Hindawi
Journal of Robotics
Volume 2023, Article ID 9446956, 11 pages
https://doi.org/10.1155/2023/9446956

Research Article

Study on Tracking Real-Time Target Human Using Deep Learning for High Accuracy

Van-Truong Nguyen and Duc-Tuan Chu

Faculty of Mechatronics, SMAE, Hanoi University of Industry, Hanoi 11900, Vietnam

Correspondence should be addressed to Van-Truong Nguyen; nguyenvantruong@haui.edu.vn

Received 7 March 2023; Revised 20 October 2023; Accepted 27 October 2023; Published 17 November 2023

Academic Editor: Bingxiao Ding

Speed and accuracy are important parts of the human tracking system. To design a system that tracks the target human working well in real time, as well as on mobile devices, a tracking real-time target human system is proposed. First, real-time human detection is performed by the combination of MobileNet-v2 and single-shot multibox detector (SSD). Subsequently, the particle filter algorithm is applied to track the target human. The proposed system is evaluated with the different color shirts and complex background conditions. In addition, the system also works with the support of a depth Kinect-v2 camera to evaluate performance. The experiment result indicates that the proposed system is efficient without the impact of colors, background, and light. Moreover, the system still tracks the human when the human has disappeared or the size of the target has a significant change, and an FPS of 12 (Kinect-v2 camera) and 22 (conventional camera) ensures the system works well in real time.

1. Introduction

In recent years, human tracking has become an important part of the computer vision field. This algorithm is used to estimate the position of the target humans in a video sequence [1]. Human tracking has a wide variety of applications in many areas such as self-driving cars, the military, robotics, and many others. The development of deep learning has brought more effective object tracking algorithms. The requirements of the human tracking algorithms are working in total occlusion, complex environments, and so on [2, 3]. They also have to work on mobile devices without changing performance, particularly when deep learning algorithms are required to work in real time. Thus, human tracking is still a challenge that needs to be solved.

Through decades, many algorithms have been proposed to solve tracking problems such as real time, disappear, size change, and accuracy. For example, the CAMshift [4] algorithm is proposed to find the original color of objects and track them. This algorithm has low computation and is easy to implement. However, the algorithm is impacted by similar background colors or texture changes a lot [5]. The Kernel-correlated filtering (KCF) [6] is a good tracking algorithm. KCF possesses the qualities of great robustness, high accuracy, and high speed. However, this algorithm is unstable in illumination, background color, and shape of target human change [6]. The Kalman filter algorithm [7] predicts the position of the target based on previous movement information, but the Kalman filter often is used in a linear system. The particle swarm method with the histogram of oriented gradients (HOG) [8] is the method often used to solve the full occlusion problem. However, the disadvantage of this method is hard to track the target when the target size changes too much. The particle filter [9] is a simple and flexible method. However, when the target is partial occlusion or local appearance changes, the particle filter cannot track the target accurately.

With the standout of deep learning algorithms in recent years, deep learning architects have been used to detect and classify objects. Some methods such as Fast-RCNN [10], Faster-RCNN [11], YOLO [12], SSD [13], and RetinaNet [14] are object detection algorithms with high accuracy and high speed. These algorithms have a common characteristic in applying a convolution neural network to extract features. They offer effective processing models, extracting

features automatically, and improving accuracy dramatically. The above deep learning methods require a high-performance computer because the amount of data needed to compute is very huge. For instance, the number of images for training in YOLO can be up to millions of images. If YOLO, RCNN family, and other deep learning algorithms work on a low-performance computer, they are expensive, slow for training, and too hard for real time.

For human tracking, some methods using both deep learning and particle filters are applied. For example, the YOLO-particle filter [15] is an efficient method for detecting and tracking humans, but it is difficult to run on low-performance devices. The SSD-particle filters [9] are the model with the backbone of VGG-16. This is also a good model, but the SSD model faces difficulty in predicting small object and needs a lot of data to train. In our recent paper [16], the combination of SSD-MobileNet-v2 [17, 18] and particle filter is proposed. This algorithm is applied in the case of target humans having different colors and full occlusion problems. In this paper, we expand the study to evaluate the efficiency of the proposed algorithm in a complex background, in low light, and when combined with the depth Kinect-v2 camera.

In this paper, the particle filter is combined with SSD-MobileNet-v2 to track target humans. The SSD-MobileNet-v2 is applied to determine the presence of humans in a video automatically. Then, the particle filter is adopted to track the target human. The contributions of this paper compared to existing works include the following:

(1) Particle filter based [19] can only track human targets based on particles. It requires manual intervention to determine the initial position and initialize particles, resulting in time-consuming and labor intensive. To determine the initial position of the human target more accurately and adjust the target human position over time, the proposed model supports updating the size of the human target and grabbing the target automatically when the target appears in the camera area.

(2) The proposed algorithm does not need too much memory and operates at high speeds without the GPU when compared with other algorithms such as VGG-16 [20], ResNet-50 [21], and YOLO [12]. The experiment shows that the proposed algorithm adapters in tracking humans with real-time speed (22 FPS).

(3) Unlike other tracking algorithms face difficulties in tracking human targets in complex conditions (occlusion, scale change, color change, scene change) such as YOLOv3-Camshift [22, 23], YOLO-CSRT [24], and VGG16-KCF [25], the proposed system solves full occlusion even if there is a huge change in the size of the target or obscured by a similar human target. Our system also runs well in complex backgrounds and different light conditions.

(4) The proposed model is tested with a Kinect-v2 camera and compared the results with results on a conventional camera. The experimental results show that the proposed model with Kinect-v2 camera is robust in low light conditions and reaches higher accuracy.

The structure of the sections of this paper includes: (1) the pros and cons of traditional algorithms, deep learning algorithms, and the combination of these two kinds of algorithms to achieve a higher efficiency system; (2) describe in detail SSD-MobileNet-v2 and particle filter and their application of them in human tracking; (3) discuss the achieved result of the algorithm; and (4) show the conclusion and give out the task that needs improvement in the future.

2. Materials and Methods

2.1. System Architect. In the literature review, to improve the performance of the particle filter for human tracking in complex conditions, the particle filter is combined with other algorithms such as Mean-Shift [26], Kalman filter [7], histogram [27], SURF [28], and so on. For example, regarding figures in a study by Iswanto and Li [29] and Lin et al. [30], the common characteristics of these algorithms are facing problems such as feature extraction automatically, real time, scale variation, scene change, similar appearance, cluttered background, and so on. In this paper, to overcome the drawbacks of the above algorithms and increase the accuracy in human tracking, the SSD-MobileNet-v2 combined with the particle filter is proposed. SSD-MobileNet-v2 is used to detect target humans automatically, and the particle filter is used for tracking the target human.

The flowchart of the proposed algorithm is shown in Figure 1. First, the SSD-MobileNet-v2 model is used to detect humans. If there is a human target, a bounding box is created for the target, and at the same time ROI is created. In the next stage, the HSV histogram of ROI is analyzed. The particle filter algorithm initializes randomly 500 particles for the bounding box of the human target. SSD-MobileNet-v2 detects all humans in the next frames and then updates the state of each particle. After that, the distances of each particle are calculated based on the HSV histogram and weight. Next, the algorithm estimates the new state of particles based on their weight and uses it as a centroid of predicted the bounding box. Intersection over Union (IOU) values between the predicted bounding box and the other bounding boxes are computed. The bounding box with the highest IOU value is kept. Then, to get the accurate trajectory of the target, the Kalman filter is applied to the centroid of the human target bounding box. Based on value predictions and the new location of the center, the Kalman filter method calculates the center's location. If the distance of all particles is higher than a certain threshold, the algorithm resamples particles having low weight, else reinitialization of all particles. Finally, the proposed system ends if it is a last frame; otherwise, the system repeats from the process of human detection to the last frame.

2.2. SSD-MobileNet-v2 Algorithm. To detect the human target, a deep learning model with six layers is used with the

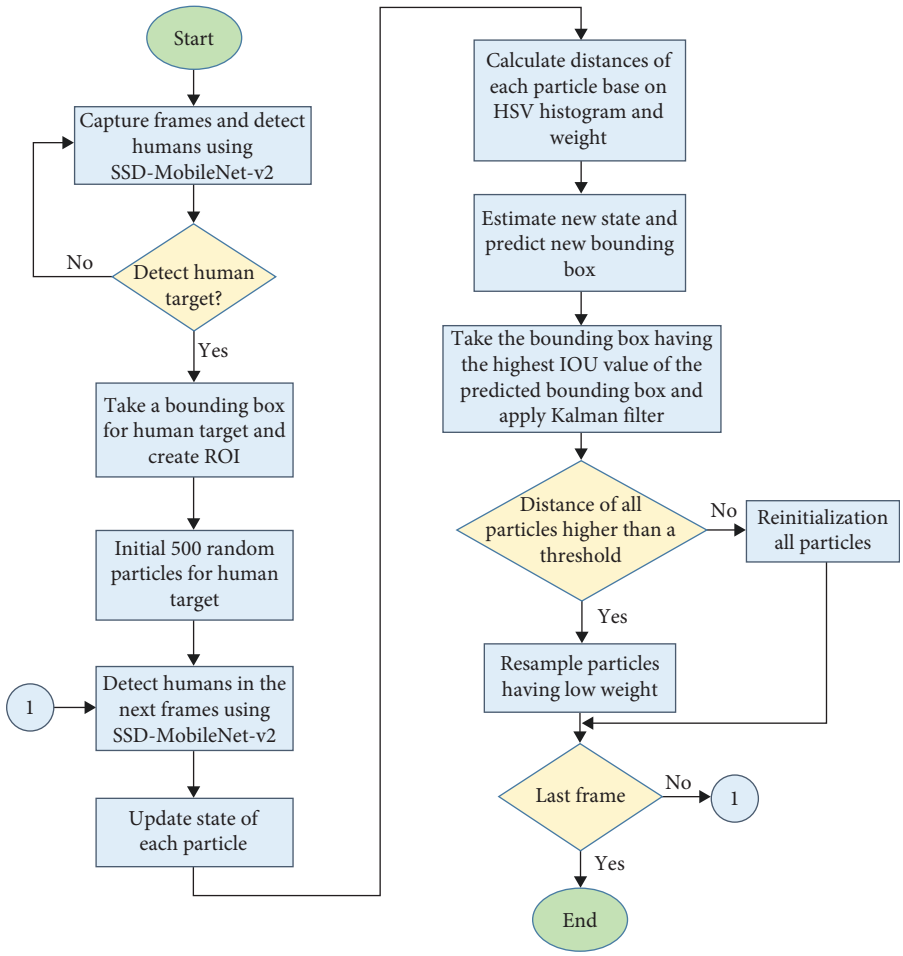


Figure 1: Flowchart of human tracking.

base network MobileNet-v2 [31]. MobileNet-v2 is selected as it can be easily implemented on mobile devices. This model has an architect inspired by the single-shot multibox detector (SSD) model with some modifications to be compatible with MobileNet-v2. Instead of using normal convolutions, separable depthwise convolutions are used to reduce the number of parameters and improve computation time [16, 32].

The proposed method has a base network MobileNet-v2 and SSD extra feature layers. This system uses input images size of $300 \times 300 \times 3$. Base network MobileNet-v2 extracts high-level features from input images. The size of the feature map is reduced, while the algorithm can detect an object at various scales by adding extra layers [16, 33]. Figure 2 shows the architecture of the SSD-MobileNet-v2 algorithm.

The process steps, as shown in Figure 2, are given as follows:

(1) *Step 1*: Read the input images size $300 \times 300 \times 3$.

(2) *Step 2*: The base network is MobileNet-v2 without fully connected layers. This network is used to extract features of images with output size $38 \times 38 \times 512$.

(3) *Step 3*: Apply convolutional layer size $3 \times 3 \times 1{,}024$ for the previous feature map, and the output feature map is obtained with size $19 \times 19 \times 1{,}024$. At the same time, a classifier with a convolutional filter 3×3 is applied to detect objects on the feature map.

(4) *Step 4*: Apply the same process in step 3 for the other feature map, and the feature map size is $19 \times 19 \times 1{,}024$, $10 \times 10 \times 512$, $5 \times 5 \times 256$, $3 \times 3 \times 256$, and $1 \times 1 \times 256$, respectively. The shape of each feature map is based on a convolutional process in the previous layer. A classifier is also used in feature map sizes $19 \times 19 \times 1{,}024$ and $3 \times 3 \times 256$.

(5) *Step 5*: Nonmax suppression algorithm is used to eliminate duplicate detections and select the best bounding box out of a set of overlapping boxes.

In MobileNet-v2, inverted residual and linear bottleneck block [34] are the new layers enabling the model to work well on mobile devices. An inverted residual block is created to reduce the number of parameters compared with the original residual block [35]. The structure of the inverted residual block is built opposite of the original one. First, the input image is widened using 1×1 convolution. This convolution layer expands input feature maps suited to nonlinear activations. Next, a 3×3 depthwise convolution is performed to

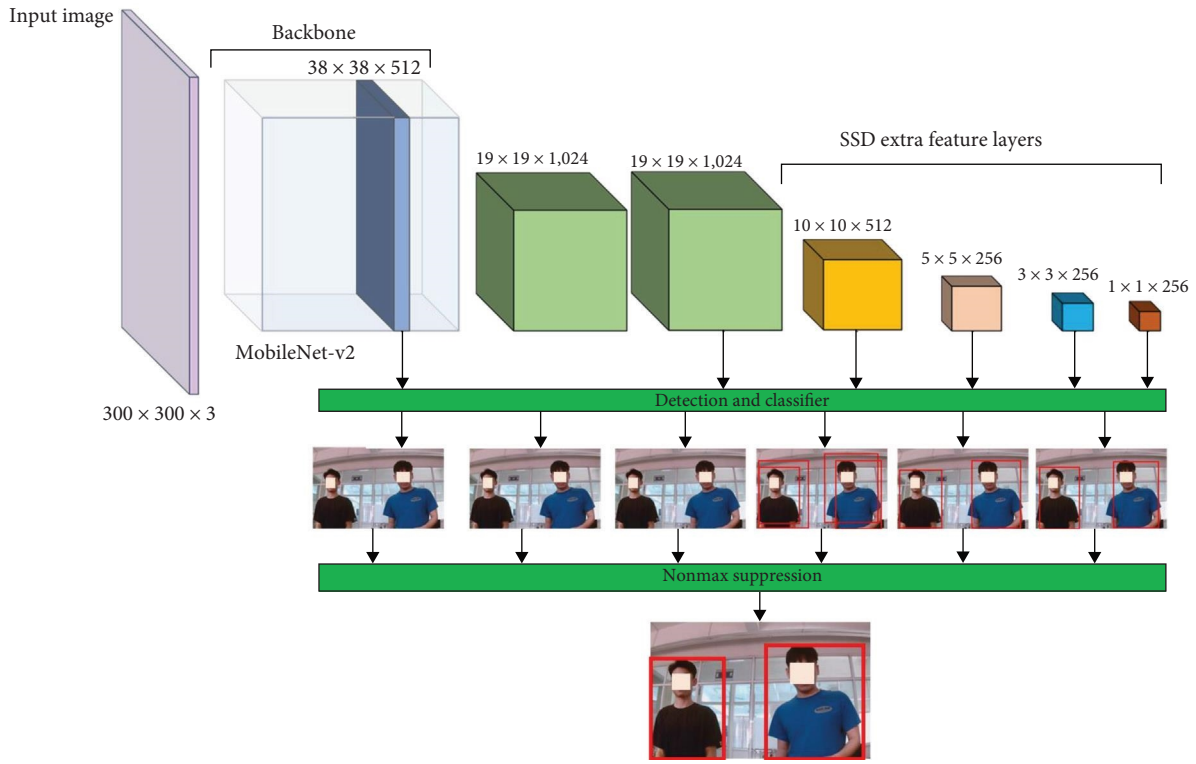


Figure 2: SSD-MobileNet-v2 architecture.

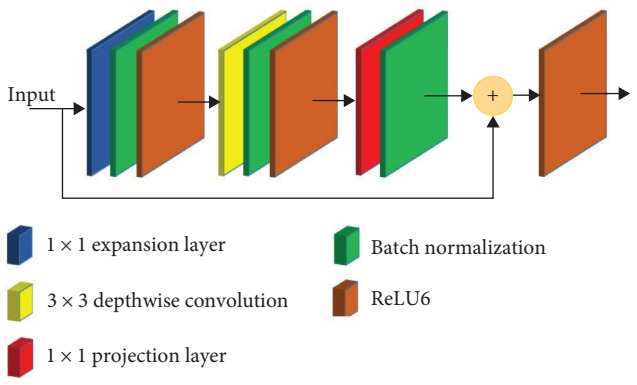


Figure 3: Bottleneck block.

reduce the number of parameters. Finally, the 1×1 convolution is used again to squeeze the network, so that the input image can be matched with the initial number of channels [36]. The batch normalization [37] and ReLU6 [37, 38] are added following each convolution block to reduce the training process of deep neural networks and generalization error [39]. ReLU6 activation is used to discard nonlinear. This activation has ranging linear values between 0 and 6. The structure of the bottleneck block is shown in Figure 3.

This system uses a feature map of the last few layers to predict the location and class of objects and then the output is passed to various convolution layers. Through the network, the size of these layers gradually decreases. In the final, the predictions are combined from each of these convolutional layers. The system outputs the locations and confidence of an object. Each location is evaluated by loss value. Localization loss is defined as follows:

$$L_{\text{loc}}(x, p, g) = \sum_{i \in \text{Pos}}^{N} \sum_{m \in \{x,y,w,h\}} x_{ij}^{k} L_1^{\text{smooth}}\left(p_i^m - \widehat{g}_j^m\right), \tag{1}$$

$$\widehat{g}_j^{cx} = \frac{\left(g_j^{cx} - d_i^{cx}\right)}{d_i^w}, \tag{2}$$

$$\widehat{g}_j^{cy} = \frac{\left(g_j^{cy} - d_i^{cy}\right)}{d_i^w}, \tag{3}$$

$$\widehat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right), \tag{4}$$

$$\widehat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right). \tag{5}$$

Smooth L_1 loss can be interpreted as a combination of L_1 loss and L_2 loss, as follows:

$$L_1^{\text{smooth}}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}. \tag{6}$$

The localization loss is only computed for positive matching ($i \in \text{Pos}$) between the predicted bounding box p and the

ground truth bounding box g. (x, y) is the center of bounding box, (w, h) is width and height of a bounding box, and d is default bounding box. $\sum_{m\in\{x,y,w,h\}} x_{ij}^k L_1^{\text{smooth}}(p_i^m - \widehat{g}_j^m)$ is the total distance between the predicted box p and ground truth box g, x_{ij}^k is the rating for matching between the default bounding box i and the ground truth box j for label k, $(\widehat{g}_j^{cx}, \widehat{g}_j^{cy})$ are the center of the predicted box compared with the center of the default box (d_i^{cx}, d_i^{cy}), and $(\widehat{g}_j^w, \widehat{g}_j^h)$ are scale values of the width and height of the predicted box compared with the center of the default box (d_i^w, d_i^h).

Confidence loss is defined by the equation as follows:

$$L_{\text{conf}}(x, c) = -\sum_{i\in\text{Pos}}^{N} x_{ij}^k \log\left(\widehat{c}_i^{\text{k}}\right) - \sum_{i\in\text{Neg}} \log(\widehat{c}_i^0), \tag{7}$$

where c is the predictions for the probabilities of belonging to different object classes, x_{ij}^k is the rating for matching between the default bounding box i and the ground truth box j for label k, $\widehat{c}_i^{\text{k}}$ is the predicted confidence score of class k for the ith ground truth object, and $\widehat{c}_i^0$ is the predicted confidence score of the class 0.

The final loss function is computed as follows:

$$L(x, c, p, g) = \frac{1}{N}\left(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, p, g)\right), \tag{8}$$

where N is the number of default boxes that match the ground truth, $L_{\text{conf}}(x, c)$ is confidence loss, and $L_{\text{loc}}(x, p, g)$ is location loss.

After prediction, there are a lot of bounding box predictions overlapping. To discard superfluous bounding boxes, the nonmax suppression algorithm is applied [31]. Then, all boxes have confidence and IOU value below a probability bound, these boxes are discarded.

2.3. Particle Filter Tracking Algorithm. A particle filter is used for tracking the human target [40]. This method finds the target human position by using random particles. Then, particle weights are computed for each particle based on its accuracy. The probability value to determine the actual target position is described by these particles and their weights in a region of state space [40]. Each particle is defined by:

$$s = \left[x, y, \frac{dx}{dy}, \frac{dy}{dt}\right]^T, \tag{9}$$

where (x, y) are the coordinates of the center of rectangle boxes and $(\frac{dx}{dt}, \frac{dy}{dt})$ are the velocities.

The initial step defines N as the number of particles. All particle coordinates are chosen at random within the bounding box (x_0, y_0, W, H) of the human target.

$$s_0^i = [x, y, 0, 0]^T, \tag{10}$$

with $i = 0, 1, 2\ldots N, x \in (x_0, x_0 + W), y \in (y_0, y_0 + H)$.

A linear differential equation is used to update all particles in each frame as follows:

$$s_t^i = A s_{t-1}^i + w_{t-1}^i, \tag{11}$$

where s_{t-1}^i is the state of each particle previously, w_{t-1}^i is an array of Gaussian random variables, and A is the transition matrix given as follows:

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{12}$$

Each particle weight is used to evaluate its effectiveness. By comparing the similarity between the HSV histograms of the rectangle box on the particle and the template bounding box on the human target, this weight is evaluated. This similarity is calculated by using the Hellinger distance as follows [41]:

$$d_t^{(i)} = \sqrt{1 - \frac{1}{M\sqrt{\overline{H}_1\overline{H}_2}} \sum_{u=1}^{M} \sqrt{H_1(u)H_2(u)}}, \tag{13}$$

$$\overline{H}_k = \frac{1}{M}\sum_{u=1}^{M} H_k(u), \tag{14}$$

where H_1 and H_2 are two HSV histograms which are compared and M is the total number of bins in a histogram.

In this paper, the HSV histogram is presented in $8 \times 8 \times 4$ bins to get the best results. Each particle's weight is calculated using distance values determined by the following equation as follows:

$$\pi_t^i = \pi_{t-1}^i \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(d_t^{(i)}\right)^2}{2\sigma^2}\right), \tag{15}$$

where σ is the standard deviation of d_t and π_{t-1}^i is the particle's previous weight.

The following step is necessary to normalize these weights to evaluate which has a higher chance of appearing. This is the final estimated state given as follows:

$$\overline{s} = \sum_{i=1}^{N} s_t^{(i)} \pi_t^{(i)}, \tag{16}$$

where $s_t^{(i)}$ is the state of the ith particle at t and $\pi_t^{(i)}$ is the weight of the ith particle at t.

Particles are sampled again in every new frame [42]. Just particles having low distances are kept and all others are discarded. The weight value is assessed by a threshold and in this paper, the weight threshold is 0.3. Then, new particles surround some particles with the highest weights. All particles are randomly reinitialized if all distances exceed the threshold or the entire human target is obscured.

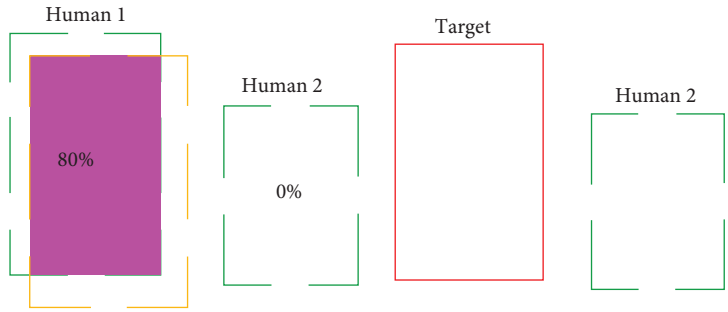


Figure 4: The process of using IOU to update the size of the target.

2.4. Update Size of Target. IOU equation is applied for each bounding box created by the proposed model to calculate the overlap rate of each bounding box (created by the particle filter). Those overlap rates are compared to each other to choose the highest value. The bounding box having the highest rate is used for updating the true size of the target.

Figure 4 shows the process of using IOU to update the size of the target. Figure 4 shows an example of applying IOU to update the size of the target. The green bounding boxes are obtained from the proposed method. The yellow box is a bounding box created by the particle filter with the prior size of the target. The red box is the bounding box of the target after the update.

2.5. Kalman Filter for Accuracy Trajectory. To get the accurate trajectory of the target, the Kalman filter is applied to the centroid of the human target bounding box. Based on value predictions and the new location of the center, the Kalman filter method calculates the center's location [43]. Prediction and correction are divided into two main stages.

The state variable is denoted as $X_T = [x_t, y_t, v_{xt}, v_{yt}]$, and measurement variable is denoted as $Z_T = [x_t, y_t]$. The state variable is updated in each frame based on equation motion without acceleration as follows:

$$\widehat{X}_t = AX_{t-1}, \tag{17}$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{18}$$

where A is the transition matrix and $\widehat{X}_t$ is the value prediction of X_t.

The predictor covariance equation is given as follows:

$$\widehat{P}_t = AP_{t-1}A^T + Q, \tag{19}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{20}$$

where $\widehat{P}_t$ is the value prediction of covariance P_t and Q is the interference factor.

Kalman filter uses three equations in the correction stage, including the Kalman gain equation, state update equation, and covariance update equation. Kalman gain equation is used to correct the stage estimate and covariance estimate. This equation is performed using the following formula as follows:

$$K_t = \widehat{P}_t H^T \left(H\widehat{P}_t H^T + R \right)^{-1}, \tag{21}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{22}$$

$$R = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \tag{23}$$

where H is the measurement matrix and R is the measurement noise covariance matrix.

Based on Kalman's gain and prediction the state $\widehat{X}_t$, the estimated state of the center point can be calculated as follows:

$$X_t = \widehat{X}_t + K_t \left(Z_t - H\widehat{X}_t \right). \tag{24}$$

The covariance also is updated by the following equation as follows:

$$P_t = \widehat{P}_t + K_t H\widehat{P}_t. \tag{25}$$

2.6. Filter Background with an RGB-D Camera. The background significantly influences the accuracy of the system.

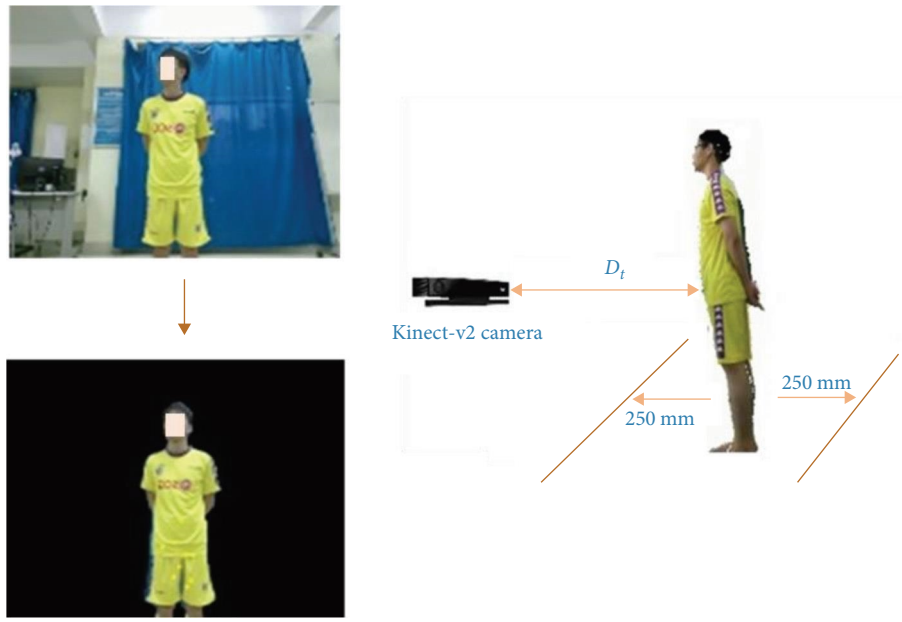


Figure 5: Remove the background on the Kinect-v2 camera.

Therefore, the elimination of the background is an essential step. The removal of the background can be facilitated through the utilization of the RGB-D camera Kinect-v2, as shown in Figure 5. Assume that D_t is the distance from the Kinect-v2 camera to the human target. The D_t value plays the most important role in determining the range for the Kinect-v2 camera. However, this value is not calculated in the current frame. D_{t-1} is used to compute D_t with the velocity of the target.

$$D_t = D_{t-1} + v_t t. \tag{26}$$

T_{below} and T_{above} are below and above the threshold.

$$T_{\text{below}} = D_t - 250\ (\text{mm}), \tag{27}$$

$$T_{\text{above}} = D_t + 250\ (\text{mm}), \tag{28}$$

$$T_{\text{below}} < D_t < T_{\text{above}}, \tag{29}$$

$$D_t - 250 < D_t < D_t + 250. \tag{30}$$

In the next sections, the structure of the sections of this paper includes section 3: showing the results of the proposed algorithm and comparing their performance with other algorithms and section 4: showing the conclusion and giving out the task that needs improvement in the future.

3. Results and Discussion

3.1. Training Deep Learning Model. To train the deep learning model for human detection, 2,500 images are downloaded from Google and 3,500 images are captured by our camera. Transfer learning method and pretrained are used to reduce training time on the COCO dataset combined with our dataset. Training parameter value of SSD-MobileNet-v2 and MobileNet-v2 is shown in Table 1.

Table 1: Training parameter value of SSD-MobileNet-v2 and MobileNet-v2.

Parameters	SSD-MobileNet-v2	MobileNet-v2
Epoch	300	300
Batch size	16	16
Momentum	0.9	0.9
Weight decay	0.95	0.95
Learning rate	0.003	0.001

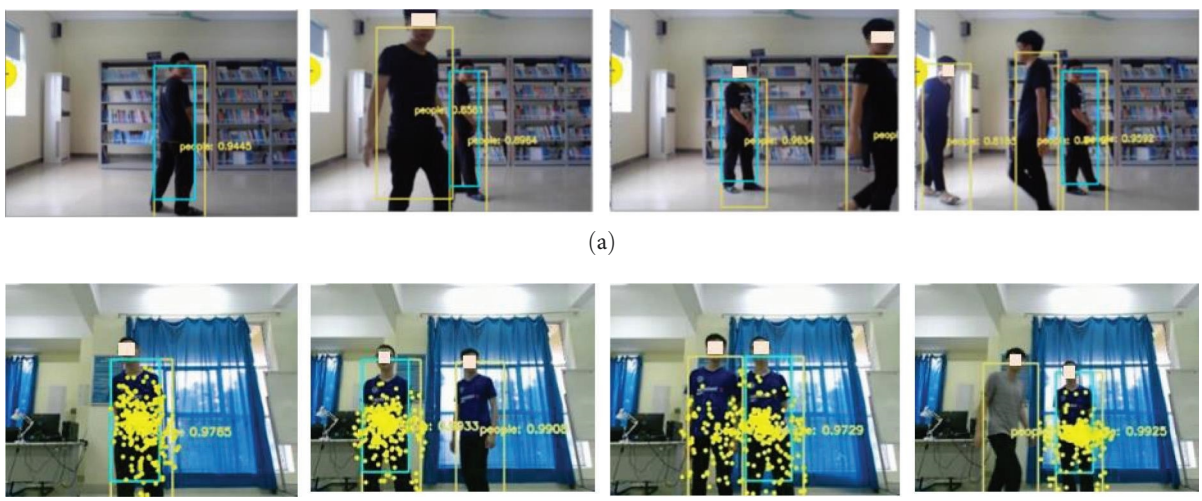
3.2. Running System and Evaluating Results. The proposed system is performed on a computer with 16 GB RAM, Intel Core i7-4800MQ CPU 2.7 GHz x8, and a camera (2 Mpx). To improve processing speed, all images are resized to size 300×300. The proposed algorithm is executed on Python 3.7, OpenCV 4.4 library, and NumPy Library. In addition, the proposed system is also evaluated with the support of a 2 Mpx Kinect-v2 camera (max 30 FPS). When the system uses the Kinect-v2 camera, the RGB image is calibrated and resized to 270×520 to combine with the depth image.

SSD-MobileNet-v2 model is used to detect target humans and then output information is used by the particle filter tracking algorithm. Depending on the number of particles, the proposed system has a different accuracy. The higher the particles, the better accuracy is received while the number of missing frames is reduced. However, the result is still the same once the number of particles reaches a certain threshold. As shown in Table 2, the accuracy is still 97.4% even when there are 1,000 particles.

Table 2: Evaluate the proposed system when using different numbers of particles.

Number of particles	Total frames	Number of misframe	Accuracy (%)	Time per frame
10	677	60	91.1	0.066
50	677	51	92.4	0.07
200	677	41	94.1	0.086
500	677	17	97.4	0.17
1,000	677	17	97.4	0.22

Figure 6: The results in human tracking of the three algorithms.



(a)

(b)

Figure 7: Evaluate the proposed algorithm with a different background. (a) The color of the background and shirts is different. (b) The color of the background and shirts is similar.

By comparing our method with the particle filter method [44], the proposed system still tracks the target human even in case the size of the target has greatly changed. Furthermore, by updating the size of the target, the bounding box of the target is extracted perfectly. The experiments are shown in Figure 6, the light blue bounding box is created by SSD-MobileNet-v2, the red color is created by the particle filter, and the purple color is created by MobileNet-v2.

The algorithm is also tested with some different colors of shirts and different backgrounds to verify the performance. Figure 7 shows the experimental results in the case of the target human using the black and blue color shirt. Yellow bounding boxes are drawn for each human in the frames, and blue boxes are drawn for the human target. The yellow points are particles for tracking the human. The results indicate that even with the same background color, the algorithm remains efficient.

Additionally, to evaluate the effectiveness of the proposed system with support for the Kinect-v2 camera, a variety of colored shirts and low-light conditions are applied. Figure 8 shows the test experiments with blue shirts and yellow shirts. As shown in Figure 8(a), the proposed system is evaluated in low-light conditions with the color of the background and shirts being similar. The yellow bounding boxes are humans who are detected by SSD-MobileNet-v2, and the blue bounding box is the target human who is selected for tracking. The results show that in low-light conditions, the algorithm is still efficient in tracking the human target. Figures 8(b) and 8(d) show RGB images, and Figures 8(c) and 8(e) show masks of color created by using

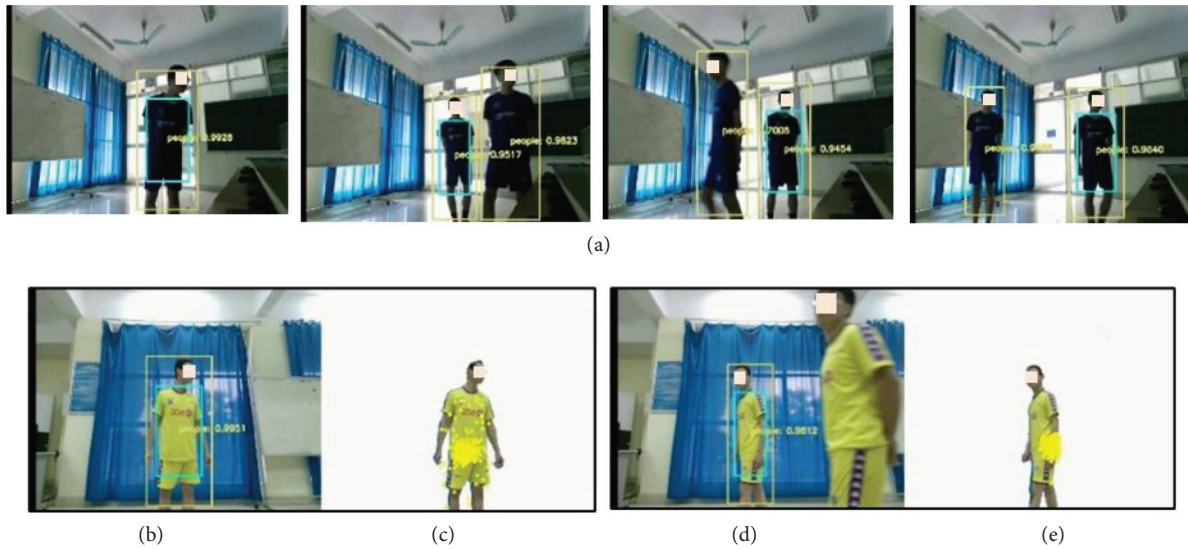


Figure 8: Evaluate the system with the support of a Kinect-v2 camera. (a) The experiment results with blue shirts in low-light conditions. (b–e) The experiment results with yellow shirts.

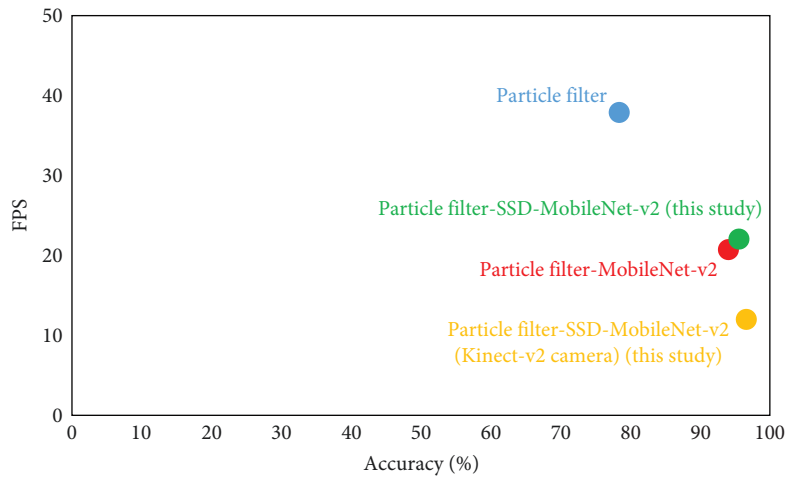


Figure 9: Comparison results of our proposed system with particle filter, and particle filter with MobileNet-v2.

depth images and RGB images of Kinect-v2 camera. The yellow points, as shown in Figures 8(c) and 8(e), are particles used to find the position of the human target.

The accuracy and speed of three algorithms are compared in performance, including particle filter [44], particle filter with MobileNet-v2 [45], and the proposed algorithm. The evaluation involved 458 frames with 500 particles. The FPS of the particle filter algorithm is 38. This is the highest FPS in the three algorithms, but the accuracy only occupies 78.60%. The other algorithms have an FPS range of 21–22, except for the proposed system using a Kinect-v2 camera (12 FPS). However, the proposed system using the Kinect-v2 camera has the highest accuracy (96.94%) followed by the proposed system using a conventional camera (96.06%). The next is a particle filter-MobileNet-v2 [45] with an accuracy of 94.54%. The proposed system has an FPS of 22 (conventional camera) and 12 (Kinect-v2 camera), so the system still works well in real time. The comparison result is shown in Figure 9.

4. Conclusions

This paper presents a method for human tracking by applying a particle filter and SSD-MobileNet-v2 model. The experimental result shows that the proposed system tracks the human in case of the same color, disappearing problem, or the size of the target has a big change. In addition, by using the depth Kinect-v2 camera, the system works better in low-light conditions. By testing and comparing our method with the particle filter and particle filter-MobileNet-v2 algorithm, our method performs better than these algorithms. The accuracy of the proposed algorithm has been greatly improved compared to using only a traditional algorithm particle filter. The tracking speed is 12 (Kinect-v2 camera) and 22 FPS (conventional camera) that are enough for real time. Speed for tracking target humans and tracking more targets at the same time are contents of works to be done in the future.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] C. I. Patel, D. Labana, S. Pandya, K. Modi, H. Ghayvat, and M. Awais, "Histogram of oriented gradient-based fusion of features for human action recognition in action video sequences," *Sensors*, vol. 20, no. 24, Article ID 7299, 2020.

[2] Y. Chen and R. Sheng, "Single-object tracking algorithm based on two-step spatiotemporal deep feature fusion in a complex surveillance scenario," *Mathematical Problems in Engineering*, vol. 2021, Article ID 6653954, 11 pages, 2021.

[3] V. T. Nguyen and D. T. Chu, "Implementation of real-time human tracking system based on deep learning using kinect camera," in *Intelligent Systems and Networks*, N. L. Anh, S.-J. Koh, T. D. L. Nguyen, J. Lloret, and T. T. Nguyen, Eds., vol. 471 of *Lecture Notes in Networks and Systems*, pp. 230–236, Springer, Singapore, 2022.

[4] N. Q. Nguyen, S. F. Su, Q. V. Tran, V. T. Nguyen, and J. T. Jeng, "Real time human tracking using improved CAM-shift," in *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*, pp. 1–5, 2017.

[5] V. T. Nguyen, A. T. Nguyen, V. T. Nguyen, H. A. Bui, and X. T. Nguyen, "Real-time target human tracking using camshift and lucaskanade optical flow algorithm," *Advances in Science, Technology and Engineering Systems Journal*, vol. 6, no. 2, pp. 907–914, 2021.

[6] H. Yan, M. Xie, P. Wang, Y. Zhang, and C. Luo, "Kernel-correlated filtering target tracking algorithm based on multi-features fusion," *IEEE Access*, vol. 7, pp. 96079–96084, 2019.

[7] S. Chen and C. Shao, "Efficient online tracking-by-detection with Kalman filter," *IEEE Access*, vol. 9, pp. 147570–147578, 2021.

[8] H. S. Dadi, G. K. M. Pillutla, and M. L. Makkena, "Face recognition and human tracking using GMM, HOG and SVM in surveillance videos," *Annals of Data Science*, vol. 5, no. 2, pp. 157–179, 2018.

[9] 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*, pp. 7–11, 2020.

[10] S. Wan and S. Goudos, "Faster R-CNN for multi-class fruit detection using a robotic vision system," *Computer Networks*, vol. 168, Article ID 107036, 2020.

[11] C. Cao, B. Wang, W. Zhang et al., "An improved faster R-CNN for small object detection," *IEEE Access*, vol. 7, pp. 106838–106846, 2019.

[12] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023.

[13] S. Zhou and J. Qiu, "Enhanced SSD with interactive multi-scale attention features for object detection," *Multimedia Tools and Applications*, vol. 80, no. 8, pp. 11539–11556, 2021.

[14] H. Zhang, H. Chang, B. Ma, S. Shan, and X. Chen, "Cascade RetinaNet: maintaining consistency for single-stage object detection," 2019, https://arxiv.org/abs/1907.06881.

[15] H. A. Abdelali, O. Bourja, R. Haouari et al., "Visual vehicle tracking via deep learning and particle filter," in *Advances on Smart and Soft Computing*, F. Saeed, T. Al-Hadhrami, F. Mohammed, and E. Mohammed, Eds., pp. 517–526, Springer, Singapore, 2020.

[16] V. T. Nguyen, A. T. Nguyen, V. T. Nguyen, and H. A. Bui, "A real-time human tracking system using convolutional neural network and particle filter," in *Intelligent Systems and Networks*, D.-T. Tran, G. Jeon, T. D. L. Nguyen, J. Lu, and T.-D. Xuan, Eds., vol. 243 of *Lecture Notes in Networks and Systems*, pp. 411–417, Springer, Singapore, 2021.

[17] T.-V. Dang, "Smart attendance system based on improved facial recognition," *Journal of Robotics and Control (JRC)*, vol. 4, no. 1, pp. 46–53, 2023.

[18] T.-V. Dang, "Smart home management system with face recognition based on arcface model in deep convolutional neural network," *Journal of Robotics and Control (JRC)*, vol. 3, no. 6, pp. 754–761, 2022.

[19] X. Li, S. Lan, Y. Jiang, and P. Xu, "Visual tracking based on adaptive background modeling and improved particle filter," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 469–473, 2016.

[20] T.-V. Dang and N.-T. Bui, "Multi-scale fully convolutional network-based semantic segmentation for mobile robot navigation," *Electronics*, vol. 12, no. 3, Article ID 533, 2023.

[21] B. Li and D. Lima, "Facial expression recognition via ResNet-50," *International Journal of Cognitive Computing in Engineering*, vol. 2, pp. 57–64, 2021.

[22] J. Zhao, C. Li, Z. Xu, L. Jiao, Z. Zhao, and Z. Wang, "Detection of passenger flow on and off buses based on video images and YOLO algorithm," *Multimedia Tools and Applications*, vol. 81, no. 4, pp. 4669–4692, 2022.

[23] V.-T. Nguyen, D.-T. Chu, D.-H. Phan, and N.-T. Tran, "An improvement of the camshift human tracking algorithm based on deep learning and the Kalman filter," *Journal of Robotics*, vol. 2023, Article ID 5525744, 12 pages, 2023.

[24] I. C. Amitha and N. K. Narayanan, "Improved vehicle detection and tracking using YOLO and CSRT," in *Communication and Intelligent Systems: Proceedings of ICCIS 2020*, pp. 435–446, 2021.

[25] S. Zhang, W. Li, and R. Wang, "KCF Tracking algorithm based on VGG16 depth framework," *International Journal of Advanced Computer Technology*, vol. 8, no. 2, pp. 05–09, 2019.

[26] J. Yang, S. Rahardja, and P. Fränti, "Mean-shift outlier detection and filtering," *Pattern Recognition*, vol. 115, Article ID 107874, 2021.

[27] R. Yan, "Researches on hybrid algorithm for moving target detection and tracking in sports video," *Cluster Computing*, vol. 22, no. S2, pp. 3543–3552, 2019.

[28] A. Pareek and N. Arora, "Re-projected SURF features based mean-shift algorithm for visual tracking," *Procedia Computer Science*, vol. 167, pp. 1553–1560, 2020.

[29] I. A. Iswanto and B. Li, "Visual object tracking based on mean-shift and particle-Kalman filter," *Procedia Computer Science*, vol. 116, pp. 587–595, 2017.

[30] S. D. Lin, J.-J. Lin, and C.-Y. Chuang, "Particle filter with occlusion handling for visual tracking," *IET Image Processing*, vol. 9, no. 11, pp. 959–968, 2015.

[31] Y. C. Chiu, C. Y. Tsai, M. D. Ruan, G. Y. Shen, and T. T. Lee, "Mobilenet-SSDv2: an improved object detection model for

embedded systems," in *2020 International Conference on System Science and Engineering*, pp. 1–5, 2020.

[32] C. B. Murthy, M. F. Hashmi, and A. G. Keskar, "Optimized MobileNet + SSD: a real-time pedestrian detection on a low-end edge device," *International Journal of Multimedia Information Retrieval*, vol. 10, no. 3, pp. 171–184, 2021.

[33] A. Arcos-Garcia, J. A. Alvarez-Garcia, and L. M. Soria-Morillo, "Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, vol. 316, pp. 332–344, 2018.

[34] G. Li, Y. Yang, and X. Qu, "Deep learning approaches on pedestrian detection in hazy weather," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 10, pp. 8889–8899, 2019.

[35] X. Zhang, Z. Wu, T. Wan, and B. Du, "Smaller and efficient mobile network design for image classification," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, pp. 6531–6544, 2023.

[36] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[37] X. Hu, H. Li, X. Li, and C. Wang, "MobileNet-SSD microscope using adaptive error correction algorithm: real-time detection of license plates on mobile devices," *IET Intelligent Transport Systems*, vol. 14, no. 2, pp. 110–118, 2020.

[38] J. Yu, H. Li, S. L. Yin, and S. Karim, "Dynamic gesture recognition based on deep learning in human-to-computer interfaces," *Journal of Applied Science and Engineering*, vol. 23, no. 1, pp. 31–38, 2020.

[39] Q. Sellat, S. Bisoy, R. Priyadarshini, A. Vidyarthi, S. Kautish, and R. K. Barik, "Intelligent semantic segmentation for self-driving vehicles using deep learning," *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 6390260, 10 pages, 2022.

[40] Q.-B. Zhang, P. Wang, and Z.-H. Chen, "An improved particle filter for mobile robot localization based on particle swarm optimization," *Expert Systems with Applications*, vol. 135, pp. 181–193, 2019.

[41] C. Kim, D. Song, C.-S. Kim, and S.-K. Park, "Object tracking under large motion: combining coarse-to-fine search with superpixels," *Information Sciences*, vol. 480, pp. 194–210, 2019.

[42] R. J. Mozhdehi and H. Medeiros, "Deep convolutional correlation iterative particle filter for visual tracking," *Computer Vision and Image Understanding*, vol. 222, Article ID 103479, 2022.

[43] L. E. Taylor, M. Mirdanies, and R. P. Saputra, "Optimized object tracking technique using Kalman filter," 2021, https://arxiv.org/abs/2103.05467.

[44] H. Li, Y. Liu, C. Wang, S. Zhang, and X. Cui, "Tracking algorithm of multiple pedestrians based on particle filters in video sequences," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 8163878, 17 pages, 2016.

[45] S. S. Bharti, R. Mahajan, S. Bhoriya, and A. Bhat, "Driver's drowsiness detection using MobileNet on web," in *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1229–1233, 2023.