Tsinghua Science and Technology

Volume 23 | Issue 4 Article 5

2018

Bidirectional Feedback Dynamic Particle Filter with Big Data for the Particle Degeneracy Problem

Xuefeng Yan
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.

Xiangwen Feng
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.

Chengbo Song
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.

Xiaolin Hu
Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 10/13 pp 463–478
DOI: 10.26599 / TST.2018.9010035
Volume 23, Number 4, August 2018

Bidirectional Feedback Dynamic Particle Filter with Big Data for the Particle Degeneracy Problem

Xuefeng Yan*, Xiangwen Feng, Chengbo Song, and Xiaolin Hu

Abstract: Particle Filter (PF) is a data assimilation method to solve recursive state estimation problem which does not depend on the assumption of Gaussian noise, and is able to be applied for various systems even with non-linear and non-Gaussian noise. However, while applying PF in dynamic systems, PF undergoes particle degeneracy, sample impoverishment, and problems of high computational complexity. Rapidly developing sensing technologies are providing highly convenient availability of real-time big traffic data from the system under study like never before. Moreover, some sensors can even receive control commands to adjust their monitoring parameters. To address these problems, a bidirectional dynamic data-driven improvement framework for PF (B3DPF) is proposed. The B3DPF enhances feedback between the simulation model and the big traffic data collected by the sensors, which means the execution strategies (sensor data management, parameters used in the weight computation, resampling) of B3DPF can be optimized based on the simulation results and the types and dimensions of traffic data injected into B3DPF can be adjusted dynamically. The first experiment indicates that the B3DPF overcomes particle degeneracy and sample impoverishment problems and accurately estimates the state at a faster speed than the normal PF. More importantly, the new method has higher accuracy for multidimensional random systems. In the rest of experiments, the proposed framework is applied to estimate the traffic state on a real road network and obtains satisfactory results. More experiments can be designed to validate the universal properties of B3DPF.

Key words: big traffic data; dynamic particle filter; particle degeneracy; particle impoverishment; Dynamic Data-Driven Application System (DDDAS)

1 Introduction

State estimation aims to estimate the internal state of dynamic systems and heavily depends on historical and real-time big traffic data collected from various sensor sources. With the popularization of the traditional sensor and the emerging traffic sensor technology, sensor data are growing exponentially, thereby ushering in the era of big data. State estimation is now becoming more data-driven[1, 2], and an increasing number of researchers are working to explore the application of big data in state estimation. This situation motivates us to rethink the state estimation problem with such a significant amount of big traffic data[3]. Normally, the input and output of a system, which reflect external characteristics, can be detected, but the dynamic laws of the system mainly depend on its internal state. The Kalman Filter (KF) is a well-known state estimation method that is used to solve linear systems with Gaussian noise, and it has been applied in many fields, such as robot vision[4], navigation systems[5], and human motion tracking[6]. The Extended Kalman Filter (EKF)

- Xuefeng Yan, Xiangwen Feng, and Chengbo Song are with College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China. E-mail: yxf@nuaa.edu.cn; 897649107@qq.com; sami_cheng@outlook.com.
- Xiaolin Hu is with Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA. E-mail: xhu@cs.gsu.edu.
* To whom correspondence should be addressed.
Manuscript received: 2017-10-18; accepted: 2017-11-13

is the nonlinear version of KF, and it is believed to be the most widely used estimation algorithm for nonlinear systems. Years of experience have shown that EKF is reliable only for systems that are almost linear on the time scale of updates[7]. The Unscented Kalman Filter (UKF), which is a combination of KF and unscented transform, is more robust and accurate than EKF in its estimation of error in all directions, but it suffers from high computational complexity and is difficult to apply to non-Gaussian problems.

The Particle Filter (PF) is a data assimilation method that is used to solve recursive state estimation problems. Unlike the KF and its optimized algorithm, the PF does not depend on the assumption of Gaussian noise and can be applied to various systems, even with nonlinear and non-Gaussian noise[8–10]. However, general PF has some shortcomings, such as particle degeneracy and loss of particle diversity. Generally, PF is improved in the aspect of particle distribution, weight calculation algorithm or resampling algorithm, and there are few investigations that have been conducted on multidimensional stochastic complex systems. To overcome particle degeneracy and maintain particle diversity, Li et al.[11] proposed a roughening approach to influence the resampled particles and simulation noise. Zhu et al.[12] chose to avoid the impoverishment problem by optimizing the particle weights. In Ref. [13], KLD sampling and Markov chain Monte Carlo sampling were simultaneously used to improve the performance of PF. Other ways can be used to improve the resampling algorithm, including EKF-based PF, UKF-based PF[14–16], and adaptive resampling PF[7].

However, all the above mentioned improved strategies used unidirectional feedback, that is, the information is only transferred from the real scene to the simulation, and feedback information, which is necessary when dealing with multidimensional random scenes, is lacking.

Dynamic Data-Driven Application System (DDDAS) was proposed by the U.S. National Science Foundation in 2000. After years of development, it has been applied in many fields. DDDAS was intended to constitute a mutual cooperation system between the simulation and the real scene, that is, the system model can adjust its simulation methods and choose to assimilate different data based on simulation results and the expert decision making. In this paper, particle degeneracy and impoverishment problems are solved based on a series of dynamic strategies inspired by the concept of DDDAS. The big traffic data collected by sensors are injected into the proposed framework (sensor data management) as input, and the execution strategies of B3DPF can be optimized based on the simulation result. The dimension and the type of data that should be collected by the sensors can be adjusted dynamically considering the computational complexity and simulation result. The proposed framework is mainly designed for highly random systems and can be used in normal scenes.

This work is inspired by our previous work[17], which was proposed to estimate traffic state by using a Dynamic data-driven Particle Filter (DPF) algorithm, which is a single algorithm that can be applied only to traffic state estimation. In this work, we extract and improve the DPF algorithm's key processes and parameters and propose a more general use algorithm.

The rest of this paper is organized as follows. Section 2 introduces related work about PF and DDDAS. Section 3 presents the general PF and its degeneracy and impoverishment problem. The dynamic improved framework for the PF is discussed in Section 4. Section 5 presents a one-dimensional experiment and a multidimensional experiment. Conclusions are drawn in the final section.

2 Related Work

In the 1950s, Hammersley and Morton[18] proposed a Sequential Important Sampling (SIS) Monte Carlo method to address a statistical problem. They used discrete random sampling to approximate the corresponding posterior probability distribution, which became the basis of the PF. Later, multiple scholars conducted related research. At the end of the 1960s, Handschin and Mayne[19] applied Monte Carlo method to an automatic control field, and in the 1970s, the application domain was expanded further[20, 21]. However, because of the limitations of computing power and the particle degeneracy problem caused by SIS, these works attracted little attention, and the Monte Carlo method developed slowly over a long period of time. In 1993, Gordon et al.[22] introduced the resampling algorithm to SIS and proposed a sampling importance resampling PF that attracted the attention of general researchers. In 1999, Carpenter et al.[23] summarized the previous work and proposed the concept of PF. In the twenty-first century, PFs became a popular research focus. Many improved PF algorithms appeared and received different names in different application fields, such as the bootstrap filter, survival of the fittest, and the sequential Monte Caro method.

The essence of a PF is the sequential importance sampling method. However, the particle degeneracy

phenomenon is difficult to avoid. Three approaches are commonly used to avoid particle degeneracy from different angles.

The first is optimizing the particles Importance Sampling Density (ISD). ISD is used to evaluate the estimated parameters posteriori probability distribution. The optimal ISD function can minimize the variance of the sample weight, thus avoiding particle degeneracy. However, the optimal ISD function is difficult to obtain. Thus the suboptimum ISD is commonly used to imitate the optimal sampling behavior. Julier and Uhlmann[7] proposed the use of EKF to form the proposed distribution, which combines newly arrived observation data and improves the quality of the sample particles. Arulampalam et al.[8] directly selected the observation likelihood function as the proposed distribution function; this study showed that the estimation accuracy of a PF can be further improved in the case of weak observation noise. This alternative approach required less calculation than the Monte Carlo method and met the criterion of Bayesian importance sampling. However, no general suboptimal importance function was available; different optimal importance functions needed to be built for different models.

The second approach is optimizing the resampling method. The core idea is to eliminate the particles with small weight and to retain and copy the particles with a larger weight for the next time step. In the framework of PF, the resampling algorithm is an effective way to solve the particle degeneracy problem. The traditional resampling algorithms are stratified resampling, system resampling, polynomial resampling, and residual resampling. The additional calculation required by the resampling step results in the incerased time complexity of improved PF based on resampling methods. Yu et al.[24] improved the stratified resampling algorithm and proposed a partially stratified resampling algorithm, which can reduce the computational complexity, thus accelerating the resampling step. Wang and Lin[25] divided the particle set into two categories by using UKF and processed the resampling step separately. Wang et al.[26] proposed an adaptive resampling algorithm for target tracking; the algorithm adaptively adjusts the target feature and the number of particles according to different targets and backgrounds. Although the resampling method can solve the particle degeneracy problem, if the measurement data are a random variable, such as the process of iterations, then the weight variance of the particle set can be control adaptive and intelligent algorithms. With the rapid development of intelligent optimization technology and the in-depth study of the PF algorithm, many scholars have considered the integration of intelligent optimization theory and PF algorithm to avoid the particle degeneracy and impoverishment problems. Xu et al.[27] introduced an ant colony optimization algorithm to the PF by transforming the iterations of particles set to the movement of ants without a resampling step. In Ref. [28], the authors introduced the idea of evolution to the PF to improve the diversity of particles by increasing particle variation. Li et al.[29] summarized the intelligent algorithm used to address the PF. These efforts were particularly efficient for presenting or alleviating sample degeneracy and impoverishment, and they concluded that finding more effective solutions in multidimensional problems remains an active and challenging topic, especially in cases where only a small number of particles are allowed.

As mentioned above, the three approaches to address the particle degeneracy problem can solve some of the problems, but each may result in other faults, considers only a limited aspect, or lacks generality. In addition, the improved strategies are unidirectional feedback methods. When faced with multidimensional random problems, the unilateral models and possible noise may result in inaccurate results. Moreover, the existing improved strategies normally assimilate specific dimensions of real data that do not fully represent the real scene, thereby resulting in a lack of accuracy. Normally, increasing the dimensions of assimilated data can increase the conjecture accuracy, but the resulting computational complexity is also increased. Therefore, a bidirectional adaptive method needs to be found to optimize the assimilated data and execution strategies in particle filtering. The bidirectional system should adjust its simulation mode to fit the real scene on the basis of the simulation state; meanwhile, the dimensions of assimilated data should be updated based on computational complexity and simulation accuracy. Thus, the simulation noise and measurement noise can be optimized in real time to accurately fit the real scene, and the strategy and parameters can be adjusted to consider different dimensions with faster speed.

Prudencio et al.[30] proposed a stochastic dynamic data-driven application system for monitoring material damage in aerospace structures in real time. In Ref. [31], a comprehensive DDDAS system architecture for supply chain systems that involve state-of-the-art information technologies was proposed by Celik et al. With regard to volcanic ash transport and dispersal, Patra et al.[32] used data from satellite imagery and observations of vent

parameters and wind fields to drive the simulations, and polynomial chaos quadrature in combination with data integration was employed to complete the DDDAS loop. An adaptive optical sensing DDDAS framework in object tracking was proposed in Ref. [33], and DDDAS was applied to the command, control, and mission planning for UAV swarms in Ref. [34]. The application range of DDDAS extends far beyond these uses.

Four key issues should be addressed before DDDAS (proposed by F. Darema in Ref. [35]) is used. These issues are application simulations, mathematical algorithms, system software, and measurement instrument interfaces. DDDAS can play an important role in many big data fields, such as oil diffusion, manufacturing process control, resource management, weather forecasting, traffic management, systems engineering, and earth exploration, as long as the above four issues are resolved.

In this paper, we focus on the above four key issues in DDDAS and then develop an improved PF framework where the necessary communication bridge between the simulation model, the simulation methods, and the real scene is built, the data management model used for dynamic assimilated data is designed, and the dynamic optimization strategies in the PF and simulation system are considered. The proposed improved PF framework aims to provide a general method for complex multidimensional random systems without suffering from the particle degeneracy and impoverishment problem while maintaining high accuracy and reasonable speed.

3 Particle Degeneracy and Impoverishment Problem in PF

A dynamic system is defined as a discrete dynamic state-space model that contains the system transition model in Eq. (1) and the measurement model in Eq. (2), as shown below. The estimate of the state s_t is based on the set of all measurements $m_{1:t}=\{m_i, i=1,2,\ldots,t\}$, where t is the time step and s_t and m_t are the state variable and the measurement variable, respectively, $f(\cdot)$ is the evolution of the state variable, $g(\cdot)$ is the mapping from the state variable to the measurement variable, and γ_t and ω_t are independent random variables that represent the state noise and the measurement noise, respectively.

$$s_{t+1}=f(s_t,t)+\gamma_t \tag{1}$$

$$m_t=g(s_t,t)+\omega_t \tag{2}$$

Set the initial Probability Density Function (PDF) as $p(s_0|z_0)$=$p(s_0)$. With the use of optimal Bayesian estimation, the state prediction equation can be described as shown in Eq. (3), and the state update is performed using Eq. (4), where $p(m_t|m_{1:t-1})=\int p(m_t|s_t)p(s_t|m_{1:t-1})\mathrm{d}s_t$. If the PDF is linear and Gaussian, then the state estimation can be solved by using the KF method. However, for nonlinear and non-Gaussian distributions, representing the complete analysis formula of the PDF is difficult for the KF and its optimized algorithm. Fortunately, the integral operation can be transformed into the sum of finite samples by using the Monte Carlo method. If a set of independent samples can be computed from $p(s_{0:t}|m_{1:t})$, then the state probability distribution can be approximated as in Eq. (5), where $\sum_{i=1}^{N} w_k^i=1$. As $N\to\infty$, $\hat{p}(s_{0:t}|m_{1:t})$ undergoes absolute convergence to $p(s_{0:t}|m_{1:t})$. Consequently, the SIS method, which is the core algorithm of PF, is obtained.

$$p(s_t|m_{1:t-1})=\int p(s_t|s_{t-1})p(s_{t-1}|m_{1:t-1})\mathrm{d}s_{t-1} \tag{3}$$

$$p(s_t|m_{1:t})=p(m_t|s_t)p(s_t|m_{1:t-1})/p(m_t|m_{1:t-1}) \tag{4}$$

$$\hat{p}(s_{0:t}|m_{1:t})\approx\sum_{i=1}^{N} w_t^i\delta(s_{0:t}-s_{0:t}^i) \tag{5}$$

In SIS, the algorithm goes through multiple iterations. In each iteration, the algorithm receives an observation m_t and produces N particles $\{s_{0:t-1}^i, i=1,\ldots,N\}$ from the importance function $q(s_{0:t}|m_{1:t-1})$, which are used to predict the next state. Then, the state PDF can be expressed by Eq. (6), where w_t^i is the particle importance weight and $w_t^i\propto p(s_{0:t}^i|m_{1:t})/q(s_{0:t}^i|m_{1:t})$. To estimate the next time state, the importance distribution function is set as $q(s_{0:t}|m_{1:t})=q(s_t|s_{0:t-1},m_{1:t})q(s_{0:t-1}|m_{1:t-1})$. Then, the new particle can be written as $\{s_{0:t}^i\}_{i=1}^{N_s}$, and the update formula for the importance weight can be described using Eq. (7). The importance weight can be normalized to obtain the new particle set $\{s_{0:t}^i, w_t^i, i=1,\ldots,N\}$. Then, SIS method moves to the next iteration.

$$\hat{p}(s_{0:t}|m_{1:t})=\sum_{i=1}^{N}\hat{w}_t^i\delta(s_{0:t}-s_{0:t}^t),\quad \hat{w}_t^i=w_t^i/\sum_{i=1}^{N} w_t^i \tag{6}$$

$$w_t^i\propto w_{t-1}^i p(m_t|s_t^i)p(s_t^i|s_{t-1}^i)/q(s_t^i|s_{t-1}^i,m_t) \tag{7}$$

The SIS method uses the importance function to act as the sample function instead of the posterior probability distribution. Ideally, the importance function is close to the posterior probability distribution, but a large deviation may exist because it cannot address the current measured value. After a few iterations, most of the normalized weights may be close to zero, resulting in the degeneracy problem.

To solve the degeneracy problem, the resample algorithm is introduced after weight normalization, followed by the basic PF. Liu[36] introduced an approximate method to measure particle degeneracy using Neff, which is shown in Eq. (8). In practice, Neff is difficult to compute; thus, its approximate value $\hat{\text{Neff}}$, shown in Eq. (9), is used in place of Neff.

$$\text{Neff} = N/(1+\text{Var}(w_t^i)) \tag{8}$$

$$\hat{\text{Neff}} = 1/\sum_{i=1}^{N}(w_t^i)^2 \tag{9}$$

In the resampling step, N offspring samples are drawn with a probability proportional to the normalized sample weights. These samples represent the posterior belief of the system state and are used for the next iteration. The basic PF can be summarized as below.

Step 1. Particle generation: N particles are sampled from the predictive distribution $s_0^i, i = 1,\ldots,N$.

Step 2. Weight computation: The N particle states at time step t are assumed to be $s_t^i, i = 1,\ldots,N$. The importance weights $w_t^i, i = 1,\ldots,N$ are computed, and the weights are normalized according to $\hat{w}_t^i \propto w_i^i[\sum_{i=1}^{N} w_t^i]^{-1}$.

Step 3. Resampling: On the basis of the resampling algorithm, the low-weight particles in the particle set $[\{s_t^i, w_t^i\}_{i=1}^N]$ are discarded. The estimation is continued based on the new particle set $[\{x_t^i, w_t^i\}_{i=1}^N]'$ for a time step.

Step 4. Set $t = t+1$, and Step 1 is repeated.

From the algorithm, we can see that if an observed value is relatively correct or the likelihood function is located at the tail of the posterior probability distribution, the particle weights become small after updating. This condition results in particles with larger weights being selected repeatedly, which reduces particle diversity and results in particle impoverishment.

4 Bidirectional Dynamic PF Improved Framework

4.1 B3DPF improved framework

Normally, the traditional PF in Fig. 1 can be treated as a narrow sense of DDDAS; it can assimilate real-time data and estimate the current real scene. However, unlike DDDAS, the PF is a one-way dynamic system. Customarily, the parameters and algorithms used in each step are settled from the initiation of the simulation and they stay the same during state estimation. In practice, the designed model and parameters may not accurately represent the real scene, especially for multidimensional random problems. Therefore, the key point in DDDAS, namely, the real-time adjustment between the simulation and the real world, motivates PF improvement. Consequently, two ways are used to address the particle degeneracy and impoverishment problems. First, the dynamic relationship is strengthened between the simulation model, the sensor data, and the estimated data in PF execution strategies based on DDDAS. Second, a stochastic model is introduced after the resampling step to optimize the designed model and its parameters.

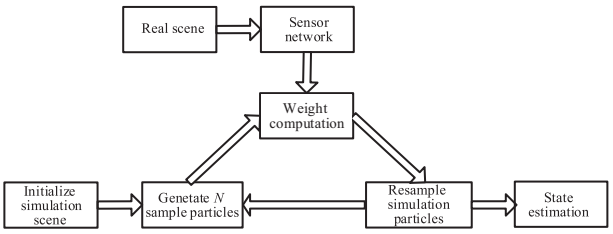

Fig. 1 Structure of the traditional PF.

The structure of the feedback B3DPF designed for multidimensional random systems without degeneracy problem is shown in Fig. 2. B3DPF is a combination of general DDDAS and normal PF that also considers the multidimensional random noise in a real scene, that is to say, B3DPF can transform the PF into a general dynamic data-driven system. Without the real-time feedback control information and the dynamic optimized strategies used in each module, B3DPF is simply the normal PF. In B3DPF, after initializing the simulation scene, each iteration of the improved PF includes four steps: sample particle generation, weight computation, resampling, and stochastic optimization. Sensor data management is used to address the data from the real scene and can supply specific sensor data (one-dimensional or multidimensional) to compute particle weights based on the feedback information from the simulation system. State storage is used to store the current and previous simulation states, which are used to support the dynamic execution strategies. User Interface is a preinstalled interface used to accept expert decisions. The Control System is used to collect the related information that influences the simulation precision and provides certain control information to other modules. State estimation is used to display the estimation result, which is based on the largest-weight particle from the resampling step.

The B3DPF improvement framework consists of weight computation, resampling, and a stochastic model. The three modules work in three different aspects and

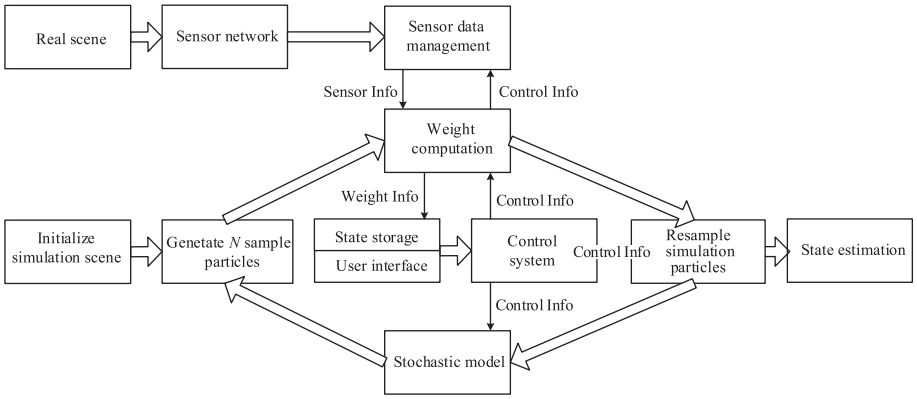

Fig. 2 Structure of the feedback B3DPF.

cooperate to solve the particle degeneracy problem: weight computation is applied when the particle set has too many effective particles; resampling is applied in the opposite situation and stochastic model is applied when too many particles are replaced by a single particle. When the particle set is initialized, the sensor data used in the weight computation can be set as the simple dimensions. After several iterations, most of the particles' weights may be relatively larger and lack diversity. In view of this situation, B3DPF can adjust the input real-time data to complex dimensions; thus, the particles can be valued based on other aspects to rebuild the diversity of the particle set. In addition, the execution speed is faster because the initial assimilated data are simple. Analogously, as the number of low-weight particles increases, the normal resampling method may replace most of the low-weight particles, which is a common cause of the particle degeneracy problem. In B3DPF, before resampling the particle set, the weights of each particle are optimized, which reduces the number of high-weight particles and increases the number of low-weight particles, thus reducing the replacement sequence in resampling and easing the degeneracy problem. In the resampling step, if one particle is used to replace other particles too many times, then the stochastic model is executed to add random factors to parameters in the replaced particles from multiple dimensions. Consequently, the diversity of the particle set can be improved.

The B3DPF improved framework considers not only the particle degeneracy problem but also the execution speed, estimation accuracy, and the adaptation of multidimensional random systems. The sensor data management module is introduced to manage multiple-dimension real-time data and can monitor the real system from multiple perspectives and represent the real scene more accurately. It can also receive real-time control information, thus constituting the basis of the dynamic feedback system. In B3DPF, the weight computation can balance the execution speed and simulation accuracy by controlling the injected data from sensor data management. The weight optimization before resampling and the random factors added to the replaced particles can be implemented selectively. More importantly, the stochastic model can also optimize the system noise and the measurement noise adaptively when addressing multidimensional random systems, which cannot be modeled accurately. The detailed principles of the B3DPF are presented in the following sections.

4.2 Adaptive data assimilation in weight computation

In PF, reasonable particle weight computation is an important aspect to obtain better estimation results. However, the weight variance is not only a manifestation of degeneracy, as in multidimensional random scenes. The real states are arbitrary and unknown. Thus, particle weights can differ greatly, which cannot be attributed to degeneracy. For example, in regard to target tracking, in the region where two or more targets cross (or are close), the particle weight is naturally much higher than that of particles in regions with few or no targets. Therefore, evaluating the particles from another perspective and obtaining the particle weight are necessary, thereby not only influencing resampling and avoiding particle degeneracy but also resulting in a more precise particle set evaluation. The sensor data management model can obtain sensor data from multiple dimensions, and we can focus on reasonable particle weights.

In B3DPF, assume M sensor positions are distributed in the real scene. Each position contains several sensors that can obtain K types of sensor data. On the basis of sensor data management, the data that represent the real scene at time step t can

be described as $\mathrm{RS}_t = \{\mathrm{Sensor}_t[1],\ldots,\mathrm{Sensor}_t[M]\}$, and $\mathrm{Sensor}_t[i]$, $i = 1,\ldots,M$ represents sensor data at the i-th detection positon, which is shown as $\mathrm{Sensor}_t[i] = \{\mathrm{cdata}R(t,1),\ldots,\mathrm{cdata}R(t,K)\}$, where $\mathrm{cdata}R(t,j)$, $j = 1,2,\ldots,K\}$, means the j-th data dimension. In the simulation model, the simulated scene is written as SS_t, and we use the measurement model MM to extract M groups of data from SS_t to RS_t compare with $\mathrm{SS}^{'} = \mathrm{MM}(\mathrm{SS}_t)$. Then, the weight of a particle can be calculated based on $\mathrm{SS}_t^{'}$ and RS_t using weighted fusion method as follows:

$$w_t^i = \sum_{j=1}^{K} \lambda_{t,j} w_{t,j}^i, \quad \lambda_{t,j} \in [0,1],$$

where

$$w_{t,j}^i = (1/(\sqrt{2\pi\sigma_{t,j}}))\exp\{-[h(\mathrm{SS}_{t,j}^{i'}, \mathrm{RS}_{t,j})]/2\sigma_{t,j}^2\}.$$

The contribution of the j-th sensor dimension at time step t to the weight of particle i is defined as $w_{t,j}^i$, $\sigma_{t,j}$ is the measurement noise of the j-th sensor at time step t, and $\lambda_{t,j}$ is the influence factor of the j-th sensor at time step t. The initial sensor data are simple. As the precision reaches 60% of the real scene, the assimilated data are changed by the complex dimensions. Thus, the particle can be valued from multiple aspects to accelerate the assimilation process. If $K = 1$, the weight computation method becomes normal. In this way, during the iteration of B3DPF, the sensor data dimension injected into the weight computation method can be adjusted by updating $\lambda_{t,j}$ to achieve weight optimization from multiple dimensions.

The accuracy of the particle weight is determined not only by the selected distribution but also by the measurement noise. Considering that the multiple sensor fusion structure includes multiple measurement data in a single filter cycle, we can obtain a variety of features of the real scene and improve the measurement noise. The variance of w_t^i can be computed by $w_{t,j}^i = (1/(\sqrt{2\pi\sigma_{t,j}}))\exp\{-[h(x_t^{P,i} - h(x_t^P) - v_{t,j}]/2\sigma_{t,j}^2\}$, and the variance $\sigma_{w_t^i}$ can be computed by

$$\sigma_{w_t^i} = \mathrm{sqrt}\left(\sum_{j=1}^{M} \lambda_{t,j}(\exp(\sigma_{t,j}^2) - 1)\exp(2h(x_t^P + \sigma_{t,j}^2))^2\right).$$

A small $\lambda_{t,j}$ indicates great accuracy of the output result. $\sigma_{w_t^i}$ mainly depends on $\lambda_{t,j}$ if the noise of different sensors at time t is roughly the same. To obtain the most accurate output result, $\lambda_{t,j}$ should be $\lambda_{t,j} = 1/(\xi_{t,j}^2 \sum_{j=1}^{M} 1/\xi_{t,j}^2)$ based on the Lagrange multiplier method. Then, $\sigma_{w_t^i}$ can be calculated by

$$\sigma_{w_t^i} = 1/\mathrm{sqrt}\left(\sum_{j=1}^{M} 1/\xi_{t,j}^2\right).$$

If the noise of different sensors at time t is in $(\sigma_{t,min}^2, \sigma_{t,max}^2)$, we obtain

$$\sigma_{w_t^i} = 1/\left(\xi_{t,\min}^2 + \mathrm{sqrt}\left(\sum_{j=1}^{M-2} 1/\xi_{t,j}^2\right)\right).$$

Therefore, if we use multi-senor data based on the weighted fusion method, then the variance of the particle weights is reduced compared with single-dimension data, even if the sensors have poor accuracy.

4.3 Selective weight optimization in resampling

Particle sampling mainly focuses on introducing the latest measurement information using the proposed distribution optimization method. Generally, it requires the introduction of a specific suboptimal filter, such as EKF or UKF. In this way, each particle is subjected to the suboptimal filter, resulting in greatly increased computation, especially in regard to the multidimensional problem. Strategies have been proposed by the PF community to combat the sample impoverishment caused by the resampling method. One idea is selective resampling, which resamples only when necessary by monitoring the variance of the importance weights instead of always resampling to reduce particle replacement. However, as discussed above, the current weights of the particle set may not accurately represent the real scene, so determining whether resampling is needed is not easy. Therefore, applying the selective resampling strategy based on the variance of the particle weights is not a straightforward method to avoid the particle degeneracy problem. Consequently, the resampling algorithm is included in all iterations of B3DPF. We use a simple, efficient, and stable residual resampling algorithm that has been applied in many real applications. First, a uniform distribution $u_i\left(\frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}, \frac{n}{n}\right)$ is drawn. For each particle's normalized weight, if $\sum_{j=1}^{m=1} \tilde{w}_t^j < u_i \leqslant \sum_{j=1}^{m} \tilde{w}_t^j$, then the m-th simulation particle is covered by the i-th particle.

In B3DPF, selective weight optimization is introduced before resampling to combat sample degeneracy while preventing sample impoverishment. The main cause of particle impoverishment is particles with greater weight being copied multiple times. After a period of time to recurrence, the particle species involved in the calculation becomes impoverished. In B3DPF, when a few particles

have large weights and most of the others have small weights, the weight optimization method is used to narrow the gap between the large-weight particles and small-weight particles, which is inspired by Ref. [12]. The weight optimization method based on the normalized weights is $\tilde{\tilde{w}}_t^i = [\tilde{w}_t^i]^\alpha / \sum_{j=1}^N [\tilde{w}_t^i]$, $\alpha \in (0,1)$. The weight optimization algorithm is executed selectively. If the particle set suffers low particle weight, which can be judged by $\hat{\mathrm{Neff}}$, then resampling is executed based on the recalculated normalized weight. Otherwise, the resampling step is executed directly. The parameters that influence the improvement degree can also be adjusted dynamically.

4.4 State random variable and noise optimization in the stochastic model

The stochastic strategy was first proposed by Gordon et al.[22] as the bootstrap filter. One general idea to rejuvenate the diversity of particles after resampling is to increase the state noise covariance or to introduce additional noise to the samples. Gordon's roughening strategy adds independent Gaussian jitter noise to each resampled particle. The jitter noise is normally distributed with zero mean and constant covariance. The standard deviation is $KEN^{1/d}$, where E is the difference between the maximum and minimum values of the state component, K is a positive tuning constant chosen subjectively by the user, N is the number of particles, and d is the dimension of the state. In Ref. [22], a proper complex random variable is defined as its pseudo covariance vanishes. Let Z be a proper complex n-dimensional Gaussian random vector with mean m and nonsingular covariance matrix,

$$\Lambda = E[(Z-m)(Z-m)^*],$$

where Λ is Hermitian and positive definite. Then, the PDF of Z is given by

$$p(Z) = [1/\pi^n \det(\Lambda)] \exp\{-(Z-m)^* \Lambda^{-1} (Z-m)\},$$

and Z is proper complex and Gaussian with covariance matrix Λ and mean m.

The probability distribution has important applications in stochastic systems. For example, a negative exponential distribution can simulate the random arrival of multi-channel vehicles, the normal distribution can describe the random probability of random events, and the Poisson distribution is suitable for expressing the occurrence of a certain event that occurs per unit time (or space). Therefore, adjusting the simulation result based on the probability distribution in PF is important. In addition, the system noise and the measurement noise in a multidimensional random system should be optimized based on meaningful calculation and a random distribution. To obtain certain state distributions, using a uniform distribution is essential. Here, we use the linear congruence generator, which was proposed in Ref. [37] by Lehmer, to generate an interval uniform distribution U(0,1) as

$$\begin{cases} x_n = (ax_{n-1} + c) \bmod m, \\ u_n = x_n / m, \quad n = 1, 2. \end{cases}$$

Then, we obtain other distributions using transformations.

In B3DPF, two approaches are presented to enhance the diversity and veracity of particles. One approach is using the particle's essential attributes, and the other is based on the system and measurement noise. Random data are added to the basic parameters based on the appropriate probability distribution after the resampling step so that different particles, even particles that overlapped in the resampling step, will have certain differences, thus preventing the particle impoverishment problem caused by the resampling algorithm. For multidimensional stochastic complex systems, the measurement noise and the system noise cannot be evaluated accurately, which may result in large discrepancies with the actual situation. Thus, the system and measurement noise are changeable based on the simulation results. In addition, the relationship between the noise and the simulation results is difficult to determine. Then, in the second part, the stochastic model is applied. Normally, the weight of a particle can be calculated by $w_t^i = f(\mathrm{SS}(t,i), \mathrm{real}(t), \beta_t)$, where $\mathrm{SS}(t,i)$ is the state of the i-th particle at time step t, $\mathrm{real}(t)$ is the real scene at time step t, and β is the measurement noise. The system noise α and measurement noise β at time step $t+1$ can be calculated by

$$\begin{cases} \alpha_{t+1} = \alpha_t + \delta_t, \\ \beta_{t+1} = f'(\overline{w}'(t)) + \gamma_t, \end{cases}$$

where δ_t and γ_t are specific stochastic distributions, and $\overline{w}'(t)$ is the average weight of the particles with high weight. As $\mathrm{SS}(t,i)$, $i = 1, 2, \ldots, N$ and $\mathrm{real}(t)$ are known, the measurement noise can be optimized using $\beta_{t+1} = f'(\overline{w}'(t)) + \gamma_t$.

In the stochastic model, random data always have a positive correlation with the estimate accuracy. That is, when the simulation result has a relatively large error, more random data should be added to increase the accuracy. When the error is smaller, less random data should be

added to maintain the stability of the system. In Ref. [11], the researcher presented four suggestions for roughening methods when addressing sample impoverishment: the roughening method may not be applied in all recursions; the roughening method may not be applied to all particles but only overlapped particles that are resampled from the same particle; the roughening method may not be applied in all dimensionalities; and roughening variance may not be greater than the measurement noise. Considering these four suggestions, the proposed stochastic model implemented all the rules, as discussed in detail in the following sections.

4.5 Sliding window based multidimensional sensor data management

The dynamic PF obtains the next time's simulation data by assimilating sensor data. Normally, the sensor data at one time do not accurately represent the highly random real scene. Many methods are applied to increase the stability of the data while maintaining its real-time characteristics, such as keeping the previous time's sensor data, considering the previous particle weight, and adding a special filter to address the current data. In B3DPF, sensor data from multiple dimensions are required, and considering the flexibility of the acquisition mode, we introduce a sliding window to cache and update the data. The sliding window is formalized as follows: $QueueW =< W_T, Length, Tail, f >$. W_T and $Length$ are the total and current sizes of the sliding window, respectively, $Tail$ is a pointer that points to the location of the newly arrived data, and f is the update rule. For $W_T = 4$ and first in first out processing, the data update process is shown in Fig. 3. On the basis of the sliding window, we proposed the sensor data management model shown in Fig. 4. Data from M collection points are gathered in the real scene, which can monitor K dimensions. Then, the collected data are organized into $M \times K$ sliding windows using sliding windows of sensor data. Assume that at time t in sliding windows of sensor data, the type of data in collection point i is cdata$_{[1-W_T]}(t,i)$, and cdata $S_{[1-W_T]}$

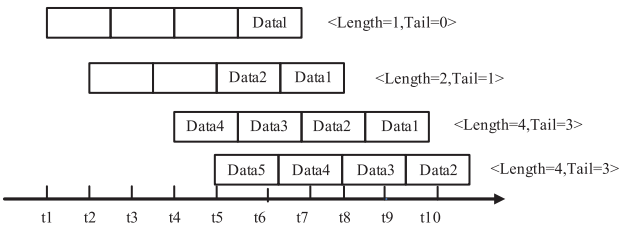

Fig. 3 Data update process using the sliding window.

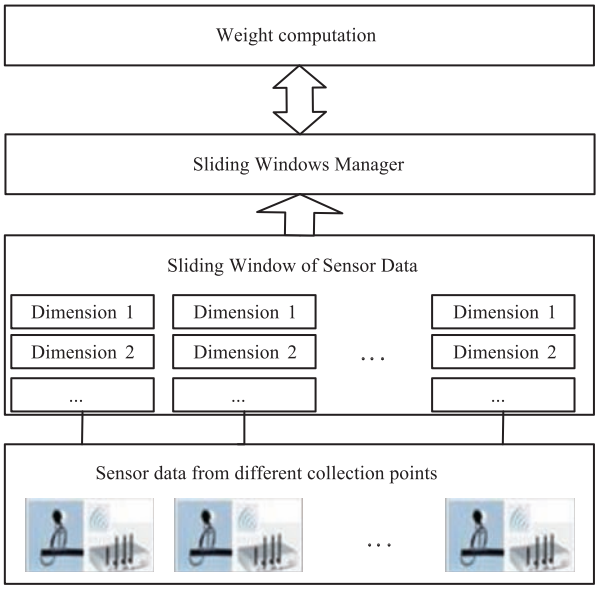

Fig. 4 Sensor data management model based on a sliding window.

(t,i) is the data sorted by time. The first data in the sorted array is the first data that came into the sliding widow. In this paper, we use cdata$R(t,i)$ to represent the sliding window, which can be computed as follows:

$$\text{cdata}R(t,i) = \sum_{j=1}^{W_T} \frac{x_j}{W_T} \text{cdata}S_{[j]}(t,i),$$

where

$$\sum_{j=1}^{W_T} x_j = W_T, \quad x_1 < x_2 < \cdots < x_{W_T}.$$

With the use of the cdata$R(t,i)$ computational formula, the type of data in collection point i at time t can be a weighted sum, thereby ensuring stability. Sliding windows manager provides sensor data to weight computation by handling the sensor data stored in the sliding windows and receives control information from weight computation to provide the simulation model with the assimilation data of specific dimensions and precision.

4.6 B3DPF operation mechanism

In according with the simulation model and methods mentioned above, the improved PF can be summarized by the following steps. We use Step 1 to initialize N simulation particles, and the loop of Steps 2 to 7 achieves continuous state-space conversion.

Step 1. N simulation particles are initialized based on the given scene. Each particle should be independent and can be synchronized to the next time step based on the

specified system model. The system noise is set as α_t, and the measurement noise is set as β_t at time step t.

Step 2. N particles are generated from the predictive distribution.

Step 3. The N particles' weights are computed. The weight of particle i at time t can be calculated by

$$w_t^i = f(t,i) = \Pi_{j=1}^{K} a_t^j v_j^i(j),$$

where K is the dimension of the given scene, and a_t^j is a penalty factor that can be set as 0 or 1.

$$a_0^0 = 1, \quad a_0^j = 0, \quad j > 0,$$

$$a_t^j = \begin{cases} 1, & a_{t-1}^j = 1, \\ a_{t-1}^{j-1} \times T, & a_{t-1}^j = 0, \end{cases}$$

$$\begin{cases} T = 1, \ \text{Vpt}, \\ T = 0, \ \text{else}, \end{cases}$$

where Vpt means $\hat{\text{Neff}}(t-1) > 0.8$ and $\overline{w}'(t-1) > 0.8$, which is calculated in Step 4. $v_t^i(j)$ can be calculated as

$$v_t^i(j) = [1/\text{sqrt}(2\pi\beta_t)] \exp(-\text{Dis}_t(j)^2/(2\beta_t)),$$

where $\text{Dis}_t(j)$ is

$$\text{Dis}_t(j) = \sum_{t=0}^{M} |\text{sim}_t^j(i) - \text{real}_t^j(i)|/M,$$

where $\text{sim}_t^j(i)$ and $\text{real}_t^j(i)$ are the estimated data and sensor data of the j-th dimension. The weights are normalized as

$$\tilde{w}_t^i = w_t^i / \sum_{j=1}^{N} w_t^j.$$

Step 4. The effective number of particles is calculated using

$$\hat{\text{Neff}} = \left[\sum_{i=1}^{N} (\tilde{w}_t^i)^2 \right]^{-1}$$

and the average weight is calculated using

$$\overline{w}(t) = \sum_{i=1}^{N} w_t^i / N.$$

Assume $w_t^1 < w_t^2 < \cdots < w_t^n$ and set $\overline{w}'(t) = \sum_{i=N/2}^{N} 2w_t^i/N$, which represents the average weight of the top $N/2$ particles. If $\hat{\text{Neff}} > 0.3N$, Step 6 is performed; otherwise, Step 5 is performed.

Step 5. To reduce particle replacement, the normalized weight is recalculate as

$$\tilde{\tilde{w}}_t^i = [\tilde{w}_t^i]^{\hat{\text{Neff}}/N} / \sum_{j=1}^{N} [\tilde{w}_t^i]^{\hat{\text{Neff}}/N}.$$

Step 6. The particle set is resampled. The replication sequence, which saves the particle number to be copied, is calculated by

$$\sum_{j=1}^{m-1} \tilde{\tilde{w}}_t^j < u_i \leqslant \sum_{j=1}^{m} \tilde{\tilde{w}}_t^j, u_i \sim \left(\frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}, \frac{n}{n} \right).$$

Then, the state of each particle is updated based on the replication sequence.

Step 7. On the basis of the stochastic model, the parameters of the resampled particles and the system and measurement noise are updated using the equations

$$P'(t,i) = P(t,i) + g(\overline{w}(t))\text{rand}(1,-1)N(\mu,\sigma^2),$$

$$\alpha_{t+1} = \alpha_t + (\hat{\text{Neff}}/N)\text{rand}(1,-1)N(\mu,\sigma^2),$$

and

$$\beta_{t+1} = f'(\overline{w}'(t)) + (\hat{\text{Neff}}(t)/N)\text{rand}(1,-1)N(\mu,\sigma^2),$$

where $g(\cdot)$ is an anti-function with $\overline{w}(t)$, $\text{rand}(1,-1)$ means randomly selected -1 or 1, $N(\mu,\sigma^2)$ is a normal distribution with mean μ and variance σ, and $f'(\cdot)$ is the inverse function of the calculation equation of w_t^i in Step 3.

Step 8. $t = t+1$ is set, and Step 2 is repeated or the algorithm is terminated.

5 Experiments and Discussion

B3DPF is demonstrated using two nonlinear and non-Gaussian experiments. The first experiment is used to show that even though the improved framework is designed for multidimensional stochastic complex systems, it performs well for simple systems. The second experiment involves a real scene, which is designed to demonstrate the accuracy and efficiency of the improved framework when analyzing real multidimensional random systems.

5.1 Validation of B3DPF

Equation (10) is commonly used in PF experiments; it is highly nonlinear and bimodal in nature. To examine the non-Gaussian and multidimensional performance of the proposed improved framework, the state equation is updated as in Eq. (11), where $\alpha_t(\text{sigma}_t^{\alpha}) \sim N(0,\sigma_{\alpha}^2)$ and $\beta_t(\text{sigma}_t^{\beta}) \sim N(0,\sigma_{\beta}^2)$. In Eq. (11), x_t, x_t', and x_t'' represent multiple dimensions, and by adding x_t, x_t', and x_t'' and using them to form y_t, we obtain a non-Gaussian form of y_t. Equation (12) shows the method used to evaluate the performance of the PF for the i-th dimension, where x_{mk}^i is the real state at time step

m for the k-th simulation, $\hat{x}^i_{mk}$ is the estimated state, n is the number of time steps, N_s is the number of simulation replicates. Equation (13) is used to evaluate the overall simulation accuracy of the PF, where $x^i_{\max}$ and $\hat{x}^i_{\min}$ represent the maximum and minimum states for the i-th dimension. Equation (14) shows the average effective particle number during the PF process, where Neff_{mk} is the effective particle number at time step m for the k-th simulation. Equation (15) is used to evaluate the average coverage times, where rep_{mk} represents the replaced particle number during the resampling at time step m for the k-th simulation. Equation (16) is the average execution time, and speed_{mk} is the time consumed at time step m for the k-th simulation.

$$\begin{cases} x_t = 0.5x_t - 1 + \dfrac{25x_{t-1}}{1+x_{t-1}^2} + 8\cos[1.2(t-1)] + \alpha_t, \\ y_t = 0.05x_t^2 + b_t, t < 50 \end{cases} \tag{10}$$

$$\begin{cases} x_t = 0.5x_t - 1 + \dfrac{25x_{t-1}}{1+x_{t-1}^2} + 8\cos[1.2(t-1)] + \alpha_t, \\ x_t^{'} = 0.05x_{t-1}^{'} + 1/x_{t-1}^{'} + \alpha_t, \\ x_t^{''} = x_{t-1}^{''} + 4\sin[1.2(t-1)] + \alpha_t, \\ y_t = 0.05x_t^2 + b_t, t < 50, \\ y_t = 0.05x_t^2 - 2\sqrt{x_t^{'}} + b_t, t \geqslant 50 \end{cases} \tag{11}$$

$$\overline{u}^i_e = \frac{1}{N_s n}\sum_{m=1}^{N_s}\sum_{k=1}^{n} e^i_{mk}, e^i_{mk} = |x^i_{mk} - \hat{x}^i_{mk}| \tag{12}$$

$$\begin{cases} \tilde{u}_e = \dfrac{1}{3}\sum\limits_{i=1} 3\tilde{u}^i_e \cdot \mathrm{seg}_i/\mathrm{seg}, \\ \mathrm{seg} = \sum\limits_{i=1}^{3} \mathrm{seg}_i, \\ \mathrm{seg}_i = |x^i_{\max} - \hat{x}^i_{\min}| \end{cases} \tag{13}$$

$$\mathrm{Neff}_{\mathrm{ave}} = \frac{1}{N_s n}\sum_{m=1}^{N_s}\sum_{k=1}^{n}\mathrm{Neff}_{mk} \tag{14}$$

$$\mathrm{rep}_{\mathrm{ave}} = \frac{1}{N_s n}\sum_{m=1}^{N_s}\sum_{k=1}^{n}\mathrm{rep}_{mk} \tag{15}$$

$$\mathrm{speed}_{\mathrm{ave}} = \frac{1}{N_s n}\sum_{m=1}^{N_s}\sum_{k=1}^{n}\mathrm{speed}_{mk} \tag{16}$$

In the experiment, the initial state is set to $(x_0, x_0^{'}, x_0^{''}) = (10, 0.5, 0.1)$, and the initial particles are generated from a Gaussian distribution $x_0^i \sim N(x_0, \sqrt{2})$. Set $N_s = 50$, n=100, and the particle number as 500. The B3DPF and its optimization strategies are verified using the four experiments shown below.

Set $(\mathrm{sigma}_t^{\alpha}, \mathrm{sigma}_t^{\beta})$ as (1, 1), (10, 10), and (100, 100), and the experimental results are shown in Tables 1–3, where B3DPF is executed following the steps described in Section 4.6. B3DPF has a similar accuracy as PF and EPF when the system noise and measurement noise are small, but the execution speed is faster. As the noise increases, B3DPF becomes more accurate than the other methods, and the number of effective particles is stable. To verify

Table 1 Result of noise set to (1, 1).

	$x - \overline{u}_e^1$	$x^{'} - \overline{u}_e^2$	$x^{''} - \overline{u}_e^3$	$\tilde{u}_e$	$\mathrm{Neff}_{\mathrm{ave}}$	$\mathrm{speed}_{\mathrm{ave}}$
PF	3.2435	9.6898	12.1986	7.3684	298.75	0.447
UPF	2.9311	9.5437	11.2851	6.7857	312.23	0.812
B3DPF	2.9201	4.3561	5.4682	3.8605	344.44	0.523

Table 2 Result of noise set to (10, 10).

	$x - \overline{u}_e^1$	$x^{'} - \overline{u}_e^2$	$x^{''} - \overline{u}_e^3$	$\tilde{u}_e$	$\mathrm{Neff}_{\mathrm{ave}}$	$\mathrm{speed}_{\mathrm{ave}}$
PF	5.9325	10.6953	14.1254	8.9999	173.45	0.446
UPF	5.7435	9.9857	13.6825	8.6040	264.50	0.821
B3DPF	5.8234	5.3521	6.6189	5.8411	353.54	0.530

Table 3 Result of noise set to (100, 100).

	$x - \overline{u}_e^1$	$x^{'} - \overline{u}_e^2$	$x^{''} - \overline{u}_e^3$	$\tilde{u}_e$	$\mathrm{Neff}_{\mathrm{ave}}$	$\mathrm{speed}_{\mathrm{ave}}$
PF	12.4325	14.6953	16.1254	13.8499	54.72	0.456
UPF	11.7234	13.1842	14.5213	12.7212	192.90	0.811
B3DPF	7.7245	6.1369	8.2160	7.3465	245.03	0.521

the multidimensional weight algorithm, $(\mathrm{sigma}_t^{\alpha}, \mathrm{sigma}_t^{\beta})$ is set to (1, 1), the experiment is repeated four times-once using general B3DPF, and three times using the normal weight computation method with assimilated data x, $x\& x'$, and $x\& x'\& x''$. The experimental results are shown in Table 4. The use of fixed dimension assimilation data ensures the estimation accuracy of the selected dimensions, but assimilating all the dimensions takes too much time. In addition, B3DPF can maintain the accuracy of all dimensions with reasonable speed.

To verify the weight optimization algorithm, $(\mathrm{sigma}_t^{\alpha}, \mathrm{sigma}_t^{\beta})$ is set to (1, 1), and the experiment is repeated twice-once using the general B3DPF and once without a weight optimization algorithm. The experimental results are shown in Table 5. The weight optimization algorithm can reduce the number of replaced particles and maintains the diversity of the particle set. In addition, it has a slight advantage with respect to simulation accuracy.

To verify the stochastic model, $(\mathrm{sigma}_t^{\alpha}, \mathrm{sigma}_t^{\beta})$ is set to (1, 1), (10, 10), and (100, 100), and each experiment is repeated twice-once using general B3DPF and once without a stochastic model. The experimental results are shown in Table 6. The stochastic model can maintain the diversity of the particle set. A comparison of the simulation results with different noise levels evidently shows that B3DPF improves the estimation accuracy when the system and measurement noise are relatively large.

5.2 Application experiment of B3DPF for traffic state estimation

The identical-twin experiment is widely used in dynamic data-driven simulation and data assimilation research. It is adopted in this paper to evaluate the B3DPF for traffic state estimation. Thus, we can study dynamic data-driven event reconstruction in ideal situations and evaluate the proximity of the prediction to the true states in a controlled manner without considering the restriction of the sensor conditions. A simulation is run, related data are recorded, and the results are considered the "real" data. These "real" data are then assimilated into the dynamic data-driven event reconstruction system to predict the system states. Consequently, we check whether the predicted states are close to the "true" states.

To evaluate the effectiveness of the improved framework in multidimensional random state estimation, a second experiment is performed to estimate the traffic state on a real road network from the Ming Palace to Zhongshan Gate in Nanjing in Fig. 5 (left). The road network is complex and intersects with the first commercial circle—Xinjiekou. The final road network defined in the simulation mode is shown in Fig. 5 (right), which is drawn based on fieldwork and Google Maps. In this figure, 10 vehicle entrances are shown. Vehicles from R71, R34, R7, and R23 can directly affect the traffic flow on the marked road. Assume that four types of vehicles are traveling on the road network, as shown in Table 7. InitS is the initial speed, IdealS is the ideal speed, MinI is the minimum interval between vehicles, Acc is acceleration, and Dec is deceleration. Table 8 shows the inflow of each entrance. Assume a number of sensors are deployed on the road network with a uniform distribution; the marked road has four sensors at 200, 400, 600, and 800 m.

In this experiment, we use the average speed and the density of the vehicles as the input data. As the simulation runs, only data from the four sensors can be assimilated to update the simulation results. The $\mathrm{Dis}_t^i(j)$ used to calculate the particle is shown in Eq. (17), and $s_t^i(j)$ and $\mathrm{sum}_t^i(j)$ are shown in Eqs. (18) and (19), where $\mathrm{rs}_t^i(j)$ is the sensor data of the j-th dimension state that corresponds

Table 4 Multidimensional weight in B3DPF.

	$x-\overline{u}_e^1$	$x'-\overline{u}_e^2$	$x''-\overline{u}_e^3$	$\tilde{u}_e$	$\mathrm{Neff}_{\mathrm{ave}}$	$\mathrm{speed}_{\mathrm{ave}}$
B3DPF	2.9132	4.5711	5.4312	3.9142	355.84	0.524
B3DPF-NoMW-1	2.8876	9.8641	10.5743	6.5179	347.65	0.503
B3DPF-NoMW-2	2.9345	5.6754	11.5871	5.4873	351.28	0.533
B3DPF-NoMW-3	3.1023	5.8953	7.5798	4.8357	334.61	0.619

Table 5 Weight optimization in B3DPF.

	$\mathrm{rep}_{\mathrm{ave}}$	$\tilde{u}_e$	$\mathrm{Neff}_{\mathrm{ave}}$	$\mathrm{speed}_{\mathrm{ave}}$
B3DPF	374	3.9142	360.12	0.526
B3DPF-NoWO	458	4.3128	312.57	0.501

Table 6 Stochastic model in B3DPF.

	$\mathrm{rep}_{\mathrm{ave}}$	$\tilde{u}_e$	$\mathrm{Neff}_{\mathrm{ave}}$	$\mathrm{speed}_{\mathrm{ave}}$
B3DPF (1,1)	374	3.9137	356.52	0.527
B3DPF-NoRM (1,1)	384	4.2132	341.34	0.514
B3DPF (10,10)	394	5.8721	356.47	0.531
B3DPF-NoRM (10,10)	381	5.9012	281.43	0.517
B3DPF (100,100)	434	7.4798	247.63	0.524
B3DPF-NoRM (100,100)	438	8.1756	201.32	0.512

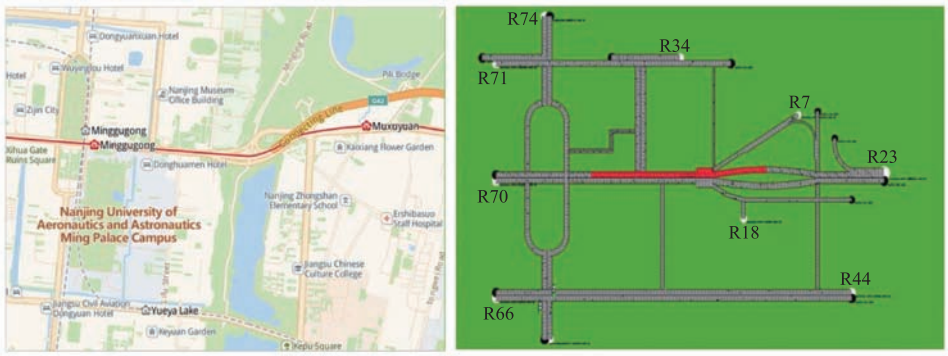

Fig. 5 Road network from the Ming Palace to Zhongshan Gate in Nanjing.

Table 7 Basic parameters of different types of vehicles.

Name	Model	Length (m)	InitS (km/h)	IdealS (km/h)	MinI (m)	Acc (m/s^2)	Dec (m/s^2)
Car	IDM	4.0	25	50	2	2.5	4.0
Small bus	IDM	7	15	45	3	2.5	4.0
Medium bus	IDM	10	10	40	4	2.0	3.5
Large bus	IDM	12	10	35	5	2.0	3.0

Table 8 Inflow of each entrance.

Entrance Id	Inflow (veh/h)	Entrance Id	Inflow (veh/h)
R7	100	R45	500
R18	200	R66	100
R23	1200	R70	800
R34	500	R71	500
R44	100	R74	800

to the position of the four sensors, $\mathrm{es}_t^i(j)$ is the estimated data, and $\mathrm{rs}_t^i(j,k)$ and $\mathrm{es}_t^i(j,k)$ are the k-th real and simulated data of the four sensors.

$$\mathrm{Dis}_t^i(j) = 1/(\lceil s_t^i(j)\rceil \cdot \lceil \mathrm{sum}_t^i(j)\rceil \tag{17}$$

$$s_t^i(j) = |\max(\mathrm{rs}_t^i(j)) - \max(\mathrm{es}_t^i(j))| \tag{18}$$

$$\mathrm{sum}_t^i(j) = \sum_{k=1}^{4} |\mathrm{rs}_t^i(j,k) - \mathrm{es}_t^i(j,k)| \tag{19}$$

A single-thread simulation is initialized based on the road network and vehicle information, and a breakdown point is set at a random position within the range of the marked road. Congestion occurs due to breakdown of the lane. The simulation is run for 300 s, and the vehicle parameters at each time are recorded as the "real" sensor data. The breakdown point is then removed, and the simulation is restarted with 10 simulation threads. In this experiment, SIS, UPF, and DPF are used to assimilate the real data, and the results are shown below.

The Neff in each algorithm is shown in Fig. 6. SIS suffers from the particle degeneracy problem, and the particles lose diversity in PF after the iteration of resampling. UPF and B3DPF maintain the particle diversity within an appropriate range, while B3DPF reduces the number of resampling operations. Figures 7 and 8 show the vehicle density and average speed along

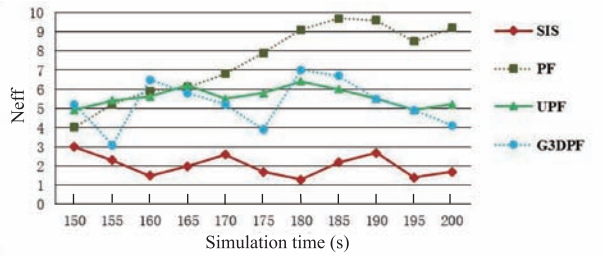

Fig. 6 Effective particle numbers in different PF algorithms.

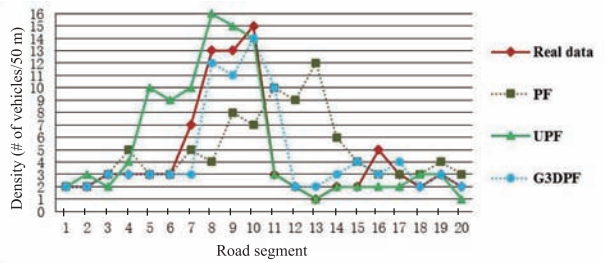

Fig. 7 Vehicle density at 180 s.

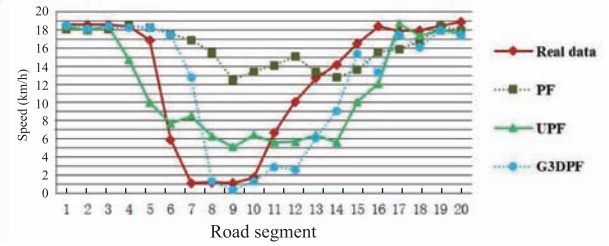

Fig. 8 Vehicle average speed at 180 s.

the marked road at 180 s. We can see that the traffic state depicted by using B3DPF is closest to the real scenario. The performance of the different PF algorithms is displayed in Table 9. B3DPF simulates the traffic flow with great accuracy and high speed, which indicates that the dynamic data-driven methods used in weight computation, resampling, and the random model solve the particle degeneracy and impoverishment problems and accelerate the execution time.

6 Conclusion

This paper presents a bidirectional feedback dynamic data-driven improved framework for PF called B3DPF. B3DPF is designed to solve the particle degeneracy and impoverishment problems while also considering the computational complexity and full-dimensional data assimilation, which make B3DPF compatible for multidimensional random application scenarios. We summarized three common improvements for PF but found that each improvement has its unavoidable limitations. Thus, after analyzing the key points of the particle degeneracy and impoverishment problems, the bidirectional feedback improvement for PF is proposed.

In B3DPF, weight computation is optimized to assimilate multidimensional sensor data according to specific control information. The data dimensions can be adjusted dynamically, and the resampling algorithm is improved by adding a weight optimization step when the number of effective particles is relatively small. In addition to the optimization strategies, we introduced a multidimensional sensor data management module and a stochastic module. The data management module can collect all the data from a real scene and store the data by using a sliding window method. Moreover, it can supply designated dimensions of data for data assimilation. The stochastic module can execute reasonable random adjustment for the replaced particles, optimize the simulation noise, and measure the noise dynamics.

The B3DPF is verified and applied to two experiments. The first experiment verified that each B3DPF optimization strategy can play its respective role and be combined to overcome particle degeneracy and maintain particle diversity. In addition, B3DPF can accurately estimate the state with faster speed than UPF, especially when analyzing multidimensional random scenes. The second experiment used B3DPF for traffic state estimation. Simulation results show that B3DPF can estimate vehicle density and speed more accurately and faster than PF and UPF.

Table 9 Performance of the PF algorithms.

Algorithm	RMSE-Density	RMSE-Speed	Runtime (s)
PF	9.523	6.231	220
UPF	4.112	4.210	200
B3DPF	4.231	3.486	180

Acknowledgment

This work was supported by the State Basic Scientific Research of National Defense (No. c0420110005), 13th Five-Year Key Basic Research Project (No. JCKY2016206B001), and the Six talent peaks project in Jiangsu Province (No. XXRJ-004).

References

[1] G. Battistelli, A. Benavoli, and L. Chisci, Data-driven communication for state estimation with sensor networks, *Automatica*, vol. 48, no. 5, pp. 926–935, 2012.

[2] C. Antoniou, H. N. Koutsopoulos, and G. Yannis, Dynamic data-driven local traffic state estimation and prediction, *Tramsportation Research Part C Emerging Technologies*, vol. 34, no. 9, pp. 89–107, 2013.

[3] F. Schimandl, M. Zhang, M. Mustafa, and L. Meng, Real time application for traffic state estimation based on large sets of floating car data, *Journal of Cellular Biochemistry*, vol. 102, no. 1, pp. 52–63, 2009.

[4] S. Y. Chen, Kalman filter for robot vision: A survey, *IEEE Trans. Ind. Electron*., vol. 59, no. 11, pp. 4409–4420, 2012.

[5] B. L. Li, G. L. Wei, J. J. Bu, Y. Yuan, J. Xu, B. Wang, H. Zeng, and J. R. Wang, Study on simulation and application of Kalman filter arithmetic in integrated navigation system, presented at the 2$^{\mathrm{nd}}$ Int. Conf. Measurement, Information and Control, Harbin, China, 2013.

[6] G. F. Welch, HISTORY: The use of the Kalman filter for human motion tracking in virtual reality, *Presence*, vol. 18, no. 1, pp. 72–91, 2009.

[7] S. J. Julier and J. K. Uhlmann, Unscented filtering and nonlinear estimation, *Proc. IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[8] 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.

[9] S. R. Chowdhury, D. Roy, and D. M. Vasu, Variance-reduced particle filters for structural system identification problems, *J. Eng. Mech*., vol. 139, no. 2, pp. 210–218, 2013.

[10] F. A. Ruslan, Z. M. Zain, R. Adnan, and A. M. Samad, Flood water level prediction and tracking using particle filter algorithm, presented at the 8$^{\mathrm{th}}$ Int. Colloquium on Signal Processing and its Applications, Melaka, Malaysia, 2012.

[11] T. C. Li, T. P. Sattar, Q. Han, and S. D. Sun, Roughening

methods to prevent sample impoverishment in the particle PHD filter, presented at the 16th Int. Conf. Information Fusion, Istanbul, Turkey, 2013.

[12] J. Zhu, X. L. Wang, and Q. S. Fang, The improved particle filter algorithm based on weight optimization, presented at 2013 Int. Conf. Information Science and Cloud Computing Companion, Guangzhou, China, 2013.

[13] F. J. Pei, P. Y. Cui, and Y. Z. Chen, Adaptive MCMC particle filter for nonlinear and Non-Gaussian state estimation, presented at the 3rd Int. Conf. Innovative Computing Information and Control, Dalian, China, 2008.

[14] Y. Zhai, X. B. Guo, and Y. G. Yan, Unscented particle filter algorithm for ballistic target tracking, *Appl. Mech. Mater.*, vol. 686, pp. 359–362, 2014.

[15] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, The unscented particle filter, presented at the 13th Int. Conf. Neural Information Processing Systems, Denver, CO, USA, 2001, pp. 563–569.

[16] X. W. Feng, X. F. Yan, and X. L. Hu, Dynamic data driven particle filter for agent-based traffic state estimation, presented at the 1st Int. Conf. Cloud Computing and Security, Nanjing, China, 2015.

[17] Z. Y. Wang, Z. T. Liu, W. Q. Liu, and Y. B. Kong, Particle filter algorithm based on adaptive resampling strategy, presented at 2011 Int. Conf. Electronic & Mechanical Engineering and Information Technology, Harbin, China, 2011.

[18] J. M. Hammersley and K. W. Morton, Poor man's Monte Carlo, *J.Roy. Stat. Soc. Ser. B*, vol. 16, no. 1, pp. 23–38, 1954.

[19] J. E. Handschin and D. Q. Mayne, Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering, *Int. J. Control*, vol. 9, no. 5, pp. 547–559, 1969.

[20] V. S. Zaritskii, V. B. Svetnik, and L. I. Shimelevich, Monte-Carlo technique in problems of optimal information processing, *Automat. Remote Control*, vol. 36, no. 12, pp. 95–103, 1975.

[21] H. Akashi and H. Kumamoto, Random sampling approach to state estimation in switching environments, *Automatica*, vol. 13, no. 4, pp. 429–434, 1977.

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

[23] J. Carpenter, P. Clifford, and P. Fearnhead, Improved particle filter for nonlinear problems, *IEE Proc. Radar Sonar Navig.*, vol. 146, no. 1, pp. 2–7, 1999.

[24] J. X. Yu, W. J. Liu, and Y. L. Tang, Improved particle filter algorithms based on partial systematic resampling, presented at 2010 IEEE Int. Conf. Intelligent Computing and Intelligent Systems, Xiamen, China, 2010.

[25] F. S. Wang and Y. J. Lin, Improving particle filter with a new sampling strategy, presented at the 4th Int. Conf. Computer Science & Education, Nanning, China, 2009.

[26] Z. Y. Wang, Z. T. Liu, W. Q. Liu, and Y. B. Kong, Particle filter algorithm based on adaptive resampling strategy, presented at 2011 Int. Conf. Electronic & Mechanical Engineering and Information Technology, Harbin, China, 2011.

[27] B. L. Xu, Q. L. Chen, J. H. Zhu, and Z. Q. Wang, Ant estimator with application to target tracking, *Signal Process.*, vol. 90, no. 5, pp. 1496–1509, 2011.

[28] B. Xu, Q. Chen, J. Zhu, and Z. Wang, A new particle filter inspired by biological evolution: Genetic filter, *Enformatika*, pp. 459, 2007.

[29] T. C. Li, S. D. Sun, T. P. Sattar, and J. M. Corchado, Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches, *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3944–3954, 2014.

[30] E. E. Prudencio, P. T. Bauman, S. V. Williams, D. Faghihi, K. Ravi-Chandar, and J. T. Oden, A dynamic data driven application system for real-time monitoring of stochastic damage, *Procedia Comput. Sci.*, vol. 18, pp. 2056–2065, 2013.

[31] N. Celik, S. Lee, K. Vasudevan, and Y. J. Son, DDDAS-based multi-fidelity simulation framework for supply chain systems, *IIE Trans.*, vol. 42, no. 5, pp. 325–341, 2010.

[32] A. Patra, M. Bursik, J. Dehn, M. Jones, M. Pavolonis, E. B. Pitman, T. Singh, P. Singla, and P. Webley, A DDDAS framework for volcanic Ash propagation and hazard analysis, *Procedia Comput. Sci.*, vol. 9, pp. 1090–1099, 2012.

[33] A. Vodacek, J. P. Kerekes, and M. J. Hoffman, Adaptive optical sensing in an object tracking DDDAS, *Procedia Comput. Sci.*, vol. 9, pp. 1159–1166, 2012.

[34] G. R. Madey, M. B. Blake, C. Poellabauer, H. S. Lu, R. R. McCune, and Y. Wei, Applying DDDAS principles to command, control and mission planning for UAV swarms, *Procedia Comput. Sci.*, vol. 9, pp. 1177–1186, 2012.

[35] F. Darema, Introduction to the ICCS2006 workshop on dynamic data driven applications systems, presented at the 6th Int. Conf. Computer Science, Reading, UK, 2007, pp. 375–383.

[36] J. S. Liu, Metropolized independent sampling with comparisons to rejection sampling and importance sampling, *Stat. Comput.*, vol. 6, no. 2, pp. 113–119, 1996.

[37] D. H. Lehmer, Mathematical methods in large-scale computing units, presented at the 2nd Symp. Large-Scale Digital Calculating Machinery, Cambridge, The United Kingdom, 1951, pp. 141–146.

Xuefeng Yan received the PhD degree from Beijing Institute of Technology, China, in 2005. He is currently a professor with the College of Computer Science and Technology at Nanjing University of Aeronautics and Astronautics. His research interests include modeling and simulation, software engineering, complex system engineering, and big data analysis.

Xiaolin Hu received the PhD degree from University of Arizona, USA, in 2004. He is currently an associate professor with the Department of Computer Science of Georgia State University, Atlanta. His research interests include modeling and simulation theory and application, agent and multi-agent systems, complex systems science, and simulation-based design methods.

Chengbo Song received the MS degree from Nanjing University of Aeronautics and Astronautics, China, in 2018. His research interests are mainly edge system modeling and simulation, system engineering, and big data analysis.

Xiangwen Feng received the MS degree from Nanjing University of Aeronautics and Astronautics, China, in 2016. His research interests are mainly edge system modeling and simulation, system engineering, and big data analysis.