Received: 30 April 2023 | Revised: 14 August 2023 | Accepted: 5 September 2023

IET Computer Vision

DOI: 10.1049/cvi2.12236

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Real-time vehicle detection using segmentation-based detection network and trajectory prediction

Nafiseh Zarei[1] | **Payman Moallem**[1] | **Mohammadreza Shams**[2]

[1]Department of Electrical Engineering, Faculty of Engineering, University of Isfahan, Isfahan, Iran

[2]Department of Computer Engineering, Shahreza Campus, University of Isfahan, Isfahan, Iran

Correspondence

Payman Moallem.
Email: p_moallem@eng.ui.ac.ir

Abstract

The position of vehicles is determined using an algorithm that includes two stages of detection and prediction. The more the number of frames in which the detection network is used, the more accurate the detector is, and the more the prediction network is used, the algorithm is faster. Therefore, the algorithm is very flexible to achieve the required accuracy and speed. YOLO's base detection network is designed to be robust against vehicle scale changes. Also, feature maps are produced in the detector network, which contribute greatly to increasing the accuracy of the detector. In these maps, using differential images and a u-net-based module, image segmentation has been done into two classes: vehicle and background. To increase the accuracy of the recursive predictive network, vehicle manoeuvres are classified. For this purpose, the spatial and temporal information of the vehicles are considered simultaneously. This classifier is much more effective than classifiers that consider spatial and temporal information separately. The Highway and UA-DETRAC datasets demonstrate the performance of the proposed algorithm in urban traffic monitoring systems.

KEYWORDS

convolutional neural nets, object detection, recurrent neural nets, vehicles

1 | INTRODUCTION

Vehicles have become an integral part of our lives. More innovative vehicles can bring us a more comfortable life, however, their presence can also lead to other issues like traffic congestion and accidents. The intelligent transportation system (ITS) tries to manage these issues appropriately. In this system, vehicles are accurately identified and counted, and not only are traffic accidents prevented, but by analysing the route of the vehicles and their numbers, the accidents that have occurred are well investigated, and traffic control is done more effectively. This achievement is precious in the traffic control system, and the vehicle detection algorithm is one of the most fundamental issues in the ITS system [1]. On the other hand, the detection of vehicles in automated driving technology also plays an essential role because by detecting vehicles, the driverless vehicle can predict the direction of movement well and, according to different scenarios in urban environments, make the right decision to continue [2]. Vehicle recognition plays a vital role in many applications, such as security issues or issues related to goods transit and intercontinental communication [1], remote sensing [3], traffic modelling [4], traffic monitoring [5], environment monitoring [6], and road planning, for estimation or simulation of air and noise pollution [7]. Many algorithms based on computer vision have been presented in this field [8–10]. These algorithms must be able to work with high speed and accuracy [11, 12]. Therefore, the design of lightweight and powerful detectors in this field is very attractive and is a challenging computer vision problem that attracts academic and industrial interests and efforts [13].

The main challenge of our study is to increase the speed of vehicle detection without losing its accuracy. Increasing speed and accuracy simultaneously in detectors is regarded as a fundamental challenge because, typically, one of them must be sacrificed to improve the other. Other researchers have only upgraded the detector network to achieve this goal. But, in this

study, we have also used the position predictor network in addition to the detector upgrade, which we explain below. In fact, in addition to the spatial information of consecutive images that other detectors use to determine the vehicle's position, we also use temporal information and do not lose the positions of the vehicles in the previous frames, which are important information. Using these two speeds up the algorithm's execution because prediction is much quicker than detection. According to the complexity of the scene, its complexities, and the user's opinion, the timing of using detector and prediction networks can be changed within the defined periodicity. This increases the flexibility of the algorithm to adjust the accuracy and speed of detection.

Another advantage of employing a predictive network is dealing with vehicle occlusion. Occlusion means that the target is covered by obstacles and is not visible from the camera's view, which causes the detector to lose the position of the vehicle. One of the solutions to the occlusion problem is the use of multiple cameras with various views, which, although they provide more information about the target, also have their problems, such as object matching between captured images from different cameras. Another solution to the problem of target loss is to use motion prediction. In the traditional methods of motion prediction, acceleration or constant velocity models are used, but in these methods, the fact is ignored that moving vehicles do not follow a specific model during movement, and in successive frames of a video, they change their direction and their speed. This is due to the driver's freedom of action in changing the direction of the vehicle and the occurrence of different conditions while driving, so it does not seem logical to use fixed-based methods to consider the speed or acceleration of vehicles. But since the changes in the direction of the vehicle depend on the movement of the vehicle in the previous frames, the movement history can be used, and, based on that, the next position can be predicted. Recurrent networks can store the history of vehicle movement and make decisions based on it. In the training phase, they learn all possible trajectories for the vehicle. Hence, the use of predictive networks can greatly reduce the problem of losing the target by dealing with the challenge of occlusion, and due to their higher speed compared to the detector network, they also increase the speed of the positioning algorithm. Our proposed algorithm also uses these networks and prevents the error from increasing over time, while several studies [14, 15] have shown that the root mean square error (RMSE) of trajectory predictors after a few seconds is two to five times the RMSE error in the first second. This is a significant achievement for our research.

Another advantage of our proposed algorithm is that our proposed detector is lighter and faster than recent real-time detectors such as the YOLO family and is comparable to them in terms of accuracy. The reason for this superiority is the use of the attention mechanism based on semantic segmentation in the proposed network. Using this mechanism, it is possible to create more detailed feature maps in which the vehicles stand out well against the image background. These feature maps improve the accuracy of vehicle positioning. The main part of this module is the encoder of a lightweight U-Net base deep network that is designed and proposed in this article.

The major contributions and novelty of the present work include:

- The proposed algorithm contains two parts: detection and prediction are very flexible, and the user can adjust the speed and accuracy of vehicle positioning according to the complexity of the road. This is done by varying the number of frames that use detection or prediction in an alternate cycle. Using the detector network increases the accuracy of the algorithm and using the predictive network increases the speed of the algorithm.
- When the vehicles are occluded, the detector network is unable to determine their location. However, the proposed algorithm's prediction network is capable of doing this well. Based on the position of the vehicle before occluding in previous frames, this network can position the occluded vehicle. It also prevents the RMSE prediction error from ascendingly over time, despite the fact that in several studies, the RMSE of position predictors after a few seconds has been reported to be two to seven times the RMSE error in the first second. This is a significant achievement for our algorithm, which decreases RMSE and avoids network deviation in trajectory prediction.
- It presents a suitable algorithm for classifying vehicle trajectories to improve the performance of the position prediction network. For the classification of movement manoeuvres, a new framework has been presented in which the spatial and temporal information of the vehicles is considered simultaneously, and it is much more effective than the classifiers that consider the spatial and temporal information separately. In comparison to other trajectory-predicting techniques, the manoeuvre classifier is more accurate.
- In the backbone of the proposed detector, the semantic attention mechanism is suggested. This powerful mechanism is implemented by a segmentation-based auxiliary map generation (SAMG) module. In the SAMG module, feature maps are produced in which vehicles are well separated from the background. In addition, the different two receptive fields are used in the extraction of the feature map by designing the two receptive field (TRF) block. The existence of these maps increases the detection accuracy. On the other hand, due to eliminating the redundancy of consecutive images in differential images and focusing on the important information of the image instead of the whole image, to gain greater accuracy, the network's depth does not need to be increased, in this way, in addition to achieving proper accuracy, the speed of the detector network can also be increased.

The remainder of the paper is organised as follows: Section 2 reviews the detection literature; Section 3 describes the proposed algorithm; Section 4 provides and discusses the results; and Section 5 concludes the study.

2 | RELATED WORKS

Before the emergence of deep neural networks, many methods had been used to identify objects in images based on non-automatic features. Among these methods, we can mention the gradient histogram, optimal flow, Kalman filter, machine learning-based techniques such as principal component analysis, fuzzy methods, and classification methods such as support vector machines for classifying complex data sets [16–23]. Geometric transformations and feature extraction have also been employed [24–26] to detect moving objects and determine their descriptors. But convolutional neural networks revolutionised the field of object detection [27]. Deep neural network algorithms developed in object detection can be classified into region-based and network-based groups. The former focuses on regions with a higher probability of vehicles rather than the entire image to save processing time. RCNN [28] is the most popular detection network in this group. Fast-RCNN [29] and Faster-RCNN [30] are the augmented variants of RCNN. The RCNN has been inspired by DPM [31]. DPM uses a coarse root filter at low resolution and several component filters at higher resolutions. Different filters are utilised to detect different parts of an object. For example, in the detection of humans, the human body is investigated at low resolution using a coarse filter, while the hands, feet, and head are examined at a higher resolution using a finer filter [32, 33]. RCNN operates based on region and identifies the candidate object region. It utilises selective search to identify such regions. Then, the features of the regions are computed, classifying the regions using a linear SVM.

Despite the advantages of RCNN over earlier detection algorithms, the RCNN-extracted features do not contain spatial information to generate precise boundaries. Moreover, the time cost of it is not desirable since training is implemented in several steps. First, a feature extraction network is used. Then, the SVM is set. Finally, the regression network is trained [31]. Drawing on the VGG16 architecture for the feature extraction network in RCNN, the detection of each image is performed in 47 s. Furthermore, the training phase of this method is long. Fast-RCNN was presented to address this problem. The Fast-RCNN trains the same network for feature extraction, classification, and regression. Then Faster-RCNN was developed. It exploits the region proposal network (RPN) as a simple neural network. This network searches the image using a sliding window to find regions containing objects. In fact, RPN substitutes for selective search in RCNN and significantly accelerates the algorithm. The regions searched by the RPN are known as anchors. In practice, a large number of anchors of different sizes cover the image. RPN searches all the anchors in parallel on the GPU in a very short time. However, RPN searches an already extracted feature map rather than searching the image directly. This enables RPN to effectively reuse the extracted features and avoid repeated computations. RPN generates two outputs for each anchor: an anchor class and an anchor box modification. The anchor classes include foreground and background. Despite good accuracy, RCNN, Fast-RCNN, and Faster-RCNN have high time costs. Therefore, to achieve higher speed, YOLO was introduced. YOLO is a single-stage real-time algorithm and has lower accuracy than region-based algorithms. But it has a much higher speed than two-stage detectors. An image is seen only once in YOLO, as with the human visual system, while images are seen thousands of times in region-based algorithms. It does not utilise external algorithms (e.g., selective search in RCNN and RPN in Faster-RCNN). Thus, it is implemented in a shorter time. However, it has poor performance in small object detection and scale variation. It is frequently employed in many applications in light of its real-time functionality [34]. Later, YOLO-based detectors were introduced [35–38]. These networks have higher accuracy than the original Yolo, but they have a higher hardware volume and number of parameters.

In addition, several studies addressed how to make detection networks more accurate. A two-stage licence plate recognition algorithm based on YOLOv3 has been proposed in ref. [39]. In the initial stage, YOLOv3 is used to find the licence plate's location and then extract it. In ref. [40], a network has been proposed using a modified Darknet53 while utilising a scale-matching strategy to select suitable scales and anchor sizes for small target detection. In ref. [41], the region generation module and region prediction module have been proposed for object detection, and a feature pyramid strategy was adopted in the generation module to make full use of features in this. For improved localisation, a complex loss function and the anchor-less YOLOpeds detection method have been proposed in ref. [42]. In ref. [43] TRC-YOLO has been introduced, which decreases model size while increasing mean average precision (mAP) and real-time detection speed. In TRC-YOLO, the YOLOV4-Tiny convolution kernel reduces in size. Ref. [44] suggests a localisation technique based on global averaging to find moving objects. It has a feature pyramid enhancement strategy that improves the multi-scale detection performance of SSD detectors by utilising receptive fields, specific features, and semantic data. To improve object localisation, ref. [45] proposes relative motion estimation and global position optimisation methods. Ref. [46] generates an attention map that is valuable and useful for identifying automobiles because, in this map, segmentation effectively separates the background from the vehicles. In fact, to create the bounding boxes, a self-attention technique is applied using this map. In ref. [47], a YOLOv3-based network has been proposed, and the acquisition method of the anchor box has been improved by combining the Birch algorithm. Ref. [48] uses residual connections and their combinations at four different scales to develop a detection network.

Despite significant advancements in deep learning networks, small object detection and the trade-off between speed and precision in object detection remain challenging. The goal of the current study is to increase the algorithm's detection speed while maintaining accuracy. We prepare auxiliary maps using differential images, which has led to a light YOLO-based detector. We demonstrate that the algorithm presented in this study prevents the error from increasing over time, despite the

fact that several studies [14, 15, 49] report that the RMSE of trajectory predictors has been two to seven times over 5 s.

3 | PROPOSED ALGORITHM

The Fast-Positioning algorithm presented in this study (Figure 1) includes two parts: detection and prediction. It splits the input video into n frame packages, then prediction is performed in m frames of these n frame packages, and detection is performed in n-m frames. The switch between prediction and detection is done using a counter that counts incoming frames and is reset according to the n frame package size. It can be said that detection and prediction are implemented in two different ways. In the first path, the YOLO-based detector network is implemented, and in the second path, the recurrent prediction network is implemented; the detector increases the accuracy of the algorithm, and the predictor increases the speed of the algorithm. They complement each other in terms of accuracy and speed.

3.1 | Detection

This paper proposes a real-time detector network. The proposed network is known as a differential segmentation-based YOLO (DS_YOLO) network because it uses the SAMG module to separate the differential input images into two classes of vehicles and backgrounds. The original YOLO network can detect 80 classes of objects, but there are much fewer classes in issues involving intelligent transportation systems. As a result, simple feature extraction networks with fewer layers can be used instead of more complicated feature extraction networks like Resnet and VGG for traffic system detectors. The proposed DS_YOLO detection network is shown in Figure 2 as an improved version of YOLO-base detectors.

The earliest layers of the DS_YOLO network's extracted feature maps contain greater resolution and help a lot with the determination of vehicle positions. On the other hand, the final layers of the network offer more semantic depth and are more useful for categorising vehicles. The presence of two detection heads with different resolutions of 13 × 13 × 36 and 26 × 26 × 42 increases the DS_YOLO detector's resistance to scale changes. In addition, the segmentation-based feature maps produced in the SAMG module contribute significantly to vehicle positioning. The pixels of these maps are divided into two classes: foreground and background. In machine vision science, the foreground can be separated from the background using different methods, for example, motion detection-based techniques, mixture-of-Gaussian (MOG), and dynamic background calculation. The movement of the camera

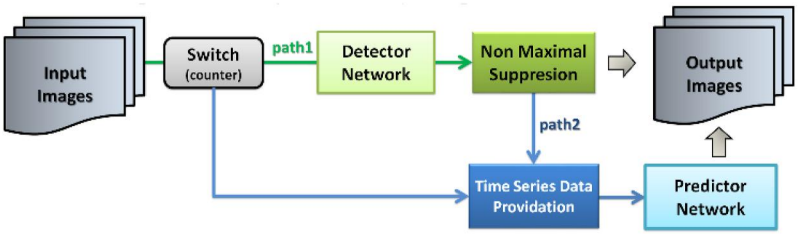

FIGURE 1 The schematic illustration of the proposed Fast-Positioning algorithm.

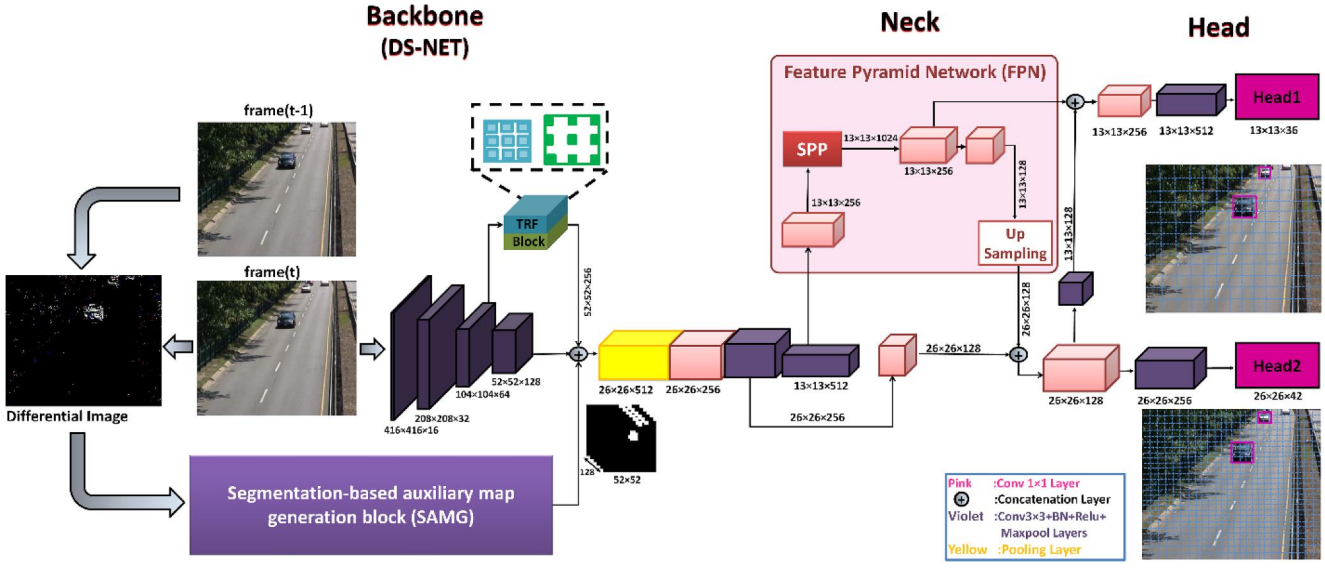

FIGURE 2 Proposed DS_YOLO network.

or other objects other than the background induces errors in these methods. The network-based method produces much better results than conventional non-network techniques. For example, Figure 3 shows an image segmented into the vehicle and background classes using two different methods: a differential image through dynamic background modelling (Figure 3b) and a network-based method (Figure 3c). The former method suffers errors in finding the foreground (i.e., vehicle) when the tree leaves move. Moreover, the vehicles that are separated from the background lack sufficient continuity. Another essential drawback is that this method can only separate moving vehicles from the background. As can be seen in Figure 3c, the deep method efficiently separates vehicles of different scales from the background and the motion of tree leaves and image shadows cannot induce errors in segmentation.

In this study, the segmentation is performed using differential images that are fed to a U-Net-based deep network (Figure 4), this network segments the feature maps into vehicle and background groups. Using differential images instead of RGB images increases the network's speed. Differential images remove the redundant and repeated data of consecutive images. As a result, when using them, emphasis is placed on useful information that is much smaller than the data of the whole image. Despite this, there is no need to increase the network depth to achieve higher segmentation accuracy.

The segmentation network contains an encoder and decoder, whose encoder is used in the proposed detector as the SAMG module. We train the SAMG block independently of the detector and then froze it while training the detector.

The SAMG block comprises three convolution layers with a 3 × 3 kernel, normalisation, a rectified linear activation function (Relu), and two pooling layers. In this module, an attempt has been made to reduce the learnable parameter of layers as much as possible because this study aims to increase the detection speed with accuracy comparable to fast detectors.

In addition to segmented maps, other factors help the vehicle detection network in the DS_YOLO network perform well, including the use of the TRF block in the DS_YOLO network, which improves accuracy. In this block, the residual connections play an effective role in dealing with the robustness of the detector to scale changes. These connections have been used in different detectors. In Figure 5, a structural comparison based on these connections between different blocks of several detectors is shown. Figure 5a shows two series residual branches in YOLOv3, while Figure 5b illustrates an Inception-like block. Figure 5c illustrates two parallel Residual branches [48], two residual connections in the YOLOV4 detector's CSP blocks are used, as shown in Figure 5d. Furthermore, Figure 5e indicates the TRF block architecture with one residual branch. The TRF block considers 3 × 3 and 5 × 5 receptive fields. In this block, two sequential convolutional layers with 3 × 3 receptive fields generate a 5 × 5 receptive field with higher accuracy than a convolutional layer with a 5 × 5 receptive field. Moreover, residual connections in the TRF block facilitate the transmission of information between layers,

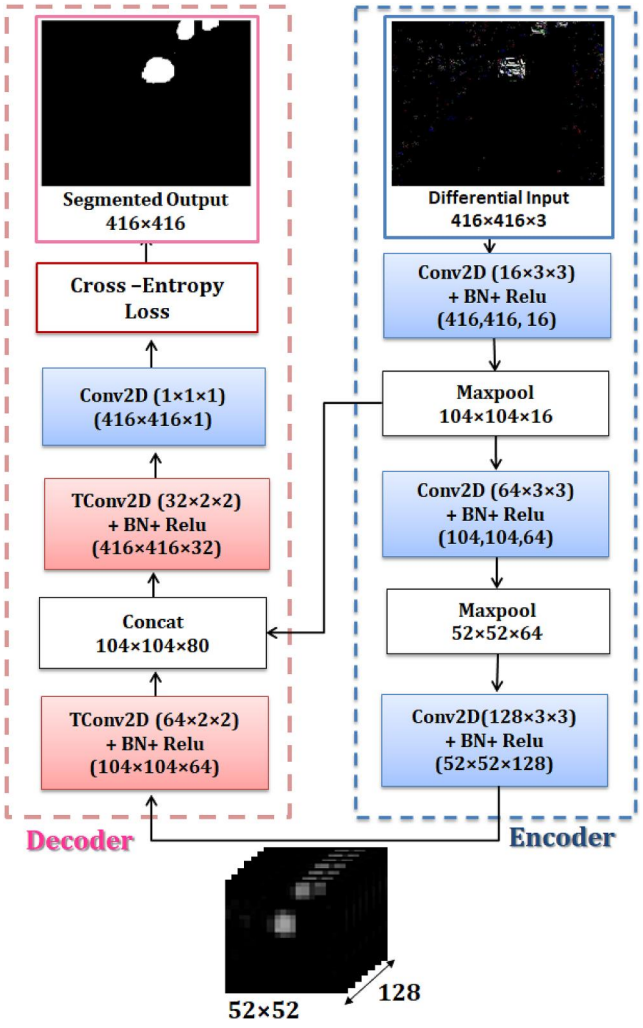

FIGURE 4 Differential segmentation network used in the proposed DS_YOLO network as SAMG block.

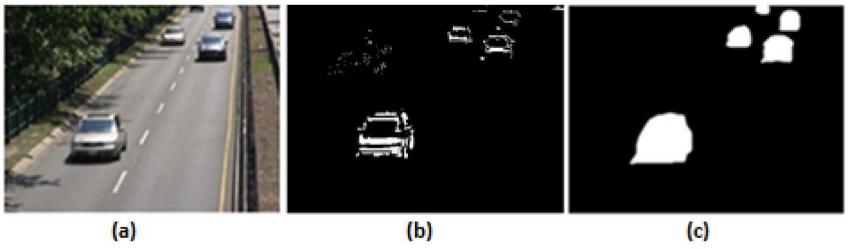

FIGURE 3 Example of image segmentation. (a) Input image. (b) Differential image through a dynamic background model. (c) Segmented image using a deep network.

contributing to network training. Since the TRF block is utilised in the initial layers of higher resolution than the other layers, more spatial details are considered in them. Thus, the position of a vehicle is more accurately recognised using this block. However, the TRF block has fewer parameters and higher speed since it uses fewer convolutional and concatenation layers.

The YOLO network uses fixed gridding. When there are several objects in the image, each of which is associated with a grid, YOLO can work well. However, when two objects in a grid overlap, anchor boxes must be employed to maintain high network performance. An anchor box is a predefined bounding box with a known height and width. Usually, these boxes are defined to record the scale and aspect ratio of the vehicle classes to be identified. They are selected based on the size of the vehicle in the training dataset. We carefully select the number and size of anchors in our proposed DS YOLO detector in order to achieve the best detection accuracy. The number of anchors is determined by the mean intersection over the union (MeanIoU) plot. The anchor size is determined using the K-means clustering algorithm in the two-dimensional space of the length and width and the three-dimensional space of the length, width, and width-length ratio of vehicles in the ground-truth data. The mean IoU for various numbers of anchors (k) is done and is found that $k = 10$ gives a good result for the evaluation of detection.

K-means clustering is an unsupervised learning algorithm aimed at finding k clusters of n unlabelled data points. Depending on how closely the data, are grouped into clusters; the data points of a given cluster have the highest similarity to each other and the lowest similarity to those of the other clusters. In fact, set $(x_1,x_2,x_3,\ldots,x_n)$, in which each data point is a d-dimensional real vector, is partitioned into $k \leq n$ sets $s = (s_1,s_2,s_3,\ldots,s_k)$ such that variance is minimised:

$$argmin_s \sum_{i=1}^{k} \sum_{x \epsilon s_i} \left\| x - \mu_i \right\|^2 = argmin_s \sum_{i=1}^{k} |s_i| \; Var(s_i) \quad (1)$$

where μ_i is the average of points in S_i. In Equation (1), the squared error between the cluster points is maximised. Since each vehicle in the ground-truth data is represented by not only its class but also length and width, the width-length ratio can be calculated as the third feature in clustering. Figure 6a plots the width-length ratio versus vehicle area in the ground-truth data.

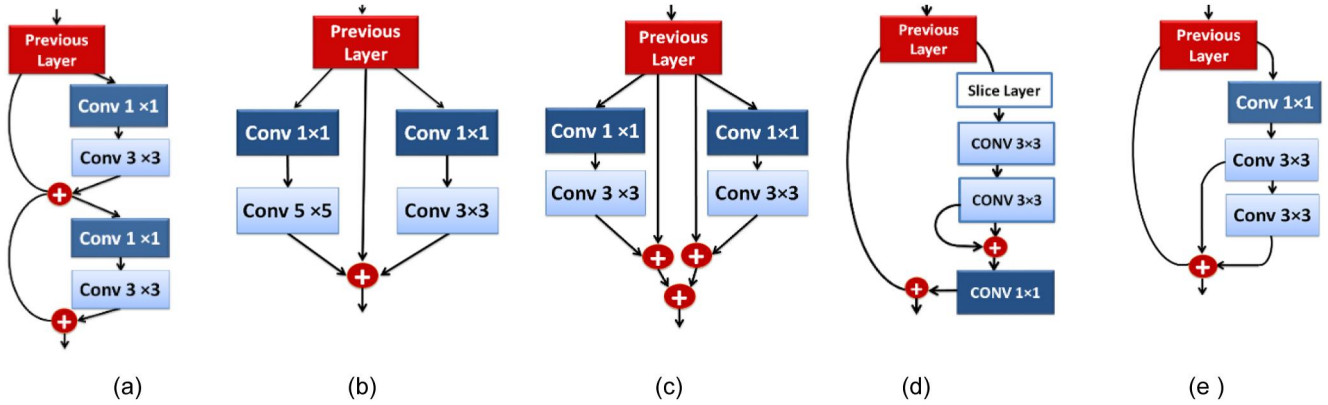

FIGURE 5 The comparison of building blocks with residual branches.

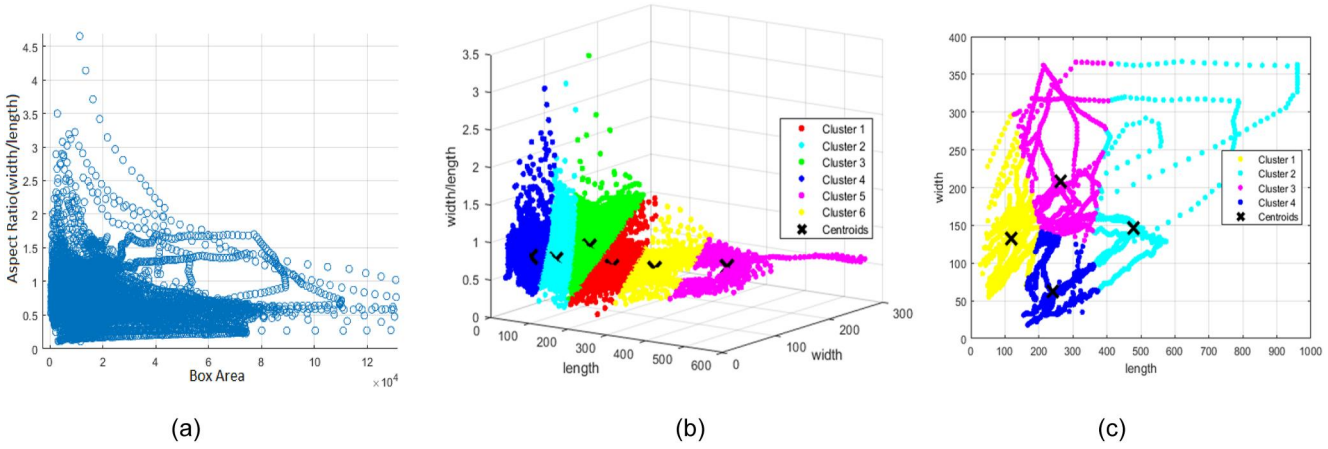

FIGURE 6 (a) Vehicle width-length ratio according to ground-truth on UA-DETRAC dataset. (b) Clustering box dimensions for car and van classes. (c) Clustering box dimensions for bus class.

These ratios are stored in an array to be utilised as the third dimension of the K-means algorithm input data. Considering the K-means algorithm and the selection of anchors at width-length ratios of 0.5 and 1.0 (as the most frequent ratios), the width-length ratio feature is not suitable for the detection of buses. Therefore, six anchors are employed in the three-dimensional space for cars and vans, while four other anchors are employed using the K-means algorithm in the two-dimensional space through the ground-truth bus data. Figure 6b shows the efficient anchor centres for cars and vans. Furthermore, Figure 6c depicts the bus anchor centres. Each point represents the feature of a vehicle in the ground-truth data, and the cluster centres stand for the determined anchor size. Therefore, a total of 10 anchors were used based on IoU for car, van, and bus classes.

3.2 | Prediction

In computer vision, approaches based on state estimation, such as the Kalman filter, are used to predict the position of vehicles. These methods are not suitable for accurate position prediction because they do not account for manoeuvres in vehicle motion. Additionally, position prediction using the Gaussian process and the Gaussian mixture model has heavy computation and isn't accurate. Additionally, it takes a long time and is highly expensive. As opposed to conventional position prediction models like the fixed acceleration model, where acceleration and velocity are presumed to be known, LSTM-based prediction networks are more precise. This presumption often isn't precise, though, as a vehicle's speed and acceleration may vary often depending on the flow of traffic and the driver's decisions. Therefore, it is better and more effective to use predictive networks, such as LSTMs. In this study, predictive networks are used alternately. The execution speed of these networks is higher than the detection networks, and for this reason, the intermittent use of these networks in consecutive frames increases the speed of vehicle positioning. The position history of the vehicle in the prior frames is used as input for prediction networks and prediction is done using them. Figure 7 shows the proposed Fast-Positioning algorithm architecture. It has sections on data preparation (DP), manoeuvres prediction and classification (MPC), and predicting vehicle position. The various sections are explained in the section that follows.

3.2.1 | Data Preparation

The input data of the prediction network are the positions of the vehicles that indicate the vehicle's trajectory.

$$X = \left[X^{(t-t_h)}, \ldots, X^{(t-1)}, X^{t}\right] \tag{2}$$

where t^h is a fixed (history) time horizon,

$$X^t = \left[x_0^{(t)}, y_0^{(t)}, x_1^{(t)}, y_1^{(t)} \ldots, x_n^{(t)}, y_n^{(t)}\right] \tag{3}$$

where x and y are vehicles ordinates at any time instant t and n is the number of vehicles present at the scene. To provide this data, each vehicle is given a unique label, and the label remains unchanged as long as that vehicle is seen in consecutive images. By using these labels, the positions of different vehicles are recorded separately in consecutive frames. To perform stable labeling, it is necessary to form the feature tensor for the vehicles. This tensor contains the feature vector of vehicles detected by the detector and is updated in each frame based on the output of the detector. The distance computed between the features of each vehicle detected in the current frame (at time t) is calculated from the feature tensor of vehicles in the previous frame (at time t-1), and its label is calculated according to the minimum distance between its features and the feature tensor. If the vehicle is detected for the first time in the current frame, a new label is assigned to it; otherwise, it is given the same label it had in the previous frame. Since labels are represented by numbers, the new label is equal to the last label in the preceding frame plus 1. Determining one of these two modes is achieved using the RMSE distance criterion and stored in $\overrightarrow{Dist}$.

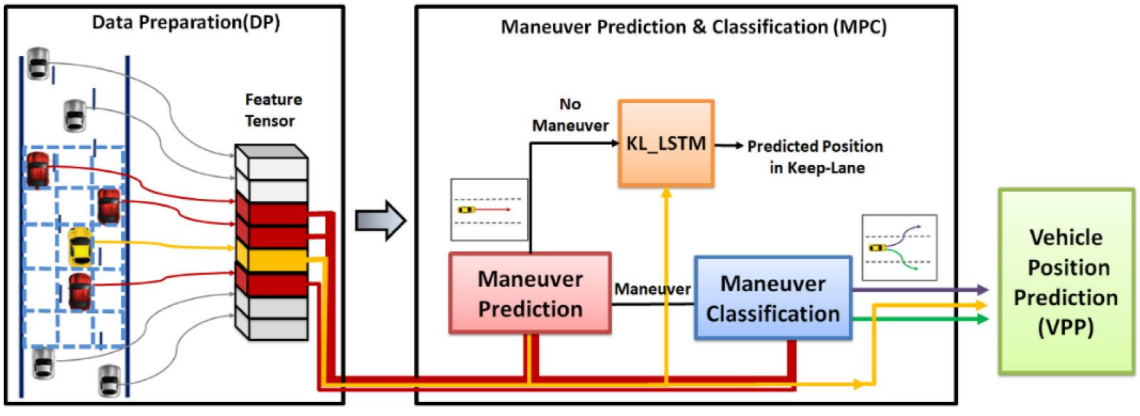

FIGURE 7 The structure of the proposed prediction algorithm.

$$Dist(i) = \sqrt[2]{\left(Loc_t - Loc_{t-1}^{i}\right)^2 + \left(Ar_t - Ar_{t-1}^{i}\right)^2 + \left(Im_t - Im_{t-1}^{i}\right)^2} \quad i = 1, \ldots, n \tag{4}$$

where location (Loc), area (Ar), and intensity mean (Im) are the elements of the feature vector of the vehicle being processed in the current frame ($\overrightarrow{Vf_t}$). Also, i denotes the vehicle label number and n denotes the number of detected vehicles in the previous frame's image. In this manner, the RMSE error for all detected vehicles in the current frame compared to the detected and labelled vehicles in the previous frame is calculated and stored in the $\overrightarrow{Dist}$. The pseudo-code of Algorithm 1 explains how to calculate the vehicle feature tensor for stable labeling.

Algorithm 1: Calculation of the feature tensor of vehicles for stable labeling

Inputs:

$\boldsymbol{Tensor}_{t\text{-}1,\,m}$, A tensor contains m rows, where each row includes the label of the vehicle and the features of any vehicles that were detected in the previous frame. The label is written in column 5, and features are written in columns 1 through 4 of each row.

Methods and variables:

RMSE, Root Mean Square Error function

Detect, the output of the detection task is the top-left corner location, length, and width of the vehicle.

$\overrightarrow{Dist}$, Array in which the distance between the vehicle feature in the current frame and the vehicles feature in the previous frame is written.

Loc_t, Vehicle location at time t (current frame)

Ar_t, Area of Vehicle at time t

Im_t, Intensity of Vehicle at time t

$\overrightarrow{Vf_t}$, Vehicle feature at time t

n, The row number from the previous frame tensor that has the shortest feature distance with the vehicle being processed in the current frame

L_{new}, New vehicle label, since labels are represented by numbers, the new label is equal to the last label in the preceding frame plus 1.

Th, A threshold limit is the smallest feature distance a vehicle can have in two consecutive frames.

L_t, Vehicle label at time t

Output:

Out_{Label}, output label

Objects = Detect (Image(t))
for each subset ⊆ Objects **do**
  $Loc_t \leftarrow Location(subset)$;
  $Ar_t \leftarrow Area(subset)$;
  $Im_t \leftarrow Intensity\ Mean(subset)$;
  $\overrightarrow{Vf_t}$, $\leftarrow (Loc_t, Ar_t, Im_t)$;
  for each $\overrightarrow{subset_2} \subseteq Tensor_{t\text{-}1,\,m}$ **do**
    $\overrightarrow{Dist}$, $\leftarrow RMSE\ (\overrightarrow{subset_2}, \overrightarrow{Vf_t})$ according to Eq. (4)
  end
  If (min ($\overrightarrow{Dist}$) < Th)
    $n \leftarrow$ Arg min ($\overrightarrow{Dist}$);
    $L_t \leftarrow \boldsymbol{Tensor}_{t\text{-}1,\,m}(n,5)$;
    $\boldsymbol{Tensor}_{t\text{-}1,\,m}(n,1{:}4) \leftarrow \overrightarrow{Vf_t}$;
    $\boldsymbol{Tensor}_{t\text{-}1,\,m}(n,5) \leftarrow L_t$;
  else
    $L_t \leftarrow L_{new}$
    $m \leftarrow m+1$
    $\boldsymbol{Tensor}_{t\text{-}1,\,m}(n,1{:}4) \leftarrow \overrightarrow{Vf_t}$;
    $\boldsymbol{Tensor}_{t\text{-}1,\,m}(n,5) \leftarrow L_{new}$;
  end
end
$\boldsymbol{Tensor}_{t,\,m} = \boldsymbol{Tensor}_{t\text{-}1,\,m}$
$Out_{Label} \leftarrow L_t$

Figure 8 shows the stable labeling results of several consecutive frames with different time intervals. Different vehicles are labelled with different numbers in different colours, and each vehicle has a specific label in consecutive frames. After labeling is complete, the position of each vehicle is stored in a unique array. The average and standard deviation of these arrays are calculated without including outliers. Then, the arrays are normalised based on them and given as the history of vehicle position as input to recurrent networks, and vehicle position is determined using recurrent networks.

3.2.2 | Maneuver prediction and classification

The trajectory of the vehicles is determined in the MPC section. To enhance the effectiveness of predictive networks, two parts of prediction and manoeuvre classification are also designed. Whether the vehicle is driving directly ahead or changing direction is determined by the trajectory prediction part. It does this by comparing the direction of the vehicle in the current frame with the average direction of its movement in the previous frames. The vehicle direction is calculated in the DP section. In lane-keeping mode, the Keep Lane LSTM network (KL-LSTM) is used to determine the vehicle position, and in the lane change mode, the Right Lane Deviation LSTM (RLD-LSTM) and Left Lane Deviation LSTM (LLD-LSTM) networks are used to determine the vehicle position. These networks have been trained for the two modes of right deviation and left deviation of the vehicle. It is the manoeuvre classifier network that determines which of these networks to use. These networks are trained by sampling different trajectories. Considering that vehicles do not change direction most of the time and maintain their previous trajectory, separating the two parts of path prediction and classification helps greatly increase the speed of the algorithm. By doing this, the classification part, which is more time-consuming than the prediction part, is performed less. In this study, accurate classification is accomplished by defining a spatial grid around the vehicle being predicted (Figure 9). The grid configuration is based on the location of the vehicles in the scene. In this study, a 5 × 3 grid is considered. The 7 × 5 temporal-spatial convolution tensor is formed according to this grid with zero-padding is 1. The position of the vehicle being predicted is determined by using the features of its surrounding vehicles in this grid. Figure 9 displays the grid and the relative spatial convolutional tensor.

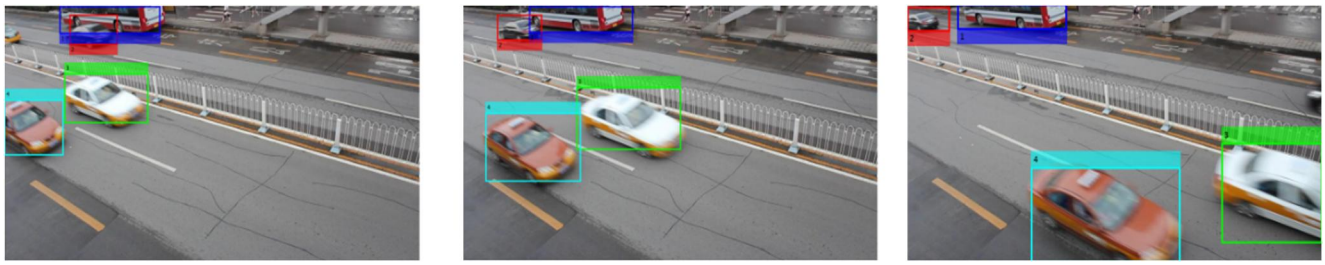

FIGURE 8 Visual results of proposed data preparation (DP).

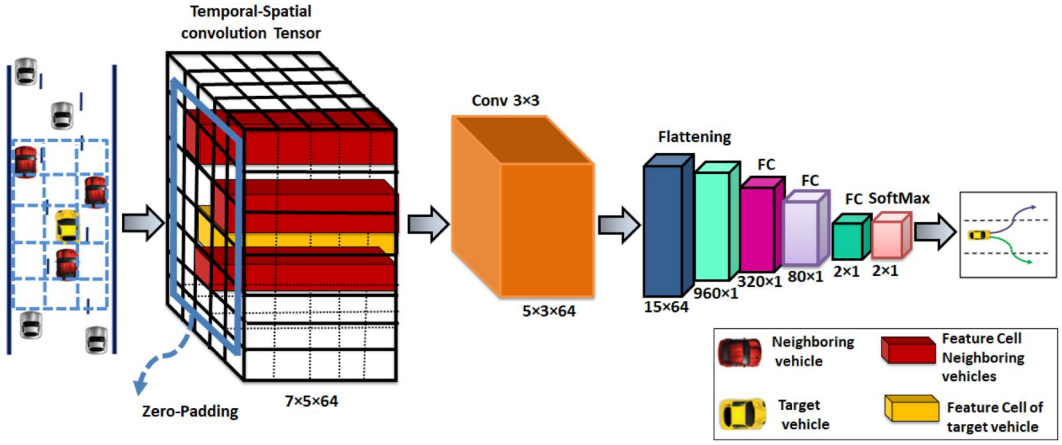

FIGURE 9 The structure of the proposed maneuver classification.

3.2.3 | Vehicle position prediction

The Vehicle position prediction (VPP) block in the algorithm presented in this study is executed when the probability of a manoeuvre is given in the MPC part. This block consists of two recurrent networks that are trained to predict the position of vehicles. These networks, which are named RLD-LSTM and LLD-LSTM, can be seen in Figure 10. The information obtained in previous frames is fed to the LSTM recurrent prediction networks, and the vehicle position is predicted by them. These networks have been trained for the two modes of right deviation and left deviation of the vehicle.

Because LSTM networks take long dependence in terms of time into account, they are more precise than traditional position prediction models like the constant acceleration (CA) model. These networks have been trained for the two modes of right deviation and left deviation. Recurrent networks learn how to predict vehicle trajectories with various types of acceleration over time. Because they have numerous nonlinear activation functions, they can predict complicated movement patterns as well as a range of trajectories with different accelerations. These functions select the data points from previously saved frames that should be retained or dropped. The gate operation of this network can be stated as follows:

$$\overrightarrow{i_t} = \sigma\left(\boldsymbol{W}_i\overrightarrow{X_t} + \boldsymbol{U}_i\overrightarrow{h_{t-1}}\right) \tag{5}$$

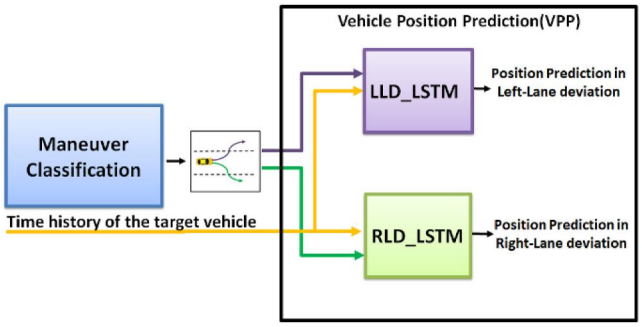

FIGURE 10 The structure of the vehicle position prediction.

$$\overrightarrow{f_t} = \sigma\left(\boldsymbol{U}_f\overrightarrow{h_{t-1}} + \boldsymbol{W}_f\overrightarrow{X_t}\right) \tag{6}$$

$$\overrightarrow{c_t} = \tanh\left(\boldsymbol{W}_s\overrightarrow{X_t} + \boldsymbol{U}_s\overrightarrow{h_{t-1}}\right) \tag{7}$$

$$\overrightarrow{o_t} = \sigma\left(\boldsymbol{W}_o\overrightarrow{X_t} + \boldsymbol{U}_o\overrightarrow{h_{t-1}}\right) \tag{8}$$

$$\overrightarrow{s_t} = \overrightarrow{c_t} \odot \overrightarrow{i_t} + \overrightarrow{s_{t-1}} \odot \overrightarrow{f_t} \tag{9}$$

$$\overrightarrow{h_t} = \overrightarrow{o_t} \odot \tanh\left(\overrightarrow{s_t}\right) \tag{10}$$

where '$\odot$' is the dot product, and $\overrightarrow{h_{t-1}}$ and $\overrightarrow{X_t}$ are the LSTM network inputs. The $\overrightarrow{h_{t-1}}$ is the output of the network in the previous frame, while $\overrightarrow{h_t}$ is the network output, and $\overrightarrow{X_t}$ is the input data in the current frame. The $\boldsymbol{W_f}$, $\boldsymbol{U_f}$, $\boldsymbol{W_i}$, $\boldsymbol{U_i}$, $\boldsymbol{W}_0$, and $\boldsymbol{U}_0$ respectively represent the weight matrices of the forgetting, input, and output gates of the network. The $\overrightarrow{s_t}$ is state cell and changed every frame. The LSTM network is employed to predict the vehicle location along multiple trajectories in the used datasets in study.

4 | RESULTS AND DISCUSSION

One of the advantages of deep networks is that can be trained for the intended application of the designer. This capability saves running time and hardware costs. For example, YOLO has been trained for 80 different classes, but many of these classes are unused in traffic control issues, and using them increases the algorithm's execution time. To perform a fair and scientific comparison, we equally treated all of the networks that we compared in Table 1 and trained them using the same data and the same classes. Of course, due to the existence of the SAMG module, which, unlike the detector, must have a fixed-size input, we resize the input image to a resolution of 416 at the beginning of the algorithm. The detection network is trained using the UA_DETRAC dataset with a batch size of 4 in 100 iterations. The SGD optimisation algorithm is implemented in training. The UA_DETRAC dataset includes four vehicle classes. Its image sequences have been captured through surveillance cameras in different locations. The images include vehicles with occlusion and scale variation. This dataset also includes other objectives than vehicles, such as humans and motorcycles. The tests were conducted in MATLAB with CPU computing facilities on a PC with a Corei7 6850k CPU with 32 GB of RAM.

4.1 | Evaluation methods

This study measures AP to evaluate detection network performance. It is calculated using the precision and recall criteria. Precision represents the percentage of true positive (TP) predictions (Equation (11)), whereas recall is the ratio of true positives to the total possible outputs (Equation (12)).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

A TP is when the model accurately predicts the detected vehicles' positive class. Similar outcomes, known as false negative and false positive, occur when the model predicts the positive and negative classes, respectively, incorrectly. The following equation is used to determine the AP computation for one class:

$$\text{AP} = \sum_{i=1}^{n-1} (r_{i+1} - r_i) pre(r_{i+1}) \tag{13}$$

where r_1, r_2, …, r_n are the recall levels at which the precision (pre) is first interpolated. Next, the mAP is the average AP across all classes, which can be defined as follows:

TABLE 1 Comparison of our proposed detector and object detection networks.

Methods	Input size	Parameters $\times 10^6$	FLOPs $\times 10^9$	Backbone	mAP (UA_DETRAC)	AP (highway)	Semantic attention mechanism	Multi head	FPS
SSD [50]	300	26.3	31.75	VGG-Net	71.40%	74.53%	False	False	46
RetinaNet [51]	-	32	156	FPN ResNet-50	76.52%	81.10%	False	False	12
Faster-RCNN [30]	-	134.7	181.12	VGG-Net	84.26%	89.26%	False	False	3
MaskRCNN [52]	-	63.7	149	ResNet-101-FPN	89.50%	93.47%	False	False	3
EfficientDet [53]	512	17	18	EfficientNets	65.48%	71.32%	False	False	21
YOLOV2 [35]	416	38.7	29.5	Darknet19	72.10%	79.56%	False	False	45
YOLOV3 [36]	416	49.2	65.8	Darknet53	86.14%	87.14%	False	True	47
YOLOV2 TINY [54]	416	7.5	5.4	FCCL	69.00%	74.30%	False	False	69
YOLOV3 TINY [55]	416	7.6	5.5	FCCL	78.30%	84.00%	False	True	78
YOLOV4 TINY [56]	416	6.06	6.96	FCCL	83.25%	86.25%	False	True	88
YOLOX TINY [57]	448	5.06	6.45	FCCL	85.00%	89.12%	False	True	58
YOLOV7 TINY [58]	416	6.2	5.8	E-ELAN-based	93.87%	95.24%	False	True	104
SSAM-YOLO [59]	416	4.28	3.1	SemAtt-Net	92.06%	96.40%	True	True	88
DS_YOLO (ours)	416	3.78	2.9	DS-Net	93.93%	95.32%	True	True	109

$$mAP = \frac{1}{M}\sum_{i=1}^{M} AP_i \tag{14}$$

where M denotes the number of classes. Another criterion to evaluate the detection network performance is missed detection rate (MR).

$$\mathrm{MR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}} \tag{15}$$

Another criterion for evaluating the detection algorithm is the frame rate of the positioning speed. Utilising RMSE, the efficiency of the position prediction net is also tested:

$$\mathrm{RMSE} = \sqrt{\left(\frac{1}{N}\right)\sum_{j=1}^{N}\left(\widehat{x}_j - x_j\right)^2 + \left(\widehat{y}_j - y_j\right)^2} \tag{16}$$

where N is the number of vehicles in the testing set. $[\widehat{x}_j^i, \widehat{y}_j^i]$ is the output of the prediction network, and $[x_j^i, y_j^i]$ is the actual position at time step j.

4.2 | Evaluation results of detection network

The proposed algorithm would maintain the box with the highest score and exclude the remaining boxes through the Non-Maximum Suppression filter when more than one box was detected for a given vehicle. Figure 11 illustrates the visual results of the DS_YOLO detection network. Figure 12 plots precision versus recall and Figure 13 plots the logarithmic target miss rate versus false positives for the detection networks DS_YOLO and several light and fast detectors. As can be seen, the proposed algorithm has higher accuracy than others despite fewer parameters for all three vehicle classes, that is, cars, buses, and vans.

Table 1 compares the number of parameters, FLOP, and mAP of DS_YOLO and other light and fast detectors. As you can see, despite having parameters that are 29% lower than those of YOLOV4 Tiny, our detector is more accurate. Also, it has a lower computational cost than YOLOV7-Tiny [58] and SSAM_YOLO [59]. Our detector has 40% fewer parameters and 50% fewer floating-point operations for YOLOV7-Tiny and 11.68% fewer parameters and 6.45% fewer floating-point operations for SSAM_YOLO. In terms of parameters, the YOLOV7-Tiny is larger than our proposed detector, and thus our proposed detector is much faster. The speed of our suggested detector is 109 fps and the speed of Yolov7-Tiny is 104 fps for an input image with a resolution of 416. Therefore, the suggested detector has less complexity and more effectiveness. In addition, DS_YOLO has a lower target miss rate than other detectors.

The output of the SAMG module effectively demonstrates its effectiveness. The vehicles stand out clearly from the background on these maps. It is incredibly helpful for network positioning and reducing miss rates to be able to distinguish between the target and background. Table 1 corresponds to the first successive convolution layers used by YOLOV2 Tiny and YOLOV3 Tiny as first consecutive convolution layers (FCCL). The parameters in FCCL layers are shown in Table 2.

FIGURE 11 Visual results of DS_YOLO detector network.

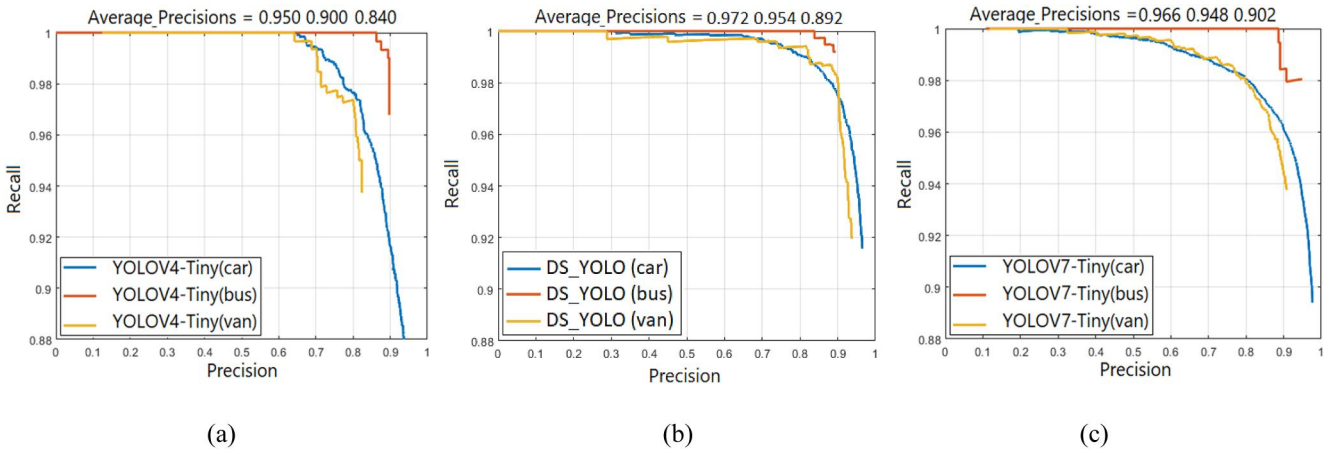

FIGURE 12 Precision recall curve performance for UA_DETRAC dataset, (a) YOLOV4-Tiny, (b) our proposed DS_YOLO, (c) YOLOV7-Tiny.

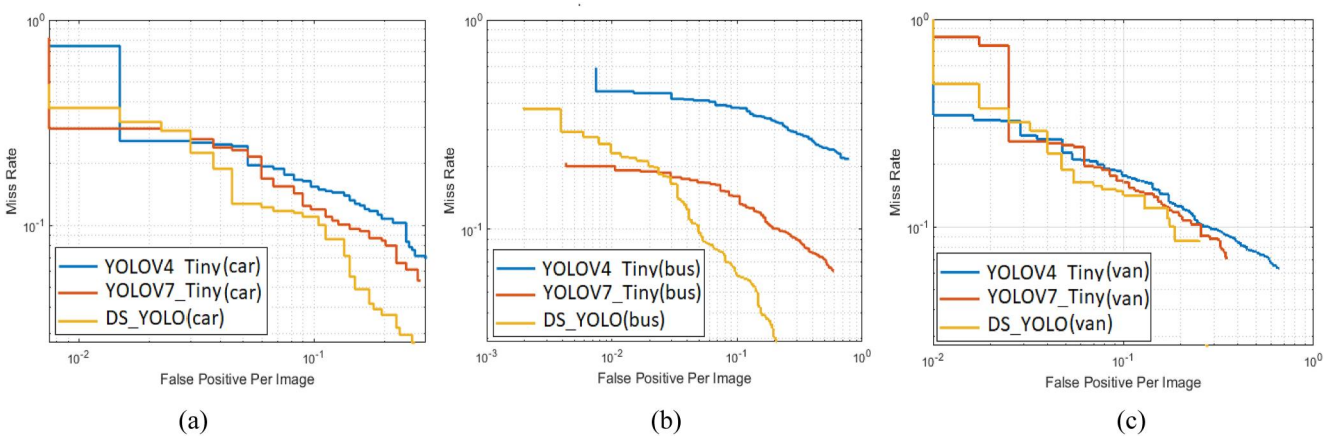

FIGURE 13 Comparison of our DS_YOLO and YOLO detectors for three classes (a) car, (b) bus, and (c) van.

Layers	Channel input	Kernel size	Channel output	Parameters number
Conv-1	3	3	16	448
Conv-2	16	3	32	4640
Conv-3	32	3	64	18,496
Conv-4	64	3	128	73,856
Conv-5	128	3	256	295,168
Conv-6	256	3	512	1,180,160

TABLE 2 Number of parameters in the first consecutive convolution layers (FCCL).

4.3 | Ablation experiments of detection network

An ablation study of the effects of the various components of the proposed detection network in this study is shown in Table 3. Our detector includes different parts: TRF, FPN, and SAMG. TRF improves the AP of our detector by 0.7%. The number of its parameter is 94,880. FPN adds 427,904 parameters, which improves the detector's AP by 2.45%. The SAMG module improves the detector's AP by 4.8%.

One of the innovations of our proposed detector is the use of differential images in the preparation of segmented feature maps using the proposed SAMG module. In differential images, the background of the image is removed, so the SAMG module has fewer tasks, and the number of layers and parameters can be chosen very lightly. But if the input of this module is RGB images instead of differential images, good maps will not be obtained, and to obtain similar maps, this module should be designed with more layers and parameters. In order to show the effectiveness of differential images in our

detector, we investigated both modes to prepare segmented feature maps in 52 × 52 resolution and reported the results: in the first mode, the network input is a differential image and the network name is SAMG Light, and in the second mode, the network input is an RGB image and the network name is SAMG_Heavy. We have chosen the first mode in our detector. In other words, SAMG_Light is the SAMG module in our suggested DS_YOLO detector.

The results reported in Table 3 show that the selection of differential images for the input of the SAMG module in the proposed detector has been very suitable and competent. Tables 4 and 5 list the layers and parameters associated with each model. Despite having more parameters than SAMG-Light with differential input, SAMG-Heavy with RGB input is 0.3% less accurate than SAMG-Light because differential images are crucial for background removal.

We also evaluated the performance of our proposed method on other detection backbones. We counted and reported the parameters up to a resolution of 52 × 52 × 512. The resolution's head-neck connection for the proposed detector is shown in Figure 14.

In addition, we have reported the number of parameters for complete backbones in Table 6. We trained our proposed network for these backbones with a batch size of 4 and a total of 200 epochs. The learning rate for the first half of the epochs is selected at 0.001; for the second half, it is selected at 0.0001. This training is additionally optimised using the Adam algorithms. Table 6 presents the findings of the evaluation of the AP and speed of the network trained using the backbones introduced for the Highway and UA-DETRAC datasets.

The results of Table 6 show that the proposed backbone of this study has better results than the other tested backbones.

TABLE 3 Ablation study of the proposed object detection network.

Differential input, SAMG-light	RGB input,	SAMG-heavy	TRF	FPN	Parameters × 10^6	AP highway	mAP UA-DERAC	FPS
×	✓	✓	✓	✓	8.376	95.02%	95.02%	65.6
×	✓	✓	×	×	7.853	91.87%	91.87%	64.7
×	✓	✓	✓	×	7.948	92.57%	92.57%	64.8
✓	×	×	×	×	3.257	92.17%	92.17%	110.2
✓	×	×	✓	×	3.352	92.87%	92.87%	109.8
×	✓	✓	×	✓	8.281	94.32%	94.32%	65.2
✓	×	×	×	✓	3.685	94.62%	94.62%	109.4
✓	×	×	✓	✓	3.78	95.32%	95.32%	109
×	✓	×	✓	✓	3.69	90.52%	95.32%	109

TABLE 4 Layers and parameters number of proposed SAMG-heavy network.

	Dim of feature-map	Parameters-number	Unit
Stage 1 (Conv + BN + Relu)	416 × 416 × 64	1792 + 128 + 0	Encoder
Stage 2 (Conv + BN + Relu)	208 × 208 × 128	73856 + 256 + 0	Encoder
Stage 3	104 × 104 × 256	295168 + 512 + 0	Encoder
Stage 4	52 × 52 × 512	1180160 + 1024 + 0	Encoder
Stage 5 (slice-layer)	52 × 52 × 128	0	Encoder
Stages 1–5		1,552,896	

TABLE 5 Layers and parameters number of SAMG-light network.

	Dim of feature-map	Parameters-number	Unit
Stage 1 (Conv + BN + Relu)	416 × 416 × 16	448 + 32 + 0	Encoder
Stage 2 (MaxPool)	208 × 208 × 16	0	Encoder
Stage 3 (MaxPool)	104 × 104 × 16	0	Encoder
Stage 4 (Conv + BN + Relu)	104 × 104 × 64	9280 + 128 + 0	Encoder
Stage 5 (MaxPool)	52 × 52 × 64	0	Encoder
Stage 6 (Conv + BN + Relu)	52 × 52 × 128	73856 + 256	Encoder
Stages 1–6		84,000	

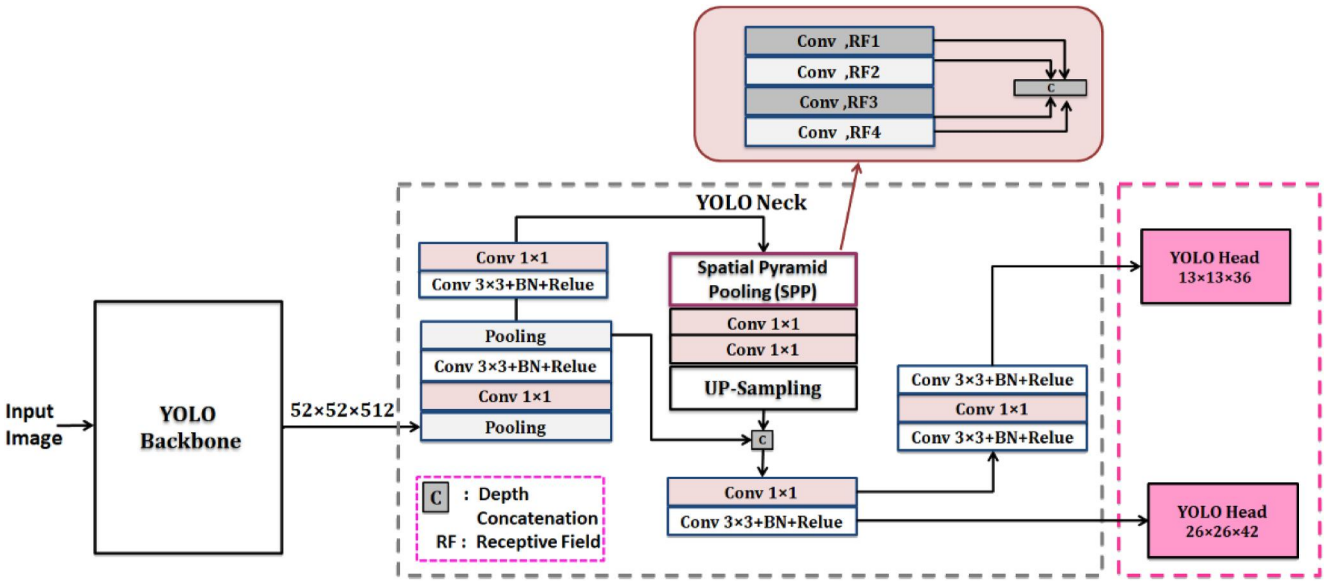

FIGURE 14 The structure of the proposed DS_YOLO network.

TABLE 6 Effectiveness of the proposed detector on other detection backbones.

Backbone	Input image size	Parameters × 10^6		AP (highway)	mAP (UA-DETRAC)	FPS
		Used part of the net (layers number)	Complete net AP (layers number)			
DarkNet53	416 × 416 × 3	3.205810 (90)	41.6 (184)	91.4%	90.1%	87.15
ResNet-50	416 × 416 × 3	1.169152 (78)	25.5 (177)	92.1%	91.4%	99.84
Densenet201	416 × 416 × 3	1.301632 (137)	20 (708)	93.4%	92.6%	68.34
VGG19	416 × 416 × 3	9.640824 (28)	143.6 (47)	90.1%	88.1%	100.12
Xception	416 × 416 × 3	0.195456 (30)	22.9 (170)	94.6%	92.3%	57.33
DS-Net (ours)	416 × 416 × 3	3.78	3.78	95.32%	93.93%	109.00

The reason is that we have focused on designing better feature maps through the SAMG module and using differential images for this module. We also increased the resistance of the proposed detector to scale changes through the TRF block in our backbone.

4.4 | Evaluation results of Fast-Positioning algorithm

In Sections 4.2 and 4.3, evaluations of the detector were carried out. In this section, the prediction network of the suggested algorithm and the outcomes of combining detection and prediction networks in the Fast-Positioning algorithm are examined and evaluated. The DS_YOLO detector network, trajectory classification, and recurrent prediction networks are used in Our *Fast-Positioning* algorithm to locate the position of the vehicle. As a result, the algorithm runs more quickly. In Figure 15b, a vehicle's actual trajectory is depicted in black, and our algorithm's predicted position is depicted in blue. Additionally, the CA model's position prediction is displayed in magenta. The vehicle that is being predicted is marked with a red box in Figure 15. In this figure, the vehicle turns to the left to change direction. The trajectory is successfully predicted, and the suggested model is more precise than the CA model, as can be seen. In traditional models such as the CA model, the location of a vehicle in the current frame (p_2) relative to the vehicle in the previous frame (p_1) is determined as follows:

$$p_2 = \frac{1}{2}a\Delta t^2 + v\Delta t + p_1 \tag{17}$$

where it is assumed that the acceleration a and the velocity v are known. However, this assumption is rarely correct, and the speed and acceleration of cars might alter numerous times based on the traffic conditions and the decision of the driver. Therefore, it is better and more effective to use predictive networks, such as LSTMs. The higher accuracy of our proposed model is due to the good performance of LSTM networks and the designed path classification.

Unlike the CA model, recurrent networks consider a longer time history for prediction. Figure 15c shows the RMSE error for these models. As shown in this figure, the error of our proposed method is lower than the CA model. To more accurately compare the two models, the prediction in Figure 15 was applied without using detector data in alternating frames, while in Figure 16, the results of our algorithm (Figure 16b) are shown more fully and compared with the results of our prediction network alone (Figure 16a). In this figure, the average error is shown in consecutive seconds. As can be seen, the error in our forecasting algorithm, unlike other research [14, 15], is not ascending, and this result is a very important issue in forecasting because it prevents network deviation. On the other hand, due to the mechanism of our proposed Fast-Positioning algorithm, the error is halved. This successful outcome is due to using our suggested detector's output in the odd frames. We have used prediction and detection networks in our algorithm to increase speed and accuracy; they complement each other. Also, our prediction is more accurate than ref. [59] because we have also calculated the effect of traffic agents in our manoeuvre classifier.

The positioning speed results of our proposed algorithm for different numbers of prediction and detection times in an alternating period are shown in Figure 17. The more frames in which prediction is used, the faster our proposed algorithm runs. By properly adjusting the number of times prediction and detection networks are used, it is easy to achieve the speed and accuracy required for different applications.

Figure 18 illustrates the time results of the proposed Fast-Positioning algorithm over an alternate period of n frames, where prediction is carried out m times and detection is carried out n-m times. The algorithm is run for various values of n and m, and the results are shown in Figure 18. Examples are 181 fps for $n = 3$ and $m = 2$, and 263 fps for $n = 4$ and $m = 3$.

In Figure 18, the red dashed line represents the average execution time of our suggested algorithm, which combines detection and prediction, while the black dashed line represents the average execution time of the predictor network, and the

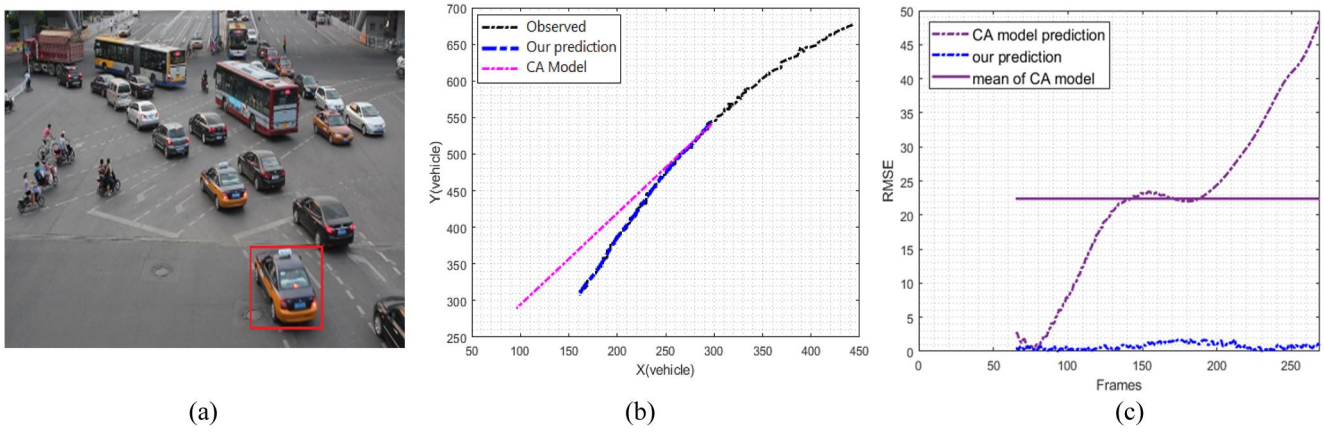

FIGURE 15 Prediction of vehicle trajectory; (a) marking the predicted vehicle with a red box, (b) our prediction and CA model for (a), (c) RMSE error.

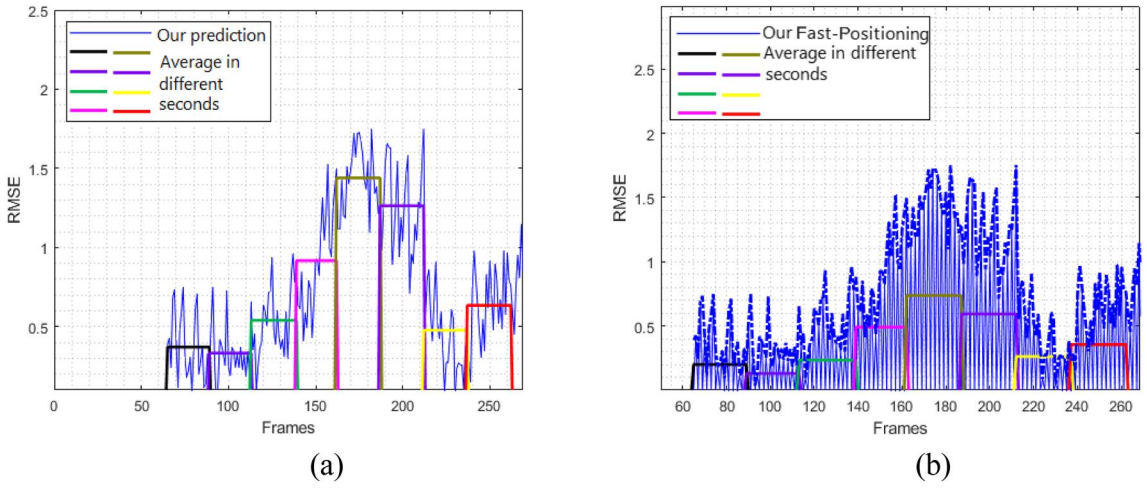

FIGURE 16 RMSE for (a) the proposed prediction model in different seconds, (b) the proposed Fast-Positioning algorithm that contains prediction and detection.

blue dashed line represents the average execution time of the detector network. As can be seen, the average time of our algorithm decreases as the number of frames in which the prediction is performed increases in n-frame cycles, while the average time of our algorithm increases as the number of frames in which the detection is performed increases in *n*-frame cycles.

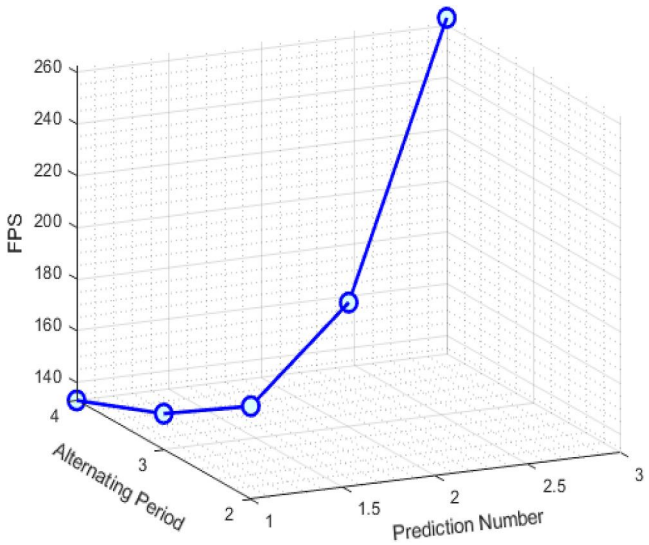

FIGURE 17 Speed results of our proposed algorithm for different numbers of prediction and detection times in an alternating period of time.

We changed the number of times we used the detector network and the prediction network in the alternating n-frame packet in our algorithm, and then we reported the results of these experiments in Table 7.

As can be seen, we have successfully varied the detector's speed over a wide range while only slightly altering the accuracy of the detector. As a result, our algorithm has been able to greatly increase vehicle positioning speed without sacrificing its accuracy and greatly improves the challenge between speed and positioning accuracy.

5 | CONCLUSION

The proposed algorithm in this study includes two parts: detection and prediction, and the needs of the user in terms of speed and accuracy, are very flexible. The detector (DS_YOLO) uses semantic attention in its SAMG block and extracts maps in which the input differential image is segmented into classes of background and vehicles, increasing the accuracy of the detector. The redundant and repeated data

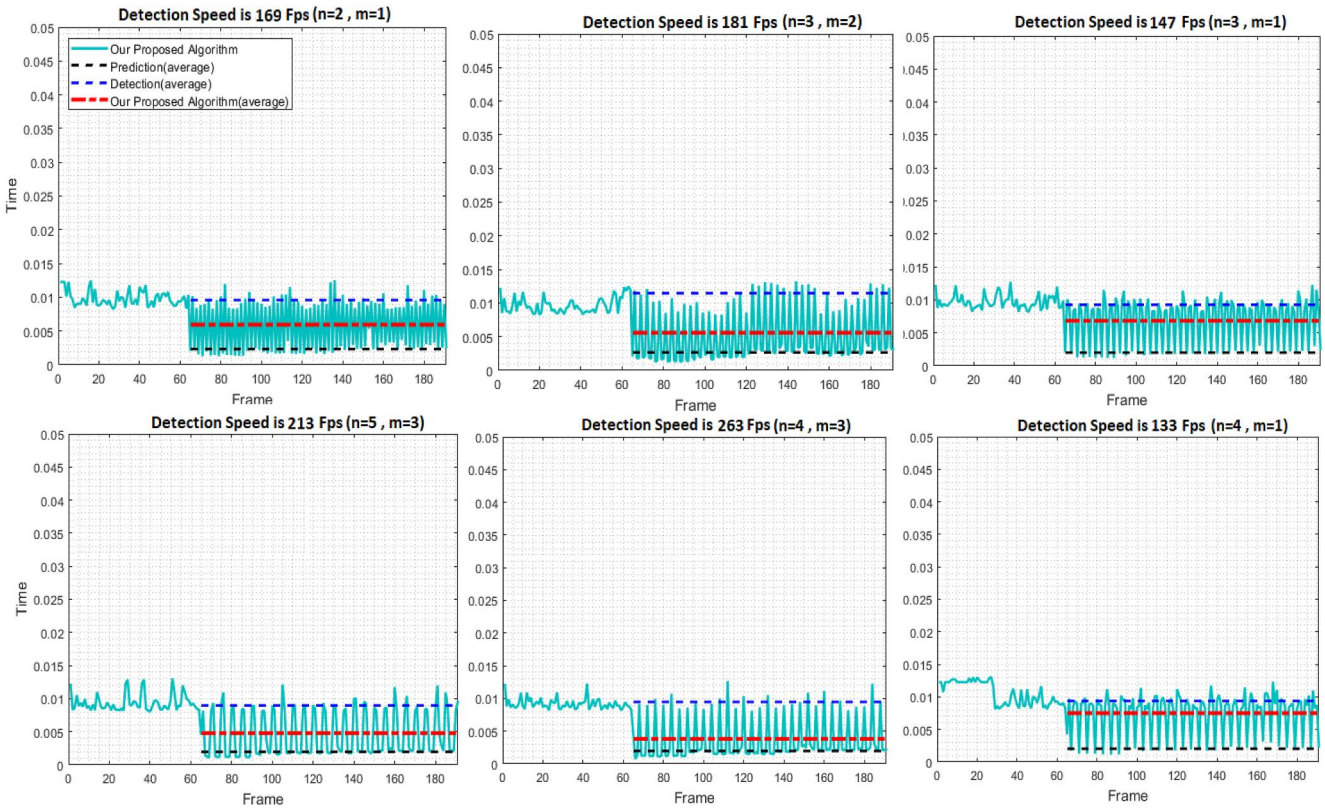

FIGURE 18 Time results of the proposed Fast-Positioning algorithm for various n-frame alternating periods in our algorithm, where the DS_YOLO detection network runs n-m times and the predictive network runs m times.

TABLE 7 Switching effects on the average precision and speed of the proposed *Fast-Positioning* algorithm.

Detection mode (*n-m*)	Prediction mode (*m*)	Period Switching (*n*)	AP (highway)	FPS
1	1	2	95.32%	169
1	2	3	95.25%	181
2	1	3	95.30%	147
1	3	4	94.82%	263
3	1	4	95.32%	133

in consecutive images are removed in differential images. As a result, with their existence, there is no necessity to increase the depth of the network to achieve better precision, and in this way, the speed of the detector network can be increased. Also, the TRF block in our proposed detector is designed to enhance the scale variation robustness. The use of efficient anchors in two-dimensional and three-dimensional spaces using a clustering algorithm further improved the accuracy of the proposed detection network. In the second path, vehicle trajectory classifier and position prediction networks are designed and implemented. The predictive network runs much faster than the detector. Another advantage is that can substantially alleviate the vehicle occlusion issue. The vehicle trajectory classifier network is designed to increase the accuracy of the predictive network. This classification is done by defining a 5×3 space grid around the projected vehicle in the scene and checking the movements of the vehicles placed in it. Two LSTM recurrent networks are trained to predict the position of vehicles. The information obtained from the proposed DS_YOLO detector is given to these networks in several frames, and the position of the vehicle is predicted by them in other frames. In this way, the accuracy of the predictive network increases. These networks are executed only if a manoeuvre has taken place. Also, our algorithm prevents the RMSE prediction error from ascendingly increasing over time. The effectiveness of the proposed Fast-Positioning algorithm for real-time vehicle detection is demonstrated by its implementation on the UA_DETRAC and CDNet2014(Highway) datasets.

AUTHOR CONTRIBUTIONS

Nafiseh Zarei: Conceptualization; data curation; formal analysis; investigation; methodology; resources; software; validation; visualization; writing – original draft. **Payman Moallem**: Conceptualization; funding acquisition; investigation; project administration; supervision; validation; writing – review & editing. **Mohammadreza Shams**: Investigation; software; supervision; visualization; writing – review & editing.

ACKNOWLEDGEMENTS

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no conflict of interest.

DATA AVAILABILITY STATEMENT

The datasets generated during and/or analysed during the current study are available from the corresponding author upon reasonable request.

ORCID

Payman Moallem https://orcid.org/0000-0001-7891-293X

REFERENCES

1. Tian, B., et al.: Hierarchical and networked vehicle surveillance in its: a survey. IEEE Trans. Intell. Transport. Syst. 18(1), 25–48 (2016)
2. Li, Q., et al.: A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. IEEE Trans. Veh. Technol. 63(2), 540–555 (2013). https://doi.org/10.1109/tvt.2013.2281199
3. Tao, Y., et al.: Low-altitude small-sized object detection using lightweight feature-enhanced convolutional neural network. J. Syst. Eng. Electron. 32(4), 841–853 (2021). https://doi.org/10.23919/jsee.2021.000073
4. Zhou, H., et al.: Video driven traffic modeling in paramics. In: 2013 UK Sim 15th International Conference on Computer Modelling and Simulation, pp. 525–530. IEEE (2013)
5. Kopsiaftis, G., Karantzalos, K.: Vehicle detection and traffic density monitoring from very high resolution satellite video data. In: 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 1881–1884. IEEE (2015)
6. Cao, X., et al.: Vehicle detection and motion analysis in low-altitude airborne video under urban environment. IEEE Trans. Circ. Syst. Video Technol. 21(10), 1522–1533 (2011). https://doi.org/10.1109/tcsvt.2011.2162274
7. Hinz, S.: Detection and counting of cars in aerial images. In: Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429), vol. 3. p. III-997. IEEE (2003).
8. Sivaraman, S., Trivedi, M.M.: Looking at vehicles on the road: a survey of vision-based vehicle detection, tracking, and behavior analysis. IEEE Trans. Intell. Transport. Syst. 14(4), 1773–1795 (2013). https://doi.org/10.1109/tits.2013.2266661
9. Mukhtar, A., Xia, L., Tang, T.B.: Vehicle detection techniques for collision avoidance systems: a review. IEEE Trans. Intell. Transport. Syst. 16(5), 2318–2338 (2015). https://doi.org/10.1109/tits.2015.2409109
10. Shirazi, M.S., Morris, B.T.: Looking at intersections: a survey of intersection monitoring, behavior, and safety analysis of recent studies. IEEE Trans. Intell. Transport. Syst. 18(1), 4–24 (2016). https://doi.org/10.1109/tits.2016.2568920
11. Chen, L., et al.: DenseLightNet: a light-weight vehicle detection network for autonomous driving. IEEE Trans. Ind. Electron. 67(12), 10600–10609 (2020). https://doi.org/10.1109/tie.2019.2962413
12. Bie, M., et al.: Real-time vehicle detection algorithm based on a lightweight You-Only-Look-Once (YOLOv5n-L) approach. Expert Syst. Appl. 213, 119108 (2022). https://doi.org/10.1016/j.eswa.2022.119108

13. Fan, S., et al.: FII-CenterNet: an anchor-free detector with foreground attention for traffic object detection. IEEE Trans. Veh. Technol. 70(1), 121–132 (2021). https://doi.org/10.1109/tvt.2021.3049805
14. Tang, L., et al.: Trajectory prediction for autonomous driving based on multiscale spatial-temporal graph. IET Intell. Transp. Syst. 17(2), 386–399 (2023). https://doi.org/10.1049/itr2.12265
15. Yan, J., et al.: Trajectory prediction for intelligent vehicles using spatial-attention mechanism. IET Intell. Transp. Syst. 14(13), 1855–1863 (2020). https://doi.org/10.1049/iet-its.2020.0274
16. Shao, W., et al.: Car detection from high-resolution aerial imagery using multiple features. IEEE Int. Geosci. Rem. Sens. Symp., 4379–4382 (2012)
17. Chen, Z., et al.: Vehicle detection in high-resolution aerial images based on fast sparse representation classification and multiorder feature. IEEE Trans. Intell. Transport. Syst. 17(8), 2296–2309 (2016). https://doi.org/10.1109/tits.2016.2517826
18. Chauhan, N.K., Singh, K.: A review on conventional machine learning vs deep learning. In: 2018 International Conference on Computing, Power and Communication Technologies (GUCON), pp. 347–352. IEEE (2018)
19. Helali, A., et al.: Hardware implementation of real-time pedestrian detection system. Neural Comput. Appl. 32(16), 12859–12871 (2020). https://doi.org/10.1007/s00521-020-04731-y
20. Chandrasekar, K.S., Geetha, P.: A new formation of supervised dimensionality reduction method for moving vehicle classification. Neural Comput. Appl. 33(13), 7839–7850 (2021). https://doi.org/10.1007/s00521-020-05524-z
21. Li, C., Xu, P.: Application on traffic flow prediction of machine learning in intelligent transportation. Neural Comput. Appl. 33(2), 613–624 (2021). https://doi.org/10.1007/s00521-020-05002-6
22. Maddalena, L., Petrosino, A.: A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection. Neural Comput. Appl. 19(2), 179–186 (2010). https://doi.org/10.1007/s00521-009-0285-8
23. Prabhakaran, N., Sudhakar, M.S.: Fuzzy curvilinear path optimization using fuzzy regression analysis for mid vehicle collision detection and avoidance system analyzed on NGSIM I-80 dataset (real-road scenarios). Neural Comput. Appl. 31(5), 1405–1423 (2019). https://doi.org/10.1007/s00521-018-3553-7
24. Minaeian, S., Liu, J., Son, Y.J.: Effective and efficient detection of moving targets from a UAV's camera. IEEE Trans. Intell. Transport. Syst. 19(2), 497–506 (2018). https://doi.org/10.1109/tits.2017.2782790
25. Sadeghi-Tehran, P., Clarke, C., Angelov, P.: A real-time approach for autonomous detection and tracking of moving objects from UAV. In: 2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS), pp. 43–49. IEEE (2014)
26. Cheraghi, S.A., Sheikh, U.U.: Moving object detection using image registration for a moving camera platform. In: 2012 IEEE International Conference on Control System, Computing and Engineering, vol. 23, pp. 355–359. IEEE (2012)
27. Bayoudh, K., et al.: A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets. Vis. Comput. 38(8), 1–32 (2021). https://doi.org/10.1007/s00371-021-02166-7
28. Girshick, R., et al.: 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 (2014)
29. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
30. Ren, S., et al.: Faster R-CNN: towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. 28 (2015)
31. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)
32. Cai, Y., et al.: Pedestrian detection algorithm in traffic scene based on weakly supervised hierarchical deep model. Int. J. Adv. Rob. Syst. 14(1), 1729881417692311 (2016). https://doi.org/10.1177/1729881417692311
33. Felzenszwalb, P.F., et al.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. 32(9), 1627–1645 (2009). https://doi.org/10.1109/tpami.2009.167
34. Redmon, J., et al.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
35. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017)
36. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767. (2018)
37. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.: Yolov4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. (2020)
38. Cai, Y., et al.: YOLOv4-5D: an effective and efficient object detector for autonomous driving. IEEE Trans. Instrum. Meas. 70, 1–3 (2021). https://doi.org/10.1109/tim.2021.3065438
39. Zou, Y., et al.: License plate detection and recognition based on YOLOv3 and ILPRNET. Signal Image and Video Processing. 16(2), 473–480 (2022). https://doi.org/10.1007/s11760-021-01981-8
40. Ju, M., et al.: A real-time small target detection network. Signal Image Video Process. 15(6), 1265–1273 (2021). https://doi.org/10.1007/s11760-021-01857-x
41. Li, Z., et al.: Real-time pedestrian detection with deep supervision in the wild. Signal Image Video Process. 13(4), 761–769 (2019). https://doi.org/10.1007/s11760-018-1406-6
42. Kyrkou, C.: 'YOLOpeds: efficient real-time single-shot pedestrian detection for smart camera applications. IET Comput. Vis. 14(7), 417–425 (2020). https://doi.org/10.1049/iet-cvi.2019.0897
43. Wang, G., et al.: TRC-YOLO: a real-time detection method for lightweight targets based on mobile devices. IET Comput. Vis. 16(2), 126–142 (2022). https://doi.org/10.1049/cvi2.12072
44. Zhao, M., et al.: Accurate and efficient vehicle detection framework based on SSD algorithm. IET Image Process. 15(13), 3094–3104 (2021). https://doi.org/10.1049/ipr2.12297
45. Xie, X., et al.: Accurate localization of moving objects in dynamic environment for small unmanned aerial vehicle platform using global averaging. IET Comput. Vis. 16(1), 12–25 (2022). https://doi.org/10.1049/cvi2.12053
46. Perreault, H., et al.: Spotnet: self-attention multi-task network for object detection. In: 2020 17th Conference on Computer and Robot Vision (CRV), pp. 230–237. IEEE (2020)
47. Huang, B., et al.: An improved YOLOv3-tiny algorithm for vehicle detection in natural scenes. IET Cyber Syst. Robot. 3(3), 256–264 (2021). https://doi.org/10.1049/csy2.12029
48. Zhang, F., et al.: CMNet: a connect-and-merge convolutional neural network for fast vehicle detection in urban traffic surveillance. IEEE Access 7, 72660–72671 (2019). https://doi.org/10.1109/access.2019.2919103
49. Messaoud, K., et al.: Non-local social pooling for vehicle trajectory prediction. In: 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 975–980. IEEE (2019)
50. Liu, W., et al.: Ssd: single shot multibox detector. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, pp. 21–37. Springer International Publishing (2016)
51. Lin, T.Y., et al.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)
52. He, K., et al.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)
53. Tan, M., Pang, R., Le, Q.V.: Efficientdet: scalable and efficient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020, pp. 10781–10790
54. Alsanad, H.R., et al.: Real-time fuel truck detection algorithm based on deep convolutional neural network. IEEE Access 8, 118808–118817 (2020). https://doi.org/10.1109/access.2020.3005391
55. Adarsh, P., Rathi, P., Kumar, M.: YOLO v3-Tiny: object detection and recognition using one stage improved model. In: 2020 6th International

Conference on Advanced Computing and Communication Systems (ICACCS), pp. 687–694. IEEE (2020)
56. Liu, Q., et al.: Object detection based on Yolov4-tiny and improved bidirectional feature pyramid network. In: Journal of Physics: Conference Series, vol. 2209(1). p. 012023. IOP Publishing (2022)
57. Ge, Z., et al.: Yolox: exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430. (2021)
58. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.: YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696. (2022)
59. Zarei, N., Moallem, P., Shams, M.: Fast-yolo-rec: incorporating yolo-base detection and recurrent-base prediction networks for fast vehicle detection in consecutive images. IEEE Access 10, 120592–120605 (2022). https://doi.org/10.1109/access.2022.3221942

How to cite this article: Zarei, N., Moallem, P., Shams, M.: Real-time vehicle detection using segmentation-based detection network and trajectory prediction. IET Comput. Vis. 18(2), 191–209 (2024). https://doi.org/10.1049/cvi2.12236