2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

MLPF Algorithm for Tracking Fast Moving Target against Light Interference

Libo Zhang*, Yuanqiang Cai*, Zakir Ullah* and Tiejian Luo*
*University of Chinese Academy of Sciences,
Beijing, China

***Abstract*—In order to deal with the difficulty of tracking the fast moving aerial targets with light interference, we propose an improved particle tracking algorithm named multi-layers particle filter (MLPF). In MLPF, the particles are divided into three categories: the main particles (M-particles), the subordinate particles (S-particles) and the regenerate particles (R-particles). In the phase of resampling and state estimating, only M-particles are involved, then the R-particles are generated and considered as new S-particles in the next cycle. To a certain extent, our algorithm maintains the diversity of particles and reduces the computation time. Besides, MLPF has significant improvements on overcoming the tracing error after the sudden disappearance of the target and solving the degradation of particles. We demonstrate effectiveness of our proposed algorithm through systematic experiments. Experimental results show MLPF has better tracking effect compared to the traditional particle filter (PF) when the target is moving fast and affected by light interference. In the first experiment, the running time has been reduced from 47s to 21s while the precision increased from 64% to 96%. And for the second experiment, the running time has been reduced from 237s to 121s while precision increased from 46% to 89%.**

I. Introduction

Target tracking has been widely used in both civil and military fields and a number of target tracking algorithms have been proposed [1]. The particle filter is one of the most popular algorithms because both nonlinear and non-Gaussian problems can be solved by it very well [2]. The particle filter is a Monte Carlo estimation algorithm in which a posteriori probability density function is constructed by using weighted particles to make it suitable for the state estimation of nonlinear and non-Gaussian systems [3]. With the nonlinear target dynamics, the nonlinear measurements and the non-Gaussian noise, the computationally intensive particle filter [4] has become a common choice which means constructing the posterior probability density function via a number of weighted particles. However, by using the traditional particle filter, it can't handle fast moving target with light interference, because the particle degradation occurs in the resampling phase, and causes the loss of the diversity of particle, which is not desirable. Other limitations of using traditional particle filter are its high computational cost and the memory requirements. Therefore, it's very hard to obtain real-time feasible implementations for these existing methods [5], [6].

In order to solve the problems motioned above, some improved methods have been proposed such as Stuck [7], CT [8] and SCM [9]. Stuck uses a kernelized structured output support vector machine (SVM), which could be learned online to provide adaptive tracking. And this method introduces a budgeting mechanism which would occur during tracking. CT employs the non-adaptive random projections that preserve the structure of the image feature of objects. A very sparse measurement matrix is adopted to efficiently extract the features for the appearance model. And SCM proposes a robust appearance model that exploits both holistic templates and local representations. This method develops a sparsity-based discriminative classifier (SD-C) and a sparsity-based generative model (SGM). But the above methods would cause large computational cost, which means the long running time while processing. Minimum Output Sum of Squared Error (MOSSE) [10] presents a new type of correlation filter, a MOSSE filter, which produces stable correlation filters using a single frame when initialized. Although it greatly improves the processing speed, it has low precision. SAMF [11] uses a very appealing tracker based on the correlation filter framework, which improves the precision but decreases the processing speed heavily. MOSSE and SAMF both solve the real-time problem but the precision is still not very good. Above all, these methods cannot deal with the fast moving objects very well with light interference condition especially in aerial target tracking.

In this article, an improved particle filter algorithm is proposed. We set three kinds of particles for keeping the diversity of particle. Our algorithm (MLPF) chooses the S-particles around the M-particle to optimize M-particle. And only the optimized principle particles participate in resampling. In next cycle, we randomly generate the equal number of R-particles to replace the previous S-particles. This approach improves the diversity of particles and solves a variety of problems associated with it. It shows better results when the target has morphological changes and temporary disappearance.

II. Signal Processing Domain Knowledge

In order to better understand our algorithm, we summarized the background knowledge of MLPF as follows.

A. Bayesian Filter

Subsection text here. System state estimation means getting the state equation of the system through the observation equation. Assuming the state equation and observation equation of the system are as following:

$$x_k = f_k\left(x_{k-1}, v_{k-1}\right) \tag{1}$$

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

$$y_k = h_k\left(x_k, n_k\right) \tag{2}$$

x is the real state of the system, and y is the observation state of the system. f and h represent the state transfer function and the observation function of the system respectively. v and h represent the process noise and measurement noise of the system respectively.

In Bayesian estimation, the system state estimation is calculating the credibility $p(x_k|y_{1:k})$ of the current state x_k based on the observed state $y_{1:k}$. Specific calculations include two steps, prediction and update. The prediction process is based on the prior acquired knowledge to predict the state, and the prior probability density of the state is calculated by the state equation (1) marked as $p(x_k|x_{k-1})$. The updating process is modifying the prior probability density by the latest observation to obtain a posteriori density. In this process, the system state is assumed to obey the first order of Markoff model, that means the current state of the system $x(k)$ is only related to the status of its previous state $x(k-1)$. At the same time, it is assumed that the observed $y(k)$ is only associated with the current state of the system $x(k)$.

Assuming the probability density function of $k-1$ moment is $p(x_{k-1}|y_{k-1})$, according to the probability density of the last moment $p(x_{k-1}|y_{1:k-1})$, we can get $p(x_k|y_{1:k-1})$:

$$\begin{aligned} &p(x_{k-1}|y_{1:k-1}) \\ &= \int p(x_k, x_{k-1}|y_{1:k-1})dx_{k-1} \\ &= \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \end{aligned} \tag{3}$$

Process updating: The posterior probabilities $p(x_k|y_{1:k})$ can be obtained according to $p(x_k|y_{1:k-1})$. It is only a prediction process in the last step, and this step amend the last step of the forecast through the observation of the k moments. It is a process of filtering. The posterior probability is obtained by this step and then brought into the next prediction process to form a recurrence.

$$\begin{aligned} &p(x_k|y_{1:k}) \\ &= \frac{\int p(y_k|x_k, y_{1:k-1})p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})} \\ &= \frac{\int p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})} \end{aligned} \tag{4}$$

$$p\left(y_k|y_{1:k-1}\right) = \int p(y_k|x_k)p(x_k|y_{1:k-1})d_k \tag{5}$$

In the first and the second step of formula (4), y_k is only associated with x_k. The likelihood function $p\left(y_k|x_k\right)$ is determined by the measurement equation. In the formula $y_k = h\left(x_k\right) + n_k$, x_k is constant, and $p\left(y_k|x_k\right)$ is only related to the probability distribution of measurement noise n_k.

B. Monte Carlo Sampling

The Monte Carlo sampling method uses the sum of the samples to accomplish the integral calculation. A series of samples $x_1, \ldots, x_n$ are obtained from the probability distribution of target $p(x)$, which are used to estimate the expected value of the function in this distribution.

In Monte Carlo method, we define $f\left(x\right) = \delta\left(x_n - x_n^{(i)}\right)$ as the Dirac function. To carry out the target tracking or filtering, the expectation of the current state needed and is calculated as below:

$$\begin{aligned} &E\left[f\left(x_n\right)\right] \\ &\approx \int f\left(x_n\right)\hat{p}\left(x_n|y_{1:k}\right)dx_n \\ &= \frac{1}{N}\sum_{i=1}^{N}\int f(x_n^{(i)}) \end{aligned} \tag{6}$$

The expected value after filtering can be obtained by averaging the state value of the sampled particles. $f(x)$ is the state function of each particle.

C. Bayesian Filter

Since the posteriori probability is unknown, it is impossible to sample directly from the posterior probability distribution. The importance sampling method is introduced to solve this problem. The expected solution can be transformed into:

$$\begin{aligned} &E\left[f\left(x_k\right)\right] \\ &= \int f\left(x_k\right)\frac{p(x_k|y_{1:k})}{q(x_k|y_{1:k})}q(x_k|y_{1:k})dx_k \\ &= \int f\left(x_k\right)\frac{W_k(x_k)}{p(y_{1:k})}q(x_k|y_{1:k})dx_k \end{aligned} \tag{7}$$

$$W_k\left(x_k\right) = \frac{p(y_{1:k}|x_k)p\left(x_k\right)}{q(x_k|y_{1:k})} \tag{8}$$

According to this formula:

$$p\left(y_{1:k}\right) = \int p(y_{1:k}|x_k)p\left(x_k\right)\ dx_k \tag{9}$$

The formula (8) can be written a step further as:

$$\begin{aligned} &E\left[f\left(x_k\right)\right] \\ &= \frac{1}{p(y_{1:k})}\int f\left(x_k\right)W_k\left(x_k\right)q(x_k|y_{1:k})dx_k \\ &= \frac{E_{q(x_k|y_{1:k})}[W_k(x_k)f(x_k)]}{E_{q(x_k|y_{1:k})}[W_k(x_k)]} \end{aligned} \tag{10}$$

By Monte Carlo method, we can calculate the average of the N sample to get their expectations, then formula (10) can be approximated as:

$$\begin{aligned} &E\left[f\left(x_k\right)\right] \\ &\approx \frac{\frac{1}{N}\sum_{i=1}^{N}W_k\left(x_k^{(i)}\right)f\left(x_k^{(i)}\right)}{\frac{1}{N}\sum_{i=1}^{N}\tilde{W}_k\left(x_k^{(i)}\right)f\left(x_k^{(i)}\right)} \\ &= \sum_{i=1}^{N}\tilde{W}_k\left(x_k^{(i)}\right)f\left(x_k^{(i)}\right) \end{aligned} \tag{11}$$

$$\tilde{W}_k\left(x_k^{(i)}\right) = \frac{W_k\left(x_k^{(i)}\right)}{\sum_{i=1}^{N}W_k\left(x_k^{(i)}\right)} \tag{12}$$

The weight is not normalized in the formula (7) but normalized in the formula (12). It is not the average of all particles' states in the formula (11), but calculated by weighting. Each particle has a corresponding weight, and the weight value of the particle is positively related to its credibility. The greater the weight of the particles is, the more trusted the particle will be.

Assuming the importance probability density function is $q(x_{0:k}|y_{1:k})$, and is decomposed into:

$$q\left(x_{0:k}|y_{1:k}\right) = q(x_{0:k-1}|y_{1:k-1})\ q(x_k|x_{0:k-1}, y_{1:k}) \tag{13}$$

Then the recursion is marked for the posterior probability density function (Y_k represent $y_{1:k}$ for convenience):

$$\begin{aligned} & p(x_{0:k}|Y_k) \\ & = \frac{p(y_k|x_{0:k},Y_{k-1})p(x_{0:k}|Y_{k-1})}{p(y_k|Y_{k-1})} \\ & \propto p(y_k|x_k)p(x_k|x_{k-1})p\left(x_{0:k-1}|Y_{k-1}\right) \end{aligned} \tag{14}$$

Because $x(k)$ becomes $x_{0:k}$, Bias estimates need to carry out the integral calculation. But the decomposed form of the posterior probability is not required to use integral calculation. The recursive form of particle weights can be expressed as:

$$\begin{aligned} & w_k^{(i)} \\ & \propto \frac{p(x_{0:k}^{(i)}|Y_k)}{q(x_{0:k}^{(i)}|Y_k)} \\ & = w_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{0:k-1}^{(i)},Y_k)} \end{aligned} \tag{15}$$

In practical application, the filter is designed to determine the current state $p\left(x_k|y_{1:k}\right)$ rather than $p\left(x_{0:k}|y_{1:k}\right)$, which is pushed in the formula (15). So our assumption is that the importance of distribution function q should satisfy:

$$q\left(x_k|x_{0:k},y_{1:k}\right) = q\left(x_k|x_{k-1},y_k\right) \tag{16}$$

According to the importance of this assumption, it can be obtained that the importance distribution is related to the system state x_{k-1} and y_k in the previous time. So the formula (15) can be written as:

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k|x_{k-1}^{(i)})}{q(x_k|x_{k-1}^{(i)},Y_k)} \tag{17}$$

Based on formulas above, the weight and the state of the particles can be obtained. The state of each particle will be weighted by using the formula (11), and then be used to estimate the target state.

III. Multi-layers Particle Filter

Particle filter is based on the content above, so the traditional PF algorithm loses the diversity of particles. Too much amount of particles will also increase the complexity of calculation. In order to reduce the processing time and improve the accuracy rate, we propose our MLPF algorithm by dividing the particles into three parts and optimizing the principle particles using S-particles. Besides, adding R-particles in the next step also improve the performance of the algorithm.

A. Define the Particles for Keeping Diversity

Considering keeping diversity of particles, we divide particles into three parts: S-particles, M-particles and R-particles.

Definition of the S-particles:

$$p^s = \left\{s^s, w^s\right\} \tag{18}$$

$s^s = (x, y)$ indicates the position of the S-particles; w^s represents the weight of the S-particles. Definition of the M-particles:

$$p^p = \left\{s^p, w^p\right\} \tag{19}$$

$s^p = (x, y)$ indicates the position of the M-particles; w^s represents the weight of the M-particles. Definition of the R-particles:

$$p^r = \left\{s^r, w^r\right\} \tag{20}$$

$s^r = (x, y)$ indicates the position of the R-particles; w^r represents the weight of the R-particles. For the convenience of easy calculation, we choose traditional tracking object model. In this model, the object is represented by a rectangle frame M while W and H represent the width and height of the rectangle frame respectively. The target state is defined by:

$$M = \{X, W, H\} \tag{21}$$

B. Calculate the Weight of the Particles

The weight of the particles is determined by comparing the similarity between the rectangle frame M of each particle in the kth frame and the target frame in the $(k-1)$th frame. There are a lot of methods to compare the similarity between two regions, such as the matching of color histogram, the decomposition of the matrix of the region, the matching of the feature points based on the region and so on. In this paper, we use the histogram matching of the particle rectangle frame to calculate the similarity between the current frame coverage area and the target area in the upper frame. Weight of all the particles is calculated by the following formula:

$$w_k = \frac{1}{\sqrt{2\pi}\delta} e^{-\frac{d}{2\delta^2}} \tag{22}$$

Where d and ρ given as:

$$d = \sqrt{1 - \rho\left(a, b\right)} \tag{23}$$

$$\rho\left(a, b\right) = \sum_{i=1}^{m} \sqrt{a\left(i\right) b\left(i\right)} \tag{24}$$

$\rho\left(a, b\right)$ represents the coefficient of PAP. a represents the center point coordinates of the detected particles. b represents the center point coordinates of the target particles in the previous frame, d represents the distance between the current particle a and previous particle b, and w_k represents the weight of the particles after the transformation of the particle area similarity.

C. Optimization of the M-particle for Reducing Complexity and Keeping Diversity

S-particle is used to optimize M-particle. In this paper, a simple method is proposed.

Fig.1 illustrates the evolution of three kinds of particles in the whole process. We use empty frames and dots to represent targets and pixels respectively. Empty frames, green frame and black frame represent weight calculated frames, target and estimated target respectively. Black dots, red dots

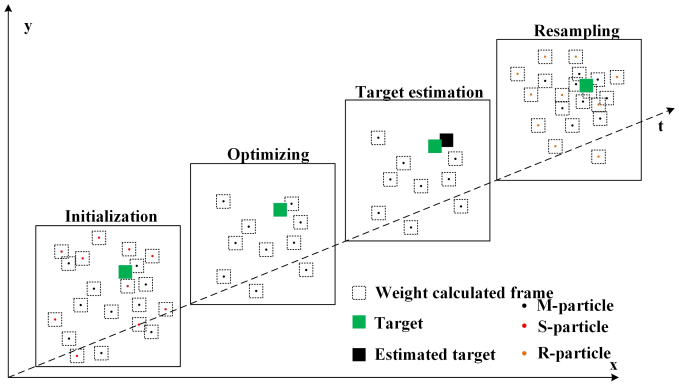

Fig. 1. The evolution of M-particles, S-particles and R-particles.

and orange dots represent M-particles, S-particles and R-particles respectively. And the dots are the center of the frame $M \times H$, which is introduced to compare the target with weight calculation. In order to display the weight of particles, the dot is represented by a solid circle with radius. Then Fig.1 can be updated to Fig. 2.

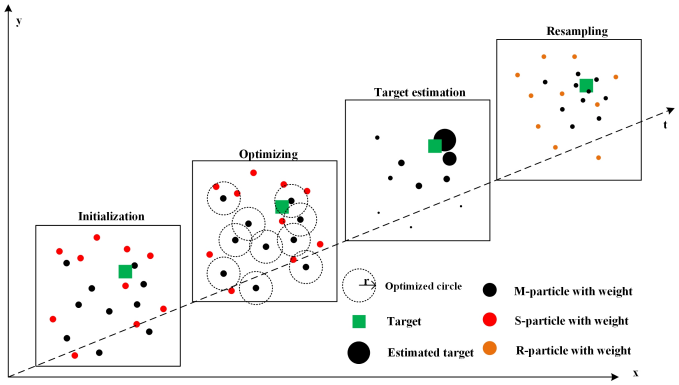

Fig. 2. The evolution of M-particles, S-particles and R-particles with weight.

Fig.2 shows the evolution of three kinds of particles with weight in the whole process. The phase of initialization randomly generates the M-particles and S-particles, which have equal number and weight. In the phase of optimization, we use an optimized circle with r radius to choose S-particles for optimizing the M-particle. In the target estimation, we regard the M-particle with largest weight as estimated target's position. Finally we randomly produce R-particles to replace the same number of S-particles and adjust M-particles' positions according to their previous weight. We can easily find that only half of total particles can participate in the target estimated stage.

Assuming the number of the M-particles is m, the center coordinates of the ith M-particle is s_i^p, and the weight is w_i^p. The state of an M-particles in a current frame is:

$$p^p = \{s_i^p, w_i^p\} \ , \ i = 1, 2, \ldots, m \tag{25}$$

$$s_i^p = \{x_i^p, y_i^p\} \tag{26}$$

x_i^p represents the ith M-particle's horizontal coordinate while y_i^p is the longitudinal coordinate of the ith M-particle. Assuming the number of the S-particles is n, the center coordinates of the jth S-particle is s_j^s, and the relative weight of the particles is w_j^s. The state of an S-particle in a current frame is:

$$p^s = \{s_j^s, w_j^s\} \ , \ i = 1, 2, \ldots, n \tag{27}$$

$$s_j^s = \{x_j^s, y_j^s\} \tag{28}$$

x_j^s represents the jth S-particle's horizontal coordinate, and y_j^s is the longitudinal coordinate of the jth S-particle. The circle drew by a dashed line in Fig. 2 is the set threshold. For every M-particle, we find out that all the S-particles satisfy:

$$\sqrt{\left(x_i^p - x_j^s\right)^2 + \left(y_i^p - y_j^s\right)^2} \leq r \tag{29}$$

We can find all the S-particles at a distance r or less. Assuming the jth S-particle, which is less than r away from the ith M-particle, is expressed as

$$p^s = \{s_{i,j}^s, w_{i,j}^s\} \tag{30}$$

$$s_{i,j}^s = \{x_{i,j}^s, y_{i,j}^s\} \tag{31}$$

Following is used to calculate the average weight of the jth S-particles around the ith M-particle obtained.

$$\overline{w_i^s} = \frac{\sqrt{\sum_{j=1}^{j} \left(w_{i,j}^s\right)^2}}{j} \tag{32}$$

Finally, weight of the ith M-particle is optimized by all S-particles around it based on the following formula.

$$\overline{w_i^p} = \overline{w_i^s} + w_i^p \tag{33}$$

D. Resampling

In order to reduce the degradation of the particles, the resampling process is introduced only for the M-particles in our algorithm. For all M-particles, weight of each particle is normalized by the following formula:

$$\eta_i = \frac{\overline{w_i^p}}{\sum_{i=1}^{i} \overline{w_i^p}} \tag{34}$$

According to the calculated weight ratio, the position of the M-particle is redistributed. Then we increase more M-particles around the M-particles with larger weight ratio and reduce the number of M-particles around the M-particles with smaller weight ratio. The update of the M-particle state is calculated by the formula (35):

$$m_i^p = m \times \eta_i \tag{35}$$

Where m is the total number of M-particles in the image, and m_i^p represents the number of M-particles around the ith

original M-particle. In a new cycle, the equal number of R-particles are generated randomly and the new R-particles are treated as a new set of S-particle. According to the weight of the M-particle, the state of the tracking target is estimated by the result of the normalized operation of the M-particles with the formula below.

$$\begin{cases} X = \sum_{i=1}^{m} x_i^p \cdot \eta_i \\ Y = \sum_{i=1}^{m} y_i^p \cdot \eta_i \end{cases} \tag{36}$$

X and Y respectively represent the target location estimated by the position and the weight ratio of the M-particles.

E. MLPF Algorithm

Based on the contents above, the description of the target tracking algorithm for multi-layers particle filter is presented in Fig 3 and Algorithm 1.

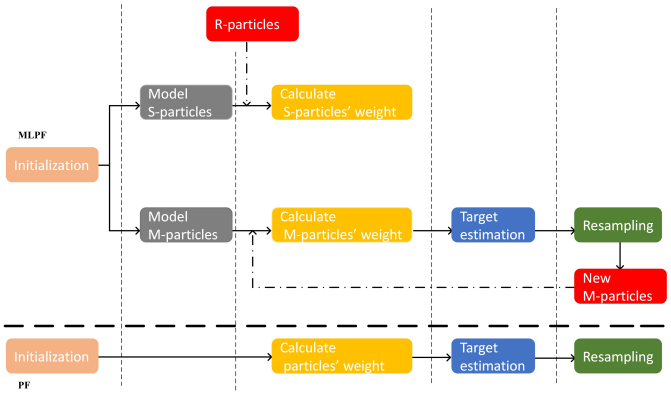

Fig. 3. MLPF tracking algorithm.

Algorithm 1 MLPF Algorithm.

Require: $p^s = \{s^s, w^s\}$, $p^p = \{s^p, w^p\}$, $M = \{X, W, H\}$, X and Y
Ensure: X and Y
  for every S-particle and M-particle **do**
    Calculate w_k through formula (22)(23)(24)
  end for
  Initialize $r = 3W$
  for every principal particle p_i^p **do**
    Find all subordinate particles satisfied the formula(29)
    Calculate $\overline{w_i^p}$ through fomula(32)(33)
  end for
  for every principal particle p_i^p **do**
    Calculate η_i through formula(34)
    Calculate m_i^p through formula(35)
  end for
  Calculate X' and Y' through formula(36)
  if $|X - X'| < \varepsilon$ and $|Y - Y'| < \varepsilon$ **then**
    $X = X' and Y = Y'$
  end if

In the kth frame image, the initial sample of the particle is built for each target. First, we set the number of M-particles to m and the number of the S-particles to n. Second, we establish state model for each M-particle and S-particle, then calculate the weight of every M-particle and S-particle. Next, we optimize the M-particles and calculate the weight of the them again. At last, we calculate the weights of M-particles through normalize calculation. In the update phase, M-particles are reallocated according to the weight ratio. Randomly generated R-particles are then considered as new S-particles.

IV. Evaluation

This paper has implemented two experiments to test the effectiveness of our proposed algorithm. In this chapter, we will describe the experiments in detail. We choose an aerial fast motion target with light interference as our dataset. We consider both accuracy rate and processing time to compare MLPF with traditional PF. The dataset comes from military parade of Mikoyan MiG-29. We use part of two videos, which can reflect light interference or fast motion. The first experiment selects the video which contains light interference. The second experiment chooses the video which contains fast motion. In order to compare the MLPF algorithm with the traditional PF algorithm, two video sequences are used. In MLPF algorithm, we set total particles' number to be 200, including 100 M-particles and 100 S-particles. Particles in the traditional particle filter is also set to be 200.

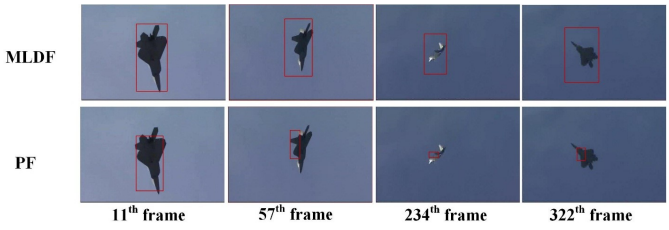

Fig. 4. MLPF and PF capture the target in video frames with light interference.

In the first experiment, light changes rapidly. The results of MLPF algorithm and traditional PF algorithm are shown in Fig. 4. They both have good performance in the 11th frame, but when light changes, the traditional PF algorithm can not catch full target. It also can be seen in the 57th frame, when the posture of the aircraft changs, the MLPF algorithm is able to track it effectively compared with traditional PF. Similarly, in the 234th frame, the traditional PF algorithm can not overcome the influence of light.

We test 100 times and calculate the average value. From table I, it can be seen that proposed MLPF algorithm and traditional PF algorithm both use 200 particles. MLPF costs 21 seconds while PF costs 47 seconds, so MLPF shows its advantage in reducing the processing time in this experiment. The computer with MLPT algorithm achieves a processing frequency of 16.48 frames per second, basically close to real-time requirements while it only achieves 7.36 with PF. If we

TABLE I
COMPARING PERFORMANCE OF MLPF AND PF AGAINST LIGHT INTERFERENCE.

	Particles' Number	Total Frames	Cost (s)	Frequency ($\sum \frac{frames}{s}$)	Performance ($\frac{righttracking}{\sum frames}$)
MLPF	100 M,100 S	346	21	16.48	96%
PF	200	346	47	7.36	64%

TABLE II
COMPARING PERFORMANCE OF MLPF AND PF AGAINST FAST MOVING.

	Particles' Number	Total Frames	Cost (s)	Frequency ($\sum \frac{frames}{s}$)	Performance ($\frac{righttracking}{\sum frames}$)
MLPF	100 M,100 S	1641	121	13.56	89%
PF	200	1641	237	6.92	46%

regard the coverage of 70% of the target as accurate tracking, the accuracy of the MLPF algorithm is 32% higher than the traditional PF algorithm.

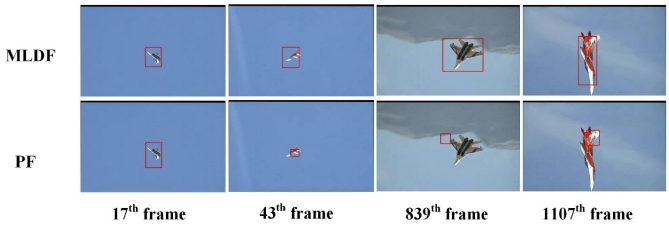

Fig. 5. MLPF and PF capture the target in video frames of fast moving.

In the experiment of the second video, the aircraft's posture changes significantly. The results of MLPF algorithm and traditional PF algorithm are shown in Fig 5. They both have good performance in the front frames of the video. But from the 43rd frame, the tracking precision of traditional PF algorithm is getting worse while the target moves very fast. In the 839th frame, traditional PF algorithm completely lost the target while our MLPF algorithm is still able to track the aircraft effectively.

We repeat the test 100 times and calculate average value. From table 2, proposed MLPF algorithm and traditional PF algorithm both use 200 particles. MLPF costs 121 seconds while PF costs 237 seconds, so MLPF also shows its advantage in reducing the processing time in this experiment. The computer with MLPT algorithm achieves a processing frequency of 13.56 frames per second basically close to real-time requirements while it only achieves 6.92 with PF. In the same way, if we regard the coverage of 70% of the target as accurate tracking, the accuracy of MLPF algorithm is 45% higher than traditional PF algorithm.

V. CONCLUSION

Traditional PF algorithm loses diversity of the particles during the phase of resampling. Adding more particles to improve the diversity of the particles will inevitably increase the complexity of computation . Because of that, it is difficult for traditional PF algorithm to track target, which is fast moving and facing light interference. In our algorithm, the particles are divided into three categories: M-particles, S-particles and R-particles. M-particles are optimized by S-particles, so M-particles retain the whole particle's attribute. Only the M-particles participate in target estimation and resampling process, so the processing time is reduced. New random R-particles can increase the diversity of total particles. Considering the condition of using the same number of particles, our algorithm has less computation complexity and higher tracking accuracy. Based on experimental results, MLPF algorithm has improved diversity of particles and outperformed traditional PF algorithm. In the first experiment, the processing speed of MLPF is almost 2.24 times faster than PF while the accuracy achieves 1.50 times. In the second experiment, the process speed of MLPF is 1.96 times faster than PF and the accuracy achieves 1.93 times.

REFERENCES

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[2] Y. Su, Q. Zhao, L. Zhao, and D. Gu, "Abrupt motion tracking using a visual saliency embedded particle filter," *Pattern Recognition*, vol. 47, no. 5, pp. 1826–1834, 2014.

[3] P. Sarkar, "Sequential monte carlo methods in practice," *Technometrics*, vol. 45, no. 1, pp. 106–106, 2003.

[4] 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.

[5] A. Doucet, B.-N. Vo, C. Andrieu, and M. Davy, "Particle filtering for multi-target tracking and sensor management," 2002.

[6] L. Zhang, L. Yang, and T. Luo, "Unified saliency detection model using color and texture features," *Plos One*, vol. 11, no. 2, 2016.

[7] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 263–270.

[8] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *European Conference on Computer Vision*. Springer, 2012, pp. 864–877.

[9] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1838–1845.

[10] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[11] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European Conference on Computer Vision*. Springer, 2014, pp. 254–265.