Optimization of the Occlusion Strategy in Visual Tracking

Jinghuan Wen, Huimin Ma*, and Xiaoqin Zhang

Abstract: Interference and anti-interference are two opposite and important issues in visual tracking. Occlusion interference can disguise the features of a target and can also be used as an effective benchmark to determine whether a tracking algorithm is reliable. In this paper, we proposed an inner Particle Swarm Optimization (PSO) algorithm to locate the optimal occlusion strategy under different tracking conditions and to identify the most effective occlusion positions and direction of movement to allow a target to evade tracking. This algorithm improved the standard PSO process in three ways. First, it introduced a death process, which greatly reduced the time cost of optimization. Second, it used statistical data to determine the fitness value of the particles so that the fitness more accurately described the tracking. Third, the algorithm could avoid being trapped in local optima, as the fitness changes with time. Experimental results showed that this algorithm was able to identify a global optimal occlusion strategy that can disturb the tracking machine with 86.8% probability over more than 10,000 tracking processes. In addition, it reduced the time cost by approximately 80%, compared with conventional PSO algorithms.

Key words: particle swarm optimization; virtual tracking; occlusion interference; death process; statistical fitness function

1 Introduction

Visual tracking has been used in many applications but presents two major challenges: how to design robust tracking algorithms to eliminate the influence of the interference, and how to determine strategies that can use interferences to allow a target to escape from tracking. This paper focuses on the second aspect. There are two kinds of common interference: feature-based interference[1] and irrelevant interference. Irrelevant interference, such as machine noise, machine tremble, and flash, is strongly dependent on the tracking machine. For the defender, feature-based interference is more significant because both machines and human operators are easily disturbed by a distractor object that appears in the visual field[1]. Feature-based interference can arise from natural objects or from design. There are two common sources of feature-based interference: distractor objects that share features with the tracking target[2], such as different people dressed similarly[3], and the presence of objects that occlude the view and disguise the features of the target, such as leaves placed in front of the camera, moving obstacles, and the smoke in the field.

Occlusion is a common interference technique to destroy most real-world tracking schemes such as multi-target tracking, human tracking, and vehicle tracking. To deal with static occlusion, some researchers establish a depth map of the walls, entrances, and other barriers in the tracking scene before tracking and change the template by computing their positional relationship to the target[4]. Depth marking can significantly reduce the problem of static occlusion: the fitting probability is more likely to be 0.5, while the algorithm without occlusion analysis drops significantly below 0.05. To deal with dynamically changing and unpredictable occlusion, some algorithms...
freeze their template until the occlusion disappears\cite{5}. These algorithms can handle occlusion for a time duration corresponding to the acquisition of no more than 25 frames and cover at most 25\% of the outliers.

Occlusion is a strong interference technique, and its performance depends crucially on the location relationship between the occluding objects and the target, the size of the occluding objects, and the occlusion time. Dynamically changing occlusion is a more serious problem than static occlusion, because the occluding objects are more likely to be identified as part of the target by the algorithm. In recent years, Particle Swarm Optimization (PSO) has been developed significantly, and has been applied to difficult optimization problems. In PSO methods, each particle represents a set of parameters or variables, but groups of particles are used in the analysis. The optimal parameters or variables are obtained by iteratively evaluating fitness and moving particles. A recent application of PSO methods was to shipboard power system management problem\cite{6}. Soudan and Saad\cite{7} have introduced a dynamic population strategy to speed up PSO execution.

Our work faces three challenges. First, the time cost for thousands of iterations is unacceptable, as each simulation takes approximately 30 seconds. Second, the data are binary. Third, there is no proper fitness function which can represent the effect of the interference strategy. In this paper, we propose an inline PSO approach, to determine the optimal occlusion strategy under defined tracking conditions. The main contribution of this work is that we modified the standard PSO method to meet these three challenges. A death process was introduced to decrease the number of iterations, thus reducing the time cost, and a proper fitness function was designed specifically for binary statistical data. This method of designing fitness functions can be easily applied to other optimization problems which present the same challenges. Our focus was on dynamically changing feature-based occlusion simulation systems.

In Section 2, we introduce our tracking simulation platform and the background of our experiment. In Section 3, we introduce our optimal algorithm, and discuss the choice of the parameter value. In Section 4, we present and analyze the result of our experiment.

\section{Simulation Platform and Experiment Background}

Our tracking scene was set outdoors, with influences from the natural elements such as the wind speed and direction and the temperature. The probability of unsuccessful tracking reflected the influence of these elements. Our model used a scenario in which the target produced a large amount of smoke at a certain position, to occlude the tracking view. Our aim was to optimize the position at which the occlusion caused tracking to fail or be seriously vitiated. Because of the difficulty of obtaining real video, we built a simulation system in which the anti-tracking algorithm competed with the tracking algorithm, as shown in Fig. 1.

Our simulation system modeled the tracking target, including its scale and radiated character, movement, the tracking machine (approaching the target at a constant velocity), and the environment. The tracking algorithm used infrared images which was related to occlusion tactics. The simulation of the tracking process contained the entire process of one tracking event, and was distinctive and convincing.

In our simulation system, the entire simulation process takes approximately 30 seconds. It is a short time for only one event simulation, but a long time for the optimization algorithm requiring thousands of event simulations. To perform the optimization, we need to run the simulation hundreds of times to collect the data. This requires huge time cost and a more effective method should be found. Our target was for the optimization loop to end at an average of 600 iterations to provide 100 results over 10 days. Figure 2 shows how the platform works.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{simulation_platform.png}
\caption{Simulation platform.}
\end{figure}
Fig. 2 An example of the occlusion process. (a) The occlusion smoke appears. (b) The tracking is disturbed. (c) The tracking center moves to the smoke. (d) The object is saved.

3 Optimal Algorithm

Over recent years, many optimization methods have been applied in many areas. Among them, swarm intelligence, which was discovered through simulation of simplified social systems, has proved an effective way of solving continuous nonlinear functions. PSO is an important swarm intelligence optimization method.

3.1 The standard PSO

The PSO method has attracted the interest of researchers in various fields. In 1995, Kennedy and Eberhart[8] performed a simulation, influenced by Hepper and Grenander’s work[9], which used analogs of bird flocks searching for corn. This was developed into a powerful optimization method, known as PSO. The original PSO can be described as follows.

A group of particles, where each particle is a set of parameters, separated in a space (the search area) share information to locate the space with the largest food availability (the highest fitness value). No single particle knows where the highest fitness is located, but it will keep searching for better places.

In 1998, Shi and Eberhart[10] added the important parameter of inertial weight $w$, which provided the basic PSO update formula:

$$\begin{align*}
V_i(t+1) &= w(t)V_i(t) + \phi_1(t)(P_i(t) - X_i(t)) + \\
&\quad \phi_2(t)(P_g(t) - X_i(t)) \\
X_i(t+1) &= X_i(t) + V_i(t+1)
\end{align*}$$

where $P_i(t)$ is the best position of the particle $i$, and $P_g(t)$ is the global best position of the swarm. $V_i(t)$ and $X_i(t)$ represent the velocity and position of the particle $i$ at the current time, respectively.

3.2 Our optimized algorithm

Building on the earlier models, we proposed a PSO strategy for a feature-based, dynamically changing occlusion.

Our algorithm was designed and implemented for optimization problems with very high time costs, binary data, and unclear fitness functions, in contrast with the standard PSO, which was designed for simple fitness functions (with little time cost) and continuous data.

An optimization problem with binary data outputs 0 or 1 for each sample, which makes it difficult to evaluate the fitness of a particle. The traditional way of evaluating fitness is to sample multiple times using one set of parameters, which requires large-scale calculation. Our method introduced a special fitness function, which takes into account the data of all nearby particles, thus significantly reducing the time cost. If the standard process of fitness evaluation requires $N$ samples at each point, our method can be $N$ times faster, because it uses each data point redundantly. This special fitness function is random and changes with time, so that the algorithm avoids being trapped in local optima.

Our algorithm also introduced a death process to remove particles that were very unlikely to be optimal. This process also reduced the time cost.

The comparison between standard PSO and our algorithm are shown in Table 1, and the flowchart of our algorithm is shown in Fig. 3.

At the start of each optimization run, we chose a certain tracking condition, including the direction and velocity of wind and the perspective of the tracking machine. We used the chosen condition to define the search space and initialize the particles. Then, we set the information of each particle, as well as the tracking condition, in our simulation platform. After the simulation for the initial particles ended, we checked for a successful interference result. If no successful result was found, the initial step was repeated at most
### Table 1 Comparison between standard PSO and our algorithm.

|                         | Standard PSO                                      | Our algorithm                              |
|-------------------------|--------------------------------------------------|--------------------------------------------|
| Applicable fitness function          | Simple, usually constant                          | Complex, can be random                     |
| Evaluating fitness of binary data   | Multiple samples at one point                    | Use all nearby data                        |
| Time cost                 | Very large (due to multiple samples)              | Normal (complex fitness function)         |
| Special process           | None                                             | Death process                              |

![Flowchart of our algorithm.](image)

#### 3.2.1 Initial step

(1) Initial particle

The choice of initial number of particles and the criterion of convergence vary among different applications\[6, 7, 10–12\]. However, our algorithm differed at several points from these PSO algorithms. The first difference was the number of initial particles. Because the tracking algorithm was optimized through the efforts of many research groups, we were unable to guarantee a successful anti-tracking result at the initial step of the first simulation. We first ran our simulation system seven thousand times with random inputs, and found that the successful anti-tracking probability was approximately 10%. The number of particles was therefore set to 80, to give a probability of approximately 0.22% that none of them would successfully distract the tracking. Ideally, the initial number of particles should be large enough that the result of the first iteration is not “all failed”, and while being as small as possible, to reduce the time cost. For some cases, for example, when the optimal position was at the corner of the search space, or its scale was too small, all the occlusion particles failed. In these cases, the initial process was reset, and the test was repeated. If no particles achieved a successful result in the repeated run, this tracking condition was defined as a stable tracking condition for which we were unable to find a set of parameters where smoke occlusion worked.

(2) Occlusion particles

As stated above, many elements influence tracking. To address this problem, we focused on a few significant elements. We defined the angle between the direction of movement of the target and the direction of the tracking machine (in a real tracking scene, the direction and the velocity of the wind does not changed). We also defined the character of our occlusion particles to be the position of the occlusion smoke and the direction in which the target moves in its attempts to escape tracking. The task was to choose the best position for the occlusion under a certain condition. The individual particle’s position vector in the search space was specified as follows:  $X_i(t) = [\theta, \phi, \theta_c]^T$, where $\theta$ and $\phi$ are the position of the occlusion smoke in the spherical coordinate system (we set $\rho = 200$ as a constant because this distance was equivalent to the scale of the smoke), and $\theta_c$ is the direction the target chooses for evasion.

(3) Search space
As the occlusion cannot appear outside the tracking view, we defined our search space as shown in Fig. 4. The shaded area was our searching area, where $\theta$ and $\phi$ are both set to the range $(0, \pi)$, and $\theta_c$ is limited to the range $(-30^\circ, 30^\circ)$ based on the turning speed of the target.

4) Maximum velocity

At each turn, there was a limit to the velocity at which the particles could continue to move as a swarm. In earlier research\cite{13}, it has been suggested that the maximum velocity should be selected based on the previous tracking process. However, this method of maximum velocity assignment is not suitable in circumstances where the fitness can change. For example, when the swarm finds that the position they are moving towards has become less suitable than another position, they will change their direction and approach the new position instead. The most efficient velocity when finding a new position is different from the small steps when moving towards the best position. Thus, we defined our maximum velocity with reference to the overall search length.

3.2.2 Fitness calculation

As with other evolutionary algorithms, PSO requires multiple fitness evaluations to be carried out. Sometimes, however, the cost is intolerable, and approximate models are used instead. However, our optimization system used the original information as the output of our simulation system, which simulated the entire process of visual tracking under the defined conditions. The result of a simulation is difficult to estimate before the program has completed. We were therefore unable to use standard fitness approaches.

To reduce effort, we can avoid the simulation step by finding the convincing probability of successful anti-tracking.

First, we set the fitness value as the probability of the successful particles being in the nearby area for all of the particles.

$$f_j(t) = \sum_{i=1}^{m} \left( \frac{1}{m} \text{Res}_i(t) \right)^\alpha$$ \hspace{1cm} (2)

where $\text{Res}_i(t)$, the simulation result of the $i$-th particle at time $t$, is a two-valued function where 0 represents failure and 1 represents successful occlusion. Only the particles with a certain distance from the $j$-th particle are considered as valid, and $m$ represents their total number.

We found that our algorithm easily fell into local optimization traps. To avoid this, we increased the influence of unsuccessful particles near the optimization candidate and increased the fitness of the initial particles. Our fitness value is acquired at every loop, so the fitness of our particles changed during the search, in contrast to standard PSO algorithms. Because the fitness is based on statistical data, maximizing the information used led to improved accuracy. This required a record to be kept of all of the positions that had been tested, to allow the fitness value to be recalculated at the end of the simulation step. We therefore changed the fitness function to

$$f_j(t) = \sum_{i=1}^{n} \frac{1}{r_{ij}(t)} \left( (1 + \alpha)\text{Res}_i(t) - \alpha \right)$$ \hspace{1cm} (3)

where $r_{ij}(t)$ is the distance between the $i$-th particle and the $j$-th particle, and $\alpha > 1$ is the weight given to the failed occlusion. This changes $f_j(t)$ by adding 1 for a successful occlusion and subtracting $\alpha$ for a failed one. After this change, our algorithm was able to identify the local “best positions” that eventually turned out to be non-optimal. We assume that the nearby unsuccessful particles would make the choice of a local best position unstable, and therefore we increased their weighting.

The series of images in Fig. 5 shows the process of escaping a local optimum. First, the particles converged to the local best position at the top left corner of the first image (Fig. 5a). However, after a new iteration, some failing occlusion particles remained nearby. The fitness calculation revealed a more promising location, which eventually became the overall optimum (Fig. 5b). All of the particles therefore moved towards that new point (Fig. 5c) and converged on the final answer (Fig. 5d).
Fig. 5 An example process of escaping a local optimum. The “+” particles are the successful particles, and the “o” ones are the failing initial particles. (a) The particles converge to the local optimum. (b) A more promising location appears. (c) The particles move towards the new position. (d) The particles converged on the final answer.

3.2.3 Death of particles

After the initial step, there were 80 particles in the search loop, which requires a large-scale computation with an unaffordable time cost. We therefore introduced a death process. This idea was inspired by the work of Xie\textsuperscript{[14]}, who introduced the extinction of the whole swarm, to increase the variety of the swarm and the exploratory power of the particles. However, our death process was a result of competition rather than extinction. After choosing the most likely food position at the initial step, the distance of each particle from that position was calculated. If the distance exceeded a threshold, it was deemed dead due to the starvation, and removed from the search loop.

The threshold was varied with the location of the best solution. To ensure that approximately the same number of particles survived at the end of the death process, the threshold $T_j$ was set as follows:

\begin{equation}
T_j = \begin{cases} 
2\beta S_j - D_j(i), & D_j(i) < \beta S_j; \\
\beta S_j, & \text{otherwise}
\end{cases}
\end{equation}

\begin{align*}
D_j(i) &= \min\{\max a_j - P_g(i)_j, P_g(i)_j - \min a_j\}, \\
S_j &= \max a_j - \min a_j,
\end{align*}

where $j$ is the index of each parameter, $i$ is the turn of the loop and $\max a_j$ and $\min a_j$ are the boundary of the searching space. Hence, $S$ is the scale of the searching space and $D(i)$ is the minimal distance from $P_g(i)$ to the boundary. $2\beta$ is the ratio of the scale of the survival space to the searching space, so almost 20% of the particles should remain in the search loop, if a typical value of $\beta = 0.1$ was used. This made our method five times faster than conventional approaches.
3.2.4 Convergence criterion

At the end of the search loop, a different convergence criterion was needed to shorten the searching. Because we did not need to specify the exact best position for occlusion, we could end the optimization as soon as a position was identified, which provided effective occlusion. We defined our convergence criteria for the three following different cases. First, if 90% of the occlusion particles were successful, the optimization was considered a success and the search was ended.

Second, if after a long search the swarm became very crowded (judged by its variance) but was still unable to achieve the minimum performance, this was considered a compromise result under poor conditions. Finally, if the swarm had still not converged after 800 rounds (i.e., 10 times of the number of the particles[11]), the search was stopped and the most recent best solution was accepted as the result. This was introduced to stop the system running for an extended period under poor conditions.

3.2.5 Parameter determination and convergence

Inertia weight is a parameter that has a significant influence on the exploratory behavior of the particles.

As the search continued, the particles moved as we defined. At each step, the PSO updated each dimension of $X_i(t + 1)$ and $V_i(t + 1)$ of every particle following Eq. (1) that can transform as follows:[15]:

$$
\begin{align*}
\begin{bmatrix}
V_i^T(t + 1) \\
X_i^T(t + 1)
\end{bmatrix} &=
\begin{bmatrix}
w(t) & -\phi_1(t) - \phi_2(t) \\
w(t) & 1 - \phi_1(t) - \phi_2(t)
\end{bmatrix}
\begin{bmatrix}
V_i(t) \\
X_i(t)
\end{bmatrix} +
\begin{bmatrix}
\phi_1(t) \\
\phi_1(t)
\end{bmatrix}
\begin{bmatrix}
\phi_2(t) \\
\phi_2(t)
\end{bmatrix}
\begin{bmatrix}
P_i^T(t) \\
P_i(t)
\end{bmatrix} +
\begin{bmatrix}
P_i^T(t) \\
P_i(t)
\end{bmatrix}
\begin{bmatrix}
P_i^T(t) \\
P_i(t)
\end{bmatrix},
\end{align*}
$$

(5)

This is a linear time-varying discrete system.

While the fitness values of the particles were changing within the loop, the best position of each particle became unreliable. Therefore, we did not record $P_i(t)$, and set $\phi_1(t)$ to zero.

We needed to focus on the value of the inertia weight. To simplify the problem, we first assumed that the inertia weight does not change with time. Equation (1) allows Eq. (5) to become

$$\begin{align*}
X_i(t + 2) &= (w + 1 - \phi_2(t + 1))X_i(t + 1) - wX_i(t) + \phi_2(t)P_i(t),
\end{align*}
$$

(6)

The condition for convergence could be identified, so the mean value of $X_i(t)$ could be defined as follows:

$$
\begin{align*}
E(X_i(t + 2)) &= (w + 1 - u)E(X_i(t + 1)) - wE(X_i(t)) + uE(P_i(t)),
\end{align*}
$$

(7)

where $u$ is the mean value of the random variable $\phi_2(t)$, which is independent of $X_i(t)$.

However, $P_i(t)$ is a stochastic process that depends upon another stochastic process, $X_i(t)$. As $P_i(t)$ does not significantly change during iteration, we assumed that $P_i(t)$ has been determined at the beginning of the iteration, and would not change during the process of optimization. This allowed the solution of Eq. (7) to be defined as

$$
\begin{align*}
\lambda_1 &= \frac{(w + 1 - u)^2 - 4w}{2}, \\
\lambda_2 &= \frac{(w + 1 - u)^2 - 4w}{2},
\end{align*}
$$

(8)

where $p = E(P_i(t))$, and $a_1$ and $a_2$ can be defined by the initial condition $\{X_i(0), X_i(1)\}$.

$$
\begin{align*}
a_1 &= -\lambda_2 E(X_i(0)) + E(X_i(1)) + (\lambda_2 - 1)p1, \\
a_2 &= -\lambda_2 E(X_i(0)) + E(X_i(1)) + (\lambda_1 - 1)p1,
\end{align*}
$$

(9)

So $|w| < 1$. And from the result of Liu’s work[6], we know if the inertia weight satisfies the equation below, the sequence $\{X_i(t)\}$ will be convergent.

$$
\begin{align*}
&\frac{u^2 - \sigma^2 - \sqrt{(u^2 - \sigma^2)^2 - 8u((u - 1)^2) + \sigma^2 - 1}}{4u} < w < \\
&\frac{u^2 - \sigma^2 + \sqrt{(u^2 - \sigma^2)^2 - 8u((u - 1)^2) + \sigma^2 - 1}}{4u}
\end{align*}
$$

(10)

where $\sigma^2$ is the variance of the random variable $\phi_2(t)$. If an inertia weight that satisfies Eq. (10) is chosen, the sequence $\{X_i(t)\}$ will be a mean square convergence, and the PSO algorithm will find the final answer. This equation provides a sufficient condition for the convergence of the PSO algorithm. Inertia weight with a value that is too large or too small will likely make the system unstable. Normally, $\phi_2(t)$ is a uniform random variable within $[0, 2]$, so $u = 1$, and the value of $w$ should be within $\left(0, \frac{1}{6} (\sqrt{13} + 1)\right)$, to enable the PSO to converge.

4 Experimental Results

To test our algorithm, we evaluated it using a simulation system. To ensure effectiveness, we use three different kinds of tracking algorithms, distinguished by the style of tracking window. One used the “adaptive window tracking”, based on the mean shift clustering algorithm, with a kernel function to deal with the occlusion[16]. The other two, “fasten window tracking” algorithm
and “group window tracking” algorithm, both used the shape features for template matching.

We implemented our method using typical parameters $\alpha = 2$ and $\beta = 0.1$, though in practice the experimental results were similar when $\alpha$ was 1.5–4 and $\beta$ was 0.05–0.2.

Our assumptions were as follows.

(1) We assumed that the tracking machine approached the target from a distance of 10 km away at a speed of 300 m/s.

(2) The occlusion smoke covered nearly 50% of the target area and appeared when the tracking machine was 7 km away.

We performed over 15,000 simulations for each tracking algorithm, under seven different tracking conditions, as shown in Table 2. The effort of the optimization strategy for the different tracking conditions was calculated. Optimization did not always achieve the best solution. Sometimes the solution was found quickly (Fig. 6). Sometimes, multiple attempts failed to disturb the tracking algorithm. In other cases, a strategy to defeat the tracking algorithm was found after several attempts at convergence (Fig. 7).

The complete results are summarized in Table 3. In 12 out of 61 conditions, none of the occlusion particles defeated the tracking machine (which we defined as a stable tracking condition) when the direction of the tracking machine was $90^\circ$ and the tracking algorithm used was “adaptive window tracking”. When the perspective of the tracking machine changes from $90^\circ$ to $0^\circ$, the simulation cost of optimization declined significantly, which means the performance of the three tracking algorithms also declined significantly. When the tracking machine approached the target from a direction of $180^\circ$, optimization was unnecessary because the potential solution could be easily ascertained after the initial step. It was demonstrated that none of these three tracking algorithms worked well when tracking directly towards the target, because the target’s features were disguised. The average time for a single optimization showed the difficulty of the optimization: the greater the interference area, the shorter the time taken to arrive at an optimal occlusion strategy. The confidence level showed the probability of the selected solution defeating the tracking machine, and therefore the reliability of our optimization algorithm and the performance of tracking algorithms. The group tracking algorithm proved slightly weaker than the other two algorithms as it required shorter time to achieve the optimization, with a higher confidence level.

## 5 Conclusion

This paper proposed an inline PSO to find the best occlusion strategy against a visual tracking algorithm. The algorithm used statistical data to set the fitness value of the particles, which reduced the time cost while providing the capability to escape local optima, without requiring information to be recorded from each particle. Because the fitness calculation took 30 seconds to perform the simulation for each particle, we added a death process after the swarm was initiated. We demonstrated that more reliable algorithms require more time to complete the optimization, and that the confidence level also declines. The result of the experiment can be used as a benchmark for tracking algorithms.

## References

[1] V. S. Störmer, S. C. Li, H. R. Heekeren, and U. Lindenberger, Feature-based interference from unattended visual field during attentional tracking in younger and older adults, *Journal of Vision*, vol. 11, no. 2, pp. 1–12, 2011.
Fig. 7 An unsuccessful result of the optimization. (a) Initial result: two potential centers; (b) After a few attempts, all failed; (c) A new convergence center appears; (d) Final convergence center.

[2] V. Ablavsky and S. Sclaroff, Layered graphical models for tracking partially-occluded objects, in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1758–1775.

[3] R. Vezzani and R. Cucchiara, Ad-hoc: Appearance driven human tracking with occlusion handling, in First Internat. Workshop Tracking Humans for the Evaluation of Their Motion in Image Sequences (THEMIS’ 2008), 2008, pp. 9–18.

[4] D. Greenhill, J. Renno, J. Orwell, and G. A. Jones, Occlusion analysis: Learning and utilising depth maps in object tracking, Image and Vision Computing, vol. 26, no. 3, pp. 430–441, 2008.

[5] H. T. Nguyen and A. W. M. Smeulders, Fast occluded object tracking by a robust appearance filter, Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 8, pp. 1099–1104, 2004.

[6] H. Liu, Research of particle swarm optimization algorithm and its application in ship engineering, (in Chinese), PhD dissertation, Harbin Engineering University, China, 2008.

[7] B. Soudan and M. Saad, An evolutionary dynamic population size PSO implementation, in Information and Communication Technologies: From Theory to Applications, 2008, pp. 1–5.

[8] J. Kennedy and R. C. Eberhart, Particle swarm optimization, in Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

[9] F. Heppner and U. Grenander, A stochastic nonlinear model for coordinated bird flocks, in The Ubiquity of Chaos, S. Krasner, Ed. American Association for the Advancement of Science, 1990, pp. 233–238.

[10] Y. Shi and R. Eberhart, A modified particle swarm optimizer, in Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on, 1998, pp. 69–73.

[11] K. Zhu, J. Li, and H. Baojin, Trajectory optimization of the exploration of asteroids using swarm intelligent algorithms, Tsinghua Science and Technology, vol. 14, no. S2, pp. 7–11, 2009.

[12] J. Hu, J. Song, M. Zhang, and X. Kang, Topology optimization for urban traffic sensor network, Tsinghua Science and Technology, vol. 13, no. 2, pp. 229–236, 2008.

[13] J. Fan, X. Shen, and Y. Wu, Closed-loop adaptation for robust tracking, in Computer Vision- ECCV 2010, 2010, pp. 411–424.

[14] X. Xie, Hybrid particle swarm optimizer with mass extinction, in Proceedings of the International Conference on Communication, Circuits and Systems, Chengdu, China, 2002, pp. 1170–1173.
Table 3  The optimization result for three tracking algorithms under different conditions.

| Algorithm            | Direction of the tracking machine (°) | Optimization times | Reliable tracking times | Average time of single optimization (h) | Confidence level |
|----------------------|----------------------------------------|--------------------|-------------------------|----------------------------------------|------------------|
| Adaptive window tracking | 0                                      | 155                | 0                       | 0.81                                   | 0.978            |
|                      | 30                                     | 84                 | 0                       | 1.49                                   | 0.927            |
|                      | 60                                     | 68                 | 0                       | 1.84                                   | 0.819            |
|                      | 90                                     | 61                 | 12                      | 2.06                                   | 0.831            |
|                      | 120                                    | 57                 | 0                       | 2.19                                   | 0.747            |
|                      | 150                                    | 44                 | 0                       | 2.84                                   | 0.769            |
|                      | 180                                    | 187                | 0                       | 0.67                                   | 1.000            |
| Group window tracking | 0                                      | 174                | 0                       | 0.72                                   | 0.990            |
|                      | 30                                     | 102                | 0                       | 1.22                                   | 0.950            |
|                      | 60                                     | 50                 | 0                       | 2.50                                   | 0.807            |
|                      | 90                                     | 55                 | 0                       | 1.84                                   | 0.727            |
|                      | 120                                    | 52                 | 0                       | 2.40                                   | 0.727            |
|                      | 150                                    | 50                 | 0                       | 2.50                                   | 0.805            |
|                      | 180                                    | 187                | 0                       | 0.67                                   | 1.000            |
| Fasten window tracking | 0                                      | 158                | 0                       | 0.79                                   | 0.977            |
|                      | 30                                     | 98                 | 0                       | 1.28                                   | 0.943            |
|                      | 60                                     | 56                 | 0                       | 2.23                                   | 0.801            |
|                      | 90                                     | 64                 | 0                       | 1.58                                   | 0.826            |
|                      | 120                                    | 58                 | 0                       | 2.16                                   | 0.774            |
|                      | 150                                    | 56                 | 0                       | 2.23                                   | 0.816            |
|                      | 180                                    | 187                | 0                       | 0.67                                   | 1.000            |

Jinghuan Wen received the BEng degree in electronic engineering from Tsinghua University, China in 2014, where he is currently pursuing the PhD degree. His research interests include 3D simulation, computer graphics, and optimization.

Xiaoqin Zhang received the BEng degree in electronic engineering from Tsinghua University, China in 2015, where he is currently pursuing the PhD degree. His research interests include 3D simulation, optimization, and multi-agent decision making.

Huimin Ma received the MS and PhD degrees in mechanical electronic engineering from Beijing Institute of Technology, China in 1998 and 2001, respectively. She is an associate professor in the Department of Electronic Engineering of Tsinghua University, and the director of 3D Image Simulation Lab. She worked as a visiting scholar in University of Pittsburgh in 2011. She is also the executive director and the vice secretary general of China Society of Image and Graphics. Her research and teaching interests include 3D object recognition and tracking, system modeling and simulation, and psychological base of image cognition.

[15] X. Jin, L. Ma, T. Wu, and J. Qian, Convergence analysis of the particle swarm optimization based on stochastic processes, Acta Automatica Sinica, vol. 33, no. 12, pp. 1263–1268, 2007.

[16] G. Su, H. Ma, and Y. Hou, A robust approach for anti-jamming target tracking in forward looking infrared imagery, in International Conference on Image and Graphics (ICIG), 2011, pp. 636–641.