RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving

Yiming Zeng, Yu Hu, Shice Liu, Jing Ye, Yinhe Han, Xiaowei Li, and Ninghui Sun

***Abstract*—For autonomous driving, vehicle detection is the prerequisite for many tasks like collision avoidance and path planning. In this letter, we present a real-time three-dimensional (RT3D) vehicle detection method that utilizes pure LiDAR point cloud to predict the location, orientation, and size of vehicles. In contrast to previous 3-D object detection methods, we used a pre-RoIpooling convolution technique that moves a majority of the convolution operations to ahead of the RoI pooling, leaving just a small part behind, so that significantly boosts the computation efficiency. We also propose a pose-sensitive feature map design which can be strongly activated by the relative poses of vehicles, leading to a high regression accuracy on the location, orientation, and size of vehicles. Experiments on the KITTI benchmark dataset show that the RT3D is not only able to deliver competitive detection accuracy against state-of-the-art methods, but also the first LiDAR-based 3-D vehicle detection work that completes detection within 0.09 s which is even shorter than the scan period of mainstream LiDAR sensors.**

***Index Terms*—Object detection, segmentation and categorization, autonomous agents.**

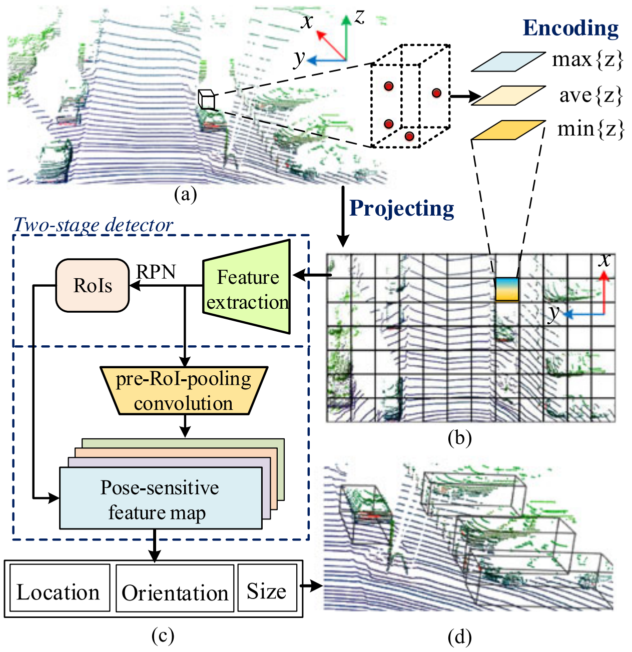

Fig. 1. The pipeline of RT3D: **(a) a 3-D point cloud of LiDAR** is projected onto **(b) a depth map in bird's-eye view** that is encoded with height information of points. Then **(c) a CNN-based two-stage detector** is utilized to: 1) generate region proposals with a Region Proposal Network (RPN) [2], 2) classify the proposed regions with pre-RoI-pooling convolutions on pose-sensitive feature maps. **(d) Result visualization** shows the detected vehicles with the orientated 3-D bounding boxes.

I. Introduction

VEHICLE detection requires the autonomous ego vehicle to detect nearby obstacle vehicles quickly and localize them accurately so that the ego vehicle correctly control its speed and direction, as well as plan a preferable path. However, real-time 3D vehicle detection remains challenging, especially when driving at high speed on urban roads. Currently, a car-equipped LiDAR sensor typically scans at 10 frames per second [1], capturing approximately one million 3D points per second. In other words, there is only 100 ms processing time for each frame of LiDAR point cloud data. LiDAR-based 3D object detection methods usually consist of two stages, i.e., a region proposal stage and a classification stage. In the region proposal stage, 3D candidate regions in point cloud space are generated, then in the subsequent stage, each candidate region are classified and generate a 3D bounding-box. However, these detections in 3D point clouds are time-consuming conventionally due to conduct classification and regression for thousands of candidate regions.

In this letter, we present a real-time vehicle detection method RT3D. The pipeline of RT3D is shown in Fig. 1. Firstly, to meet the input format requirement of the convolution neural network, we project the discrete 3D point cloud into a 2D grid representation. Secondly, we encode the height information of point data in the 2D grid for further 3D detection. Thirdly, a CNN-based two-stage detector generates region proposals with a RPN and then conduct classification and location regression for each Region of Interest (RoI) with pose-sensitive feature maps. To accelerate the classification stage, inspired by [3], we use pre-RoI-pooling convolution, which means a majority of convolution operations in classification stage are moved from behind of the RoI pooling to ahead of the RoI pooling. Hence, these time-consuming operations are shared and are conducted only once for all RoIs instead of being conducted in a RoI-wise manner at the conventional classification stage.

Manuscript received February 24, 2018; accepted June 17, 2018. Date of publication July 4, 2018; date of current version August 2, 2018. This letter was recommended for publication by Associate Editor S. Behnke and Editor T. Asfour upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grants 61274030 and 61532017. *(Corresponding author: Yu Hu.)*

The authors are with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: zengyiming@ict.ac.cn; huyu@ict.ac.cn; liushice@ict.ac.cn; yejing@ict.ac.cn; yinhes@ict.ac.cn; lxw@ict.ac.cn; snh@ict.ac.cn).

Digital Object Identifier 10.1109/LRA.2018.2852843

Benefited from the pre-RoI-pooling convolution and pose-sensitive feature map techniques, the RT3D achieves comparable detection accuracy against the state-of-the-art methods on the KITTI benchmark dataset [1]. Moreover, to the best of our knowledge, the RT3D is the first LiDAR-based 3D vehicle detection work that completes detection within 0.09 seconds and is even shorter than the scan period of mainstream LiDAR sensors.

II. Related Work

Convolutional neural networks improve object detection significantly. In Fast-RCNN [2], a Region Proposal Network (RPN) was proposed to generate high-quality candidate regions. Then Faster-RCNN [4] becomes a de facto pipeline of CNN-based object detection. Afterwards, 2D object detector [3] was proposed to improve the detection accuracy, and [5], [6] were proposed to accelerate the detection speed. Recently, researchers conducted 3D object detection on RGB images. Chabot *et al*. [7] proposed a multi-task neural network to create coarse-to-fine object proposals and to estimate vehicle orientation and location through 2D/3D point matching. Chen *et al*. [8] exploited stereo images to place proposals in the form of 3D bounding boxes. The authors also extended their work to monocular image [9] by projecting candidate 3D boxes to the image plane and scoring boxes via prior knowledge.

Besides RGB images, deep learning based techniques also conducted 3D object detection in point clouds. Qi *et al*. [10] proposed a PointNet that directly took point cloud as input. Li [11] designed a 3D fully convolution neural network to generate region proposals for 3D object detection. Engelcke *et al*. [12] proposed a sparse convolution layer to classify region proposals generated by a sliding window. Zhou *et al*. [13] divided the point cloud into 3D voxels, aggregated local voxel features through voxel feature encoding (VFE) layers, and transformed point clouds into a high-dimensional volumetric representation.

Besides directly operating in the 3D space, some other detection approaches projected the 3D point cloud onto a 2D plane. [9], [14], [15] constructed a compact 2D representation of 3D point cloud from bird's-eye view, while [16] placed 3D windows on a front-view range map for vehicle detection. Ref. [17] and [18] generated 3D region proposals from a stack of height maps and conducted classification and location estimation via fusing features extracted from LiDAR point clouds and RGB images. However, these methods are time-consuming, which limits the practical usage of these methods in autonomous driving.

III. Proposed Approach

As illustrated in Fig. 1, the discrete 3D point cloud $P = \{p_i = (x, y, z),\ i = 1, 2, \ldots\}$ is projected onto a 2D grid $G_{N\times N}$ on the $xy\text{-}plane$ with height information embedded. The 2D grid representation is then fed into a convolution neural network to estimate the center location, orientation, and size of a vehicle.

A. 2D Grid Representation of 3D Point Cloud

Firstly, the 3D point cloud is projected onto a 2D grid in the birds-eye view. As shown in Fig. 1, from (a) to (b), we project 3D point cloud $P = \{p_i = (x, y, z),\ i = 1, 2, \ldots\}$ to the ground ($xy\text{-}plane$), where x, y, z is the location of point p. The point cloud is projected onto a ground-plane grid with resolution 0.05 × 0.05 m, where each grid cell records $(min(z), ave(z), max(z))$ of relevant projected points. Grid cells with no point clouds are assigned a triple of $(0, 0, 0)$. This 2D grid representation of LiDAR points fits for a convolutional neural network that has three input channels.

B. Vehicle Detection Network

The RT3D network contains a region proposal subnetwork (RPN) and a classification subnetwork, as shown in Fig. 2. Features are extracted from the 2D grid by a convolutional neural network derived from the ResNet-50 [20]. These feature maps are shared by the RPN and the classification subnetworks.

1) The RPN Subnetwork: We adopt the sliding window and anchor technique proposed in the Faster-RCNN [4] to generate region proposals on the feature map. Observing that the variation of vehicle size is relatively small compared to the size variation among different object categories, we simplify the pyramid of multi-scale anchors in the Faster-RCNN to a single-scale anchor. The region proposals $(\Delta x, \Delta y, w, l, \theta)$ are generated on the $xy\text{-}plane$, where $(\Delta x, \Delta y)$ represents the offset of the vehicle's center on the $xy\text{-}plane$, and (w, l, θ) represents the width, length, and yaw of the vehicle. As to the RPN loss, we calculate confidence scores and Smooth L1 loss [4] for regression of region proposals. During training, we calculate the Intersection-over-Union (IoU) between bounding-boxes of orientated RoIs $(\Delta x, \Delta y, w, l, \theta)$ and the bounding-boxes of ground truth $(\Delta x^*, \Delta y^*, w^*, l^*, \theta^*)$. A sample is considered to be positive if IoU ≥ 0.7, negative if IoU ≤ 0.5, and otherwise ignored. We apply non-maximum suppression (NMS) to remove redundant RoIs.

2) The Classification Subnetwork: Since RoI pooling is the bridge between the region proposal stage and the classification stage in the CNN-based two-stage detection pipeline, convolutions after the RoI pooling are responsible for classifying RoIs in a RoI-wise manner. Therefore, for hundreds of RoIs, these convolutions are conducted hundreds of time. If we move a majority of convolutions to ahead of the RoI pooling, then these pre-RoI-pooling convolutions will be conducted once for all RoIs, which will save much time. However, such speed boost is at the cost of detection quality. As only a few convolution layers are preserved in the classification subnetwork, the classification accuracy will decrease and the location regression will suffer from translation-invariance [3].

Inspired by the position-sensitive score map idea in [3], we construct a group of pose-sensitive feature maps to mitigate the adverse effect raised by the pre-RoI-pooling convolutions. As LiDAR points are unevenly distributed in space, different parts of the car contain different amounts of valid point cloud data, which leads to different estimation probabilities on vehicle location, size, and orientation. For instance, if the car is in the left front of the LiDAR, then LiDAR point data mostly concentrate on the right side of the obstacle car. So we firstly divide the car into k^2 parts, using each part of the car to estimate

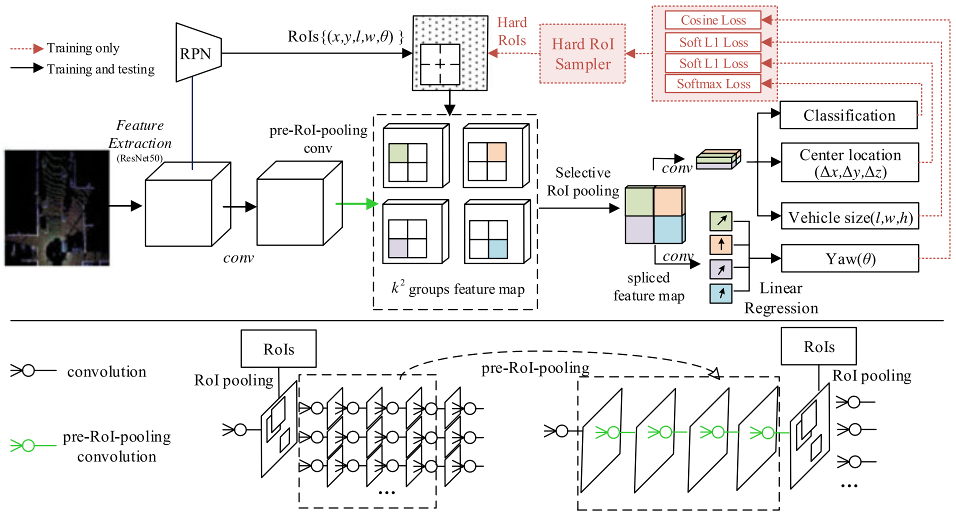

Fig. 2. The network architecture of RT3D. It contains a region proposal subnetwork (RPN) and a classification subnetwork. We apply pre-RoI-pooling convolutions before the RoI pooling operation and construct pose-sensitive feature maps to enhance classification and regression. Besides, online hard example mining (OHEM) [19] is employed to train the classifier. The bottom left subfigure shows the original RoI pooling and convolutions in the Faster-RCNN. The bottom right subfigure shows the RoI pooling and convolutions in the RT3D. The pre-RoI-pooling convolutions run only once for all RoIs, while a few convolution operations after the RoI pooling are preserved, which run for every RoI.

position, contour, and orientation. Then we generate the final results based on the estimations from all parts. Thus, we construct k^2 groups of feature maps correspondingly to k^2 parts of a car, which are called pose-sensitive feature maps, as shown in Fig. 2.

Meanwhile, each RoI is also divided into k^2 regions. Taking $k = 2$ as an example, four groups of feature maps are constructed correspondingly to the upper-left, upper-right, bottom-left and bottom-right parts of the car. Then the selective RoI pooling will sample the upper-left RoI region from the feature maps which are corresponding to the upper-left part of the car. The remaining three RoI regions will be sampled in a similar way. Afterward, features from different parts are merged into one "spliced feature map" for classification and estimation of location, size, and orientation. In fact, the selective RoI pooling is a more general expression of RoI pooling. As shown in Fig. 2, selective RoI pooling divides a RoI into k^2 parts and samples features from k^2 groups of feature maps. When $k = 1$, selective RoI pooling reduces to RoI pooling. A RoI is divided into k^2 regions $\{R_{i,j} | 1 \leq i \leq k, 1 \leq j \leq k\}$, for a RoI of a size $h \times w$, a region is of a size $\frac{h}{k} \times \frac{w}{k}$. Let (x, y) denote the offset to the top-left corner to a RoI, selective RoI pooling is defined as $S(x, y) = I_{R_{i,j}}(x, y) F_{i \times k + j}(x, y)$, where $i = \left\lfloor \frac{xk}{h} \right\rfloor, j = \left\lfloor \frac{yk}{w} \right\rfloor$ and $I_{R_{i,j}} = 1, if (x, y) \in R_{i,j}\ else\ 0$. The $F_{i \times k + j}$ term means the $(i \times k + j)$th group of feature maps.

The spliced feature map is used for classifying a vehicle and for regression to get the center and size $(\Delta x, \Delta y, \Delta z, w, l, h)$ of the vehicle, where $\Delta x, \Delta y, \Delta z$ is the offset of the vehicle center in the 3D point cloud space. The parts of feature maps from different groups, which are denoted with short arrowed lines, are used for calculating the directions $\theta_i, i = 1, \ldots, k^2$, of different locations, respectively. Finally, a linear regression is deployed to generate the yaw $\theta = \sum_{i=1}^{k^2} \alpha_i \theta_i + \beta_i$ of the vehicle.

3) Training: The loss defined on each RoI is the summation of the classification loss, the bounding box regression loss, and the cosine loss of the yaw angle:

$$L(s, p_{\Delta x, \Delta y, \Delta z, w, l, h}, \theta) = L_{cls}(s_{c*}) + \lambda_1 1_{[c* \neq 0]} L_{reg}(p, p^*) \\ + \lambda_2 1_{[c* \neq 0]} L_{cos}(\theta, \theta^*)$$

where $L_{cls}(s_{c*}) = -\log(s_{c*})$, $L_{reg}(p, p^*)$ is Smooth L1 loss defined in [2]; $L_{cos}(\theta, \theta^*) = (\cos\theta - \cos\theta^*)^2$. $1_{[c* \neq 0]}$ equals 1 if $c^* \neq 0$, otherwise equals 0; and $c^* = 0$ indicates the background.

Two issues caused by the sparsity of the point cloud need to be addressed: 1) during sliding window on feature maps, there are many anchors which have no data. To reduce computation, we delete these empty anchors; 2) many region proposals contain no vehicles or many region proposals contain simple examples, so online hard example mining (OHEM) [19] is adopted to automatically select hard examples to make training more effective and efficient. Besides, a vehicle usually occupies about 32×80 grids. The size will be reduced by 32 times in the original ResNet-50 feature map, which is too small for RoI pooling. To enlarge the feature map, we modify the parameter $stride = 2$ in the convolution layers on the conv5 "residual block" to $stride = 1$, and we apply the "hole" algorithm [21] to all following convolution filters on the conv5 "residual block." In this way, the size of the generated feature map is doubled.

Our network can be trained end-to-end. For each mini-batch, we use one image and sample 128 ROIs for back-propagation. We select 25% of the ROIs as positive to balance the distribution of examples, which helps to train a more accurate classifier [22]. We initialize the RT3D network with pre-trained ResNet-50 model [20] and use a weight decay of 0.0005 and a momentum of 0.9. Then we train the network by using SGD with a learning rate of 0.001 for 80K iterations and 0.0001 for the following 20K iterations.

TABLE I
3D LOCALIZATION PERFORMANCE: AVERAGE PRECISION (AP_{loc}[%])

Method	Data	Time (s)	Experimental Platform	IoU=0.5			IoU=0.7		
				Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D[17]	Mono	4.2	N/A	30.5	22.39	19.16	5.22	5.19	4.13
3DOP[8]	Stereo	3	Titan X	55.04	41.25	34.55	12.63	9.49	7.59
VeloFCN[16]	LiDAR	1	N/A	79.68	63.82	62.80	40.14	32.08	30.47
DOBEM[14]	LiDAR	0.6	Titan X	79.28	80.17	80.05	54.92	60.07	60.89
MV3D[9]	LiDAR+Mono	0.36	Titan X	96.34	89.39	88.67	86.55	78.10	76.67
VoxelNet[13]	LiDAR	0.225	Titan X	N/A	N/A	N/A	89.60	84.81	78.57
RT3D	LiDAR	0.089	Titan X	89.35	80.89	81.15	88.29	79.87	80.42

TABLE II
3D DETECTION PERFORMANCE: AVERAGE PRECISION (AP_{3D} [%])

Method	Data	Time (s)	IoU=0.25			IoU=0.5			IoU=0.7		
			Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D[17]	Mono	4.2	62.94	48.2	42.68	25.19	18.2	15.52	2.53	2.31	2.31
3DOP[8]	Stereo	3	85.49	38.82	64.09	46.04	34.63	30.09	6.55	5.07	4.1
VeloFCN[16]	LiDAR	1	89.04	81.06	75.93	67.92	57.57	52.56	15.2	13.66	15.98
MV3D[9]	LiDAR+Mono	0.36	96.52	89.56	88.94	96.02	89.06	88.38	71.29	62.68	56.56
VoxelNet[13]	LiDAR	0.225	N/A	N/A	N/A	N/A	N/A	N/A	81.97	65.46	62.85
RT3D	LiDAR	0.089	89.48	81.01	81.24	89.04	80.56	80.91	72.85	61.64	64.38

4) Testing: As shown in Fig. 2, the 2D grid, generated by projection and encoding, is fed into the two-stage detector. At the region proposal stage, the feature maps shared with RPN and classification network are computed. Then the RPN proposes thousands of RoIs and only 300 RoIs on the xy-$plane$ will be retained by the NMS operation with IoU threshold of 0.7. At the classification stage, the pose-sensitive feature maps are generated and the vehicles' position, yaw, and size are regressed from individual RoI. The results are projected onto the xy-$plane$ and then are filtered by NMS using a threshold of 0.1 IoU to ensure that vehicles do not occupy the same 2D grid.

IV. EXPERIMENTS

We evaluate the proposed method on the public dataset KITTI [1] for autonomous driving. This dataset is composed of 7481 training images and 7518 testing images. Since ground truth annotations for the testing set are not released, following [8], we divide the KITTI training set into a *training set* and a *validation set*. Note that the samples in the training set and verification set are exactly the same as [8]. We implement our method based on Caffe [23] and measure the runtime on the computer with an NVIDIA Titan X GPU.

We conduct the evaluation on the *validation set* and present our results for three levels of difficulty (Easy, Moderate and Hard) according to different occlusions and truncations, as proposed by the KITTI. Evaluation on the test dataset is also conducted.

A. 3D Localization

3D localization evaluation is conducted in the KITTI bird's-eye view benchmark. Both the accuracy and the runtime are evaluated. In the road scene, vehicles are driving on the ground plane. Evaluation criterion uses a bounding box overlap on the ground (xy-$plane$), i.e., the information on the height axis is ignored. The 2D occupancy grid is the most popular map to represent the environment and to plan paths. To evaluate the accuracy, we use Average Precision (AP_{loc}) for the bounding box on the ground. Meanwhile, we compare the runtime of the different methods because detecting obstacles in real-time is very important to vehicle safety. Table I presents results on 3D localization on the *validation set*. Compared to the previous works, our RT3D method has the fastest detection speed which is 2.5 times faster than the existing best work VoxelNet [13]. The detection time of RT3D is within 0.1 seconds, which allows the RT3D to be deployed in real-time systems. In terms of accuracy, when in the strictest constraint IoU $= 0.7$, our method does have some loss of accuracy compared to MV3D [9] and VoxelNet [13]. But compared to VeloFCN [16], DOBEM [14], Mono3D [17], and 3DOP [8], our method still achieves at least 13% higher accuracy.

B. 3D Detection

The 3D detection is evaluated on the KITTI 3D object detection benchmark. Table II shows the runtime and the average precision (AP_{3D}) on the *validation set*. As 3D IoU thresholds are rather strict for image-based methods, the IoU $= 0.25$ case is considered in the 3D detection. We visualize some 3D detection results in Fig. 3.

The comparisons of RT3D and the pure LiDAR-based methods on the KITTI online benchmark (the test dataset) are shown in Table III. The performance of RT3D on the test dataset is not as good as on the validation dataset. We analyze some failure cases in Fig. 4. The first and the second case show inaccurate predictions of position and orientation. The third case shows a missed detection due to too few point cloud data. The fourth

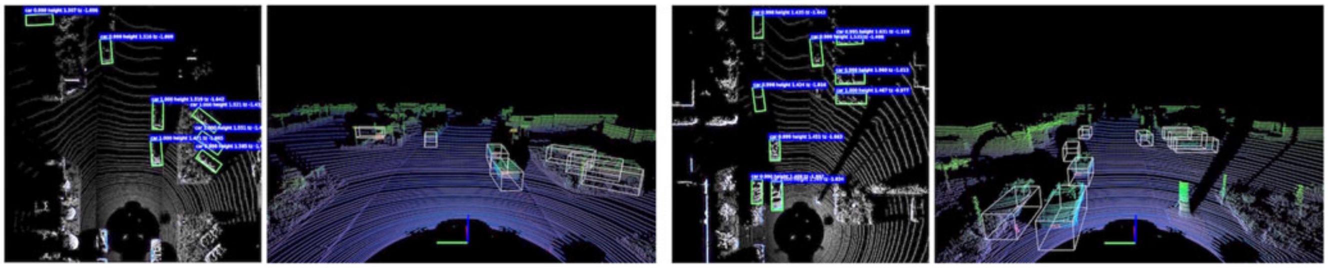

Fig. 3. Result Visualization. This example shows detection results of four different scenes. In each scene, the result of the 3-D localization on the $xy\text{-}plane$ is presented on the left, and the 3-D detection in point cloud is presented on the right.

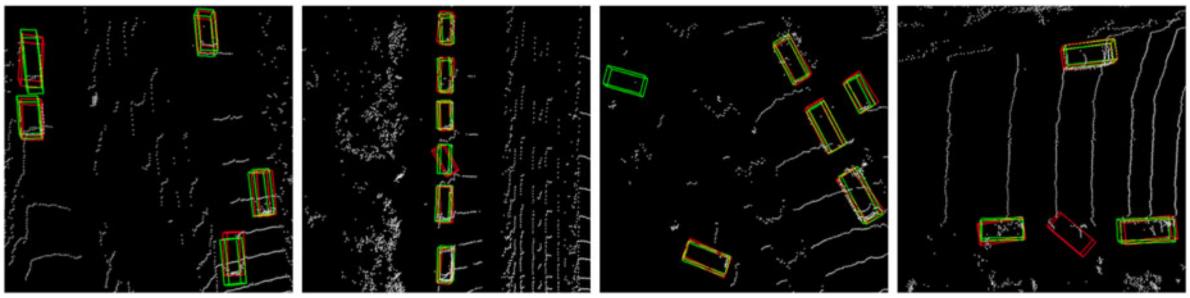

Fig. 4. Failure cases. Ground-truth are shown in green and our results are shown in red.

TABLE III
RESULTS ON KITTI ONLINE TEST DATASET

Methods	AP_{3D}			AP_{loc}			Times(s)
	Easy	Moderate	Hard	Easy	Moderate	Hard	
Voxelnet[13]	77.47	65.11	57.73	89.35	79.26	77.39	0.225
MV3D(Lidar)[17]	66.77	52.73	51.31	85.82	77.00	68.94	0.36
RT3D	23.49	21.27	19.81	54.68	42.10	44.05	0.09
BirdNet[24]	14.75	13.44	12.04	75.52	50.81	50.00	0.11
DoBEM[14]	7.42	6.95	13.45	36.49	36.95	38.10	0.60

case shows a false detection as the point cloud is similar to a cluttered vehicle point cloud. Besides, the point cloud becomes very sparse and the features of the car in the distance are hard to be preserved. To solve these problems, an alternative way is fusing sparse point cloud and dense RGB images. Besides, most labeled samples in the training dataset are located near the inspection vehicle, which negates the generalization of detecting vehicles in various ranges.

TABLE IV
VEHICLES CENTER DEVIATION

Methods	center (m)	X (m)	Y (m)	Z (m)
Mono3D[17]	0.9449	0.8587	0.3933	0.1374
3DOP[8]	0.7495	0.6627	0.2935	0.0911
MV3D[9]	0.2104	0.1362	0.0975	0.0645
RT3D	0.1865	0.0959	0.0902	0.0913

C. Center Deviation

To more intuitively describe the vehicle position, we analyze deviations between the predicted centers C and the ground truth of vehicle centers C^*. The center deviation $|C - C^*|$ is the distance of the predicted vehicle center C and the ground truth vehicle center C^*.

Table IV shows the average center deviation $ave(|C - C^*|)$ in the 3D point cloud space, $ave(|C_x - C_x^*|)$ in the X-coordinate, $ave(|C_y - C_y^*|)$ in the Y-coordinate, and $ave(|C_z - C_z^*|)$ in the Z-coordinate. We list the results of Mono3D [17], 3DOP [8], and MV3D [9], because their results are public, while the results of VoxelNet [13] are unavailable. In comparison to Mono3D [17], 3DOP [8], and MV3D [9], the proposed method achieves the least average deviation. However, in the direction of the Z-axis, we have some loss of accuracy because height information is not fully preserved.

Fig. 5 shows the distributions of the center deviations and the cumulative distribution. We can see that the center deviation of the RT3D is mainly concentrated in the 0.05–0.25 m range. The cumulative distribution graph shows that 87% deviation of the detected vehicles is within 0.3 m, and the ratio increases to 97.8% when the deviation is within 0.4 m, superior to the other three methods.

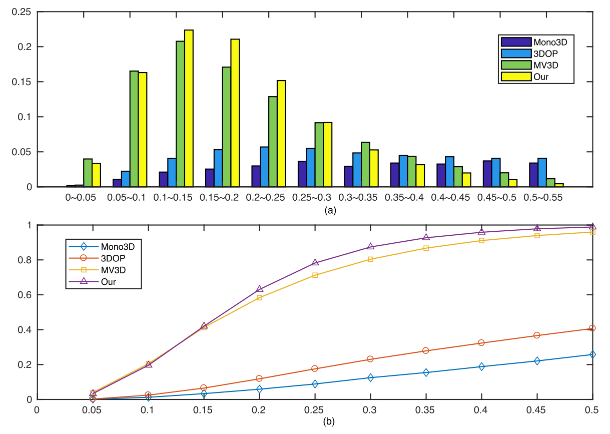

Fig. 5. (a) Distribution of vehicle center deviations. (b) Cumulative distribution of vehicle center deviations.

TABLE V
THE QUANTITATIVE EFFECT OF PRE-ROI-POOLING CONV

model	AP_{3d}		
	Easy	Moderate	Hard
Naive	59.05	49.19	43.01
Baseline	63.43	51.56	45.09

TABLE VI
AP_{3D} WITH DIFFERENT k (IOU $= 0.5$)

	AP_{3d}			AP_{bev}		
	Easy	Moderate	Hard	Easy	Moderate	Hard
k=1	63.43	51.56	45.09	79.90	66.04	65.19
k=2	74.95	63.06	58.66	84.31	75.98	70.28
k=3	71.17	61.31	59.42	82.41	74.73	71.62

D. Effect of Pre-RoI-pooling Convolution

The quantitative effect of pre-RoI-pooling convolution is presented in Table V. The baseline model has no pre-RoI-pooling convolution and no pose-sensitive feature maps, while the naive model has pre-RoI-pooling convolution and also has no pose-sensitive feature maps. Compared to the baseline, the AP_{3D} of the naive model is lower due to the weak classification subnetwork.

E. Effect of Parameter k

The parameter k represents the number of divisions of a RoI. To analyze the effect of different divisions of RoI, we present the AP_{3D} with different k with IoU $= 0.5$ in Table VI. When $k = 1$, selective RoI pooling reduces to RoI pooling, thus the performance decreases significantly.

F. Performance at Different Distances

To analyze the performance of RT3D at different distances, we present the AP_{3D} of vehicles in different ranges, as shown in Table VII. In the range of 10 to 20 meters, the performance reaches a peak, due to most of the marked vehicle samples being located in this range. When the range exceeds 30 m, the performance of the algorithm drops drastically because the point cloud becomes very sparse and the features of the car are hard to be preserved.

TABLE VII
AP_{3D} OF DETECTIONS IN DIFFERENT RANGES

Range	IoU=0.5			IoU=0.7		
(m)	Easy	Moderate	Hard	Easy	Moderate	Hard
0-10	90.58	94.29	95.22	68.92	76.62	78.74
10-20	97.87	98.71	99.01	80.13	84.07	85.67
20-30	95.05	97.06	98.06	63.79	74.02	71.25
> 30	48.02	52.59	53.14	26.91	33.61	36.12

V. CONCLUSION

In this letter, a real-time 3D vehicle detection method is proposed. Two key techniques of pre-RoI-pooling convolution and pose-sensitive feature map help to improve computation efficiency and detection accuracy. Experiments on the KITTI benchmark dataset show that the RT3D achieves comparable detection accuracy against the state-of-the-art methods while completing detection within 0.09 seconds. In the future, we will use more sophisticated encoding scheme to keep more height information in the 2D grid.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[2] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
[3] 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.
[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[5] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.
[6] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Cision*, 2016, pp. 21–37.
[7] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau, "Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1827–1836.
[8] X. Chen *et al.*, "3D object proposals for accurate object class detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 424–432.
[9] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6526–6534.
[10] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.
[11] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1513–1518.
[12] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 1355–1361.
[13] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognition*, 2018, pp. 4490–4499.

[14] S. L. Yu, T. Westfechtel, R. Hamada, K. Ohno, and S. Tadokoro, "Vehicle detection and localization on bird's eye view elevation images using convolutional neural network," in *Proc. IEEE Int. Symp. Safet. Secur. Rescue Robot.*, 2017, pp. 102–109.

[15] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes, "Depthcn: Vehicle detection using 3D-lidar and convnet," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, Oct. 2017, pp. 1–6.

[16] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," in *Proc. Robotics: Sci. Syst.*, 2016, pp. 18–22.

[17] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3D object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2147–2156.

[18] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3D proposal generation and object detection from view aggregation," arXiv:1712.02294, 2017.

[19] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 761–769.

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

[21] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[22] J. Hosang, R. Benenson, P. Dollar, and B. Schiele, "What Makes for Effective Detection Proposals?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 814–830, Apr. 2016.

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

[24] J. Beltran, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. de la Escalera, "BirdNet: A 3D object detection framework from LiDAR information," arXiv:1805.01195.