A dual channel perturbation particle filter algorithm based on GPU acceleration

LI Fan^{1,*}, BI Hongkui², XIONG Jiajun², YU Chenlong¹, and LAN Xuhui²

1. Department of Graduate, Air Force Early-warning College, Wuhan 430019, China;

2. Department of Early Warning Intelligence, Air Force Early-warning College, Wuhan 430019, China

Abstract: The particle filter (PF) algorithm is one of the most commonly used algorithms for maneuvering target tracking. The traditional PF maps from multi-dimensional information to one-dimensional information during particle weight calculation, and the incorrect transmission of information leads to the fact that the particle prediction information does not match the weight information, and its essence is the reduction of the information entropy of the useful information. To solve this problem, a dual channel independent filtering method is proposed based on the idea of equalization mapping. Firstly, the particle prediction performance is described by particle manipulations of different dimensions, and the accuracy of particle prediction is improved. The improvement of particle degradation of this algorithm is analyzed in the aspects of particle weight and effective particle number. Secondly, according to the problem of lack of particle samples, the new particles are generated based on the filtering results, and the particle diversity is increased. Finally, the introduction of the graphics processing unit (GPU) parallel computing the platform, the “channel-level” and “particle-level” parallel computing the program are designed to accelerate the algorithm. The simulation results show that the algorithm has the advantages of better filtering precision, higher particle efficiency and faster calculation speed compared with the traditional algorithm of the CPU platform.

Keywords: particle filter (PF), dual channel filtering, graphic processing unit (GPU), parallel operation.

DOI: 10.21629/JSEE.2018.04.20

1. Introduction

The particle filter (PF) is an optimal regression Bayesian filtering algorithm based on Monte-Carlo simulation [1–3], which employs a set of sampled particles with correlation weights to approximate the system state distribution and outputs filtering results based on the particle weighted summation method. Compared with other filtering

algorithms such as Kalman and H_∞ , PF has low requirements on environment and system, and this algorithm also has good expansibility and universality. It is widely used in aerospace, control and electronic fields. The main problems of PF are particle weight degradation [4,5] and sample scarcity [6], so far, many scholars have been devoted to the research of particle weight degradation and put forward a variety of solutions. The first is resampling, by copying the weight particles and removing the small weight particles to mitigate the weight degradation of the particles. There are mainly classification samplings such as combined sampling, residue sampling and importance resampling (SIR) and other methods. However, the resampling can lead to the loss of particle diversity or even serious sample shortage problems. The second is to improve the importance of PF density function, Kalman filter (KF), extended Kalman filter (EKF) and unscented Kalman filter (UKF) [7–11]. Many methods are introduced into the distribution function of PF, such as Kalman particle filter (KPF) proposed by Xu et al. [12], extended particle filter (EPF) algorithm proposed by Zuo et al. [13], and unscented particle filter (UPF) algorithm proposed by Liu et al. [14]. These methods suppress the degradation of the weight of particles to a certain extent, but the tracking effect is not ideal when the target maneuvers because of the limitations of these algorithms. The lack of samples and the degradation of weights are a contradiction between PFs and difficult to be completely eliminated, the degradation of weight is manifested as the weight in the partial particle concentration. The lack of the sample shows that the particle state distribution is too concentrated. In general, the more serious the degradation of particle weight, the more serious the lack of diversity after resampling. The lack of diversity is the negative effect of solving the weight degradation. Therefore, the key to solve this dual problem is the design of the resampling method, the current emergence of double sampling [15], effective particle size and other methods [16].

Manuscript received March 30, 2017.

*Corresponding author.

This work was supported by the National High-tech R&D Program of China (2015AA7056045; 2015AA8017032P), and the National Natural Science Foundation of China (61401504).

In addition, the performance of the PF algorithm is closely related to the number of particles, although the large number of particles can get better filtering effect, it will lead to the problem of low computational efficiency and low real-time performance [17–19].

In this paper, the PF algorithm is used to track the near space hypersonic target, the difference of prediction accuracy of particles in different dimensions is analyzed, a dual-channel perturbation particle filter algorithm is proposed based on graphic processing unit (GPU) acceleration (GPU-DC-PF), and the improvement of particle degradation is proved from the aspects of particle weight and effective particle number. Then, filtering results perturbation is used to generate new particles to alleviate the problem of particle scarcity. Finally, the GPU computing platform is introduced, and the implementation of the parallel scheme acceleration algorithm is designed.

2. GPU-DC-PF improved method

Assuming that the motion equation and the measurement equation of the system are respectively:

$$\mathbf{X}(k+1) = \mathbf{F}(T(k))\mathbf{X}(k) + \mathbf{G}(k)\mathbf{u}(k) + \mathbf{V}(k) \quad (1)$$

$$\mathbf{Z}(k+1) = \mathbf{H}(k+1)\mathbf{X}(k+1) + \mathbf{W}(k+1) \quad (2)$$

where $\mathbf{X}(k)$ and $\mathbf{Z}(k)$ are the state vector and the measurement vector, respectively; $\mathbf{F}(k)$ and $\mathbf{H}(k)$ are the state transition matrix and the measurement matrix, respectively; $\mathbf{G}(k)$ and $\mathbf{u}(k)$ are the input control matrix and the control signal, respectively; $\mathbf{V}(k)$ and $\mathbf{W}(k)$ are zero mean white Gaussian process noise and measurement noise, and their covariance are $\mathbf{Q}(k)$ and $\mathbf{R}(k)$ respectively.

2.1 PF algorithm

PF is essentially a sample of random particles $\{\mathbf{X}^i(0:k), q^i(k)\}_{i=1}^{N_s}$ to approximate the statistical distribution of the state through the posterior probability density $p[\mathbf{X}(0:k)|\mathbf{Z}(1:k)]$, after prediction and filtering to output the final filtering result based on particle weighting. $\mathbf{X}(0:k)$ is the state vector from 0 to k , $\mathbf{Z}(0:k)$ is the measurement vector from 0 to k , $\mathbf{X}^i(0:k)$ is the particle vector from 0 to k , $q^i(k)$ is the i th particle weight, N_s is the number of particles. However, since it is difficult to sample directly from $p[\mathbf{X}(0:k)|\mathbf{Z}(0:k)]$, the importance probability density function $\pi[\mathbf{X}|\mathbf{Z}]$ is usually used to obtain samples. The standard PF algorithm based on sequential important resampling (SIS) is as follows [20].

Step 1 Particle sampling

Randomly select N_s samples from the importance density function $\pi[\mathbf{X}|\mathbf{Z}]$.

Step 2 Calculate the sample weights

$$q^i(k) = \frac{p[\mathbf{Z}(k)|\mathbf{X}^i(k)]p[\mathbf{X}^i(k)|\mathbf{X}^i(k-1)]}{\pi[\mathbf{X}^i(k)|\mathbf{X}^i(k-1), \mathbf{Z}(k)]}. \quad (3)$$

The weights are normalized as

$$q^i(k) = \frac{q^i(k)}{\sum_{j=1}^{N_s} q^j(k)}. \quad (4)$$

Step 3 Resampling

Preserve large weights particles and remove small weight particles. Then the posterior probability density distribution is approximately:

$$p[\mathbf{X}(0:k)|\mathbf{Z}(1:k)] \approx \sum_{i=1}^{N_s} q^i(k) \delta(\mathbf{X}(k) - \mathbf{X}^i(k)). \quad (5)$$

Step 4 State and covariance estimation

$$\widehat{\mathbf{X}}(k) = \sum_{i=1}^{N_s} q^i(k) \mathbf{X}^i(k) \quad (6)$$

$$\mathbf{P}(k) = \sum_{i=1}^{N_s} q_k^i (\mathbf{X}^i(k) - \widehat{\mathbf{X}}(k)) (\mathbf{X}^i(k) - \widehat{\mathbf{X}}(k))^T. \quad (7)$$

Conventional maneuvering target tracking is mostly performed in two-dimensional plane or three-dimensional space. The traditional PF maps from multi-dimensional information to one-dimensional information in Step 2 performs weight calculation, which leads to particle prediction information and weight mismatch. The physical nature of this problem is that the information transmission is incorrect and the information entropy is weakened, where the information entropy is the amount of information contained in the source. Sampling to get the current state of the forecast in Step 1, because the prediction errors of different particles in different dimensions are different, the information entropy of each particle is also different, the information difference between the particles in the dimension is neglected in the process of Step 1 to Step 2, which leads to the decrease of the information entropy of the particles in Step 2. In view of this problem, this paper adopts the method of information equalization mapping to solve the entropy reduction problem of useful information in Step 1 to Step 2.

2.2 Particle prediction accuracy analysis

Assume that the target is a two-dimensional trajectory. The realization of PF weights can be calculated by likelihood function.

$$\begin{aligned}
\Lambda_j(k) = & \frac{1}{\sqrt{2\pi\mathbf{S}_j(k)}} \exp\left(-\frac{1}{2}(\mathbf{v}_j(k))^T(\mathbf{S}_j(k))^{-1}\mathbf{v}_j(k)\right) = \\
& \frac{1}{\sqrt{2\pi\mathbf{S}_j^x(k)\mathbf{S}_j^y(k)}} \exp\left(-\frac{1}{2}\left(\frac{(\mathbf{v}_j^x(k))^2}{\mathbf{S}_j^x(k)} + \frac{(\mathbf{v}_j^y(k))^2}{\mathbf{S}_j^y(k)}\right)\right)
\end{aligned} \quad (8)$$

where $\mathbf{v}_j(k) = \begin{bmatrix} \mathbf{v}_j^x(k) \\ \mathbf{v}_j^y(k) \end{bmatrix}$ is the residuals of the j th particles in different dimensions at time k , $\mathbf{v}_j^x(k), \mathbf{v}_j^y(k) \in \mathbf{R}^1$; $\mathbf{S}_j(k) = \begin{bmatrix} \mathbf{S}_j^x(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_j^y(k) \end{bmatrix}$ is the corresponding covariance, $\mathbf{S}_j^x(k), \mathbf{S}_j^y(k) \in \mathbf{R}^1$. Let $x_v = (\mathbf{v}_j^x(k))^2$, $y_v = (\mathbf{v}_j^y(k))^2$, the likelihood function based on the residual distribution of residuals of different dimensions is shown in Fig. 1.

Fig. 1 Likelihood functions based on residual joint distribution

Fig. 1 shows the likelihood functions based on residual joint distribution, it is a curved surface, when the likelihood value $\Lambda_j(k)$ is determined, the curved surface is crossed with a plane parallel to xoy , and the resulting tangent projected on the xoy plane is shown in Fig. 2. There are numerous residual solutions satisfying the current likelihood. Obviously, the residuals differences of the particles in different dimensions are assimilated. In the traditional PF algorithm, the residual composition in the neighborhood of point B defaults to the current residual distribution in Fig. 2, but the difference in the prediction accuracy of the particle in different dimensions is ignored in this algorithm. However, in the actual target tracking process, the target is in different dimensions of the movement speed, mobility cannot be completely consistent, that is, the residual distribution cannot be located in the B point neighborhood. There may be A , C and D points in Fig. 2. Although the A , C and D points in the x and y dimension residual distribution differences are obvious, it is difficult to indicate this difference based on the joint distribution residual.

Fig. 2 Residual distribution of likelihood function

Fig. 3 shows mapping relationship between the particle residual, likelihood function and particle weight, assuming that the mapping of the residual to the likelihood function is $f_{xy} : M_{xy} \rightarrow N_{xy}$, where $M_{xy} = \{(\mathbf{v}_j^x(k))^2, (\mathbf{v}_j^y(k))^2\}$, $N_{xy} = \{\Lambda_j(k)\}$, and there are $\forall m_1, m_2 \in M_{xy}, m_1 \neq m_2, \exists f_{xy}(m_1) \neq f_{xy}(m_2)$, that is, the mapping of f_{xy} is not unique. Similarly, f_{xy}^{-1} is not unique. It can be seen that there is no one-to-one correspondence between the joint distribution residual, the likelihood functions and the particle weight in the traditional PF algorithm.

Fig. 3 PF mapping relationship between residual, likelihood function and particle weight

In the traditional PF algorithm, the weight of the particles is obtained based on the joint distribution residual, in fact, the accuracy of particle prediction for different dimensions has been compromised. If the accuracy of the particle j in different dimensions is similar at a certain time, the weight can indicate the real prediction accuracy of the particle j , but in most cases, the prediction accuracy of the majority of the particles in different dimensions is quite different.

Considering this problem, the GPU-DC-PF uses the edge distribution residual to describe the likelihood function, so that the algorithm can achieve a one-to-one mapping relationship between the residual, likelihood function and particle weight. The likelihood function of particle j in x -dimension is

$$\begin{aligned}
\Lambda_j^x(k) = & \frac{1}{\sqrt{2\pi\mathbf{S}_j^x(k)}} \exp\left(-\frac{1}{2}(\mathbf{v}_j^x(k))^T(\mathbf{S}_j^x(k))^{-1}\mathbf{v}_j^x(k)\right) = \\
& \frac{1}{\sqrt{2\pi\mathbf{S}_j^x(k)}} \exp\left(-\frac{1}{2}\frac{(\mathbf{v}_j^x(k))^2}{\mathbf{S}_j^x(k)}\right).
\end{aligned} \quad (9)$$

$$\mathbf{v}_j^x(k) = \mathbf{Z}^x(k) - \mathbf{H}_j^x(k) \widehat{\mathbf{X}}_j^x(k|k-1) \quad (10)$$

$$\mathbf{S}_j^x(k) = \mathbf{H}_j^x(k) \mathbf{P}_j^x(k|k-1) (\mathbf{H}_j^x(k))^T + \mathbf{R}^x(k) \quad (11)$$

where $\mathbf{Z}^x$ is the measurement component, $\mathbf{R}^x$ is the measured noise variance component, $\mathbf{H}_j^x$ is the x -dimensional measurement matrix, and $\mathbf{P}_j^x(k|k-1)$ is the covariance input.

According to (9), the shape of x -dimensional likelihood function $\Lambda_j^x(k)$ is a curve. When the likelihood $\Lambda_j^x(k)$ is determined, the distribution characteristic of the particle j in the x -dimensional prediction error can be accurately represented.

Fig. 4 shows the GPU-DC-PF mapping relationship between residual, likelihood function and particle weight. Let the mapping relationship of the residual to the likelihood function be $f_x : M_x \rightarrow N_x$, where $M_x = \{(\mathbf{v}_j^x(k))^2\}$, $N_x = \{\Lambda_j^x(k)\}$, $\forall n \in N_x, \exists m \in M_x$, s.t. $n = f_x(m)$, that is, f_x is a subjective, and any residual has a likelihood value corresponding to it; $\forall m_1, m_2 \in M_x, m_1 \neq m_2$, there is $f_x(m_1) \neq f_x(m_2)$, f_x is a single shot, and any residual has only one likelihood corresponding to it.

Fig. 4 GPU-DC-PF mapping relationship between residual, likelihood function and particle weight

Assume that the mapping relationship of the residual to the likelihood function is $g_i : R(f_x) \rightarrow M_x$, $\forall n \in R(f_x), \exists m_1, m_2 \in N_x$, s.t. $g_i(n) = m_i$ ($i = 1, 2$), from f_x for a single shot that $m_1 = m_2$, f_x^{-1} is unique, and any likelihood value exists with a residual expected to correspond to it; $\forall n \in R(f_x)$, if $f_x^{-1}(n_1) = f_x^{-1}(n_2)$, where $f_x(f_x^{-1}(n_1)) = f_x(f_x^{-1}(n_2))$. As defined by f_x^{-1} , we know that $n_1 = n_2$, that is, f_x^{-1} is a single shot, any likelihood has and only one residual corresponding to it.

There is one-to-one correspondence between the residuals, the likelihood functions and the particle weights in the GPU-DC-PF algorithm.

2.3 Particle degradation analysis

Assuming that the target observation state is $\mathbf{Z}(k) = [\mathbf{Z}^x(k), \mathbf{Z}^y(k)]^T$ at the time k , in the traditional PF, the weights of the two-dimensional particles are solved based on the current measurement information. Only a few of particles of considerable weight are involved in the final filtering calculation, and the efficiency is low. The essence of the PF algorithm is to select the current measurement of $\mathbf{Z}(k) = [\mathbf{Z}^x(k), \mathbf{Z}^y(k)]^T$ neighborhood particles as the effective particle set U_{xy} , and through the effective particle set to get the filtering output, however, in the actual

tracking process, the state of different dimensions is independent, and the prediction accuracy of any particle in different dimensions is different. The advantages of GPU-DC-PF are analyzed in the aspects of particle weight and effective particle number.

2.3.1 Particle weight analysis

For convenience, let the covariance of residual at time k $\mathbf{S}_j^x(k) = \mathbf{S}_j^y(k)$ be constant a , then there is

$$\Lambda_j(k) = \frac{1}{\sqrt{2\pi a^2}} \exp \left(-\frac{1}{2} \left(\frac{(\mathbf{v}_j^x(k))^2 + (\mathbf{v}_j^y(k))^2}{2a} \right) \right) \quad (12)$$

$$\Lambda_j^x(k) = \frac{1}{\sqrt{2\pi a}} \exp \left(-\frac{1}{2} \frac{(\mathbf{v}_j^x(k))^2}{a} \right) \quad (13)$$

Fig. 5(a) is a three-dimensional comparison of weights of the joint distribution and the edge distribution, where the edge distribution only considers x -dimensional measurement information. Fig. 5(b) is the plane projection of Fig. 5(a). The weight of the edge distribution is always greater than the weight of the joint distribution, when the difference of $\mathbf{S}_j^x(k)$ and $\mathbf{S}_j^y(k)$ increases, the GPU-DC-PF has higher discrimination in particle size prediction in different dimensions, and can effectively suppress weight degradation.

(a) Comparison of 3D weights

(b) Comparison of 2D weights

Fig. 5 Comparison of weights of different distributions

2.3.2 Analysis of effective particles numbers

Point A in Fig. 6 has a very high approximation in the x -dimension, but low in y -dimension. This difference leads to $A \notin U_{xy}$, where U_{xy} is the effective particle set of x and y dimensions. In this paper, a method to approximate the real state of particles based on different dimensions is proposed, the effective particle sets U_x and U_y are selected based on different dimensions, and the different dimension filter output is calculated according to different sets.

Fig. 6 Particle distribution diagram based on real state

To sum up, $U_x \supseteq U_{xy}$, $U_y \supseteq U_{xy}$, the distribution of effective particles in the GPU-DC-PF is larger, the number of effective particles is larger, and the survivability of the particles is higher compared to the traditional PF.

2.4 Information entropy analysis

According to the definition of information entropy, we can get the information entropy of x -dimensional residual and likelihood function respectively:

$$\begin{cases} h(\mathbf{v}^x(k)) = -\sum_{i=1}^{N_s} p(\mathbf{v}_i^j(k)) \log_2(\mathbf{v}_i^j(k)) \\ h(\Lambda^x(k)) = -\sum_{i=1}^{N_s} p(\Lambda_i^j(k)) \log_2(\Lambda_i^j(k)) \end{cases} \quad (14)$$

where $p(\cdot)$ is the probability. The PF algorithm due to different dimensions of the residual coupling has $h(\mathbf{v}^x(k)) + h(\mathbf{v}^y(k)) > h(\Lambda(k))$, that is the lost part of the information in the transmission process, and it leads to a reduction in useful information. There is $h(\mathbf{v}^x(k)) = h(\Lambda^x(k))$ in GPU-DC-PF, the useful information in the delivery process is not eliminated and it ensures the correctness of the information transmission. In addition, $h(\mathbf{v}^x(k))$ and $h(\Lambda^y(k))$ are independent of each other, the information of different dimensions does not have coupling, and this feature is

consistent with the rules of parallel computing. Similarly, other dimensions of GPU-DC-PF have the same nature.

2.5 Particle perturbation update

The lack of sample refers to the resampling after a few large weight particles are repeatedly copied, and this process will lead to a significant reduction in the number of different particles. In response to this problem, each time after filtering, the filter results are superimposed by randomized perturbation [21], which improves the particle diversity and alleviates the problem of particle depletion [22–24].

From the structure point of view, GPU-DC-PF does not change the main structure of the PF algorithm, the only difference is that the PF algorithm is carried out in different directions, PF only needs to calculate a likelihood function, a weight for a particle. In the GPU-DC-IMM, a particle needs to calculate the likelihood function and the weights separately in different dimensions. In addition, GPU-DC-PF based on the results of the perturbation of new particles generates particles to alleviate the problem of lack of particle samples.

3. GPU-based dual-channel PF parallel computing design

The strong law of large numbers shows that the larger the number of particles, the better the accuracy and convergence performance of the filter, but the larger number of particle operations will have larger computational redundancy, which leads to the real-time problem that limits the application of the PF algorithm. In view of this problem, on the one hand, we can optimize the filter design, balance the contradiction of estimation accuracy and calculation redundancy; on the other hand, based on the current hardware level, open up a new computing platform to optimize algorithm implementation.

With the major breakthroughs in floating point operations, GPU-based parallel computing has become a feasible solution to real-time problems, in particular, since the emergence of the compute unified device architecture (CUDA) [25–27], GPU-based parallel computing has become one of the mainstream in aerospace, data mining, supercomputers and other high-performance common computing. The NVIDIA released a feasibility study based on GPU acceleration to enable the phased array radar missile tracking system to achieve 10 000 times performance improvements at NVIDIA's GTC conference in 2016. At this point, GPU-based parallel computing has become a phased array radar to achieve high real-time weathervane. This paper combines the characteristics of GPU multi-core multi-thread to plan GPU-DC-PF algorithm parallel computing

tasks [28–30], a “channel-level” and “particle-level” parallel computing scheme is designed [25], while the GPU-DC-PF “channel level” parallel corresponds to the GPU thread block parallel, and the “particle level” parallel corresponds to the thread parallel. The specific programs are shown in Fig. 7 and Fig. 8.

Fig. 7 “Channel-level” parallelism

Fig. 8 “Particle-level” parallelism

3.1 “Channel level” in parallel

The operation of different channels corresponds to different dimension filtering. The data are calculated independently in different channels, respectively, to complete the particle prediction, weight calculation, resampling, perturbation to generate new particle, and achieve different dimensions of the filter calculation.

3.2 “Particle level” in parallel

Each particle corresponds to a thread in the “particle-level” parallel program, the total time consuming of the algorithm

is the time consuming for a single particle with the longest computation time. The detailed steps in the GPU-DC-PF are as follows.

Step 1 Particle parallelism prediction

Build the kernel threads with the same number of particles to complete the particle prediction, residual likelihood calculation, independently updating the state of each particle to achieve maximum parallelization.

Step 2 Weight calculation and normalization

Based on the residual likelihood of each particle, the weight of the particle is solved and normalized (serial).

Step 3 Particle serial resampling

Particle resampling is done by serial calculation when the correlation between particles is unavoidable for all particles resampling.

Step 4 Filter estimation

All the resampled particles are weighted summed, build the same number of kernel threads as the number of particles where each thread completes a particle state weighting to maximize the advantages of GPU parallel computing.

Step 5 Perturbs to generate new particles

With the target filter state as the mean, the state error covariance as the variance perturbation generates new particles instead of the multiple copies of the particles. The filter results are output to the CPU. The filter is completed at this moment and then the next moment cycle.

3.3 Time complexity analysis

If the traditional serial computing method is adopted, the time complexity of the particle filter algorithm is a function of the number of particles N_s , the time complexity is the linear order of the number of particles $O(N_s)$. In the weight calculation and normalization stage, according to (3) and (4), the time complexity is $O(N_s)$. In the resampling stage, using sequential importance sampling (SIS) for resampling, PF time complexity for the square order is about $O(N_s^2)$. In the state and covariance estimation phase, the complexity is still $O(N_s)$. GPU-DC-PF is divided into different dimensions at the above stage. The GPU-DC-PF calculation is twice the PF in each stage. However, the superposition of time complexity of the same order does not affect the result, therefore, the GPU-DC-PF is the same as the PF time complexity in the above stages. In addition, GPU-DC-PF also generates new particles by perturbation. At this stage, it is first necessary to detect and remove duplicate particles, similar to bubble sequencing, and its time complexity is $\frac{N_s(N_s - 1)}{2}$. And then produce new particles, the time complexity is $\max[O(N_s - N_h^x), O(N_s - N_h^y)]$, where N_h^x and N_h^y are the number of particles remaining after the particles are deleted in different dimensions. Since N_h^x is not constant

to N_h^y , its complexity is determined by the party that produces the majority of the particles.

While GPU-DC-PF using GPU to achieve parallel computing in the actual process, combined with GPU-DC-PF implementation steps, the time complexity is constant order $O(1)$ in the parallel computing part, and the time complexity in the serial calculation section does not change.

4. Simulation analysis

Assuming that the target is released at a high altitude of 15 km, jumping cruises at 60 to 100 km altitude, the full ballistic motion pattern is complex and the maneuverability is strong. The state transition matrix $F(T) =$

$$\begin{bmatrix} \mathbf{f} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{f} \end{bmatrix}, \text{ and } \mathbf{f} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \text{ the control}$$

matrix $\mathbf{F}(k) = \begin{bmatrix} 0 & 0 & 0 & T^2/2 & T & 1 \\ T^2/2 & T & 1 & 0 & 0 & 0 \end{bmatrix}^T$, observed noise $w(k)$ and process noise $n(k)$ are Gaussian random noise for N (unit/m) distribution, the sampling time interval is 1 s, with the simulation platform operating system for Windows 7, CPU model for the dual-core Intel (R) Core (TM) i7-3770 3.40 Hz, GPU model for the NVIDIA GeForce GTX 650, programming environment for Microsoft Visual Studio 2008. Compared with SIR, adapting particle filter (APF) [31] and sequential quadratic programming-particle filter (SQP-PF) algorithm [32], the number of particles is 1 000, and 100 Monte Carlo simulation results are shown in Fig. 9, where only the SQP-PF is compared with the GPU-DC-PF due to the limitation of the picture.

In order to illustrate the advantages of the GPU-DC-PF algorithm compared with SIR, APF and SQP-PF, the particle weight distribution of 400–420 s in different directions is compared and analyzed.

(a) Comparison of real trajectory and filtering trajectory

(b) Root mean square error of x dimension position error

(c) Root mean square error of y dimension position error

Fig. 9 Comparison of filtering results

In order to verify the advantages of GPU-DC-PF in particle efficiency and computational efficiency, the filtering effect of the four algorithms is shown in Table 1 by changing the number of particles. Set the following parameters to evaluate the performance of the algorithm, overall root mean square error (ORMSE) is the total variance (unit/m), N_e is the number of effective particles, f is the number of filtering failures, T is the number of steps to track, and T_c is the time consuming (unit/s) of the algorithm.

$$\text{ORMSE} = \sqrt{\frac{1}{M} \sum_{i=1}^M (\hat{X}(i) - X(i))^2} \quad (15)$$

$$N_e = \sum_{i=1}^M \frac{N_{eff-i}}{M} \quad (16)$$

$$N_{eff-i} = \frac{\sum_{k=1}^T \text{ceil} \left[1 / \sum_{j=1}^{N_s} (q_i^j(k))^2 \right]}{T} \quad (17)$$

where M and N_{eff-i} are the Monte Carlo times of successful filtering, the number of effective particles in the i th

Monte Carlo simulation experiment, $q_i^j(k)$ is the weight of the normalized particle j at the time k in the i th experiment, $\text{ceil}[\cdot]$ is the rounding function, and q_k^j is the weight of the normalized particle j at time k .

Table 1 Comparison of algorithm performance

Number of particles		1 000	500	100
Parameter	Algorithm			
$X\text{-ORMSE}$	SIR	346.53	393.45	—
	APF	327.45	374.81	—
	SQP-PF	309.21	352.26	—
	GPU-DC-PF	263.64	321.63	357.21
$Y\text{-ORMSE}$	SIR	351.64	402.33	—
	APF	336.10	389.02	—
	SQP-PF	317.07	365.31	—
	GPU-DC-PF	283.51	335.21	361.84
N_e	SIR	189	84	—
	APF	257	112	—
	SQP-PF	367	137	—
	GPU-DC-PF	583	257	68
f	SIR	39	69	100
	APF	23	54	100
	SQP-PF	12	37	100
	GPU-DC-PF	0	0	23
T_c	SIR	50.814	27.364	—
	APF	58.312	29.507	—
	SQP-PF	60.089	30.012	—
	GPU-DC-PF	12.731	8.542	2.688

Compared with the above filter results, the GPU-DC-PF algorithm has the following advantages.

(i) The tracking accuracy improves. It can be seen from Fig. 9 and Table 1 that the GPU-DC-PF algorithm has a relatively high accuracy when the number of particles is the same. GPU-DC-PF through the dual-channel particle distribution filter avoids the particles to predict the accuracy of assimilation in different directions, and more accurate characterization of different directions of different patterns of movement, especially when the difference in maneuvering strength is large in different directions, the traditional PF algorithm shows multiple error spikes, GPU-DC-PF can maintain a more stable tracking accuracy.

(ii) Effectively mitigate the degradation of particle weights. It can be seen from Fig. 10 that the particle weight distribution in different directions of the SQP-PF algorithm is difficult to accurately indicate the accuracy of particle prediction, that is, as shown in Fig. 10(a) and Fig. 10(c), there is a problem that the particle has a higher prediction accuracy in the y direction but lower in the x direction. The essential reason is that the prediction accuracy of the particle in the y direction affects the prediction accuracy in the other direction. The particles must simultaneously achieve higher prediction accuracy in different directions in order to obtain a larger weight in SQP-PF, but the actual tracking process in different directions of movement independent, this feature greatly reduces the number of effective particles, the particle weights are calculated through

different channels to maximize the prediction advantage of particles in either direction in GPU-DC-PF, Fig. 10(b) and Fig. 10(d). Table 1 shows that GPU-DC-PF has a more uniform particle weight distribution, more effective particles, better adaptability and fewer filter divergence times than SIR, APF and SQP-PF.

(iii) Real-time enhancement. Speed up the algorithm through the GPU to achieve rapid operation, reducing the filter time consuming, and greatly expanding the particle filter in the field of high real-time filtering applications.

(d) Weight distribution of GPU-DC-PF in y direction

Fig. 10 Normalized particle weight distribution comparison

(iv) To a certain extent inhibit the lack of particles. As can be seen from Table 1, the GPU-DC-PF algorithm requires the smallest number of particles, and the generation of new particles is based on the results of filtering results in the expansion of particle diversity, which alleviates the problem of lack of particle diversity due to repeated replication of particles during resampling.

5. Conclusions

The GPU-DC-PF algorithm is proposed for the difference of the effectiveness of particles in different dimensions. Firstly, through the selection of effective particles in different dimensions, the problem of the degradation of traditional PF weights is alleviated, and the prediction advantage of particles in any dimension is brought into full play. Secondly, the new particles enhance the particle diversity based on the perturbation of the filtering results. Finally, in order to improve the real-time performance of the algorithm, a parallel scheme is designed according to the characteristics of GPU parallel operation, the simulation results show that GPU-DC-PF has improved performance in tracking accuracy, effective particle number, particle diversity and real-time performance.

References

- [1] CAPPE O, GODSILL S J, MOULINES E. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 2007, 95(5): 899–924.
- [2] DOUCET A, JOHANSEN A M. A tutorial on particle filtering and smoothing: 15 years later. *Handbook of Nonlinear Filtering*, 2011: 1–39.
- [3] GUSTAFSSON F. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 2010, 25(7): 53–81.
- [4] LI T C, BOLIC M, DJURIC P M. Resampling methods for particle filtering: classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 2015, 32(3): 70–86.
- [5] LI T C, VILLARRUBIA G, SUN S D, et al. Resampling methods for particle filtering: identical distribution, a new method, and comparable study. *Frontiers of Information Technology and Electronic Engineering*, 2015, 16(11): 969–984.
- [6] LI T C, SUN S D, SATTAR T P, et al. Fight sample degeneracy and impoverishment in particle filters: a review of intelligent approaches. *Expert Systems with Applications*, 2014, 41(8): 3944–3954.
- [7] LI X R, JILKOV V P. A survey of maneuvering target tracking, part VIc: approximate nonlinear density filtering in discrete time. *Proc. of the SPIE Defense, Security, and Sensing*, 2012: 1–12.
- [8] SARKKA S. On unscented Kalman filtering for state estimation of continuous-time nonlinear systems. *IEEE Trans. on Automatic Control*, 2007, 52(9): 1631–1641.
- [9] BARRETT L, HENZI S P. Particle filters for positioning, navigation and tracking. *IEEE Trans. on Signal Processing*, 2002, 50(2): 425–437.
- [10] ARASARATNAM I, HAYKIN S. Cubature Kalman filters. *IEEE Trans. on Automatic Control*, 2009, 54(6): 1254–1269.
- [11] ZHANG W J, WANG S Y, FENG Y L, et al. Huber-based high-degree cubature Kalman tracking algorithm. *Acta Physica Sinica*, 2016, 65(8): 088401. (in Chinese)
- [12] XU C, GAO M, YANG Y. Self-turning hierarchical Kalman-particle for efficient target tracking. *Infrared and Laser Engineering*, 2015, 44(6): 1942–1949.
- [13] ZUO J Y, JIA Y N. Particle filter guided by iterated extended Kalman filter. *Proc. of the 13th Automation and Systems International Conference on Control*, 2013: 1605–1609.
- [14] LIU C Y, SHUI P L, LI S. Unscented extended Kalman filter for target tracking. *Journal of Systems Engineering and Electronics*, 2011, 22(2): 188–192.
- [15] LI T C, SUN S D. Double-resampling based Monte Carlo localization for mobile robot. *Acta Automatma Sinica*, 2010, 36(9): 1279–1286. (in Chinese)
- [16] LI T C, FAN H Q, SUN S D. Particle filtering: theory, approach, and application for multi-target tracking. *Acta Automatma Sinica*, 2015, 41(12): 1981–2002.
- [17] JILKOV V P, WU J D. Efficient GPU-accelerated implementation of particle and particle flow filters for target tracking. *Journal of Advances in Information Fusion*, 2015, 10(1): 73–88.
- [18] BROWN J A, CAPSON D W. A framework for 3D model-based visual tracking using a GPU-accelerated particle filter. *IEEE Trans. on Visualization & Computer Graphics*, 2011, 18(1): 68–80.
- [19] BASHI A S, JILKOV V P, LI X R, et al. Distributed implementations of particle filters. *Proc. of the International Conference on Information Fusion*, 2003: 1164–1171.
- [20] DEL M P, DOUCET A. Particle methods: an introduction with applications. *Esaim: Proceedings*, 2014, 44: 1–46.
- [21] ZHANG Q, QIAO Y K, KONG X Y, et al. Study on stochastic perturbation strong tracking particle filter. *Acta Physica Sinica*, 2014, 63(11): 110505. (in Chinese)
- [22] BOLIC M, DJURIC P M, HONG S. Resampling algorithms and architectures for distributed particle filters. *IEEE Trans. on Signal Processing*, 2005, 53(7): 2442–2450.
- [23] STEINBRING J, HANEBECK U D. GPU-accelerated progressive Gaussian filtering with applications to extended object tracking. *Proc. of the IEEE International Conference on Information Fusion*, 2015: 1038–1045.
- [24] TAGER S. GPU-specific Kalman filtering and retrodiction for large-scale target tracking. *Electrical & Computer Engineering*, 2013: 1–113.
- [25] HASHMI M F, PAL R, SAXENA R, et al. A new approach for real time object detection and tracking on high resolution and multi-camera surveillance videos using GPU. *Journal of Central South University*, 2016, 23(1): 130–144.

- [26] GOYAL B, BUDHRAJA T, BHATNAGAR R, et al. Implementation of particle filters for single target tracking using CUDA. Proc. of the 5th IEEE International Conference on Advances in Computing and Communications, 2016: 28–32.
- [27] KIM S, CHO J, PARK D. Moving-target position estimation using GPU-based particle filter for IOT sensing applications. Applied Sciences, 2017, 7(11): 1–14.
- [28] LIU K, WEI S, CHEN Z, et al. A real-time high performance computation architecture for multiple moving target tracking based on wide-area motion imagery via cloud and graphic processing units. Sensors, 2017, 17(2): 1–16.
- [29] TAN F, XIAO S, LI L. Enhanced CMT tracking algorithm with CUDA acceleration. Proc. of the IEEE International Conference on Multimedia & Expo Workshops, 2017: 441–446.
- [30] COOK S. CUDA programming: a developer's guide to parallel computing with GPUs. San Francisco: Morgan Kaufmann, 2012.
- [31] ZOU W J, GONG X, BO Y M. Adaptive layered-sampling auxiliary particle filter's research and application in video tracking. Acta Photonica Sinica, 2010, 39(3): 571–576. (in Chinese)
- [32] LI F, BI H K, DUAN M, et al. Particle filter algorithm based on sequential quadratic programming. Modern Radar, 2016, 38(9): 50–56. (in Chinese)

Biographies

LI Fan was born in 1992. He received his B.S degree and M.S degree from the Air Force Early-warning College in 2010 and 2016, respectively. Now, he is a Ph.D. candidate and a lecturer in Department of Early Warning Intelligence at the Air Force Early-warning College. His research interest is the near space hypersonic vehicle early warning detection technology.
E-mail: 1746338543@qq.com

BI Hongkui was born in 1964. She received her B.S and M.S degrees from Huazhong University of Science and Technology in 1986 and 1989, respectively. Now, she is a professor in Air Force Early-warning College. Her research interests include radar data processing, radar equipment technology and application.
E-mail: bhk001@126.com

XIONG Jiajun was born in 1961. He received his B.S. degree in computer software from Huazhong University of Science and Technology in 1983, M.S degree from National University of Defense Technology in 1988, and Ph.D degree from Huazhong University of Science and Technology in 2004. Now, he is a professor in Air Force Early-warning College. His research interests include data fusion and early warning intelligence analysis.
E-mail: 13871163420@139.com

YU Chenlong was born in 1989. He received his B.S. degree from South China University of Technology in 2012 and M.S. degree from Air Force Early-warning College in 2015. Now he is a Ph.D. candidate in Air Force Early-warning College. His research interests include signal processing, target tracking and target detection.
E-mail: 342583844@qq.com

LAN Xuhui was born in 1976. He received his B.S degree, M.S degree and Ph.D. degree from Air Force Early-warning College in 1999, 2002, and 2012 respectively. Now he is an associate professor in Air Force Early-warning College. His research directions are multi-sensor data fusion and integrated electronic information system.
E-mail: Lansoft007@sohu.com