Hindawi
Journal of Advanced Transportation
Volume 2017, Article ID 2823617, 10 pages
https://doi.org/10.1155/2017/2823617

WILEY | Hindawi

Research Article

Car Detection from Low-Altitude UAV Imagery with the Faster R-CNN

Yongzheng Xu,[1,2] Guizhen Yu,[1,2] Yunpeng Wang,[1,2] Xinkai Wu,[1,2] and Yalong Ma[1,2]

[1]*Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems and Safety Control, School of Transportation Science and Engineering, Beihang University, Beijing 100191, China*
[2]*Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, SiPaiLou No. 2, Nanjing 210096, China*

Correspondence should be addressed to Guizhen Yu; yugz@buaa.edu.cn

Received 2 December 2016; Revised 12 July 2017; Accepted 25 July 2017; Published 29 August 2017

Academic Editor: Pascal Vasseur

UAV based traffic monitoring holds distinct advantages over traditional traffic sensors, such as loop detectors, as UAVs have higher mobility, wider field of view, and less impact on the observed traffic. For traffic monitoring from UAV images, the essential but challenging task is vehicle detection. This paper extends the framework of Faster R-CNN for car detection from low-altitude UAV imagery captured over signalized intersections. Experimental results show that Faster R-CNN can achieve promising car detection results compared with other methods. Our tests further demonstrate that Faster R-CNN is robust to illumination changes and cars' in-plane rotation. Besides, the detection speed of Faster R-CNN is insensitive to the detection load, that is, the number of detected cars in a frame; therefore, the detection speed is almost constant for each frame. In addition, our tests show that Faster R-CNN holds great potential for parking lot car detection. This paper tries to guide the readers to choose the best vehicle detection framework according to their applications. Future research will be focusing on expanding the current framework to detect other transportation modes such as buses, trucks, motorcycles, and bicycles.

1. Introduction

Unmanned aerial vehicles (UAVs) hold promise of great value for transportation research, particularly for traffic data collection (e.g., [1–5]). UAVs have many advantages over ground based traffic sensors [2]: great maneuverability and mobility, wide field of view, and zero impact on ground traffic. Due to the high cost and challenges of image processing, UAVs have not been extensively exploited for transportation research. However, with the recent price drop of off-the-shelf UAV products and widely applications of surveillance video technologies, UAVs are becoming more prominent in transportation safety, planning, engineering, and operations.

For UAV based applications in traffic monitoring, one essential task is vehicle detection. This task has become challenging due to the following reasons: varying illumination conditions, background motions due to UAV movements, complicated scenes, and different traffic conditions (congested or noncongested). Many traditional techniques, such as background subtraction [6], frame difference [7], optical flow [8], and so on, can only achieve low accuracy; and some methods, such as frame difference and optical flow, can only detect moving vehicles. In order to improve detection accuracy and efficiency, many object detection schemes have been applied for vehicle detection from UAV images, including Viola-Jones (V-J) object detection scheme [9], the linear support machine (SVM) with histogram of orientated gradient (HOG) features [10] (SVM + HOG), and Discriminatively Trained Part Based Models (DPM) [11]. Generally, these object detection schemes are less sensitive to image noise and complex scenarios therefore are more robust and efficient for vehicle detection. However, most of these methods are sensitive to objects' in-plane rotation; that is, only objects in one particular orientation can be detected. Furthermore, many methods, like V-J, are sensitive to illumination changes.

In recent years, convolutional neural network (CNN) has shown impressive performance on object classification and detection. The structure of CNN was first proposed by LeCun et al. [12]. As a feature learning architecture, CNN contains convolution and max-pooling layers. Each convolutional

layer of CNN generates feature maps using several different convolution kernels on the local receptive fields from the preceding layer. The output layer in the CNN combines the extracted features for classification. By applying down-pooling, the sizes of feature map can be decreased and the extracted features become more complex and global. Many studies [13–15] have shown that CNN can achieve promising performance in object detection and classifications.

However, directly combining CNN with sliding window strategy has difficulties to precisely localize objects [16, 17]. To address above issues, region-based CNN, that is, R-CNN [18], SPPnet [19], and Fast-R-CNN have, been proposed to improve objected detection performance. But the region proposal generation step consumes too much computation time. Therefore, Ren et al. further improved Fast R-CNN [20] and developed the Faster R-CNN [21], which achieves state-of-the-date object detection accuracy with real-time detection speed. Inspired by the success of Faster R-CNN [21] in object detection, this research aims to apply Faster R-CNN [21] for vehicle detection from UAV imagery.

The rest of the paper is organized as follows: Section 2 briefly reviews some related work about vehicle detection with CNN from UAV images, followed by the methodological details of the Faster R-CNN [21] in Section 3. Section 4 presents a comprehensive evaluation of the Faster R-CNN for car detection. Section 5 presents a discussion on some key characteristics of Faster R-CNN. Finally, Section 6 concludes this paper with some remarks.

2. Related Work

A large amount of research has been performed on vehicle detection over the years. Here we only focus on vehicle detection with CNN from UAV images. Some of the most related work is reviewed here.

Pérez et al. [22] developed a traditional object detection framework based on the sliding window strategy with a classifier. This paper designed a simple CNN network instead of using traditional classifiers (SVM, Boosted Trees, etc.). As the sliding window strategy is time-consuming when handling multiscale objects detection, the framework of [22] is time-consuming for vehicle detection from UAV images.

Ammour et al. [23] proposed a two-stage car detection method, including candidate regions extraction and classification stage. In the candidate regions extraction stage, the authors employed the mean-shift algorithm [24] to segment images. Then fine-tuned VGG16 model [25] was used to extract region feature. Finally, SVM was used to classify the features into "car" and "non-car" objects. The proposed framework of [23] is similar to R-CNN [18], which was time-consuming when generating region proposals. Besides, different models should be trained for the three separate stages, which increases the complexity of [23].

Chen et al. [15] proposed a hybrid deep convolutional neural network (HDNN) for vehicle detection in satellite images to handle large-scale variance of vehicles. However, when applying HDNN for vehicle detection from satellite images, it takes about 7-8 seconds to detect one image even using Graphics Processing Unit (GPU).

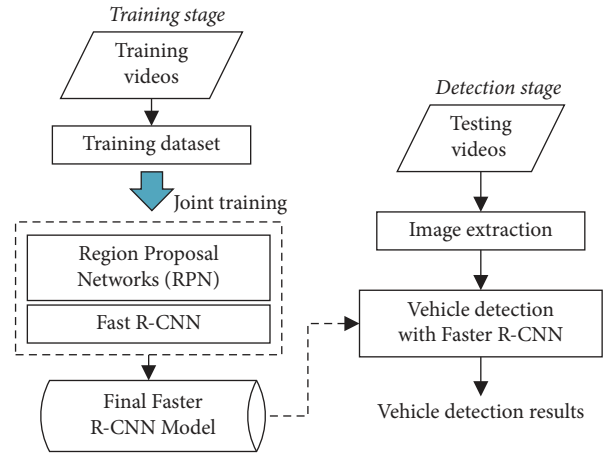


Figure 1: Car detection framework with the Faster R-CNN.

Inspired by the success of Faster R-CNN in both detection accuracy and detection speed, this work proposed a car detection method based on Faster R-CNN [21] to detect cars from low-altitude UAV imagery. The details of the proposed method are presented in the following section.

3. Car Detection with Faster R-CNN

Faster R-CNN [21] has achieved state-of-the-art performance for multiclass object detection in many fields (e.g., [19]). But so far no direct application of Faster R-CNN on car detection from low-altitude UAV imagery, particularly under urban environment, has been applied. This paper aims to fill this gap by proposing a framework for car detection from UAV images using Faster R-CNN, as shown in Figure 1.

3.1. Architecture of Faster R-CNN. The Faster R-CNN consists of two modules: the Regional Proposal Network (RPN) and the Fast R-CNN detector (see Figure 2). RPN is a fully convolutional network for efficiently generating region proposals with a wide range of scales and aspect ratios which will be fed into the second module. Region proposals are rectangular regions which may or may not contain candidate objects. Fast R-CNN detector, the second module, is used to refine the proposals. The RPN and Fast R-CNN detector share the same convolutional layers, allowing for joint training. The Faster R-CNN runs through the CNN only once for the entire input image and then refines object proposals. Due to the sharing of convolutional layers, it is possible to use a very deep network (e.g., VGG16 [25]) for generating high-quality object proposals. The entire architecture is a single and unified network for object detection (see Figure 2).

3.2. Fast R-CNN Detector. The Fast R-CNN detector takes multiple regions of interest (RoIs) as input. For each RoI (see Figure 2), a fixed-length feature vector is extracted by the RoI pooling layer from the convolutional layer. Each feature vector is fed into a sequence of fully connected (FC) layers. The final outputs of the detector through the softmax layer and the bounding-box regressor layer include (1) softmax

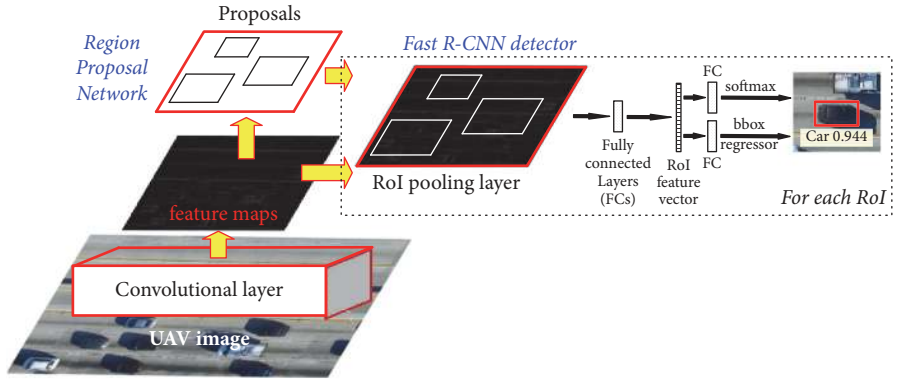


Figure 2: The architecture of Faster R-CNN, from [20, 21].

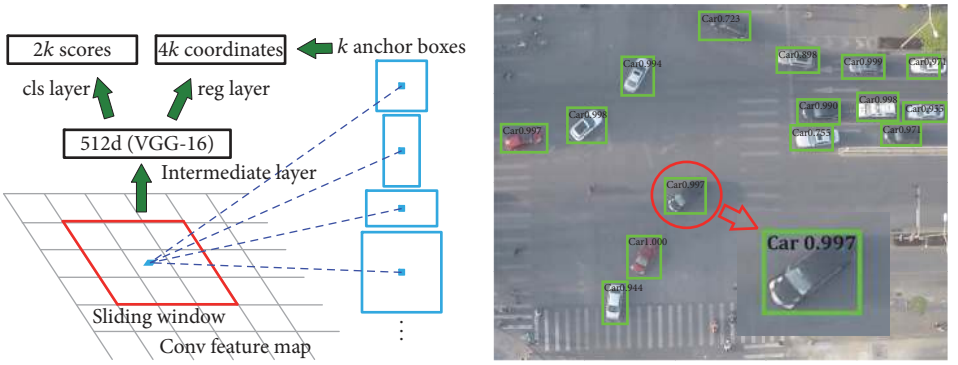


Figure 3: (a) Region Proposal Network (RPN), from [21]. (b) Car detection using RPN proposals on our UAV image.

probabilities which estimate over K object classes plus the "background" class and (2) related bounding-box (bbox) values. In this research, the value of K is 1, namely, the object classes only contain one object "passenger car" plus the "background" class.

3.3. Region Proposal Networks and Joint Training. When using RPN to predict car proposals from UAV images, the RPN takes a UAV image as input and outputs a set of rectangular car proposals (i.e., bounding boxes), each with an objectless score. In this paper, the VGG-16 model [25], which has 13 shareable convolutional layers, was used as the Faster-RCNN convolutional backend.

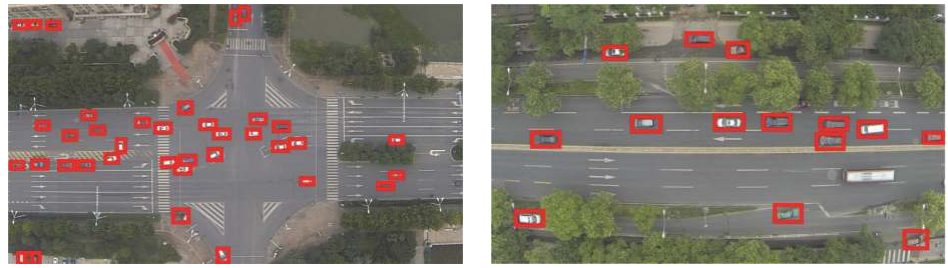
The RPN utilizes sliding windows over the convolutional feature map output by the last shared convolutional layer to generate rectangular region proposals for each position (see Figure 3(a)). A $n \times n$ spatial window (filter) was convolved with the input convolutional feature map. Then each sliding window is projected to a lower-dimensional feature (512-d for VGG-16), by convolving with two 1 by 1 filters, respectively, for a box-regression layer (reg) and a box-classification layer (cls). For each sliding window location, k possible proposals (i.e., anchors in [21]) were generated in the cls layer. For the reg layer, $4k$ outputs were generated to encode the coordinates of k bounding boxes. Meanwhile, $2k$ objectness scores were output in the cls layer to estimate probability whether each proposal contains a car or a non-car object (see Figure 3(b)).

As many proposals highly overlap with each other, nonmaximum suppression (NMS) was applied to merge proposals that have high intersection-over-union (IoU). After NMS, the remaining proposals were ranked based on the object probability score, and only the top N proposals are used for detection.

For training RPNs, each proposal is assigned a binary class label which indicates whether the proposal is an object (i.e., car) or just background. A positive training example is designated if the proposal overlaps with a ground-truth box with an IoU more than a predefined threshold (0.7 in [21]), or if it has the highest IoU with a ground-truth.

A proposal will be assigned as a negative example if its maximum IoU is lower than the predefined threshold (0.3 in [21]) for all ground-truth boxes. Following the multitask loss in Fast R-CNN network [20], the RPN is trained by a multitask loss, which is defined as

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{\text{cls}}}\sum_i L_{\text{cls}}\left(p_i, p_i^*\right) + \lambda\frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}\left(t_i, t_i^*\right), \tag{1}$$



(a) (b)

Figure 4: Car detection. (a) Signalized intersection; (b) arterial road.

where i is the index of an anchor and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive and 0 if the anchor is negative. The multitask loss has two parts, a classification component L_{cls} and a regression component L_{reg}. In (1), t_i is a vector representing the 4 parameterized coordinates of the predicted bounding-box; and t_i^* is the vector of the ground-truth box associated with a positive anchor. These two terms are normalized by N_{cls} and N_{reg} and weighted by a balancing parameter λ. In the released code [26], the cls term in (1) is normalized by the minibatch size (i.e., $N_{\text{cls}} = 256$), the reg term is normalized by the number of anchor locations (i.e., $N_{\text{reg}} \sim 2{,}400$), and λ is set as 10.

Bounding-box regression is to find the best nearby ground-truth box of an anchor box. The parameterization of the 4 coordinates of an anchor is described as follows:

$$
\begin{aligned}
t_x &= \frac{(x - x_a)}{w_a}, \\
t_y &= \frac{(y - y_a)}{h_a}, \\
t_w &= \log\left(\frac{w}{w_a}\right), \\
t_h &= \log\left(\frac{h}{h_a}\right), \\
t_x^* &= \frac{(x^* - x_a)}{w_a}, \\
t_y^* &= \frac{(y^* - y_a)}{h_a}, \\
t_w^* &= \log\left(\frac{w^*}{w_a}\right), \\
t_h^* &= \log\left(\frac{h^*}{h_a}\right),
\end{aligned}
\tag{2}
$$

where x, y, w, and h denote the bounding-box's center coordinates, width, and height, respectively. x, x_a, and x^* are for the predicted box, anchor box, and ground-truth box, respectively. Similar definitions apply for y, w, and h.

The bounding-box regression is achieved by using features with the same spatial size on the feature maps. A set of k bounding-box regressors are trained to adapt for varying size.

Since the RPN and Fast R-CNN detector can share the same convolutional layers, these two networks can be trained jointly to learn a unified network through the following 4-step training algorithm: first, training the RPN as described above; second, training the detector network using proposals generated by the RPN trained in the first step; third, initializing RPN training by the detector network but only train the RPN specific layers; and finally, training the detector network using the new RPN's proposals. Figure 4 shows two screenshots of car detection with the Faster R-CNN.

4. Experiments

4.1. Data Set Descriptions. The airborne platform used in this research is a DJI Phantom 2 quadcopter integrated with a 3-axis stabilized gimbal (see Figure 5).

Videos are collected by a Gopro Hero Black Edition 3 camera mounted on the UAV. The resolution of the videos is 1920 × 1080 and the frame rate is 24 frames per second (f/s). The stabilized gimbal is used to stabilize the videos and eliminate video jitters caused by UAV therefore greatly reducing the impact from external factors, such as wind. In addition, an On-Screen Display (OSD), an image transmission module, and a video monitor are installed in the system for data transmission and airborne flying status monitoring and control.

A UAV image dataset is built for training and testing the proposed car detection framework. For training video collection, we followed the following two key suggestions: (1) collecting videos with cars of different orientations; (2) collecting videos with cars of a wide range of scales and aspect ratios. To collect videos with cars of different orientations, UAV videos from signalized intersections were recorded; since cars at intersections have different orientations while making turning. To collect videos covering cars of a wide range of scales and aspect ratio, UAV videos at different flight height, ranging from 100 m to 150 m, were recorded. In this work, UAV videos were collected from two different signalized intersections. For each intersection, videos 1-hour long were captured. Totally, videos two hours long were collected for building the training and testing datasets.

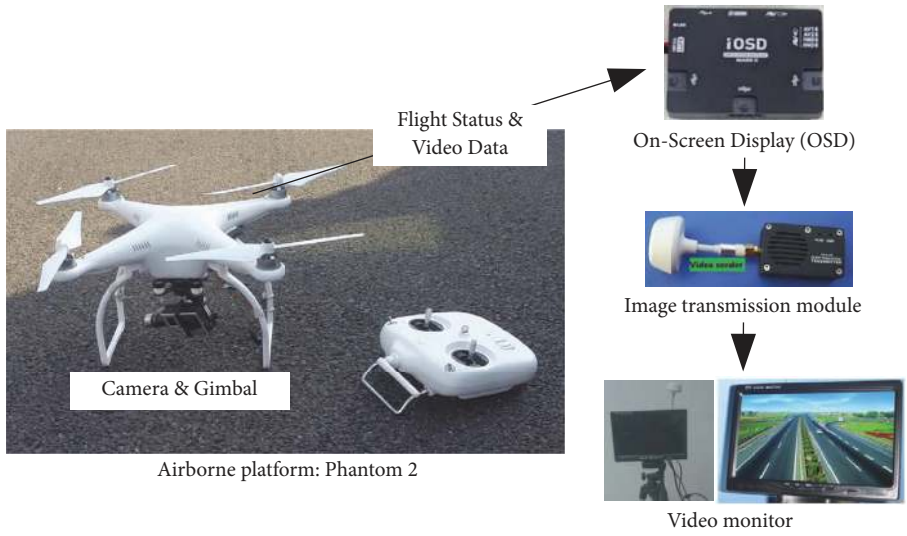


Figure 5: UAV system architecture.

In our experiment, the training and testing datasets include 400 and 100 images, respectively. Note the images for training and testing are collected from different UAV videos. The whole dataset contains 400 images with 12,240 samples for training and 100 images with 3,115 samples for testing. Note the samples for training and testing are collected from different UAV videos. Training and testing samples are annotated using the tool LabelImg [27]. During the testing and training stage, in order to avoid the same car in consecutive frames being used too many times, images were extracted every 10 seconds from UAV videos.

4.2. Training Faster R-CNN Model. Faster-RCNN was powerful in multiclass object detection. But in this research, we only trained the Faster-RCNN model for passenger cars. Particularly, we applied the VGG-16 model [25]. For the RPN of the Faster-RCNN, 300 RPN proposals were used. The source code of Faster R-CNN was from [26]. GPU was used during the training. The main configurations of the computer used in this research are

(i) CPU: Intel Core i7 hexa-core 5930 K@3.5 GHz, 32 GB DDR4;

(ii) Graphics card: Nvidia TITAN X, 12 GB GDDR5;

(iii) Operating system: Linux (Ubuntu 14.04).

The training and detection implementation in this paper is all performed on the open source code released by the authors of Faster R-CNN [21]. The inputs for training and testing are images with the original size (1920 × 1080) without any preprocessing steps.

4.3. Performance Evaluation

4.3.1. Evaluation Indicator. The performance of car detection by Faster R-CNN is evaluated by four typical indicators: detection speed (frames per second, f/s), Correctness, Completeness, and Quality, as defined in (3):

$$\begin{aligned} \text{Correctness} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{Completeness} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{Quality} &= \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}, \end{aligned} \tag{3}$$

where TP is the number of "true" detected cars; FP is the number of "false" detected objects which are non-car objects; and FN is the number of cars missed. In particular, Quality is considered as the strictest criterion, which contains both possible detection errors (false positives and false negatives).

4.3.2. Description of Algorithms for Comparison. To comprehensively evaluate the car detection performance of Faster R-CNN from UAV images, four other algorithms were included for comparison. The four algorithms are

(1) ViBe, a universal background subtraction algorithm [6];

(2) Frame difference [7];

(3) The AdaBoost method using Haar-like features (V-J) [9];

(4) Linear SVM classifier with HOG features (HOG + SVM) [10].

As ViBe [6] and frame difference [7] are sensitive to background motions, image registration [28] is applied first to compensate UAV motions and delete UAV video jitters. The time for image registration is included in the detection time for these two methods. The performance indicators are calculated based on the same 100 images as the testing dataset.

Table 1: Car detection results.

Metrics	ViBe	Frame difference	V-J	HOG + SVM	Faster R-CNN
Correctness (%)	76.64%	78.17%	84.74%	84.33%	98.43%
Completeness (%)	38.65%	39.78%	41.89%	43.18%	96.40%
Quality (%)	34.58%	35.80%	38.96%	39.97%	94.94%
Detection speed (f/s)					
CPU mode	7.42	11.83	3.38	1.45	0.018
GPU mode	N/A	N/A	20.61	6.82	2.10

Note, for ViBe and Frame Difference, the postprocessing for blob segmentation results is very important for the final car detection accuracy as blob segmentation using ViBe and Frame Difference may yield segmentation errors. In this work, two rules are designed to screen out segmentation errors: (1) the area of a detected blob is too large (2 times larger than that of a normal passenger car) or too small (smaller than 1/2 of a normal passenger car); (2) the aspect ratio of the minimum enclosing rectangle of a detected blob is larger than 2. Note, the area of the normal passenger car was obtained by human. If any of the two rules is met, the detected blob will be screened out as segmentation errors.

The V-J [9] and HOG + SVM [10] methods are trained on 12,240 positive samples and 30,000 negative samples. These 12,240 samples only contain cars orientated in the horizontal direction. Besides, all positive samples are normalized to a compressed size of 40×20. The performance evaluations of Faster R-CNN, V-J, and HOG + SVM are run on our testing dataset (100 images, 3,115 testing samples).

4.3.3. Experiment Results. The testing results of five methods are presented in Table 1. The detection speed was an average of the 100 tested images. To comprehensively evaluate the performance of different algorithms on both CPU and GPU architectures, detection speeds for V-J, HOG + SVM, and Faster R-CNN were tested on the i7 CPU and the high-end GPU, respectively.

The results show that Faster R-CNN achieved the best *Quality* (94.94%) compared with other four methods. ViBe and Frame Difference achieved fast detection speed under CPU mode but with very low *Completeness*. The reason is that many stopped cars (such as cars waiting at the traffic light) are recognized as background objects, therefore generating many false negatives and leading to a low *Completeness*. Only when those stopped cars run again could they be detected. As many moving non-car objects (such as tricycles and moving pedestrians) lead to false positives, the *Correctness* of those two methods is low (76.64% and 78.17%, resp.).

Although the two object detection schemes V-J and HOG + SVM are nonsensitive to image background motions compared with ViBe and Frame Difference, the *Completeness* of these two methods is also as low as 41.61% and 42.89%, respectively, which is only slightly higher than that of ViBe and Frame Difference. The reason, as mentioned in Section 1, is that both V-J and HOG + SVM are sensitive to objects' in-plane rotation. Only cars in the same orientation with the positive training samples could be detected. In this paper, only cars in the horizontal direction can be detected. A sensitivity analysis of the impact of cars' in-plane rotations has been provided in Discussion.

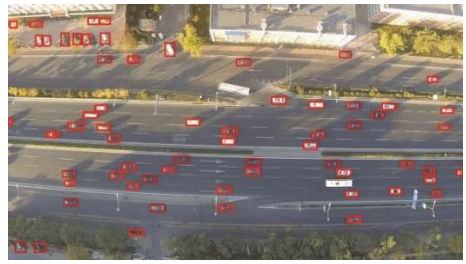


Figure 6: Car detection under illumination changing condition using Faster R-CNN.

The method of Faster R-CNN achieved the best performance (*Quality*, 94.94%) among all five methods. As Faster R-CNN can intelligently learn the information of orientation, aspect ratio, and scale during training, this method is not sensitive to cars' in-plane rotation and scale variations. Therefore, Faster R-CNN achieves high *Correctness* (98.43%) and *Completeness* (96.40%).

Though Faster R-CNN achieved 2.1 f/s under GPU mode, which is slower than other methods, 2.1 f/s can still satisfy real-time applications.

5. Discussion

5.1. Robustness to Illumination Changing Condition. For car detection from UAV videos, one most challenging issue is the illumination changing. Our testing datasets (100 images, 3,115 testing samples) do not contain cars in such scenes; for example, cars travel from an illumination (or shadowed) area to a shadowed (or illumination) area. Therefore, we further conducted an experiment using a 10 min long video captured under illumination changing condition to evaluate the performance of the Faster R-CNN (see Figure 6).

The testing results are highlighted in Table 2. The results show that Faster R-CNN achieved a *Completeness* of 94.16%, which is slightly lower than that in Table 1 (96.40%), due to the oversaturation of the image sensor under strong illumination condition. The *Correctness* of Faster R-CNN is 98.26%. The results shown in Table 2 confirm that illumination changing condition has little impact on the accuracy of vehicle detection using Faster R-CNN.

Table 2: Vehicle detection under illumination changing condition.

Metrics	ViBe	Frame difference	V-J	HOG + SVM	Faster R-CNN
Correctness (%)	81.91%	80.15%	87.27%	88.45%	98.26%
Completeness (%)	67.90%	64.69%	81.36%	82.38%	94.16%
Quality (%)	59.05%	55.76%	72.73%	74.38%	92.61%

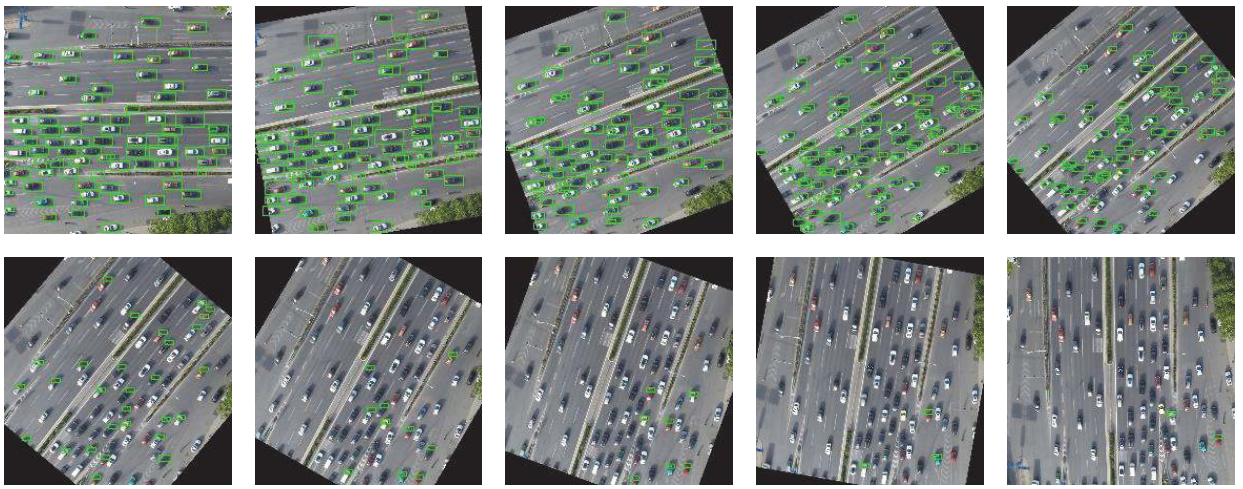


Figure 7: Car detection by HOG + SVM using image dataset which contain cars orientated in different orientations (0°, 10°, 20°, 30°, 40°, 50°, 60°, 70°, 80°, and 90°).

The methods of ViBe and Frame Difference achieved higher *Quality* than that shown in Table 1. That is because this test scene is an arterial road (see Figure 6), where most cars were running fast along the road; therefore these moving cars can be easily detected by ViBe and Frame Difference. However, many black cars that have similar color as the road surface and cars under strong illuminations could not be detected; therefore, the *Completeness* of ViBe and Frame Difference are still low (67.90% and 64.69%, resp.). The V-J and HOG + SVM methods achieved higher *Completeness* (81.36% and 82.38%, resp.) than those shown in Table 1 (41.61% and 42.89%, resp.); because most of these cars in this testing scene (see Figure 6) are orientated in the horizontal direction; thus these vehicles can be successfully detected by V-J and HOG + SVM. However, the *Completeness* of these two methods is significantly lower than that of the Faster R-CNN. As argued by some research [29], methods like the V-J method are sensitive to lighting conditions.

5.2. Sensitivity to Vehicles' In-Plane Rotation. As mentioned in Section 1, methods like V-J and HOG + SVM are sensitive to vehicles' in-plane rotation. As the vehicle orientations are generally unknown in UAV images, the detection rates *(Completeness)* of different methods may be affected significantly by the vehicles' in-plane rotation.

To analyze the sensitivity of different methods to vehicles' in-plane rotation, experiments are conducted based on dataset which contains vehicles orientated in different directions (see Figure 7). The dataset contains 5 groups of images; each group contains 19 images which orientated in different orientations as $0°, 5°, 10°, \ldots, 85°, 90°$ at an interval of 5°.

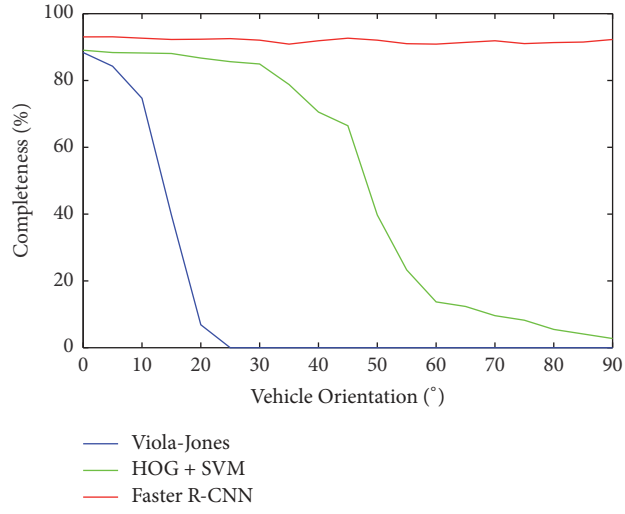


Figure 8: Sensitivity to vehicles' in-plane rotation.

From Figure 8 we can see that the *Completeness* of the V-J downgrades significantly as the vehicles' orientation exceeds 10 degrees. Compared to V-J, HOG + SVM is less sensitive to vehicles' in-plane rotation, but the *Completeness* of HOG + SVM still downgrades significantly when the vehicles' orientation exceeds about 45 degrees.

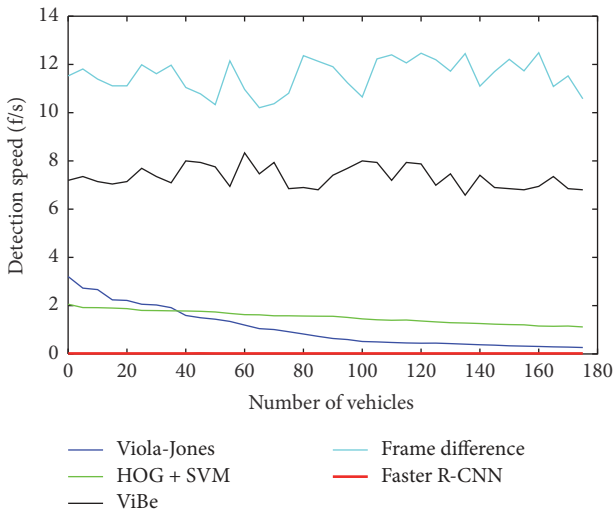


Figure 9: Sensitivity of detection speed to different detection load (tested on i7 CPU).

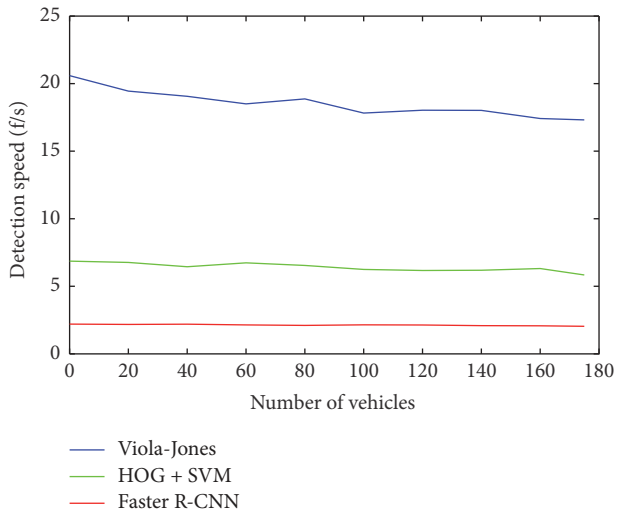


Figure 10: Sensitivity of detection speed to different detection load (tested on GPU).

Compared with V-J and HOG + SVM, Faster R-CNN is insensitive to vehicles' in-plane rotation (the red curve in Figure 8). The reason is that the Faster R-CNN can automatically learn the information of orientation, aspect ratio, and scale of vehicles from vehicle training samples during the training. Therefore, Faster R-CNN is insensitive to vehicles' in-plane rotation.

5.3. Sensitivity of Detection Speed to Different Detection Load. Detection speed is crucial for real-time applications. Detection speed can be easily affected by many factors, such as the detection load (i.e., the number of detected vehicles in one image), hardware configuration, and video resolution. Among these factors, the most important factor is detection load.

To comprehensively explore the speed characteristic of Faster R-CNN, experiments on images which contain different number of detected vehicles have been conducted (see Figure 9). Other four methods are also included for comparison. To fairly evaluate the detection speed of different algorithms on different architectures, the speed tests are performed on the i7 CPU and the high-end GPU, respectively. We explored the detection speed on i7 CPU for all five methods (see Figure 9) and explored the detection speed on GPU for VJ, HOG + SVM, and Faster R-CNN (see Figure 10).

From Figure 9 we can see that the detection speeds of V-J and HOG + SVM are monotonically decreasing with the increase of the number of detected vehicles. The V-J method presents a higher descending rate than HOG + SVM as the number of detected vehicles increases. The speed curves of ViBe and Frame Difference are unsmooth, but we can see that the increase of the number of detected vehicles has little influence on the detection speed of the two methods.

The detection speed of Faster R-CNN was very slow under CPU mode (see Figure 9). Under GPU mode (see Figure 10), the detection speed of Faster R-CNN was about 2 f/s. From Figures 9 and 10, we can find that the Faster R-CNN holds similar speed characteristic as the ViBe and Frame Difference but with a smooth speed curve. The detection load almost has no influence on the detection speed of Faster R-CNN. The reason is that when detecting vehicles using Faster R-CNN, the method is applied on the entire image. In the proposal regions generation stage, 2000 RPN proposals are generated from the original image [21]. The top-300 ranked proposal regions are fed into the Fast R-CNN [20] to check whether the proposal region contains one car. The computational cost is almost the same for each frame; therefore, the detection speed of Faster R-CNN is nearly insensitive to detection load.

Table 3: Training cost.

Metrics	V-J	HOG + SVM	Faster R-CNN
Training time	6.8 days	5 minutes	21 hours

5.4. Training Cost Comparison. When applying the Faster R-CNN for vehicle detection, one important issue that should be considered is the computational cost of training procedures. As the training samples may change, it is necessary to efficiently update the Faster R-CNN model to satisfy the requirement of vehicle detection. The training costs of three different methods are shown in Table 3. Because the open source code of Faster R-CNN can only support training function under GPU mode, only training time under GPU mode was provided. For V-J and HOG + SVM, as the open source code only supports CPU mode, only training time under CPU mode was provided.

As shown in Table 3, the AdaBoost method using Haar-like features (V-J) trained on 12,240 positive samples and 30,000 negative samples takes about 6.8 days. The training procedure was only run on CPU without parallel computing or other acceleration schemes. The linear SVM classifier with HOG features (HOG + SVM) shares the fastest training speed among all the three methods. It only takes 5 minutes on the same training set as the V-J method. Although HOG + SVM

has the fastest training speed, its detection performance is significantly lower than that of Faster R-CNN (see Table 1). The training of Faster R-CNN takes about 21 hours to complete. For practical applications, 21 hours is acceptable, as the annotation of training samples may take several days. For example, in this paper, the annotation of the whole dataset (12,240 training samples and 3,115 testing samples, totally 500 images) using the tool LabelImg [27] costs 4 days by two research fellows.

6. Concluding Remarks

Inspired by the impressive performance achieved by Faster R-CNN on object detection, this research applied this method for passenger car detection from low-altitude UAV imagery. The experimental results demonstrate that Faster R-CNN can achieve highest Completeness (96.40%) and Correctness (98.43%) with real-time detection speed (2.10 f/s), compared with four other popular vehicle detection methods.

Our tests further demonstrate that Faster R-CNN is robust to illumination changing and cars' in-plane rotation; therefore, Faster R-CNN can be applied for vehicle detection from both static and moving UAV platforms. Besides, the detection speed of Faster R-CNN is insensitive to the detection load (i.e., the number of detected vehicles). The training cost of Faster R-CNN network is about 21 hours, which is acceptable for practical applications.

It should be emphasized that this research provided a rich comparison of different vehicle detection techniques which covers a lot of aspects of object detection challenges that are usually partially covered in object detection papers: detection rate without in-plane rotation, sensitivity to in-plane rotation, detection speed, and sensitivity to the number of vehicle in the image as well as the training cost. This paper tries to guide the readers to choose the best framework according to their applications.

However, due to the lack of enough training samples, this research only tested the Faster-RCNN networks for passenger cars. Future research will expand this method for the detection of other transportation modes such as buses, trucks, motorcycles, and bicycles.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is partially supported by the Fundamental Research Funds for the Central Universities and partially by the National Science Foundation of China under Grant nos. 61371076 and 51278021.

References

[1] A. Angel, M. Hickman, P. Mirchandani, and D. Chandnani, "Methods of analyzing traffic imagery collected from Aerial platforms," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 2, pp. 99–107, 2003.

[2] M. Hickman and P. Mirchandani, "Airborne traffic flow data and traffic management," in *Proceedings of the 75 Years of the Fundamental Diagram for Traffic Flow Theory: Greenshields Symposium*, pp. 121–132, 2008.

[3] B. Coifman, M. Mccord, R. G. Mishalani, and K. Redmill, Surface Transportation Surveillance from Unmanned Aerial Vehicles, 2004.

[4] J. Leitloff, D. Rosenbaum, F. Kurz, O. Meynberg, and P. Reinartz, "An operational system for estimating road traffic information from aerial images," *Remote Sensing*, vol. 6, no. 11, pp. 11315–11341, 2014.

[5] B. Coifman, M. McCord, R. Mishalani, M. Iswalt, and Y. Ji, "Roadway traffic monitoring from an unmanned aerial vehicle," *IEE Proceedings-Intelligent Transport Systems*, vol. 153, no. 1, pp. 11–20, 2006.

[6] O. Barnich and M. van Droogenbroeck, "ViBe: a universal background subtraction algorithm for video sequences," *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709–1724, 2011.

[7] A. C. Shastry and R. A. Schowengerdt, "Airborne video registration and traffic-flow parameter estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 4, pp. 391–405, 2005.

[8] H. Yalcin, M. Hebert, R. Collins, and M. J. Black, "A flow-based approach to vehicle detection and background mosaicking in airborne video," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR '05)*, p. 1202, San Diego, CA, USA, June 2005.

[9] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 511–518, December 2001.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 886–893, June 2005.

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[13] Y. Huang, R. Wu, Y. Sun, W. Wang, and X. Ding, "Vehicle logo recognition system based on convolutional neural networks with a pretraining strategy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1951–1960, 2015.

[14] J. Tang, C. Deng, G.-B. Huang, and B. Zhao, "Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 3, pp. 1174–1185, 2015.

[15] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 10, pp. 1797–1801, 2014.

[16] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. Lecun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proceedings of the 26th IEEE Conference on Computer Vision*

and Pattern Recognition (CVPR '13), pp. 3626–3633, IEEE, June 2013.

[17] R. Vaillant, C. Monrocq, and Y. Le Cun, "Original approach for the localization of objects in images," *IEE Proceedings: Vision, Image and Signal Processing*, vol. 141, no. 4, pp. 245–250, 1994.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 580–587, Columbus, Oh, USA, June 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[20] R. Girshick, "Fast R-CNN," in *Proceedings of the 15th IEEE International Conference on Computer Vision (ICCV '15)*, pp. 1440–1448, December 2015.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[22] A. Pérez, P. Chamoso, V. Parra, and A. J. Sánchez, "Ground vehicle detection through aerial images taken by a UAV," in *Proceedings of the 17th International Conference on Information Fusion, (FUSION '14)*, July 2014.

[23] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair, "Deep learning approach for car detection in UAV imagery," *Remote Sensing*, vol. 9, no. 4, p. 312, 2017.

[24] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Computer Science, 2014.

[26] Faster R-CNN, 2016, https://github.com/rbgirshick/py-faster-rcnn.

[27] LabelImg, 2016, https://github.com/tzutalin/labelImg.

[28] Y. Ma, X. Wu, G. Yu, Y. Xu, and Y. Wang, "Pedestrian detection and tracking from low-resolution unmanned aerial vehicle thermal imagery," *Sensors*, vol. 16, no. 4, p. 446, 2016.

[29] R. Padilla, C. F. F. Costa Filho, and M. G. F. Costa, "Evaluation of Haar Cascade Classifiers Designed for Face Detection," in *Proceedings of the International Conference on Digital Image Processing*, 2012.