Real-Time Traffic Flow Parameter Estimation From UAV Video Based on Ensemble Classifier and Optical Flow

Ruimin Ke, *Student Member, IEEE*, Zhibin Li, Jinjun Tang, Zewen Pan, and Yinhai Wang

***Abstract*— Recently, the availability of unmanned aerial vehicle (UAV) opens up new opportunities for smart transportation applications, such as automatic traffic data collection. In such a trend, detecting vehicles and extracting traffic parameters from UAV video in a fast and accurate manner is becoming crucial in many prospective applications. However, from the methodological perspective, several limitations have to be addressed before the actual implementation of UAV. This paper proposes a new and complete analysis framework for traffic flow parameter estimation from UAV video. This framework addresses the well-concerned issues on UAV's irregular ego-motion, low estimation accuracy in dense traffic situation, and high computational complexity by designing and integrating four stages. In the first two stages an ensemble classifier (Haar cascade + convolutional neural network) is developed for vehicle detection, and in the last two stages a robust traffic flow parameter estimation method is developed based on optical flow and traffic flow theory. The proposed ensemble classifier is demonstrated to outperform the state-of-the-art vehicle detectors that designed for UAV-based vehicle detection. Traffic flow parameter estimations in both free flow and congested traffic conditions are evaluated, and the results turn out to be very encouraging. The dataset with 20,000 image samples used in this study is publicly accessible for benchmarking at http://www.uwstarlab.org/research.html.**

***Index Terms*— Convolutional neural network, ensemble classifier, Haar cascade, optical flow, UAV video, traffic flow parameter.**

I. Introduction

THE use of unmanned aerial vehicle (UAV) in traffic monitoring applications is becoming more and more popular. Compared to traditional monitoring devices, UAV is considered more cost-effective [1]–[3]. Most traditional traffic monitoring devices capture traffic conditions at fixed and discrete locations, hence it requires many units to monitor a single road segment [4]–[7]. In contrast, a few UAV(s) can cover a continuous stretch of roadway or even a traffic network. UAV's view point is another advantage for traffic monitoring: by achieving a top-view perspective, occlusions between road users in conventional surveillance videos are not likely to appear in UAV videos. Also, in surveillance video frames, pixels at different locations very likely correspond to different real-world sizes due to the camera tilt angle, but the corresponding real-world sizes of pixels in an UAV video frame are very close to one another, thus, camera calibration and estimation of real-world distances from an UAV video are easier. As a flexible platform with high mobility, UAV can also provide rapid reconnaissance and assessment of incident sites where few stationary sensors are placed for emergency response [1], [2]. Compared to manned aircrafts, UAVs have much lower operating and purchase cost. In addition, they fly closer to the ground, and thus have better robustness to adverse weather [1].

In addition to several practical concerns such as short battery life and privacy issue, the biggest technical challenge in automatic UAV-based traffic monitoring is the ego-motion issue, which can be generated either intentionally (e.g., by pilot) or unintentionally (e.g., by wind). The video background movement caused by UAV ego-motion makes the traditional vehicle detection and traffic flow estimation methods designed for stationary surveillance videos not work well. To simplify the examination of UAV applications in traffic monitoring, UAV videos with little ego-motion were used for some preliminary studies. For example, Zhao *et al.* [35] applied convolutional neural network (CNN) and speeded up robust features (SURF) to estimate traffic flow parameters from UAV videos with no ego-motion. Their method achieved high accuracy in detection and tracking but would not work for UAV videos with moving background because the motion-vector estimated by SURF tracking was actually the sum of traffic motion-vector and background motion-vector. Yamazki *et al.* [25] estimated vehicle speed based on two consecutive frames of UAV video. Their method made use of the aforementioned view point benefit, but the speed detection in their paper assumed a fixed video background.

With the good foundation laid by these studies, more efforts have been put to address the ego-motion issue. While the objectives of monitoring tasks could be different, existing methods on addressing ego-motion issue can be divided into two categories: image registration [18], [21], [24],

Manuscript received December 2, 2016; revised July 13, 2017, November 16, 2017, and January 12, 2018; accepted January 22, 2018. Date of publication March 7, 2018; date of current version December 21, 2018. This work was supported in part by NSF China under Grant 51329801 and in part by the Shenzhen Science and Technology Planning Project under Grant GJHZ20150316154158400. The Associate Editor for this paper was S. S. Nedevschi. *(Corresponding author: Yinhai Wang.)*

R. Ke and Y. Wang are with the Department of Civil and Environmental Engineering, University of Washington, Seattle WA 98195 USA (e-mail: ker27@uw.edu; yinhai@uw.edu).

Z. Li is with the School of Transportation, Southeast University, Nanjing 210096, China (e-mail: lizhibin@seu.edu.cn).

J. Tang is with the School of Traffic & Transportation Engineering, Central South University, Changsha 410075, China (e-mail: jinjuntang@163.com).

Z. Pan is with the College of Resources and Civil Engineering, Northeastern University, Shenyang, China. (e-mail: panzewen_neu@163.com).

Digital Object Identifier 10.1109/TITS.2018.2797697

[25], [27], [36], and optical flow-based ego-motion estimation [18]–[21], [28], [29]. Image registration aims to turn the moving background into fixed background, thus makes it possible to apply traditional methods such as background subtraction. Image registration is probably the most intuitive method for addressing ego-motion issue and has been adopted by many previous studies. In these studies, a common assumption is that feature points mostly come from the video background rather than the vehicles (foreground). However, this assumption may be invalid in dense traffic situations, in which many feature points belong to the video background, hence the background motion could be inaccurately estimated.

Optical flow is a powerful method for video analysis due to its ability to extract the motion pattern. This method has been examined in UAV video processing. In previous studies, optical flow is normally combined with image registration or unsupervised learning to estimate the UAV ego-motion. It acts as an efficient feature extraction and feature matching tool to speed up the image registration [18], [21]. When optical flow collaborates with clustering algorithms, video background and foreground can be separated [28], [29]. But the ways previous studies employ optical flow are still not suitable for dense traffic scenarios. Besides the aforementioned concern in image registration-based ego-motion estimation, optical flow-based clustering would classify some interest points on the vehicles with low speed as background points, which may cause errors as well. These considerations motivate us to come up with the idea of integrating optical flow with supervised learning-based vehicle detection. In this way, vehicle pattern, which is not sensitive to traffic density, will be used for the separation of video foreground and background, instead of interest points or motion information.

In this paper, the authors propose a framework for traffic flow parameter estimation from UAV videos that incorporates supervised learning-based vehicle detection methods and optical flow. This framework aims at filling the gaps in traffic flow parameter estimation from UAV videos with ego-motion. It is designed to work for both free flow and dense traffic. The framework is composed of four stages: The first two stages are designed for vehicle detection and the last two traffic flow parameter (speed, density, volume) estimation. Specifically, the first two stages in this framework combine Haar cascade and CNN as an ensemble classifier. Although Haar cascade classifier and CNN have been examined in UAV-based vehicle detection separately [16], [34], [35], the proper combination of them in our study improves both the efficiency and accuracy of vehicle detection. Haar cascade acts as the region proposal method to largely reduce the number of region of interest (ROI) efficiently and CNN then determine the final detection results with its high accuracy. Stage three and four are designed as a general process for traffic flow parameter estimation in UAV videos. In other words, this process would still work even if our ensemble classifier were replaced by other vehicle classifiers.

II. Literature Review

In the realm of transportation engineering, an increasing amount of research was about utilizing UAV videos as a new type of data source. Most recent work has focused on one of the three categories: road detection [3], [8]–[10], vehicle detection and tracking [11]–[22], [34], or traffic parameter estimation [23]–[29], [35], [36]. UAV-based road detection is about localizing the region where traffic is likely to appear thus to improve the efficiency of automatic traffic monitoring. Also, road detection is crucial for UAV navigation systems, especially those based on vision information. For instance, Zhou *et al.* [3] designed an efficient algorithm for road detection and efficient tracking. While road detection has been done before in UAV-based monitoring, this was the first study that aimed at speeding up the road localization with a tracking method developed. Their method was tested on multiple challenging scenarios and has been gaining increasing popularity. Another representative study was conducted by Kim [9]. Their proposed algorithm for road detection was relatively simple but practical: their algorithm first learned the road structure from a single video frame and then identified the road in the remaining video frames.

The second category focuses on designing methodologies for vehicle detection and/or tracking in UAV videos [11]–[22], [34]. Vehicle detection normally serves as the initialization process in the methodological framework that determines the overall tracking or traffic flow extraction performance. Vehicle tracking, especially multi-vehicle tracking techniques enable deep analysis on traffic flow. Hence, developing fast and accurate vehicle detection/tracking methods is very important to UAV-based traffic monitoring and management. For example, Cao *et al.* [18] proposed a robust vehicle detection and tracking system by multi-motion layer analysis, which demonstrated great potential to be widely used in vehicle detection in UAV videos. This is one of the most representative studies in vehicle detection/tracking using UAV video data. However, in all of the motion analysis-based work, unless sufficient feature points could be extracted from the background in the video, the detection would not be properly done. Supervised learning methods have been applied more and more recently. With them, vehicles are identified based on their patterns, therefore the complicated video background motion issue can be skipped in vehicle detection tasks. Popular learning methods such as CNN or SVM have already been demonstrated to work well for UAV-based vehicle detection [13], [35].

The third category is traffic flow parameter estimation. Multiple studies have been conducted to extract a variety of traffic flow parameters such as speed, density, travel time, delay, and annual average daily traffic (AADT) from UAV videos [23]–[29], [35], [36]. Normally, different vehicle detection methods apply to the parameter estimation process in different traffic scenes or conditions. For instance, McCord *et al.* [23] developed a modeling framework to estimate AADT using satellite imagery and aerial photos. Although the vehicle detection process was done manually, their work was still one of the most important studies in this field since they were among the first groups of people who proposed the idea to extract traffic flow parameters from aerial photos or videos. Shastry and Schowengerdt [27] exploited both image registration and motion information to

successfully estimate basic traffic flow parameters. To our knowledge, their method was the first to incorporate KLT tracker in this field. Ke *et al.* [29] proposed a novel framework making use of KLT tracker, k-means clustering, connected graph and the basic equation of traffic flow theory to estimate bidirectional traffic speed, density and volume in real-time. However, the vehicle detection in this study was based on a "similar motion" criterion, which would largely decrease the estimation accuracy in heavy congestion situations.

To the authors' best knowledge, until now, no previous studies have achieved real-time traffic parameter estimation from UAV videos for both free-flow and congested traffic conditions. This study targets this issue by proposing a new framework, which is presented in the following sections. The contributions of the paper can be summarized as follows (all the contributions are under the context of UAV video processing):

1) a new real-time framework for traffic flow parameter estimation is proposed;

2) supervised learning is incorporated for the first time for traffic flow parameter estimation with background motion in the UAV video;

3) this is the first framework that work for both free flow and dense traffic with respect to traffic flow parameter estimation in UAV video with background motion;

4) a new method that addresses UAV height changes in real-time is developed;

5) stage three and four of the framework is a new general process designed for traffic flow parameter estimation;

6) the ensemble classifier (Haar cascade + CNN) is examined for the first time for vehicle detection (in UAV video);

7) a publicly available dataset containing 20,000 training samples for UAV-based vehicle detection is published as a benchmark.

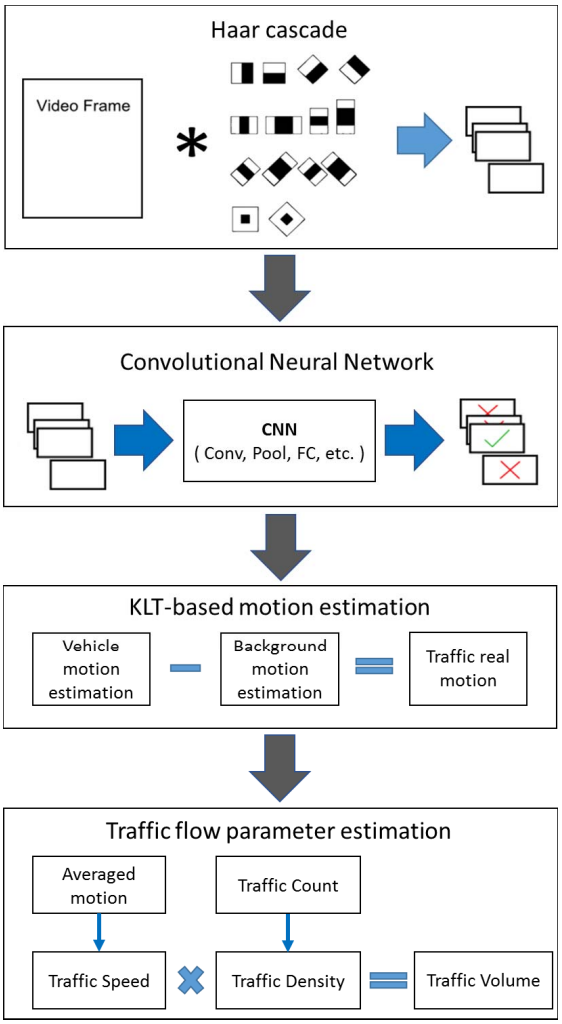


Fig. 1. Summary of the four-stage processing framework.

III. Methodology

A. Overview

The proposed framework contains four main stages (see Fig. 1). The first two stages deal with vehicle detection and the last two about traffic flow parameter estimation. In the first stage, the Haar cascade classifier trained using randomly generated Haar-like features is applied as the region proposal method in order to determine the ROI. Haar cascade is very efficient thus can largely reduce the searching space for the final strong classifier. CNN is a powerful neural network model that has been proved effective in many detection tasks. In the second stage, CNN is tested and selected as the final vehicle classifier. In the third stage, the traffic real traffic motion is estimated by subtracting background motion from vehicle motion. Both the background motion and vehicle motion are estimated using KLT optical flow tracker based on the detection results. In the fourth stage, the traffic counts and estimated vehicle motion in pixels per frame are converted to traffic density and speed with the conversion rate (converting pixel length to physical length) computed using reference markings. Then, traffic volume can be calculated using the basic traffic flow equation. Implementation details and further illustrations will be discussed in the next several sub-sections.

B. Data Description

Several UAV video clips taken from UAVs flying over different roadway segments were specifically used for training samples collection. In total 20,000 samples have been collected. Positive samples are mostly the top-view of cars. Only a small portion of positive samples include other types of vehicles such as buses and trucks due to the data availability, more trucks and buses will be collected in the future with more UAV videos taken. Negative samples are randomly cropped background images. Both the positive and negative samples were manually cropped from the training videos in the original size of 60×40 (width $\times$ height). In the Haar cascades training, the sizes of the samples were kept 60×40. However, in CNN training, the samples were scaled to the size of 20×13 (before any pooling action) in order to reduce the dimensionality of the input feature vector from 2400 to 260, thus to speed up the training processes. The dataset was split into 18,000 samples for training and 2,000 for testing. It was composed of 40% positive samples and 60% negative samples.

This split ratio was selected based on the consideration that background patterns had more variations than vehicle patterns.

It is worth noting that the images for training the vehicle classifiers in this study were collected by the authors and it is available at http://www.uwstarlab.org/research.html. This dataset is one of the first publicly available datasets for UAV-based vehicle detection. We have conducted some experiments and conclude that even the best pre-trained object classifiers so far such as YOLO9000 [41] performs poorly in vehicle detection in UAV videos due to the lack of vehicle samples from top-view. It is believed this dataset will benefit the community.

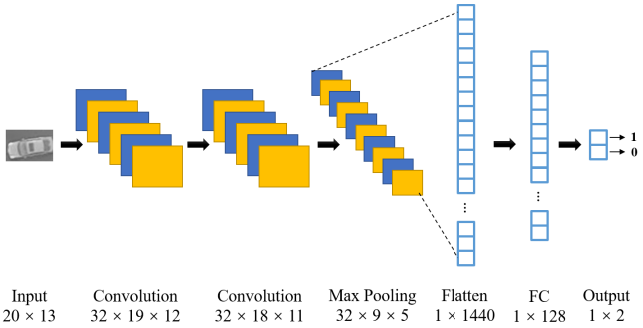


Fig. 2. The proposed CNN model for vehicle detection in UAV video.

C. Stage 1: Haar Cascade Classifier for Region Proposal

Originally, Haar cascade classifier was employed as a statistical approach to handle the large variety of human faces [31]. Haar-like features and AdaBoost learning algorithm [33] are the two major components of the approach.

In this study, the Haar cascade classifier was trained using OpenCV 2.4.12 library [30]. Besides setting the training image size to 60×40 as aforementioned, several other key parameters need to be set in the Haar cascade training process. First, the number of training samples for each stage needs properly set. The basic idea here is not to train too few stages which would make the region proposal effect not significant in terms of reducing candidate windows, but too many stages would very likely filter out some true positives. Based on this consideration and the total number of samples, each stage took 2,500 samples for training (1,000 positive samples and 1,500 negative samples), thus resulted in 7 stages.

Another two parameters needed for the training are min hit rate and max false-positive rate per stage. Since Haar cascade is the region proposal method in this study, min hit rate should be set close to 1 to make sure we recall all vehicles. The max false-positive rate per stage should be set not larger than 0.5. But if it is too small, say close to 0, the training time and the chance of overfitting would both sharply increase. As long as our recall is high, it is totally acceptable that each stage of our Haar cascade is just slightly better than a random classifier. Thus, in our training, these two parameters are set to 0.999 and 0.5. Non-maximum suppression is adopted as the last part of stage 1 to further reduce the candidate windows CNN needs to examine.

D. Stage 2: Convolutional Neural Network for Vehicle Detection

With the regions of interest proposed by Haar cascade in the first stage, a convolutional neural network, or CNN, is designed as the final classifier for vehicle detection. In this way of combination, the high efficiency of Haar cascade and the high accuracy of CNN are well utilized, thus can enable real-time vehicle detection with high detection rate. This is the first time that the ensemble classifier (Haar cascade + CNN) is examined in UAV-based vehicle detection. The first successful applications of CNN were developed by Yann LeCun for document recognition [37]. CNN has a typical layer called convolutional layer, which is the core building block of it. Compared to fully-connected (FC) multi-layer perceptron (MLP) neural network, CNN has fewer parameters because of convolutional layer's local connectivity, thus the chance of overfitting can be reduced. Moreover, convolutional layer captures more representative features, particularly for image inputs. Another type of layer in CNN architecture is called pooling layer, which performs downsampling operation along the spatial dimension. Convolutional layer and pooling layer enable more effective feature selection and more efficient learning of features at different scales. FC layer and activation function are still important components of CNN architecture with respect to image classification problems.

In our study, CNN was developed using Keras in python and trained on an Nvidia GTX 1080 GPU. With a trial and error process, the architecture of our CNN was chosen to contain two convolutional layers, one pooling layer and one hidden FC layer (see Fig. 2). The two convolutional layers have a same dimension of $32 \times 2 \times 2$ with sigmoid activation function; then the pooling layer is added to downsample the second convolutional layer's outputs by a scale factor of 2; and the FC layer with 128 nodes is added between the pooling layer and the final outputs. Compared to other popular CNNs such as AlexNet [42] or VGG [43] with deeper structures and more output nodes, the proposed CNN structure is light-weight with much fewer layers and parameters. This is motivated by our requirement for real-time operation and a smaller number of categories (i.e., vehicle and background). It is found that two convolutional layers can already satisfy the accuracy requirement. It is worth noting that there is no pooling layer in between the two convolutional layers. This is because the training and testing losses turn out to be higher while the overall detection speed is not significantly improved if adding the pooling layer. Based on our tests and analyses, the increased losses are mainly caused by the small image size after pooling. Since the dimension of our image samples is 20×13, if they are downsampled by the pooling layer, the features extracted by the second convolutional layer would not be as representative.

The training of CNN was done on 18,000 samples and testing on 2,000 samples. RMSprop (Root Mean Square Propagation) [39] was selected as the optimizer because of its better performance than others like traditional SGD (Stochastic Gradient Descent) [40] in similar cases based on experience and tests. The batch size for optimization was set to 30.

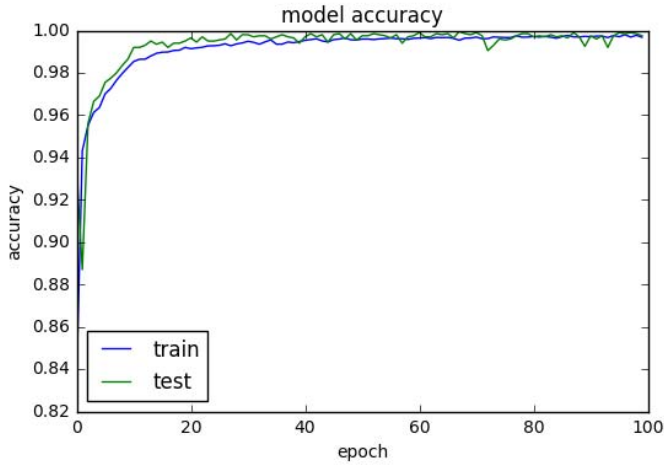


Fig. 3. CNN Model accuracy curves (train and test) during the 100-epoch training process.

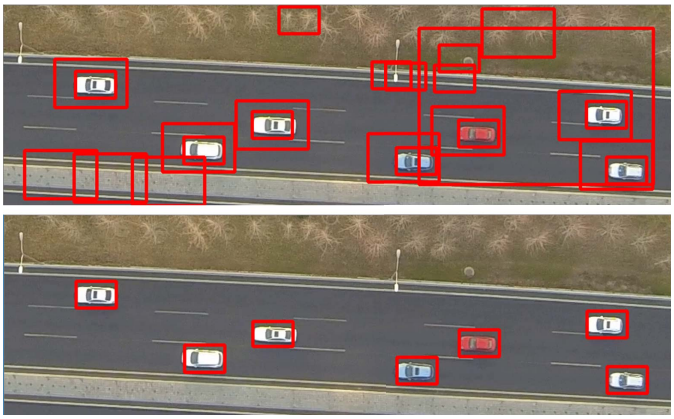


Fig. 4. Vehicle detection results after stage 1 (top) and stage 2 (bottom). Haar cascade classifier acts as the efficient region proposal method and then CNN does the final detection by examining far less candidate windows than sliding the whole frame.

Our CNN vehicle classifier reached 99.55% classification accuracy on the test data in 100 epochs of training, which was very encouraging. The model accuracy curves during the training process are shown in Fig. 3.

As aforementioned, the top figure of Fig. 4 shows the candidate windows proposed by Haar cascade classifier. It can be seen there are still some false-positives, but compared to letting the strong classifier (i.e., CNN in our framework) slide the whole frame in different scales, the number of candidate windows have been largely reduced by Haar cascade. CNN is then applied to check all the candidate windows and gives out the final vehicle detection results. One example frame of the vehicle detection results is shown in the bottom figure of Fig. 4.

E. Stage 3: KLT-Based Motion Estimation

With the detection results obtained, stage 3 and 4 define a general process for traffic motion estimation and traffic flow parameter estimation. To maximize the UAV's view point benefit, we assume an orthographic camera projection which

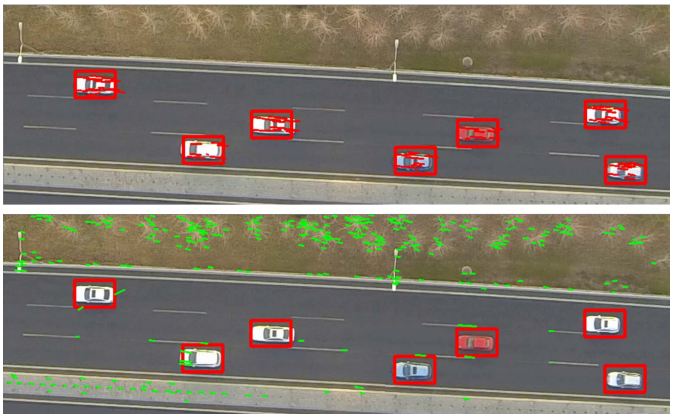


Fig. 5. The proposed method for traffic motion estimation using KLT tracker. Based on the detection results, vehicle motion (red dots at the top figure) and background motion (green dots at the bottom figure) can be estimated. The motion-vector representing the traffic real motion in pixels per frame is computed by subtracting average background motion from average vehicle motion.

is an approximation with a downward facing UAV camera. This general process would work with any supervised learning-based vehicle detector to complete traffic flow parameter estimation from UAV videos. Motion estimation is often based on object tracking and a known video frame rate. However, in UAV videos with background motion, many methods such as Kalman filter and particle filter cannot achieve accurate traffic motion estimation even if they may achieve multiple-vehicle tracking. This is due to their inability to estimate ego-motion (background motion). KLT method [32] is a tracking method based on interest point, thereby it has the ability to estimate background motion in light traffic conditions [18]. But with relatively denser traffic, directly applying KLT tracker would lead to large errors in motion estimation. Our efficient Haar + CNN vehicle detection process is particularly designed to addresses this issue.

The vehicle detection results split a video frame into two types of regions: traffic (inside the detection windows) and background (outside the detection windows). Hence, KLT can be applied to estimate vehicle motion and background motion after CNN detection. In Fig. 5, the top image shows the motion-vectors extracted inside detection windows and the bottom image outside detection windows. The average of all the motion-vectors in the same category (inside or outside detection windows) represents traffic motion (with ego-motion added) and background motion, respectively. Suppose $\overrightarrow{mt}_i$ and $\overrightarrow{mb}_j$ denote the i-th motion-vector extracted for traffic and the j-th motion-vector for background, respectively, the true traffic motion $\overrightarrow{m}$ in pixels per frame is calculated as follows in Eq. (1):

$$\overrightarrow{m} = \frac{\sum_{i=1}^{Nt} \overrightarrow{mt}_i}{Nt} - \frac{\sum_{j=1}^{Nb} \overrightarrow{mb}_j}{Nb} \tag{1}$$

where Nt is the total number of motion-vectors extracted for traffic and Nb is for background.

It is worth noting that the way we calculate $\overrightarrow{m}$ in Eq. (1) using the mean values of $\overrightarrow{mt}_i$ and $\overrightarrow{mb}_j$ may not be very

accurate in the cases where there are quite a few outliers in the process of motion-vector estimation. These cases could be caused by low video quality, low lighting conditions, etc. In such cases, we suggest using median values of $\overrightarrow{mt}_i$ and $\overrightarrow{mb}_j$ to calculate the true traffic motion $\overrightarrow{m}$ instead of mean values due to median's better robustness to noise.

F. Stage 4: Traffic Flow Parameter Estimation

In transportation engineering, speed, density and volume are the most important three parameters to describe traffic flow, and their relationship is given by Eq. (2):

$$volume = speed \times density \times NOL, \tag{2}$$

where *NOL* denotes number of lanes. With the vehicles detected in stage 2 and traffic motion estimated in stage 3, traffic density and speed can then be calculated with reference markings. Density is defined as vehicle counts per lane per unit freeway length (mile, kilometer, etc.). Speed will be converted from pixels per frame to miles/kilometers per hour. Reference markings, such as standard school buses and lane markings, are able to avoid complicated camera calibrations and often sufficient for computing real-world sizes in UAV videos [27]–[29].

At a starting frame, we assume a known object to have a real-world size l_1 and pixel size l_2, thus the conversion rate is $r = l_1/l_2$, and the road segment length L pixels. The initial pixel lengths of l_2 and L are measured offline. Frame rate *fr* is assumed to be constant during monitoring. If real-time height information is recorded in the data, it would be helpful to determine if the conversion rate needs to be changed. However, most UAV video datasets do not contain real-time height information, so we proposed a method to estimate whether the height has significant changes. Normally, UAV tends not to change its flight height during a short time, thus it is unnecessary to check possible height changes in every pair of consecutive frames. The pixel to real-world size conversion rate in the first frame is denoted r_1, and r_n for the n-th frame f_n. Then, we compare the mean window width of detected vehicles in frame f_n and f_{n-1} to see if there is a significant difference using t-test with 0.05 as the threshold for statistical significance. If no significant changes detected, we will continue using the rate last updated for traffic flow calculation; if yes, r_n will be updated as in Eq. (3):

$$r_n = r_{n-1} \times \frac{w_n}{w_{n-1}} \tag{3}$$

where w_n is the mean width of detection windows in f_n, and w_{n-1} in f_{n-1}. With these definitions and calculations above, traffic speed and density are calculated using Eq. (4)-(5):

$$speed = \overrightarrow{m} \times fr \times r \tag{4}$$

$$density = \frac{N}{L \times r \times NOL} \tag{5}$$

where N is the number of vehicles detected in the current frame and r is the last updated real-world to pixel conversion rate. Another basic traffic flow parameter, i.e. volume, is calculated using Eq. (2).

IV. Experimental Results

A. Vehicle Detector Evaluation

To analyze the performance of the proposed framework, we first tested our vehicle detector's performance and compare them with the state-of-the-art vehicle detectors developed for UAV video data [13], [16], [34], [35], [38]. Specifically, Haar cascade, CNN, MLP, HOG + SVM, Haar cascade + MLP, and Haar cascade + CNN (proposed ensemble classifier) were tested and compared using our collected vehicle samples.

The Haar cascade classifier was trained using OpenCV 2.4.12 build-in functions, and the CNN classifier was trained using a python program developed by our team. This program made use of the Keras deep learning library and ran on the Theano backend. Both Haar cascade and CNN classifiers were stored as xml files. Another standalone python program was then developed to implement the entire detection and estimation process. This program first loaded the two xml files and read video frames one by one as inputs, then it ran the proposed pipeline and outputted traffic flow density, speed, and volume. For MLP and SVM training as well as performance evaluation we adopted Keras and Sklearn libraries. CNN and MLP were trained on Nvidia GTX 1080 GPU while other training, detection, and data management programs were implemented on an Intel Core i7-6700 CPU.

All the detectors were trained using the same training samples (i.e., the 18,000 vehicle samples cropped from UAV videos) and tested on the remaining 2,000 samples. Besides accuracy evaluation, efficiency of these detectors were tested on an UAV video clip with 1000×300 resolution. Though video resolution may impact processing speed of a detector, the result here presents the relative order with respect to detector efficiency, which is not likely to change.

Precision, which is the fraction of relevant instances among the retrieved instances, and recall, which is the fraction of relevant instances that have been retrieved over total relevant instances, are commonly used in detectors performance evaluation. They are defined as in the following equations,

$$precision = \frac{TP}{TP + FP} \tag{6}$$

$$recall = \frac{TP}{TP + FN} \tag{7}$$

where TP is short for true-positive, FP for false-positive, and FN for false-negative. Precision and recall both reach their best value at 1 and worst at 0.

In the 2,000 test samples, 814 of them are labeled as positive samples (vehicles) and 1,186 negative samples (backgrounds). Detailed detection performance evaluation results are presented in Table I. Previous studies that used Haar cascade for UAV-based vehicle detection normally dealt with relatively simpler traffic scenes, and had fewer samples for training. For specific purposes, some study even just included one individual vehicle in the training samples [34]. While Haar cascade performed well in previous studies, with more variety in vehicle color, vehicle type, etc. appearing in our dataset, Haar cascade generated many fps as shown in Table I.

TABLE I
DETECTOR PERFORMANCE EVALUATION AND COMPARISON RESULTS

	Haar Cascade	CNN	MLP	Haar + MLP	HOG + SVM	**Ensemble Classifier**
Num. False-Positive	386	4	128	120	29	**4**
Num. False-Negative	35	1	117	122	179	**35**
Precision	0.669	0.995	0.845	0.852	0.956	**0.995**
Recall	0.957	0.999	0.856	0.850	0.781	**0.957**
Processing Speed (fps)	81.00	0.29	0.44	69.00	3.85	**67.00**

Fig. 6. Sample frames in video #1 (left, free flow) and video #2 (right, dense traffic) showing the detection and motion estimation results. Continual ego-motion including cruising, rotation and vibration exist in either video clip.

However, Haar cascade retained a good recall (0.957) and very fast processing speed (81 fps). Standalone CNN was the best detector in terms of accuracy, which generated just 4 FPs and 1 FNs. But it is very slow in computing with 0.29 fps processing speed. MLP achieved good precision and recall rates, i.e., 0.845 for precision and 0.856 for recall. However, they were not comparable to CNN or the ensemble classifier in term of either precision or recall, let alone its slow processing speed. Haar cascade + MLP method [38] had a similar processing logic as the proposed ensemble classifier, thus the processing speed was fast. But the precision and recall values were at the same level with standalone MLP. HOG + SVM is another popular detector that has been applied in different tasks, it was examined by Cao *et al.* [13] in UAV-based vehicle detection. They achieved a very high precision value with much less FPs generated than Haar cascade or MLP. Its processing speed was also faster than CNN or MLP. But its FN rate was high thus led to a recall rate lower than 0.80. Our ensemble classifier achieved 0.995 precision, 0.957 recall, and 67 fps processing speed. The excellent performance of our detector was beneficial from the combination of Haar cascade and CNN. Haar cascade generated many fps but they were eliminated by CNN at stage 2, thus resulting in very high precision rate. Then, as the recall rates of both Haar cascade and CNN stayed high, the combination of them still had a high recall. The processing speed of the ensemble classifier was very close to that of Haar cascade or Haar + MLP, and was much faster than other detectors. As aforementioned, this was because the final strong classifier (i.e., CNN) examined much less candidate windows than sliding the whole frame.

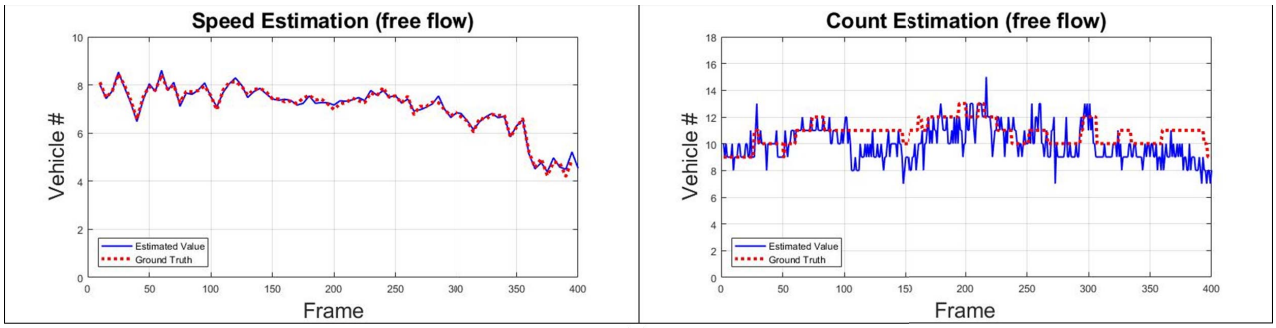
B. Traffic Flow Parameter Estimation Results

In the experiment, in total about thirty minutes' video clips were tested. Specifically, two representative 400-frame video footages were manually examined in detail frame by frame as examples to evaluate the effectiveness of the framework. The two video clips were not used for any training samples collection. Video #1 was taken by a UAV moving over a freeway segment, monitoring a three-lane freeway with smooth traffic flow. Video #2 was another video clip taken over an urban arterial where the traffic was dense. Continual background motion that caused by ego-motion including cruising, rotation and vibration existed in the test videos.

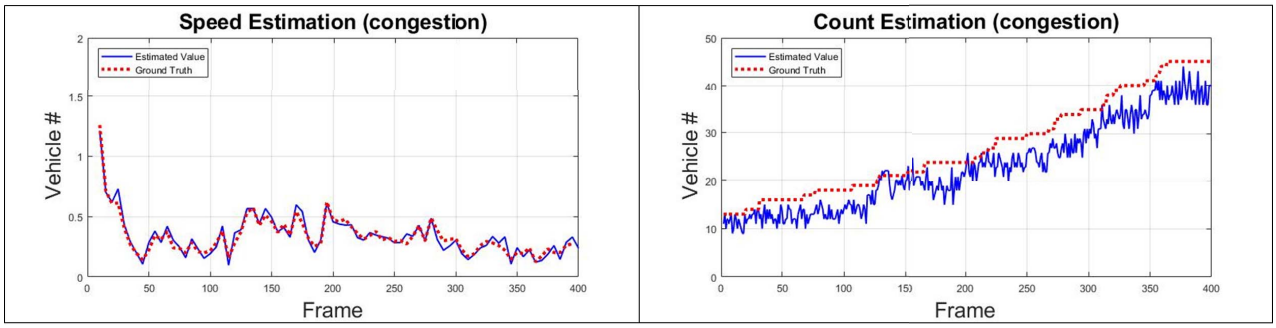
Fig. 6 shows some randomly selected sample frames in the two video clips with the detection windows and

TABLE II
SUMMARY OF ESTIMATED TRAFFIC FLOW PARAMETERS AND PERFORMANCE EVALUATION RESULTS

	Test Video #1	Test Video #2
Total Frames	400	400
Dimension of ROI	1000 × 300	1220 × 350
Traffic Condition	Free Flow	Congested
Estimated Speed (mph)	31.5	1.7
Estimated Count (per frame)	9.9	23.0
Estimated Density (pc/mi/lane)	31.9	45.5
Estimated Volume (pc/h)	3010.6	309.4
Speed Estimation Accuracy (%)	97.0	92.4
Count Estimation Accuracy (%)	95.8	85.1
Processing Speed (fps)	29.7	25.4



(a)



(b)

Fig. 7. Plots presenting the accuracy of traffic flow speed estimation and vehicle count estimation. Subfigure (a) presents the speed estimation results and count estimation results in video #1; subfigure (b) shows the speed and count estimation results in video #2. Ground truth and estimated value overlay in each of the four plots, where the dotted red curves represent the ground truths and solid blue curves are the estimated values.

motion-vectors marked. The three frames on the left are from video #1 and right video #2. In practice, reporting instantaneous traffic flow parameters frame by frame is not meaningful, therefore averaged speed, density and volume were calculated and listed in Table II. The average traffic speed in video #1 was 31.5 mph, which was reasonable for an urban arterial in Beijing, China. The average density was 31.9 pc/mi/lane (passenger cars per mile per lane) and the volume was 3010.6 pc/h (passenger cars per hour). The estimated speed of video #2 was much lower than that of video #1 and the density was higher. From Table II, the speed of video #2 was only 1.7 mph, and the density was 45.5 pc/mi/lane. The volume then turned out to be 309.4 pc/h, which was much lower than that of video #1.

C. System Performance Evaluation and Analysis

To validate the traffic flow parameter estimation accuracy, vehicle speed and vehicle count were selected as the metrics.

Ground truth speed data was manually collected using an on-screen pixel measurement tool. The speeds of individual vehicles were measured over intervals of five consecutive frame pairs. We considered five frames reasonable as the measuring interval. Basically, if the interval was too small, the manual measurement error of speed would be large, and if it was too large, the data resolution would deteriorate. Five frames took about 0.2 seconds, during which the traffic speed could still be viewed as constant. Hence, detected speeds was averaged for each five consecutive frames and used for the speed accuracy analysis. Ground truth vehicle count information was collected by manually counting the vehicles frame by frame.

Plots in Fig. 7 showed the estimated count, ground truth count, estimated speed and ground truth speed for the two cases. The average speeds, counts and accuracies were presented in Table II. The estimation accuracies were very high for video #1, reaching 97.0% and 95.8% for speed and vehicle count estimation, respectively. It can be seen the estimation was generally more accurate in video #1 than #2. We examined the frames and found that this was mainly caused by the traffic composition. In video #2, more buses and trucks were included, but as aforementioned, the samples for classifier training contained very few buses and trucks due to the amount of training videos available. Moreover, the road surface color was very similar to some vehicles in video #2, thus making it very challenging for vehicle localization tasks. But in general our system still achieved good estimation performance in the second case, resulting in 92.4% accuracy for speed estimation and 85.1% for vehicle count estimation.

Vehicle count estimation accuracy strongly relied on the detector's performance since the count was the number of detection windows in each frame. Speed estimation accuracy also tended to be influenced by the vehicle detection results in the proposed framework. This was because every false-positive or false-negative literally increased not only the count estimation error but also the traffic motion estimation error. According to the proposed processing logic, false-positives would include motion-vectors belonging to the background into traffic motion. Likewise, false-negatives would include motion-vectors which belong to traffic into background motion. Thus, it was reasonable to see a higher speed estimation accuracy in video #1 since its count estimation was more accurate (see Table II). Considering that volume was a product of density and speed, this finding actually showed the importance of vehicle detector performance in the proposed framework.

While our proposed system achieved good traffic flow estimation for both free flow and congestion, one interesting fact worth mentioning was that, the difference between speed estimation accuracy and count estimation accuracy was larger in congestion case than free flow (7.3% vs 1.2%). This was because false-positives and false-negatives would cause less errors to speed estimation in UAV videos with dense traffic. Specifically, in congestion scenarios, vehicles have low or even zero speed. In other words, from the UAV's view, the background motion and traffic motion are closer to each other than uncongested scenarios. Hence, false detections or missed detections would have less influence on motion estimation in our proposed framework.

As real-time traffic information is so important for traffic control or route guidance, the processing speed of our method is considered as a critical performance measurement. The experiments were conducted on a desktop computer with an Intel i7-6700 CPU @ 3.40 GHz processor and 20G of memory. Under current parameter settings, the average processing speeds for the two videos were 29.7 fps and 25.4 fps, respectively. Considering the frame rates of most videos are no more than 25 fps, real-time traffic flow parameter estimation from UAV videos can be supported.

V. Conclusion and Future Work

In this paper, we proposed a new four-stage framework to extract traffic flow parameters (i.e., speed, density, and volume) from UAV videos with moving background. In the first two stages, Haar cascade classifiers (stage 1) and convolutional neural network (stage 2) were trained separately and combined as an ensemble classifier for vehicle detection from top-view perspective. Haar cascade reduced the searching space efficiently as a region proposal method and CNN examined the remaining candidate windows as a strong classifier. In the third stage, KLT optical flow method was implemented to extract motion-vectors of both vehicles (inside detection windows) and the video background (outside detection windows) based on the detection results. Then the true traffic motion was represented by the subtraction of averaged vehicle motion and averaged background motion. In the fourth stage, a new algorithm was developed to estimate traffic flow parameters by integrating reference markings, height change detection, and traffic flow theory. The proposed ensemble classifier was compared with the state-of-the-art vehicle detectors that have been examined for UAV-based vehicle detection and demonstrated its high efficiency and accuracy. Experimental results showed that the proposed methods achieved very good estimation accuracy and real-time processing speed in both free flow and congested traffic scenarios. In addition to the methodological part, the training and testing datasets containing 20,000 image samples were made publicly available for benchmarking.

Future work will focus on the following aspects. First, our team plans to collect more samples to feed the training process to see how the detection ability could be improved. Instead of labeling the samples as positive and negative, multi-group classification (e.g., car, truck, bus, motorcycle, background) would further benefit the machine's understanding of traffic scene. Second, the proposed framework will be further evaluated over a longer monitoring time and more complicated scenarios: low video quality (e.g., low resolution, motion blur), different lighting conditions, etc. Third, in this study, we aimed at addressing the computer vision challenges in UAV research, and tested the data processing speed on a computer; in future work, we will develop the algorithm on the ARM controller to explore more operational challenges.

Acknowledgment

The authors would like to thank Beihang University for providing the UAV videos. They would also like to express

their gratitude to Mr. Sung Kim and Mr. Wenbo Zhu for their help in collecting and publishing the datasets.

REFERENCES

[1] B. Coifman, M. McCord, R. Mishalani, and K. Redmill, "Surface transportation surveillance from unmanned aerial vehicles," in *Proc. 83rd Annu. Meeting Transp. Res. Board*, Washington, DC, USA, 2004, pp. 1–9.

[2] C. Kyoungah, I. Lee, J. Hong, T. Oh, and S. W. Shin, "Developing a UAV-based rapid mapping system for emergency response," *Proc. SPIE*, vol. 7332, p. 733209, Apr. 2009.

[3] H. Zhou, H. Kong, L. Wei, D. Creighton, and S. Nahavandi, "Efficient road detection and tracking for unmanned aerial vehicle," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 297–309, Feb. 2015.

[4] J. Tang, F. Liu, Y. Zou, W. Zhang, and Y. Wang, "An improved fuzzy neural network for traffic speed prediction considering periodic characteristic," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2340–2350, Sep. 2017.

[5] S. Zhang, J. Tang, H. Wang, Y. Wang, and S. An, "Revealing intra-urban travel patterns and service ranges from taxi trajectories," *J. Transp. Geogr.*, vol. 61, pp. 72–86, May 2017.

[6] Y. Wang, R. Ke, W. Zhang, Z. Cui, and K. Henrickson, "Digital roadway interactive visualization and evaluation network applications to WSDOT operational data usage," Univ. Washington, Seattle, WA, USA, Tech. Rep. WA-RD 854.1, 2016.

[7] R. Ke, Z. Pan, Z. Pu, and Y. Wang, "Roadway surveillance video camera calibration using standard shipping container," in *Proc. IEEE Intl. Smart Cities Conf.*, Wuxi, China, Sep. 2017, pp. 1–6.

[8] Y. Lin and S. Saripalli, "Road detection from aerial imagery," in *Proc. IEEE Int. Conf. Robot. Autom.*, Saint Paul, MN, USA, May 2012, pp. 3588–3593.

[9] Z. Kim, "Realtime road detection by learning from one example," in *Proc. IEEE Workshop Appl. Comput. Vis.*, Jan. 2005, pp. 455–460.

[10] R. Pless and D. Jurgens, "Road extraction from motion cues in aerial video," in *Proc. 12th Annu. ACM Int. Workshop Geograph. Inf. Syst.*, 2004, pp. 31–38.

[11] S. Hinz, J. Leitloff, and U. Stilla, "Context-supported vehicle detection in optical satellite images of urban areas," in *Proc. IEEE Int. Conf. Geosci. Remote. Sens. Symp*, Jul. 2005, pp. 2937–2941.

[12] H.-Y. Cheng, C.-C. Weng, and Y. Chen, "Vehicle detection in aerial surveillance using dynamic Bayesian networks," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 2152–2159, Mar. 2012.

[13] X. Cao, C. Wu, J. Lan, P. Yan, and X. Li, "Vehicle detection and motion-analysis in low-altitude airborne video under urban environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 10, pp. 1522–1533, Oct. 2011.

[14] T. Zhao and R. Nevatia, "Car detection in low resolution aerial images," *Image Vis. Comput.*, vol. 21, no. 8, pp. 693–703, Aug., 2003.

[15] T. P. Breckon, S. E. Barnes, M. L. Eichner, and K. Wahren, "Autonomous real-time vehicle detection from a medium-level UAV," in *Proc. 24th Int. Unmanned Air Vehicle Syst. Conf.*, 2009, pp. 29–37.

[16] A. Gaszczak, T. P. Breckon, and J. Han, "Real-time people and vehicle detection from UAV imagery," *Proc. SPIE*, vol. 7878, p. 78780B, Jan. 2011.

[17] K. Kaaniche, B. Champion, C. Pegard, and P. Vasseur, "A vision algorithm for dynamic detection of moving vehicles with a UAV," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1878–1883.

[18] X. Cao, J. Lan, P. Yan, and X. Li, "Vehicle detection and tracking in airborne videos by multi-motion layer analysis," *Mach. Vis. Appl.*, vol. 23, no. 5, pp. 921–935, Sep. 2012.

[19] X. Cao, C. Gao, J. Lan, Y. Yuan, and P. Van, "Ego motion guided particle filter for vehicle tracking in airborne videos," *Neurocomputing*, vol. 124, pp. 168–177, Jan. 2014.

[20] Q. Yu and G. Medioni, "Motion pattern interpretation and detection for tracking moving vehicles in airborne video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 2671–2678.

[21] H. Yalcin, M. Hebert, R. Collins, and M. J. Black, "A flow-based approach to vehicle detection and background mosaicking in airborne video," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, p. 1202.

[22] C. Aeschliman, J. Park, and A. C. Kak, "Tracking vehicles through shadows and occlusions in wide-area aerial video," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 1, pp. 429–444, Jan. 2014.

[23] M. McCord, Y. Yang, Z. Jiang, B. Coifman, and P. Goel, "Estimating annual average daily traffic from satellite imagery and air photos: Empirical results," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1855, pp. 136–142, 2003.

[24] A. Angel, M. Hickman, P. Mirchandani, and D. Chandnani, "Methods of analyzing traffic imagery collected from aerial platforms," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 2, pp. 99–107, Jun. 2003.

[25] F. Yamazaki, L. Wen, and T. Vu, "Vehicle extraction and speed detection from digital aerial images," in *Proc. IEEE Int. Conf. Geosci. Remote. Sens. Symp.*, Boston, MA, USA, Jul. 2008, pp. III-1334–III-1337.

[26] C. K. Toth and D. Grejner-Brzezinska, "Extracting dynamic spatial data from airborne imaging sensors to support traffic flow estimation," *ISPRS J. Photogramm. Remote Sens.*, vol. 61, nos. 3–4, pp. 137–148, Dec. 2006.

[27] A. C. Shastry and R. A. Schowengerdt, "Airborne video registration and traffic-flow parameter estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 4, pp. 391–405, Dec. 2005.

[28] R. Ke, S. Kim, Z. Li, and Y. Wang, "Motion-vector clustering for traffic speed detection from UAV video," in *Proc. IEEE Conf. Smart Cities*, Guadalajara, Mexico, Oct. 2015, pp. 1–5.

[29] R. Ke, Z. Li, S. Kim, J. Ash, Z. Cui, and Y. Wang, "Real-time bidirectional traffic flow parameter estimation from aerial videos," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 4, pp. 890–901, Apr. 2017.

[30] G. Bradski, "The OpenCV Library," *Dr. Dobb's J. Softw. Tools*, 2000.

[31] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Dec. 2001, pp. 511–518.

[32] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. IJCAI*, 1981, pp. 674–679.

[33] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[34] G. Guido, V. Gallelli, D. Rogano, and A. Vitale, "Evaluating the accuracy of vehicle tracking data obtained from unmanned aerial vehicles," *Int. J. Transp. Sci. Technol.*, vol. 5, no. 3, pp. 136–151, Oct. 2016.

[35] X. Zhao, D. Dawson, W. A. Sarasua, and S. T. Birchfield, "Automated traffic surveillance system with aerial camera arrays imagery: Macroscopic data collection with vehicle tracking," *J. Comput. Civil Eng.*, vol. 31, no. 3, p. 04016072, Nov. 2016.

[36] Y. Du, C. Zhao, F. Li, and X. Yang, "An open data platform for traffic parameters measurement via multirotor unmanned aerial vehicles video," *J. Adv. Transp.*, vol. 2017, Jan. 2017, Art. no. 8324301.

[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[38] R. Ke, "A novel framework for real-time traffic flow parameter estimation from aerial videos," Ph.D. dissertation, Univ. Washington, Seattle, WA, USA, 2016.

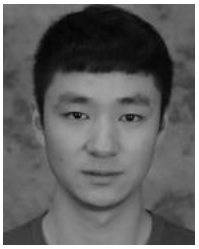
[39] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA, Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–30, 2012.

[40] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTA*, 2010, pp. 177–186.

[41] J. Redmon and A. Farhadi. (2016). "YOLO9000: Better, faster, stronger." [Online]. Available: https://arxiv.org/abs/1612.08242

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[43] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556



Ruimin Ke (S'15) received the B.S. degree from the Department of Automation, Tsinghua University, in 2014. He is currently pursuing the Ph.D. degree in civil and environmental engineering with University of Washington. Since 2014, he has been a Research Assistant with the Smart Transportation Applications and Research Laboratory, University of Washington. His research interests include intelligent transportation systems, traffic sensing technologies, traffic safety analysis, traffic flow theory, and computer vision. He serves as a member of the Statewide Transportation Data and Information Systems Committee of Transportation Research Board. He was also a Technical Program Committee Member at the 2015 IEEE International Smart Cities Conference. He is currently the Chair of Student Affairs with Chinese Overseas Transportation Association.

Zhibin Li received the Ph.D. degree from the School of Transportation, Southeast University, Nanjing, China, in 2014. From 2010 to 2012, he was a Visiting Student with University of California at Berkeley, Berkeley. From 2015 to 2016, he was a Post-Doctoral Researcher with the Smart Transportation Applications and Research Laboratory, University of Washington, USA. He is currently a Professor with Southeast University. He received China National Scholarship in 2012 and 2013, and was a recipient of the Best Doctoral Dissertation Award by China Intelligent Transportation Systems Association in 2015.

Jinjun Tang received the B.S. and M.S. degrees from Shandong University of Technology, Zibo, China, in 2006 and 2009, respectively, and the Ph.D. degree in traffic information engineering and control from Harbin Institute of Technology, Harbin, China, in 2016. He was a Visiting Student with University of Washington, Seattle, WA, USA, from 2014 to 2016. He is currently an Associate Professor with Central South University. He published over 30 technical papers in journal and conference proceeding as first author and co-author. His research interests include traffic flow prediction, data mining in transportation system, intelligent transportation systems, and transportation modeling.

Zewen Pan is currently pursuing the bachelor's degree with the College of Resources and Civil Engineering, Northeastern University, Shenyang, China. He was an Intern with the Smart Transportation Applications and Research Laboratory, University of Washington, where he was actively involved in intelligent transportation systems (ITS) research including video-based traffic data collection and mobile sensing. He will further enhance his research in ITS through his college study.

Yinhai Wang received the master's degree in computer science from University of Washington (UW) and the Ph.D. degree in transportation engineering from University of Tokyo in 1998. He is currently a Professor in transportation engineering and the Founding Director of the Smart Transportation Applications and Research Laboratory, UW. He also serves as the Director for Pacific Northwest Transportation Consortium, USDOT University Transportation Center for Federal Region 10. His active research fields include traffic sensing, urban mobility, e-science of transportation, and transportation safety. He has authored over 140 peer reviewed journal articles and delivered over 150 invited talks and nearly 240 other academic presentations. He serves as a member of the Transportation Information Systems and Technology Committee and the Artificial Intelligence and Advanced Computing Committee of the Transportation Research Board. He was an elected member of the Board of Governors for the IEEE ITS Society from 2010 to 2013 and co-chaired the First and Third IEEE International Smart Cities Conferences. He was a recipient of the ASCE Journal of Transportation Engineering Best Paper Award for 2003. He is currently a member of the IEEE Smart Cities Technical Activities Committee and a Vice President of the ASCE Transportation & Development Institute. He is also an Associate Editor for two journals: *Journal of Intelligent Transportation System* and *Journal of Transportation Engineering*.