Hindawi
Computational Intelligence and Neuroscience
Volume 2023, Article ID 7974201, 11 pages
https://doi.org/10.1155/2023/7974201

Research Article

Realtime Vehicle Tracking Method Based on YOLOv5 + DeepSORT

Lixiong Lin,[1] **Hongqin He**,[2] **Zhiping Xu**,[1] **and Dongjie Wu**[3]

[1]*School of Ocean Information Engineering, Jimei University, Xiamen 361021, China*
[2]*Zhejiang Dahua Technology Co., Ltd., Hangzhou 310051, China*
[3]*Department of Automation, Xiamen University, Xiamen, Fujian 361005, China*

Correspondence should be addressed to Lixiong Lin; elelinlixiong@139.com

Received 3 January 2023; Revised 20 March 2023; Accepted 22 March 2023; Published 15 June 2023

Academic Editor: S. P. Raja

In actual traffic scenarios, the environment is complex and constantly changing, with many vehicles that have substantial similarities, posing significant challenges to vehicle tracking research based on deep learning. To address these challenges, this article investigates the application of the DeepSORT (simple online and realtime tracking with a deep association metric) multitarget tracking algorithm in vehicle tracking. Due to the strong dependence of the DeepSORT algorithm on target detection, a YOLOv5s_DSC vehicle detection algorithm based on the YOLOv5s algorithm is proposed, which provides accurate and fast vehicle detection data to the DeepSORT algorithm. Compared to YOLOv5s, YOLOv5s_DSC has no more than a 1% difference in optimal mAP0.5 (mean average precision), precision rate, and recall rate, while reducing the number of parameters by 23.5%, the amount of computation by 32.3%, the size of the weight file by 20%, and increasing the average processing speed of each image by 18.8%. After integrating the DeepSORT algorithm, the processing speed of YOLOv5s_DSC + DeepSORT reaches up to 25 FPS, and the system exhibits better robustness to occlusion.

1. Introduction

The increasing number of vehicles has caused great difficulties in traffic management. Vehicle tracking is an application of a target tracking in the field of transportation, which can serve to alleviate the pressure of traffic management [1–3]. At present, the mainstream target tracking method is the discriminative tracking method, which adds the step of target detection and makes the tracking more accurate. Discriminant tracking methods mainly include tracking methods based on sparse representation [4–6], tracking methods based on correlation filtering [7–9], and tracking methods based on deep learning. Li and Huang [10] proposed the TOD (tracking object based on detector) algorithm, which used YOLOv3 for target detection, and tracked the target according to LBP (local binary pattern) features and color histogram. Bertinetto et al. [11] proposed the SiamFC (Siamese fully convolutional) algorithm, which took the target object in the first frame as one input of the SiameseNet and the search area in the subsequent frames as another input and then found out the area closest to the target object to realize the target tracking. However, the target loss can easily happen, while the target size changes. Zhu et al. [12] adopted a distractor recognition model to update the tracking template online, which could well deal with the problems of serious occlusion and appearance change of the target. Li et al. [13] introduced the deep network into the Siamese Net framework and played the role of the deep network through multilayer aggregation.

Multitarget tracking is harder than the single-target tracking. Problems such as appearance similarity among targets, occlusion, and the start and end of single-target tracking tasks pose significant challenges in the field of multitarget tracking. Bewley et al. [14] proposed the SORT (simple online and realtime tracking) algorithm, which used the Kalman filter to predict the tracking frame information of the tracked object in the next frame and performed data associated with the detection frame information in the next frame to achieve multitarget tracking. The algorithm had small memory footprint and high speed, but the accuracy was very low when the target was occluded. Wojke et al. [15] proposed the DeepSORT algorithm based on the idea of

SORT. The algorithm considered the motion information and appearance information in the tracking process and resolved the problem of target occlusion. At present, detection-based tracking algorithms still have many problems, such as a lack of datasets, inaccurate target detection, and insufficient realtime performance,.

Traditional target detection algorithms rely on image features and classifiers such as SVM (support vector machine) [16], Adaboost [17], Random Forest [18], artificially designed color features [19], gradient features [20], and pattern features [21]. Target detection algorithms based on deep learning have stronger adaptability to complex scenes, including target detection methods based on candidate regions and target detection methods based on regression. The representative algorithm based on candidate region is R-CNN (Region-CNN) series [22–24]. Owing to the need to process large number of candidate frames, such methods face the problem of low efficiency and do not have the ability for realtime detection. The regression-based target detection method reduces the steps of generating candidate regions and improves the speed significantly. It has been widely used for developing realtime target detection systems. The YOLO (you only live once) algorithm [25] proposed in 2016 used a grid to divide an image and generated a series of initial anchor boxes in each grid of the image. By learning to fine tune the initial box, the predicted box was generated to be closer to the actual box. The YOLOv2 algorithm introduced batch normalization and used DarkNet-19 as the backbone network, which could dynamically adjust the input and achieve better precision for small targets [26]. On this basis, the YOLOv3 algorithm used DarkNet-53 as the backbone network, introduced FPN (feature pyramid network) structure to obtain feature maps at different scales, and used a logistic classifier to predict the category of targets [27]. The YOLOv4 algorithm added data enhancement and self-antagonistic training methods at the input end [28]. The backbone network used CSPDarkNet53 and improved the loss function of the output layer, which greatly improved the speed and accuracy. The YOLOv5 has the same performance as YOLOv4. However, YOLOv5 is faster and has a detection speed of 140 FPS on Tesla P100. Sasagawa and Nagahara [29] used YOLO to locate and identify objects and proposed a method for detecting objects under low illumination by utilizing the power of transfer learning. Krišto et al. [30] used thermal images on YOLO to improve target detection performance in challenging conditions such as adverse weather, night time, and dense areas. Xiao et al. [31] fused the context information in the YOLO backbone network to avoid the loss of low-level context features, retain lower spatial features, and solve the problem of difficult detection of targets under dim light. Guo et al. [32] designed an improved SSD (single shot multibox detector) detector, which used the method of single data deformation data amplification to transform the color gamut and affine of the original data and could detect targets that were close to each other. To improve feature fusion for small tassel detection, Liu et al. [33] proposed a novel algorithm referred to as YOLOv5-tassel to detect tassels. To enrich feature information and improve the feature extraction ability, Bie et al. [34] proposed an improved YOLOv5 algorithm based on bidirectional feature pyramid network for multiscale feature fusion. Wang et al. [35] proposed a novel vehicle detection and tracking method for small target vehicles to achieve high detection and tracking accuracy based on the attention mechanism. In summary, research based on the improved YOLOv5 algorithm mainly focuses on the accuracy of small object detection, while research on detection speed and occlusion robustness in vehicle tracking still has great research value. The main contributions of this article are as follows:

(1) To solve the problems of large number of vehicles, fast-moving speed of vehicles, substantial similarity of vehicle appearance, and vehicle occlusion in the actual urban traffic scene, the DeepSORT algorithm is used for vehicle tracking, which has better realtime performance and tracking robustness than traditional vision-based vehicle tracking methods.

(2) To reduce the calculation amount of YOLOv5s, reduce its inference time, and improve the operation speed, a YOLOv5s_DSC algorithm with faster inference speed is proposed.

(3) Combining YOLOv5s_DSC with the DeepSORT algorithm, the robustness of occlusion of the proposed algorithm is verified and the realtime performance of the algorithm is tested in the cases of vehicles being occluded by foreign objects or vehicles being occluded by each other.

2. Algorithmic Framework

2.1. Overall Framework. The DeepSORT algorithm adopts a two-stage idea of detection and tracking, using the Kalman filter and Hungary algorithm to track the target and introducing a deep convolutional neural network to extract the appearance information of the tracked target for data association, which solves the problem that the target occlusion is difficult to track accurately. Stable and accurate vehicle detection result is an important guarantee for the DeepSORT algorithm in the vehicle tracking task. Considering the realtime requirements of the realistic application scenarios, the YOLOv5 target detection algorithm is studied in this article. To further reduce the memory and computing resources occupied by the algorithm, a DSC structure with residual is introduced into YOLOv5s, and the YOLOv5s_DSC algorithm with a smaller model and faster speed is proposed. YOLOv5s_DSC is used as the detector of the DeepSORT algorithm, and its excellent detection accuracy can make tracking more accurate and provide better realtime performance.

2.2. The DeepSORT Algorithm. Figure 1 is a framework diagram of the DeepSORT algorithm. First, the Kalman filter is used to predict the tracking frame information of the tracked target in the next frame, and all the detection frame information is obtained by the target detection algorithm in the subsequent frame. Then, the Hungarian algorithm is

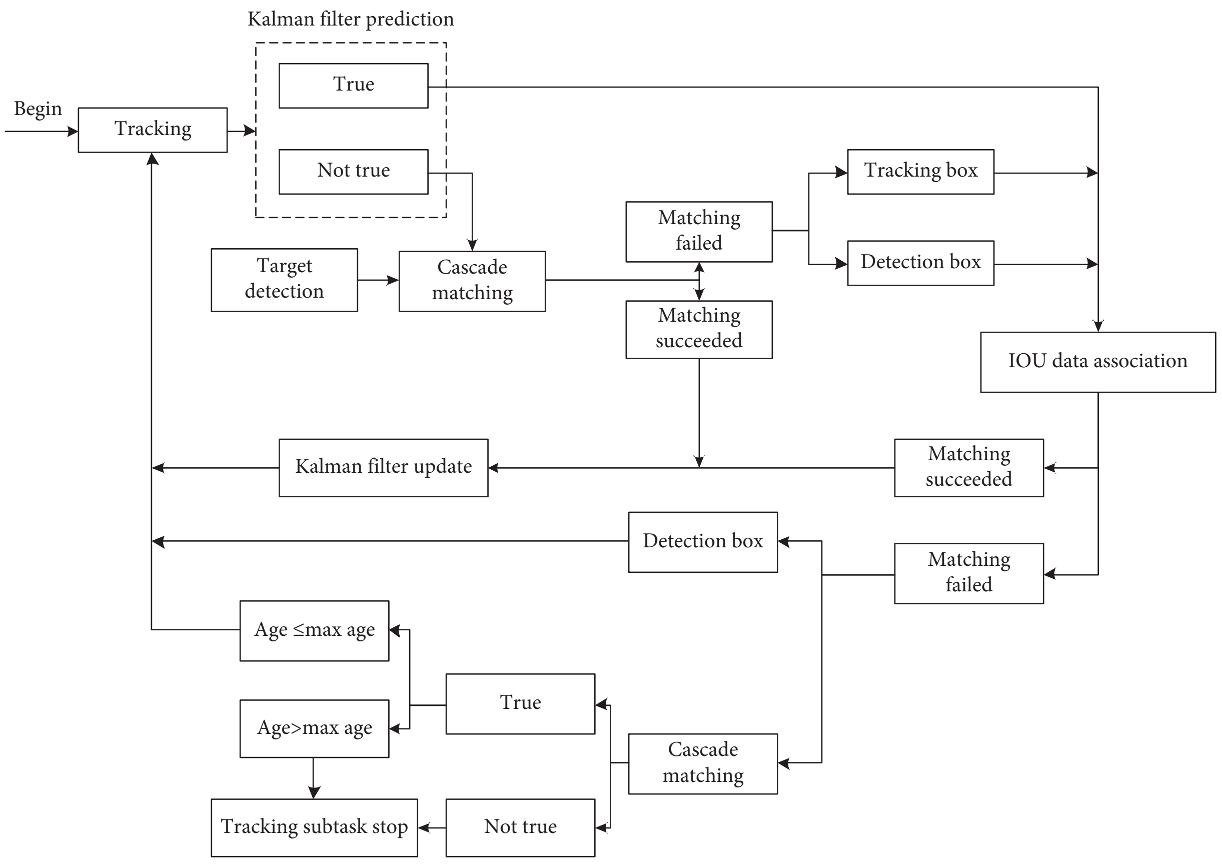


Figure 1: Block diagram of the DeepSORT algorithm.

used to find an optimal allocation for the minimum cost between all the detection frames and the tracking prediction frames. The cost matrix used in this step contains not only the Mahalanobis distance but also the cosine distance of the appearance features constructed from the appearance features extracted by the deep convolutional neural network. After solving by the Hungarian algorithm, the optimal combination of the prediction frame and the detection frame can be obtained. The DeepSORT algorithm uses cascade matching, and the shorter the number of frames from the last successful matching is, the higher the priority is in this matching. The tracking frame information is updated according to the detected frame information after the matching is successful, and the tracking frame information of the tracked target in the next frame is continued to predict according to the tracking information. For the samples that fail to match, the cost matrix will be constructed again with the IOU calculation results of the remaining tracking frame and the prediction frame and then transferred to the Hungarian algorithm for the solution. After the matching is successful, the tracking frame is updated according to the detection frame information, and the tracking frame information of the tracked target in the next frame is continuously predicted according to the tracking frame information. Whether the match is successful or not is determined by marking "true" and "false." For the detection frame that fails to match, a flag will be added–"false," and three subsequent rounds (age is the round and max age = 3) of investigation will be conducted. If all three rounds of matching are successful, the flag will be changed to "true." For tracking frame that fail to match, if they are marked as "false," the tracking task will be stopped, and if they are marked as "true," the lifespan will be set. Within the lifespan, the following three rounds of investigation will also be conducted. If all three rounds of matching are successful, the mark will be changed to "true."

State estimation methods mainly include state observers and various linear and nonlinear discrete estimators based on the Kalman filter. Liu et al. [36] proposed a novel vehicle sideslip angle estimation algorithm with the fusion of dynamic model and vision for vehicle dynamic control. A vehicle attitude angle observer based on the square-root cubature Kalman filter (SCKF) is designed in [37] to estimate the roll and pitch to reject the gravity components induced by the vehicle roll and pitch. For simplicity, this article uses the Kalman filter as state estimation. The prediction equation of the Kalman filter is as follows:

$$\begin{aligned} \hat{x}_k^- \&9; &= F_k \hat{x}_{k-1}^+, \\ P_k^- \&9; &= F_k P_{k-1}^+ F_k^T + Q_k, \end{aligned} \tag{1}$$

where $\hat{x}_k^-$ is the state estimation at the time k, $\hat{x}_{k-1}^+$ is the state estimation at the time $k-1$, P_k^- is the covariance matrix of the state estimation, and F_k is the state transition matrix. The measurement update equation of the Kalman filter is as follows:

$$\begin{aligned} K_k &= P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}, \\ \hat{x}_k^+ &= \hat{x}_k^- + K_k \left(y_k - H_k \hat{x}_k^-\right), \\ P_k^+ &= \left(I - K_k H_k\right) P_k^-, \end{aligned} \tag{2}$$

where y_k is the measurement vector at a time k, H_k is the measurement matrix, R_k is the covariance matrix of measurement noise, K_k is the Kalman gain used to correct the state estimation, and I is the identity matrix. The state vector of the DeepSORT algorithm can be described as follows:

$$x = [u, v, r, h, \dot{u}, \dot{v}, \dot{r}, \dot{h}]^T, \tag{3}$$

where u, v, r, and h represent the target box center coordinates of x, y aspect ratio, and height, respectively. $\dot{u}$, $\dot{v}$, $\dot{r}$, and $\dot{h}$ represent the corresponding value in the next frame predicted with Kalman filtering. The DeepSORT algorithm uses the cost matrix constructed by Mahalanobis distance and cosine distance of appearance features in the first data association. The Mahalanobis distance correlation metric is calculated as follows:

$$d^{(1)}(i, j) = \left(d_j - y_i\right)^T S_i^{-1} \left(d_j - y_i\right), \tag{4}$$

where $d_j = [u_j, v_j, r_j, h_j]^T$ represents the jth detection state, $d_i = [\dot{u}_i, \dot{v}_i, \dot{r}_i, \dot{h}_i]^T$ represents the ith tracking target which predicts the state of the current frame according to the state of the previous frame. S_i is the covariance of the detected state with the predicted state. The cosine distance measurement formula of appearance features is as follows:

$$d^{(2)}(i, j) = \min\left\{1 - r_j^T r_k^{(i)}, r_k^{(i)} \in R_i\right\}, \tag{5}$$

where r_j corresponds to the feature vector of the jth detection frame, $r_k^{(i)}$ correspond to the feature vector of the tracking frame, and R_i is for the last set of features k times successfully tracked. The DeepSORT algorithm constructs a deep convolutional neural network to extract the appearance features of the tracking target and uses L2 standardization to project the features. The network structure is shown in Table 1.

Due to the nonconstant update frequency of image frames, we use the time difference between the two frames as the time step of the Kalman filter during the discretization process. This approach allows us to dynamically adjust the state update rate of the Kalman filter based on the actual situation, which helps to better track the target.

3. Improved Yolov5 Vehicle Detection Method

To achieve high precision vehicle tracking tasks, the vehicle detection algorithm is studied in this subsection. To further improve the realtime performance of vehicle detection, the DSC structure with residual is introduced, and the YOLOv5s_DSC vehicle detection algorithm is proposed, which has a lower number of parameters and calculation and faster detection speed.

3.1. Depth Separable Convolution. With the help of grouping convolution, DSC uses point-by-point convolution to fuse the feature information of different channels, which can achieve the purpose of a lightweight deep learning network, while ensuring feature extraction. It divides into the following two steps:

(1) Channel-by-channel convolution: the input image is $H_{\text{in}} * W_{\text{in}} * C_{\text{in}}$. Each channel consists of a $K * K * 1$. The convolution kernel performs an independent convolution operation to obtain C_{in} characteristic map, whose size is $H_{\text{out}} * W_{\text{out}}$. The parameter number of the convolution kernel is $K * K * C_{\text{in}}$. As shown in Figure 2, if 3 channels of images are as inputs in the point-by-point convolution, 3 single-channel will be obtained.

(2) Pointwise convolution: using $1 * 1 * C_{\text{in}} * C_{\text{out}}$, convolution kernel performs convolution operation output of (1) to obtain the characteristic map with $H_{\text{out}} * W_{\text{out}} * C_{\text{out}}$. As shown in Figure 3, the number of parameters of the characteristic map is $1 * 1 * C_{\text{in}} * C_{\text{out}}$.

The number of parameters of the whole DSC is as follows:

$$P = K * K * C_{\text{in}} + 1 * 1 * C_{\text{in}} * C_{\text{out}}. \tag{6}$$

This is similar to the packet convolution with a number of packets C_{in}. The difference is that the results of group convolution are the splicing of each group result, while the results of DSC are the weighted combination of each group of result by point-by-point convolution, which can make full use of the characteristic information of each channel at the same position.

Table 1: DeepSORT deep convolutional neural network.

	C_{in}	C_{out}	Kernel_size	Stride
Conv	3	32	3	1
Conv	32	32	3	1
MaxPool	32	32	3	2
Residual	32	32	3	1
Residual	32	32	3	1
Residual	32	64	3	2
Residual	64	64	3	1
Residual	64	128	3	2
Residual	128	128	3	1
	Dense		128	
	Batch and L2 normalization		128	

3.2. Yolov5s Improvement Strategy. The YOLOv5s model has 283 layers in total, the number of parameters is 7,071,633, and the amount of calculation is 16.4GFLOPS. To further simplify the network structure, reduce the amount of calculation, and reduce the reasoning time of the model, the DSC structure is introduced to replace the C3 structure of the backbone part in YOLOv5s. As shown in Figure 4, the first C3 structure in the YOLOv5s network contains five convolutions, and the parameters are shown in Table 2.

From Table 2, it can be calculated that the number of parameters for Conv1 and Conv2 is 2048, the number of parameters for Conv3 is 4,096, the number of parameters for

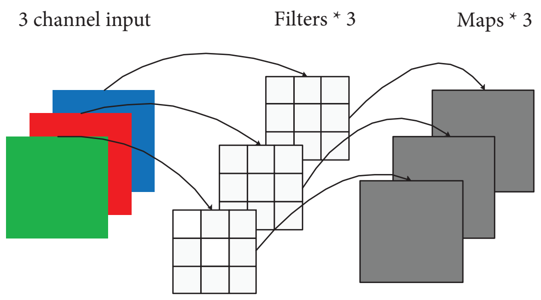


FIGURE 2: Depthwise convolution.

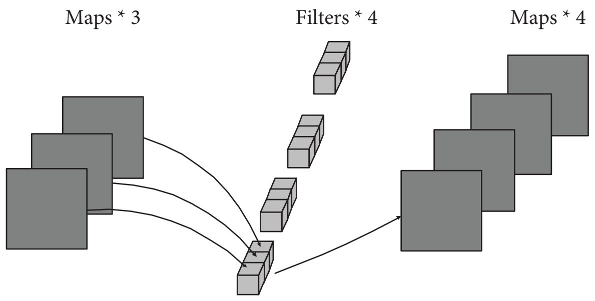


FIGURE 3: Pointwise convolution.

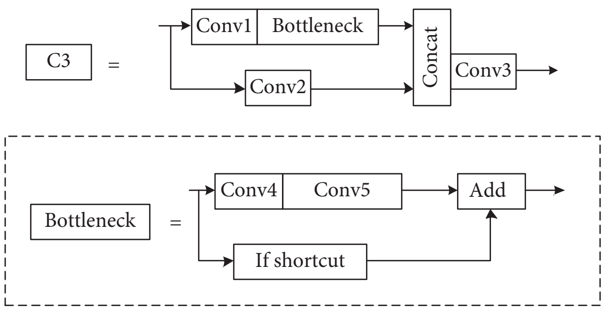


FIGURE 4: The first C3 structure of YOLOv5s.

TABLE 2: The convolution parameter of the first C3.

	C_{in}	C_{out}	Kernel_size
Conv1	64	32	1
Conv2	64	32	1
Conv3	64	64	1
Conv4	32	32	1
Conv5	32	32	3

Conv4 is 1,024, and the number of parameters for Conv5 is 9,216. Therefore, the number of parameters of the first C3 structure in the YOLOv5s network amounts to 18,432.

DSC performs two convolutions. The first convolution obtains the features of each channel. The second convolution fuses the position information of each channel. In contrast to the first C3 structure in the YOLOv5s network, the input and output channels of the DSC are also set to 64, and the size of the channel-by-channel convolution is 3×3. Then, the number of parameters of the channel-by-channel convolution is 576, and the size of the point-by-point convolution is 1×1. Then, the number of parameters of the pointwise convolution is 4,096. Thus, the number of parameters of the DSC structure amounts to 4,672, which is 13,760 lower than that of the first C3 structure in the YOLOv5s network. The backbone of the YOLOv5s network contains four C3 structures, which are replaced by DSC structures in turn. To avoid network degradation caused by replacing with DSC, a residual structure is introduced, as shown in Figure 5.

The introduction of DSC can effectively reduce the number of parameters and make the network model smaller. The comparison of the number of parameters after replacement is shown in Table 3. The number of parameters of each structure includes the parameters of convolution, deviation, and batch normalization in the structure. The improved network framework is presented in Table 4.

4. Experimental Results and Analysis

4.1. Dataset Preparation. The VeRi dataset [38] is a large vehicle rerecognition dataset, which contains vehicle images from multiple angles and under different light intensities. It is suitable for related research on vehicle rerecognition. As shown in Figure 6, each folder contains pictures taken by the same vehicle from different angles, with a total of 776 folders. The training set and the test set are distributed according to the proportion of 8 : 1.

UA-DETRAC [39] is a vehicle dataset, collected from the real traffic environment of Beijing and Tianjin, labeled with four vehicle categories of "Bus," "Car," "Van," and "Others," including vehicle images of different angles and periods, covering most of the traffic conditions. The UA-DETRAC dataset contains a total of 60 image folders collected from different road sections and periods, and each folder corresponds to an XML tag file. We use the code to strip the tag corresponding to each image in the XML file and convert all the XML tag files obtained into TXT format. The training set and the test set are distributed according to the proportion of 9 : 1. There are 73,876 pieces of training sets and 8,209 pieces of test sets in total. The data structure of images and labels is shown in Figure 7.

4.2. Training of DeepSORT Deep Convolutional Neural Network. The vehicle rerecognition dataset is used to train the DeepSORT deep convolutional neural network, so that it can correctly extract the appearance features of the vehicle for the calculation of the cosine distance of the appearance features. Since the task requirement is vehicle tracking, the input of the network is set $128\,(h) \times 64\,(w)$, according to the aspect ratio of the vehicle image. The network model is built under the PyTorch framework. The initial learning rate is set at 0.1, which is reduced to 0.1 times every 40 epochs. The training loss curve is shown in Figure 8. After reaching 100 epochs, the loss tends to be stable and the accuracy on the test set reaches 88%.

4.3. Yolov5s_DSC Network Training and Result Analysis. The YOLOv5s network model and YOLOv5s_DSC network model are constructed under the PyTorch framework, respectively. The UA-DETRAC dataset is used for training.

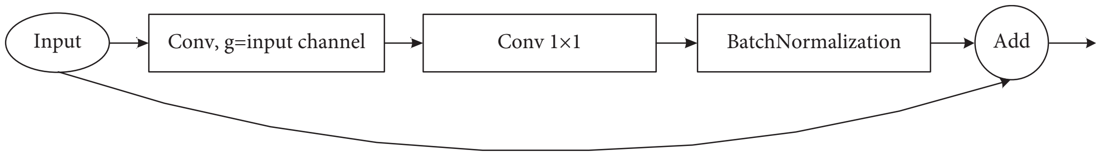


Figure 5: DSC structure with residual.

Table 3: Number of parameters before and after replacement.

Network structure	Parameter quantity
C3_1	18816
C3_2	156928
C3_3	625152
C3_4	1182720
DSC_1	4928
DSC_2	54144
DSC_3	206592
DSC_4	268800

Table 4: The network structure of YOLOv5s_DSC.

	C_{in}	C_{out}	Kernel_size	Stride	Padding
Focus	3	32			
Conv	32	64	3	2	1
DSC_1	64	8			
Conv	8	128	3	2	1
DSC_2	128	128			
Conv	128	256	3	2	1
DSC_3	256	256			
Conv	256	512	3	2	1
SPP	512	512			
DSC_4	512	512			
Conv	512	256	1	1	0
Upsample concat					
C3	512	256			
Conv	256	128	1	1	0
Upsample concat					
C3	256	128			
Conv	128	128	3	2	1
Upsample concat					
C3	256	256			
Conv	256	256	3	2	1
Concat					
C3	512	512			

Figure 6: VeRi data set.

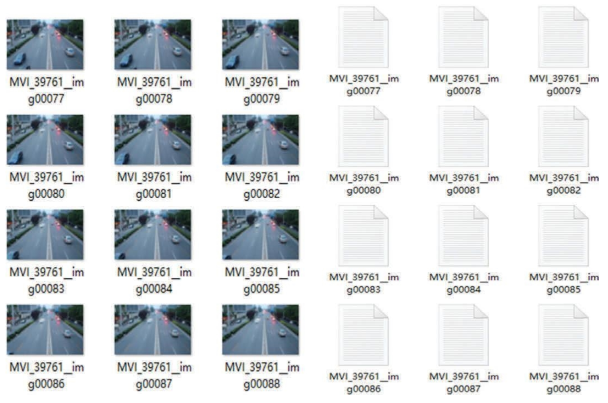


Figure 7: UA-DETRAC dataset.

The batch size is 128, and 50 epochs are trained. The training loss is shown in Figure 9. The YOLOv5s_DSC network decreases as fast as YOLOv5s in the regression loss, the classification loss, and the target loss, where the lowest values of the three losses of YOLOv5S are 0.01722, 0.0011741, and 0.02758, respectively. However, the lowest values of the three losses of YOLOv5s_DSC are 0.01835, 0.0013954, and 0.02933, respectively, which indicates that the introduction of DSC structure with residual error does not bring too much impact on the training difficulty of the network.

Compare the performance of YOLOv5s and YOLOv5s_DSC in mAP, precision, and recall. In Figure 10, the curves of the two networks are almost coincident, which indicates that the introduction of the DSC structure with residuals brings about a decrease in the number of network parameters but does not cause a decrease in the accuracy of the network. The YOLOv5s_DSC with KF (Kalman filter) is smoother than the YOLOv5s_DSC. The YOLOv5s with KF is smoother than the YOLOv5s. This indicates that KF can dynamically adjust its update rate, which helps to better track the targets. In Table 5, mAP (mean average precision) indicates better performance of the detector, except for the optimal mAP0.5: 0.95; the difference between the optimal mAP0.5, precision, and recall of YOLOv5s_DSC and YOLOv5s is not more than 1%, while the number of parameters is reduced by 23.5%. The amount of calculation is reduced by 32.3%, and the size of the weight file is decreased by 20%. In the hardware environment, where the graphics card is NVIDIA GeForce RTX 3080 and the CPU is Inter (R) Xeon (R) CPU E5-2670 v3, the average processing speed of each image is improved by 18.8%, which proves that the proposed algorithm is faster while ensuring the accuracy.

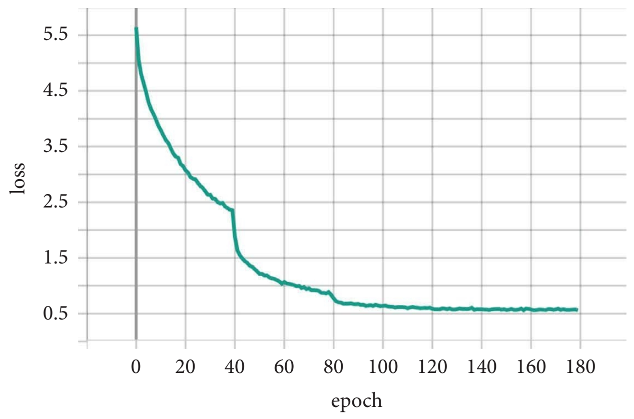


Figure 8: Training loss graph of DeepSORT deep convolutional neural network.

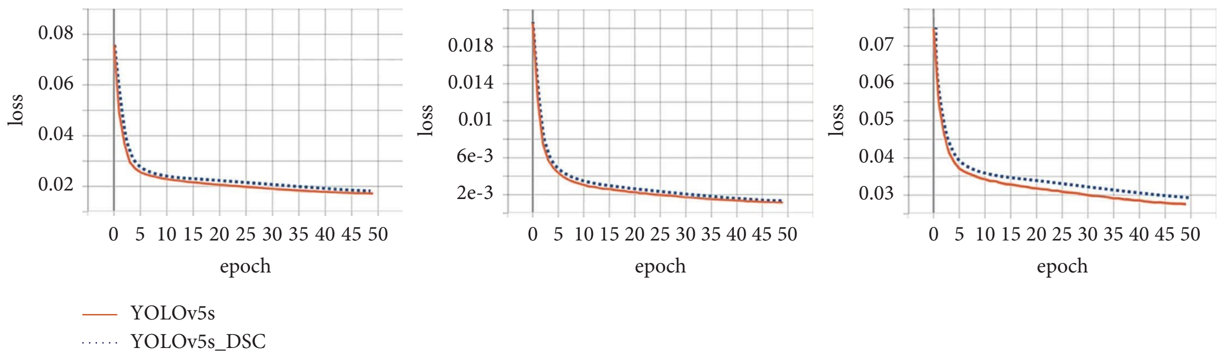


Figure 9: Comparison of training loss between YOLOv5s and YOLOv5s_DSC.

4.4. Verification Experiment. Select a video of traffic flow captured from the front of the intersection as the input. As shown in Figure 11, the YOLOv5s_DSC vehicle detection algorithm can effectively detect vehicles and correctly classify vehicles under the window. Each detection frame contains two information: vehicle category name and category confidence. In the hardware environment shown in Table 6, the detection speed of the algorithm reaches 77 FPS. Select a video of traffic flow captured from the oblique side of the intersection as the input, and YOLOv5s_DSC vehicle detection algorithm can also effectively detect the vehicles and correctly classify the vehicles under this window, as shown in Figure 12. YOLOv5s_ DSC can accurately detect vehicles from different angles. As shown in Figure 12(b), local mutual occlusion between vehicles does not affect the detection effect of the algorithm. Therefore, the advantages of the YOLOv5s_ DSC algorithm for vehicle detection can provide realtime and accurate vehicle detection information for vehicle tracking.

To test the effect and the robustness of occlusion of the algorithm on vehicle tracking, the YOLOv5s_DSC is as a detector connected to YOLOv5s_DSC and DeepSORT. As shown in Figure 13, the tracking boxes of different types of vehicles have different colors, and each tracking box includes a tracking ID in addition to the category and category confidential information of the vehicle. In the hardware environment shown in Table 6, the YOLOv5s_DSC + DeepSORT algorithm achieves a processing speed of 25 FPS.

Next, the robustness of the proposed algorithm is verified in the occlusion scene. Consider the tracking performance of two occlusion situations: (1) the target is occluded by foreign objects and (2) the targets are occluded by each other. First, the robustness of the proposed algorithm is verified when the target is occluded by external objects. The effect of rerecognition and retracking after the target disappears is tested. A traffic video blocked by a pillar is supported to verify the algorithm. Figure 14 shows four consecutive images. The dark car with tracking ID 3 reappears after being blocked by a pillar and can still be tracked by the algorithm. This result shows that the algorithm exhibits strong robustness and accuracy in occluded scenes, providing strong support for target tracking in practical applications.

We also evaluate the algorithm's performance in scenarios where targets are occluded by other targets. Specifically, we test the algorithm's ability to track targets that are partially occluded by other targets. A video sequence in which a bus partially occludes a car that is being tracked is selected. As shown in Figure 15, the vehicle with tracking ID 4 is partially blocked by the bus with tracking ID 1. Despite the occlusion, the tracking ID for the car remains unchanged. It is shown that the proposed algorithm is capable of handling partial occlusions between targets. These results

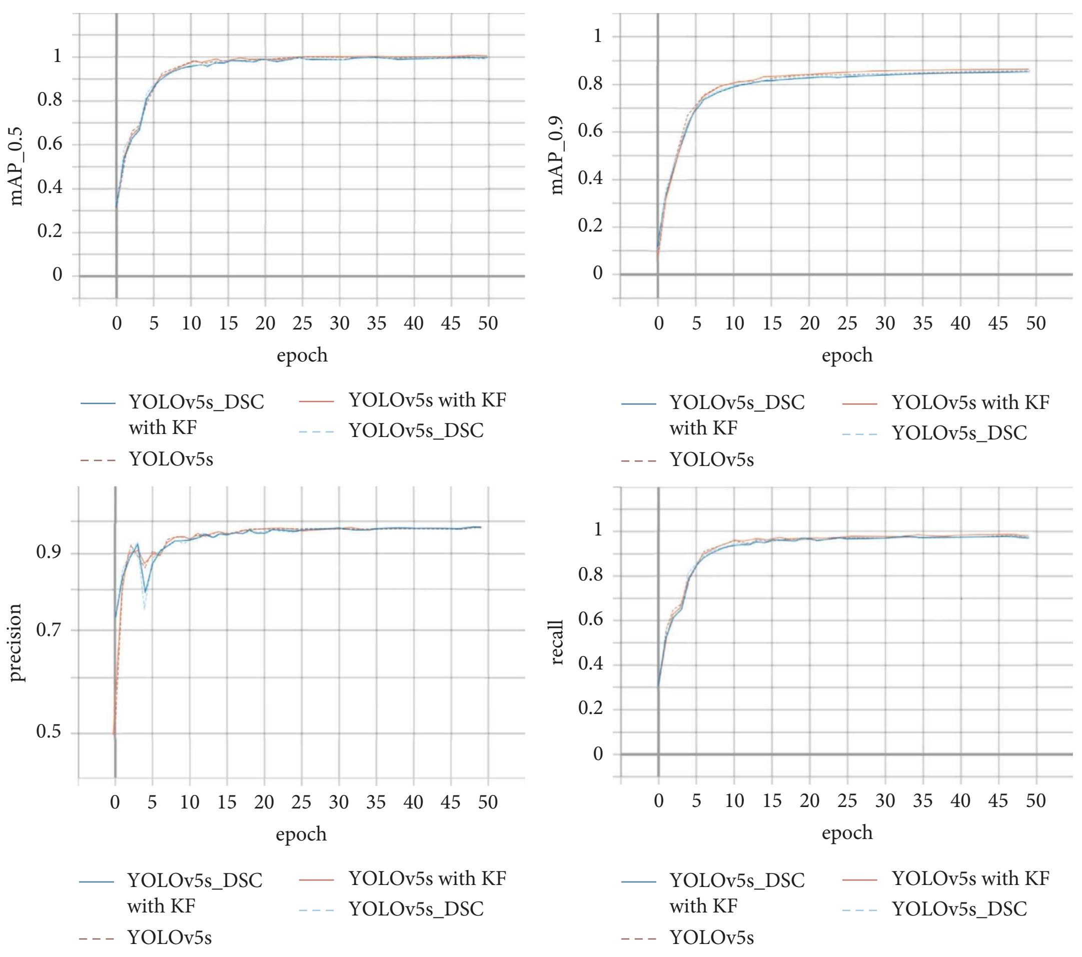


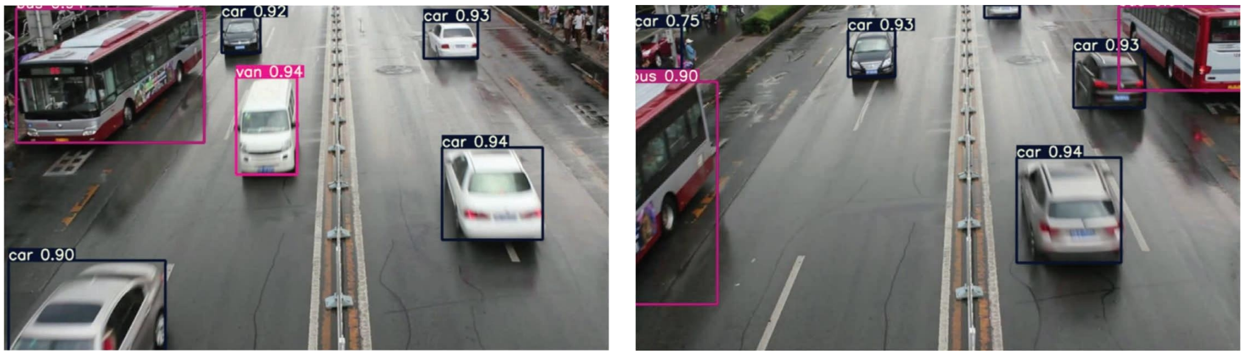
Figure 10: mAP, accuracy, and recall of YOLOv5s and YOLOv5s_DSC.

Table 5: Data comparison of YOLOv5s_DSC and YOLOv5s.

	mAP0.5	mAP0.5 : 0.95	Accuracy	Recall	Parameters	Calculation (GFLOPS)	Model size (M)	Image processing speed (s)
YOLOv5s	0.993	0.865	0.981	0.983	7071633	16.4	14.4	0.016
YOLOv5s_DSC	0.993	0.851	0.980	0.976	5622481	12.3	11.5	0.013

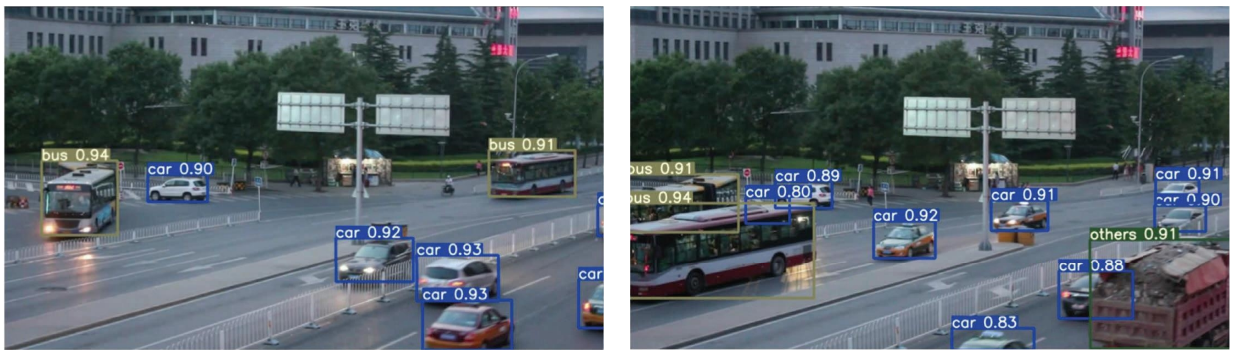
Table 6: Hardware environment configuration.

Configuration	Model
CPU	Inter (R) xeon (R) CPU E5-2670 v3
GPU	NVIDIA GeForce RTX 3080
CUDA	11.0
PyTorch	1.7.1
Python	3.8



(a) (b)

Figure 11: YOLOv5s_DSC vehicle detection at the frontal windows.



(a) (b)

Figure 12: YOLOv5s_DSC vehicle detection at the oblique side of the intersection.

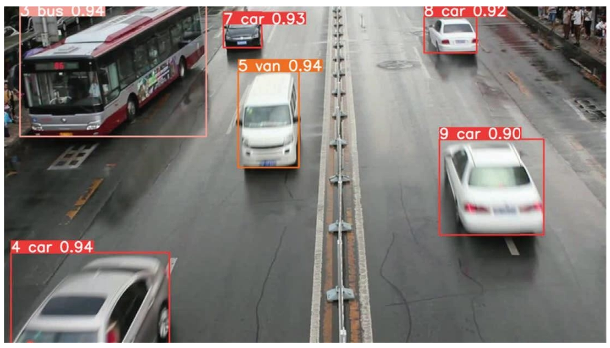
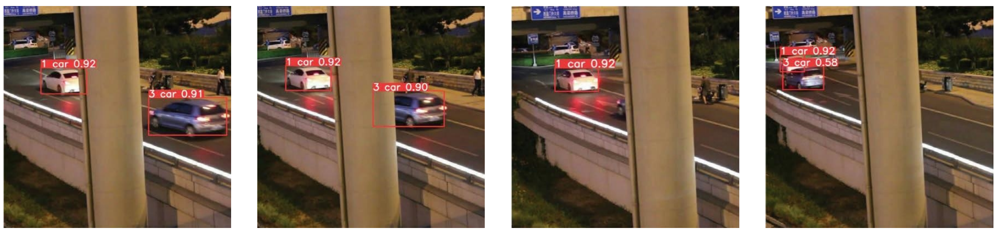
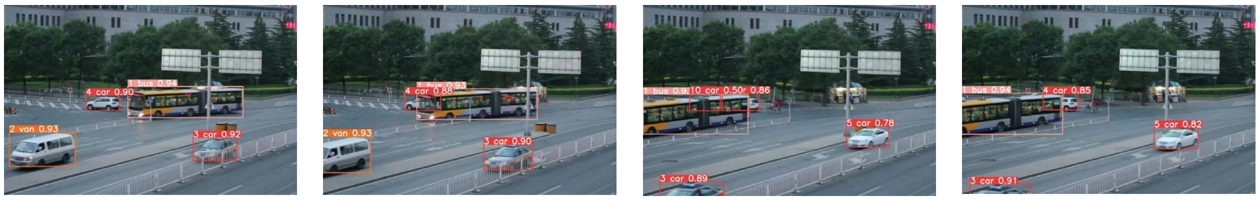


Figure 13: YOLOv5s_DSC vehicle detection effect in front of an intersection.



(a) (b) (c) (d)

Figure 14: YOLOv5s_DSC + DeepSORT recognition and tracking effect.



(a) (b) (c) (d)

Figure 15: Tracking effect of YOLOv5s_DSC + DeepSORT under local mutual occlusion of the target.

further demonstrate the robustness and effectiveness of the algorithm in occluded scenes, which is crucial for the practical application of the target tracking.

5. Conclusions

This article investigates the application of the DeepSORT algorithm in vehicle tracking, using vehicle flow videos from different scenarios to verify the effectiveness and robustness of the YOLOv5s_DSC vehicle detection algorithm. The YOLOv5s + DeepSORT algorithm is validated by reproducing traffic flow videos that block each other after the vehicle disappears. It is showed that the algorithm has good rerecognition and retracking ability and robustness against partial occlusion of targets. However, the algorithm in this article does not take into account the detection effect in different weather environments such as rainy days, foggy weather, and vehicle video blurring. In future work, the application of model compression methods will be studied to further compress network models, while maintaining a certain accuracy and improving the speed of network reasoning and combining algorithms such as environment optimization to achieve more scene applications.

Data Availability

The data used to support the findings of this study are available upon request from the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Lixiong Lin conceptualized the study, proposed the methodology, validated the study, and wrote and prepared the original draft; Hongqin He and Dongjie Wu performed formal analysis, performed investigation, provided resources, and performed data curation; Zhiping Xu wrote, reviewed, and edited the manuscript, visualized the study, and performed project administration. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

The research was supported by the Jimei University Startup Research Project, China, under grant ZQ2022002, by the Education Department Foundation of Fujian Province, China, under grant JAT220169, by the Xiamen Key Laboratory of Marine Intelligent Terminal R&D and Application, China, under grant B18208, and by the Xiamen Ocean and Fishery Development Special Fund Project, China, under grant 21CZB013HJ15.

References

[1] H. Chen and J. Guan, "Teacher–student behavior recognition in classroom teaching based on improved YOLO-v4 and internet of things technology," *Electronics*, vol. 11, no. 23, p. 3998, 2022.
[2] M. Li, L. Zhang, L. Li, and W. Song, "Yolo-based traffic sign recognition algorithm," *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 2682921, 10 pages, 2022.
[3] K. Du, Y. Ju, Y. Jin, G. Li, Y. Li, and S. Qian, "Object tracking based on improved Mean Shift and SIFT," in *Proceedings of the 2012 2nd International Conference on Consumer Electronics, Communications, and Networks*, pp. 2716–2719, Yichang, China, April 2012.
[4] L. Liu, C. Ke, H. Lin, and H. Xu, "Research on pedestrian detection algorithm based on MobileNet-YoLo," *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 8924027, 12 pages, 2022.
[5] X. Mei, H. Ling, Y. Wu, E. P. Blasch, and L. Bai, "Efficient minimum error bounded particle resampling L1 tracker with occlusion detection," *IEEE Transactions on Image Processing*, vol. 22, no. 7, pp. 2661–2675, 2013.
[6] G. Yan, G. Qu, and M. Yu, "Real-time L1-tracker based on hear-like features," *Computer Science*, vol. 44, pp. 300–306, 2017.
[7] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.
[8] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: learning continuous convolution operators for visual tracking," in *Proceedings of the European Conference on Computer Vision*, pp. 472–488, Amsterdam, The Netherlands, October 2016.
[9] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Eco: efficient convolution operators for tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6638–6646, Honolulu, HI, USA, July 2017.
[10] J. Li and S. Huang, "YOLOv3 based object tracking method," *Electronics Optics and Control*, vol. 26, p. 87, 2019.
[11] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. Torr, "Fully-convolutional siamese networks for object tracking," in *Proceedings of the European Conference on Computer Vision*, pp. 850–865, Amsterdam, The Netherlands, October 2016.

[12] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proceedings of the European Conference on Computer Vision*, pp. 101–117, Munich, Germany, September 2018.

[13] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4282–4291, Seoul, South Korea, October 2019.

[14] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and real-time tracking," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 3464–3468, Amsterdam, The Netherlands, October 2016.

[15] N. Wojke, A. Bewley, and D. Paulus, "Simple online and real time tracking with a deep association metric," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 3645–3649, Honolulu, HI, USA, July 2017.

[16] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: a review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, no. 3, pp. 159–190, 2006.

[17] P. Wang, C. Shen, N. Barnes, and H. Zheng, "Fast and robust object detection using asymmetric totally corrective boosting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 33–46, 2012.

[18] S. Bmeshram and S. M Shinde, "A survey on ensemble methods for high dimensional data classification in biomedicine field," *International Journal of Computer Application*, vol. 111, no. 11, pp. 5–7, 2015.

[19] G. Díaz-San Martín, L. Reyes-González, S. Sainz-Ruiz, L. Rodriguez-Cobo, and J. M. Lopez-Higuera, "Automatic ankle angle detection by integrated RGB and depth camera system," *Sensors*, vol. 21, no. 5, p. 1909, 2021.

[20] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[21] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *Proceedings of the European Conference on Computer Vision*, pp. 469–481, Prague, Czech Republic, May 2004.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, Zurich, Switzerland, September 2014.

[23] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, Santiago, Chile, December 2015.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, Amsterdam, The Netherlands, October 2016.

[26] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271, Honolulu, HI, USA, July 2017.

[27] J. Redmon and A. Farhadi, "Yolov3: an incremental improvement," 2018, https://arxiv.org/abs/1804.02767.

[28] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "Yolov4: optimal speed and accuracy of object detection," 2020, https://arxiv.org/abs/2004.10934.

[29] Y. Sasagawa and H. Nagahara, "Yolo in the dark-domain adaptation method for merging multiple models," in *Proceedings of the European Conference on Computer Vision*, pp. 345–359, Glasgow, UK, August 2020.

[30] M. Krišto, M. Ivasic-Kos, and M. Pobar, "Thermal object detection in difficult weather conditions using YOLO," *IEEE Access*, vol. 8, pp. 125459–125476, 2020.

[31] Y. Xiao, A. Jiang, J. Ye, and M. Wang, "Making of night vision: object detection under low-illumination," *IEEE Access*, vol. 8, pp. 123075–123086, 2020.

[32] K. Guo, X. Li, M. Zhang, Q. Bao, and M. Yang, "Real-time vehicle object detection method based on multi-scale feature fusion," *IEEE Access*, vol. 9, pp. 115126–115134, 2021.

[33] W. Liu, K. Quijano, and M. M. Crawford, "YOLOv5-Tassel: detecting tassels in RGB UAV imagery with improved YOLOv5 based on transfer learning," *Ieee Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 8085–8094, 2022.

[34] M. Bie, Y. Liu, G. Li, J. Hong, and J. Li, "Real-time vehicle detection algorithm based on a lightweight You-Only-Look-Once (YOLOv5n-L) approach," *Expert Systems with Applications*, vol. 213, Article ID 119108, 2023.

[35] J. Wang, Y. Dong, S. Zhao, and Z. Zhang, "A high-precision vehicle detection and tracking method based on the attention mechanism," *Sensors*, vol. 23, no. 2, p. 724, 2023.

[36] W. Liu, L. Xiong, X. Xia, Y. Lu, L. Gao, and S. Song, "Vision-aided intelligent vehicle sideslip angle estimation based on a dynamic model," *IET Intelligent Transport Systems*, vol. 14, no. 10, pp. 1183–1189, 2020.

[37] W. Liu, X. Xia, L. Xiong, Y. Lu, L. Gao, and Z. Yu, "Automated vehicle sideslip angle estimation considering signal measurement characteristic," *IEEE Sensors Journal*, vol. 21, no. 19, pp. 21675–21687, 2021.

[38] X. Liu and Z. Zheng, "VeRi dataset," https://github.com/JDAI-CV/VeRidataset.

[39] L. Wen, D. Du, Z. Cai et al., "UA-DETRAC: a new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, vol. 193, pp. 1–20, 2020, https://detrac-db.rit.albany.edu/.