R³-Net: A Deep Network for Multioriented Vehicle Detection in Aerial Images and Videos

Qingpeng Li, *Student Member, IEEE*, Lichao Mou, *Student Member, IEEE*, Qizhi Xu, *Member, IEEE*, Yun Zhang, *Member, IEEE*, and Xiao Xiang Zhu, *Senior Member, IEEE*

Abstract—Vehicle detection is a significant and challenging task in aerial remote sensing applications. Most existing methods detect vehicles with regular rectangle boxes and fail to offer the orientation of vehicles. However, the orientation information is crucial for several practical applications, such as the trajectory and motion estimation of vehicles. In this paper, we propose a novel deep network, called a rotatable region-based residual network (R³-Net), to detect multioriented vehicles in aerial images and videos. More specially, R³-Net is utilized to generate rotatable rectangular target boxes in a half coordinate system. First, we use a rotatable region proposal network (R-RPN) to generate rotatable region of interests (R-RoIs) from feature maps produced by a deep convolutional neural network. Here, a proposed batch averaging rotatable anchor strategy is applied to initialize the shape of vehicle candidates. Next, we propose a rotatable detection network (R-DN) for the final classification and regression of the R-RoIs. In R-DN, a novel rotatable position-sensitive pooling is designed to keep the position and orientation information simultaneously while downsampling the feature maps of R-RoIs. In our model, R-RPN and R-DN can be trained jointly. We test our network on two open vehicle detection image data sets, namely, DLR 3K Munich Data set and VEDAI Data set, demonstrating the high precision and robustness of our method. In addition, further experiments on aerial videos show the good generalization capability of the proposed method and its potential for vehicle tracking in aerial videos. The demo video is available at https://youtu.be/xCYD-tYudN0.

Index Terms—Aerial images and videos, deep learning, multioriented detection, remote sensing, vehicle detection.

I. INTRODUCTION

ALONG with the now widespread availability of airplanes and unmanned aerial vehicles (UAVs), the detection and localization of small targets in high-resolution airborne imagery have been attracting a lot of attentions in the remote sensing community [1]–[6]. They have numerous useful applications, to name a few, surveillance, defense, and traffic planning [7]–[11]. In this paper, vehicles are considered the small targets of interest, and our task is to automatically detect and localize vehicles from complex urban scenes (see Fig. 1). This is actually an exceedingly challenging task, because of: 1) huge differences in visual appearance among cars (e.g., colors, sizes, and shapes) and 2) various orientations of vehicles.

A. Detection of Vehicles Using Feature Engineering

Since effective feature representation is a matter of great importance to an object detection system, traditionally, vehicle detection in remote sensing images was dominated by works that make use of low-level, hand-crafted visual features (e.g., color histogram, texture feature, scale-invariant feature transform (SIFT) [12], and histogram of oriented gradients (HOG) [13]) and classifiers. For example, Shao *et al.* [14] incorporate multiple visual features, local binary pattern (LBP) [15], HOG, and opponent histogram, for vehicle detection from high-resolution aerial images. Moranduzzo and Melgani [16] first use SIFT to detect interest points of vehicles and then train a support vector machine (SVM) to classify these interest points into the vehicle and nonvehicle categories based on the SIFT descriptors. They later present an approach [17] that performs filtering operations in horizontal and vertical directions to extract HOG features and yield vehicle detection after the computation of a similarity measure, using a catalog of vehicles as a reference. Liu and Mattyus [18] make use of an integral channel concept, with Haar-like features and an AdaBoost classifier in a soft-cascade structure, to achieve fast and robust vehicle detection. ElMikaty and Stathaki [19] use a sliding window framework consisting of four stages, namely, window evaluation, extraction and encoding of features, classification, and postprocessing to detect cars in complex urban environments by using a combined feature of the

Fig. 1. Examples of multioriented vehicle detection produced with the proposed network, over two scenes taken from DLR 3K Munich Data set. Best viewed zoomed in.

local distributions of gradients, colors, and texture. Moreover, Kalantar *et al.* [20] utilize a multigraph region-based matching method to detect moving vehicles in segmented UAV video frames. Zhou *et al.* [21] apply a bag-of-words model and a local steering kernel with the sliding window strategy to detect vehicles in arbitrary orientations, but this method is quite time consuming.

B. Detection of Vehicles Using Convolutional Neural Networks

The aforementioned methods mainly rely on manual feature engineering to build a classification system. Recently, as an important branch of the deep learning family, convolutional neural networks (CNNs) have become the method of choice in many computer vision and remote sensing problems [22] (e.g., object detection [5], [7], [11], [23]–[28]) as they are capable of automatically extracting mid- and high-level features from raw images for the purpose of visual analysis. For example, Cheng and Han [29] review the recent progress of deep learning-based generic object detection in optical remote sensing images. Chen *et al.* [30] propose a vehicle detection model, called a hybrid deep neural network, which consists of a sliding window technique and CNN. The main insight behind their model is to divide feature maps of the last convolutional layer into different scales, allowing for the extraction of multiscale

features for vehicle detection. Ammour *et al.* [31] segment an input image into homogeneous superpixels that can be considered as vehicle candidate regions, making use of a pretrained deep CNN to extract features and train a linear SVM to classify these candidate regions into vehicle and nonvehicle classes. Moreover, several recent works focus on a similar task, vehicle instance segmentation. For instance, Audebert *et al.* [10] propose a deep learning-based three-stage method called "segment-before-detect" for the semantic segmentation and subsequent classification of several types of vehicles in high-resolution remote sensing images. The use of SegNet [32] in this method is capable of producing pixel-wise annotations for vehicle semantic mapping. Mou and Zhu [27] propose a unified multitask learning network that can simultaneously learn two complementary tasks, namely, segmenting vehicle regions and detecting semantic boundaries. The latter subproblem is helpful for differentiating "touching" vehicles, which are usually not correctly separated into instances.

C. Is Nonrotatable Detection Enough for Vehicle Detection?

There are already some relative researches working toward rotation-insensitive remote sensing object detection, such as [33]–[35]. However, as our survey of related work shows above, most of the existing car detection approaches have focused on nonrotatable car detection [16]–[19], [36]–[38], i.e., detecting all instances of vehicles and localizing them in the image in the form of horizontal bounding boxes with confidence scores. Detecting vehicles of arbitrary orientations in complex urban environments has received much less attention and remains a challenge for most car detection algorithms, while the orientation information of vehicles is of importance for some practical applications such as traffic monitoring. In this paper, we make an effort to build an effective, rotatable detection model for vehicles of arbitrary orientations in complicated urban scenes.

In this paper, we propose an end-to-end trainable network, rotatable region-based residual network (R^3-Net), for simultaneously localizing vehicles and identifying their orientations in high-resolution remote sensing images. To this end, we introduce a series of effective rotatable operations in the network, aiming at generating multioriented bounding boxes for our task. When directly applied to detect vehicles of arbitrary orientations, conventional detection networks (e.g., Faster R-CNN [39] and R-FCN [40]) that are primarily designed for horizontal detection would result in low precisions. In contrast, the proposed rotatable network is capable of offering better performance, especially in some complex scenes such as a crowded parking lot. In this paper, we also try to apply our network to car detection and tracking in aerial videos and find that R^3-Net is able to provide satisfactory vehicle trajectories. Moreover, when we take into account the temporal information of a video (i.e., the frame association produced by multiple object tracking algorithms), better detection results can be obtained. This paper contributes to the literature in the following three aspects.

1) We take advantage of the axial symmetry property of vehicles to create a novel network architecture, which is based on a conventional two-stage object detection
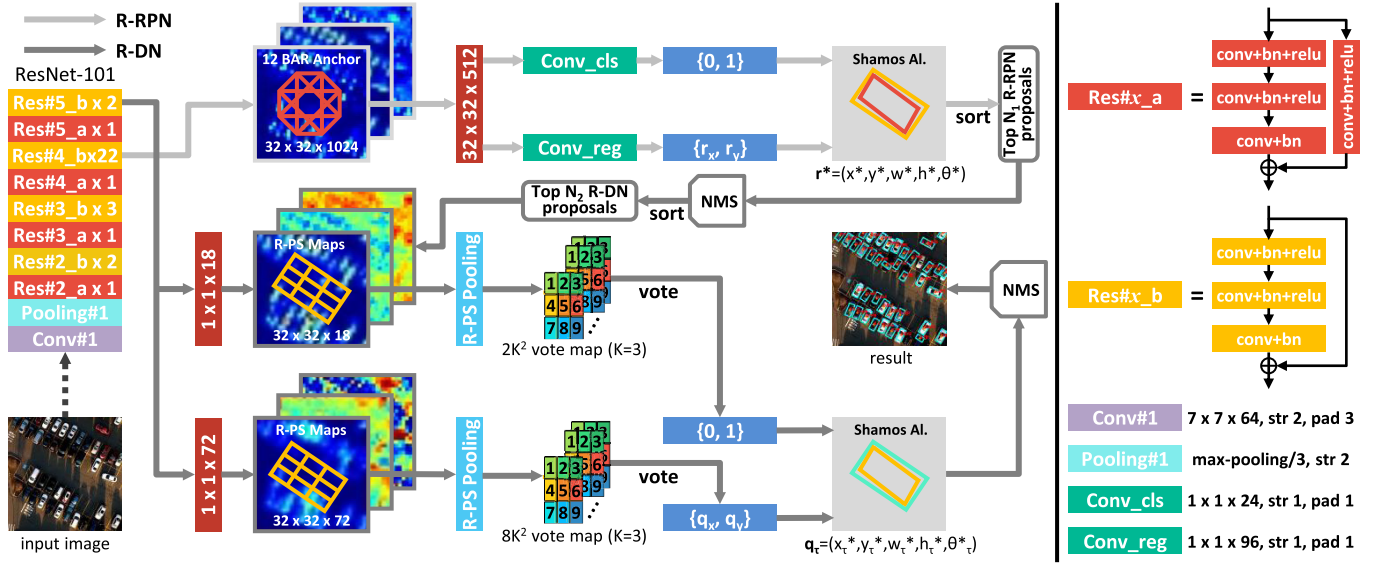
Fig. 2. Overall architecture of the proposed R^3-Net. There are two main branches. One is R-RPN, and the other is R-DN. R-RPN is for generating R-RoIs for R-DN, and then R-RoIs are fed into R-DN to produce final rotatable detection boxes. Key: str = stride; pad = padding; conv = convolution; bn = batch normalization; and relu = ReLU function.

framework (e.g., R-FCN) but able to generate and handle rotatable bounding boxes by two tailored modules, namely, rotatable region proposal network (R-RPN) and rotatable detection network (R-DN). In addition, on the top of R-RPN and R-DN, we use a modified Shamos algorithm to obtain regular quadrilaterals.

2) We propose a novel rotatable position-sensitive RoI pooling operation, namely, R-PS pooling, in order to reduce the dimension of feature maps of rotatable regions and, meanwhile, keep the information of targets in specific directions.

3) We propose a novel strategy, called batch averaging rotatable anchors (BAR anchor), to initialize rotatable anchors of region proposals using the size information of vehicles in the training set. Experimental results show that this strategy is able to estimate vehicle poses more accurately as compared to the traditional anchor generation method.

The remainder of this paper is organized as follows. After Section I, detailing vehicle detection from high-resolution remote sensing imagery, we enter Section II, dedicated to the details of the proposed R^3-Net for multioriented vehicle detection. Section III then provides data set information, implementation settings, and experimental results. Finally, Section IV concludes this paper.

II. METHODOLOGY

Our approach for vehicle detection utilizes an end-to-end trainable two-stage detection framework, including an R-RPN and an R-DN. As shown in Fig. 2, after the feature extraction by a ResNet-101 [41], we devise the framework in a rotatable detection domain to preserve competitive detection rates, and the details are elaborated in the following subsections. In addition, we also utilize two other typical deep networks for feature extraction, VGG-16 [42] and ResNet-101, with feature pyramid network (FPN) [43] (see Fig. 3).
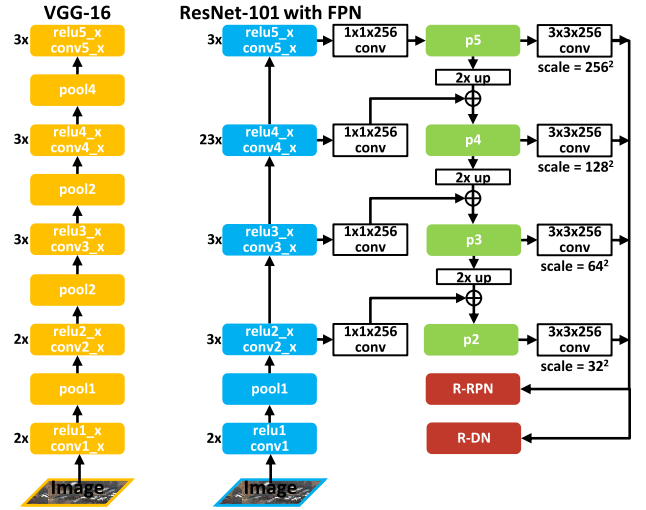
Fig. 3. Two typical network architectures: VGG-16 and ResNet-101 with FPN.

A. Rotatable Region Proposal Network

The main task of R-RPN is to generate cursory rotatable region of interests (R-RoIs) for the subsequent thorough detection. Referring to the strategy of generating RoIs in Faster R-CNN [39], we propose to take BAR anchors as the input of R-RPN and produce a series of R-RoIs as output. We next describe the main ingredients of R-RPN.

1) BAR Anchors: In the region proposal network stage of two-stage detection methods, such as Faster R-CNN [39] and R-FCN [40], anchors are initial shapes of RoIs on each feature point, and the pattern of the anchors contributes to the pattern of RoIs. For instance, an anchor in the shape of rectangle finally regresses to an RoI of rectangle shape. Likewise, an R-RoI can be obtained by a rotatable anchor, and the size of anchors depends on the size of detected objects.

With regard to the vehicle detection task from remote sensing images, we use anchors of permanent size in each training minibatch for the most invariant overall dimension of the vehicle. Formally, we consider that there are M minibatches in the training process, and batch size is set to N_m. We use $\mathcal{I}_i^{(m)}$ to represent the ith ($i = 1, \ldots, N_m$) training sample in the mth ($m = 1, \ldots, M$) minibatch, so the jth rotatable ground truth box in $\mathcal{I}_i^{(m)}$ can be represented by a coordinate $(x_{ij}, y_{ij}, w_{ij}, h_{ij}, \theta_{ij})_m$. Then we define the average width $\hat{w}_m$ and height $\hat{h}_m$ of initialized anchors in mth training minibatch as follows:

$$(\hat{w}_m, \hat{h}_m) = \frac{1}{\sum_{i=1}^{N_m} N_i} \sum_{i=1}^{N_m} \sum_{j=1}^{N_i} (w_{ij}, h_{ij}) \qquad (1)$$

where N_i is the number of rotatable ground truth boxes in $\mathcal{I}_i^{(m)}$. Hence, for the mth training minibatch, the coordinate of BAR anchor on feature point (x, y) can be defined as $\mathbf{a}^* = (x, y, \kappa\hat{w}_m, \kappa\hat{h}_m, \theta)$, where $\theta \in \{-45°, 0°, 45°, 90°\}$ and scale factor κ is set to $\{0.5, 1, 2\}$ in this paper. As a result, we get 12 BAR anchors on each feature point.

2) Definition of Parameters in Regression: Unlike traditional object detection networks that generate horizontal RoIs and represent them by 4-D vectors (i.e., an RoI's center coordinates and its width and height), the proposed method needs to produce R-RoIs, which should be defined by 5-D vectors, namely, an RoI's center coordinates (x, y), width w, height h, and intersection angle θ ($-90° < \theta \le 90°$), between its lengthwise direction and horizontal direction. However, in our experiments, we found that the use of a 8-D vector for representing R-RoI box $\mathbf{r} = (\mathbf{r_x}, \mathbf{r_y})$ and ground truth box $\mathbf{g} = (\mathbf{g_x}, \mathbf{g_y})$, where $\mathbf{r_x} = (x_1, \ldots, x_4)$, $\mathbf{r_y} = (y_1, \ldots, y_4)$, $\mathbf{g_x} = (x_{g1}, \ldots, x_{g4})$, and $\mathbf{g_y} = (y_{g1}, \ldots, y_{g4})$, can make regression loss easier to be optimized. This is mainly because the 8-D vector-based way is capable of alleviating the unsteadiness caused by parameter θ when computing the regress loss. The following experimental results will show different influences of these two definitions.

To match with the dimension of R-RoI, we also convert the BAR anchors into the 8-D vectors from $\mathbf{a}$ as follows:

$$\mathbf{a} = G(\mathbf{a}^*) = (\mathbf{a_x}, \mathbf{a_y}) \qquad (2)$$

where $G(\cdot)$ is a biunique geometric transformation, $\mathbf{a_x} = (x_{a1}, \ldots, x_{a4})$, and $\mathbf{a_y} = (y_{a1}, \ldots, y_{a4})$. Notably, the collation of these 4 vertexes in each BAR anchor's representation $\mathbf{a}$ should be rigorously matched with those in ground truth labels and R-RoIs, and the collation rule of all rotatable rectangles is defined as follows:

1) confirm the intersection angle α between its lengthwise direction and horizontal direction ($-90° < \alpha \le 90°$);
2) rotate the rectangle around its center to the horizontal direction by $-\alpha$;
3) label four vertexes in the order of coordinate.

Consequently, in the bounding box regression step, for the nth feature point f_n ($n = 1, \ldots, N$, e.g., $N = 32 \times 32$ in ResNet-101), we define $\mathbf{t}_n^{(k)} = (\mathbf{t_x}, \mathbf{t_y})_n^{(k)}$ as an 8-D vector representing eight parameterized coordinates of a predicted bounding box, and $\hat{\mathbf{t}}_n^{(k)} = (\hat{\mathbf{t}}_x, \hat{\mathbf{t}}_y)_n^{(k)}$ is its corresponding

ground truth box where k ($k = 1, \ldots, 12$) is the index of a BAR anchor on the feature point f_n. Then we define parameterizations of four coordinates as follows [44]:

$$\mathbf{t_x} = \frac{\mathbf{r_x} - \mathbf{a_x}}{\kappa\hat{w}_m}, \quad \mathbf{t_y} = \frac{\mathbf{r_y} - \mathbf{a_y}}{\kappa\hat{h}_m}$$
$$\hat{\mathbf{t}}_x = \frac{\mathbf{g_x} - \mathbf{a_x}}{\kappa\hat{w}_m}, \quad \hat{\mathbf{t}}_y = \frac{\mathbf{g_y} - \mathbf{a_y}}{\kappa\hat{h}_m}. \qquad (3)$$

3) Multitask Loss: In common with other object detection networks, such as Faster R-CNN [39] and R-FCN [40], we define a multitask loss to combine both classification and regression loss together. The loss function in R-RPN is defined as an effective multitask loss L_1 [45], which combines both classification loss and regression loss for each image. It can be computed as follows:

$$L_1 = \frac{1}{N_{\text{cls}}} \sum_{n,k} L_{\text{cls}}\big(\mathbf{p}_n^{(k)}, \hat{\mathbf{p}}_n^{(k)}\big)$$
$$+ \lambda_1 \frac{1}{N_{\text{reg}}} \sum_{n,k} \phi_n^{(k)} L_{\text{reg}}\big(\mathbf{t}_n^{(k)}, \hat{\mathbf{t}}_n^{(k)}\big) \qquad (4)$$

where $\mathbf{p}_n^{(k)} = (p^0, p^1)_n^{(k)}$ is the predicted probability of the kth ($i = 1, \ldots, 12$) anchor in the nth ($n = 1, \ldots, 1024$) feature point being background and a target, and ground truth indicator label $\hat{\mathbf{p}}_n^{(k)}$ is $(0, 1)$ if the anchor is positive (i.e., *overlap* ≥ 0.5), and is $(1, 0)$ if the anchor is negative. Ground truth indicator label $\phi_n^{(k)}$ is 1 when there exist targets in the anchor.

We define the classification loss L_{cls} as log loss over two classes (background and target) as follows:

$$L_{\text{cls}}\big(\mathbf{p}_n^{(k)}, \hat{\mathbf{p}}_n^{(k)}\big) = -\mathbf{p}_n^{(k)} \log \mathbf{p}_n^{(k)}. \qquad (5)$$

Then we define the regression loss L_{reg} as

$$L_{\text{reg}}\big(\mathbf{t}_n^{(k)}, \hat{\mathbf{t}}_n^{(k)}\big) = R\big(\mathbf{t}_n^{(k)} - \hat{\mathbf{t}}_n^{(k)}\big) \qquad (6)$$

where $R(\cdot)$ is a smooth ℓ_1 loss function which is defined in [45], and formally, it can be calculated by

$$R(x) = \begin{cases} 0.5 \times^2, & \text{if } |\times| < 1 \\ |\times| - 0.5, & \text{otherwise.} \end{cases} \qquad (7)$$

In (4), two loss terms are normalized by N_{cls} and N_{reg} and balanced by hyperparameter λ_1. In our experiments, we set $N_{\text{cls}} = 64$, $N_{\text{reg}} = 1000$, and $\lambda_1 = 10$.

4) Shamos Algorithm for R-RoIs: The rotating calipers method was first used in [46]. Shamos [46] used this method to generate all antipodal pairs of points on a convex polygon and to compute the diameter of a convex polygon in $\mathcal{O}(n)$ time. Then Houle and Toussaint [47] developed an application for computing the minimum width of a convex polygon. After R-RPN, we actually obtain R-RoIs in the form of irregular quadrilateral denoted by 8-D vectors. In order to get R-RoIs in the form of regular quadrilateral to feed them into R-DN, here, we propose to use the Shamos algorithm to calculate the minimum multioriented rectangular bounding boxes. The coordinate transformation can be described as follows:

$$\mathbf{r}^* = (x^*, y^*, w^*, h^*, \theta^*) = S(\mathbf{r}) \qquad (8)$$

where a 5-D vector $(x^*, y^*, w^*, h^*, \theta^*)$ is utilized to represent a minimum multioriented rectangular R-RoI box $\mathbf{r}^*$, which can
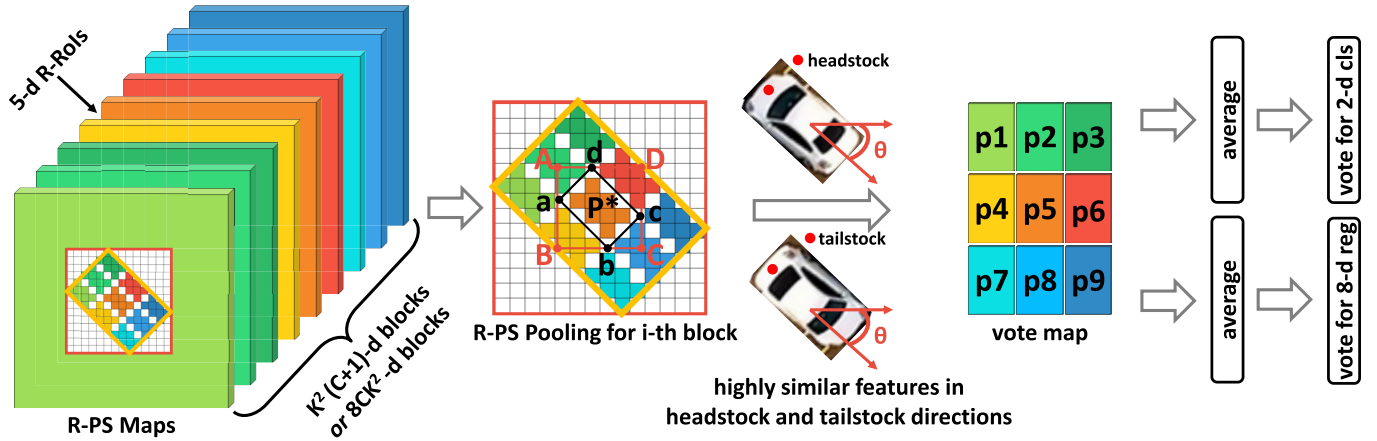
Fig. 4. Details of R-PS pooling operation. In this paper, we generate R-PS maps with 18-D ($3 \times 3 \times 2$) blocks and 72-D ($3 \times 3 \times 8$) blocks for classification task and regression task in R-DN, respectively. Each block corresponds to the relative activation of classification and regression task of 3×3 spatial positions. On the other hand, in order to avoid vehicle direction misidentification between headstock and tailstock, we span the angle of an R-RoI from $-90°$ to $90°$.

be obtained from an 8-D vector of an irregular quadrilateral R-RoI box $\mathbf{r}$ by the Shamos Algorithm $S(\cdot)$. Now, we can feed the rectangular R-RoI boxes into R-DN.

B. Rotatable Detection Network

Suppose that we generate T R-RoIs in total after R-RPN, and these R-RoIs are subsequently fed into R-DN for the final regression and classification tasks. Here, an improved position-sensitive RoI pooling strategy, called R-PS pooling, is proposed to generate scores on rotatable position-sensitive score maps (R-PS maps) for each R-RoI. We next describe the main ingredients of our R-DN.

1) R-PS Maps of R-RoIs: Given R-RoIs defined by 5-D vectors, each target proposal can be located on the feature maps extracted from an adjusted ResNet-101 [41], which uses a randomly initialized 1×1 convolutional layer with 1024 filters instead of a global average pooling layer and a fully connected layer. The size of output feature maps is $32 \times 32 \times 1024$.

For the classification task in R-DN, we apply k^2 R-PS maps for each category and $k^2(C + 1)$-channel output layer with C object categories ($C = 1$ for our vehicle detection task and $+1$ for background). The bank of k^2 R-PS maps corresponds to a $k \times k$ spatial grid describing relative positions, and we set $k = 3$ in this paper.

Different from position-sensitive score maps in R-FCN [40], the R-PS maps in our method do not encode cases of an optional object category, but cases of a potential vehicle category. As shown in Fig. 4, R-RoIs of vehicles are always approximate central symmetric so that there are always highly similar features in headstock and tailstock direction for most vehicles, making it hard to identify the exact direction. In order to avoid this almost "unavoidable" misidentification, in our model, the angle θ^* of an R-RoI $\mathbf{r}^*$ is kept spanning $-90° < \theta^* \leq 90°$. For example, $\theta^* = -45°$ shows two possible cases that the vehicle direction might be $-45°$ or $135°$, and these two possible cases are encoded into R-PS maps in the same order.

2) R-PS Pooling on R-PS Maps: We divide each R-RoI rotatable rectangular box into 3×3 bins by a parallel grid. For an R-RoI $\mathbf{r}^* = (x^*, y^*, w^*, h^*, \theta^*)$ ($w^* \leq h^*$), a bin is of size $\approx w^*/3 \times h^*/3$ [45], [48]. For the (i, j)th bin ($i, j = 1, 2, 3$), the R-PS pooling step over the (i, j)th R-PS map of cth category ($c = 1, 2$) is defined as follows:

$$r_{i,j,c}(\mathbf{w}) = \sum_{(u,v)\in\mathbf{B}_{i,j}} \frac{1}{n_p} z_{i,j,c}(u, v|\mathbf{w}) \tag{9}$$

where $r_{i,j,c}$ denotes the pooled output in the (i, j)th bin $\mathbf{B}_{i,j}$ for the cth category, $z_{i,j,c}$ represents one R-PS map out of the $k^2(C + 1)$ score maps, $\mathbf{w}$ is all learnable parameters of the network, n_p is the number of pixels in the (i, j)th bin $\mathbf{B}_{i,j}$, and (u, v) is the global coordinate of feature point $P_{i,j}$ which can be defined by the following affine transformation equation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\hat{\theta} & -\sin\hat{\theta} & u_0 \\ \sin\hat{\theta} & \cos\hat{\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \\ 1 \end{bmatrix} \tag{10}$$

where $(\Delta u, \Delta v)$ means local coordinates of feature point $P_{i,j}$ and $\lfloor (i - 1)w^*/3 \rfloor \leq \Delta u < \lceil iw^*/3 \rceil$, $\lfloor (j - 1)h^*/3 \rfloor \leq \Delta v < \lceil jh^*/3 \rceil$, and (u_0, v_0) means the top-left corner of an R-RoI. Formally, the rotation angle $\hat{\theta}$ can be calculated by

$$\hat{\theta} = \begin{cases} \theta^* - 90° & -90° < \theta^* \leq 0° \\ 90° - \theta^* & 0° < \theta^* \leq 90°. \end{cases} \tag{11}$$

When the (i, j)th bin $\mathbf{B}_{i,j}$ is pooled into a score map, we can get $(\Delta u, \Delta v)$ by (u, v) as follows:

$$\begin{aligned} \Delta u &= u\cos\hat{\theta} - v\sin\hat{\theta} - u_0 \\ \Delta v &= v\cos\hat{\theta} - u\sin\hat{\theta} - v_0 \end{aligned} \tag{12}$$

and we then get the limitation of u and v when $(u, v) \in \mathbf{B}_{i,j}$ in the following inequalities:

$$\lfloor (i - 1)w^*/3 \rfloor \leq u\cos\hat{\theta} - v\sin\hat{\theta} - u_0 < \lceil iw^*/3 \rceil$$
$$\lfloor (j - 1)h^*/3 \rfloor \leq v\cos\hat{\theta} - u\sin\hat{\theta} - v_0 < \lceil jh^*/3 \rceil. \tag{13}$$

3) Voting by Scores: After performing R-PS pooling operation on R-PS maps in 3×3 positions with 2 layers, we can then puzzle the 3×3 blocks into one voting map for each R-RoI. Here, a voting map is a kind of feature map that can keep rotatable position-sensitive features on different position blocks. Hence, we use total outputs $r_c(\mathbf{w})$ on 3×3 blocks to compute the score of the cth category by

$$r_c(\mathbf{w}) = \sum_{i,j} r_{i,j,c}(\mathbf{w}), \quad i, j = 1, 2, 3. \tag{14}$$

The softmax response of the cth category ($c = 1, 2$) can be computed as follows:

$$s_c(\mathbf{w}) = \frac{e^{r_c(\mathbf{w})}}{\sum_\delta e^{r_\delta(\mathbf{w})}}, \quad \delta = 1, 2. \tag{15}$$

4) Loss Function in R-DN: Similarly, for regression task in R-DN, we use an 8-D vector to represent eight parameterized coordinates of a predicted bounding box and apply 3×3 R-PS maps to each dimension for regression. Thus, the R-PS score maps are fed into a 1×1 convolutional layer with 72 filters for bounding box regression. Then we pool these feature maps into a 72-D vector which is aggregated into an 8-D vector by average voting for the τth ($\tau = 1, \ldots, T$) predicted bounding box $\mathbf{q}_\tau = (\mathbf{q_x}, \mathbf{q_y})_\tau$.

Likewise, we also define a multitask loss L_2 for N RoIs

$$\begin{aligned} L_2 &= \sum_\tau L(\mathbf{s}_\tau, \hat{\mathbf{p}}_\tau, \mathbf{q}_\tau, \hat{\mathbf{t}}_\tau) \\ &= \sum_\tau L_{\text{cls}}(\mathbf{s}_\tau, \hat{\mathbf{p}}_\tau) + \lambda_2 \sum_\tau \phi_\tau L_{\text{reg}}(\mathbf{q}_\tau, \hat{\mathbf{t}}_\tau) \end{aligned} \tag{16}$$

where $\hat{\mathbf{p}}_\tau$ represents ground truth (e.g., 0 and 1 stand for background and vehicle, respectively), and ground truth indicator label ϕ_τ is 1 when there exist vehicles in the τth ($\tau = 1, \ldots, T$) R-RoI's predicted box (i.e., overlap ≥ 0.5). $\mathbf{s}_\tau$ is the predicted score of the R-RoI being a vehicle, and $\mathbf{s}_\tau = (s_\tau^1, s_\tau^2)$ covers two categories. As usual, we use a fully connected layer activated by a softmax function to compute $\mathbf{s}_\tau$ in (15). Moreover, $\mathbf{q}_\tau = (\mathbf{q_x}, \mathbf{q_y})_\tau$ and $\hat{\mathbf{t}}_\tau = (\hat{\mathbf{t}}_x, \hat{\mathbf{t}}_y)_\tau$ represent coordinates of the predicted bounding box and ground truth box, which are similarly parameterized by (3). In addition, we define the classification loss L_{cls} as log loss for two classes like (5) and the regression loss L_{reg} as smooth ℓ_1 loss referring to (6) and (7). In (16), two terms of the loss are also normalized by a hyperparameter, i.e., λ_2. By default, we set $\lambda_2 = 1$.

C. Joint Loss Function

The joint loss function L of our end-to-end trainable two-stage detection framework is a combination of R-RPN loss L_1 and R-DN loss L_2, and we use a loss weight η to balance them. For each minibatch (batch size $\Theta = 64$ in our experiment), we compute the joint loss as follows:

$$L = \sum_\theta^\Theta L_1^{(\theta)} + \eta \sum_\theta^\Theta L_2^{(\theta)} + \varphi \|\mathbf{w}\|^2 \tag{17}$$

where $L_1^{(\theta)}$ and $L_2^{(\theta)}$ can be calculated by (4) and (16) for the θth ($\theta = 1, 2, \ldots, \Theta$) image in a minibatch. In this paper,
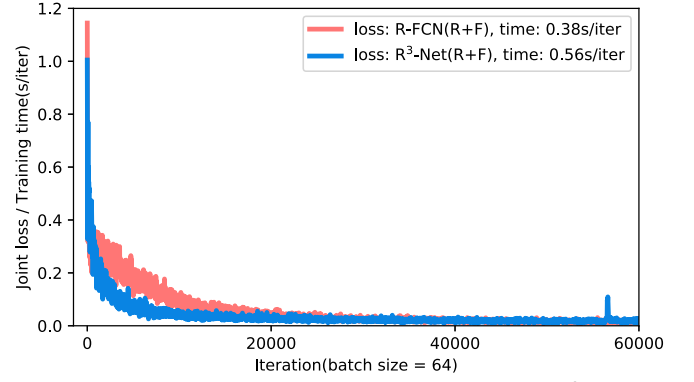

Fig. 5. Joint loss and training time curves of R-FCN and R³-Net. Key: (R + F) = ResNet-101 with FPN.

we set the weight $\eta = 1$. We also conducted experiments to find better η out, and the details will be given in Section III.

D. End-to-End Training

An end-to-end training strategy is utilized to train our model, and we use a minibatch gradient descent algorithm to update network weights $\mathbf{w}$. Here, we define $\mathbf{w} = \{\mathbf{w}_{\text{res}}, \mathbf{w}_{\text{rrpn}}, \mathbf{w}_{\text{rdn}}\}$ to represent learnable parameters of ResNet-101, R-RPN, and R-DN, respectively, and $\mathbf{w}_{\text{res}} = \{\mathbf{w}_{\text{res\_1}}, \ldots, \mathbf{w}_{\text{res\_5}}\}$ indicates parameters of five blocks in ResNet-101. In this paper, to train our model more efficiently, we only update parameters of the last two blocks of ResNet-101 and keep those of the first three blocks fixed. There are two training stages for R-RPN and R-DN, respectively. In R-RPN, feature maps are extracted by the first four blocks of ResNet-101, and $L_1^{(\theta)}$ is computed and accumulated in a minibatch Θ. In R-DN, we extract feature maps using all five blocks, compute $L_2^{(\theta)}$ of an image, and accumulate it in a minibatch Θ. We then compute the joint loss L by (17) and independently perform backpropagation at the end of each minibatch. Fig. 5 shows the joint loss and training time curves of R-FCN and R³-Net.

As aforementioned, there are two transmission routes that are regarded as forward propagation in our network. We initialize the network parameters $\{\mathbf{w}_{\text{res\_4}}, \mathbf{w}_{\text{res\_5}}\}$ and $\{\mathbf{w}_{\text{rrpn}}, \mathbf{w}_{\text{rdn}}\}$ by pretrained model and Gaussian distribution, respectively. At the end of R-RPN and R-DN, according to the discrepancy between ground truth and the stepwise output of R-RPN and R-DN, we can use the gradient descent algorithm to update learnable parameters of different parts with a learning rate ε as follows:

$$\mathbf{w}_{\text{rrpn}} := \mathbf{w}_{\text{rrpn}} - \varepsilon \sum_\theta^\Theta \frac{dL_1^{(\theta)}}{d\mathbf{w}_{\text{rrpn}}} + 2\varphi \|\mathbf{w}_{\text{rrpn}}\|$$

$$\mathbf{w}_{\text{rdn}} := \mathbf{w}_{\text{rdn}} - \varepsilon \sum_\theta^\Theta \frac{\eta dL_2^{(\theta)}}{d\mathbf{w}_{\text{rdn}}} + 2\varphi \|\mathbf{w}_{\text{rdn}}\|$$

$$\mathbf{w}_{\text{res\_5}} := \mathbf{w}_{\text{res\_5}} - \varepsilon \sum_\theta^\Theta \frac{\eta dL_2^{(\theta)}}{d\mathbf{w}_{\text{res\_5}}} + 2\varphi \|\mathbf{w}_{\text{res\_5}}\|$$

$$\mathbf{w}_{\text{res\_4}} := \mathbf{w}_{\text{res\_4}} - \varepsilon \sum_\theta^\Theta \frac{dL_1^{(\theta)} + \eta dL_2^{(\theta)}}{d\mathbf{w}_{\text{res\_4}}} + 2\varphi \|\mathbf{w}_{\text{res\_4}}\|. \tag{18}$$

Item	DLR 3K Munich Dataset	VEDAI Dataset
Category	Car, Bus Truck	Car, Pickup Truck, Van
Data tpye	RGB	RGB & NIR
Image size	512×512	512×512
Target size (Avg.)	41.14×20.04	23.70×8.25
Total image (Tr. / Te.)	575 / 408	1066 / 1066
Original veh. (Tr. / Te.)	5214 / 3054	2792 / 2702
Augment veh. (Tr. / Te.)	31284 / 3054	16752 / 2702

1) Hyperparameter Settings: Prior to network training, all new layers are randomly initialized by drawing weights from a zero-mean Gaussian distribution with standard deviation of 0.01, and all other layers are initialized by a pretrained model on ImageNet [49]. In the minibatch gradient descent, we use a learning rate ε of 0.001 for the first 10 K iterations and 0.0001 for the next 10 K iterations on the data set. The momentum and weight decay are set to 0.9 and 0.0005, respectively. As a result, we find that it works well after about 10 K iterations. The minibatch size Θ is set to 64 in this paper.

III. EXPERIMENTS

We use VGG-16 [42], ResNet-101 [41], and ResNet-101 with FPN [43] to extract features. The models are implemented using Caffe and Caffe 2 [50] and run on an NVIDIA GeForce GTX1080Ti with 12-GB on-board memory.

A. Data Set

To evaluate the performance of our method, we use two open vehicle detection data sets, namely, DLR 3K Munich Vehicle Data set [18] and VEDAI Vehicle Data set [51], in which vehicles are accurately labeled by rotatable rectangular boxes. In our experiments, we regard various types of vehicles as one category. A statistic of the two data sets for our experiments can be found in Table I. In addition, we make use of data augmentation (translation transform, scale transform, and rotation transform) to extend the number of training samples (cf. Table I).

B. Experimental Analysis

In our detection task, there are two outputs, rotatable rectangular bounding boxes and categories (i.e., vehicle or not). In general, we evaluate the performance of detection methods by using different choices of intersection over union (IoU) which indicates the overlap ratio between a predicted box and its ground truth box. The IoU can be defined as

$$\text{IoU} = \left(S_{\text{bbox}} \bigcap S_{\text{gt}} \right) / \left(S_{\text{bbox}} \bigcup S_{\text{gt}} \right) \qquad (19)$$

where S_{bbox} and S_{gt} are the areas of the predicted box and the ground truth box, respectively, in the shape of regular rectangle. Therefore, we convert our predicted rotatable

rectangular bounding boxes and rotatable rectangular ground truth boxes into regular rectangular ones (i.e., minimum bounding rectangles) for the purpose of calculating IoUs.

Then, average precision (AP) and precision–recall curve are applied to evaluate object detection methods. Quantitatively, AP means the average value of precision for each object category from recall = 0 to recall = 1. For computing AP value, we define and count true positives (TPs), false positives (FPs), false negatives (FNs), and true negatives (TNs) in detection results.

For the vehicle detection task, we can regard the region of a regular rectangle bounding box as a TP in the case that the IoU is more than the given threshold value. Otherwise, if the IoU is less than the given threshold, the region is considered as an FP (also called false alarm). Moreover, the region of a target is regarded as an FN (also called miss alarm) if no predicted bounding box covers it. Otherwise, we regard the region as a TN (also called correct rejection). Consequently, we use the following definitional equations to formulate precision and recall indicators:

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

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \qquad (20)$$

In addition, we use F_1-score to evaluate the comprehensive performance of precision and recall, which can be calculated as follows:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \qquad (21)$$

1) Recall–IoU Analysis: Fig. 6 shows Recall–IoU comparisons of our method and other baseline methods. Although a lower IoU usually means more TPs and less FPs, it may give inaccurate locations of targets at the same time. Therefore, to obtain a better tradeoff between detection and location accuracies, we habitually set the IoU threshold to 0.5. Fig. 6 also displays recall–IoU trends of three different feature extraction networks (i.e., VGG-16, ResNet-101, and ResNet-101 with FPN). We can see that the proposed method significantly obtains better performance as compared to other networks, which indicates that the proposed network is capable of learning more robust feature representations for multioriented vehicle detection tasks. In addition, it can be seen that FPN can generate better localization results as it can embed low-level features into high-level ones.

2) Orientation Accuracy Analysis: In addition to evaluating localization accuracy, for multioriented vehicle detection tasks, we also need to assess the performance of the proposed R^3-Net in terms of vehicle orientation estimation. Thus, we make a statistic to show the probability of the deviation $\Delta\theta$ ($-90° < \Delta\theta \leq 90°$) between predicted angles and ground truth angles. In Fig. 7, we compare two anchor generation strategies, i.e., using the proposed BAR anchor and traditional anchor [39], for our method (using ResNet-101 with FPN as feature extraction network) on DLR 3K Munich Data set. It can be seen that applying BAR anchor to region proposal network can offer better estimations of vehicle orientations,
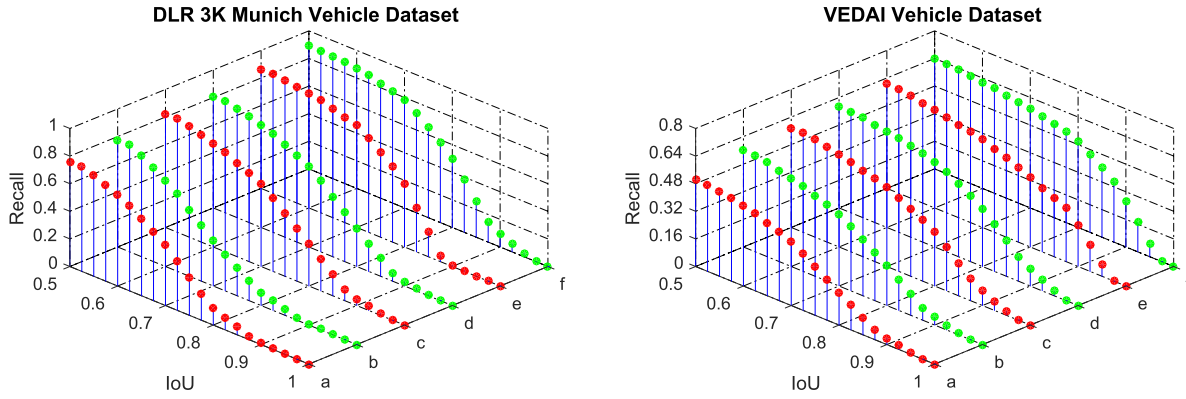
Fig. 6. Recall-IoU comparisons between the baseline method R-FCN and the proposed R³-Net. Key: a = R-FCN with VGG-16; b = R³-Net with VGG-16; c = R-FCN with ResNet-101; d = R³-Net with ResNet-101; e = R-FCN with ResNet-101 and FPN; and f = R³-Net with ResNet-101 and FPN.
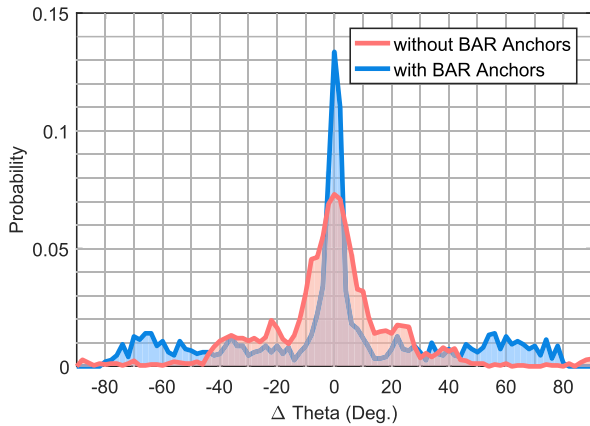

Fig. 7. Statistic of deviations of various vehicle angles on DLR 3K Munich Data set.

TABLE II
RECALLS OF DIFFERENT NUMBERS OF R-RPN AND R-DN PROPOSALS.
(R^3-NET WITH RESNET-101, IoU = 0.5)

Num. of R-RPN proposals	100	300	500	1000	2000
Num. of R-DN proposals	300	500	1000	2000	3000
DLR 3K Munich Dataset	0.605	0.753	0.809	0.812	0.816
VEDAI Dataset	0.426	0.522	0.586	0.589	0.592

which may attribute to the prior information of vehicle sizes for anchored areas.

3) Analysis of the Number of Proposals: As one of the important parameters of two-stage detection frameworks, the number of proposals always influences the tradeoff between detection accuracy and processing time. However, with the increase in the number of proposals, the detection accuracy cannot get continuous increase. Here, we test the recall rate with different numbers of R-RPN proposals and R-DN proposals in the proposed R³-Net (with ResNet-101) on DLR 3K Munich Data set and VEDAI Data set, respectively. We set the IoU value to 0.5. In Table II, we display the recall rates with different proposal number settings. We set the number of proposals in R-RPN and that in R-DN in the range of {100, 300, 500, 1000, 2000} and {300, 500, 1000, 2000, 3000}, respectively. It can be seen that for both DLR 3K Munich Data set and VEDAI Data set, when the number of proposals in R-RPN is more than 500 and that in R-DN is set more than 1000, respectively, the recall rate is nearly the same. We, therefore, use 500 and 1000 as the abovementioned numbers for our following experiments.

4) Loss Weight Analysis: When training our network with the joint loss, there are several important hyperparameters (i.e., η, λ_1, and λ_2) that control weights of all components of the loss function. We, therefore, conduct a series of experiments to seek an optimal combination of them. In our method, the loss weight η is to balance the weight of R-RPN and R-DN, and λ_1 and λ_2 are to balance the weight of classification tasks and regression tasks in R-RPN and R-DN, respectively. We first fix λ_1 and λ_2 ($\lambda_1 = \lambda_2 = 1$) and tweak η, and then assess the influence of λ_1 and λ_2 by fixing η. In our experiments, we use recall rate under IoU = 0.5 to see the performance of different parameter settings. In Table III, we show the trend of recall rate in relation to η. It can be seen that $\eta = 1$ is a turning point of recall rates, i.e., when η is smaller than 1, the recall rate increases, and it decreases when η is larger than 1. Hence, we set $\eta = 1$ for a good tradeoff between R-RPN loss and R-DN loss.

In addition, it is necessary to conduct experiments to find out the balance between λ_1 and λ_2, which can be tweaked against *gradient domination*. On the one hand, we set $\lambda_2 = 10$ and observe λ_1 in a range from 0.01 to 100 with seven values. According to the recall rates on DLR 3K Munich Data set and VEDAI Data set, we choose $\lambda_1 = 1$ for DLR 3K Munich Data set and $\lambda_1 = 10$ for VEDAI Data set. On the other hand, we set $\lambda_1 = 1$ and $\lambda_1 = 10$ for DLR 3K Munich Data set and VEDAI Data set, respectively, and then select λ_2 in the range of 0.01 to 100 with seven values. Finally, we find that $\lambda_2 = 10$ is optimal for both of two data sets. In Table III, we can see that there is no *gradient domination*, which can prove that the proposed method is robust. Fig. 8 gives some examples of the multioriented vehicle detection results of the proposed method on the two open data sets.

Fig. 8. Examples of multioriented vehicle detection of the proposed method. First two rows: test samples in DLR 3K Munich Data set. Last two rows: test samples of VEDAI Data set (two image modes: RGB and NIR). Best viewed zoomed in.

TABLE III
RECALLS WITH DIFFERENT LOSS WEIGHTS.
(R^3-NET WITH RESNET-101, IoU = 0.5)

η ($\lambda_1 = \lambda_2 = 1$)	0.01	0.1	1	10	100
DLR 3K Munich Dataset	0.730	0.764	0.798	0.792	0.753
VEDAI Dataset	0.508	0.520	0.565	0.537	0.528
λ_1 ($\eta = 1$, $\lambda_2 = 10$)	0.01	0.1	1	10	100
DLR 3K Munich Dataset	0.741	0.766	0.809	0.781	0.769
VEDAI Dataset	0.536	0.558	0.571	0.586	0.562
λ_2 ($\eta = 1$, $\lambda_1 = \{1, 10\}$)	0.01	0.1	1	10	100
DLR 3K Munich Dataset	0.733	0.778	0.798	0.809	0.790
VEDAI Dataset	0.528	0.545	0.569	0.586	0.575

C. Comparisons on Detection Task With Other Methods

We compare the proposed network (based on VGG-16, ResNet-101, and ResNet-101 with FPN) with two one-stage CNN-based object detection methods (i.e., SSD[1] [52] based on VGG-16 and YOLOv3[2] based on DarkNet-53 [53]), a two-stage CNN-based object detection method Faster R-

CNN[3] [39] based on VGG-16, and a baseline method R-FCN[4] [40] (based on VGG-16, ResNet-101, and ResNet-101 with FPN).

1) Parameter Settings: Before further comparisons between our method and other competitors, we set proper parameters for each method. The batch size of region proposal network-based methods (i.e., Faster R-CNN and R-FCN) and YOLOv3 is set to 64, and that of SSD is set to 8 due to the limitation of GPU memory. We apply 9 regular rectangular BAR anchors (i.e., two with the size of $\kappa\hat{w}_m \times \kappa\hat{h}_m$ and one with the size of $\kappa\hat{h}_m \times \kappa\hat{h}_m$ on each scale of κ, scale factor κ is set to $\{0.5, 1, 2\}$, $\hat{w}_m$ and $\hat{h}_m$ can be computed in (1), and $\hat{w}_m \leq \hat{h}_m$) to anchor-based methods for a fair comparison, which can cover the same region proposal area with 12 rotatable rectangular BAR anchors used in R^3-Net. Moreover, the region proposal network-based methods are implemented on the Caffe framework, and we keep top 2000 proposals produced by the region proposal network. Other parameter settings of those CNN-based methods can refer to the open source code.

2) Precision–Recall Curve and AP: We show precision–recall curves and APs of our method and other competitors on both DLR 3K Munich Data set and VEDAI Data set, respectively, in Fig. 9 and Table IV.

---

[1] https://github.com/weiliu89/caffe/tree/ssd
[2] https://github.com/pjreddie/darknet

[3] https://github.com/rbgirshick/py-faster-rcnn
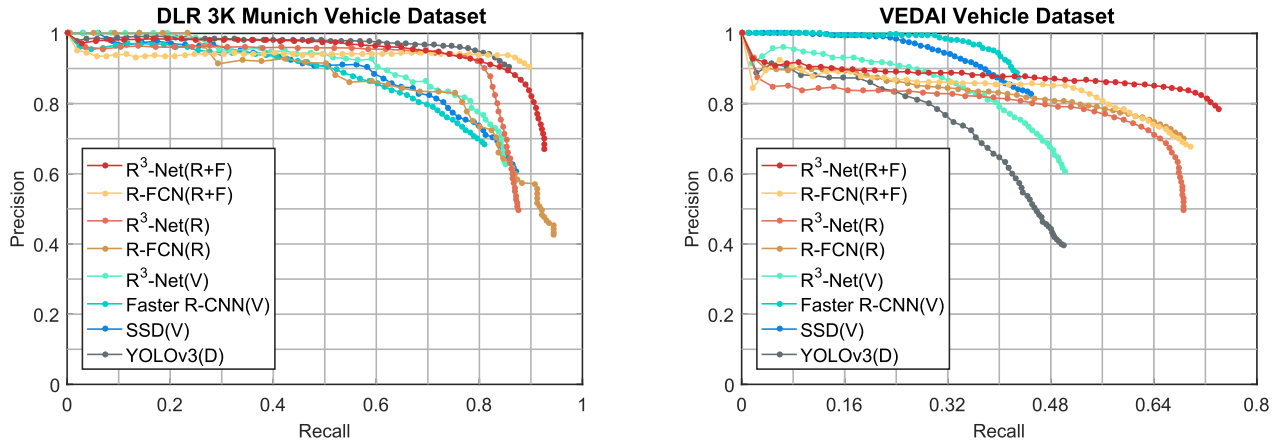[4] https://github.com/YuwenXiong/py-R-FCN

Fig. 9. Precision–recall comparisons between R^3-Net and other methods on (Left) DLR 3K Munich Data set and (Right) VEDAI Data set. (IoU = 0.6). Key: (D) = DarkNet-53; (V) = VGG-16; (R) = ResNet-101; (R + F) = ResNet-101 with FPN.

TABLE IV
AP (%) (IoU = 0.6). KEY: (D) = DARKNET-53; (V) = VGG-16; (R) = RESNET-101; (R + F) = RESNET-101 WITH FPN

Method	DLR 3K Munich Dataset	VEDAI Dataset
YOLOv3(D)	80.2%	41.6%
SSD(V)	74.7%	43.8%
Faster R-CNN(V)	73.4%	44.8%
R^3-Net(V)	74.2%	45.2%
R-FCN(R)	80.1%	53.2%
R^3-Net(R)	79.5%	53.4%
R-FCN(R+F)	85.9%	61.8%
R^3-Net(R+F)	**87.0%**	**64.8%**

TABLE V
F1-SCORE AND mAP (IoU = 0.5). KEY: (V) = VGG-16; (R) = RESNET-101; (R + F) = RESNET-101 WITH FPN

Method	DLR 3K Mun. (F1)	4-cls VEDAI (mAP)
Viola-Jones* [18]	0.61	−
Liu's* [18]	0.77	−
AVPN_basic* [25]	0.80	−
AVPN_large* [25]	0.82	−
Fast R-CNN(AVPN)* [25]	0.82	−
DPM* [51]	−	0.46
SVM+LTP* [51]	−	0.51
SVM+HOG31+LBP* [51]	−	0.50
R^3-Net(V), IoU = 0.5	0.83	0.47
R^3-Net(R), IoU = 0.5	0.85	0.56
R^3-Net(R+F), IoU = 0.5	**0.91**	**0.69**

On DLR 3K Munich Data set and VEDAI Data set, we set the recall threshold from 0 to 1 and 0 to 0.8, respectively, and show precision–recall curves of different methods. It can be seen that two-stage CNN-based detection methods are with higher accuracies than one-stage ones, which can be attributed to the fact that two-stage classifiers can obtain more accurate shots for classification tasks than one-stage ones.

On the other hand, from Table IV, in comparison with the baseline model, we can see that the proposed method fails to raise the AP performance too much and even drop down the AP when taking VGG-16 and ResNet-101 as feature extraction networks. However, the proposed method can promote the AP performance a lot by using features extracted by ResNet-101 together with FPN, which probably thanks to the efficient feature fusion mechanism in FPN, contributing to more precise localization for outputting bounding boxes in the shape of rotatable rectangles. In addition, it can be seen that deeper networks are able to offer better results with their stronger capabilities of nonlinear representation.

In Table V, we report F1-scores of several methods, including Viola–Jones detector [54], Liu's method [18], and accurate-vehicle-proposal-network (AVPN) method with different settings [25]. Moreover, we also report mean APs

(mAPs) of deformable part-based model (DPM) [55] and some traditional detectors which use hand-crafted features (e.g., LBP [15], HOG [13], and local ternary pattern (LTP) [56]) on four selected vehicle classes, i.e., car, pickup, truck, and van. The results show that the proposed method outperforms others.

D. Experiments on Aerial Videos

In addition to image data, we also test our method on two aerial videos (see Fig. 10), Parking Lot UAV Cruise Video and Busy Parking Lot UAV Surveillance Video [27]. The former is captured in low-altitude cruise mode, and the latter is acquired by a camera onboard a UAV hovering above the parking lot of Woburn Mall, Woburn, MA, USA.[5] We apply our model (R^3-Net with ResNet-101 and FPN trained on DLR 3K Munich Vehicle Data set) to Parking Lot UAV Cruise Video for detection task to assess the performance of the proposed model on video frames in a dynamic scenery and on Busy Parking Lot UAV Surveillance Video for detection

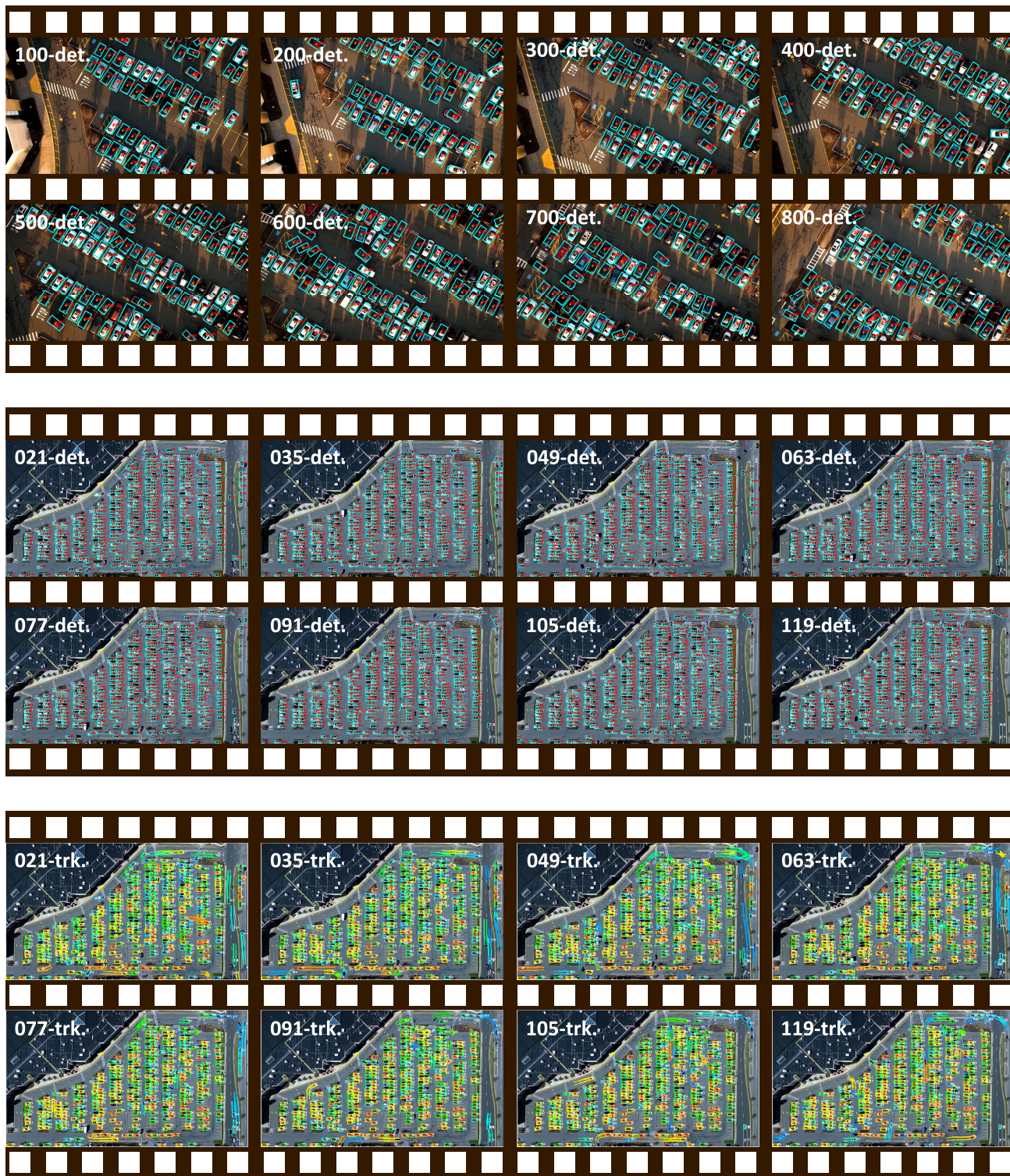[5]https://www.youtube.com/watch?v=yojapmOkIfg

Fig. 10.   Detection and tracking results on two video data sets. First two rows: detection result on UAV Parking Lot UAV Cruise Video; the first 2 rows show the detection result on Busy Parking Lot UAV Surveillance Video. Best viewed zoomed in. A part of tracking result is available at https://youtu.be/xCYD-tYudN0. Key: Det. = Detection; Trk. = Tracking.

as well as tracking to test the detector's performance in a crowed and complex scene from a relatively static surveillance view. Furthermore, we are curious to know if the temporal information of the second video data can strengthen our detection network's performance. The detection results are given in Table VI.

1) Detection on Parking Lot UAV Cruise Video: In order to qualitatively evaluate our model on this video, we manually labeled 20 ground truths for 20 frames and then compute average TPs, average FNs, average FPs, precision, and recall based on these ground truths. For simplicity, we use transferred regular rectangle boxes to compute IoU as mentioned before.

TABLE VI

DETECTION RESULTS ON TWO UAV VIDEOS. ON UAV VIDEO 1 (CRUISE MODE), WE UTILIZE R³-NET (WITH RESNET-101 AND FPN) AS DETECTION METHOD. ON UAV VIDEO 2 (SURVEILLANCE MODE), WE UTILIZE R³-NET (WITH RESNET-101 AND FPN) AS DETECTION METHOD TOGETHER WITH KF AS TRACKING METHOD. KEY: DET. = DETECTION; TRK. = TRACKING

Task	Mode	Frame size	Frames	a-FPS	a-TPs	a-FNs	a-FPs	Precision	Recall	Num. of proposals	Test speed
Det.	Cruise	1280×720	1080	24.0	67	6	0	100.0%	93.8%	1000 / 500	10.3 fps
Det.	Surveillance	1920×1080	1452	24.2	435	46	2	99.3%	90.4%	5000 / 3000	1.8 fps
Det. by trk.†	Surveillance	1920×1080	1452	24.2	439	42	0	100.0%	91.3%	5000 / 3000	1.6 fps

TABLE VII

COMPARISON WITH INSTANCE SEGMENTATION-BASED DETECTION METHODS ON FOUR LABELED FRAMES ON UAV VIDEO 2. KEY: (R) = RESNET-101; (R + F) = RESNET-101 WITH FPN

Method	Frame@1s				Frame@15s				Frame@30s				Frame@45s			
	F_1	Pre.	Rec.	Tim.	F_1	Pre.	Rec.	Tim.	F_1	Pre.	Rec.	Tim.	F_1	Pre.	Rec.	Tim.
B-Xception-FCN* [27]	91.4	89.7	**93.2**	2.59	90.2	86.8	**93.8**	2.76	90.1	87.7	92.7	2.81	90.4	87.6	93.2	2.74
B-ResFCN* [27]	93.3	95.2	91.5	1.54	92.6	91.5	93.6	1.67	93.6	94.0	**93.2**	1.92	93.1	94.3	91.8	1.77
Mask R-CNN(R+F)	95.3	**99.8**	91.2	**0.24**	95.6	99.6	91.9	**0.25**	95.0	99.6	90.8	**0.25**	95.0	99.3	91.1	**0.24**
R³-Net(R+F)	94.9	99.6	90.6	0.56	94.9	99.3	90.9	0.55	94.3	99.3	89.8	0.56	95.1	99.8	90.9	0.55
R³-Net(R+F)+KF†	**95.5**	**99.8**	91.6	0.63	**95.9**	**100.0**	92.3	0.63	**95.8**	**99.8**	92.0	0.63	**96.5**	**99.8**	**93.4**	0.62

The IoU threshold is set to 0.5. Table VI shows that the precision is 100%, and the recall is 93.8%, which indicates that our trained model has a high *generalization ability* on this video. In addition, the test speed with a frame size of 1280×720 is about 10.3 fps, which can nearly satisfy the requirement of a real-time vehicle detection task using UAV videos with a low-speed cruise mode. Here, the number of proposals in R-RPN and R-DN is set to 1000 and 500, respectively.

2) Detection and Tracking on Busy Parking Lot UAV Surveillance Video: We also test our model on this data with ten manually labeled frames. In this video, the main challenge for vehicle detection is that a great number of tiny vehicle appear densely. In Table VI, we can see that our trained model has satisfactory flexibility in such case with a precision of 99.3% and a recall of 90.4%. Here, we set the number of proposals in R-RPN and R-DN to 5000 and 3000, respectively, and the speed during test phase with a frame size of 1920×1080 is 1.8 fps, which can nearly meet the requirement of a high-altitude surveillance platform.

To facilitate our research, we try to perform multiple-object tracking task on Busy Parking Lot UAV Surveillance Video using the detection results produced by the proposed network. We exploit a simple online and real-time tracking algorithm for multiple-object tracking in video sequences,[6] which utilizes a Kalman filter (KF) to predict tracking boxes. From the result given in Table VI, it can be seen that the capability of the proposed detection network can be upgraded with a precision of 100% and a recall of 91.3%,[7] which is attributed to the fact

that the relevance of context information in multiple frames can offset the deviation of single-frame detection. The joint speed of detection and tracking during the test phase is about 1.6 fps. In Fig. 10, we show vehicle IDs, predicted boxes, and vehicle trajectories. A part of the tracking result is available at https://youtu.be/xCYD-tYudN0.

3) Comparison With Instance Segmentation-Based Detection Methods: We compare our method with several state-of-the-art instance segmentation-based detection methods, namely, B-Xception-FCN [27], B-ResFCN [27], and Mask R-CNN [57]. Here, B-Xception-FCN model and ResFCN model are trained on ISPRS Potsdam Semantic Labeling Data set [58], and Mask R-CNN model with ResNet-101 and FPN is trained on DLR 3K Munich Data set. The parameter setting of Mask R-CNN can refer to the open source code.[8] In Table VII, we show the comparison on precision, recall, and F_1-score. We find that the proposed models get lower recall than those instance segmentation-based methods, in general, however, they have better performance on precision and F_1-score, which shows the satisfactory flexibility of the proposed method on this video data.

4) Tradeoff Between Accuracy and Test Time: In Fig. 11, we evaluate the test time cost and AP of our method with different numbers of proposals on the two labeled UAV video data in order to find a good tradeoff between the accuracy and time cost. As a result, we set the number of proposals in R-RPN and R-DN to 1000 and 500 for Parking Lot UAV Cruise Video, and 5000 and 3000 for Busy Parking Lot UAV Surveillance Video, respectively.

---

[6]https://github.com/abewley/sort

[7]In this paper, we only discuss the result of indicators for the detection task rather than tracking task.

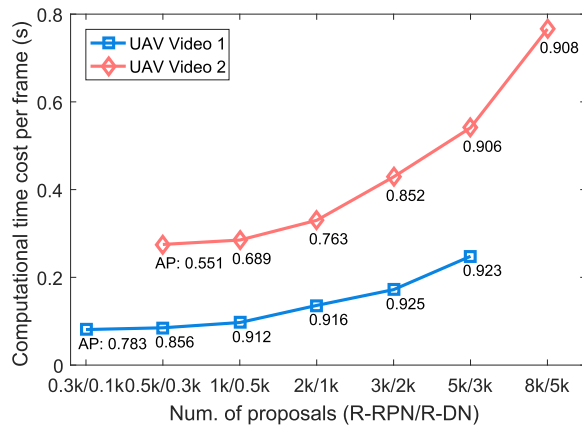[8]https://github.com/facebookresearch/Detectron

Fig. 11. Computational time cost per frame in two UAV videos with the change of the number of proposals. Key: AP = Average precision; UAV Video 1 = Parking Lot UAV Cruise Video; and UAV Video 2 = Busy Parking Lot UAV Surveillance Video.

IV. CONCLUSION

In this paper, a novel method is proposed to detect multi-oriented vehicles in aerial images and videos using a deep network call R^3-Net. First, one typical CNN is utilized to extract deep features. Second, we use R-RPN to generate R-RoIs encoded in 8-D vectors. A novel strategy called BAR anchor is applied to initialize templates of rotatable candidates. Third, we use R-DN as classifier and regressor to obtain the final 5-D rotatable detection boxes. Here, we propose a new downsampling method for R-RoIs called R-PS pooling to achieve fast dimensionality reduction on R-RoI feature maps and keep the information of positions and orientations. In addition, we modify the Shamos algorithm for the conversion of 5-D and 8-D detection boxes in R-RPN and R-DN. In our method, R-RPN and R-DN can be jointly trained for high efficiency. Then we evaluate the proposed method from two perspectives. On the one hand, we perform experiments on two open vehicle detection image data sets, i.e., DLR 3K Munich Data set and VEDAI Data set, to compare with other state-of-the-art detection methods. On the other hand, we conduct extra experiments on two aerial videos using models trained on DLR 3K Munich Data set. Experimental results show that the proposed R^3-Net outperforms other methods on both aerial images and aerial videos. Especially R^3-Net can be well combined with multiple-object tracking methods to acquire further information (e.g., vehicle trajectory), showing the satisfactory performance on multioriented vehicle detection tasks.

REFERENCES

[1] G.-S. Xia *et al.*, "AID: A benchmark data set for performance evaluation of aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3965–3981, Jul. 2017.

[2] B. Kellenberger, M. Volpi, and D. Tuia, "Fast animal detection in UAV images using convolutional neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2017, pp. 866–869.

[3] T. Moranduzzo, F. Melgani, M. L. Mekhalfi, Y. Bazi, and N. Alajlan, "Multiclass coarse analysis for UAV imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 12, pp. 6394–6406, Dec. 2015.

[4] N. Audebert, B. Le Saux, and S. Lefèvre, "Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks," *ISPRS J. Photogramm. Remote Sens.*, vol. 140, pp. 20–32, Jun. 2017.

[5] G.-S. Xia *et al.*, "DOTA: A large-scale dataset for object detection in aerial images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Feb. 2018, pp. 3974–3983.

[6] L. Mou and X. X. Zhu. (2018). "RiFCN: Recurrent network in fully convolutional network for semantic segmentation of high resolution remote sensing images." [Online]. Available: https://arxiv.org/abs/1805.02091

[7] F. Zhang, B. Du, L. Zhang, and M. Xu, "Weakly supervised learning based on coupled convolutional neural networks for aircraft detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 9, pp. 5553–5563, Sep. 2016.

[8] L. Mou and X. X. Zhu, "Spatiotemporal scene interpretation of space videos via deep neural network and tracklet analysis," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2016, pp. 1823–1826.

[9] L. Mou *et al.*, "Multitemporal very high resolution from space: Outcome of the 2016 IEEE GRSS data fusion contest," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 8, pp. 3435–3447, Aug. 2017.

[10] N. Audebert, B. Le Saux, and S. Lefèvre, "Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images," *Remote Sens.*, vol. 9, no. 4, p. 368, 2017.

[11] Q. Li, L. Mou, Q. Liu, Y. Wang, and X. X. Zhu, "HSF-Net: Multiscale deep feature embedding for ship detection in optical remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 12, pp. 7147–7161, Dec. 2018.

[12] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2. Sep. 1999, pp. 1150–1157.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2005, pp. 886–893.

[14] W. Shao, W. Yang, G. Liu, and J. Liu, "Car detection from high-resolution aerial imagery using multiple features," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2012, pp. 4379–4382.

[15] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[16] T. Moranduzzo and F. Melgani, "Automatic car counting method for unmanned aerial vehicle images," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 3, pp. 1635–1647, Mar. 2014.

[17] T. Moranduzzo and F. Melgani, "Detecting cars in UAV images with a catalog-based approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 10, pp. 6356–6367, Oct. 2014.

[18] K. Liu and G. Mattyus, "Fast multiclass vehicle detection on aerial images," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 9, pp. 1938–1942, Sep. 2015.

[19] M. ElMikaty and T. Stathaki, "Detection of cars in high-resolution aerial images of complex urban environments," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 10, pp. 5913–5924, Oct. 2017.

[20] B. Kalantar, S. B. Mansor, A. A. Halin, H. Z. M. Shafri, and M. Zand, "Multiple moving object detection from UAV videos using trajectories of matched regional adjacency graphs," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 9, pp. 5198–5213, Sep. 2017.

[21] H. Zhou, L. Wei, C. P. Lim, D. Creighton, and S. Nahavandi, "Robust vehicle detection in aerial images using bag-of-words and orientation aware scanning," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 12, pp. 7074–7085, Dec. 2018.

[22] X. X. Zhu *et al.*, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017.

[23] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, "Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 6, pp. 3325–3337, Jun. 2015.

[24] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7405–7415, Dec. 2016.

[25] Z. Deng, H. Sun, S. Zhou, J. Zhao, and H. Zou, "Toward fast and accurate vehicle detection in aerial images using coupled region-based convolutional neural networks," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 8, pp. 3652–3664, Aug. 2017.

[26] Q. Li, Y. Wang, Q. Liu, and W. Wang, "Hough transform guided deep feature extraction for dense building detection in remote sensing images," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2018, pp. 1872–1876.

[27] L. Mou and X. X. Zhu, "Vehicle instance segmentation from aerial image and video using a multitask learning residual fully convolutional network," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6699–6711, Nov. 2018.

[28] Q. Li, L. Mou, K. Jiang, Q. Liu, Y. Wang, and X. X. Zhu, "Hierarchical region based convolution neural network for multiscale object detection in remote sensing images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2018, pp. 4355–4358.

[29] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *ISPRS J. Photogramm. Remote Sens.*, vol. 117, pp. 11–28, Jul. 2016.

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

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

[32] V. Badrinarayanan, A. Handa, and R. Cipolla. (2015). "SegNet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling." [Online]. Available: https://arxiv.org/abs/1505.07293

[33] K. Li, G. Cheng, S. Bu, and X. You, "Rotation-insensitive and context-augmented object detection in remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 2337–2348, Apr. 2018.

[34] G. Cheng, J. Han, P. Zhou, and D. Xu, "Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection," *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 265–278, Jan. 2019.

[35] X. Yang *et al.*, "Automatic ship detection in remote sensing images from Google Earth of complex scenes based on multiscale rotation dense feature pyramid networks," *Remote Sens.*, vol. 10, no. 1, p. 132, 2018.

[36] L. Cao *et al.*, "Weakly supervised vehicle detection in satellite images via multi-instance discriminative learning," *Pattern Recognit.*, vol. 64, pp. 417–424, Apr. 2016.

[37] L. Cao, Q. Jiang, M. Cheng, and C. Wang, "Robust vehicle detection by combining deep features with exemplar classification," *Neurocomputing*, vol. 215, pp. 225–231, Nov. 2016.

[38] Y. Xu, G. Yu, X. Wu, Y. Wang, and Y. Ma, "An enhanced Viola–Jones vehicle detection method from unmanned aerial vehicles imagery," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1845–1856, Jul. 2017.

[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[40] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[43] T.-Y. Lin and P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 936–944.

[44] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[45] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep. 2015, pp. 1440–1448.

[46] M. I. Shamos, "Computational geometry," Ph. D. dissertation, Dept. Comput. Sci., Yale Univ., New Haven, CT, USA, May 1978.

[47] M. E. Houle and G. T. Toussaint, "Computing the width of a set," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-10, no. 5, pp. 761–765, Sep. 1988.

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

[49] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[50] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[51] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery: A small target detection benchmark," *J. Vis. Commun. Image Represent.*, vol. 34, pp. 187–203, Jan. 2016.

[52] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[53] J. Redmon and A. Farhadi. (2018). "YOLOv3: An incremental improvement." [Online]. Available: https://arxiv.org/abs/1804.02767

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

[55] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[56] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1635–1650, Jun. 2010.

[57] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *Proc. ICCV*. Washington, DC, USA: IEEE Computer Society, 2017, pp. 2980–2988.

[58] F. Rottensteiner *et al.*, "The ISPRS benchmark on urban object classification and 3D building reconstruction," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 1, no. 3, pp. 293–298, 2012.

Qingpeng Li (S'18) received the bachelor's degree in mechanical engineering and automation from Beijing Jiaotong University, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with Beihang University, Beijing.

From 2015 to 2016, he was with the Digital Precise Forming Group, Beihang University, Beijing, where he was involved in the research of machine vision. His research interests include computer vision and machine/deep learning applications in remote sensing, especially object detection in remote sensing images and videos.
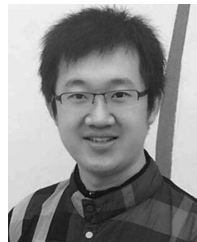
Lichao Mou (S'16) received the bachelor's degree in automation from the Xi'an University of Posts and Telecommunications, Xi'an, China, in 2012, and the master's degree in signal and information processing from the University of Chinese Academy of Sciences, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the German Aerospace Center, Wessling, Germany, and the Technical University of Munich, Munich, Germany.

In 2015, he was with the Computer Vision Group, University of Freiburg, Freiburg im Breisgau, Germany. His research interests include remote sensing, computer vision, and machine/deep learning, especially remote sensing video analysis and deep networks with their applications in remote sensing.

Qizhi Xu (M'16) received the B.S. degree from Jiangxi Normal University, Nanchang, China, in 2005, and the Ph.D. degree from Beihang University, Beijing, China, in 2012.

He was a Post-Doctoral Fellow with the University of New Brunswick, Fredericton, NB, Canada. He is currently an Associate Professor of the School of Computer Science and Engineering, Beihang University, under the support of the Foreign Young Talent Program of Beihang University. His research interests include image fusion, image understanding, and big data analysis of remote sensing.

Dr. Xu was a recipient of the Technological Invention Award First Prize from the Chinese Institute of Electronics for his image fusion research in 2017.

Yun Zhang (M'02) received the B.Sc. degree from Wuhan University, Wuhan, China, in 1982, the M.S. degree from East China Normal University, Shanghai, China, in 1989, and the Ph.D. degree from the Free University of Berlin, Berlin, Germany, in 1997.

He is currently a Professor with the University of New Brunswick, Fredericton, NB, Canada, and a Visiting Professor with the Massachusetts Institute of Technology, Cambridge, MA, USA. He has authored or co-authored over 200 research papers. He has invented eight patented technologies (three with his students) and six patent-pending technologies (two with his students). The research outcomes from his laboratory have resulted in more than ten commercial technology licenses. Users of the technologies include NASA, Google, Natural Resources of Canada, DigitalGlobe, and many others.

Dr. Zhang is a Fellow of the Canadian Academy of Engineering and a Canada Research Chair in Advanced Geomatics Image Processing. He was a recipient of many prestigious national and international research awards.

Xiao Xiang Zhu (S'10–M'12–SM'14) received the M.Sc., Dr.Ing., and Habilitation degrees in signal processing from the Technical University of Munich (TUM), Munich, Germany, in 2008, 2011, and 2013, respectively.

She is currently a Professor of signal processing in earth observation with TUM and the German Aerospace Center (DLR), Wessling, Germany; the Head of the Department of Earth Observation Data Science, Earth Observation Center, DLR; and also the Head of the Helmholtz Young Investigator Group, Signal Processing in Earth Observation, DLR, and TUM. Her research interests include remote sensing and Earth observation, signal processing, machine learning, and data science, with a special application focus on global urban mapping.

Dr. Zhu is an Associate Editor of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.