JURNAL ELTEK
Vol. 24, No. 2, October 2026, pp. 60~69
p-ISSN: 1693 – 4024 | e-ISSN: 2355-0740
DOI: 10.33795/eltek.v24i2.9879

A color-based particle filter optimized using firefly and black hole algorithms for moving object tracking

Devira Anggi Maharani[1], Lenni Yulianti[2], Carmadi Machbub[3], Pranoto Hidaya Rusmin[4], Suci Aulia[5], Rizki Dwi Romadhona[6]
e-mail: [1]devira.anggi@polinema.ac.id, [2]lenni@itb.ac.id, [3]carmadi.machbub@itsb.ac.id, [4]pranoto.hr@itb.ac.id, [5]suciaulia@telkomuniversity.ac.id, [6]rizki.dwiramadhona@polinema.ac.id
[1,6]Department of Electrical Engineering, Politeknik Negeri Malang, Malang, Indonesia
[2,4] School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia
[3]Department of Informatics, Institut Teknologi Sains Bandung, Cikarang Pusat, Bekasi, Indonesia
[5]School of Applied Science, Telkom University, Bandung, Indonesia

Article Information

Article History

Received May 12, 2026
Revised July 7, 2026
Published October 30, 2026

***Keywords*:**

Tracker,
Black Hole,
Firefly,
Particle Filter,
PTZ Camera

ABSTRACT

Many researchers, particularly in the field of computer vision, have increasingly focused on moving object tracking. However, accurately tracking a moving object remains challenging, especially under conditions such as occlusion, illumination changes, abrupt motion, and variations in object posture. Although recent developments have enhanced the performance of visual object tracking methods, the task remains challenging under real-world conditions. Particle Filter (PF), a Monte Carlo-based estimation framework, has been extensively applied to visual object tracking due to its ability to handle nonlinear and non-Gaussian systems. However, conventional PF methods still suffer from particle degeneracy and sample impoverishment, particularly after repeated resampling processes. Although resampling can reduce degeneracy by removing particles with insignificant weights, it may also decrease particle diversity and degrade tracking performance. To address these limitations, this study introduces a Black Hole Optimization Particle Filter (BHOPF), in which the Black Hole Optimization algorithm is incorporated into the PF resampling process. The proposed approach updates particles through the Black Hole optimization mechanism, offering a simple computational structure with minimal parameter tuning. The integration of Black Hole Optimization is expected to improve particle diversity, increase the effective number of particles, reduce RMSE, and enhance the robustness of PF-based visual object tracking. The experimental results show that, using 30 particles and 5 iterations, BHOPF increased the effective number of particles to 26. In comparison, the Firefly Optimization Particle Filter (FAOPF) achieved 23 effective particles, while the conventional Particle Filter achieved only 16 effective particles. The proposed system achieved real-time operation through multithreading and pipeline-based processing, with a processing speed of up to 30 FPS.

Corresponding Author:
Devira Anggi Maharani,
Department of Electrical Engineering,
Politeknik Negeri Malang,
Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, East Java, Indonesia.
Email: devira.anggi@polinema.ac.id

1. INTRODUCTION

Tracking moving objects has become a major research focus, particularly in the field of computer vision. However, accurately tracking a moving object remains challenging, especially when the object undergoes occlusion, illumination changes, sudden movements, and pose variations. Although many studies have shown significant progress in recent years, estimating the state of a moving target in complex tracking scenarios remains difficult. In recent years, Monte Carlo-based approaches, particularly the Particle Filter (PF), have been widely applied to tracking problems and continue to attract considerable attention in visual tracking research. The PF algorithm and its variants have been extensively developed to handle nonlinear and non-Gaussian estimation problems [1][2] across various applications, including visual tracking[3][4][5], sensor fusion [6], speaker tracking [7], and tracking human movement [8][5][9][10][4]. Despite its successful implementation, particularly in visual tracking tasks, the conventional PF still suffers from two major limitations: particle degeneracy and sample impoverishment [11][12], Resampling is commonly employed to alleviate degeneracy by removing particles with insignificant weights; however, this process may reduce particle diversity and lead to sample impoverishment. When sample impoverishment occurs, the PF may lose its ability to adequately represent the posterior distribution, thereby reducing its accuracy in estimating the target state.

Several methods have been introduced to address these limitations. One approach employs the K-Means clustering algorithm by selecting a representative particle from each cluster as an important particle during the resampling stage [13]. Another study mitigates sample impoverishment by classifying particles based on their weights [14]. In addition, an adaptive fission Particle Filter has been proposed to enhance particle diversity through the application of a splitting factor [15]. The Auxiliary Particle Filter (APF) incorporates a particle index as an additional variable during the resampling process [16]. Although these methods have contributed to improving resampling strategies, sample impoverishment remains an unresolved challenge in PF-based tracking.

More recently, optimization algorithms have been increasingly developed and successfully applied to a wide range of optimization problems. Several algorithms, including Particle Swarm Optimization (PSO) (PSO) [17], Firefly algorithm [18][5], Nomadic People Optimizer[19], Global-best Local Neighbourhood oriented Particle Swarm Optimization (GbLN-PSO) [20] have been adopted to improve particle-based optimization by adjusting particle positions toward more optimal solutions. Inspired by these studies, this research proposes the integration of the Black Hole Optimization (BHO) algorithm into the resampling process of the Particle Filter. In the proposed method, particles are moved and updated according to the BHO mechanism. The Black Hole algorithm offers several advantages, including a simple computational structure, ease of implementation [21]–[24], and a parameterless optimization mechanism [25]. To address the need for a simpler and more parameter-efficient optimization strategy, this study integrates the Black Hole Optimization (BHO) algorithm into the Particle Filter (PF) framework. The proposed method aims to maintain particle diversity, increase the effective number of particles, and reduce tracking error in real-time visual object tracking. Its performance is evaluated using the effective number of particles (N_{eff})and the Root Mean Square Error (RMSE).

The main contributions of this research are threefold. First, this study proposes a real-time Black Hole Optimization Particle Filter to mitigate particle degeneracy and sample impoverishment by updating and eliminating particles based on the BHO strategy. Second, the proposed BHO-based Particle Filter is compared with the conventional Particle Filter and the Firefly Optimization Particle Filter in terms of Neff and RMSE. Third, the proposed method is implemented in real time using 30 particles and achieves a processing speed of more than 15 frames per second (FPS) through the implementation of multithreading and pipeline-based camera streaming.

2. RESEARCH METHOD

The Particle Filter (PF) algorithm is a recursive estimation technique based on the Bayesian filtering framework and is widely used in object tracking applications. It estimates the posterior probability distribution of the target state by integrating prior state information with the current observation. In this framework, the target dynamics are assumed to satisfy the first-order Markov property, meaning that the current state is influenced only by its preceding state. As stated in [2], PF provides a reliable numerical estimation approach for nonlinear system problems, including object tracking in video sequences [11]. The state transition model is expressed as:

$$\mathbf{x}_c = \mathrm{f}_c(\mathbf{x}_{c-1}, \mathbf{g}_c) \tag{1}$$

The corresponding stochastic measurement model is expressed as:

$$\boldsymbol{y}_c = \mathrm{h}_c(\mathbf{x}_c, \mathbf{v}_c) \tag{2}$$

where $\mathbf{x}_c$denotes the hidden target state at frame c, while $\mathbf{x}_{c-1}$represents the target state in the previous frame. The function $\mathrm{f_c}(\cdot)$defines the nonlinear state transition model, and $\mathbf{g}_c$represents the process noise. Meanwhile, $\mathbf{y}_c$denotes the measurement vector obtained at frame c, $\mathrm{h}_c(\cdot)$represents the nonlinear observation model, and $\mathbf{v}_c$indicates the measurement noise. Therefore, $\mathrm{f}_c(\cdot)$describes the relationship between the previous and current target states, whereas $\mathrm{h}_c(\cdot)$maps the current state into the measurement space.

The posterior distribution $p(\mathbf{x}_c \mid \mathbf{y}_{1:c})$of the target state is estimated recursively through Bayesian filtering, which consists of prediction and update stages. In the prediction stage, the prior distribution of the current state is obtained by propagating the posterior distribution from the previous frame:

$$p(\mathbf{x}_c|\boldsymbol{y}_{1:c-1}) = \int p(\mathbf{x}_c|\mathbf{x}_{c-1})p(\mathbf{x}_{c-1}|\boldsymbol{y}_{1:c-1})d\mathbf{x}_{c-1} \quad (3)$$

$$p(\mathbf{x}_c|\boldsymbol{y}_{1:c}) = \frac{p(\mathbf{y}_c|\mathbf{x}_c)p(\mathbf{x}_c|\mathbf{y}_{1:c-1})}{p(\mathbf{y}_c|\mathbf{y}_{1:c-1})}$$

where the normalization term is given by:

$$p(\boldsymbol{y}_c|\boldsymbol{y}_{1:c-1}) = \int p(\boldsymbol{y}_c|\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{y}_{1:c-1})d\boldsymbol{x}_c \quad (4)$$

The recursive Bayesian filtering process provides the posterior density $p(\mathbf{x}_c \mid \mathbf{y}_{1:c})$, which is used to estimate the current target state. Nevertheless, in nonlinear and non-Gaussian systems, obtaining an exact analytical solution is often impractical [11]. Therefore, the Particle Filter approximates the posterior distribution using a finite number of weighted particles generated by Monte Carlo simulation. The posterior distribution can be represented as $\{\mathbf{x}_c^i, \mathbf{w}_c^i\}_{i=1}^N$ where $N$is the total number of particles, $\mathbf{x}_c^i$represents the i-th possible target state at frame c, and w_c^idenotes its corresponding weight. Each particle corresponds to a potential target state, and its weight represents the probability contribution of that particle with respect to the current observation.

In general, the Particle Filter consists of two essential processes: state estimation and resampling. The state estimation process determines the estimated target position based on the weighted particle set, while the resampling process selects and regenerates particles according to their weights to maintain an effective approximation of the target state distribution. However, repeated resampling may reduce particle diversity and lead to sample impoverishment, which can degrade the tracking performance of the PF.

2.1. Sequential Importance Sampling

SIS is introduced to address the difficulty of directly sampling from the posterior distribution $p(\mathbf{x}_{0:c} \mid \mathbf{y}_{1:c})$ is generally difficult, particularly in nonlinear and non-Gaussian tracking problems. Instead of drawing samples directly from the true posterior distribution, SIS generates particles from an importance density function and assigns appropriate weights to approximate the posterior distribution. Through this mechanism, the posterior density can be represented by a finite set of weighted particles that are updated sequentially as new observations become available.

Assuming that the importance density function can be factorized recursively, it can be written as:

$$(\mathbf{x}_{0:c}|\mathbf{y}_{1:c}) = q(\mathbf{x}_c|\mathbf{x}_{0:c-1}, \mathbf{y}_{1:c})q(\mathbf{x}_{0:c-1}|\mathbf{y}_{1:c-1}) \quad (5)$$

The posterior distribution of the complete state trajectory can be recursively derived using Bayes' theorem as follows:

$$p(\mathbf{x}_{0:c}|\mathbf{y}_{1:c}) = \frac{p(\mathbf{x}_{0:c}, \mathbf{y}_c|\mathbf{y}_{1:c-1})}{p(\mathbf{y}_c|\mathbf{y}_{1:c-1})} \quad (6)$$

$$= \frac{p(\mathbf{y}_c|x_{0:c}|\mathbf{y}_{1:c-1})p(\mathbf{x}_{0:c}|\mathbf{y}_{1:c-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:c-1})}$$

$$= \frac{p(\mathbf{y}_c|\mathbf{x}_{0:c}, \mathbf{y}_{1:c-1})\mathbf{p}(\mathbf{x}_c|\mathbf{x}_{0:c-1}, \mathbf{y}_{1:c-1})}{p(\mathbf{x}_{0:c-1}|\mathbf{y}_{1:c-1})}$$

$$= \frac{p(\mathbf{x}_{0:c-1}|\mathbf{y}_{1:c-1})}{p(\mathbf{z}_c|\mathbf{y}_{1:c-1})}$$

Based on Markov's law, the above formula can be written as:

$$= \frac{p(\boldsymbol{y}_c|\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{x}_{:c-1})\, p(\boldsymbol{x}_{0:c-1}|\boldsymbol{y}_{1:c-1})}{p(\boldsymbol{y}_c|\boldsymbol{y}_{1:c-1})} \quad (7)$$

So, the weight $\mathbf{w}_c^i$ can be expressed recursively as follows:

$$\boldsymbol{w}_c^i \propto \frac{p(\boldsymbol{x}_{0:c}^i|\boldsymbol{y}_{1:c})}{q(\boldsymbol{x}_{0:c}^i|\boldsymbol{y}_{1:c})} \quad (8)$$

$$\boldsymbol{w}_{c-1}^i \propto \frac{p(\boldsymbol{y}_c|\boldsymbol{x}_c^i)p(\boldsymbol{x}_c^i|\boldsymbol{x}_{c-1}^i)}{q(\boldsymbol{x}_c^i|\boldsymbol{x}_{0:c-1}^i, \boldsymbol{y}_{1:c})}$$

In this case, only posterior filtered density is required $p(\mathbf{x}_c|\mathbf{y}_{1:c})$, then $q(\mathbf{x}_c|\mathbf{x}_{0:c-1}, \mathbf{y}_{1:c}) = q(\mathbf{x}_c|\mathbf{x}_{c-1}, \mathbf{y}_c)$:

$$\boldsymbol{w}_c^i \propto \boldsymbol{w}_{c-1}^i \frac{p(\boldsymbol{y}_c|\boldsymbol{x}_c^i)p(\boldsymbol{x}_c^i|\boldsymbol{x}_{c-1}^i)}{q(\boldsymbol{x}_c^i|\boldsymbol{x}_{c-1}^i, y_c)} \quad (9)$$

Thus, the posterior approximation can be written as:

$$p(\mathbf{x}_c \mid \mathbf{y}_{1:c}) \approx \sum_{i=1}^{N} w_c^{i,new}\, \delta(\mathbf{x}_c - \mathbf{x}_c^{i,new}) \quad (10)$$

The conventional posterior approximation is modified by replacing the original particle position $\mathbf{x}_c^i$with the Black Hole-updated particle position $\mathbf{x}_c^{i,new}$. After the likelihood of each particle is evaluated, the particle with the highest weight is selected as the black hole, and the remaining particles are moved toward this particle before resampling. Therefore, the posterior distribution is approximated using the updated particle set $\{\mathbf{x}_c^{i,new}, w_c^{i,new}\}_{i=1}^N$, allowing the particle population to concentrate around more promising target regions while maintaining diversity through the Black Hole replacement mechanism.

2.2. Firefly Optimization Algorithm Based Particle Filter (FOAPF)

The Firefly Algorithm (FA) is a bio-inspired metaheuristic optimization technique that emulates the flashing behavior of fireflies as a natural mechanism for attraction and communication [26], In natural environments, fireflies generate short, rhythmic flashes, usually from the lower part of their bodies, and each species has its own characteristic flashing pattern. These light signals are mainly used to attract potential mates and capture prey. The brightness perceived by another firefly decreases as the distance from the light source increases. This relationship can be mathematically described as $I = \frac{K}{r^2}$. where Idenotes the perceived light intensity and $r$represents the distance from the light source. This equation indicates that a greater distance results in lower perceived brightness.

These natural behaviors were adopted by Xin-She Yang [27] to develop the Firefly Algorithm as an optimization technique. For each incoming video frame, the target ROI was represented by a reference color histogram, and each particle was treated as a candidate target location. The histogram of each particle region was compared with the reference target histogram to determine its likelihood or brightness value. Particles with lower brightness were then moved toward brighter particles according to the firefly update mechanism before the resampling stage. To maintain real-time performance, the number of particles and Firefly iterations was limited, and the tracking process was executed within the multithreaded pipeline architecture. In this implementation, video streaming was handled by the main thread, while the tracking threads performed likelihood evaluation, firefly-based particle refinement, state estimation, and resampling.

2.3. Our Strategy

The Black Hole Algorithm is a population-driven metaheuristic optimization approach inspired by the strong gravitational characteristics of black holes [28][29]. In astrophysics, a black hole is known as a region in space with an extremely dense mass concentration, generating a gravitational field so intense that any object entering its boundary cannot escape. In the optimization framework, this phenomenon is represented by assigning each candidate solution as a star, whereas the candidate with the best fitness value is selected as the black hole.

At the beginning of the optimization process, a population of candidate solutions is randomly generated within the search space. Each candidate solution is evaluated using a predefined fitness function. The candidate with the best fitness value is selected as the black hole, whereas the remaining candidates are regarded as stars. During each iteration, all stars are attracted toward the black hole, allowing the population to exploit the region around the current best solution. At the same time, the replacement of absorbed stars with newly generated random candidates supports the exploration of other regions in the search space.

Let the position of the i-th star at iteration c be denoted by $\mathbf{x}_i^c$, and let the position of the current black hole be represented by $\mathbf{x}_{BH}^c$. The movement direction of the i-th star is determined by the vector difference between the black hole and the star position, expressed as:

$$\Delta\mathbf{x}_i^c = \mathbf{x}_{BH}^c - \mathbf{x}_i^c \quad (11)$$

To control the step size of the movement, a random coefficient $\alpha$is introduced, where $\alpha$is uniformly distributed in the interval $[0, 1]$. The displacement of the i-th star toward the black hole can therefore be written as:

$$\mathbf{s}_i^c = \alpha(\mathbf{x}_{BH}^c - \mathbf{x}_i^c) \quad (12)$$

Thus, the updated position of the i-th star at the next iteration is formulated as:

$$\mathbf{x}_i^{c+1} = \mathbf{x}_i^c + \alpha(\mathbf{x}_{BH}^c - \mathbf{x}_i^c) \quad (13)$$

where $\mathbf{x}_i^{c+1}$denotes the updated position of the i-th star, $\mathbf{x}_i^c$is its current position, $\mathbf{x}_{BH}^c$is the current black hole position, and $\alpha \in [0,1]$is a random scaling factor. Equation (13) indicates that each star moves partially toward the black hole, with the movement distance controlled by the random coefficient. This stochastic movement helps prevent all stars from converging to the same point too rapidly.

After the particle movement is completed, the fitness value of each updated particle is recalculated based on its similarity to the target model. If an updated particle achieves a better fitness value than the current black hole, it replaces the black hole and becomes the new center of attraction. The previous black hole is then treated as an ordinary particle. This role-exchange mechanism enables the tracking process to continuously follow the most reliable target candidate in the current frame. To maintain particle diversity, the event horizon mechanism is applied. Particles that move too close to the black hole and cross the event horizon are considered absorbed and are replaced by newly generated particles within the search space. In the real-time object tracking system, this mechanism is executed for each incoming video frame before the resampling stage. The particles are first evaluated between their color histograms and the reference target histogram. The best particle is selected as the black hole, the remaining particles are updated toward it, and less representative particles are replaced when necessary. By integrating this Black Hole-based update into the Particle Filter loop, the proposed method improves particle distribution, reduces degeneracy and sample impoverishment, and guides particles toward more accurate target states while maintaining real-time tracking performance.

3. RESULTS AND DISCUSSION

The experiment was carried out in an indoor environment with an illumination level of 1200 lux as in Figure 1. The PTZ camera initially captured video frames with a resolution of 1920×540 pixels under an illumination level of 2000 lux. To minimize computational complexity and support real-time processing, each frame was resized or cropped to 640×360 pixels. This frame size was considered adequate for the proposed moving object tracking system while maintaining sufficient visual information for target localization. The camera was configured to operate at a maximum rate of 30 FPS. Four datasets were used in the evaluation, and the performance of the baseline Particle Filter was measured based on the distance between the estimated object position and the corresponding ground-truth position, as reported in Table I.

Table I. Distance between ground-truth and predicted object locations using the baseline Particle Filter algorithm

Particles	Dataset 1	Dataset 2	Dataset 3	Dataset 4
10	30.55	26.83	19.55	59.26
20	19.84	24.53	12.98	12.29
30	8.65	15.17	8.40	11.75
60	6.09	13.41	7.37	11.34
90	6.31	12.38	7.35	10.66

Based on Table I, increasing the number of particles generally improves the tracking accuracy of the baseline Particle Filter, as indicated by the reduction distance between the predicted object location and the ground truth. A substantial decrease in error can be observed when the number of particles is increased from 10 to 30, particularly in Dataset 1, Dataset 3, and Dataset 4. Further improvement is achieved when 60 and 90 particles are used, although the reduction becomes less significant at higher particle numbers. The baseline PF obtained its best overall performance when 90 particles were used, with distance values of 6.31, 12.38, 7.35, and 10.66 for Datasets 1, 2, 3, and 4, respectively. Although Dataset 1 produced a slightly lower error with 60 particles, the 90-particle configuration provided more consistent performance across all datasets. However, increasing the number of particles also increases the computational burden. Therefore, the maximum number of particles in this study was limited to 90 in order to maintain real-time processing capability. Under this configuration, the system was still able to operate in real time.

When 30 particles were used, the baseline PF produced an error of 8.65 in Dataset 1. This result suggests that the tracking error may still be further reduced by incorporating optimization-based Particle Filter variants, such as the Firefly Particle Filter or the proposed Black Hole Optimization Particle Filter. These optimization-based approaches are expected to improve particle distribution, reduce particle degeneracy, and enhance tracking accuracy while maintaining real-time performance.

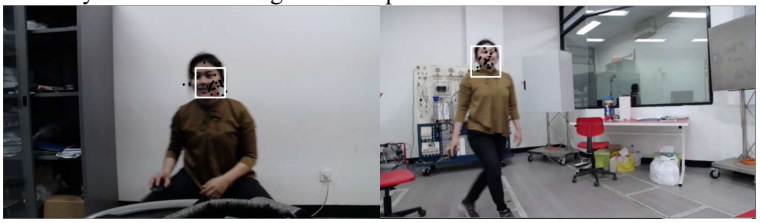


Figure 1. Moving Object Tracking Results Using PF Method on our Datasets

3.1. Implementation Results of Moving Object Tracking Using the Firefly Particle Filter

Four datasets were tested using the Firefly Optimization Algorithm-based Particle Filter (FOAPF). The comparison between the conventional Particle Filter and FOAPF was conducted using a color-based likelihood function. For each particle, a color histogram was extracted and compared with the reference histogram of the tracked target. The resulting histogram distance was used to evaluate the similarity between the particle and the target model.

In FOAPF, the particle position $\mathbf{x}_{i,c}$represents the c-th component of the i-th firefly. The movement of a particle toward a brighter firefly follows the Firefly Algorithm mechanism, where attractiveness is determined based on the similarity between the particle histogram and the initial target ROI histogram. The attractiveness function $\beta(r)$is updated according to the distance and similarity between particles. The fitness value is computed between the brightest firefly and the surrounding particles.

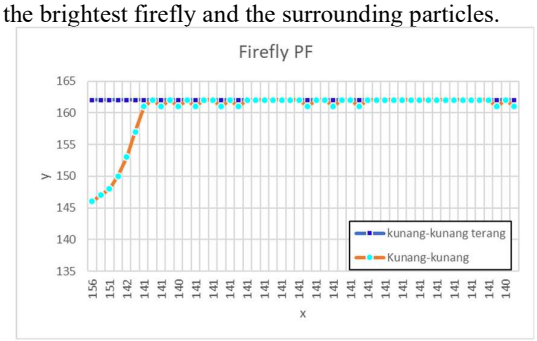


Figure 2. Displacement of Fireflies Toward the Brightest Firefly

Figure 2 illustrates the movement of one particle over 50 iterations. The particle gradually approaches the brightest firefly as the iteration increases. However, to support real-time object tracking, the number of iterations $m$was limited to five. This setting was sufficient to guide the particles toward the target state while keeping the computational cost low.

Table II. The time needed for the optimization process uses the Firefly Algorithm

Iteration (30 particles)	2	5	20	30	50
Compute time (ms)	16	16	31	46	78
FPS	21	21	16	13	10

The computation time and FPS were then evaluated to determine the real-time capability of FOAPF. As shown in Table II, increasing the iteration number increases computation time and reduces FPS. With 30 particles, two and five iterations required 16 ms and achieved 21 FPS. However, the frame rate decreased to 16 FPS, 13 FPS, and 10 FPS when the iterations were increased to 20, 30, and 50, respectively. Therefore, five iterations were selected as the maximum configuration for maintaining real-time performance on the NVIDIA GeForce 940MX. Within this configuration, the particle movement of approximately ± 5 pixels toward the brightest firefly contributed to improved tracking accuracy while preserving real-time operation.

3.2. Implementation Results of Moving Object Tracking Using the Black Hole Particle Filter

The Black Hole Particle Filter (BHOPF) was implemented using an initial population of 30 particles to support real-time operation. In the proposed framework, each particle was represented as a star, while the particle with the highest histogram similarity to the target model was selected as the black hole. The maximum number of iterations was limited to five to balance tracking accuracy and computational efficiency.

Figure 3 shows the displacement of particles during the Black Hole optimization process. The distance $d_c$between each star and the black hole reflects the deviation of a particle from the most promising target state. As the stars move toward the black hole, the particle distribution becomes more concentrated around the estimated target location. The results show that the particles were able to approach the black hole within four iterations, with a displacement of approximately ± 10 pixels. This demonstrates that the BHOPF can refine particle positions effectively with only a small number of iterations.

Table III. Computation Time Required for the Optimization Process Using the Black Hole Algorithm

Iteration (30 particles)	2	5	20	30	50
Compute time (ms)	15	15	31	31	46
FPS	21	21	21	16	13

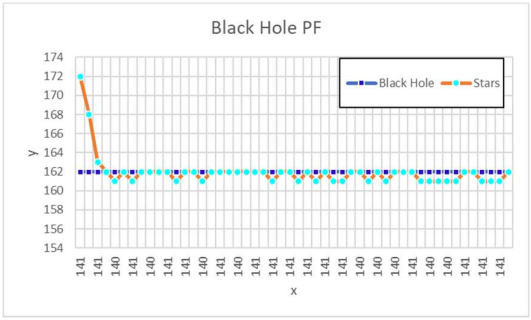


Figure 3. Star Displacement Toward the Black Hole Position

The computation time was evaluated using 30 particles with different iteration numbers, as shown in Table III. The results indicate that the proposed BHOPF maintained real-time performance up to 20 iterations, achieving 21 FPS. Increasing the iteration number to 30 and 50 reduced the frame rate to 16 FPS and 13 FPS, respectively. Therefore, the use of five iterations was considered sufficient for real-time implementation while still allowing particles to move closer to the target state.

The RMSE comparison shows that the optimization-based Particle Filter methods improved tracking accuracy compared with the conventional PF. The lowest RMSE was obtained on the people-playing-tennis dataset with 90 particles, where BHOPF achieved an RMSE of 3.18 and FOAPF achieved an RMSE of 4.38. These findings suggest that the Black Hole optimization strategy provides a more effective particle refinement mechanism, leading to improved tracking accuracy while maintaining real-time performance. The RMSE comparison across the four datasets using different particle configurations and tracking algorithms is summarized in Table IV.

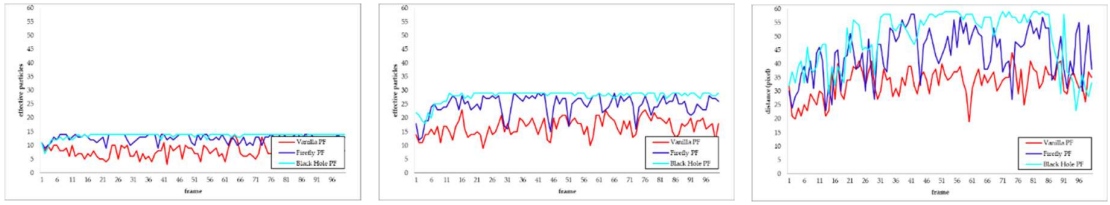
Table IV. RMSE Values of Each Algorithm on Four Datasets

Number of Particles	Dataset 1			Dataset 2			Dataset 3			Dataset 4		
	PF	FOAPF	BHO PF	PF	FOAPF	BHO PF	PF	FOAPF	BHO PF	PF	FOAPF	BHO PF
10	30.55	19.78	17.39	26.83	22.25	21.27	19.55	13.99	12.96	59.26	12.70	11.67
20	19.84	18.33	16.64	24.53	20.93	19.53	12.98	10.40	8.56	12.29	9.17	8.57
30	8.65	6.87	5.38	15.17	14.39	13.20	8.40	7.85	6.63	11.75	8.90	7.90
60	6.09	4.67	3.33	13.41	12.67	10.79	7.37	6.35	6.05	11.34	8.81	7.82
90	6.31	4.38	**3.18**	12.38	11.33	**10.46**	7.35	6.24	**5.43**	10.66	8.02	**7.40**

As presented in Table IV, the best tracking performance was achieved by the Black Hole Particle Filter using 90 particles, which produced the lowest RMSE value among the evaluated methods. The Black Hole optimization mechanism improves particle movement by guiding particles toward candidates with smaller $d_c$values, indicating a closer correspondence to the target model. This performance improvement is mainly attributed to the exploration mechanism of the Black Hole algorithm. When a candidate solution, or star, crosses the event horizon and is absorbed by the black hole, it is replaced by a newly generated random star within the search space. This replacement process helps preserve particle diversity and reduces the risk of premature convergence to a local minimum.

In contrast, the conventional Particle Filter relies on a recursive weighting and resampling process that may gradually reduce particle diversity. During resampling, high-weight particles are more likely to be retained and replicated, while low-weight particles are progressively eliminated during resampling. Although this strategy is effective in reducing particle degeneracy, it can also lead to sample impoverishment. Consequently, the new particle set may contain less diverse information, which limits its ability to accurately represent the actual target state.

To address these limitations, this study incorporates Firefly and Black Hole optimization algorithms into the Particle Filter framework. These optimization mechanisms are introduced to refine particle positions, improve particle diversity, and enhance the representation of the posterior target distribution. The effectiveness of the proposed methods was further analyzed using the effective number of particles, denoted as N_{eff}. The obtained N_{eff} values are illustrated in Figure 4.

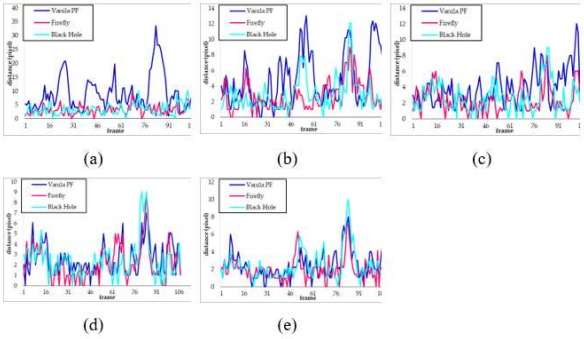


(a) (b) (c)

Figure 4(a). Comparison of $\widehat{\boldsymbol{N}}_{\boldsymbol{eff}}$ Values Using 15 Particles, (b). 30 Particles, and (c). 90 Particles

Dataset 2 was selected for the N_{eff} evaluation to compare the ability of each algorithm to maintain particle effectiveness during the tracking process. As shown in Figure 4a-c, the optimization-based Particle Filter methods produced higher N_{eff}values than the conventional Particle Filter. This result indicates that the proposed methods are more effective in maintaining a well-distributed particle set and reducing particle degeneration. A higher N_{eff}also suggests that more particles contribute meaningfully to the state estimation process, allowing the particle population to better approximate the actual target position across different particle configurations. When N_{eff} remains high, the particle set maintains sufficient diversity to represent several possible target states, allowing the tracker to adapt more smoothly across consecutive frames. However, when N_{eff} decreases, the particle distribution becomes less diverse, making the tracker more inaccurate likelihood estimation. Therefore, the higher N_{eff}values obtained by the Firefly Particle Filter and Black Hole Particle Filter indicate that the optimization mechanisms help preserve particle diversity and produce more stable tracking performance compared with the Particle Filter.

Furthermore, the RMSE trend for the people-playing-tennis dataset was analyzed using several particle distributions. The results are presented in Figure 5.



(a) (b) (c)

(d) (e)

Figure 5. RMSE Results of Dataset 1 with Particle Distributions of (a) 30, (b) 60, (c) 90, (d) 180, and (e) 500 Particles

Figure 5 shows the RMSE behavior of each algorithm under different particle population sizes. The comparison demonstrates that optimization-based Particle Filter methods provide better tracking accuracy than the conventional Particle Filter, particularly when a sufficient number of particles is used. Among the evaluated approaches, the Black Hole Particle Filter shows the strongest ability to reduce tracking error, indicating that its particle replacement and attraction mechanisms can effectively improve target localization accuracy while maintaining particle diversity.

3.3. Overall Real-Time Scenario Test Results

In a conventional single-threaded implementation, video acquisition, frame buffering, preprocessing, and tracking computation are executed sequentially. This structure may create a processing bottleneck because the next frame cannot be processed until the previous tracking computation is completed. As a result, frame delay may increase, particularly when the Particle Filter requires additional optimization steps. To overcome this limitation, the proposed system separates the video streaming and tracking computation into different execution threads. Three threads were implemented: the main thread, thread 1, and thread 2. The main thread handles video acquisition and streaming through a pipeline mechanism, enabling continuous frame capture while maintaining a minimum streaming rate of 60 FPS. Meanwhile, thread 1 and thread 2 execute the Particle Filter-based tracking process. This separation allows frame acquisition and tracking computation to run concurrently rather than sequentially. From a real-time processing perspective, this architecture reduces idle time between frame capture and tracking execution. While one thread processes the current frame, the pipeline can continue preparing the next frame for subsequent analysis. This is particularly important for optimization-based Particle Filter methods, such as Firefly PF and Black Hole PF, because additional particle movement

steps may increase computation time. The use of multithreading also improves system stability under real-time conditions. Since the streaming process is isolated from the tracking computation, temporary increases in tracking workload do not directly interrupt video acquisition. This helps maintain smoother frame delivery and prevents significant frame drops. In addition, the pipeline mechanism ensures that the tracking module receives updated frames continuously, which is essential for maintaining target localization accuracy in dynamic scenes.

4. CONCLUSION

This study developed an optimization-based object tracking method by integrating the Particle Filter algorithm with two metaheuristic optimization techniques, namely the Firefly Algorithm and the Black Hole Algorithm. The target position was estimated by comparing the reference target histogram. The experimental results demonstrate that the proposed optimization-based Particle Filter methods improve tracking accuracy compared with the conventional Particle Filter. The integration of optimization algorithms into the standard Particle Filter framework also increased the effective number of particles. Using 30 particles, the Black Hole Particle Filter achieved 26 effective particles, while the Firefly Particle Filter achieved 23 effective particles. In contrast, the conventional Particle Filter without optimization produced only 16 effective particles. These results indicate that the proposed optimization mechanisms can better preserve particle diversity, reduce sample impoverishment, and improve the representation of the target state during the tracking process.

5. FUTURE WORKS

Future work will extend the proposed Black Hole Particle Filter to more challenging tracking scenarios and explore adaptive particle allocation or hybrid optimization strategies to improve robustness while preserving real-time performance. Further implementation on embedded or edge-computing platforms will also be considered to evaluate its suitability for practical real-time systems.

6. REFERENCES

[1] H. Fang, N. Tian, Y. Wang, M. Zhou, and M. A. Haile, 'Nonlinear Bayesian estimation: From Kalman filtering to a broader horizon', *IEEE/CAA J. Autom. Sin.*, vol. 5, no. 2, pp. 401–417, 2018, doi: 10.1109/JAS.2017.7510808.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, 'A tutorial on particle filters for online nonlinear/nongaussian bayesian tracking', in *Bayesian Bounds for Parameter Estimation and Nonlinear Filtering/Tracking*, 2007, vol. 50, no. 2, pp. 723–737. doi: 10.1109/9780470544198.ch73.

[3] M. H. Abdelpakey, M. S. Shehata, M. M. Mohamed, and M. Gong, 'Adaptive framework for robust visual tracking', *IEEE Access*, vol. 6, no. c, pp. 55273–55283, 2018, doi: 10.1109/ACCESS.2018.2871659.

[4] D. A. Maharani, C. Machbub, L. Yulianti, and P. H. Rusmin, 'Particle filter based single shot multibox detector for human moving prediction', in *2020 IEEE 10th International Conference on System Engineering and Technology, ICSET 2020 - Proceedings*, 2020, pp. 7–11. doi: 10.1109/ICSET51301.2020.9265355.

[5] D. A. Maharani, C. Machbub, L. Yulianti, and P. H. Rusmin, 'Real-time Human Tracking System using Histogram Intersection Distance in Firefly Optimization Based Particle Filter', *Int. J. Electr. Eng. Informatics*, vol. 13, no. 4, pp. 853–872, 2021, doi: 10.15676/ijeei.2021.13.4.7.

[6] H. Song, W. Choi, and H. Kim, 'Robust Vision-Based Relative-Localization Approach Using an RGB-Depth Camera and LiDAR Sensor Fusion', *IEEE Trans. Ind. Electron.*, vol. 63, no. 6, pp. 3725–3736, 2016.

[7] Q. Zhang, Z. Chen, and F. Yin, 'Speaker Tracking Based on Distributed Particle Filter in Distributed Microphone Networks', *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 47, no. 9, pp. 2433–2443, 2017, doi: 10.1109/TSMC.2016.2523939.

[8] L. Lin and M. Zhu, 'Efficient tracking of moving target based on an improved fast differential evolution algorithm', *IEEE Access*, vol. 6, pp. 6820–6828, 2018, doi: 10.1109/ACCESS.2018.2793298.

[9] D. A. Maharani, C. MacHbub, P. H. Rusmin, and L. Yulianti, 'Real-Time Human Tracking Using Multi-Features Visual With CNN-LSTM and Q-Learning', *IEEE Access*, vol. 12, pp. 13233–13247, 2024, doi: 10.1109/ACCESS.2024.3355785.

[10] D. A. Maharani, C. Machbub, P. H. Rusmin, and L. Yulianti, 'Feature Fusion with Deep Neural Network in Kernelized Correlation Filters Tracker', in *2021 IEEE 11th International Conference on System Engineering and Technology (ICSET)*, 2021, pp. 363–367. doi: 10.1109/icset53708.2021.9612567.

[11] P. Chiranjeevi and S. Sengupta, 'Rough-Set-Theoretic Fuzzy Cues-Based Object Tracking under Improved Particle Filter Framework', *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 3, pp. 695–707, 2016, doi:

10.1109/TFUZZ.2015.2471811.
[12] X. Qiang, Y. Zhu, and R. Xue, 'SVRPF: An Improved Particle Filter for a Nonlinear/Non-Gaussian Environment', *IEEE Access*, vol. 7, pp. 151638–151651, 2019, doi: 10.1109/ACCESS.2019.2947540.
[13] W. Yang, L. Song, C. A. T. Tee, Y. Zheng, and Y. Liu, 'Unsupervised learning grouping-based resampling for particle filters', *IEEE Access*, vol. 7, pp. 127265–127275, 2019, doi: 10.1109/ACCESS.2019.2937586.
[14] M. S. Sharifian, A. Rahimi, and N. Pariz, 'Classifying the weights of particle filters in nonlinear systems', *Commun. Nonlinear Sci. Numer. Simul.*, vol. 31, no. 1–3, pp. 69–75, 2016, doi: 10.1016/j.cnsns.2015.05.021.
[15] X. Han, H. Lin, Y. Li, H. Ma, and X. Zhao, 'Adaptive Fission Particle Filter for Seismic Random Noise Attenuation', *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 9, pp. 1918–1922, 2015, doi: 10.1109/LGRS.2015.2438229.
[16] M. S. Haque, S. Choi, and J. Baek, 'Auxiliary particle filtering-based estimation of remaining useful life of IGBT', *IEEE Trans. Ind. Electron.*, vol. 65, no. 3, pp. 2693–2703, 2018, doi: 10.1109/TIE.2017.2740856.
[17] J. Zhao and Z. Li, 'Particle filter based on Particle Swarm Optimization resampling for vision tracking', *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8910–8914, 2010, doi: 10.1016/j.eswa.2010.05.086.
[18] M. L. Gao, L. L. Li, X. M. Sun, L. J. Yin, H. T. Li, and D. S. Luo, 'Firefly algorithm (FA) based particle filter method for visual tracking', *Optik (Stuttg).*, vol. 126, no. 18, pp. 1705–1711, 2015, doi: 10.1016/j.ijleo.2015.05.028.
[19] S. Q. Salih and A. R. A. Alsewari, 'A new algorithm for normal and large-scale optimization problems: Nomadic People Optimizer', *Neural Comput. Appl.*, vol. 32, no. 14, pp. 10359–10386, 2020, doi: 10.1007/s00521-019-04575-1.
[20] Z. Musa, M. Z. Salleh, R. A. Bakar, and J. Watada, 'GbLN-PSO and model-based particle filter approach for tracking human movements in large view cases', *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 8, pp. 1433–1446, 2016, doi: 10.1109/TCSVT.2015.2433172.
[21] A. Harb, H. Kassem, and K. Ghorayeb, 'Black hole particle swarm optimization for well placement optimization', *Comput. Geosci.*, 2019, doi: 10.1007/s10596-019-09887-8.
[22] S. Yaghoobi and H. Mojallali, 'Modified Black Hole Algorithm with Genetic Operators', *Int. J. Comput. Intell. Syst.*, vol. 9, no. 4, pp. 652–665, 2016, doi: 10.1080/18756891.2016.1204114.
[23] H. A. Abdulwahab, A. Noraziah, A. A. Alsewari, and S. Q. Salih, 'An Enhanced Version of Black Hole Algorithm via Levy Flight for Optimization and Data Clustering Problems', *IEEE Access*, vol. 7, pp. 142085–142096, 2019, doi: 10.1109/ACCESS.2019.2937021.
[24] H. R. E. H. Bouchekara, 'Optimal power flow using black-hole-based optimization approach', *Appl. Soft Comput. J.*, vol. 24, pp. 879–888, 2014, doi: 10.1016/j.asoc.2014.08.056.
[25] H. R. E. H. Bouchekara, 'Optimal design of electromagnetic devices using a black-Hole-Based optimization technique', *IEEE Trans. Magn.*, vol. 49, no. 12, pp. 5709–5714, 2013, doi: 10.1109/TMAG.2013.2277694.
[26] Suyanto, *Swarm Intelligence*. 2020. doi: 10.1201/9780429028618.
[27] X. S. Yang, 'Firefly algorithm, stochastic test functions and design optimization', *Int. J. Bio-Inspired Comput.*, vol. 2, no. 2, pp. 78–84, 2010, doi: 10.1504/IJBIC.2010.032124.
[28] A. Hatamlou, 'Black hole: A new heuristic optimization approach for data clustering', *Inf. Sci. (Ny).*, vol. 222, pp. 175–184, 2013, doi: 10.1016/j.ins.2012.08.023.
[29] S. Kumar, D. Datta, S. Kumar Singh, A. T. Azar, and S. Vaidyanathan, 'Black hole algorithm and its applications', *Stud. Comput. Intell.*, vol. 575, pp. 147–170, 2015, doi: 10.1007/978-3-319-11017-2_7.