Received April 2, 2019, accepted April 25, 2019, date of publication May 1, 2019, date of current version May 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2914254

Video-Based Vehicle Counting Framework

ZHE DAI[1], HUANSHENG SONG[1], XUAN WANG[2], YONG FANG[1], XU YUN[1], ZHAOYANG ZHANG[1], AND HUAIYU LI[1]
[1]School of Information Engineering, Chang'an University, Xi'an 710064, China
[2]School of Computer and Control Engineering, Yantai University, Yantai 264005, China

Corresponding author: Huansheng Song (hshsong@chd.edu.cn)

This work was supported in part by the Natural Science Foundation of China under Grant 61572083, in part by the Ministry of Education Joint Fund Project of China under Grant 6141A02022610, in part by the Fundamental Research Fund for the Central Universities of China under Grant 310824173601, Grant 300102249304, and Grant 300102248303, in part by the Fundamental Research Funds for the Central Universities Team Cultivation Project under Grant 300102248402, and in part by the Funds for Shaanxi Key R&D Program under Grant 2018ZDXMGY-047.

ABSTRACT The continuous development in the construction of transportation infrastructure has brought enormous pressure to traffic control. Accurate and detailed traffic flow information is valuable for an effective traffic control strategy. This paper proposes a video-based vehicle counting framework using a three-component process of object detection, object tracking, and trajectory processing to obtain the traffic flow information. First, a dataset for vehicle object detection (VDD) and a standard dataset for verifying the vehicle counting results (VCD) were established. The object detection was then completed by deep learning with VDD. Using this detection, a matching algorithm was designed to perform multi-object tracking in combination with a traditional tracking method. Trajectories of the moving objects were obtained using this approach. Finally, a trajectory counting algorithm based on encoding is proposed. The vehicles were counted according to the vehicle categories and their moving route to obtain detailed traffic flow information. The results demonstrated that the overall accuracy of our method for vehicle counting can reach more than 90%. The running rate of the proposed framework is 20.7 frames/s on the VCD. Therefore, the proposed vehicle counting framework is capable of acquiring reliable traffic flow information, which is likely applicable to intelligent traffic control and dynamic signal timing.

INDEX TERMS Object detection, object tracking, trajectory processing, vehicle counting.

I. INTRODUCTION

Estimating the number of vehicles in a traffic video sequence is an essential component in intelligent transportation systems (ITS), which provides valuable traffic flow information. The number of vehicles on the road reflects traffic conditions, such as traffic status, lane occupancy, and congestion levels, which can be utilized for accident warnings, congestion prevention, and automatic navigation [1]. Meanwhile, traffic flow information at different intersections during different time periods is used for dynamic signal timing [2], which can improve traffic flow and provide substantial economic benefits.

In traditional ITS, special sensors such as magnetic coil, microwave, or ultrasonic detectors are primary tools to count vehicles. However, these sensors have limitations in obtaining detail information and installation cost. Video-based vehicle counting systems have begun to attract attention due to the development of image processing technology and its powerful capabilities. Compared to traditional sensor methods, these systems provide more traffic parameters, such as detecting vehicle category, density, speed, and traffic accidents for low costs, simple installations, and easy maintenance [3]. The machine vision vehicle counting method is an integrated procedure comprised of detection, tracking, and trajectory processing.

The detection of vehicle objects is the first step in obtaining traffic flow information. The purpose of object detection is to obtain the location and classification of the object from an image. Its primary task is to acquire the features of the object. In the past, it mainly relied on artificially designed features such as SIFT [4], HOG [5], and Haar-like [6], and then put these features into the classifier for training, such as SVM and Adaboost, etc. Felzenszwalb established a deformable part model (DPM) by using both HOG and SVM [7]. It exhibits better performance if the object has some deformation or scale change. However, this method cannot adapt to large rotations, has poor stability, and is slow to calculate.

The associate editor coordinating the review of this manuscript and approving it for publication was Huanqiang Zeng.

A classification framework based on deep convolutional neural network (DCNN) was previously reported. Its accuracy has been greatly improved compared with traditional classification algorithms, which lays a foundation for deep learning-based object detection research. Ross Girshick et al. used region proposal method to search all possible regions of the object, which is a selective search algorithm named regions with CNN (RCNN) [8]. Due to the slow detection speed of RCNN, He et al. proposed spatial pyramid pooling in deep convolutional networks for visual recognition network (SPPNet) [9]. Ross Girshickrb et al. proposed Fast RCNN [10], adding bounding box regression and multi-task loss function. Ren added a new region proposal network based on the fast RCNN algorithm and proposed the Faster RCNN [11]. The accuracy of the Faster RCNN has been greatly improved and is rated the best in all current detection algorithms, but the speed is one of its draw back. Liu proposed an end-to-end detection algorithm, single shot multibox detector (SSD) [12], which obtains proposal regions by uniform extraction and greatly enhances the detection speed. In the most recent year, Redmon et al. proposed YOLO V3 [13] by using multi-scale prediction and improving the basic classification network, with fast detection speed, low false detection rate, and strong versatility.

There are two main solutions for current object tracking. (1) Detect objects for each frame of the video sequence, complete tracking based on detection results of consecutive frames, and obtain the trajectory information. (2) Detect objects in the initial frame to obtain the features, search region to match the features in the subsequent image sequence, and track the objects to get the trajectory information. In the first solution, Meyer et al. proposed a contour-based target detection and tracking method which leads to a good effect, but this method has the disadvantages of poor anti-noise ability and high calculation volume. The object tracking based on deep learning [14], [15] has higher detection accuracy than the traditional algorithm. However, the detection is unstable, and there is a problem that the target may be missed in detection, which leads to tracking failure. The second solution relies less on object detection which avoided the disadvantage of the first one. The extraction of object features is the key of the program. One way is to extract feature points from the object. Commonly used feature point extraction methods include Harris corner detector [16] and SIFT [4]. Feature point-based methods [17], [18] can adapt to rotation and illumination changes of the object, but excessive feature extractions leads to difficult matching. Too few feature extractions are easy to cause false detection. Additionally, the feature extraction process is complicated and time consuming. Another way is to extract the feature of the object as a whole, including image edges, shapes, textures, color histogram, etc., and the reliability of the object features is enhanced by combination of multiple features. After feature extraction of the object, the similarity measurement is used to relocate the object to achieve object tracking. The feature-based tracking algorithm is insensitive to changes in the scale, deformation, and brightness of the moving object. Even if the object portion is occluded, the tracking task can be completed based on the partial feature information. However, it is sensitive to image blur, noise, etc. The extraction effect of features depends on the setting of various extraction parameters. In addition, the correspondence between consecutive frames is difficult to determine and has a negative impact on tracking performance.

The existing vehicle counting approaches can be mainly divided into three categories: regression-based methods [19]; cluster-based methods [20], [21]; and detection-based methods [22]–[24]. The regression-based method aims to learn the regression function using detection region characteristics. The cluster-based method is used to track features of objects to obtain their trajectories, and the trajectories are clustered to count the objects. The detection-based method can be further divided into four different categories: (1) frame difference method [25]; (2) optical flow method [26]; (3) background subtraction (BS) method [27], [28]; and (4) Convolutional neural network (CNN) method [29], [30]. The frame difference method is fast and simple. However, only parts of the moving objects are detected through the comparison between moving objects and background in consecutive frames. In contrast, the optical flow method calculates the optical flow information of the entire picture, resulting in a slower pace. The BS method uses background modeling to find moving objects by comparing the difference between the input image and the background. One of its main weaknesses is that it is difficult to build a good background model in a real scene. CNN is currently the most popular object detection method. In this method, the object of interest is marked in a large sample which builds up a dataset. Then a model is obtained by training the dataset using CNN. Eventually, objects of interest in a image can be detected by using both the model and CNN. There are, however, some common problems in the counting methods mentioned above: video angle limitations [31]; slow speed calculation [32]; and its inability to handle complex scenes [33].

In this study, we established two datasets, a vehicle counting dataset (VCD) and a vehicle detection dataset (VDD). Additionally, we proposed a video-based vehicle counting framework including three steps: object detection, object tracking, and trajectory processing. The VDD was used in the object detection step, and the VCD was used to validate the framework. Our results showed that the proposed framework is capable of obtaining reliable traffic flow information.

II. DATASET

A. VEHICLE COUNTING DATASET (VCD)

In this Section, we present the Vehicle Counting Dataset (VCD) introduced for the problem of vehicle counting in real-world conditions. It is used in the Experiment section for validation. This data is publicly accessible and can be downloaded from the following link: https://pan.baidu.com/s/1_iwXh8OijACMb5WTxyCuOg.

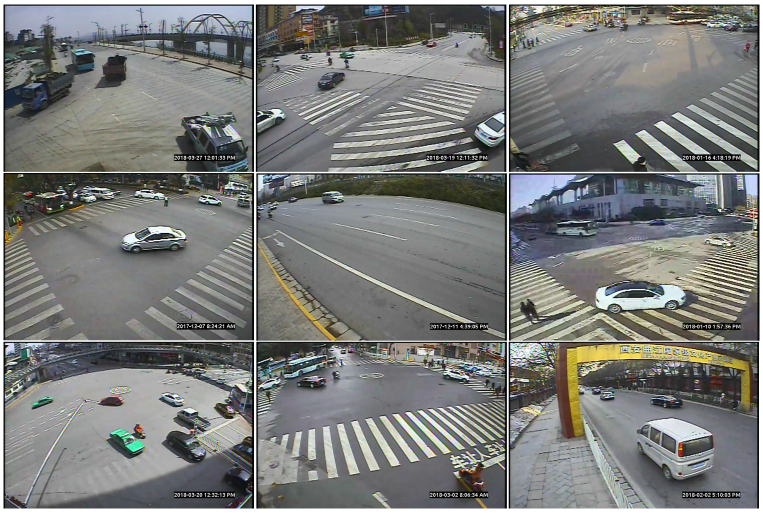
FIGURE 1. Different scenarios captures from VCD.

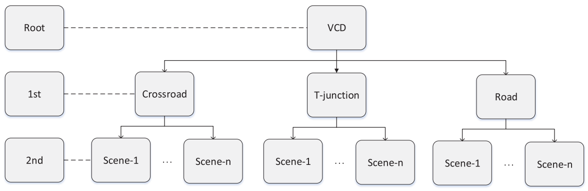

FIGURE 2. Folder structure of VCD.

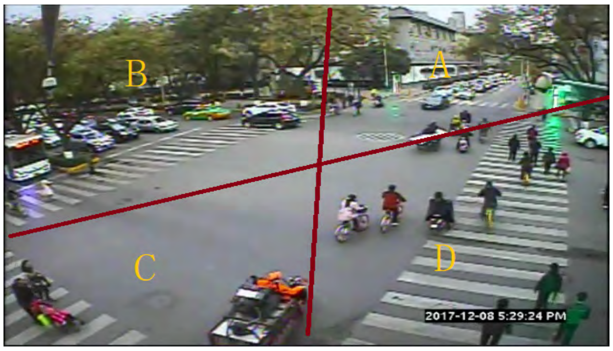

FIGURE 3. Region encoding diagram of the video named 20180205 in VCD.

The data consists of videos of urban road and intersection scenes recorded using Miovision camera (Miovision company). The camera is mounted on the roadside, and captures vehicle entering or exiting the different roads and intersections. Fig. 1 illustrates nine representative scenes from the dataset. Due to the real-world scenarios, complexity of the data is apparent from the Fig. 1. Videos in VCD were taken from diverse angles and time periods.

We divide the videos in the dataset based on the different road and intersection types with the data. There are three kinds of scenes in the dataset, crossroad, T-junction, and straight road. For crossroad, the data has been collected on 6 different days. We spent 2 days respectively for T-junction and straight road. Thus, in total, there are 10 different scenes in our dataset. In Fig. 2, we provide the folder structure of the proposed dataset.

More detailed description of Fig. 2, there is one video file in each scene. Each video lasts for an hour. The scene is named after the road or intersection. The video is named after the acquisition date. For each scene, we will give a screenshot of the video with encoding information as shown in Fig. 3.

For each video, we calculated the vehicle entry and exit road or intersection, and counted the numbers by vehicle category and vehicle moving route. The counting result was made into a label file, which was placed under a scene folder with a corresponding video. The label of counting result is shown in Table 1.

Table 1 takes the crossroad as an example, summarizes the number of vehicle in different type entering and exiting the area for each sub-category. Moreover, the total number is also provided.

B.Video information

In Table 2, we summarize the basic attributes of the videos in our dataset. We note that these video attributes along the camera parameter details are also provided in each folder of the dataset.

B. VEHICLE DETECTION DATASET (VDD)

In this Section, we describe our dataset VDD(Vehicle Detection Dataset) specifically designed for the task of vehicle object detection. The dataset is publicly available for download at the following URL: https://pan.baidu.com/s/1CHO3fjy00T1qOcN6ereqAQ. VDD is a vehicle dataset. It contains 6134 RGB images. Each image includes 720*480 pixels. There are corresponding artificially obtained annotations, including the size of the image, the objects number of the image, and the bounding box coordinate and classification of each object in the image. There are three categories in the dataset, car, truck, and bus. Fig. 4 shows the

TABLE 1. Label diagram of counting result named 20180205.

	From A			From B			From C			From D			Total
Route	A-B	A-C	A-D	B-A	B-C	B-D	C-A	C-B	C-D	D-A	D-B	D-C	
Mouth		412			2097			0			1373		
Car	145	158	109	402	230	1297	0	0	0	19	1172	79	1319
Truck	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus	0	0	1	0	0	123	0	0	0	0	103	0	227
Total	145	158	109	402	230	1465	0	0	0	19	1275	79	3882

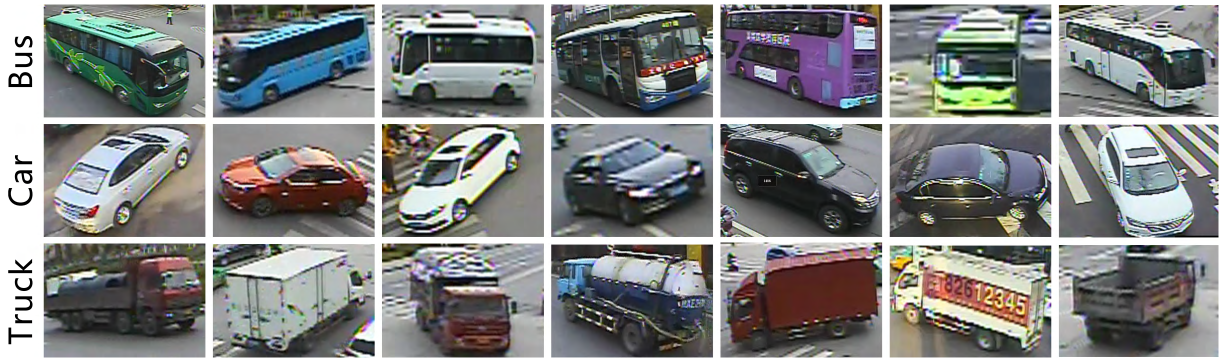

FIGURE 4. Category diagram of VDD.

TABLE 2. Results of abandoned object.

Type	FPS	Resolution	Duration	Format
RGB video	29	720*480	1 hour	MP4

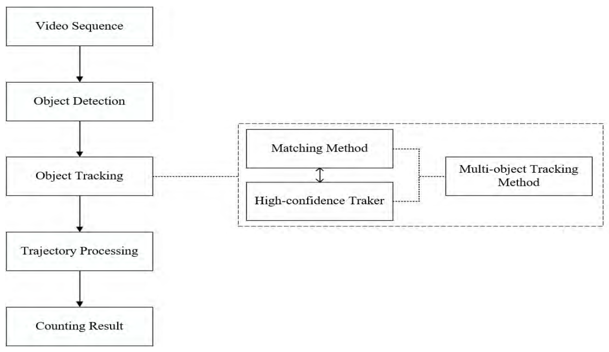

FIGURE 5. Strategy of the vehicle counting framework.

categories in the dataset, as well as 7 random images from each category.

III. PROPOSED FRAMEWORK

In this section, we introduced the proposed vehicle counting framework. Fig. 5 gives the framework of the proposed object detection and tracking system. There are three main stages in this framework: Object detection, Object tracking, Trajectory processing.

Object detection is done to give the bounding-box and classification of each object in one frame. Object tracking is used for obtaining trajectory of each object while it appears in the video stream. The object tracking is completed by a combination of proposed template matching method and KCF [17] algorithm. Trajectory processing provides a detailed result by using a region encoding method.

IV. OBJECT DETECTION

The aim of the object detection stage is to identify the category and location of the vehicle object in a picture. From the past ten years or so, the object detection algorithms for natural images are roughly divided into two categories. One is based on traditional manual features before 2013, the other one is based on deep learning used thereafter. Since the emergence of deep learning, the object detection has made a huge breakthrough. The most important two kinds of deep learning are: region proposal-based method represented by RCNN [8] (Fast-RCNN [10], Faster-RCNN [11], etc.; 2 regression-based method represented by YOLO (YOLOV3 [13], SSD [12], etc.). The former one is superior in accuracy, and the latter one is superior in speed.

Because deep learning method has excellent performance in object detection, this framework selects the YOLOV3 algorithm to complete the detection task. By training the VDD dataset that we introduced in 2.2, we can get the detection result information of the vehicle object in each frame of the video sequence.

$Frame_t$represents the all information of tth. We read the I_t from the video sequence in a loop. t represents the frame number, I_t represents the pixel information of the$t$th image, including three attributes, width (W), height (H), and size (S) of the tth image. We used $DB_t = \{BB_i, i = 1, \ldots, n\}$ represents the detection result of I_t. Each BB_i represents the detection result of one object and including 11 attributes: left-top($LT = x, y$), right-bottom($RB = x, y$), and center ($Cen = x, y$) coordinate, width (W), height (H), and size (S) of the bounding box of the detected object, the pixel information of the object bounding box in current image (*Roi*), the detected confidence(P), classification of the object (*Cls*), frame number (t), and a predict bounding box in next frame (*PB*). *PB* has the same attributes structure from *BB*, among them the *Roi* is taken from I_{t+1} by a tracking method, *Cls* and t keep consistent with corresponding *BB*. If there is no detected object in I_t, DB_t will be null. If a *BB* in I_t has no predict result in I_{t+1}, *PB* will be null. Finally, we bind I_t and DB_t to $Frame_t$.

V. OBJECT TRACKING

In the stage of object tracking, we proposed a muti-object tracking method based on detection. According to Input the $Frame_t$, outputting the all trajectories($TA = \{T_i, i = 1, \ldots, n\}$) of vehicle objects in the video. A $T_i = \{B_i, i = 1, \ldots, n\}$, and B_i could be a *BB* or a *PB*. The $TS = \{T_i, i = 1, \ldots, m\}$, $T_i = \{B_i, i = 1, \ldots, m, last\}$ is a intermediate variables, which is used to save trajectories that are being tracked. There are three attributes of each T, including a timer(*Timer*) that represents the number of consecutive occurrences of *PB* in a T, classification(*Cls*) that represents the vehicle category of the T, and length(*Len*) that represents the number of B_i of a T.

1) PROPOSED MATCHING METHOD

We proposed a matching method to complete the two tasks of creating new trajectory and tracking. At first, we need to calculate the overlap(*OL*) and distance(*Dis*) between two bounding box of two objects such as *BB* or *PB*. Cause *BB* and *PB* have the same structure of attributes, so we will describe the stuff as *BB* in the following text. Formula 1(BB_1,BB_2) is used to calculate the *OL*,

$$W = \min(x_1^{RB}, x_2^{RB}) - \max(x_1^{LT}, x_2^{LT})$$
$$H = \min(y_1^{RB}, y_2^{RB}) - \max(y_1^{LT}, y_2^{LT})$$
$$OL = \begin{cases} 0 & \text{if } W \leq 0 \text{ or } H \leq 0 \\ \dfrac{W * H}{S_1 + S_2 - W * H} & \text{otherwise} \end{cases} \tag{1}$$

where x_1^{RB} means the attribute x of *RB* of BB_1, so do x_1^{RB}, x_1^{RB}, andx_1^{RB}. The S_1 and S_2 means that the attribute S of BB_1 and BB_2. Formula 2(BB_1,BB_2) is used to calculate the *Dis*,

$$Dis = \sqrt{(x_1^{Cen} - x_2^{RB})^2, (y_1^{RB} - y_2^{RB})^2} \tag{2}$$

where x_1^{Cen} means the attribute x of *Cen* of BB_1, so do x_2^{RB}, y_1^{RB}, and y_2^{RB}. Then, matching value(*MV*) is calculated by *OL* and *Dis* in algorithm 1.

2) SELECT A HIGH-CONFIDENCE TRACKER

In order to complete muti-object tracking, a high-confidence tracker is also needed to provide some additional information. We choose KCF algorithm [17] as the high-confidence tracker. KCF method has a *KCF* tracker with inputting a *BB* in I_{t1}. Then using the extraction method $init() \Leftarrow KCF$ to obtain the features of the *BB*. Finally, use the search method $update() \Leftarrow KCF$ to find the object in another image I_{t2}.

Algorithm 1 Calculation of matching values of two bounding boxes.

```
Require: $BB_1, BB_2, I_t$
Ensure: $MV$
  $OL$ = Formula 1($BB_1, BB_2$)
  $Dis$ = Formula 2($BB_1, BB_2$)
  $IDis = \sqrt{(W \Leftarrow I_t)^2 + (H \Leftarrow I_t)^2}$
  if $OL = 0$ or $\frac{Dis}{IDis} \geq \max(W, H \Leftarrow BB_1)$ then
    return $MV = 0$
  else
    return $MV = 0.8 \leq OL + 0.2 \leq \frac{Dis}{IDis}$
  end if
```

Algorithm 2 shows that how we use KCF to support the tracking stage.

Algorithm 2 Tracking an object using KCF algorithm.

```
Require: $BB \Leftarrow I_{t1}, I_{t2}, KCF$
Ensure: $PB \Leftarrow BB$
  initial $init(), update() \Leftarrow KCF\ init(Roi \Leftarrow BB)$
  if $PB = update(I_{t2})$ not $null$ then
    return $PB$
  else
    return $null$
  end if
```

3) FIND THE NEXT POSITION OF A TRAJECTORY

This part introduces that how to find the next position of a trajectory. By inputting the current frame $Frame_t$ and TS. First, we use algorithm 1 to match the $B_{last} \Leftarrow T_i \Leftarrow TS$ with $BB_i \Leftarrow DB_t$, simultaneously perform algorithm 2 with $B_{last} \Leftarrow T_i \Leftarrow TS$ and get $PB \Leftarrow B_{last} \Leftarrow T_i$ as feedback. Obviously, due to the difference between the number of trajectories and the number of detected objects, there are three cases after the processing of algorithm 1: case1, $B_{last} \Leftarrow T_i$ successful matching, then add the matched BB_i as next node to the T_i; case2, $B_{last} \Leftarrow T_i$ matching failed, add the $PB \Leftarrow BB_i$ as next node to the T_i. If *PB* is null, then do not add a new node to the T_i. Finally add 1 to the value of the $Timer \Leftarrow T_i$; case3, $BB_i \Leftarrow DB_t$ matching failed, then create a new trajectory T_{new} with using BB_i as the first node of it and add T_{new} to TS. Algorithm 3 is shown as followed:

4) UPDATE APPROACH

When the objects have left from the video, we need to move the trajectories from TS to TA. Two conditions are used to judge the departure of the object: $Timer \Leftarrow T_i > 30$, or $B_{last} \Leftarrow T_i$ is located on the edge of the image from video.

5) MULTI-OBJECT TRACKING METHOD

The whole process of the multi-object tracking method we propose is: Firstly, read the video frame in a loop and obtain the detection result of each frame. Secondly, the strategy of

Algorithm 3 Tracking objects.

```
Require: $TS, Frame_t$
Ensure: $TS$
  initial $P \Leftarrow BB_i \Leftarrow DB_t > \xi, I_t \Leftarrow Frame_t$ $init(Roi \Leftarrow BB)$
  for all $BB_i \Leftarrow DB_t$ with $B_{last} \Leftarrow T_i$ do
    algorithm 1$(BB_i, BB_{last})$
    algorithm 2$(B_{last}, I_t)$
  end for
  case 1:
  $B_{n+1} \Leftarrow T_i = B_{last} \Leftarrow T_i$
  $B_{last} \Leftarrow T_i = BB_i$
  case 2:
  $B_{n+1} \Leftarrow T_i = B_{last} \Leftarrow T_i$
  $B_{last} \Leftarrow T_i = PB$
  case 3:
  $T_{n+1} = BB_i$
```

Algorithm 4 Obtaining of the complete trajectory of each object.

```
Require: $TS, TA, I_t$
Ensure: $TS, TA$
  for all $T_i \Leftarrow TS, Cen, W, H \Leftarrow B_{last \Leftarrow T_i}$ do
    if $Timer \Leftarrow T_i > 30$ or $B_{last} \Leftarrow T_i$ out of $I_t$ border then
      if $Len \Leftarrow T_i > \eta$ then
        Add $T_i$ to $TA$
      end if
      Delete $T_i$ from $TS$
    end if
  end for
  return $TS, TA$
```

finding the next node of the trajectory is used to continuously track the object. Finally, the update approach is also a necessary step to ensure the stability of tracking, and the complete trajectory of all the objects in the video is stored. Algorithm 5 illustrates the whole process.

Algorithm 5 Obtaining of node of each complete trajectory.

```
Require: sequence of $Frame_t$
Ensure: $TA$
  initial $TS = \{\}, TA = \{\}$
  for all $Frame_t$ do
    $TS$ = algorithm 3$(Frame_t, TS)$
    $TS, TA$ = algorithm 4$(TA, TS)$
  end for
  return $TA$
```

VI. TRAJECTORY PROCESSING

Due to the complexity of the scene and the uncertainty of vehicle motion, the trajectory obtained from V-.2 is cluttered. In order to calculate the traffic flow information of roads and intersections, the moving direction and category information of vehicle objects are particularly important. The purpose of the trajectory processing is to obtain the direction and category information of each trajectory, and classify and count the numbers.

The first step is to confirm the category of each track (where car = 1, bus = 2, and truck = 3), and by weighting all the node frame categories in a track, the class of its track is weighted. The specific algorithm is as follows:

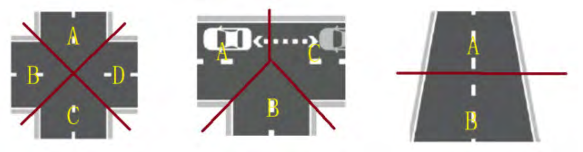

FIGURE 6. Region encoding of different scenes.

Algorithm 6 Calculating category of each object in the node of trajectory.

```
Require: $TA$
Ensure: $TA$
  initial $B_i \Leftarrow T_i$
  for all $T_i \Leftarrow TA$ do
    $Cls \Leftarrow T_i = \frac{num}{den}$
  end for
  return $TA$
```

Then, clustering is performed by extracting the coordinates of the starting point of the trajectory, and several intersection areas in the image are obtained. The partition model is shown in Fig. 6. Finally, the vehicle counting result are completed according to the region encoding model and the location area of the start and end points of the trajectories.

VII. EXPERIMENTAL RESULTS

A. DETECTION RESULT

In this section, the main purpose of the experiment is to choose an optimal detection algorithm that can be applied in our framework. In order to reach a good detection effect and a better result, we selected to use deep learning method. With different deep learning algorithm, we trained the training set of VDD as proposed in 2.2 and verified it on the test set to obtain the results of different networks in terms of detection speed and accuracy. We have selected the most advanced SSD300, SSD512, YoloV3, and Faster-RCNN algorithms to compare their outcomes of object detection. We performed the following experiment on a single 1080Ti graphics card. In the training, we used the 0.001 learning rate for 40 k iterations, then continued training for 10 k iterations with 0.0001 and 0.00001, respectively. When using the trained model for validation, we determined the precision and speed using threshold of IOU (Intersection over Union) = 0.5 and $\xi = 0.7$. Precision and recall were calculated as:

$$Recall = \frac{TruePositive}{TruePositive + FalsePositive},$$

TABLE 3. Result of detection on VDD test.

Method	Avg.Precision(%) car	bus	truck	mAP(%)	Avg.Recall(%)	FPS
Faster-RCNN	90.7	90.2	89.4	90.1	81.9	7
SSD-300(InceptionV2)	83.1	85.6	84.8	84.5	73.8	56
SSD-512(VGG16)	91.0	88.4	87.9	89.1	79.5	23
YOLOV3(Darknet-19)	89.1	87.3	86.4	87.6	79.2	78

Avg. , average; mAP, mean average precision; FPS, frame per second

TABLE 4. The label result of the video from VCD named 20180324.

Label	From A			From B			From C			From D			Total
Route	A-B	A-C	A-D	B-A	B-C	B-D	C-A	C-B	C-D	D-A	D-B	D-C	
Mouth		522			307			569			176		
Car	17	399	39	14	241	32	252	178	59	22	67	72	1392
Truck	0	43	6	0	0	0	0	39	7	0	0	5	100
Bus	0	18	0	0	12	8	25	9	0	0	10	0	82
Total	17	460	45	14	253	40	277	226	66	22	77	77	1574

TABLE 5. The framework result of the video from VCD named 20180324.

Framework	From A			From B			From C			From D			Total
Route	A-B	A-C	A-D	B-A	B-C	B-D	C-A	C-B	C-D	D-A	D-B	D-C	
Mouth		498			291			583			154		
Car	13	367	44	11	223	34	226	218	55	7	75	46	1319
Truck	2	46	5	0	1	0	2	41	7	0	2	8	114
Bus	0	21	0	2	9	11	19	15	0	1	15	0	93
Total	15	434	49	13	233	45	247	274	62	8	92	54	1526

TABLE 6. The vehicle counting error of the video from VCD named 20180324.

Error(%)	From A			From B			From C			From D			Total
Route	A-B	A-C	A-D	B-A	B-C	B-D	C-A	C-B	C-D	D-A	D-B	D-C	
Mouth		-4.6			-5.2			2.4			-12.5		
Car	-23.5	-8.0	12.8	-21.4	-7.4	6.2	-10.3	22.4	-6.7	-68.1	11.9	-36.1	-5.2
Truck	/	6.9	-16.6	0	/	0	/	5.1	0	0	/	60.0	14.0
Bus	0	16.6	0	/	-25.0	37.5	-24.0	66.6	0	/	50.0	0	10.9
Total	-11.7	-5.6	8.8	-7.1	-7.9	12.5	-10.8	21.2	-6.1	63.6	19.5	-29.8	-3.1

$$Precision = \frac{TruePositive}{TruePositive + TrueNagtive} \tag{3}$$

TruePositive indicates the target of correct detection. *FalsePositive* indicates that there are no detected targets. *TrueNagtive* indicates the detection of an error.

The resolutions of the input images for Faster-RCNN, SSD300, SSD512, and YOLOV3 are 720*480, 300*300, 512*512, 416*416, respectively. Apparently, larger input size led to better precision and recall. Faster R-CNN exhibited the best precision but the lowest speed. SSD300 and YOLOV3 are both real-time detection method, but the mean average precision of YOLOV3 was 3.1% higher. By using images with larger input size, SSD512 displayed 1.5% higher in mAP than YOLOV3, but the speed was over three-fold slower (Table 3). Taken together, we recommend using the YOLOV3 algorithm to detect the object in the vehicle counting framework.

B. COUNTING RESULT

Method is based on the category and moving route of vehicle. In this paper, there are three categories of vehicle and three moving routes at each mouth of an intersection. Therefore, we have 36, 18, and 6 counters of crossroad, T-junction, and

TABLE 7. The label result of the video from VCD named 20180328.

Label	From A		From B		From C		Total
Route	A-B	A-C	B-A	B-C	C-A	C-B	
Car	182	518	269	19	409	4	1401
Truck	19	0	0	0	21	4	44
Bus	12	20	16	0	12	0	60
Total	213	538	285	19	442	8	1505

straight road, respectively. We performed counting experiment using the videos of the VCD dataset by our framework and compared the counting results with those of the labels. Threshold η in Algorithm4 represents that when the number of nodes of a trajectory is less than η, it is considered to be an error trajectory, which will be discarded. According to our experience, it is set at 30. Threshold ξ in Algorithm3 represents that when the P of a detection box is higher than the ξ, it is considered to be reliable. ξ is set at 0.7 with our experience. The errors in tables 1-8 can be obtained by:

$$Error = \frac{Framework \quad result - Label \quad result}{Label \quad result} \tag{4}$$

We selected three different scenarios in the VCD dataset, and the experimental results were displayed in Tables 4-12, the videos screenshot corresponding to them are Figs. 7, 8,

TABLE 8. The framework result of the video from VCD named 20180328.

Framework	From A		From B		From C		Total
Route	A-B	A-C	B-A	B-C	C-A	C-B	
Car	164	539	295	22	411	4	1435
Truck	11	1	0	0	17	3	32
Bus	6	15	24	0	12	0	57
Total	181	555	319	22	440	7	1524

TABLE 9. The vehicle counting error of the video from VCD named 20180328.

Error(%)	From A		From B		From C		Total
Car	-9.9	4.1	9.6	15.8	0.5	0	2.4
Truck	-42.1	/	0	0	-19.1	-25.0	-27.3
Bus	-50.0	-25.0	50.0	0	0	0	-5.0
Total	-15.0	3.1	11.9	15.8	-0.5	-12.5	1.3

TABLE 10. The label result of the video from VCD named 20180202.

Label	From A	From B	Total
Route	A-B	B-A	
Car	0	2545	2545
Truck	0	24	24
Bus	0	204	204
Total	0	2773	2773

TABLE 11. The framework result of the video from VCD named 20180202.

Framework	From A	From B	Total
Route	A-B	B-A	
Car	0	2434	2434
Truck	0	13	13
Bus	0	189	189
Total	0	2636	2636

TABLE 12. The vehicle counting error of the video from VCD named 20180202.

Error(%)	From A	From B	Total
Route	A-B	B-A	
Car	0	-4.4	-4.4
Truck	0	-45.8	-45.8
Bus	0	-7.4	-7.4
Total	0	-4.9	-4.9

and 9. They are the video in the crossroad scene named 20180324, the video in the T-junction scene is 20180328, and the video of road scene named 20180202.

As shown in Tables 4, 5, and 6 with Fig. 7, the overall error detected at the cross was −3.1%. Generally, we found the larger the sample volume, the lower the absolute number of the error. For an example, at the road mouths A to D, the vehicle numbers were counted 522, 307, 569, and 176, respectively. The errors detected using the framework were −4.6%, −5.2%, 2.4%, and −12.5%, respectively. Thus the errors detected at each mouth were quite similar, indicating that the results obtained using our framework is highly reliable.

The overall error determined at the T-Junction was 1.3% (Tables 7, 8, and 9 with Fig. 8). The correlation between sample volumes and errors rate was similar compared to that detected from the cross, suggesting that the detection result using our framework will be remarkably accurate when the sample volume is large enough.

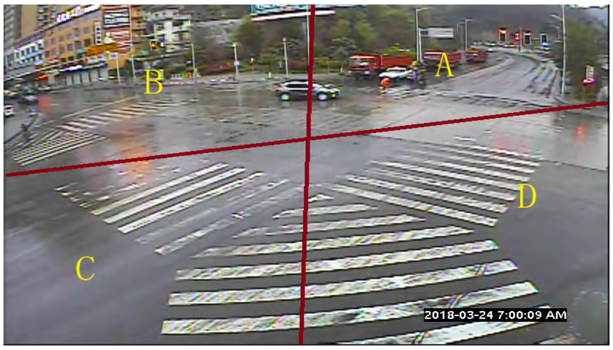

FIGURE 7. region encoding diagram of the video (20180324).

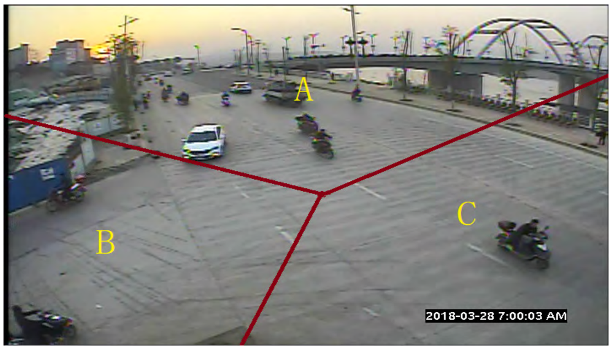

FIGURE 8. region encoding diagram of the video (20180328).

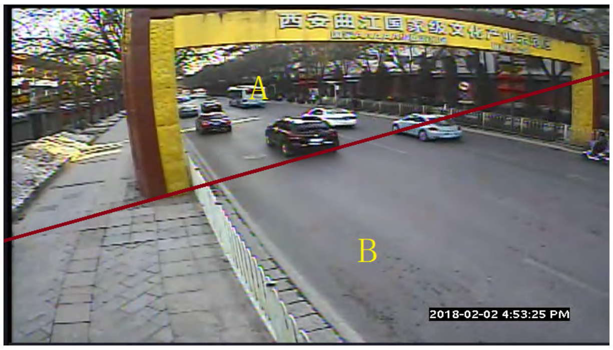

FIGURE 9. region encoding diagram of the video (20180202).

Tables 10, 11, and 12 with Fig. 9 demonstrated that the overall error was −4.9% detected at the straight road. In this scenario, the error of trucks was the highest among the three vehicle categories. It was likely caused by the less numbers of trucks detected during this time period compared to other vehicles. Only 24 trucks were labeled, whereas the numbers for the cars and buses were 2545 and 204, respectively.

From the perspective of total errors of the three road scenarios, the overall error at the T-junction was the lowest, whereas the straight road exhibited the highest error. The high error of the straight road was likely due to the video angle, which was constant for the vehicle counting in a fixed direction. In contrast, at the cross and the T-junction, the errors at the different directions could be either positive or negative, thus the total

TABLE 13. Video running time result.

Video name	Total number of vehicles	Video duration	Framework duration	Realtime rate
20180324	1574	60.0 min	78.4 min	1.3
20180328	1505	60.0 min	75.1 min	1.2
20180202	2773	60.0 min	85.6 min	1.4

TABLE 14. VCD running time result.

Mouth type	Crossroad	T-Junction	Straight road	Average
Video frame rate	29.0 fps	29.0 fps	29.0 fps	29.0 fps
Framework running rate	18.9 fps	20.4 fps	22.7 fps	20.7 fps

error was low after addition of the errors detected from these directions. Comparing between the results of the cross and the T-junction, the errors detected from the T-junction was generally lower because the complexity of the intersection is lower, which might result from fewer occlusion problems of the camera and occlusion would enhance detection errors and tracking errors. In the T-junction experiment, the error of the relevant direction of the mouth B was high, as it was the closest to the camera. The vehicle moved to a large proportion in the image, and the occlusion of the image was severe. Therefore, camera angle also plays a significant role in the accuracy of object detection.

The outcomes of the experiments in this study are generally in line with our expectations. Although the errors caused by occlusion still exists, the counting method is not significantly affected by it because the counting condition of the counting method is that the starting point and the end point fall in two different areas. If a trajectory breaks or is missing during the target movement, the error can be eliminated by the counting frame, thereby achieving a higher counting accuracy.

C. RUNNING TIME RESULT

As a video-based framework, the performance of both accuracy and speed are important. In this section, a speed experiment is achieved to evaluate the speed of the proposed method. First, Table 13 shows the running time results on the three videos in the section VII-B. At the same time, Table 14 indicates the running time results of all the videos in the VCD, and the average frame per second according to different mouth types. The real-time rate in Table 13 can be obtained by:

$$Realtime \quad rate = \frac{Framework \quad duration}{Video \quad duration}. \tag{5}$$

The smaller the value of the real-time rate is, illustrate the faster the framework is. When the value of the real-time rate is less than or equal to 1, the proposed framework can realize real-time processing.

As can be seen from Table 13, when the vehicle number are 1574, 1505, and 2773, the real-time rate are 1.3, 1.2, and 1.4, respectively. As the number of vehicles in the video increases, the running time of the proposed framework also increases. Because of a large number of vehicles will make the object tracking stage takes more time. Therefore, the number of vehicles in the video is positively correlated with the running time of the proposed framework. Table 14 shows that the framework running rate of different mouth types are 18.9 fps, 20.4 fps, and 22.7 fps, respectively, corresponding to crossroad, T-junctions, and straight road. It can be seen that the complexity of the video scene also affects the speed of the framework.

From the experimental results in this section, it can be seen that the speed of the framework is affected by the number of vehicles in the video and the complexity of the video scene. The running rate of the proposed framework is 20.7 fps on the VCD. However, the real-time counting of the traffic flow information of the video can also be determined by the video frame rate.

VIII. CONCLUSION

In this paper, a vehicle counting framework based on video in traffic scenes is proposed, which is tested in different traffic scenes. The results show that the accuracy of detection method using YoloV3 with our dataset is very high, reaching 87.6%, even if the traffic condition is quite complex. Moreover, the framework is capable of counting vehicles in different categories and moving route with an overall accuracy of more than 90%. The running rate of the proposed framework is 20.7 fps on the VCD. In the future study, we will also detect pedestrians and bikers. The final framework will provide more insights to improve the traffic flow at different settings such as intersections near hospitals and commercial centers.

REFERENCES

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[2] A. M. Pereira, "Traffic signal control for connected and non-connected vehicles," in *Proc. Smart City Symp. Prague (SCSP)*, Prague, Czech Republic, May 2018, pp. 1–6.

[3] Y. Liu, B. Tian, S. Chen, F. Zhu, and K. Wang, "A survey of vision-based vehicle detection and tracking techniques in its," in *Proc. IEEE Int. Conf. Veh. Electron. Saf.*, Jul. 2013, pp. 72–77.

[4] L. Juan and O. Gwon, "A comparison of SIFT, PCA-SIFT and SURF," *Image Process. Pattern Recognit.*, vol. 3, no. 4, pp. 143–152, 2009.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. CVPR*, Jun. 2005, pp. 886–893.

[6] T. Mita, T. Kaneko, and O. Hori, "Joint Haar-like features for face detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2005, pp. 1619–1626.

[7] P. Felzenszwalb, D. Mcallester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. CVPR*, vol. 8, Jun. 2008, pp. 1–8.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), IEEE Conf. Comput. Vis. Pattern Recognit. (CVF), IEEE Comput. Soc, 27th IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Columbus, OH, USA, Jun. 2014, pp. 580–587.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[10] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[12] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2015, pp. 21–37.

[13] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: https://arxiv.org/abs/1804.02767

[14] D. Meyer, J. Denzler, and H. Niemann, "Model based extraction of articulated objects in image sequences for gait analysis," in *Proc. Int. Conf. Image Process.*, Oct. 1997, pp. 78–81.

[15] J. Bouvrie, "Notes on convolutional neural networks," In Pract, 2006, pp. 47–60.

[16] C. Harris, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, vol. 3, 1988, pp. 147–151.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[18] Z. Cui, S. Xiao, J. Feng, and S. Yan, "Recurrently target-attending tracking," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1449–1458.

[19] J. Barandiaran, B. Murguia, and F. Boto, "Real-time people counting using multiple lines," *Proc 9th Int. Workshop Image Anal. Multimedia Interact. Services*, May 2008, pp. 159–162.

[20] G. Antonini and J. P. Thiran, "Counting pedestrians in video sequences using trajectory clustering," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 8, pp. 1008–1020, Aug. 2016.

[21] H. Song, X. Wang, C. Hua, W. Wang, Q. Guan, and Z. Zhang, "Vehicle trajectory clustering based on 3D information via a coarse-to-fine strategy," *Soft Comput.*, vol. 22, no. 5, pp. 1433–1444, Mar. 2018.

[22] C. Zeng and H. Ma, "Robust head-shoulder detection by PCA-based multilevel HOG-LBP detector for people counting," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 2069–2072.

[23] L. D. Pizzo, P. Foggia, A. Greco, G. Percannella, and M. Vento, "Counting people by RGB or depth overhead cameras," *Pattern Recognit. Lett.*, vol. 81, pp. 41–50, Oct. 2016.

[24] F. Y. Shih and X. Zhong, "Automated counting and tracking of vehicles," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 31, no. 12, 2017, Art. no. 1750038.

[25] J. Q. Ren and Y. Z. Chen, "Multiple objects parameter detection in urban mixed traffic scene," *J. Transp. Inf. Saf.*, vol. 27, pp. 47–54, Apr. 2009.

[26] A. Abdagic, O. Tanovic, A. Aksamovic, and S. Huseinbegovic, "Counting traffic using optical flow algorithm on video footage of a complex crossroad," in *Proc. ELMAR*, Sep. 2010, pp. 41–45.

[27] L. Unzueta, M. Nieto, A. Cortes, J. Barandiaran, O. Otaegui, and P. Sanchez, "Adaptive multicue background subtraction for robust vehicle counting and classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 527–540, Jun. 2012.

[28] H. Yang and S. Qu, "Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition," *IET Intell. Transp. Syst.*, vol. 12, no. 1, pp. 75–85, Nov. 2017.

[29] S. Bouaich, M. A. Mahraz, J. Riffi, and H. Tairi, "Vehicle counting system in real-time," in *Proc. Int. Conf. Intell. Syst. Comput. Vis. (ISCV)*, Fez, Morocco, Apr. 2018, pp. 1–4.

[30] M. Sun, Y. Wang, T. Li, J. Lv, and J. Wu, "Vehicle counting in crowded scenes with multi-channel and multi-task convolutional neural networks," *J. Vis. Commun. Image Represent.*, vol. 49, pp. 412–419, Nov. 2017.

[31] G. Li, P. Ren, X. Lyu, and H. Zhang, "Real-time top-view people counting based on a Kinect and NVIDIA jetson TK1 integrated platform," in *Proc. IEEE 16th Int. Conf. Data Mining Workshops (ICDMW)*, Dec. 2016, pp. 468–473.

[32] G. Liu, Z. Yin, Y. Jia, and Y. Xie, "Passenger flow estimation based on convolutional neural network in public transportation system," *Knowl. Based Syst.*, vol. 123, pp. 102–115, May 2017.

[33] Y. P. Kocak and S. Sevgen, "Detecting and counting people using real-time directional algorithms implemented by compute unified device architecture," *Neurocomputing*, vol. 248, pp. 105–111, Jul. 2017.

ZHE DAI received the bachelor's degree from Chang'an University, Xi'an, in 2015, where he is currently pursuing the Ph.D. degree in traffic information engineering and control.

HUANSHENG SONG received the Ph.D. degree in information and communication engineering from Xi'an Jiaotong University, in 1996. In 2004, he was with the Information Engineering Institute, Chang'an University, where he became a Professor, in 2006. In 2012, he was the Dean of the Information Engineering Institute.

XUAN WANG received the B.S. degrees in network engineering, and the Ph.D. degree in traffic information engineering and control from Chang'an University, in 2013 and 2018, respectively. Since 2019, she has been with the School of Computer and Control Engineering, Yantai University. Her current research interests include image processing and recognition, pattern recognition, and intelligent transportation systems.

YONG FANG received the Ph.D. degree from Xidian University, in 2005. From 2006 to 2007, he was a Lecturer and a Postdoctoral Fellow with the College of Telecommunications Engineering, Northwestern Polytechnical University. From 2007 to 2009, he was a Research Professor with the Department of Electronics and Computer Engineering, Hanyang University, South Korea. From 2009 to 2016, he was a Professor with the College of Information Engineering, Northwest A&F University. In 2017, he was a Professor with the College of Information Engineering, Chang'an University.

XU YUN received the bachelor's degree from Chang'an University, in 2018, where he is currently pursuing the master's degree. His research interests include pattern recognition, image processing, deep learning, and computer vision.

ZHAOYANG ZHANG received the B.S. and Ph.D. degrees in applied mathematics from Northwestern Polytechnical University (NWPU), Xi'an, China, in 2007 and 2013, respectively. He visited the Department of Computer Science Centre, University of Alberta, Edmonton, Canada, from 2011 to 2012. He has been a Teaching Fellow with Chang'an University, Xi'an, China, since 2014. His current research interests include linear and nonlinear feature extraction methods, pattern recognition, and remote sensing image processing.

HUAIYU LI received the master's degree from Xidian University, in 2010. He is currently pursuing the Ph.D. degree with the College of Information Engineering, Chang'an University. His current research interests include computer vision and machine learning.