Received: 13 August 2018 | Revised: 7 January 2019 | Accepted: 13 February 2019

DOI: 10.4218/etrij.2018-0435

ORIGINAL ARTICLE

ETRI Journal WILEY

Dynamic swarm particle for fast motion vehicle tracking

Grafika Jati[1] | **Alexander Agung Santoso Gunawan**[2] | **Wisnu Jatmiko**[1]

[1]Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia

[2]School of Computer Science, Bina Nusantara University, Jakarta, Indonesia

Correspondence
Wisnu Jatmiko, Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia.
Email: wisnuj@cs.ui.ac.id

Nowadays, the broad availability of cameras and embedded systems makes the application of computer vision very promising as a supporting technology for intelligent transportation systems, particularly in the field of vehicle tracking. Although there are several existing trackers, the limitation of using low-cost cameras, besides the relatively low processing power in embedded systems, makes most of these trackers useless. For the tracker to work under those conditions, the video frame rate must be reduced to decrease the burden on computation. However, doing this will make the vehicle seem to move faster on the observer's side. This phenomenon is called the fast motion challenge. This paper proposes a tracker called dynamic swarm particle (DSP), which solves the challenge. The term particle refers to the particle filter, while the term swarm refers to particle swarm optimization (PSO). The fundamental concept of our method is to exploit the continuity of vehicle dynamic motions by creating dynamic models based on PSO. Based on the experiments, DSP achieves a precision of 0.896 and success rate of 0.755. These results are better than those obtained by several other benchmark trackers.

KEYWORDS
dynamic model, fast motion, particle filter, particle swarm optimization, vehicle tracking

1 | INTRODUCTION

An intelligent transportation system (ITS) is an application of information technology in the transportation system to improve safety and mobility [1]. ITS implementation needs supporting technologies, which can be divided into two categories: infrastructure-side technology and vehicle-side technology. In light of the availability of low-cost cameras and embedded systems, we would like to exploit these devices for implementing the supporting technologies based on computer vision [2]. Therefore, the grand objective of our research is to develop an automatic vehicle tracking system by using low-cost cameras to acquire the traffic data and embedded systems to process the data.

Tracking using a low-cost camera and an embedded system has received considerable research attention due to its wide applications in Internet of Things (IoT). Nevertheless, there are two main challenges of visual object tracking in IoT applications: 1) the low-cost camera may only produce low-frame-rate videos with low resolution [3] and 2) the limited storage and processing power of an embedded system can only process low-frame-rate videos. Based on the literature [4], low-cost cameras can produce 1 GB to 10 GB of data in a single day. Even though there are many existing tracking methods, the previously mentioned limitations can make most of these tracking methods useless in ITS implementation. A common solution is to reduce the video frame rate to decrease the computational burden of performing object tracking. However, this solution will cause the movement of the target objects seem faster on the observer's side. Hence, we need to modify the tracking method to solve the challenges in fast motion.

In this study, we formulate our main concern as developing a tracking method to handle the fast motion challenges.

Wu et al [5] defined the fast motion challenge as the motion of an object that is larger than 20 pixels. We use this definition as the base standard in our experiments, and then experiment with larger threshold values. We also exploit the fact that the video frame rate is related to the speed of the tracked object [6]. Therefore, to analyze the performance of the tracker, we first speed up the video frame rate by manipulating the stride of a certain video sequence. Then, we evaluate the tracker performance to track the targeted object in the manipulated video sequence. We also calculate how many pixels the object target moves in the manipulated video sequence as a comparison to the base standard.

The backbone of the tracking algorithm is the observation model and the transition model [7]. The fast motion challenge is more related to the transition model, which tries to capture the dynamic motion of the targeted object. However, much of the research in visual object tracking has focused on the observation model of the targeted object. Consequently, the fast motion challenge has been much less investigated. Del Bimbo and Dini [8] integrated the first-order dynamic model in a particle filter (PF)-based tracker. Guo et al [9] used a modification of the PF, which is inspired by an insect's vision, to increase the accuracy of motion estimation. They successfully developed a more adaptive transition model to predict object movements more accurately by modeling the light distribution of a moving image.

For diagnosing the strengths and weaknesses of visual trackers, Wang et al [7] decomposed a tracker into the following five main parts: transition model, feature extractor, observation model, model updater, and ensemble post-processor. They found that setting the parameters in the transition model properly is crucial to achieving a good performance in the fast motion challenge.

In contrast, the transition model has a close relation with the occlusion challenge. Gunawan and Wasito [10] proposed a non-retinotopic PF algorithm to overcome the occlusion challenge. This algorithm monitors the reliability level of the previous object tracking. When the quality level of tracking reliability falls below a certain threshold, the algorithm will modify its motion dynamics. Related to the occlusion in vehicle movement, Yildirim et al [11] modified the PF using information of vehicle angles for assigning weights to the particle. Particles moving in the direction of the vehicle will receive a higher weight.

To solve the fast motion challenge, we propose a tracking method called, dynamic swarm particle (DSP), where the term "particle" is related to the PF in a Monte Carlo simulation [12], a framework that can effectively solve the nonlinear dynamics and non-Gaussian distribution problems in visual object tracking. Meanwhile, the term "swarm" refers to the particle swarm optimization (PSO) algorithm [13], which seeks the optimal candidate solution heuristically. To design a tracking system for overcoming the fast motion challenges, we employ our previous results [14] as an observation model to handle the changes in the object's appearance. This observation model is based on the state-of-the-art object recognition task using deep learning techniques, which can learn the changes in the object's appearance automatically. The fundamental concept of our method is to exploit the continuity of vehicle dynamic motions by creating dynamic models based on PSO. In this paper, we focus on manipulating the transition model based on the continuity of motions.

The rest of this paper is organized as follows. Section 2 describes the visual object tracking terminology in the transition model and the observation model. It also explains the conventional PF and PSO methods. In Section 3, the proposed method, DSP, will be explained. Section 4 depicts the experiment setup. Finally, Section 5 discusses the results of the proposed method as compared to the existing methods, and section 6 presents our conclusion.

2 | RELATED WORK

2.1 | Visual object tracking

Object tracking is a process that estimates the state variable x_t based on a set of observations $z_{1:t-1}$ in a discrete time t. In the general framework, object tracking comprises the following four parts: the input frame; the search mechanism, which includes the transition and observation models; and the final prediction [7].

1. Input Frame: In the first frame, the target is appointed first. Alternatively, the target can be obtained by object detection. In this study, the initialization process is conducted by providing a bounding box on the tracked object. Furthermore, the object will be represented as a state variable in a transformation space.
2. Search Mechanism: One of the most important components of object tracking is search mechanism, which estimates the object's location. Two approaches can be used as the search mechanisms: deterministic and stochastic approaches. The deterministic approach considers the object tracking problem as an optimization problem, which can be solved using the gradient descent or optimization algorithms. In contrast, the stochastic approach is more widely used for object tracking. This method utilizes Bayesian frameworks such as PF and Markov chain Monte Carlo [5]. Furthermore, the search mechanism consists of two parts:
   - Transition Model: This model represents movements among the state variables in a certain space [12].
   - Observation Model: This model aims to describe the target object, and begins by selecting the features that can distinguish objects. Afterward, the object model is built based on feature selection [15]. Several previous studies have used various observation models such as

the raw grayscale, raw color, haar-like feature, HOG, and HOG plus raw color. Recently, Kang et al [16] proposed selected invariant feature as the observation model. A survey conducted by Wang et al [7] showed that more complex feature extraction methods produce more accurate tracking. The rapid development of the deep learning algorithm has reached state-of-the-art level in feature extraction. Therefore, several previous studies have utilized deep learning in developing object tracking algorithms, such as Wang and Yeung [17], Gunawan and Jatmiko [18], Zhang et al [19], and Ma et al, [20].

3. Final Prediction The result of object tracking is a predicted bounding box of objects on the n^{th} frame.

2.2 | Particle filter

A PF is widely used as a transition model. Wang et al [7] compared the PF with other transition models such as sliding windows and radius sliding windows, and found that the PF obtained the highest precision because it maintains a probabilistic estimate on each frame. Each frame can find several target object candidates. PFs can keep candidates so that the tracker can regain the object when the tracking fails. Additionally, PFs can accommodate the transformation of the target object, such as scale, aspect ratio, rotation, and skewness.

A PF uses a set of weighted particles to estimate the posterior distribution. This approach is utilized to estimate the nonlinear and non-Gaussian distributions in Bayesian estimation. A PF is the most common formulation of the sequential importance sampling method. For details on the PF, please refer to [12].

Based on the previous research, a PF gives a poor result while tracking fast motion objects. It achieves a success plot and a precision of 0.458 and 0.623, respectively, which is worse than the precision obtained by the sliding window and radius sliding windows [7]. A traditional PF uses random Gaussian for spreading particles. If some objects have similar appearance, the particles will be easily trapped in the local optima. Furthermore, particle degeneration will occur as a consequence of eliminating the low-weighted particles due to a less-precise proposal distribution. This leads to tracking loss. Particle degeneration is solved using the resampling procedure. However, if the tracker loses diversity among the particles, then the tracker fails to catch the dynamic movement of the target.

2.3 | Particle swarm optimization

Particle swarm optimization is used to optimize a solution by seeking the optimal value developed by Kennedy [13]. It utilizes a population of particles with a metaheuristic procedure to search for an optimum value by trial and error. This procedure has a trade-off in randomness and local search. There is no guarantee that the PSO will be able to obtain the best solution. In addition, the solution is dependent on the searching time. PSO comprises two phases: exploration and exploitation. In the exploration phase, particles are spread so that they can explore the search space. This phase reduces the risk of particles to be trapped at the local optimum, which however results in a slower convergence rate. In the exploitation phase, particles are spread only in a local area that finds the best solution at that time. This phase aims to obtain the optimal solution to achieve a higher convergence rate with the risk of being trapped in the local optimum. This causes the solution to be dependent on the starting point.

There are several tracking methods that are used in combination with the Bayesian framework with optimization. Guo et al [9] put particles into an area with the highest posterior value, and did not distribute them in the sampling-importance resampling area, such as the standard PF. Xiaowei et al [21] performed a self-adaptive crossover and mutation to produce new particles in large quantities. Walia and Kapoor [22] proposed methods of evolutionary PF using cuckoo search to overcome the problem of decreasing particles in the standard PF. The research claimed to be more reliable and efficient in addressing the issue of scaling and rotational errors compared to the standard PF and PF with PSO.

3 | DSP TRACKER

This paper proposes a new tracking method, called DSP. We proposed a dynamic model using the velocity of the target object, so it brings out the term "dynamic." DSP is shown in Figure 1. DSP is a free model-based and short-term tracker

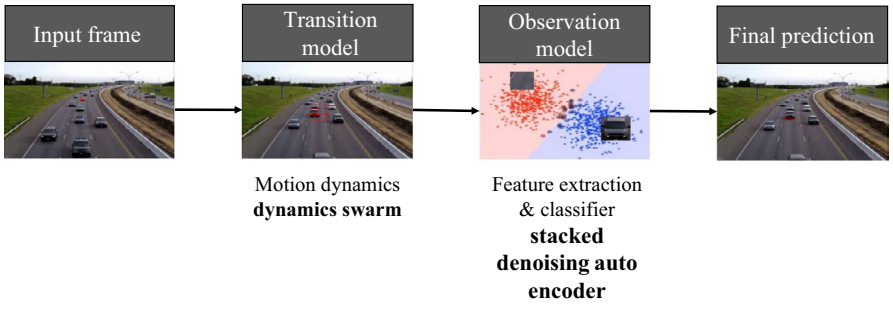

FIGURE 1 Proposed method, dynamic swarm particle

that processes a sequence of frames. The object target is determined in the first frame. Afterward, the tracker depends on that frame. DSP also does not redetect while losing the object.

3.1 | Affine parameter as a particle representation

In visual object tracking, the target object can encounter several transformations in the image frame, such as scaling, rotation, reflection, shearing, and translation. According to [23], the affine representation can track the object's shape more precisely compared to the representation by vector position. By using affine representation, we can recognize the changes in objects in an image frame, as illustrated in Figure 2, where the black car is the initial shape of the target object and the blue or green parts are the possibility of the object's transformation due to its movement. Therefore, we use 2D affine groups as the target object representation to anticipate these transformations.

For our proposed algorithm, DSP, we utilize affine representation based on Lie group Aff(2), which is similar to our previous research [18], where this approach was called the geometric transformation method. For the affine transformation class, geometric transformation can be visualized in six modes, as seen in Figure 2. In vehicle tracking, the perceived vehicle movements depend on the direction of the vehicle to the camera's angle of view. When a vehicle moves away or closer to the camera, it appears to be smaller or larger, respectively. These changes can be represented using similarity transformation, which is mode E_1, as seen in Figure 2. When a vehicle translates on the x or y-axis, it will change its position in the image frame. These x and y translations can be represented as modes E_5 and E_6, respectively. These modes are important to be anticipated in the fast motion challenge. If there is a change in the camera's viewpoint, the perceived vehicle in the image frame can be deformed. This deformation can be represented by modes E_2 and E_4, as seen in Figure 2. If there is an unusual vehicle motion that involves rotation, which might occur when the vehicle slips or crashes, it can be represented using mode E_3. All six modes seen in Figure 2 will create the general affine transformation.

3.2 | Proposed transition model

As stated in the Introduction, we would propose a tracking method to deal with the fast motion challenge. The main idea is to manipulate the transition model by exploiting the continuity of vehicle dynamic motions and integrating it into the PSO algorithm. Physical phenomena such as vehicle traffic can be modeled if we can assume that it is a continuum, meaning that the phenomenon is distributed continuously and can be divided into infinitesimal parts with uniform properties. According to [24], the movement of cars can be effectively modeled as a continuum. One of the model that is very effective is the cellular automaton model [25,26], which uses discrete variables to represent the traffic dynamical system. In this approach, the road is divided into road part length Δx and the time is discretized into Δt. In addition, the vehicle dynamics are modeled in the iterative form as follows: $x_{t+1} = x_t + v\Delta t + \varepsilon$. v is the velocity of the system, which needs to be defined further, and ε is a random error.

By using the above insight, we integrate the velocity concept in our state representation and our geometric model. In the geometric approach, a curve space-like Lie group can be transformed accurately to a linear-space-like Lie algebra. The fundamental idea is that all interactions related to the movement parameters are manipulated in the Lie algebra. Therefore, they can be controlled easily. The end calculation is then transformed back to the Lie group. By using this

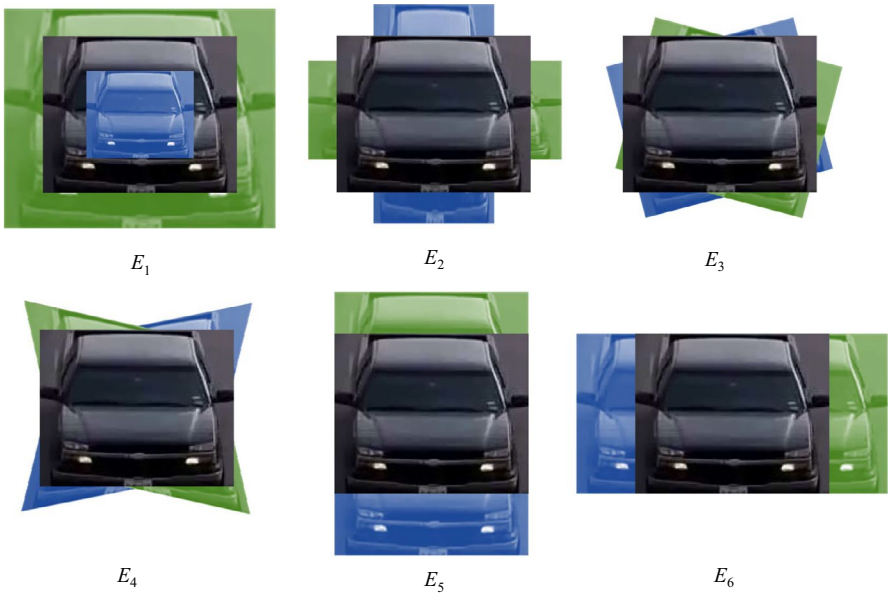

FIGURE 2 Affine mode transformation for car

approach, we can create a movement dynamic model more precisely. In addition, it is easier to control their parameters. Note that the detail of the derivation of the geometric approach can be seen in [18]. Suppose $X_t \in \mathrm{Aff}(2)$ is the state variable that is analog to position x_t in the model below. The velocity in our representation can be defined as shown in (1):

$$V = X_{t-1}^{-1} \times X_t. \tag{1}$$

Because the state representation is located in the Lie group and the model is in the vector representation, or Lie algebra, we have to make a transformation between these two spaces using the relation observed in Figure 3. First, we transform V to the Lie algebra space, with α as a tuned parameter, as shown in (2):

$$A_t = \alpha \log(X_{t-1}^{-1} \times X_t). \tag{2}$$

The random error ε can be modeled in the Lie algebra space as a linear combination of six modes, as shown in Figure 2 and expressed as follows:

$$dW_t = \sum_{i=1}^{6} \epsilon_{t,i} E_i. \tag{3}$$

Finally, the updated model can be expressed as follows:

$$X_{t+1} = X_t \times \exp(A_t \Delta t + dW_t). \tag{4}$$

This formula has already been implemented in our previous research [18] as the transition model of the visual object tracking algorithm.

In the proposed method, we further utilize the continuity of vehicle dynamics by using the PSO algorithm to catch the fast motion. In Figures 4–9, we illustrate the proposed method. To manipulate the particle filter using PSO, we work on a linear space, Lie algebra, and assume that each particle is a PSO particle. Furthermore, we use confidence probability to measure the fitness function of PSO. The confidence level is determined using the observation model in

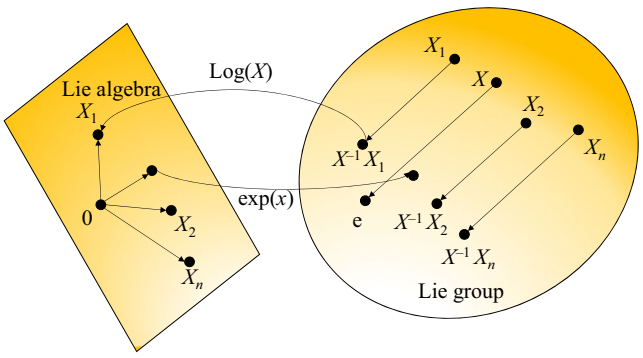

FIGURE 3 Relation of Lie algebra and Lie group

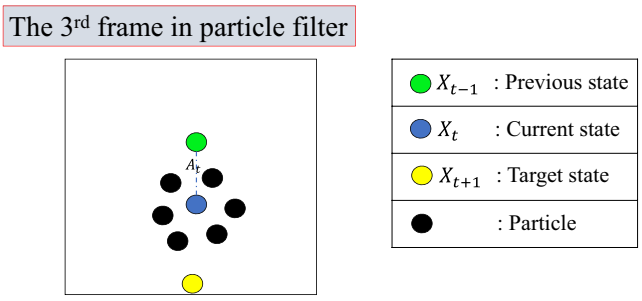

FIGURE 4 Dynamic swarm particle illustration: step 1 (top to bottom)

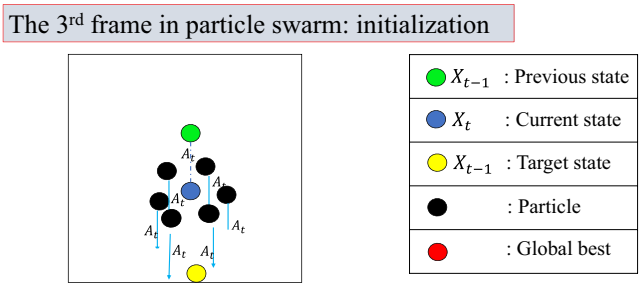

FIGURE 5 Dynamic swarm particle illustration: step 2 (top to bottom)

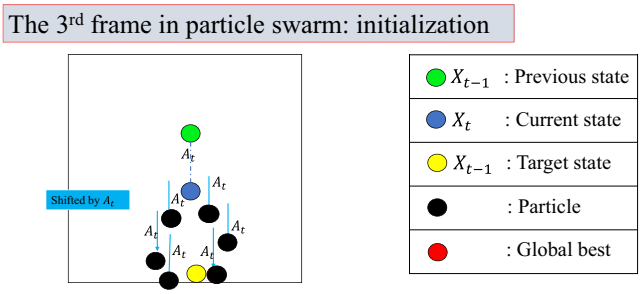

FIGURE 6 Dynamic swarm particle illustration: step 3 (top to bottom)

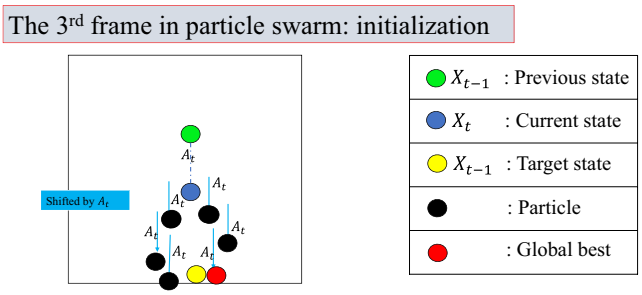

FIGURE 7 Dynamic swarm particle illustration: step 4 (top to bottom)

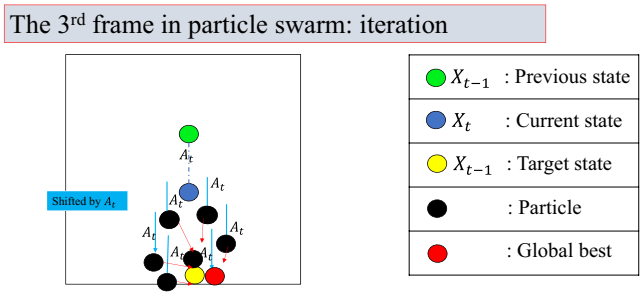

FIGURE 8 Dynamic swarm particle illustration: step 5 (top to bottom)

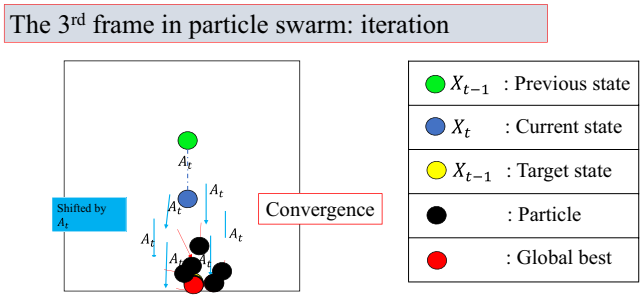

FIGURE 9 Dynamic swarm particle illustration: step 6 (top to bottom)

the range of 0.0 to 1.0. The larger the value of confidence probability, the more precise is the tracking result. Details of the observation model will be explained in the next section.

The last position of the PF phase in Figure 4 will be the initial position of our PSO particle. In Figure 4, the green particles denote the previous state; the blue particles denote the current state; and the yellow particles denote the target state. However, in reality, we do not know where the target state is, but we can calculate the confidence probability to indicate it. Initially, to use the PSO paradigm, we compute the state velocity A_t and shift each particle based on it. The shifting can be seen in Figure 5. After this shifting, we can calculate the local best particles by comparing the confidence probability before and after the shifting, as illustrated in Figure 6. After the shifting, we also compute the global best particle by comparing the best confidence probability among the current state particles, as seen in Figure 7. Next, we use the information of the local and global best particles to modify our version of the geometric PSO algorithm:

$$V_i^{n+1} = K \times V_i^n \times \exp\left(c_1 \times \varepsilon_1 \times \log m\left(Xp_i^n \times (X_i^n)^{-1}\right) + c_2 \times \varepsilon_2 \times \log m\left(X_g^n(t) \times (X_i^n)^{-1}\right)\right), \quad (5)$$

$$X_i^{n+1} = X_i^n + V_i^{n+1}, \quad (6)$$

where V_i^{n+1} is the i^{th} particle velocity on the $n+1$ iteration; V_i^n is the i^{th} particle velocity on the n iteration; X_i^n is the i^{th} particle position on the n iteration; X_i^{n+1} is the i^{th} particle position on the $n+1$ iteration; K is the constriction factor with value between 0 and 1; c_1 is the cognitive factor that influences particle exploitation; c_2 is the social factor that influences particle exploration; Xp_i^n is the i^{th} particle's local best position on the n iteration; X_g^n is the particle's global best position on the n iteration; $\exp(A)$ is the exponential operation of Matrix **A** for transforming to Lie group; $\log m(B)$ is the exponential operation of Matrix **B** for transforming to Lie algebra.

In addition, ε_1 and ε_2 are two random numbers with values between 0 and 1. The parameters c_1 and c_2 are set around 2.05, which is to ensure swarm convergence [27]. We choose the same c_1 and c_2 to create a balance between exploration and exploitation. Furthermore, the two parameters produce a constriction factor K, which has a value of 0.8. All parameters factored in (5) are set constantly for each dataset.

As a result, the velocity and position of the particles are updated based on (5) and (6), respectively. In our experiment, we set the stopping criteria when the confidence probability reaches 0.8. If the particle has not reached this value, it will update the position using the velocity. This searching process is shown in Figure 8. Finally, the PSO iteration will stop when the global best particle is really close to the target, as shown in Figure 9.

3.3 | Observation model

The observation model aims to create a representation for describing the object that we are tracking. An important part of the observation model is the feature extractor, which extracts the characteristics of a raw image into a more informative representation. The observation model is our fitness function that calculates the confidence of each particle based on the representation and produces $p(z_t|x_t)$. The confidence of each particle will determine the particle's best position Xp_i^n and global best position X_g^n, as shown in (5).

We utilize a deep learning method, called stacked denoising auto encoder [17], with image data from Torralba [28]. This approach has created a good image representation, especially for visual object tracking, which can be learned automatically. In this research, we adopt our version of the approach from our previous research [14,18] by adding the extreme learning machine layer to enhance the speed of the observation model (Figure 10).

4 | EXPERIMENT SETUP

4.1 | Data

Based on the object tracking benchmark [5], if an object moves more than 20 pixels ($\tau = 20$), then the object motion is

classified as fast motion. This study uses a vehicle sequence that contains fast motion and low occlusion objects in a highway traffic. We test our method using several selected vehicle datasets, namely car1 and car2, from OTB100 [29] and racing and tunnel from VOT2016/2018 [30]. We also use other highway data, namely car003, motorcycle001, and motorcycle002. The target object is taken from the static and dynamic cameras from various viewpoints. Some vehicles are tracked from the front-side view, while others are tracked from the rear-side view and bird's-eye view. These will create different size changes, where the object will become either larger or smaller.

To evaluate the robustness of DSP, we also create an artificial sequence by eliminating some frames (every two, three, or four frames). The eliminating procedure simulates the fast motion-based technique developed by Roy and Yusuke Inoue [31]. The sequence is divided into three levels of fast motion, as follows:

1. Standard, if the motion of the object ranges from 0 to 40 pixels. It consists of six sequences, namely car1, car2, tunnel, car003fast40, motorcyle001fast40, and motorcycle002fast40.
2. Medium, if the motion of the objects ranges from 20 to 60 pixels. It consists of five sequences, namely car003fast40, motorcyle001fast40, motorcycle002fast40, car003fast60, and motorcyle001fast60.
3. Extreme, if the motion of the object ranges from 20 to 100 pixels. It consists of seven sequences, namely racing, car003fast40, motorcyle003fast40, motorcycle002fast40, car003fast60, motorcyle001fast60, and car003fast100.

In addition, DSP needs to be tested in a non-fast motion sequence, car4, to show that the method is still stable under this condition.

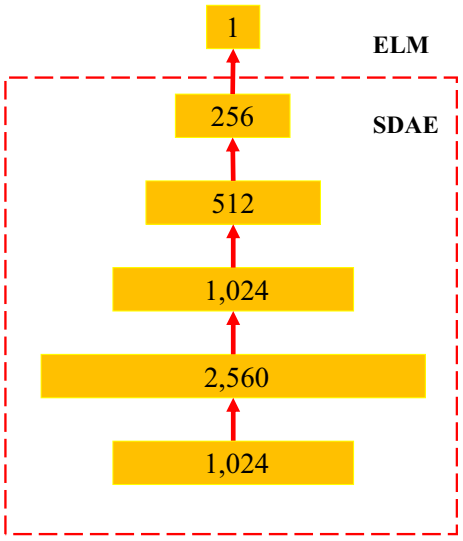

FIGURE 10 Architecture for online tracking (top to bottom)

4.2 | Parameters

In DSP, the number of PF particles is the same as that of PSO particles. This is because DSP transforms the PF particle into the PSO particle. To perform fair tracker benchmarking, DSP will use a single parameter for all data, which are 50 particles with five PSO iterations.

4.3 | Experimental environment

DSP is executed on a device with Ubuntu 16.04.1 Operating System, Intel i5-7400, 3.0 GHz CPU with 24 GB RAM, and an 8 GB NVIDIA GeForce GTX 1070 graphic card. The tracker's performance is evaluated based on the precision and success rate measurement. We use on-pass-evaluation, which evaluates the tracker from the first frame, and then moves on to evaluate all sequences. Furthermore, the tracker's performance is evaluated based on the precision and success rate measurements based on the object tracking benchmark [5]. To provide a fair evaluation, we use a threshold value of $\tau = 20$ pixels [32]. Each tracker will be measured on the overlap value S, which is above the threshold t. This research determines $t = 0.5$ from the range value of t, which is 0.0 to 1.0. The precision and success rates are presented in an area under the curve graph, which is obtained by adjusting the variation in the threshold τ from 0 to 50 and the threshold t from 0 to 1.

5 | RESULT DISCUSSION

5.1 | DSP compared to baseline method

The baseline method of DSP is deep learning tracker (DLT) [17], which uses particle filter as the motion model and deep learning approach for the observation model. Hereafter, Gunawan enhanced DLT using KLD to obtain an efficient tracker [33]. Gunawan proposed a method called geometric deep particle filter (GDPF), which has three variations depending on its transition models: Brownian or random sampling (GDPF-Brow), autoregressive with constant α (GDPF-Fix), and autoregressive with incremental α (GDPF-Inc). Autoregressive means that the method utilizes object velocity information in spreading the particles.

We run all sequences and compare DSP with the baseline method in terms of precision and success rate. Table 1 shows that DSP is superior for all levels of fast motions. DSP obtains a precision above 0.873 for all levels of fast motion, indicating that the modification of the motion model successfully catches the vehicle movement. It shows the capability of transition model modification within the same base method, especially for fast motion tracking. The next position is obtained by GDPF-Inc, which is then followed by GDPF-Fix. These two methods obtain a relatively same result, which is still not much better than the other methods. It depicts that

TABLE 1 Precision result comparison of dynamic swarm particle (DSP) and baseline method

	Level of fast motion		
Method	**Standard**	**Medium**	**Extreme**
DSP	0.946	0.934	0.873
GDPF-Inc [33]	0.572	0.668	0.655
GDPF-Fix [33]	0.373	0.685	0.620
GDPF-Brow [33]	0.468	0.649	0.618
DLT [17]	0.283	0.152	0.148

TABLE 2 Success rate result comparison of dynamic swarm particle (DSP) and baseline method

	Level of fast motion		
Method	**Standard**	**Medium**	**Extreme**
DSP	0.803	0.785	0.733
GDPF-Inc [33]	0.479	0.480	0.473
GDPF-Fix [33]	0.292	0.527	0.477
GDPF-brow [33]	0.358	0.428	0.420
DLT [17]	0.191	0.110	0.107

TABLE 3 Computation speed in fps comparison of dynamic swarm particle (DSP) and baseline method

	Type of fast motion		
Methods	**Standard**	**Medium**	**Extreme**
DSP	21.73	13.06	12.23
GDPF-Fix [33]	17.55	8.82	10.27
GDPF-brow [33]	16.01	10.32	11.73
GDPF-Inc [33]	17.44	9.37	11.46
DLT [17]	9.80	4.62	5.14

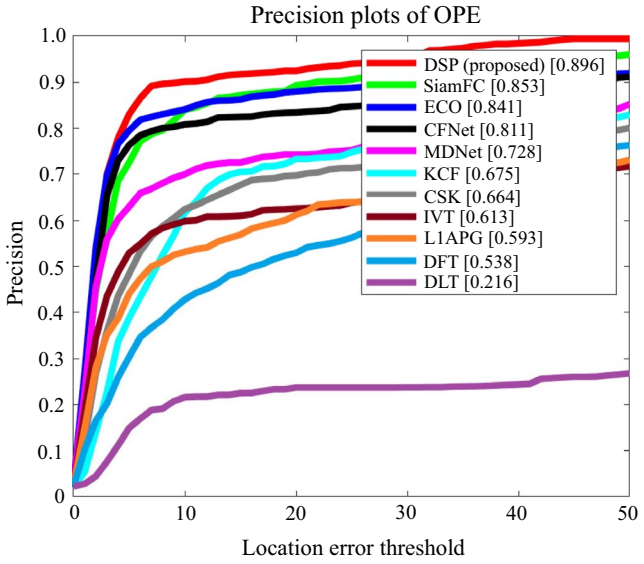

FIGURE 11 Precision result comparison of dynamic swarm particle and state-of-the-art method on fast motion sequence

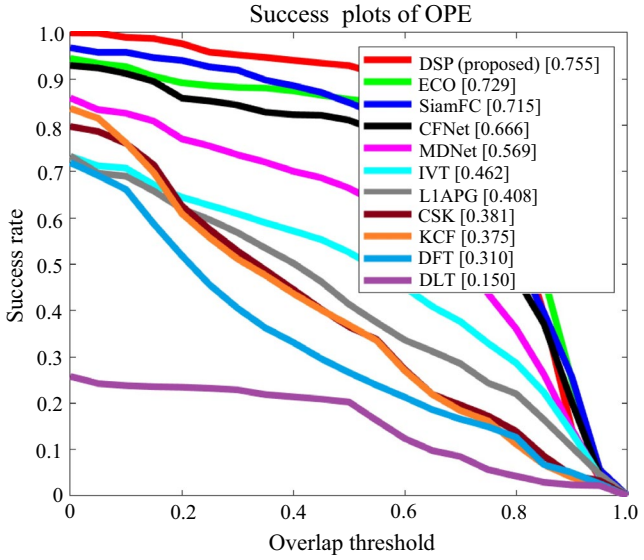

FIGURE 12 Success rate result comparison of dynamic swarm particle and state-of-the-art method on fast motion sequence

solely using the velocity information for important sampling can track the vehicle movement. However, it fails to catch the fast motion with dynamic velocity. The next two methods, GDPF-brow and DLT, with no information velocity lose the object.

The DSP results indicate that the object velocity information is essential for the BPF transition model. The dynamic model using autoregressive gives good effects for tracking fast motion cars. However, the GDPF-Inc result is 0.572 because α increments constantly, and spreads the BPF particles to jump over the target area. This makes the particle fail to track the object. GDPF brow also fails to track the fast motion because of the random transition model, which leads the tracker to be trapped in a similar vehicle that moves around the target. This also occurs in DLT, which cannot catch up with the movement of objects that move further away from the initial position. The precision results of DSP are more accurate than those of the GDPF-based method due to the transition model of DSP, which has an exploration and an exploitation mechanism. Both these mechanisms enable the particles to exchange information to further move into the position with the highest observation.

Table 2 describes the success rate of DSP, which is better than that of others. The exploration and exploitation of the particles cause bounding box tracker changes dynamically. In the standard fast motion dataset, the scale changes smoothly, while in the medium and extreme fast motions, it changes dramatically.

Based on Table 3, DSP has the fastest frame per second (fps) than its baseline method. Although it has an additional

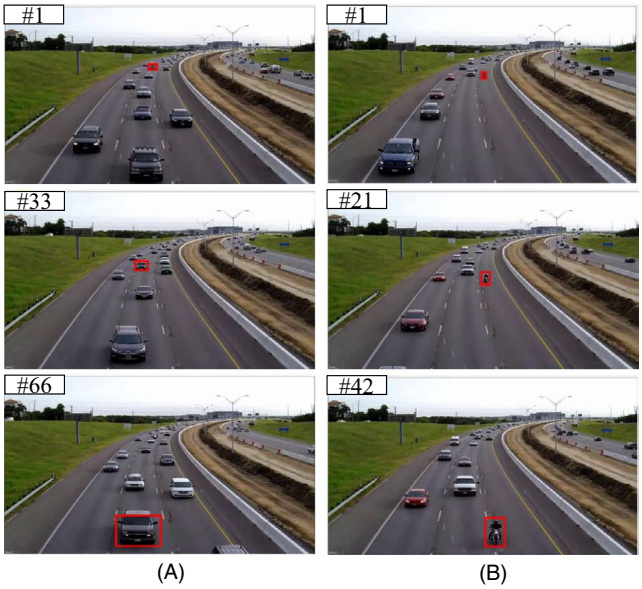

FIGURE 13 Dynamic swarm particle (red bounding box) tracking result (from top to bottom): (A) car003fast40 and (B) motorcycle001fast40

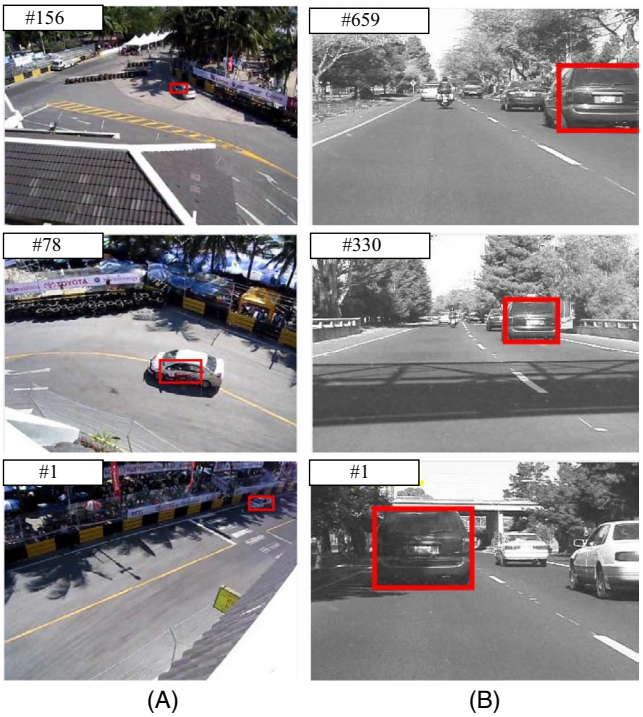

FIGURE 14 Dynamic swarm particle (red bounding box) tracking result (from bottom to top): (A) racing-fast motion and (B) car4-non fast motion

step, the effectiveness of the particle movements in DSP makes the process of finding objects faster.

5.2 | DSP compared to state-of-the-art method

Dynamic swarm particle is also compared with other state-of-the-art tracking methods such as ECO [34], KCF [35], SiamFC [36], CFNet [37], MDNet [38], L1APG [39], DFT [40], IVT [41], and CSK [42]. DSP achieves the best precision in the experiment by using fast motion data with a precision value of 0.896, as shown in Figure 11, which is followed by the SiamFC tracker, which proposes a fully convolutional Siamese Network for spatial searching. SiamFC achieves the second-best precision because it uses not only offline training but also online training to enrich the detection model. The precision rank is then followed by correlation filter-based methods such as ECO, CFNet, and KCF. Another convolution-based method, MDnet, samples candidate windows randomly from the previous target. This approach performs poorly in fast motion because the current target, which usually has a different appearance from the previous target.

Other trackers also face challenges in fast motion tracking. CSK is a tracking-based detection, and hence, it tends to fail if the tracking exists in another similar vehicle. Neither DLT nor IVT can cope with the object's rapid movement. These two methods use the random transition model. The L1APG tracker uses sparse approximation in a template sub-space, and obtains a good result when faced with an occlusion problem. The DFT tracker uses the deterministic gradient descent.

TABLE 4 Comparison of computation speed, in fps, of dynamic swarm particle (DSP) and state-of-the-art methods. All methods are run in CPU, except MDNet, SiamFC, and CFNet, which are run in GPU

	Type of fast motion		
Methods	**Standard**	**Medium**	**Extreme**
CSK [42]	545.10	541.34	518.59
KCF [35]	163.45	292.45	315.48
DFT [40]	31.70	26.11	26.17
IVT [41]	25.34	26.28	25.90
DSP	21.73	13.06	12.23
SiamFC [36]	9.87	7.45	8.00
DLT [17]	9.80	4.62	5.14
CFNet [37]	9.51	7.20	7.78
L1APG [39]	2.22	2.28	2.23
ECO [34]	1.53	2.30	2.45
MDNet [38]	0.64	0.60	0.60

This approach leads the DFT to be trapped in a local minimum problem.

Dynamic swarm particle's success rate is better than that of these other methods, as seen in Figure 12. DSP achieves 0.755 overlap by affine particle representation, which enables the bounding box to scale adaptively. This approach exhibits a significantly increased overlap performance. ECO also

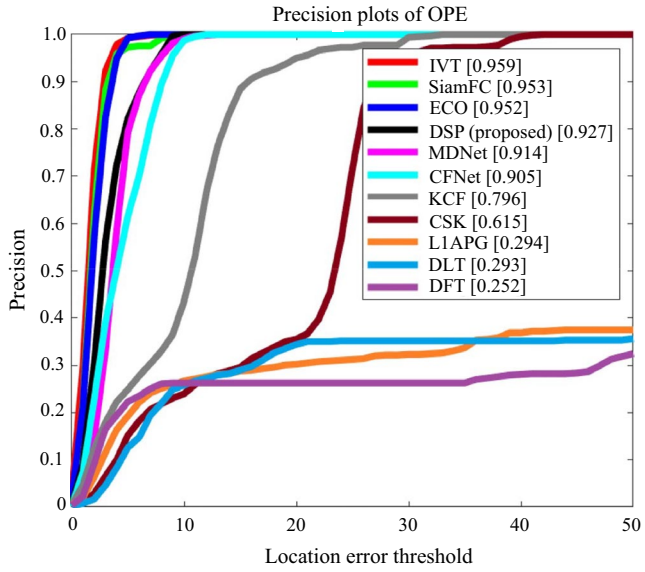

FIGURE 15 Precision result comparison of dynamic swarm particle and state-of-the-art methods in non-fast motion sequence

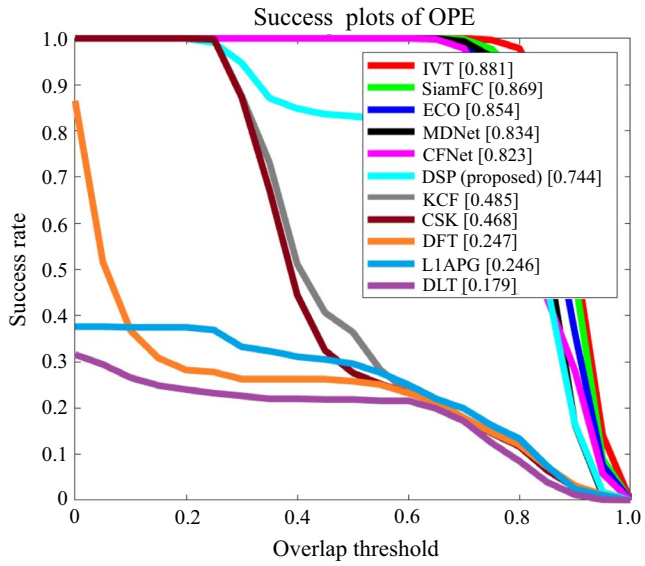

FIGURE 16 Success rate result comparison of dynamic swarm particle and state-of-the-art methods in non-fast motion sequence

exhibits scaling capability, so its overlap reaches 0.729. For the overall result, DSP marginally enhances the performance of the baseline method, DLT. This indicates that DSP is more adaptive and resilient to the datasets, from standard, medium, and even fast motions at extreme levels, where the object displacement can reach 100 pixels between each frame. The tracking results can be seen in Figures 13 and 14.

In terms of computational speed, CSK is the fastest because it uses Gaussian kernel on the Fast Fourier Transform (FFT). It is faster due to the small number of FFTs called. The correlation filter in KCF also performs fast computation but

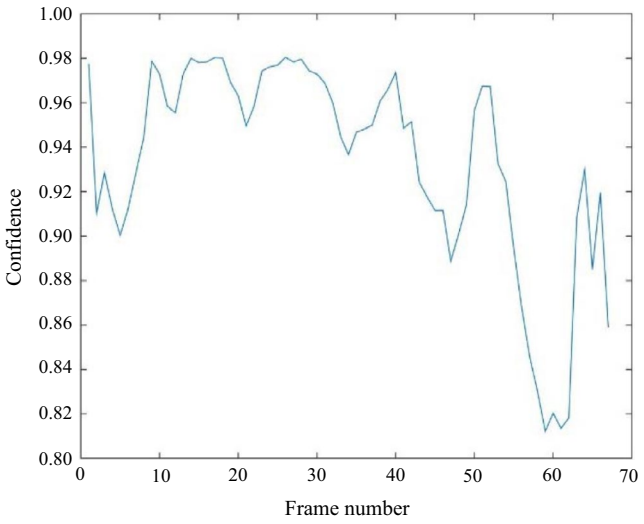

FIGURE 17 Dynamic swarm particle confidence on the object in Car003Fast40 sequence frame by frame

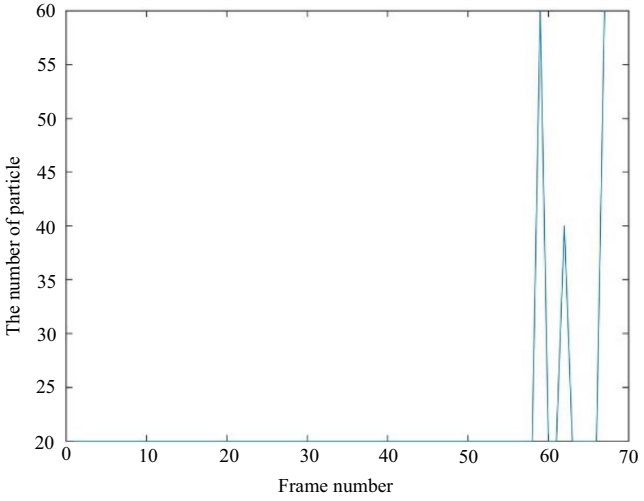

FIGURE 18 Number of particles utilized for Car003Fast40 sequence frame by frame

the bounding box size does not change according to the target. This affects the precision and success rate performance. The proposed method, DSP, reaches the real-time (20 fps [43]) tracker computation, which is 21.73 for the standard fast motion. For medium and extreme fast motions, DSP needs more particle per iteration so the computation is slightly slower. Overall, DSP has a faster computation speed compared to the other method, even though it is tested in the CPU, while other trackers such as MDNet, SiamFC, and CFNet are performed in the GPU. Table 4 shows the performance recap of the trackers in terms of computational time.

Based on the type of fast motion, DSP shows a decrease in computational speed. This is indicated by the drop in FPS, where in a standard fast motion situation, DSP can process an average of 21.73 fps. However, the fps drops to 13.06 and 12.23 in the medium and extreme fast motion

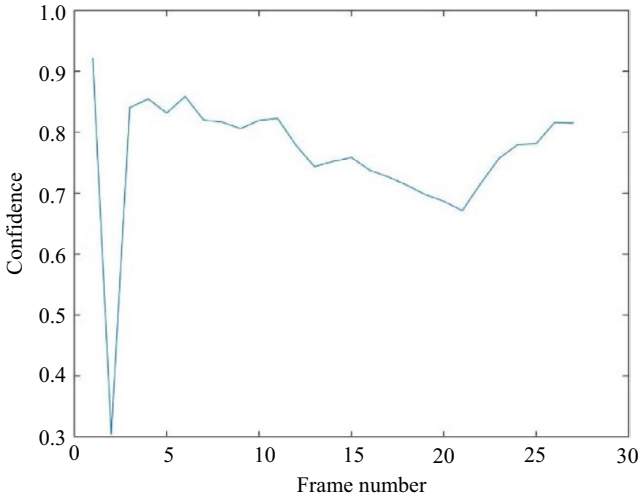

FIGURE 19 Dynamic swarm particle confidence on the object in Motorcycle002fast40 sequence frame by frame

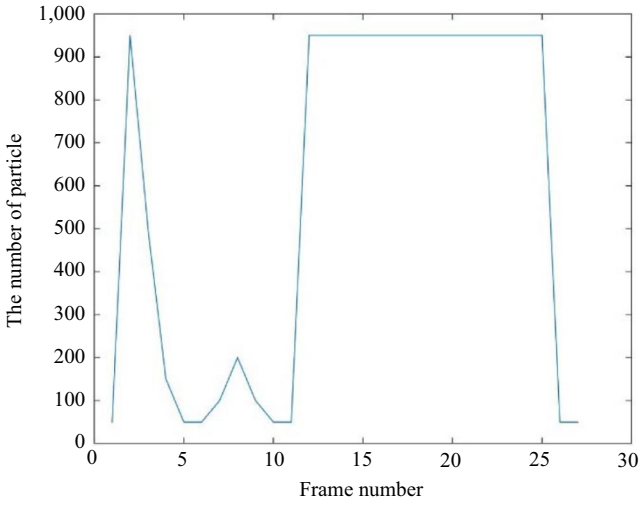

FIGURE 20 Number of particles utilized for Motorcycle002fast40 sequence frame by frame

situations, respectively. In both situations, the target moves further away, and therefore, there is a need to increase the amount and iteration of the particle. The decreased computational speed also occurs on CSK, DFT, SiamFC, DLT, and CFNet. There are also other trackers that exhibit consistent speed (IVT, L1APG, ECO, and MDNet). KCF shows an increase in computational speed because it is not affected by the distance of the target movement. However, the precision and success rate of this tracker are lower than those of DSP.

5.3 | DSP stability for non-fast-motion sequence

One way to demonstrate the stability of a particle is to test the DSP method in a long, non-fast-motion sequence, namely car4, which has 659 frames. Based on Figures 15 and 16,

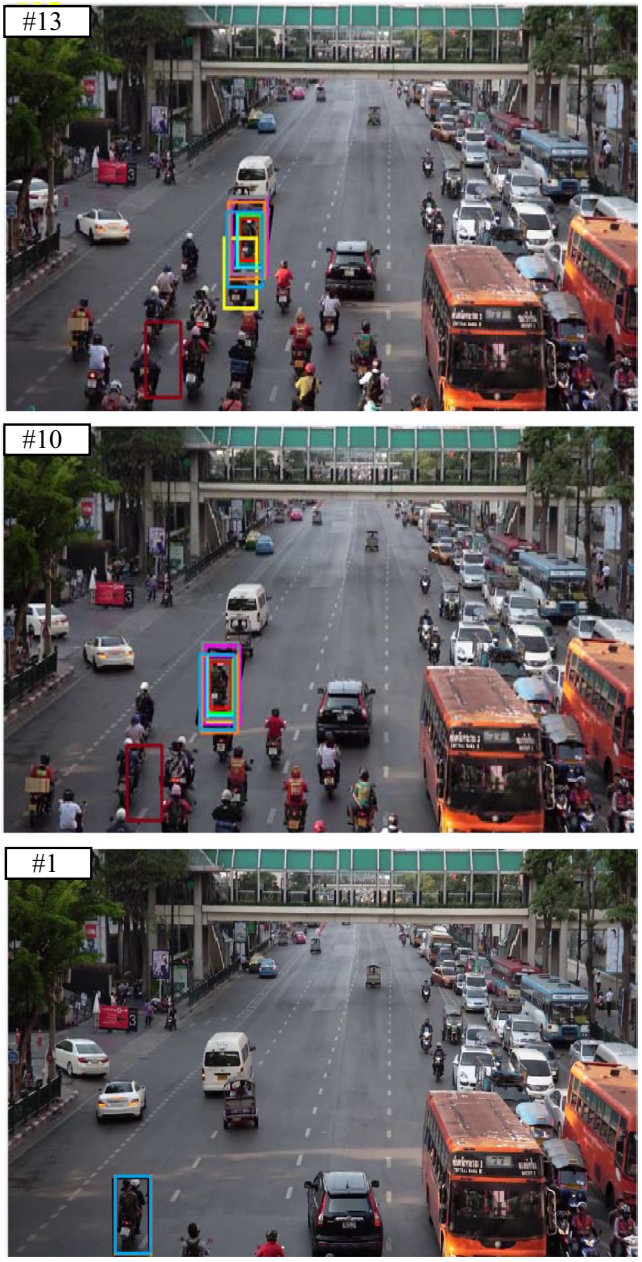

FIGURE 21 Tracker result in motorcycle002fast40 sequence (from bottom to top frame #1 is initialization). DSP = red, ECO = green, SiamFC = blue, DLT = black, KCF = magenta, MDNet = yellow, CFNet = cyan, CSK = gray, IVT = dark red, DFT = orange, L1APG = turquoise

DSP still achieves high results, even though it is not the best. DSP remains in the variance of a stable distribution. This makes DSP to not require resampling. The resampling process is replaced with a search using swarm particles. The tracking result can be seen in Figure 14B

5.4 | How does DSP work?

The performance of the DSP method can be analyzed by evaluating the confidence and number of particles; we use

only 20 particles in the analysis. We evaluate the tracker's confidence frame by frame. One of the sequences that is used is the Car003Fast40 sequence. Figure 18 shows that we only utilize 20 particles in the 1st frame to the 50th frame, which means that it ran on one PSO iteration. In that period, the tracker's confidence is high (Figure 17), above 0.9. Suddenly, in the 55th frame, the confidence decreases. Therefore, the particles iterate again and the number of particles increases to catch the object's movement. Afterward, we obtain a high confidence again. We can see the number of utilized particles frame-by-frame in Figure 18.

By combining the motion velocity in the optimization process, DSP enhances the standard particle filter. In the motorcycle002fast40 sequence, DSP has low confidence, 0.3 (see Figure 19), for the first three frames. This is due to the limited vehicle velocity information. However, we overcame this condition by increasing the number of utilized particles (Figure 20). After the tracker keeps the confidence stable, DSP reduces the number of particles. However, after the 10th frame, the tracker's confidence declines because there are many objects that have a similar appearance. This is shown in Figure 21. Afterward, DSP increases the number of particles to find the object. This strategy escalates the opportunity to find the object even though it is costly. Therefore, our observation model can robustly distinguish different objects.

Note that several trackers lose the target after the 55th frame in the Car003fast40 sequence and the 3rd frame in Motorcycle002fast40 sequence. Therefore, we obtain a new insight for future problems and challenges in tracking vehicles where the conditions are similar to those in these frames. This information is useful for other trackers if they want to enhance the approach. As a new viewpoint, DSP is a novel method that determines the difficulty level of the tracking data based on the number of utilized particles in each frame.

6 | CONCLUSIONS

Based on the assumption of motion continuity, transition models on the PF can be modified using a dynamic model based on PSO. This modification of the PF is called DSP. The transition models in the DSP method are constructed from a combination of the objects' velocity information and PSO algorithms. The DSP algorithm can face the challenges of a fast-moving vehicle tracking while maintaining the ability to handle other challenges. Based on our experiment, DSP achieves better performance in terms of precision and success rate in all categories of fast motion sequence.

ACKNOWLEDGMENTS

The authors thank the Grant of competence of indexed international journal publication No.2523/UN2.R12/HKP.05.00.2016 year 2016-2017, entitled "Smart Security Camera Using Object Tracking and Scene Understanding," by the Universitas Indonesia.

REFERENCES

1. F. J. Martinez et al., *Emergency services in future intelligent transportation systems based on vehicular communication networks*, IEEE Intell. Transp. Syst. Mag. **2** (2010), 6–20.
2. W. Jian et al., *A survey on video-based vehicle behavior analysis algorithms*, J. Multimed. **7** (2012), 223–230.
3. M. Bommes et al., *Video based intelligent transportation systems state of the art and future development*, Transp. Res. Proc. **14** (2016), 4495–4504.
4. E. Toropov et al., *Traffic flow from a low frame rate city camera*, in Proc. IEEE Int. Conf. Image Process., Quebec, Canada, Sept. 2015, pp. 3802–3806.
5. Y. Wu, J. Lim, and M. Yang, *Object tracking benchmark*, IEEE Trans. Pattern Anal. Mach. Intell. **37** (2015), 1834–1848.
6. P. Korshunov and W. T. Ooi, Reducing frame rate for object tracking, *Advances in multimedia modeling* (S. Boll, Q. Tian, L. Zhang, Z. Zhang and Y.-P. Phoebe Chen, eds.), Springer, Berlin, Heidelberg, 2010, pp. 454–464.
7. N. Wang et al., *Understanding and diagnosing visual tracking systems*, in Proc. IEEE Int. Conf. Comput. Vision, Santiago, Chile, Dec. 2015, pp. 3101–3109.
8. A. Del Bimbo and F. Dini, *Particle filter-based visual tracking with a first order dynamic model and uncertainty adaptation*, Comput. Vis. Image Underst. **115** (2011), 771–786.
9. W. Guo, Q. Zhao, and G. Dongbing, *Visual tracking using an insect vision embedded particle filter*, Math. Probl. Eng. **2015** (2015), 1–16.
10. A. A. S. Gunawan and I. Wasito, *Nonretinotopic particle filter for visual tracking*, J. Theor. Appl. Inf. Technol. **63** (2014), 104–111.
11. A. Del Bimbo and F. Dini, *Particle filter-based visual tracking with a first order dynamic model and uncertainty adaptation*, Comput. Vis. Image Underst. **115** (2011), 771–786.
12. W. Guo, Q. Zhao, and G. Dongbing, *Visual tracking using an insect vision embedded particle filter*, Math. Probl. Eng. **2015** (2015), 1–16.
13. A. A. S. Gunawan and I. Wasito, *Nonretinotopic particle filter for visual tracking*, J. Theor. Appl. Inf. Technol. **63** (2014), 104–111.
14. A. A. S. Gunawan, M. I. Fanany, and W. Jatmiko, *Deep extreme tracker based on bootstrap particle filter*, J Theor. Appl. Inf. Technol. **66** (2014), 857–863.
15. A. Yilmaz, O. Javed and M. Shah, *Object tracking: A survey*, ACM Comput. Surv. **38** (2006), 13:1–13:45.
16. K. Kang et al., *Invariant-feature based object tracking using discrete dynamic swarm optimization*, ETRI J. **39** (2017), 151–162.
17. N. Wang and D.-Y. Yeung, Learning a deep compact image representation for visual tracking, *Advances in neural information processing systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger, eds.), Curran Associates, Inc., San Diego, CA, 2013, pp. 809–817.
18. A. A. S. Gunawan and W. Jatmiko, *Geometric deep particle filter for motorcycle tracking: development of intelligent traffic system in Jakarta*, Int. J. Smart Sens. Intell. Syst. **8** (2015), 429–463.
19. K. Zhang et al., *Robust visual tracking via convolutional networks without training*, IEEE Trans. Image Process. **25** (2016), 1779–1792.

20. C. Ma et al., *Hierarchical convolutional features for visual tracking*, in Proc. IEEE Int. Conf. Comput. Vision, Santiago, Chile, Dec. 2015, pp. 3074–3082.
21. Z. Xiaowei, L. Hong, and S. Xiaohong, *Object tracking with an evolutionary particle filter based on self-adaptive multi-features fusion*, Int. J. Adv. Rob. Syst. **10** (2013), 61–71.
22. G. S. Walia and R. Kapoor, *Intelligent video target tracking using an evolutionary particle filter based upon improved cuckoo search*, Expert Syst. Appl. **41** (2014), 6315–6326.
23. J. Kwon, K. M. Lee, and F. C. Park, *Visual tracking via geometric particle filtering on the affine group with optimal importance functions*, in Proc. IEEE Conf. Comput. Vision Pattern Recogn., Miami, FL, USA, June 2009, pp. 991–998.
24. Y. Xue and S.-Q. Dai, *Continuum traffic model with the consideration of two delay time scales*, Phys. Rev. E **68** (2003), 1–6.
25. Q. Chen and Y. Wang, *A cellular automata (ca) model for motorized vehicle flows influenced by bicycles along the roadside*, J. Adv. Transport. **50** 949–966.
26. Y. Luo et al., *Modeling the interactions between car and bicycle in heterogeneous traffic*, J. Adv. Transport. **49** (2015), 29–47.
27. M. Clerc and J. Kennedy, *The particle swarm - explosion, stability, and convergence in a multidimensional complex space*, IEEE Trans. Evol. Comput. **6** (2002), 58–73.
28. A. Torralba, R. Fergus and W. Freeman, *80 millions tiny images: a large dataset for non-parametric object and scene recognition*, IEEE Trans. Pattern Anal. Mach. Intell. **30** (2008), 1958–1970.
29. Y. Wu, J. Lim, and M. Yang, *Object tracking benchmark*, IEEE Trans. Pattern Anal. Mach. Intell. **37** (2015), 1834–1848.
30. M. Kristan et al., *The visual object tracking VOT2016 challenge results*, in Proc. Comput. Vision – ECCV 2016 Workshops, Amsterdam, Netherlands, Oct. 2016, pp. 777–823.
31. A. Royet et al., *Tracking benchmark and evaluation for manipulation tasks*, in Proc. IEEE Int. Conf. Robotics Autom., Seattle, WA, USA, May 2015, pp. 2448–2453.
32. B. Babenko, M.-H. Yang, and S. Belongie, *Robust object tracking with online multiple instance learning*, IEEE Trans. Pattern Anal. Mach. Intell. **33** (2011), 1619–1632.
33. A. A. S. Gunawan et al., *Tracking efficiency measurement of dynamic models on geometric particle filter using KLD-resampling*, Int. Conf. Adv. Comput. Sci. Inf. Syst. **1** (2014), 385–388.
34. M. Danelljan et al., *Eco: efficient convolution operators for tracking*, in Proc. IEEE Conf. Comput. Vision pattern Recogn., Honolulu, HI, USA, July 2017, pp. 6931–6939.
35. J. F. Henriques et al., *High-speed tracking with kernelized correlation filters*, IEEE Trans. Pattern Anal. Mach. Intell. **37** (2015), 583–596.
36. L. Bertinetto et al., *Fully-convolutional siamese networks for object tracking*, in Proc. Comput. Vision – ECCV 2016 Workshops, Amsterdam, Netherlands, Oct. 2016, pp. 850–865.
37. J. Valmadre et al., *End-to-end representation learning for correlation filter based tracking*, in Proc. IEEE Conf. Comput. Vision Pattern Recogn., Honolulu, HI, USA, July 2017, pp. 5000–5008.
38. H. Nam and B. Han, *Learning multi-domain convolutional neural networks for visual tracking*, in Proc. IEEE Conf. Comput. Vision Pattern Recogn., Las Vegas, NV, USA, June 2016, pp. 4293–4302.
39. C. Bao et al., *Real time robust L1 tracker using accelerated proximal gradient approach*, in Proc. IEEE Conf. Comput. Vis. Pattern Recogn., Providence, RI, USA, June 2012, pp. 1830–1837.
40. L. Sevilla-Lara and E. Learned-Miller, *Distribution fields for tracking*, in Proc. IEEE Conf. Comput. Vis. Pattern Recogn., Providence, RI, USA, June 2012, pp. 1910–1917.
41. D. A. Ross et al., *Incremental learning for robust visual tracking*, Int. J. Comput. Vision **77** (2007), 125–141.
42. J. F. Henriques et al., *Exploiting the circulant structure of tracking-by-detection with kernels*, Lect. Notes Comput. Sci. (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7575 LNCS (2012), no. PART 4, 702–715.
43. M. Kristan et al., *The sixth visual object tracking VOT-2018 challenge results*, in Eur. Conf. Comput. Vis. Workshops, Munich, Germany, Sept. 2018, pp. 3–53.

AUTHOR BIOGRAPHIES

Grafika Jati received his BS and MSc in Computer Science from Universitas Indonesia in 2014 and 2016, respectively. Currently, he works as a research assistant at the Faculty of Computer Science, Universitas Indonesia. His research interest includes visual object tracking and autonomous robot.

Alexander Agung Santoso Gunawan received his BS in Mathematics and MSc in Electrical Engineering from Bandung Institute of Technology in 2000 and 2003, respectively. In 2016, he received his Doctoral degree in Computer Science from Universitas Indonesia. Currently, he works as a lecturer. His research interest includes computer vision, machine learning, and intelligent transportation system.

Wisnu Jatmiko received his BS in Electrical Engineering and MSc in Computer Science from Universitas Indonesia in 1997 and 2000, respectively. In 2007, he received his Dr. Eng. degree from Nagoya University, Japan. Currently, he works as a Full Professor at the Faculty of Computer Science, Universitas Indonesia. His research interest is autonomous robot, optimization, and realtime traffic monitoring systems.