Hindawi
Journal of Electrical and Computer Engineering
Volume 2021, Article ID 2838971, 8 pages
https://doi.org/10.1155/2021/2838971

Research Article

Research on the Particle Filter Single-Station Target Tracking Algorithm Based on Particle Number Optimization

Lieping Zhang,[1] **Jinghua Nie**,[1] **Shenglan Zhang**,[1] **Yanlin Yu**,[1] **Yong Liang**,[1] **and Zuqiong Zhang**[2]

[1]*College of Mechanical and Control Engineering, Guilin University of Technology, Guilin 541004, China*
[2]*Network and Information Center, Guilin University of Technology, Guilin 541004, China*

Correspondence should be addressed to Zuqiong Zhang; zzq@glut.edu.cn

Received 24 May 2021; Revised 9 August 2021; Accepted 24 August 2021; Published 6 September 2021

Academic Editor: Yang Li

Given that the tracking accuracy and real-time performance of the particle filter (PF) target tracking algorithm are greatly affected by the number of sampled particles, a PF target tracking algorithm based on particle number optimization under the single-station environment was proposed in this study. First, a single-station target tracking model was established, and the corresponding PF algorithm was designed. Next, a tracking simulation experiment was carried out on the PF target tracking algorithm under different numbers of particles with the root mean square error (RMSE) and filtering time as the evaluation indexes. On this basis, the optimal number of particles, which could meet the accuracy and real-time performance requirements, was determined and taken as the number of particles of the proposed algorithm. The MATLAB simulation results revealed that compared with the unscented Kalman filter (UKF), the single-station PF target tracking algorithm based on particle number optimization not only was of high tracking accuracy but also could meet the real-time performance requirement.

1. Introduction

Target tracking refers to the process of state modeling, filtering estimation, and continuous tracking of a moving target by all kinds of observation means and computing methods. In fact, it aims to solve the target state estimation problem through the filtering approach [1]. As an important constituent part of a target tracking system, filtering algorithm directly decides the target tracking accuracy and has a great bearing on the real-time performance of the tracking system. Hence, filtering algorithm has always been a difficult and key problem in the studies regarding target tracking.

Filtering theory refers to the method estimating the system state by an optimal statistical approach according to certain filtering rules based on the observed signal measurement of this system [2]. Filtering algorithms play an important role in dealing with target tracking [3–5], including some classical algorithms, such as Kalman filter (KF), extended Kalman filter (EKF), unscented Kalman filter (UKF), and particle filter (PF). In recent years, various improvement schemes have been proposed by domestic and foreign scholars based on the KF theoretical framework, and gratifying progress has been achieved. Ning et al. [6] put forward the EKF target tracking optimization algorithm, which, compared with the traditional EKF algorithm only considering the position measurement, improved the target tracking effect. Xu and Peng [7] proposed the weight adaptive Gaussian and KF target tracking algorithm with nonlinear constraints and corrected the state estimation of moving targets according to the prior information of state constraints. This algorithm shows a higher tracking accuracy in comparison to the traditional KF algorithm with nonlinear constraints. Jondhale and Deshpande [8] presented a modified KF based approach of real-time tracking of single target moving in 2D in wireless sensor network (WSN), which can deal with uncertainties in measurement noises and abrupt changes in target velocity. Jondhale and Deshpande [9] proposed real-time target tracking algorithm based on general regression neural network and KF Framework in WSN. However, for a strong nonlinear system

or the condition when the process noise and observation noise are not Gaussian white noises, the filtering algorithm under the KF framework fails to ensure stable tracking performance and can easily cause a large tracking error, which, in a severe case, will even be divergent [10]. Particle Filter (PF), an algorithm based on Monte Carlo method, is capable of solving the filtering problem under the nonlinear non-Gaussian scenario that cannot be solved by the KF algorithm [11]. In order to guarantee the particle diversity, Li et al. [12] proposed a deterministic resampling PF algorithm so that the low-weight particles would not be blindly discarded. Park et al. [13] applied the idea of particle swarm optimization and ant colony optimization algorithms to the sample updating of PF algorithm, thus realizing the information sharing between particles and enhancing the overall optimization ability of this algorithm. In order to improve the effective accuracy of PF algorithm in state estimation, Xu et al. [14] established error ellipses with different confidence levels during the resampling process according to the error covariance matrix of particles and applied them to target positioning and tracking of wireless sensor network. Lin and Suo [15] improved the PF algorithm from two aspects: adaptive inertia weight and learning factor, and used the improved algorithm to optimize the PF algorithm, thus reaching the goal of reducing the target tracking error.

In this study, the PF-based single-station target tracking algorithm was explored, and the optimal number of particles of PF algorithm was determined through the simulation experiment. Different from the existing irregular selection of the number of particles for PF algorithm concerning the racking accuracy only, the proposed method could reach a balance between tracking accuracy and tracking velocity during particle number selection. Besides, the corresponding experimental method was designed for the optimal selection. In the end, a target tracking simulation experiment was carried out, and the proposed optimal selection method of the number of particles and PF target tracking algorithm were experimentally verified. The simulation experiment results showed that, compared with the UKF algorithm, the tracking trajectory of PF algorithm based on the particle number optimization was more approximate to the real trajectory, showing a smaller target tracking error. Moreover, this algorithm could simultaneously meet the requirements for the target tracking accuracy and real-time performance.

2. PF Algorithm

After acquiring some random samples with the corresponding weights under a selected importance probability density, the PF algorithm adjusts the weight and position of each particle according to the state observation, uses theses samples to approach the state posterior distribution, and finally takes their weighted sum as the estimated value of state [16].

For a discrete-time nonlinear dynamic system, the state transition equation of a target state sequence and observation equation of the system can be expressed as equations (1) and (2) [17].

$$x_k = f(x_{k-1}, w_{k-1}), \tag{1}$$

$$zt_k = h(x_k, v_k), \tag{2}$$

where x_k, z_k, w_{k-1}, and v_k stand for the state variable, observed value, process noise, and observation noise of the system, respectively. In particular, the filtering estimation value of x_k derives from all observed values $z_{1:\,k} = \{z_i, i = 1, \ldots, k\}$ at time $1 - k$.

The filtering problem means acquiring the estimated value of the quantity of state (x_k) through the observation data $z_{1-k} = \{z_i\}_{i=1}^{i=k}$ at time $1 - k$. Under the given $p(x_0 \mid z_0) = p(x_0)$, the probability density function of posterior state distribution $p(x_k \mid z_{1:\,k})$ can be acquired through the prediction equations (equations (3) and (4)).

$$p(x_k \mid z_{1:\,k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid z_{1:\,k-1}) \mathrm{d}x_{k-1}. \tag{3}$$

In equation (3), $p(x_k \mid x_{k-1})$ is called the transfer density function of state, which contains an assumption: $p(x_k \mid x_{k-1}) = p(x_k \mid x_{k-1}, z_{1:\,k-1})$.

$$p(x_k \mid z_{1:\,k}) = \frac{p(z_k \mid x_k) p(x_k \mid z_{1:\,k-1})}{p(z_k \mid z_{1:\,k-1})}. \tag{4}$$

In equation (4), the constant quantity $p(z_k \mid z_{1:\,k-1}) = \int p(z_k \mid x_k) p(x_k \mid z_{1:\,k-1}) \mathrm{d}x_k$ is codetermined by observation model equation (2) and statistical value $p(x_k \mid z_{1:\,k-1})$ predicted in the first step. In the updating phase, the observed value z_k in equation (4) is used to correct the prior overview, so as to obtain the posterior probability of current state.

Sampling from $p(x_k \mid z_{1:\,k})$ is approximately a special form of Monte Carlo method. In reality, a more general form is to sample from the complete posterior distribution $p(x_{0:\,k} \mid z_{1:\,k})$. For a general nonlinear system, it is difficult to sample directly from $p(x_{0:\,k} \mid z_{1:\,k})$ or obtain the analytical solution of posterior probability density function. A relatively popular solution to the aforementioned problem is the sequential importance sampling method [18]. When sampling at time k, this method does not change the past state sequence $x_{0:\,k-1}$, but instead, it calculates the importance weight in a recursive fashion. The importance density function can be decomposed into an expression as shown in the following equation:

$$q(x_{0:\,k} \mid z_{1:\,k}) = q(x_k \mid x_{0:\,k-1}, z_{1:\,k}) q(x_{0:\,k-1} \mid z_{1:\,k-1}). \tag{5}$$

The weight updating formula is as seen in the following equation:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(z_k \mid x_k^{(i)}\right) p\left(x_k^{(i)} \mid x_{k-1}^{(i)}\right)}{q\left(x_k^{(i)} \mid x_{k-1}^{(i)}, z_k\right)}. \tag{6}$$

Hence, the posterior probability density function can be approximated as shown in the following equation:

$$\widehat{p}(x_k \mid z_{1:\,k}) = \sum_{i=1}^{N} \widetilde{w}_k^{(i)} \delta\left(x_k - x_k^{(i)}\right), \tag{7}$$

where $\tilde{w}_k^{(i)}$ denotes the normalized weight and N is the number of samples.

3. Single-Station Target Tracking Modeling and Algorithm Implementation

3.1. Single-Station Target Tracking Modeling. Correct modeling of target motion state is of great importance for a model-based target tracking system, and the model quality directly influences the tracking effect. To simplify the problem, the common target motion under ideal conditions was modeled in this study, and the single-state single-target tracking modeling steps were as follows [19]:

With the uniform linear motion of a target in a two-dimensional (2D) space as an example, the target motion state was assumed as $x(k) = \left[x(k) \;\; v_x(k) \;\; y(k) \;\; v_y(k) \right]^T$, where $x(k)$ and $y(k)$ represent the target positions in the directions X and Y at time k, respectively. $v_x(k)$ and $v_y(k)$ denote the target velocity in directions X and Y at time k, respectively. The target was certainly subjected to noise interference in the motion process, and the interfering noise $w = \left[w_x \;\; w_{vx} \;\; w_y \;\; w_{vy} \right]^T$ could be regarded as random acceleration, as seen in the following equation:

$$\begin{cases} x(k) &= x(k-1) + v_x(k-1) \times T + \dfrac{1}{2} w_x(k) \times T^2, \\ v_x(k) &= v_x(k-1) + w_x(k) \times T, \\ y(k) &= y(k-1) + v_y(k-1) \times T + \dfrac{1}{2} w_y(k) \times T^2, \\ v_y(k) &= v_y(k-1) + w_y(k) \times T, \end{cases} \tag{8}$$

where T represents the sampling time. Equation (8) can be transformed into a matrix form as seen in the following equation:

$$\begin{bmatrix} x(k) \\ v_x(k) \\ y(k) \\ v_y(k) \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ v_x(k-1) \\ y(k-1) \\ v_y(k-1) \end{bmatrix} + \begin{bmatrix} 0.5T^2 & 0 \\ T & 0 \\ 0 & 0.5T^2 \\ 0 & T \end{bmatrix} \begin{bmatrix} w_x \\ w_y \end{bmatrix}. \tag{9}$$

The state equation model of the target in equation (10) could be obtained by further simplifying equation (9), where F is the state transition matrix and Γ is the noise-driven matrix.

$$x_k = F x_{k-1} + \Gamma w_{k-1}. \tag{10}$$

Equation (10) reflected the real motion information of the target, while the station was not aware of the target motion state, so it was necessary to establish an observation model for the target tracking system. In the single-station target tracking system, the station was assumed at the position $[x_0, y_0]$, where the target was generally detected and tracked by means of radio, infrared ray, ultrasonic wave, etc. In a distance-based target tracking system, the station position and target presented a relation as shown in the following equation:

$$z_k = \sqrt{(x_k - x_0)^2 + (y_k - y_0)^2} + v_k. \tag{11}$$

Similarly, for an orientation-based target tracking system, the observation equation as shown in equation (12) could be obtained according to the angle information between the target and station.

$$z_k = \arctan \frac{y_k - y_0}{x_k - x_0} + v_k. \tag{12}$$

In general, the observation model equations (10) and (11) were expressed as follows:

$$z_k = h(x_k) + v_k. \tag{13}$$

3.2. Single-Station PF Target Tracking Algorithm. It was assumed that the tracked target did uniform linear motion on a 2D plane and the station position was $[x_0, y_0]$, where a distance sensor was used. The target state equation and observation equation could be expressed as follows:

$$\begin{cases} x_k &= F x_{k-1} + \Gamma w_{k-1}, \\ z_k &= \sqrt{(x_k - x_0)^2 + (y_k - y_0)^2} + v_k, \end{cases} \tag{14}$$

where $x(k) = \left[x(k) \;\; v_x(k) \;\; y(k) \;\; v_y(k) \right]^T$ and $[x(k), \; y(k)]$ represent the target positions in directions X and Y at time k, respectively. $v_x(k)$ and $v_y(k)$ denote the target velocity in directions X and Y at time k, respectively. F is the state transition matrix, Γ is the noise-driven matrix, $w_{k-1} - N(0, Q_{k-1})$, $v_k - N(0, R_k)$, and

$$F = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \Gamma = \begin{bmatrix} 0.5T^2 & 0 \\ T & 0 \\ 0 & 0.5T^2 \\ 0 & T \end{bmatrix}. \tag{15}$$

On the precondition of already known state equation and observation equation of target motion, the single-station target tracking steps based on PF algorithm were as follows:

Step 1: initialization: N particles $\left\{ x_0^{(i)} \right\}_{i=1}^N$ were sampled according to the prior distribution $p(x_0)$.

Step 2: importance sampling: the prior density $p(x_k \mid x_{k-1}^{(i)})$ could be obtained according to the state equation of target motion. In general, the prior density was usually taken as the importance density function, namely, $q(x_k \mid x_{k-1}^{(i)}, z_k) = p(x_k \mid x_{k-1}^{(i)})$, and the particle $x_k^{(i)} - p(x_k \mid x_{k-1}^{(i)})$ could be sampled from this function.

Step 3: $p(z_k \mid x_k^{(i)})$ could be acquired according to the observation equation of target tracking, and the weight value $w_k^{(i)} = w_{k-1}^{(i)} p(z_k \mid x_k^{(i)})$ was further calculated and normalized as $\widetilde{w}_k^{(i)} = w_k^{(i)} / \sum_{i=1}^{N} w_k^{(i)}$.

Step 4: resampling: resampling should be implemented when the estimated number of effective samples was $\widehat{N}_{\text{eff}} < N_{\text{th}}$, and new particles were generated and the weight $w_k^{(i)} = \widetilde{w}_k^{(i)} = 1/N$ was reassigned.

Step 5: output: the state estimation at time k was calculated as $\widehat{x}_k = \sum_{i=1}^{N} \widetilde{w}_k^{(i)} x_k^{(i)}$.

Step 6: $k = k + 1$ was set, and Steps 2–5 were repeated.

4. Particle Number Selection Method in PF Algorithm

4.1. Experimental Environment Settings for Particle Number Selection. In the practical application of the traditional PF algorithm, except for a minority of particles, the normalized weights of most particles will tend to zero after several iterations, and this phenomenon is referred to as particle degeneracy [20]. Due to particle degeneracy, a large quantity of computing work is wasted in updating those particles almost unusable for the $\widehat{p}(x_k \mid z_{1:k})$ estimation. In case of particle degeneracy, the degeneracy can be mitigated by increasing the number of particles (N), which, however, will enlarge the calculated quantity and affect the real-time performance of the algorithm. In this study, the number of particles was selected through the experimental method in an effort to explore the influence of particle number on the performance of PF algorithm and realize particle number optimization. The simulation experiment on particle number optimization was implemented by selecting a typical single-variable nonstatic growth model [19]. This system, which was of high nonlinearity with double-peak likelihood function, could hardly be processed by the traditional KF method. Its state equation and observation equation are as seen in equations (16) and (17):

$$x_k = 0.5x_{k-1} + \frac{25x_{k-1}}{1 + x_{k-1}^2} + 8\cos[1.2(k-1)] + w_k, \quad (16)$$

$$z_k = \frac{x_k^2}{20} + v_k, \quad (17)$$

where both w_k and v_k are the zero-mean Gaussian white noises, and the process noise variance and observation noise variance are $Q = 1$ and $R = 1$, respectively [19]. In this simulation experiment, the system was assumed at an initial state of $x_0 = 1$, with the sampling time of $T = 1$ s and time step of 50. In order to verify the influence of particle number on the filtering effect, the number of particles was set as $N = 300$, $N = 500$, $N = 1000$, and $N = 2000$ in the simulation experiment.

4.2. Experimental Results for Particle Number Selection. To intuitively reveal the influence of particle number on the tracking accuracy, the Root Mean Square Error (RMSE) was used in this study as a performance index for evaluation. The RMSEs obtained after 100 times of Monte Carlo simulation are given in Figure 1, and this index was calculated as per the following equation:

$$\text{RMSE}(k) = \sqrt{\frac{\sum_{n=1}^{100} (x_k - \widehat{x}_k^n)^2}{100}}, \quad (18)$$

where x_k denotes the true value at time k, and $\widehat{x}_k^n$ represents the estimated value in the n^{th} Monte Carlo simulation at time k.

It could be seen from Figure 1 that when the number of particles was increased from 300 to 500 and then to 1000, the filtering accuracy was obviously enhanced. This is because the particle weight variance will be iterated with time and continuously increase owing to the particle degeneracy in the PF algorithm. The particle diversity could be reinforced and the particle degeneracy phenomenon could be effectively delayed by increasing the number of particles. Nevertheless, it could also be discovered from the figure that, in comparison with the situation under 1000 particles, the filtering accuracy was improved not obviously when the number of particles was 2000. This is because, with the continuous increase in the number of particles, the particle density was already very high, and the filtering accuracy cannot be effectively improved by purely increasing the number of particles.

The average RMSEs obtained through 100 times of Monte Carlo simulation experiments under $N = 300$, $N = 500$, $N = 1000$, and $N = 2000$ are listed in Table 1. It could be seen from the table that the estimation accuracy was improved by 38% when N was increased from 300 to 500 and by 10% when N was increased from 500 to 1000 but not obviously improved when N was increased from 1000 to 2000.

The simulation experiment results indicated that when the number of particles was elevated, the particle degeneracy problem could be solved to some extent, and the PF accuracy could be improved. The large number of particles required strong data storage and computing power, which certainly lengthened the algorithm operation time and degraded the computing efficiency. The changes in operation time under the increasing number of particles should also be compared in order to achieve good real-time performance. The comparison of real-time performance under the increasing number of particles is as shown in Figure 2.

As shown in Figure 2, when the number of particles was increased from 300 to 500, 1000 and 2000, the needed filtering time was lengthened from the original 0.01 s to 0.02 s, 0.04 s and 0.07 s, which were 2, 4, and 7 times that of the original filtering time. However, it could be known from Figure 1 that the filtering accuracy was improved not evidently under 2000 particles in comparison to the situation under 1000 particles, so the situation with 2000 particles was not considered. In order to improve the computing efficiency, the number of sampled particles could be properly reduced to improve the real-time performance while meeting the tracking accuracy requirement. Through the conjoint analysis of Figures 1 and 2 and Table 1, it could be seen that the estimation accuracy was apparently improved

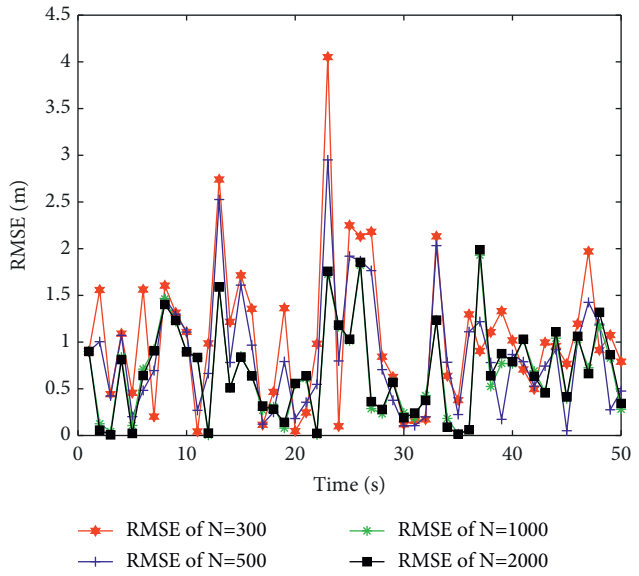

Figure 1: RMSE comparison under the increasing number of particles.

Table 1: Average RMSEs under the increasing number of particles.

	$N = 300$	$N = 500$	$N = 1000$	$N = 2000$
RMSE	1.2474	0.7692	0.6941	0.6885

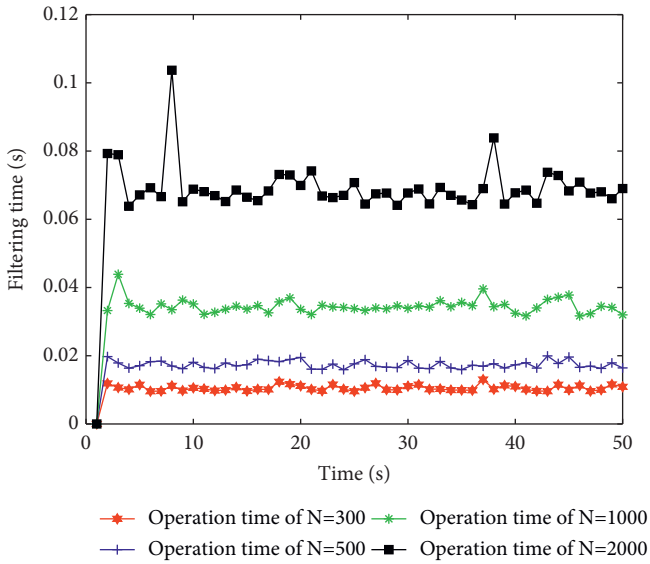

Figure 2: Comparison of real-time performance under the increasing number of particles.

under $N = 500$ in comparison with that under $N = 300$. But the computing burden was not increased greatly, and the time consumed was always shorter than 0.02 s, thus meeting the real-time performance requirement. Given this, the experiment was implemented in this study under $N = 500$, in order to balance the computing efficiency and estimation accuracy requirement.

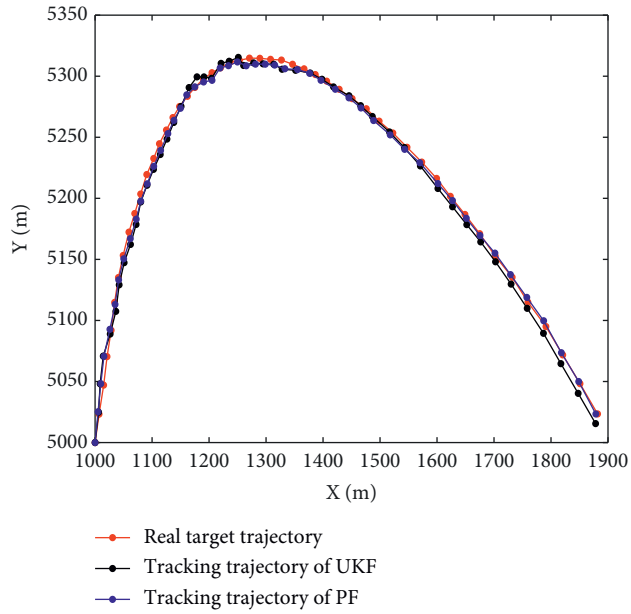

Figure 3: Real target trajectory and tracking trajectories under different algorithms.

5. Target Tracking Experiment and Result Analysis

5.1. Environment Settings for the Target Tracking Experiment. The parameters of targets in the uniformly accelerated linear motion process on the 2D plane were set as follows: The initial value of target motion was $x_0 = [\,1000\,\text{m}\ \ 5000\,\text{m}\ \ 10\,\text{m/s}\ \ 50\,\text{m/s}\ \ 2\,\text{m/s}^2\ \ -4\,\text{m/s}^2\,]^T$, the station could be at any position, which was set as $[1350\,\text{m}, 5150\,\text{m}]$, and the sampling time was set as $T = 0.5\,\text{s}$ and the time step as 50. Meanwhile, the correlation coefficients in the UT transformation were set as $\alpha = 0.01$, $\kappa = 0$, and $\beta = 2$ [21, 22], and the number of sampled particles was set as $N = 500$. The noise parameters in the target motion process were set as follows [23]: The process noise variance, which was calculated as per equation (19), was Q_k and the observation noise variance was $R_k = 5^2$.

$$Q_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01^2 \end{bmatrix}. \tag{19}$$

5.2. Analysis of Target Tracking Experiment Results. The UKF algorithm was comparatively analyzed with the PF algorithm to verify the algorithm performance. The target tracking effects of the two algorithms are shown in Figure 3. It could be observed from Figure 3 that the later-stage tracking trajectory of PF target tracking algorithm slightly deviated from the real target trajectory, while that of the UKF

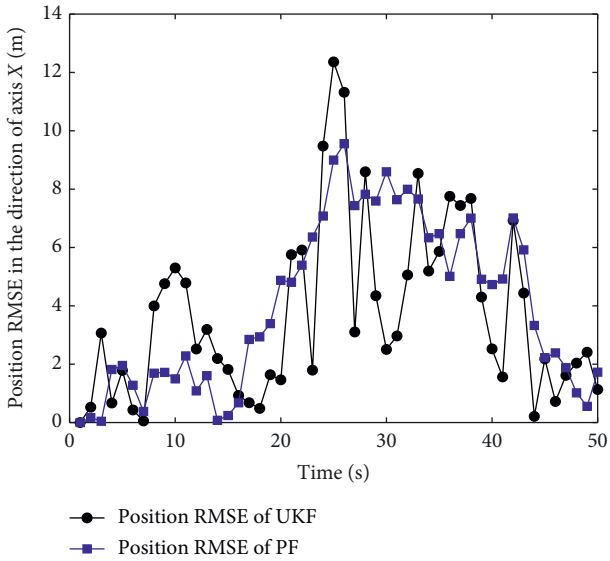

Figure 4: RMSE curves of the target position in the direction of axis X.

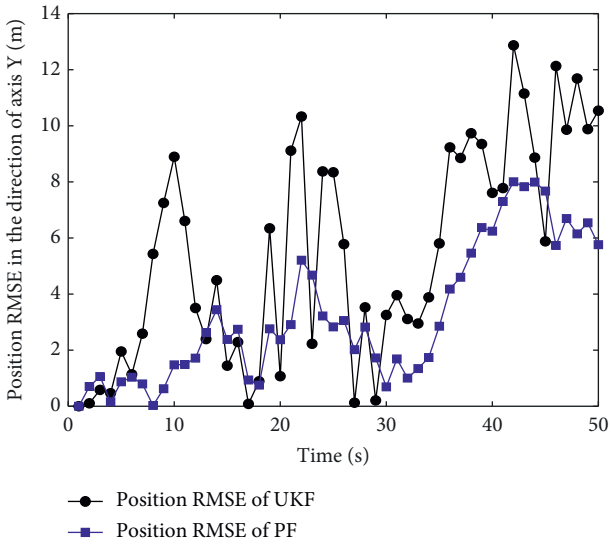

Figure 5: RMSE curves of the target position in the direction of axis Y.

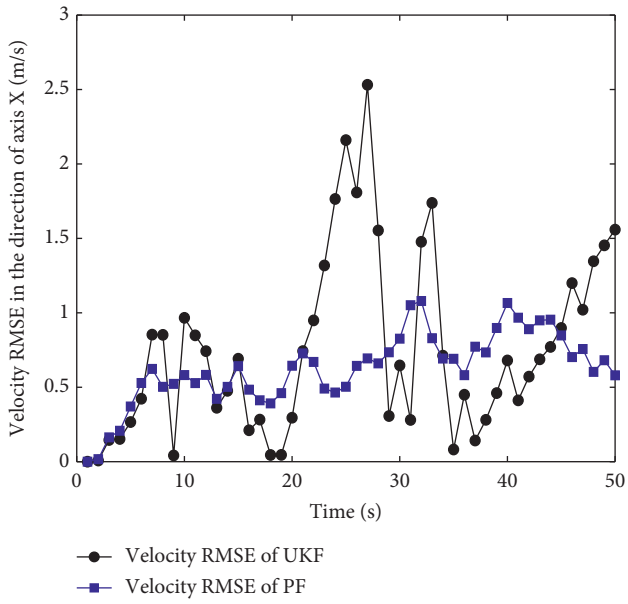

Figure 6: RMSE curves of target velocity in the direction of axis X.

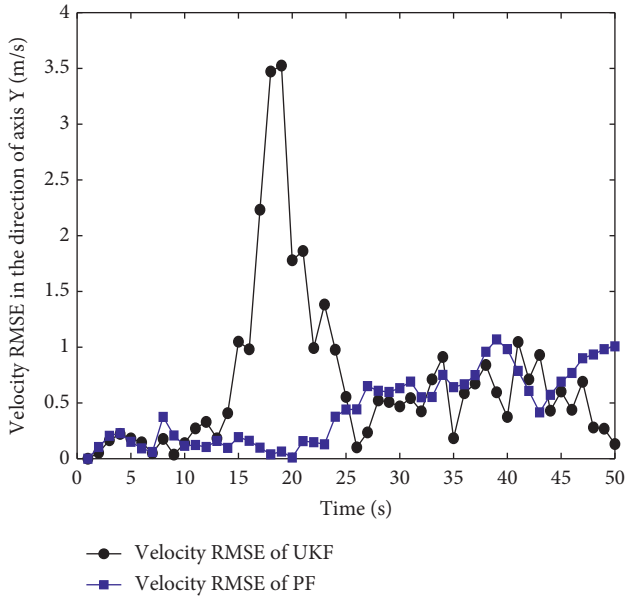

Figure 7: RMSE curves of target velocity in the direction of axis Y.

algorithm deviated a lot from the true estimation of the target nearby the coordinates [1300 m 5300 m], and it still experienced a large deviation from the real motion trajectory in the later tracking stage.

In order to intuitively display the tracking effects of the two algorithms, the RMSE of target position was compared with that of target velocity. The RMSEs calculated through 100 times of Monte Carlo simulation are displayed in Figures 4–7.

It could be seen from Figures 4–7 that both PF algorithm and UKF algorithm could roughly estimate the variation trends of target position and velocity. Although not every estimation result obtained through the PF algorithm was more accurate than that obtained through the UKF algorithm, the overall position and velocity RMSEs were smaller than those in UKF algorithm, and the tracking performance was better.

The operation time of a tracking algorithm is directly related to the tracking real-time performance and also an important index in the effect analysis of target tracking algorithm. The average position and velocity RMSEs and single operation time of UKF algorithm and PF algorithm in directions X and Y during the 100 Monte Carlo experiments are presented in Table 2.

It could be seen from Table 2 that the PF filtering algorithm should be chosen if the tracking accuracy was the primary factor to consider. However, if the tracking real-time performance was the primary goal, the UKF

Table 2: Average RMSE and single operation time comparison of UKF and PF algorithms.

Evaluation index	UKF algorithm	PF algorithm
Average position RMSE in axis X (m)	4.4780	4.3317
Average position RMSE in axis Y (m)	4.3105	3.4626
Average velocity RMSE in axis X (m/s)	0.7598	0.6267
Average velocity RMSE in axis Y (m/s)	0.6880	0.4415
Operation time of single Monte Carlo simulation (s)	0.0086	1.8016

algorithm was preferred, as it is of the minimum computing burden and best real-time performance. By analyzing the data in the X direction in Table 2, it could be known that both position accuracy and velocity accuracy of the PF algorithm were superior to those of the UKF algorithm, but it required a larger calculated quantity with longer operation time and poorer real-time performance. Nevertheless, the real-time performance requirement could still be satisfied.

6. Conclusion

(1) In this study, the number of particles was experimentally selected, followed by a comparison with the target tracking effect of the traditional UKF algorithm. The experimental results revealed that if the tracking accuracy is the primary factor to consider, the PF filtering algorithm is a better choice, but the UKF algorithm will be preferred if the tracking real-time performance is the primary goal. These conclusions are of certain reference values for studying the target tracking technology based on the filtering algorithm under the single-station condition.

(2) During the experimental process, only two circumstances—uniform linear and uniformly accelerated linear motions—were discussed for the target motion model. But the motion model may be very complex if the target is under complex maneuvering motion status. Given this, the complex maneuvering motion can be explored in the follow-up study by combining the PF target tracking algorithm and KF filtering algorithm, so as to overcome their respective deficiencies and improve the tracking effect.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61741303), the Key Laboratory of Spatial Information and Geomatics (Guilin University of Technology) (no. 19-185-10-08), and Scientific Research Basic Ability Improvement Project of Young and Middle-Aged Teachers in Colleges and Universities in Guangxi (no. 2021KY0260).

References

[1] J. G. Chen, *Filtering Methods in Target Tracking System*, Xidian University Press, Xi'an, China, 2013.

[2] T. Çimen and A. O. Merttopçuoğlu, "Asymptotically optimal nonlinear filtering: theory and examples with application to target state estimation," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 8611–8617, 2008.

[3] Cheng, X. T. Zhang, and Y. Q. Fang, "Target tracking estimation based on diffusion kalman filtering algorithm," *Computer Applications and Software*, vol. 38, no. 2, pp. 191–197, 2021.

[4] W. Zhou and J. Hou, "A new adaptive robust unscented kalman filter for improving the accuracy of target tracking," *IEEE Access*, vol. 7, pp. 77476–77489, 2019.

[5] S. S. Moghaddasi and N. Faraji, "A hybrid algorithm based on particle filter and genetic algorithm for target tracking," *Expert Systems with Applications*, vol. 147, Article ID 113188, 2020.

[6] Q. H. Ning, Y. B. Zhang, L. Liu, Z. Lu, and B. T. Guo, "Optimization algorithm for target tracking based on extended kalman filtering," *Journal of Detection & Control*, vol. 38, no. 1, pp. 90–94, 2016.

[7] Z. Xu and L. Peng, "Target tracking algorithm based on adaptive gaussian sum kalman filtering with nonlinear constraints," *Computer Measurement & Control*, vol. 27, no. 6, pp. 241–246, 2019.

[8] R. Jondhale and R. S. Deshpande, "Modified kalman filtering framework based real-time target tracking against environmental dynamicity in wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 40, pp. 119–143, 2018.

[9] S. R. Jondhale and R. S. Deshpande, "Kalman filtering framework-based real time target tracking in wireless sensor networks using generalized regression neural networks," *IEEE Sensors Journal*, vol. 19, no. 1, pp. 224–233, 2019.

[10] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, 2000.

[11] X. Wang, C. Xu, S. Duan, and J. Wan, "Error-ellipse-resampling-based particle filtering algorithm for target tracking," *IEEE Sensors Journal*, vol. 20, no. 10, pp. 5389–5397, 2020.

[12] T. Li, T. P. Sattar, and S. Sun, "Deterministic resampling: unbiased sampling to avoid sample impoverishment in particle filters," *Signal Processing*, vol. 92, no. 7, pp. 1637–1645, 2012.

[13] S. Park, J. P. Hwang, E. Kim, and K. Hyung-Jin, "A new evolutionary particle filter for the prevention of sample impoverishment," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 801–809, 2009.

[14] C. Xu, X. X. Wang, S. H. Duan, and J. W. Wan, "Particle filter tracking algorithm based on error ellipse resampling," *Chinese Journal of Scientific Instrument*, vol. 41, no. 12, pp. 76–84, 2020.

[15] X. Lin and J. Suo, "Particle filtering tracking algorithm based on adaptive particle swarm optimization," *Modern Electronics Technique*, vol. 43, no. 17, pp. 11–15, 2020.

[16] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[17] J. W. Wang, X. Li, H. Zhang, and H.-X. Ma, "Survey of nonlinear filters in the framework of recursive bayesian estimation," *Computer Science*, vol. 37, no. 8, pp. 21–25, 2010.

[18] J. Cornebise, É. Moulines, and J. Olsson, "Adaptive methods for sequential importance sampling with application to state space models," *Statistics and Computing*, vol. 18, no. 4, pp. 461–480, 2008.

[19] Y. B. Zhang, "Researches on maneuvering target modeling and tracking method," Master Thesis, University of Electronic Science and Technology of China, Chengdu, China, 2015.

[20] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[21] E. A. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 153–158, IEEE, Lake Louise, AB, Canada, October 2000.

[22] Y. G. Zhang, Y. L. Zhang, Z. M. Wu, and L. Ning, "A high order unscented kalman filtering method," *Acta Automatica Sinica*, vol. 40, no. 5, pp. 838–848, 2014.

[23] X. P. Huang and Y. Wang, *Kalman Filter Principle and Application: MATLAB Simulation*, Publishing House of Electronics Industry, Beijing, China, 2015.