prediction accuracy and performance of parallel/distributed particle filters with the hybrid resampling algorithm. In the work, we transfer 10% of the particle between neighboring PUs in the decentralized resampling and invoke the centralized resampling every four steps. We mainly study the performance issues and don’t check how they impact the prediction accuracy in this work.

The related issues and results can be found in [19].

In this work, we theoretically analyze the time complexities for parallel/distributed particle filters with different resampling algorithms. We only analyze them for a one-time step since particle filters execute the sampling and resampling in a stepwise manner. Although the results show that the performance of particle filters with the hybrid resampling is better than those with the centralized resampling and worse than those with the decentralized resampling, different particle routing policies and parameters in the hybrid resampling greatly impact their performance too. The inappropriate choice of parameters may lead to the decreased performance compared with the centralized resampling. Therefore, the work provides the general guidelines for algorithm choices and one can refer to the details in the work of [19].

It is obvious that the state size influences the communication time and the system complexity mainly affects the computation time. There are tradeoffs between the reduced computation time and the increased communication time for parallel/distributed particle filters. In this study, we use the complexity-state ratio to measure the feature of systems and applications. If the complexity-state ratio is high, the system is computation-intensive, and parallel/distributed particle filters may help. Otherwise, the performance may drop due to high communication costs. Therefore, not all parallel/distributed particle filters perform better than sequential particle filters and this is one of the motivations of this work. A large number of PUs may improve the speedup factor, but with the increase in the number of PUs, the improvement of the speedup factor is slight due to the occurred more communication time. This is especially true for parallel/distributed particle filters with the centralized resampling and those with the hybrid resampling due to the fully or partially global nature of the resampling procedure. The dimension of the model influences the dimension of the particles and the communication cost. The high model dimension increases the communication time, and thus decreases the speedup factors.
In this paper, we analyze the speedup factors and time complexities of parallel/distributed particle filters with various resampling algorithms. The work provides a solid foundation and guidelines for choosing different resampling algorithms when performance is the main focus of the systems. The experimental results show that the performance of the parallel/distributed particle filters with the hybrid resampling is between that with the centralized resampling and that with the decentralized resampling. Our future work has the following folds. Firstly, we will apply different algorithms to different types of applications with particle filters to achieve performance improvements. Second, we will investigate more effective selection mechanisms for the transferred particles in order to further improve the prediction accuracy. Third, based on the findings of this work, we will propose other more efficient resampling algorithms for the performance improvement of parallel/distributed particle filters.

ACKNOWLEDGMENT
Support for this project was provided by a PSC-CUNY Award, jointly funded by The Professional Staff Congress and The City University of New York.

REFERENCES
[1] R. E. Kalman, “A new approach to linear filtering and prediction problems,” J. Basic Eng., vol. 82, no. 1, pp. 35–45, Mar. 1960.
[2] I. Arasaratnam and S. Haykin, “Cubature Kalman filters,” IEEE Trans. Autom. Control, vol. 54, no. 6, pp. 1254–1269, May 2009.
[3] R. Olfati-Saber, “Distributed Kalman filter with embedded consensus filters,” in Proc. 44th IEEE Conf. Decis. Control, Dec. 2005, pp. 8179–8184.
[4] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” IEEE Trans. Signal Process., vol. 50, no. 2, pp. 174–188, 2002.
[5] R. Chen and J. S. Liu, “Mixture Kalman filters,” J. R. Stat. Soc., Ser. B, Stat. Methodol., vol. 62, no. 3, pp. 493–508, 2000.
[6] P. Li and V. Kadirkamanathan, “Particle filtering based likelihood ratio approach to fault diagnosis in nonlinear stochastic systems,” IEEE Trans. Syst., Man Cybern., C, Appl. Rev., vol. 31, no. 3, pp. 337–343, Aug. 2001.
[7] A. Doucet, S. Godsill, and C. Andrieu, “On sequential Monte Carlo sampling methods for Bayesian filtering,” Stat. Comput., vol. 10, no. 3, pp. 197–208, 2000.
[8] F. Gustafsson, “Particle filter theory and practice with positioning applications,” IEEE Aerosp. Electron. Syst. Mag., vol. 25, no. 7, pp. 53–82, Jul. 2010.
[9] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, “Particle filters for positioning, navigation, and tracking,” IEEE Trans. Signal Process., vol. 50, no. 2, pp. 425–437, Jul. 2002.
[10] U. Helmke and J. B. Moore, Optimization and dynamical systems, Springer, 2012.
[11] N. J. Gordon, D. J. Salmond, and A. F. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” in IEE Proc. F, Radar Signal Process., vol. 140, no. 2, pp. 107–113, Apr. 1993.
[12] P. Del Moral, “Non-linear filtering: Interacting particle resolution,” Markov Processes Rel. Fields, vol. 2, no. 4, pp. 555–581, 1996.
[13] W. J. Freeman, “Simulation of chaotic EEG patterns with a dynamic model of the olfactory system,” Biol. Cybern., vol. 56, nos. 2–3, pp. 139–150, May 1987.
[14] K. Ikeda and K. Matsumoto, “High-dimensional chaotic behavior in systems with time-delayed feedback,” Phys. D, Nonlinear Phenomena, vol. 29, nos. 1–2, pp. 223–235, Nov. 1987.
[15] L. Kocarev and U. Parlitz, “General approach for chaotic synchronization with applications to communication,” Phys. Rev. Lett., vol. 74, no. 25, p. 5028, 1995.

XUDONG ZHANG received the B.S. degree in material science and engineering from Jilin University, and the Ph.D. degree in mechanical engineering from the City College of New York. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Graduate Center of the City University of New York. His recent research interests include deep learning, computer vision, modeling, and high-performance computing.
LIANG ZHAO (Member, IEEE) received the B.Sc. degree (Hons.) in mathematics from Tsinghua University, in 2008, and the Ph.D. degree in mathematics from the Graduate Center of the City University of New York, in 2017. He is currently an Assistant Professor with the Department of Computer Science, Lehman College, City University of New York. His current research interests include designing efficient deep learning models with fast matrix and tensor computation algorithms for applications in image processing, agent-based simulation, and graph computations.

WEI ZHONG (Senior Member, IEEE) received the Ph.D. degree in computer science from Georgia State University, USA, in 2006. He is currently a Full Professor with the Division of Math and Computer Science, University of South Carolina Upstate. His research interests include deep learning, data mining, bioinformatics, and high-performance computing. He is an Elected Fellow of the International Society of Intelligent Biological Medicine.

FENG GU received the B.S. degree in mechanical engineering from the China University of Mining and Technology, the M.S. degree in information systems from the Beijing Institute of Machinery, and the M.S. and Ph.D. degrees in computer science from Georgia State University. He is currently an Associate Professor of computer science with the College of Staten Island, City University of New York. He is also a Doctoral Faculty Member with the Graduate Center of the City University of New York. His research was supported by the National Science Foundation, the National Institute of Justice, CUNY IRG, and PSC-CUNY. His research interests include modeling and simulation, complex systems, and high-performance computing.

* * *

LIANG ZHAO (Member, IEEE): Performance Analysis of Resampling Algorithms of Parallel/Distributed Particle Filters