Real-time Human Tracking System using Histogram Intersection Distance in Firefly Optimization Based Particle Filter

Devira Anggi Maharani, Carmadi Machbub, Lenni Yulianti, and Pranoto Hidaya Rusmin

School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Indonesia
deviraanggi@students.itb.ac.id, carmadi@lskk.ee.itb.ac.id, lenni@lskk.ee.itb.ac.id, pranoto@lskk.ee.itb.ac.id

Abstract: Real-time human tracking in a video have numerous applications. For security and surveillance application, the tracking system with PTZ (Pan, Tilt, and Zoom) camera is expected to track an object correctly regardless of the object orientation. Numerous studies reported that Particle Filter (PF) is reliable for color object tracking. However, the PF algorithm still suffers from impoverishment and degeneration in the resampling process. These problems can be resolved by combining the PF with Firefly Optimization (FO) in the resampling process. This research proposes the use of Histogram Intersection distance to build a likelihood function in PF to achieve real-time implementation. The Firefly Optimization Algorithm-based Particle Filter (FOAPF) with Histogram Intersection distance was compared to FOAPF with Bhattacharyya distance, resulting in lower RMSE (Root Mean Square Error) in tracking TB datasets. The result shows that when the Histogram Intersection distance was implemented, a faster average time of 1.8 ms was achieved than 1.9 ms when using Bhattacharyya distance. It shows the time result slightly different. The FOAPF with Histogram Intersection distance results in the TB datasets perform a low RMSE of 4.96 and 12.07, and private datasets show a low RMSE of 16.92 and 8.80, with the real-time implementation of 30 FPS and 50 particles. The comparison presents the successful implementation of the proposed method as a tracker to enhance human movement tracking with real-time implementation.

Keywords: Particle Filter, Histogram Intersection Distance, Firefly Algorithm, Real-time Human Tracking

1. Introduction

The use of a PTZ camera in surveillance and security systems is increasing in popularity. This PTZ camera has a limited field of view (FOV) when it zooms into an object to generate a high frame per second (FPS), which becomes very important. The real-time implementation depends on the FPS result, and we know that the visual-based moving object tracking system became a research topic deserving more exploration. Nowadays, many popular tracking methods have been developed, such as [1][2], the recursive filter that measures the state of each time step to determine the location of a moving object. Even recursive filters use sequentially processed data rather than data stored in a data set. One approach for resolving this issue is the use of PF [3]. Numerous studies reported that PF is reliable for color tracking [4]–[6]. The Particle Filter variation of the Kalman Filter [7] approach [3] deals with nonlinear problems and non-gaussian distributions [5]. The use of color features has been reported by several studies [8]–[10]. The color feature is more commonly employed as a target observation model because it is faster to calculate and resistant to target distortion, rotation, and partial occlusion. As an example, a study mentioned in a 2002 lecture note [11] established a target distribution with HSV (Hue, Saturation, Value) color space for multitarget tracking. However, the computational process of PF is high. High accuracy in PF can be accomplished by increasing the number of particles, which causes high-cost computational activity[12]. Therefore, there is an opportunity for improvement of the PF in using less particle distribution.

Despite the success of the application, especially in visual tracking fields, the PF algorithm still suffers from degeneracy and impoverishment issues [13]–[15]. Where resampling is intended to solve the problem of degeneracy, however, it also causes sample impoverishment. When impoverishment of sample appears, it is very dominant on the PF algorithm to provide the

Received: August 17th, 2020. Accepted: October 13rd, 2021
DOI: 10.15676/ijeei.2021.13.4.7

correct target state information. In the resampling phase, the K-Means machine learning clustering algorithm picks one particle from each cluster as the most important particle. [16]. This study [17] improved their samples by categorizing the particles according to their weights. Using adaptive fission PF and the solution factor, this study [18] increased particle diversity, the Auxiliary Particle Filter (APF) introduces resampling with the particle index as an additional variable [19]. The resampling procedure was the subject of these studies. However, the issue of sample degradation has not been entirely resolved.

The development of optimization algorithms has accelerated in recent years, and they have been successfully applied to a variety of optimization issues. Adopt Particle Swarm Optimization (PSO) [20], GA (Genetic Algorithm), Nomadic People Optimizer[21], Firefly algorithm [22], and Global-best Local Neighbourhood oriented Particle Swarm Optimization (GbLN-PSO) [23] used to solve optimization issues with attempting to identify the most optimum point by adjusting parameters. Previous researches stated that the Firefly Algorithm outperforms both PSO and GA because it gives better and faster convergence to optimal conditions [24]. Since the advent of this algorithm, it has succeeded in solving optimization problems, such as structural optimization, image compression, and feature assortment [25]–[30].

Several previous studies proposed the Firefly algorithm to improve the Particle filter algorithm. The study in [22] proposed the use of state model position in pixel coordinate with scale parameter and Bhattacharyya distance as a measurement model. Another study performed simulation of multi-target tracking with different state and measurement model [31], and a simulation of tracking a 'snake- like' maneuvering anti-ship missile with different computation model [32]. The use of Histogram Intersection distance was 3–4 times faster than the Bhattacharyya distance due to simpler computation [33]. Fewer processor operations, time, and resources are valuable in real-time embedded processing with a multitasking environment. Therefore, this article proposes Firefly optimization using Histogram Intersection distance-based Particle Filter Algorithm for real-time human tracking applications. The contributions that should be highlighted are:

- We propose a real-time FOAPF to optimize particle distributions with minimal iteration and fix the problems of degeneracy and impoverishment by moving particles to a higher likelihood area according to the FO technique.
- In the likelihood function, we design using a Histogram Intersection distance instead of the Bhattacharyya distance.

2. Preliminary Knowledge

The Particle Filter approach for monitoring moving objects is a recursive application of Bayesian estimation to create a posterior probability distribution of the target state using prior information. This technique can provide good numerical predictions for cases involving nonlinear systems, such as object tracking in sequence videos [13]. The dynamic of the target is a first-order Markov model in most cases in equations (1) and (2). Arulampalam gives a mathematical explanation of a particle filter [3].

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

and stochastic measurement process:

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

At the time $\boldsymbol{n}, \mathbf{x}_n$as a not-observable state, $\mathbf{v}_n$ denotes measurement noise vectors, and $\mathbf{g}_n$ is a dynamic process noise. Using the posterior probability distribution to find solutions $p(\mathbf{x}_n|\mathbf{y}_{1:n})$target in each frame. Equation (3) could be used to predict and update the target state [3]:

$$p(\mathbf{x}_n|\boldsymbol{y}_{1:n-1}) = \int p(\mathbf{x}_n|\mathbf{x}_{\mathrm{n}-1})p(\mathbf{x}_{n-1}|\boldsymbol{y}_{1:n-1})d\mathbf{x}_{n-1} \tag{3}$$

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

With normalization constant:

$$p(\boldsymbol{y}_n|\boldsymbol{y}_{1:n-1}) = \int p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n \tag{4}$$

By applying a Bayesian recursive filter, the target probability density function $p(\mathbf{x}_n|\boldsymbol{y}_{1:n})$ which is tracked can be found. In nonlinear and non-gaussian systems, analytical solutions are difficult to find [13], thus the Particle Filter concept was introduced. The Particle Filter concept represents the posterior probability density function by providing the weight of each particle $\{\mathbf{x}_n^i, \mathbf{w}_n^i\}_{i=1}^N$ with Monte Carlo simulation. Each sample $\mathbf{x}_n$ and weights of $\mathbf{w}_n$ denotes the quality of the sample and indicates the hypothesis of the state target. The mean state estimate and the resampling procedure, both dependent on the weight of the particle, are two operations carried out by the Particle Filter (calculated based on the observation model).

A. Sequential importance sampling

Because sampling the posterior probability density function $p(\mathbf{x}_{0:n}|\mathbf{y}_{1:n})$ of the true target is challenging, and sequential importance sampling was used. To determine the posterior probability density, use sequence sampling and sequential analysis methods in statistics. Assuming that the importance of density function can be decomposed into equation (5) [34]:

$$q(\mathbf{x}_{0:n}|\mathbf{y}_{1:n}) = q(\mathbf{x}_n|\mathbf{x}_{0:\mathrm{n}-1}, \mathbf{y}_{1:n})q(\mathbf{x}_{0:n-1}|\mathbf{y}_{1:n-1}) \tag{5}$$

The posterior probability distribution analytic formula currently is:

$$p(\mathbf{x}_{0:n}|\mathbf{y}_{1:n}) = \frac{p(\mathbf{x}_{0:n,}\mathbf{y}_n|\mathbf{y}_{1:n-1})}{p(\mathbf{y}_n|\mathbf{y}_{1:n-1})} \tag{6}$$

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

$$= \frac{\dfrac{p(\mathbf{y}_n|\mathbf{x}_{0:n}, \mathbf{y}_{1:n-1})\mathbf{p}(\mathbf{x}_n|\mathbf{x}_{0:n-1}, \mathbf{y}_{1:n-1})}{p(\mathbf{x}_{0:n-1}|\mathbf{y}_{1:n-1})}}{p(\mathbf{z}_n|\mathbf{y}_{1:n-1})}$$

Based on Markov's law:

$$= \frac{p(y_n|x_n)p(x_n|x_{:n-1})\, p(x_{0:n-1}|y_{1:n-1})}{p(y_n|y_{1:n-1})} \tag{7}$$

Therefore $\mathbf{w}_n^i$ weights can be expressed recursively:

$$\boldsymbol{w}_n^i \propto \frac{p(x_{0:n}^i|y_{1:n})}{q(x_{0:n}^i|y_{1:n})}$$

$$\boldsymbol{w}_{n-1}^i \propto \frac{p(y_n|x_n^i)p(x_n^i|x_{n-1}^i)}{q(x_n^i|x_{0:n-1}^i, y_{1:n})} \tag{8}$$

In this case, $q(\mathbf{x}_n|\mathbf{x}_{0:n-1}, \mathbf{y}_{1:n}) = q(\mathbf{x}_n|\mathbf{x}_{n-1}^i, \mathbf{y}_n)$ then :

$$\boldsymbol{w}_n^i \propto \boldsymbol{w}_{n-1}^i \frac{p(y_n|x_n^i)p(x_n^i|x_{n-1}^i)}{q(x_n^i|x_{n-1}^i, y_n)} \tag{9}$$

The posterior probability density function $p(\mathbf{x}_n|\mathbf{y}_{1:n})$:

$$p(\mathbf{x}_n|\mathbf{y}_{1:n}) = \sum_{i=1}^N \mathbf{w}_n^{\sim i}\delta(\mathbf{x}_n - \mathbf{x}_n^i) \tag{10}$$

Where $\boldsymbol{N}$ equally the number of particles, with $N \to \infty$, and as a result, the actual target state probability density of the posterior is becoming closer $p(\mathbf{x}_n|\mathbf{y}_{1:n})$.

B. Importance Density Function

Importance Density Function $q(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{y}_n)$ associated with the effective sample size of the Particle Filter as in the algorithm [34]:

$$q(\mathbf{x}_n|\mathbf{x}_{n-1}^i,\mathbf{y}_n) = p(\mathbf{x}_n|\mathbf{x}_{n-1}^i,\mathbf{y}_n) \tag{11}$$
$$= \frac{p(\mathbf{y}_n|\mathbf{x}_n^i,\mathbf{x}_{n-1}^i)p(\mathbf{x}_n^i|\mathbf{x}_{n-1}^i)}{p(y_n^i|\mathbf{x}_{n-1}^i)}$$
$$= \frac{p(\mathbf{y}_n|\mathbf{x}_n^i)p(\mathbf{x}_n^i|\mathbf{x}_{n-1}^i)}{p(y_n^i|\mathbf{x}_{n-1}^i)}$$

The updated particle weights:

$$\mathbf{w}_n^i = \mathbf{w}_{n-1}^i p(y_n|\mathbf{x}_{n-1}^i) \tag{12}$$

$$\mathbf{w}_n^i = \mathbf{w}_{n-1}^i \int p(\mathbf{y}_n|\mathbf{x}_n^i)p(\mathbf{x}_n^i|\mathbf{x}_{n-1}^i)d\mathbf{x}_n^i$$

The importance probability density function has to be sampled from $q(\mathbf{x}_n|\mathbf{x}_{n-1}^i,\mathbf{y}_n)$, and each new state is integrated to acquire an effective distribution; the prior probability density function is reflected as an importance density function:

$$q(\mathbf{x}_n|\mathbf{x}_{n-1}^i,\mathbf{y}_n) = p(\mathbf{x}_n^i|\mathbf{x}_{n-1}^i) \tag{13}$$

So that the weight of the particles is updated:

$$\boldsymbol{w}_n^i = \boldsymbol{w}_{n-1}^i p(\boldsymbol{y}_n|\boldsymbol{x}_n^i) \tag{14}$$

C. Resampling process

Most of the particle weights become very small or even zero with the iteration process [34]. Calculating particles with small weights takes time. The importance variance weight will progressively grow, enhancing the posterior target probability density function. One of the most common methods has been to employ a good recommendation distribution and resampling technology. This equation (15) can be used to determine the degree of particle degradation based on weight [3].

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}(w_n^i)^2} \tag{15}$$

According to the formula above, the higher the particle weight, the fewer the effective particles are, and particle weights can degrade. The most common approach for calculating particle weights is the uniform sampling $w_n^i = {}^1/_N$. Although resampling procedures can minimize the impacts of weight degeneracy, a new problem known as sample impoverishment will emerge after the resampling step. When the number of particles with significant weights grows very small, and low-weight particles are eliminated during the resampling process, this issue arises. As a result of the resampling procedure, weight degeneracy becomes sample impoverishment. It could be done by moving particles employing genetic algorithms or the Firefly optimization technique. As a result, low-weight particles can be maintained, and estimate accuracy can be achieved only with a few particles.

In this implementation, the selected state consists of the target position $(\mathbf{x}_n,\mathbf{y}_n)$ and target velocity (x'_n,y'_n) and scale of the target change s then, $\mathbf{x}_n = [x_n \quad y_n \quad x'_n \quad y'_n \quad s]^T$. Assuming that the target being tracked is constantly moving and there is no sudden movement, the system model equation can be approached with a constant velocity model. $\mathbf{x}_{n+1} = A\,\boldsymbol{x}_{n-1} + \boldsymbol{g}_n$. Where A is the transition matrix.

3. Histogram Intersection distance in Particle Filter Algorithm with Firefly Algorithm Optimization

A. Particle Filter with Firefly Algorithm Optimization

Since its invention [35], Firefly Algorithm has been developed by several types of research. For instance, [31] Firefly Algorithm was used to track multitarget, [27][22] to do object tracking, and simulation [32]. According to Yang [35] the Firefly algorithm was enthused by firefly social behavior and interaction in their group through the light on their tail. Generally, fireflies generate short and periodic flashes of light that have unique patterns in each species. The sparkling light of fireflies has two functions: attracting other fireflies to be a couple and trapping.

The light intensity I with specific gaps r from a flashes light source is inversely proportional with the square of the gap, which means the intensity of the light will reduce if the gap r increase. It can be stated as $I \propto \frac{1}{r^2}$. Besides, the air can absorb light; therefore, as distance increases, the light weakens. Because of these two phenomena, fireflies can only be seen from a short distance. Generally, this condition is good enough for fireflies to communicate. The firefly characteristics can be seen as more ideal and adapted into the algorithm by the first originator Xin-She Yang [24], who named Firefly Algorithm (FA) in 2010. This FA has three rules, according to Yang, specifically:

- Every firefly is unisex and will be involved with each other.
- The attractiveness of fireflies is related to flashes of light intensity belonging. Fireflies with less bright flashes of light will be involved and move toward more luminous fireflies. If no fireflies are shining brighter than themselves, they will move randomly.
- The objective function determines the intensity of fireflies light. For a maximization problem, the light intensity is related to the object value. Other appearances of light intensity can be described in a similar method using a fitness equation.

The attractiveness of fireflies is related to flashes of light intensity, which others can see. The attractiveness is expressed as [24]:

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{16}$$

where $\beta(r)$ as the attractiveness of fireflies at specific distance r, β_0 is the attractiveness of fireflies at $r = 0, \gamma$ is coefficient of light absorption, and r is the distance between the source fireflies, which has the brightest light and other fireflies who see them. Distance between two fireflies i and j in the coordinate position x_i and x_j are Cartesian distance as [24]:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2} \tag{17}$$

Since knowing the $x_{i,k}$ as a $k -$ component in x_i coordinate of fireflies i, the movement of fireflies i which is attracted to fireflies j (which are brighter or have higher attractiveness) [24] are expressed as

$$x_i = x_i + \beta_0 e^{-\gamma r^2}\left(x_j - x_i\right) + \alpha\left(rand - \frac{1}{2}\right) \tag{18}$$

Rand is the function of generating random numbers with uniform distribution with a range of [0,1]. Generally, $\beta_0 = 1$ and $\alpha \in [0,1]$. The randomization process can be carried out using normal distribution $N(0,1)$ or another distribution. The FOAPF Algorithm is added to move fireflies to come near the brightest fireflies, which has the least distance of color histogram. The flowchart of these algorithms is shown in Figure 1.

$$RMSE = \sqrt{\frac{\sum_{i-1}^{M}(predicted_i - groundtruth_i)^2}{M}} \tag{19}$$

The Firefly Algorithm has been utilized for optimization in PF. We tested our algorithm with three datasets and showed the performance of the proposed algorithm in tracking the object correctly. RMSE value, in (19), will be used to measure discrepancy from the ground truth. The smaller RMSE value indicates that our proposed algorithm results approach the actual value.

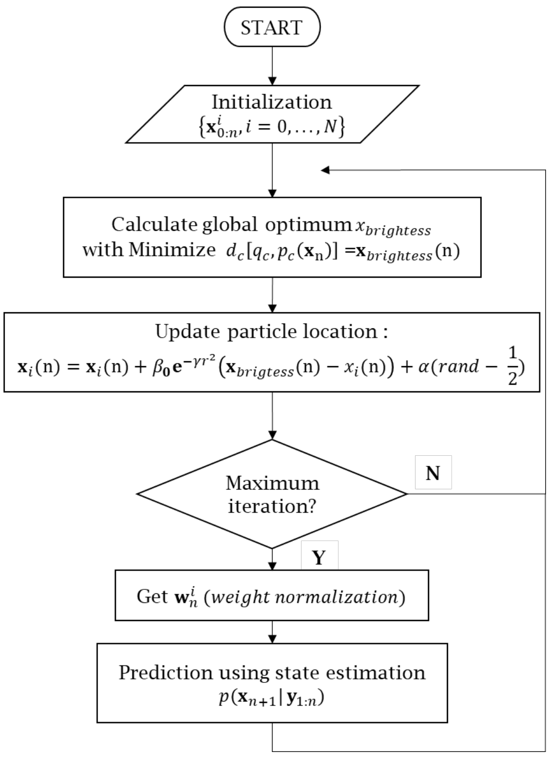

START
Initialization $\{\mathbf{x}^i_{0:n}, i = 0, \ldots, N\}$
Calculate global optimum $x_{brightess}$ with Minimize $d_c[q_c, p_c(\mathbf{x}_n)] = \mathbf{x}_{brightess}(n)$
Update particle location : $\mathbf{x}_i(n) = \mathbf{x}_i(n) + \beta_0 \mathbf{e}^{-\gamma r^2}(\mathbf{x}_{brigtess}(n) - x_i(n)) + \alpha(rand - \frac{1}{2})$
Maximum iteration?
N
Y
Get $\mathbf{w}^i_n$ (weight normalization)
Prediction using state estimation $p(\mathbf{x}_{n+1} | \mathbf{y}_{1:n})$

Figure 1. Flowchart of FOAPF

B. Histogram Intersection distance in Particle Filter algorithm

The likelihood function used is based on the color of the target being tracked. We can measure the similar distance between the target object and the searched target with Histogram Intersection distance [36], as shown in Fig. 2.

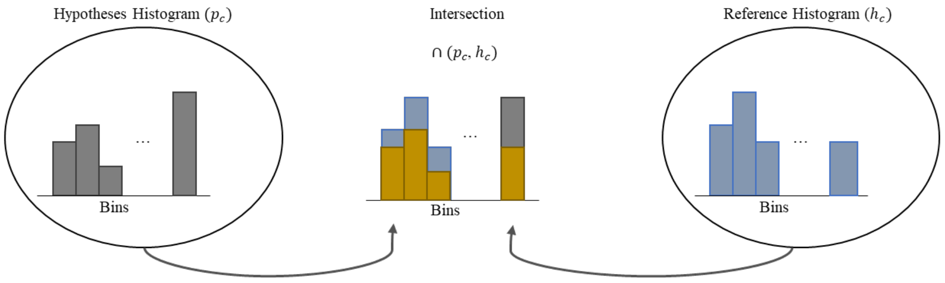


Figure 2. Histogram Intersection distance

$$\cap\,(p_c, h_c) = \frac{\sum_{m=1}^{K} min(p_{c,m}, h_{c,m})}{\sum_{m=1}^{K} h_{c,m}} \tag{20}$$

where h_c as a color histogram of reference and p_c as a color hypotheses histogram of particle which established at time step k. Considered in (20), normalized histograms $\cap\,(p_c, h_c) = 1$, indicated the perfect similarity between hypotheses histogram and reference histogram, and vice versa as in [33]. In equation (20), the min fuction acquires two arguments of two values an takes a smallest one and divided it by the number of histogram reference pixels. To achieve the Histogram Intersection distance d_c, make a complement as in equation (21):

$$d_c[p_c, h_c] = 1 - \cap\,(p_c, h_c) \tag{21}$$

We use a similar distance with Histogram Intersection distance instead of Bhattacharyya distance as in equation (23):

$$\rho[p_c, h_c] = \sum_{m=1}^{K} \sqrt{p_{c,m}, h_{c,m}} \quad (22)$$

$$d_{Bhattacharyya}[p_c, h_c] = \sum_{m=1}^{K} \sqrt{1 - \rho[p_c, h_c]} \quad (23)$$

Next, color likelihood is calculated by the formula [11]:

$$p(\boldsymbol{z}_k | \boldsymbol{x}_k^i) \propto exp(-\lambda \times {d_c}^2) \quad (24)$$

Based on [11], due to the consistent exponential behavior, $\lambda = 25$ in this experiments. The equation (24) as a likelihood function to calculate the state estimates.

4. Dataset

This study uses four datasets and is divided into two types: two private datasets and two the TB dataset [37][38]. This private dataset consists of two categories, fast-moving objects, and slow-moving objects. Researchers commonly use the TB dataset in computer vision, and first, we were converted into a video format. We used the TB dataset to analyze when the object had partial occlusion and total occlusion.

5. Overall System Architectures

After we have designed the FOAPF method, the proposed system is shown in Fig. 3. This implementation using PTZ Camera as a sensor for tracking and open-source libraries [39]. The ROI will be selected manually for testing the proposed algorithm. Moreover, it will be compared to vanilla PF. The proposed algorithm will be compared with five popular methods, i.e., Vanilla PF, Kernelized Correlation Filter (KCF), Minimum Output Sum of Square Error (MOSSE), Continuously Adaptive Mean Shift and Kalman Filter (Cam-shift Kalman Filter), Tracking Learning Detection (TLD) to know the performance. The pseudocode of the FOAPF with Histogram Intersection distance shows in Fig. 3.

```
FOAPF Algorithm
Particle sampling with $N$ particle and calculate weight
Make a prediction based on the state model $\{\mathbf{x}_{n-1}^{i}, \mathbf{w}_{n-1}^{i}\}$
Updating particle based on the observation model $\boldsymbol{y}_n$
Calculate histogram intersection distance $d_i$ at $x_i (i = 1,2,3, \ldots n)$
Create a population of fireflies
Define $\gamma$, $\alpha$, and $\beta_0$
**While (t < the number of iteration)**
Get the firefly location $x_j$ that has minimum histogram intersection distance ($d_j$)
    **For** $i$=1 *in the total number of fireflies*
        Move firefly $x_i$ to $x_j$
        Calculate $r$ distance between fireflies
        Calculate and update the attractiveness $\beta(r)$ with a distance $r$
    **End for *i***
Update the firefly position
***End while***
Normalize weight
Iteration from prediction step
```

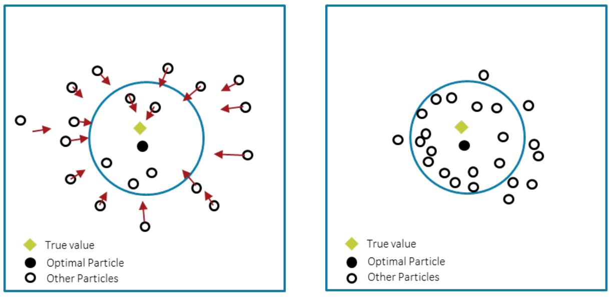
Figure 3. The pseudocode of FOAPF with Histogram Intersection distance

6. Experimental Results

The PTZ camera produces 9120x540 pixels for each frame with a light intensity of 1200 LUX in the real-time implementation. Therefore, to achieve the real-time implementation, each frame was resized into 640x360. This image size is assumed to be sufficient to track the moving object with a real-time application. The maximum speed produced by PTZ Camera is 30-FPS.

The observation technique in Vanilla PF is based on color features. Histogram value has been generally used. The color histogram for each particle will be calculated with uniform distribution as the initial distribution particle. The distribution of these particles is spread around the Initial Region of Interest (ROI). This research was conducted on our private dataset and TB dataset [37][38]. The types of data sets are tracking slow-moving human datasets, fast-moving human datasets, and TB datasets.

A comparison of algorithms between Vanilla PF and FOAPF is the determining technique to get the optimum likelihood function from the target being tracked. The color histogram of each particle is computed for each time step and compared to the model target being tracked with the histogram intersection equation calculated as in (21). Each particle distribution $x_{i,k}$ is the component of k particle in x_i coordinate and fireflies i. The firefly movement or particles i are attracted to fireflies j (brighter or has higher attractiveness) and compare their histogram value with initial ROI or saved model. The condition of particle distribution before being optimized is shown in Fig. 4. The purpose of adding this optimization algorithm is, with a few particles, all particles are expected to approach the actual color histogram model value, which produces less computational cost.



(a) (b)

Figure 4. Particle Distribution: (a) Before optimization; (b)After optimization using Firefly Algorithm.

FO Algorithm has parameters that should be set, such as $\beta(r)$ value, which is the attractiveness of fireflies at a distance r hence it should be updated until the system is convergent. Based on the experiment, we tried to start the value of $\gamma = 0$, and the system failed to track because based on equation (16), it will be constant and make $\beta = \beta_0$. Then, we start with the value =0.1, and the system could track the object. We analyzed the range parameter of $\gamma = (0.1 - 1)$ and divided into ten parts with 0.1 intervals. After several experiments, we get the best the γ value range is $0 > \gamma <= 0.3$. The system fails to track the object when the value of $\gamma > 0.3$. The experiment results show that the track works well when the value of α is $0 > \alpha \leq 0.2$. The value of $\beta_0 = 1$ is attractiveness at $r = 0$ or when the distance between two or more fireflies equals 0. So, based on Yang, usually, the value of $\beta_0 = 1$. Therefore, this paper also uses the value $\beta_0 = 1$. The parameter that should be set is $\gamma = 0.3, \alpha = 0.0001, \beta_0 = 1$. As noticed, Table 1 shows the result of attractiveness value in our video data set, particularly in frame 12. This table shows that 200, 300, 400, 500, 600, 800, and 1000 particles acquired the most rapid convergence with $m = 3$. The attractiveness of firefly $\beta(r)$ was proportional to the light intensity of other fireflies. $\beta(r)$ value used was based on (16) that is shown in Table 1. Fireflies i movement, which is attracted to fireflies j (brighter), is essential to determine convergence speed and how this Firefly Algorithm works.

However, the iteration process and the number of particles affect real-time tracking performance. Therefore, the proposed FOAPF Algorithm needs only a one-time iteration. Candidate particles or fireflies, which have high attractiveness compared to the target model (fireflies j, which has brighter light), produce a smaller distance and vice versa.

Table 1. The results of the attractiveness value of fireflies

Iteration	$\boldsymbol{\beta(r)}$ with different numbers of particles							
	200	300	400	500	600	800	900	1000
m=1	0.87	0.85	0.88	0.85	0.86	0.91	0.93	0.88
m=2	1.00	0.97	1.03	0.96	0.99	1.12	1.15	1.04
m=3	1.02	0.99	1.06	0.98	1.01	1.17	1.21	1.07
m=4	1.02	0.99	1.06	0.98	1.02	1.18	1.23	1.08
m=5	1.02	0.99	1.06	0.98	1.02	1.18	1.24	1.08
m=6	1.02	0.99	1.06	0.98	1.02	1.18	1.24	1.08
m=7	1.02	0.99	1.06	0.98	1.02	1.18	1.24	1.08
m=8	1.02	0.99	1.06	0.98	1.02	1.18	1.24	1.08
m=9	1.02	0.99	1.06	0.98	1.02	1.18	1.24	1.08
m=10	1.02	0.99	1.06	0.98	1.02	1.18	1.24	1.08

To begin the comparison, we chose four challenging videos. These videos show various objects in difficult settings, quick and slow motions, low resolution, and partial or full occlusions. First, we used the Histogram Intersection distance to calculate the likelihood function, tested on the Panda dataset from TB dataset [38]. This video is a partially or fully occluded target type with low resolution. The RMSE results with the proposed algorithm as in Fig 5 (a). and 5 (b). The real-time implementation could be achieved with up to 50 distributions of particles.

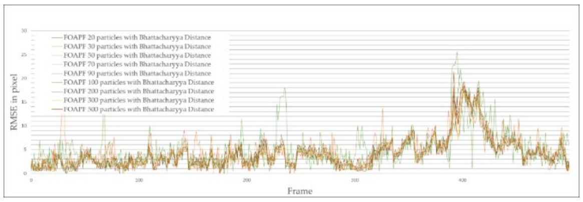


Figure 5 (a). RMSE of FOAPF with Bhattacharyya Distance (Panda dataset)

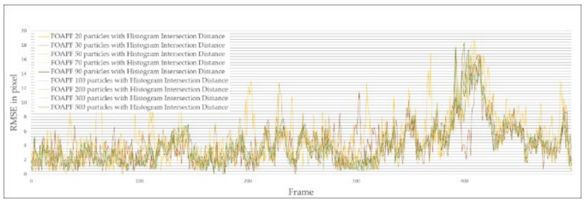


Figure 5 (b). RMSE of FOAPF with Histogram Intersection distance (Panda dataset)

Table 2 shows that with 50 particles and 5.02 pixels RMSE, using FOAPF with Bhattacharyya Distance could achieve real-time implementation with a specific processor. With 50 particles of FOAPF and Histogram Intersection distance, a lower RMSE of 4.96 and 30 FPS is obtained. It shows that the proposed method could enhance the tracking accuracy (shows in lower RMSE value). We chose the Histogram Intersection distance rather than the Bhattacharyya distance because, with lower RMSE, it ran up to 1.8 ms. While using Bhattacharyya, it ran up to 1.9 ms. It shows in FPS value with distribution particle of 70-500.

Table 2. RMSE tracker with FOAPF (Panda dataset)

Number of Particles	RMSE of FOAPF with Histogram Intersection Distance	FPS	RMSE of FOAPF with Bhattacharyya Distance	FPS
20	5.95	30	6.96	30
30	5.28	30	5.97	30
50	**4.96**	**30**	**5.02**	**30**
70	4.81	22	5.00	21
90	4.74	17	4.96	16
100	4.74	16	5.00	15
200	4.64	9	4.87	9
300	4.64	6	4.81	5
500	4.65	3	4.91	3

Figure 6 shows the BB (Bounding Box) result of the proposed method on the Panda dataset. Even if the target is partially/fully occluded and has low-resolution environments, the system can still track the target correctly.

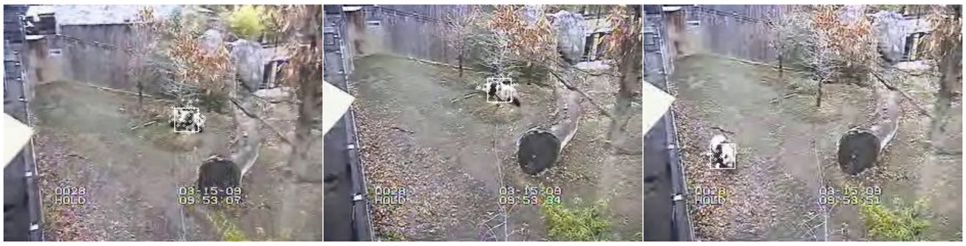


Figure 6. The BB results of FOAPF on Panda Dataset.

After we test the panda dataset, we get a low RMSE of FOAPF with the Histogram Intersection distance. Therefore, we test the other three videos with the same parameters, the models, and the likelihood function (Histogram Intersection distance) in Vanilla Particle Filter and FOAPF. RMSE values between Vanilla PF without optimization algorithm in slow human movement tracking dataset are shown in Figure 7, and FOAPF Algorithm is in Figure 8.

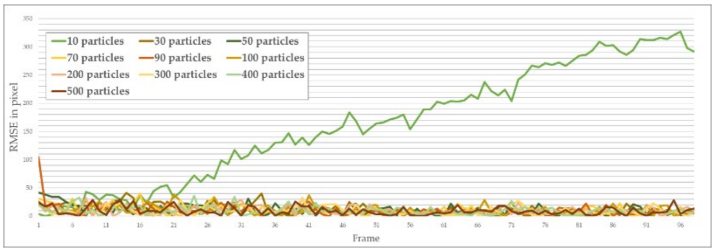


Figure 7. RMSE of Vanilla PF

The Vanilla PF produces the smallest RMSE with 500 particle distributions. However, with this result, the real-time system was not achieved. The distribution of 50 particles has reached 30 FPS. It means the real-time system has been reached a 19.83 RMSE value. Therefore, the FOAPF algorithm was tested to generates RMSE, as in Figure 8.

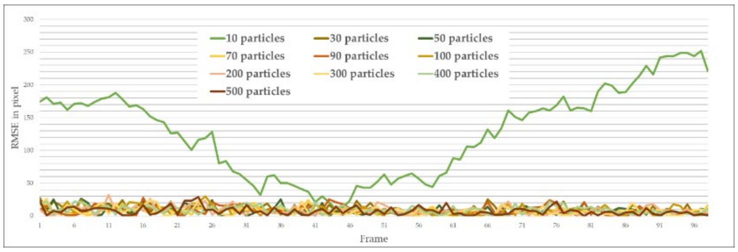


Figure 8. RMSE of FOAPF result

More detailed RMSE values are shown in Table 3. The RMSE result with a different number of particle distributions using Vanilla PF and FOAPF Algorithm is presented. Based on Table 3, the RMSE value of FOAPF was smaller than Vanilla PF Algorithm with no optimization. This method uses a Histogram Intersection to calculate the distance instead of using the Bhattacharyya distance.

Table 3. FPS and RMSE result between two algorithms in slow-moving human tracking

Number of Particles	Vanilla PF		FOAPF	
	RMSE	FPS	RMSE	FPS
10	189.63	30	142.90	30
30	20.66	30	18.28	30
50	**19.83**	**30**	**16.92**	**30**
70	19.611	21	16.70	21
90	15.77	16	11.09	16
100	13.81	16	10.73	16
200	12.90	9	10.49	7
300	12.46	5	9.97	5
500	12.13	3	9.65	3

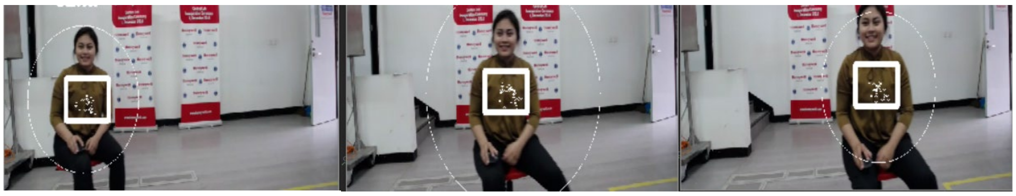


Figure 9. Slow human movement tracking

The Firefly Algorithm is particularly useful for estimating stability likelihood function in human movement tracking with the PF Algorithm. FOAPF RMSE values of each particle almost produce the same value because the optimization method for moving particles is only done in a one-time iteration and shows significantly different values from Vanilla PF Algorithm. All hypotheses of particle value can approximate the model position by looking for the most similar color histogram. As shown, Fig. 9 represents slow-moving human tracking using FOAPF Algorithm. Then, after being tested with the first dataset, the FOAPF Algorithm has shown that it could produce an RMSE value of at least 9.65 with 500 particle distributions. However, in the computational speed view, the maximal 30 FPS reached RMSE 16.92 with a distribution of 10-50 particles implied with our data set so that the real-time implementation could be reached.

Afterward, the FOAPF and Vanilla PF were tested with a second data set (with the same parameters as the first dataset). Fast human movement is in Figure 10 with a distance between the object, and PTZ camera of around $\pm 2m$ with lighting condition is 1200 LUX. The proposed algorithm produced high FPS in fast-walking human tracking activity with less RMSE value, as in Figure 11.

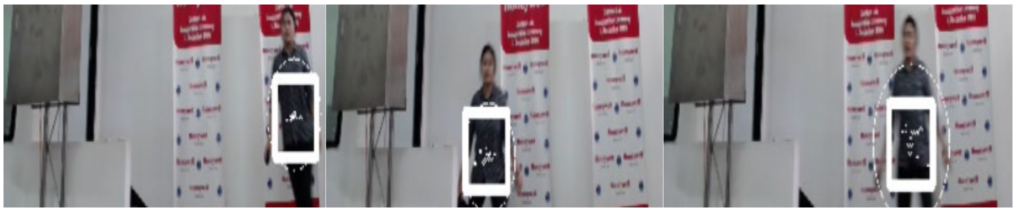


Figure 10. Fast human movement tracking

This algorithm was acceptable with our data set to produce low RMSE. With 10-200 particle distribution, the best-performing model was chosen and compared to other algorithms. The proposed algorithm was able to produce a near-real position value. Two threads are created to increase computing speed and make fast-walking and slow-moving human to be tracked correctly, which can run in parallel technique without waiting for other processes. This technique is more responsive and economical in dividing memory and resources even though using part of the program while executing another operation. The implementation used multi-threading and NVIDIA GEFORCE 940MX core i5 GPU specifications, as shown in Table 4.

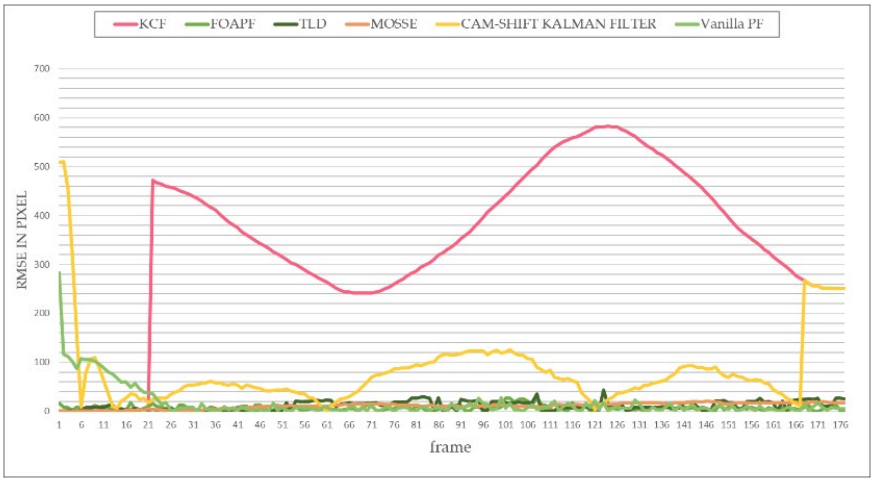


Figure 11. RMSE of different algorithms

The FOAPF Algorithm achieved a real-time process with 10-100 particle distributions with a second data set. In this case, computational speed and accuracy were trade-offs so that by using FOAPF, in tracking fast-walking human, the system had RMSE of 7.99 as in Table 4 with a minimum computing speed of 16 FPS. This result shows that FPS achieves above minimum value to reach a real-time implementation based on the previous study. Another result was 30 FPS, which means the proposed FOAPF produced a smaller RMSE value of 8.80 with 50 particle distributions, as shown in Table 4.

Figure 12 shows that the KCF algorithm was not able to track fast human movements. The speed suddenly decreased in the 21st frame. The object speed changed quickly from 177 pixels/sec to 42 pixels/sec.

Table 4. RMSE between Vanilla PF and FOAPF in fast-moving human tracking

Number of Particles	Vanilla PF		FOAPF	
	RMSE	FPS	RMSE	FPS
10	151.49	30	21.11	30
30	32.87	30	9.34	30
50	**36.36**	**30**	**8.80**	**30**
70	13.64	21	8.08	21
90	12.53	16	8.03	16
100	12.45	16	7.99	16
200	11.59	9	7.84	7
300	9.84	5	7.56	5
500	9.77	3	7.02	3

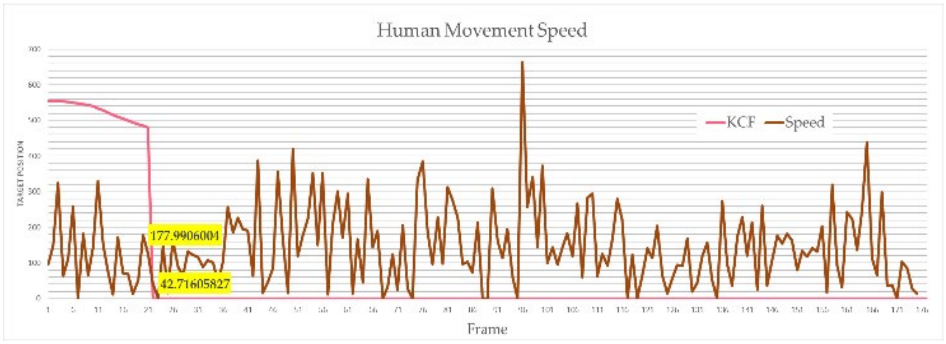


Figure 12. Human movement speed per pixel

Due to the complex background behind the target, as in Fig. 13 made the KCF algorithm in extracting features has not enough time for the detection and tracking process. The image captured was blurry because the object moved too fast with the camera distance to the object $\pm 2m$, so the KCF algorithm could not detect the next target position.

Table 5. RMSE with several algorithms in fast-moving human tracking

	Vanilla PF	FOAPF	Cam-Shift Kalman Filter	KCF	TLD	MOSSE
RMSE	36.36	8.80	111.31	381	14.40	12.55

The reliability of the FOAPF Algorithm tested with a fast-walking target with an average speed of 96 pixels/second. When the result is compared to several other methods, the proposed method produces smaller RMSE, as shown in Table 5. Cam-shift Kalman Filter method produced RMSE 111. 31. The result position has a slow response to the fast-walking human movement because the modeling is assumed to be tracking objects with constant speed. The TLD [40] and MOSSE [41] methods produce lower RMSE than the Cam-shift Kalman Filter and KCF [42] methods. This FOAPF method is very reliable, and the speed of fireflies with only one iteration can get near the fireflies with brighter light. Almost all particles or fireflies have a color histogram value, which closes to the model.

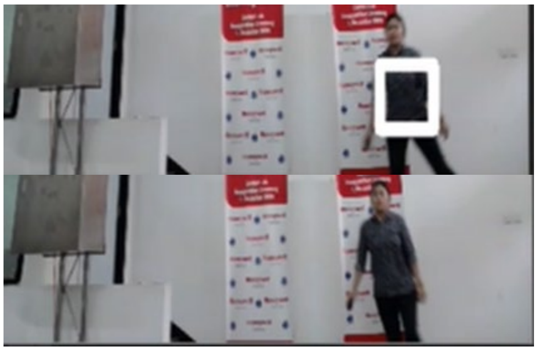


Figure 13. Human movement speed per pixel

This algorithm was tested using the TB occlusion dataset [37] with the same parameter as the first and second datasets for further validation. The RMSE value of the FOAPF Algorithm and others are presented in Table 6. Cam-shift Kalman Filter shows the lowest RMSE value. It was hard for the cam-shift algorithm as an observer in Kalman Filter to handle the occlusion case. However, because of the severe occlusion, it could not detect and predict the actual object location. The RMSE of the FOAPF Algorithm produced 12.07 with 50 distributions of particles as in Table 6.

Table 6. RMSE tracker with FOAPF, Kalman-Filter and Vanilla PF Algorithm

	Vanilla PF	FOAPF	Cam-Shift Kalman Filter
RMSE	24.99	12.07	49.25

The experiment result with TB dataset as in Fig. 14 shows a Firefly method for optimization in PF can obtain the minimum objective function in the hypotheses area.

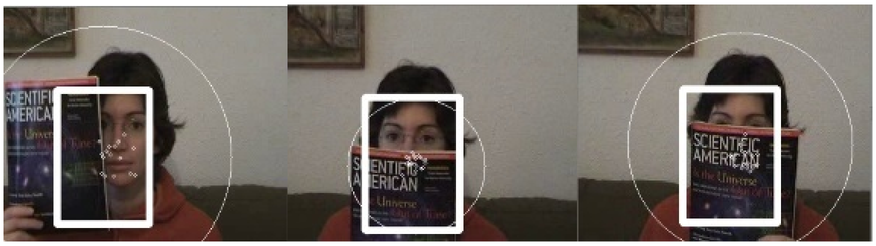


Figure 14. The BB results of FOAPF on TB Occlusion Dataset.

To achieve a real-time implementation, we utilize only one iteration in the Firefly algorithm while there is a probability of getting local minimal conditions, and the RMSE result is shown in Fig. 15.

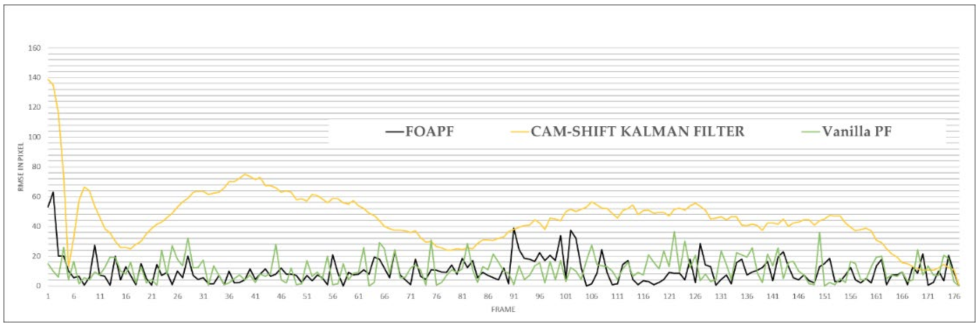


Figure 15. RMSE of FOAPF on the TB dataset.

7. Discussion

This proposed method was designed with multi-threading to get a real-time process and less particle distribution in the PF algorithm. We have tested in fast-moving objects and using Cam-shift Kalman Filter. It cannot follow the target correctly. The detection technique between Kalman Filter and PF is quite similar, which uses color distribution. However, an optimization algorithm to find the color (like the object model) has been added to PF. The PF is a tracker that repeatedly uses a state estimation technique based on the recursive Bayesian Filter (a prediction and updating process).

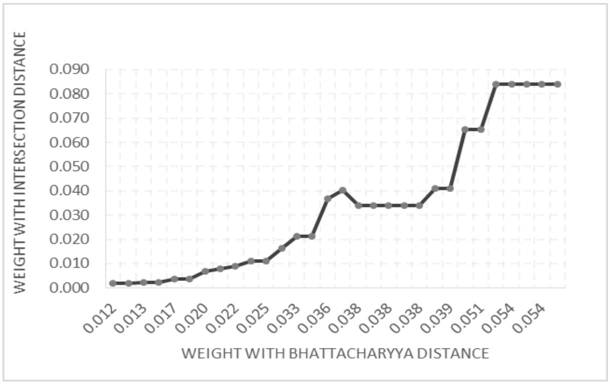
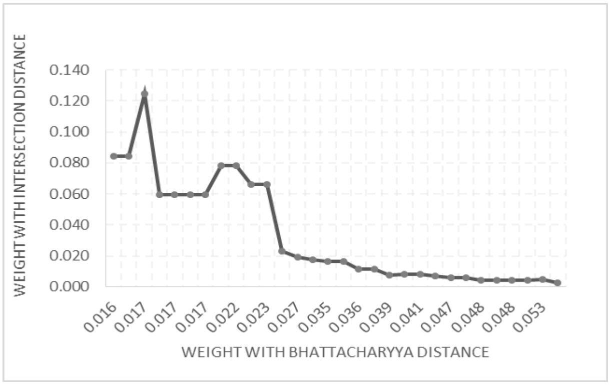


Figure 16 (a). Illustration of the effect of likelihood weight values using the Histogram Intersection and Bhattacharyya distance.



(b). Scatter of likelihood weight values with Histogram Intersection and Bhattacharyya distance

The PF Algorithm is a nonlinear filtering method to execute the state estimation process using the Bayes equation. Bayes equation has difficulties finding an analytical solution so that it uses the Monte Carlo sampling technique of selected density proposal and weighs according to the observed value. The number of particle distributions in BB is the probability value to find the actual object position. In this study, the selected color feature, measurement equation in the updating process, uses Histogram Intersection in (20), which uses the color histogram information captured by the camera sensor.

Figure 16(a) shows the comparison between weighting a Bhattacharyya distance corresponding to Histogram Intersection distance. For a perfect match between histograms that have been normalized is $d_c = 1$. To achieve a dissimilarity measure, multiply the intersection by the complement to get the Histogram Intersection distance as in equation (21). So, if we use the dissimilarity function of the histogram intersection distance, plotting between histogram intersection distance and Bhattacharyya distance is shown in Fig. 16(b). We chose the Histogram Intersection distance distance rather than the Bhattacharyya distance because it ran up to 1.8 ms and the Bhattacharyya with 1.9 ms. This result shows the time result slightly different, and in real-time processing, this is quite useful.

This PF is a tracker with a detection process included in the measurement equation for the updating process. Then, the measurement resulted in the updating process weighs the particles. If the weight value is large, it means that the probability of the object in that location is also large. The task of resampling is to remove particles that have insignificant weight and only focus on significant-weighted particles. Then, this large, weighted particle will be broken down again. In Fig. 17, the application of the Firefly Algorithm as optimization is to avoid the phenomenon of sample degeneration and find the actual position object.

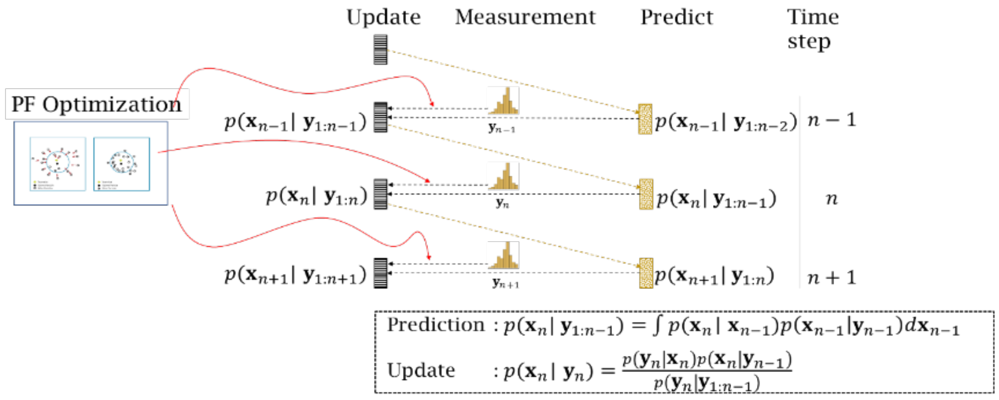

Update
Measurement
Predict
Time step
PF Optimization
$p(\mathbf{x}_{n-1}|\ \mathbf{y}_{1:n-1})$
$\mathbf{y}_{n-1}$
$p(\mathbf{x}_{n-1}|\ \mathbf{y}_{1:n-2})$
$n-1$
$p(\mathbf{x}_{n}|\ \mathbf{y}_{1:n})$
$\mathbf{y}_{n}$
$p(\mathbf{x}_{n}|\ \mathbf{y}_{1:n-1})$
n
$p(\mathbf{x}_{n+1}|\ \mathbf{y}_{1:n+1})$
$\mathbf{y}_{n+1}$
$p(\mathbf{x}_{n+1}|\ \mathbf{y}_{1:n})$
$n+1$
Prediction : $p(\mathbf{x}_n|\ \mathbf{y}_{1:n-1}) = \int p(\mathbf{x}_n|\ \mathbf{x}_{n-1})p(\mathbf{x}_{n-1}|\mathbf{y}_{n-1})d\mathbf{x}_{n-1}$
Update : $p(\mathbf{x}_n|\ \mathbf{y}_n) = \frac{p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{y}_{n-1})}{p(\mathbf{y}_n|\mathbf{y}_{1:n-1})}$

Figure 17. The proposed algorithm steps

The study results, tested on three different data sets, show that the FOAPF Algorithm can produce the probability of each sample. It has a significant effect on the calculation of pdf approximation. Therefore, by using 50 particles (few particles) and adding the optimization process, the computation reaches the maximum capability of a PTZ camera of 30 FPS with a relatively lower RMSE than the Vanilla PF. It can be concluded that this system can be applied in real-time conditions with small RMSE.

8. Conclusion

The proposed method solves weight degradation and impoverishment particles in the Particle Filter algorithm, affecting tracking accuracy. The pixels video produced by the PTZ camera uses 640x360, and it is done in 1200 LUX room lighting. This system has been succeeding in tracking the fast-moving object wherever the object moves and is oriented. The research aims to achieve real-time implementation for object tracking using a small particle distribution combined. This method results in PF with fewer particle distribution, and when adding FO, it can be applicable and very reliable if implemented for real-time fast-walking human tracking. Using a Histogram Intersection distance with Firefly Algorithm makes most particles move to high probability areas. It can maintain the weight degradation and impoverishment with a small particle distribution of 50 particles and achieve real-time implementation. Real-time color tracking is based on FOAPF and multi-threading in fast-walking human tracking. The average speed of 96 pixels/sec with the distance between objects and a camera of 2 m. The result achieves RMSE as low as 8.80 by distributing 50 particles in PTZ camera maximum of 30 FPS. In slow-moving human tracking presents RMSE of 16.92, and the TB dataset shows RMSE values of 4.96 and 12.07. According to the RMSE result, the proposed FOAPF as a color tracker with few particle distributions can also enhance human movement tracking with real-time implementation.

9. Acknowledgement

This research is a part of Research, Community Services, and Innovation program (Program Penelitian, Pengabdian kepada Masyarakat dan Inovasi / P3MI 2020) funded by Institute for Research and Community Services at the Institut Teknologi Bandung (LPPM ITB).

10. References

[1] J. Shin, H. Kim, D. Kim, and J. Paik, “Fast and robust object tracking using tracking failure detection in kernelized correlation filter,” *Appl. Sci.*, vol. 10, no. 2, 2020, doi: 10.3390/app10020713.

[2] B. Zhou and T. Wang, “Adaptive context-aware and structural correlation filter for visual tracking,” *Appl. Sci.*, vol. 9, no. 7, 2019, doi: 10.3390/app9071338.

[3] 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.

[4] L. Yulianti, B. R. Trilaksono, A. S. Prihatmanto, and W. Adiprawita, “Particle filter-based multitarget multicamera tracking system utilizing random finite sets and distributed estimation process,” *Int. J. Electr. Eng. Informatics*, vol. 8, no. 3, pp. 675–695, 2016, doi: 10.15676/ijeei.2016.8.3.14.

[5] Y. Wang *et al.*, “Particle Filter Vehicles Tracking by Fusing Multiple Features,” *IEEE Access*, vol. 7, pp. 133694–133706, 2019, doi: 10.1109/access.2019.2941365.

[6] H. Chu, K. Wang, and X. Xing, “Target Tracking via Particle Filter and Convolutional Network,” *J. Electr. Comput. Eng.*, vol. 2018, 2018, doi: 10.1155/2018/5381962.

[7] D. A. Maharani, C. Machbub, and P. H. Rusmin, “Enhancement of Missing Face Prediction Algorithm with Kalman Filter and DCF-CSR,” in *Proceedings of the International Conference on Electrical Engineering and Informatics*, 2019, vol. 2019-July, pp. 395–399, doi: 10.1109/ICEEI47359.2019.8988867.

[8] K. Nummiaro, E. Koller-Meier, and L. Van Gool, “An adaptive color-based particle filter,” *Image Vis. Comput.*, vol. 21, no. 1, pp. 99–110, 2003, doi: 10.1016/S0262-8856(02)00129-4.

[9] S. Wildermann and J. Teich, “3D person tracking with a color-based particle filter,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4931 LNCS, pp. 327–340, 2008, doi: 10.1007/978-3-540-78157-8_25.

[10] M. T. N. Truong and S. Kim, “Parallel implementation of color-based particle filter for object tracking in embedded systems,” *Human-centric Comput. Inf. Sci.*, vol. 7, no. 1, 2017, doi: 10.1186/s13673-016-0082-1.

[11] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, “Color-based probabilistic tracking,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2350, pp. 661–675, 2002, doi: 10.1007/3-540-47969-4_44.

[12] Z. Wang, B. Chen, and J. Wu, “Effective Inertial Hand Gesture Recognition Using Particle Filtering Based Trajectory Matching,” *J. Electr. Comput. Eng.*, vol. 2018, pp. 1–9, 2018, doi: 10.1155/2018/6296013.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] W. C. Cheng, “PSO algorithm particle filters for improving the performance of lane detection and tracking systems in difficult roads,” *Sensors (Switzerland)*, vol. 12, no. 12, pp. 17168–17185, 2012, doi: 10.3390/s121217168.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] X. S. Yang, S. S. S. Hosseini, and A. H. Gandomi, “Firefly Algorithm for solving non-convex economic dispatch problems with valve loading effect,” in *Applied Soft Computing Journal*, 2012, vol. 12, no. 3, pp. 1180–1186, doi: 10.1016/j.asoc.2011.09.017.

[26] J. Jumadinova and P. Dasgupta, “Firefly-inspired synchronization for improved dynamic pricing in online markets,” in *Proceedings - 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2008*, 2008, pp. 403–412, doi: 10.1109/SASO.2008.26.

[27] M. L. Gao, X. H. He, D. S. Luo, J. Jiang, and Q. Z. Teng, “Object tracking using firefly algorithm,” *IET Comput. Vis.*, vol. 7, no. 4, pp. 227–237, 2013, doi: 10.1049/iet-cvi.2012.0207.

[28] Y. Tsukamoto, Y. Matsumoto, and T. Wada, “Tracking a firefly -a stable likelihood estimation for variable appearance object tracking-,” *Proc. - Int. Conf. Pattern Recognit.*, pp. 2–5, 2008, doi: 10.1109/icpr.2008.4761478.

[29] A. Liu, K. Chen, Q. Liu, Q. Ai, Y. Xie, and A. Chen, “Feature selection for motor imagery EEG classification based on firefly algorithm and learning automata,” *Sensors (Switzerland)*, vol. 17, no. 11, 2017, doi: 10.3390/s17112576.

[30] P. Wu, S. Su, Z. Zuo, X. Guo, B. Sun, and X. Wen, “Time difference of arrival (TDOA) localization combining weighted least squares and firefly algorithm,” *Sensors (Switzerland)*, vol. 19, no. 11, 2019, doi: 10.3390/s19112554.
[31] M. Tian, Y. Bo, Z. Chen, P. Wu, and C. Yue, “Multi-target tracking method based on improved firefly algorithm optimized particle filter,” *Neurocomputing*, vol. 359, no. xxxx, pp. 438–448, 2019, doi: 10.1016/j.neucom.2019.06.003.
[32] W. Zhou, L. Liu, and J. Hou, “Firefly Algorithm-Based Particle Filter for Nonlinear Systems,” *Circuits, Syst. Signal Process.*, vol. 38, no. 4, pp. 1583–1595, 2019, doi: 10.1007/s00034-018-0927-0.
[33] P. Dunne and B. Matuszewski, “Choice of similarity measure, likelihood function and parameters for histogram based particle filter tracking in CCTV grey scale video,” *Image Vis. Comput.*, vol. 29, no. 2–3, pp. 178–189, 2011, doi: 10.1016/j.imavis.2010.08.013.
[34] T. Wang, W. Wang, H. Liu, and T. Li, “Research on a face real-time tracking algorithm based on particle filter multi-feature fusion,” *Sensors (Switzerland)*, vol. 19, no. 5, 2019, doi: 10.3390/s19051245.
[35] X. S. Yang, “Firefly algorithms for multimodal optimization,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5792 LNCS, pp. 169–178, 2009, doi: 10.1007/978-3-642-04944-6_14.
[36] M. J. Swain and D. H. Ballard, “Color indexing,” *Int. J. Comput. Vis.*, vol. 7, no. 1, pp. 11–32, 1991, doi: 10.1007/BF00130487.
[37] “http://cvlab.hanyang.ac.kr/tracker_benchmark/seq/FaceOcc1.zip.” .
[38] “http://cvlab.hanyang.ac.kr/tracker_benchmark/seq/Panda.zip.” .
[39] Bradski G, “The OpenCV library,” *Dr. Dobb’s J. Softw. Tools*, vol. 25, no. 120, pp. 122–125, 2000.
[40] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-learning-detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, 2012, doi: 10.1109/TPAMI.2011.239.
[41] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, “Visual object tracking using adaptive correlation filters,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2544–2550, 2010, doi: 10.1109/CVPR.2010.5539960.
[42] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, 2015, doi: 10.1109/TPAMI.2014.2345390.

Devira Anggi Maharani received B.Sc. and M.Sc degrees from Politeknik Negeri Malang and Institut Teknologi Bandung, in 2015 and 2018. As a doctoral student at Institut Teknologi Bandung, her research projects explore the computer vision and control, especially in real-time implementation of object tracking to Instrumentation and Control.

Carmadi Machbub received his Bachelor degree in electrical engineering from Institut Teknologi Bandung (ITB) in 1980, DEA in Control Engineering and Industrial Informatics in 1988 and Doctoral degree in Engineering Sciences majoring in Control Engineering and Industrial Informatics from Université de Nantes/Ecole Centrale de Nantes in 1991. He is now professor and Head of Control and Computer Systems Research Division, School of

Electrical Engineering and Informatics, ITB. His current research interests are in control, machine perception and intelligent systems.

Lenni Yulianti received her Bachelor (Cum Laude), Master's, and Doctoral degree in Electrical Engineering from Institut Teknologi Bandung (ITB), School of Electrical Engineering and Informatics in Indonesia. She is currently a lecturer and a researcher in the School of Electrical Engineering and Informatics, ITB. Her research interests include statistical signal processing, visual-based tracking, state estimation, control system, and implementation of machine learning

Pranoto Hidaya Rusmin was born in Magelang, Indonesia in 1972. He received B.Eng., M.Eng., and Doctor degrees in electrical engineering from Institut Teknologi Bandung (ITB), Indonesia, in 1996, 1999, 2009, respectively. Since 1998, he is a Lecturer at School of Electrical Engineering and Informatics ITB, Indonesia. His research interest is Internet Congestion Control.