IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 8, 2020, accepted December 16, 2020, date of publication December 28, 2020, date of current version March 2, 2021.

*Digital Object Identifier 10.1109/ACCESS.2020.3047711*

# A Fast Obstacle Detection Method by Fusion of Double-Layer Region Growing Algorithm and Grid-SECOND Detector

**JIONG LI[1], MIANHAO QIU[2], YU ZHANG[2], NEAL XIONG[3], (Senior Member, IEEE), AND ZHIXIONG LI[4], (Senior Member, IEEE)**

[1]Postgraduate Training Brigade, Army Military Transportation University, Tianjin 300181, China
[2]Army Academy of Armored Forces, Beijing 100072, China
[3]Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK 74401, USA
[4]Yonsei Frontier Lab, Yonsei University, Seoul 03722, Republic of Korea

Corresponding authors: Mianhao Qiu (15801000128@163.com) and Zhixiong Li (zhixiong.li@yonsei.ac.kr)

This work was supported by the National Key Research and Development Program of China under Grant 2020YFF0304000.

**ABSTRACT** Adjacent obstacles are difficult to be distinguished, and remote obstacles are detected easily to split. Besides, limited deep learning samples easily result in missed detection of obstacles in urban environment. In view of this, a fast and robust detection method is proposed by fusing Double-Layer Region Growth algorithm and Grid-SECOND detector. At first, SECOND detector is improved by replacing voxel grids with 2D grids and adopting multi-dimensional features to detect obstacles, which can reduce the time consumption and ensure the accurate detection of remote obstacles. Then, the first Region Growing algorithm is used to cluster the undetected and non-empty grids, which can detect obstacles outside the training set. At last, the second Region Growing algorithm is used to refine the detection results of obstacles with larger volume and multi-obstacles grids, and complete the obstacle detection. Through testing in our extracted urban dataset and KITTI dataset, it is verified that the proposed method outperforms state-of-the-art methods and can accurately achieve obstacle detection. The average duration of the entire process is about 50ms.

**INDEX TERMS** Intelligent vehicles, LIDAR, Grid-SECOND, region growing, obstacle detection.

## I. INTRODUCTION

Obstacle detection in urban environment has long been the focus of the research on unmanned driving environment perception. The commonly used sensors for obstacle detection include LIDAR, millimeter wave radar and camera. Due to the fact that the camera is greatly affected by light and cannot be used all day long, this study adopts LIDAR characterized by high measurement accuracy and strong anti-interference ability [1]. The purpose of obstacle detection based on LIDAR is to distinguish the point clouds of different obstacles from raw point clouds, and provide necessary basis for the following work such as obstacle classification [2], tracking [3] and LIDAR localization [4], [5].

In general, obstacle detection methods based on LIDAR are currently divided into two categories based on traditional clustering [6], [7] and deep learning [8], [9], respectively.

The study of traditional clustering methods is relatively early due to its strong stability and real-time performance. For example,

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

Nurunnabi *et al.* [10] adopted the distance from the point to its nearest fitting plane and the change of local surface normal direction for obstacle detection. However, this method has a strong dependence on the initial cluster center, and it is difficult to distinguish adjacent obstacles. Feng *et al.* [11] proposed an Ordering Points To Identify The Clustering Structure (OPTICS) method which replaces the core point with the center point of the grid, and optimizes Density-Based Spatial Clustering of Applications with Noise (DBSCAN) method. Compared with traditional DBSCAN, this method has higher detection accuracy and reduces time consuming. But it is easy to detect splitting for remote obstacles. Desheng *et al.* [12] proposed an Eight-neighbor grids clustering method. Although this method can detect obstacles quickly, it still cannot distinguish adjacent obstacles accurately. The adaptive K-means algorithm is used by Benyue *et al.* [13] to detect obstacles. The method can improve the problem of Xie's work [12]. But it is still easy to detect splitting for remote obstacles and has poor real-time performance.

With the continuous development and progress of deep learning technology, the deep learning detection methods based on LIDAR have gradually begun to be widely studied due to its high detection and classification accuracy. Aiming at the disorder of point cloud, Charles *et al.* [14] proposed a neural network PointNet. This network can process a single sampling point, and then connect all point clouds features. But it only considers the global features of each point instead of the local environment features. To solve this problem, Charles *et al.* [15] had done a lot of researches on local feature extraction, and proposed a network structure PointNet++. The network adopted the idea of hierarchical feature extraction to solve the problem of uneven sampling density of point clouds. Li *et al.* [16] proposed an obstacle detection method based on Full Convolution Neural Network (FCN). This network can detect the obstacles through a large number of training data set without artificial target features, but it has poor real-time performance. Zhou and Tuzel [17] of Apple Company proposed an end-to-end network VoxelNet, which can improve the problem of Su's work [13]. However, it has poor performance of direction estimation. In order to obtain the accurate orientation of obstacles, Yan *et al.* [18] proposed a Sparsely Embedded Convolutional Detection (SECOND), which can make full use of the rich 3D information present in point cloud data. Shi *et al.* [19] proposed a new two-stage 3D detection framework PointRCNN with high robustness and accurate 3D detection performance. However, it is still time-consuming. Although the above deep learning networks can solve the problem of sparsity of point cloud and realize the accurate detection of remote obstacles, it is difficult to detect the obstacles outside the training dataset.

In order to improve the detection accuracy and solve the problems suffered by the above methods, a fast and robust obstacle detection method is proposed in this paper. Firstly, the point clouds after road segmentation are rasterized and the grids in Region Of Interest (ROI) are filtered. Secondly, SECOND detector is improved by replacing voxel grids with 2D grids and adopting multi-dimensional features to detect obstacles, which can solve the problem of remote obstacle detection splitting. After that, the first Region Growing algorithm is adopted to cluster the undetected and non-empty grids, which can solve the missing detection problem caused by the limited dataset. Finally, the second Region Growing algorithm is adopted to refine the obstacles with larger volume and multi-obstacles grids detected by the first Region Growing algorithm, and effectively distinguishes the adjacent obstacles.

The main contributions of this paper are summarized as follows.

1. Grid-SECOND detector is proposed to improve SECOND detector by adopting multi-dimensional features and replacing voxel grids with 2D grids. The Grid-SECOND detector can reduce the time consumption and improve the original detection accuracy.

2. Double-Layer Region Growing Algorithm is proposed to solve the problem of difficult to distinguish adjacent obstacles, while ensuring stable detection of obstacles.

3. This paper adopts Double-Layer Region Growing Algorithm to supplement the Grid-SECOND detector and solve the problem of obstacles outside the training set undetected.

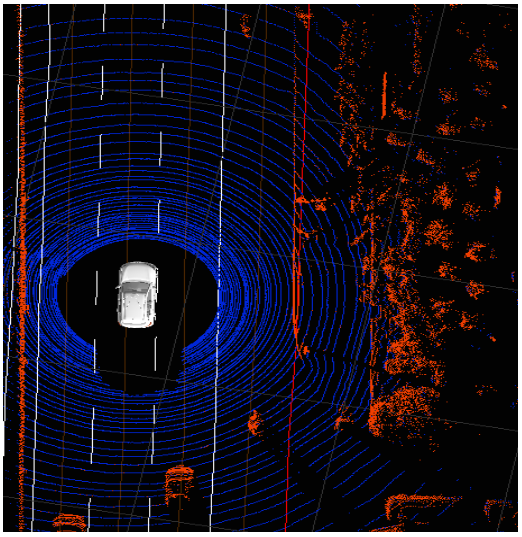


**FIGURE 1.** Road segmentation.

The remainder of the paper is structured as follows. Point cloud data preprocessing is in Section II. In Section III, we respectively introduce three methods: Double-Layer Region Growing, Grid-SECOND detector, and a fast detection method by fusion of Double-Layer Region Growing algorithm and Grid-SECOND detector. Section IV conducts experiments on our extracted urban dataset and KITTI data set respectively and analyses the results. At last, Section V presents some conclusions and challenges.

## II. DATA PREPROCESSING

HDL-64 LIDAR is adopted in this paper to improve the ability of obstacle detection in urban environment, which can generate nearly 130000 points per frame. If the point clouds are processed directly, it is difficult to guarantee real-time performance. Therefore, to improve the detection speed and accuracy, this study adopts the method utilized by Jiong *et al.* [20] to complete the road segmentation, as shown in Figure 1. The blue points represent the ground points and the red points represent the obstacle points. Then, we adopt a 40 cm×40cm grid to rasterize the point clouds, thus creating a 200 × 200 grid occupancy map, as shown in Figure 2. The center point of the grid is then transformed from LIDAR coordinate to Geodetic Coordinate System(GCS) using equation (1). At last, we judge whether the center of grid is in the high-precision map composed of quadrilateral by equation (2), and complete ROI filtering.

$$\boldsymbol{P}_t = \boldsymbol{R}_t \times \boldsymbol{P} + \boldsymbol{T}_t \tag{1}$$

$$\begin{cases} [(\boldsymbol{P}_1-\boldsymbol{P}_2)\times(\boldsymbol{P}_1-\boldsymbol{P}_t)]\times[(\boldsymbol{P}_3-\boldsymbol{P}_4)\times(\boldsymbol{P}_3-\boldsymbol{P}_1)]\geq 0 \\ [(\boldsymbol{P}_2-\boldsymbol{P}_3)\times(\boldsymbol{P}_2-\boldsymbol{P}_t)]\times[(\boldsymbol{P}_4-\boldsymbol{P}_1)\times(\boldsymbol{P}_4-\boldsymbol{P}_t)]\geq 0 \end{cases} \tag{2}$$

where, $\boldsymbol{P}_t$ is coordinate values of grid center in GCS. $\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3, \boldsymbol{P}_4$ denote quadrilateral four vertices, respectively. $\boldsymbol{T}_t$, $\boldsymbol{R}_t$ denote translation matrix and rotation matrix from LIDAR coordinate to GCS at time t,respectively.

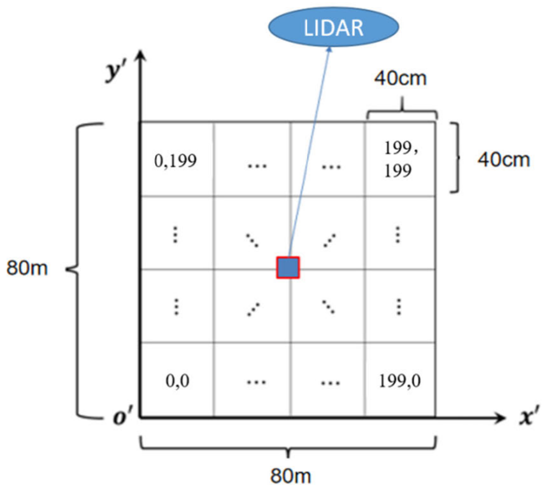


**FIGURE 2.** Point cloud data rasterizing.

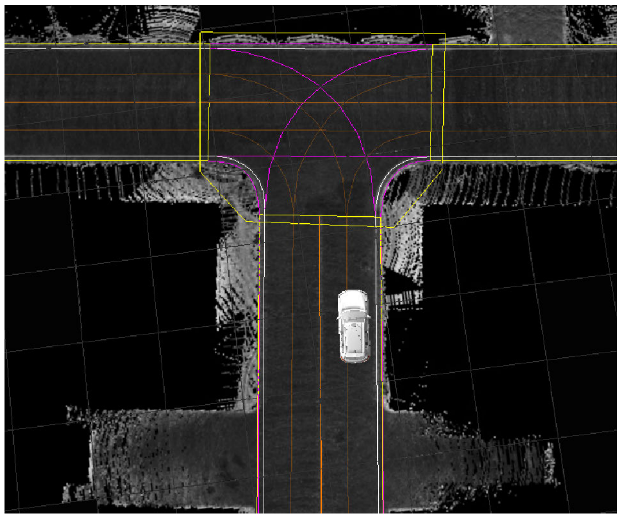


**FIGURE 3.** High precision map.

The production process of high-precision map is as follows: first of all, the method of Dongbin *et al.* [21] is used to complete reflectivity calibration of LIDAR. After that, the method of Wen *et al.* [22] is adopted to overlay ground point cloud. Next, Josm map editing software is used to plot a high-precision map [22] on the superimposed gray-scale map, as shown in Figure 3.

## III. METHODS

### A. DOUBLE-LAYER REGION GROWING

As an image segmentation algorithm, Region growing algorithm can segment the connected regions with the same features, thus ensuring good edge information. Although this algorithm is simple and fast, it may easily lead to over-segmentation of obstacles when the grid is smaller, and it is difficult to distinguish adjacent obstacles when the grid is larger. Therefore, a Double-Layer Region Growing algorithm is proposed in this paper, which first uses larger grids to avoid obstacle over-segmentation, and then uses smaller grids to select obstacle that are characterized by large volume and

**Algorithm 1 :** Region Growing

```
1.  intput: grid map for point clouds, ObList
2.  output: ObList
3.  for i = 0 to row do
4.    for j = 0 to col do
5.      SGrid←grids[i,j];
6.      if isSeedGrid (SGrid)==true and flag(SGrid)==0
7.        flag(SGrid)←1;  list.add(SGrid);
8.        extend cluster(8 neighbor grids of SGrid, list);
9.        ObList.add(list), list.Clear;
10.     end if
11.   end for
12. end for
```

**FIGURE 4.** Region Growing.

**Algorithm 2 :** Double-Layer Region Growing

```
1.  Create a ObList, BigList, grid map1(40cm╳40cm) for
    point clouds, $T_{ch}$←0.3;
2.  Region Growing(ObList, grid map1);
3.  For i = 0 to ObList.count do
4.    If isBigOb(ObList[i])
5.      BigList.add(ObList[i])
6.      ObList.remove(ObList[i])
7.    end if
8.  end for
9.  Create a FianlList, grid map2(20cm╳20cm) for BigList,
    $T_{ch}$←0.15;
10. Region Growing(FianlList, grid map2);
11. FianlList.add(ObList);
```

**FIGURE 5.** Double-Layer Region Growing.

may contain multiple targets for refinement. The structure of the algorithm is shown in Figure 4 and 5.

*Step 1:* Firstly, traversal flag of the grid $F_{visit}$ is initialized to 0. Then, equation (3), (4) and (5) are used to calculate the height difference $\Delta h$, barycenter $\boldsymbol{b}_i$ and variance value $\sigma_i$ for each grid in ROI. The grid with $\Delta h$ greater than height difference threshold $T_{\Delta h}(T_{\Delta h} = 0.4)$ is selected as the seed grid. The grid with $\sigma_i$ greater than the variance threshold $T_v(T_v = 2.0)$ is marked as multi-obstacles.

$$\Delta h = h_{\max} - h_{\min} \tag{3}$$

$$\boldsymbol{b}_i = \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{p}_j \tag{4}$$

$$\sigma_i = \frac{1}{n}\sum_{j=1}^{n}\sqrt{\left(b_{ix}-p_{jx}\right)^2+\left(b_{iy}-p_{jy}\right)^2} \tag{5}$$

where, $h_{\max}$, $h_{\min}$ denote the maximum and minimum height of points in the grid, respectively. $\boldsymbol{p}_j = \left(p_{jx}, p_{jy}\right), \boldsymbol{b}_i = \left(b_{ix}, b_{iy}\right)$ denote point $j$ and the barycenter in the grid $i$, respectively. $\sigma_i$ is the variance of points in grid $i$.

*Step 2:* 8-neighbor grids extended clustering. We assume that grid A in Figure 6 is the seed grid. The 8-neighbor grids of the grid A are searched counterclockwise, as shown in Figure 6. The grid whose $F_{visit}$ is 0 is processed as follows:

1) If $\Delta h > T_{\Delta h}$ in grid B, the grid is an obstacle grid. In addition, if the maximum height difference between grid B

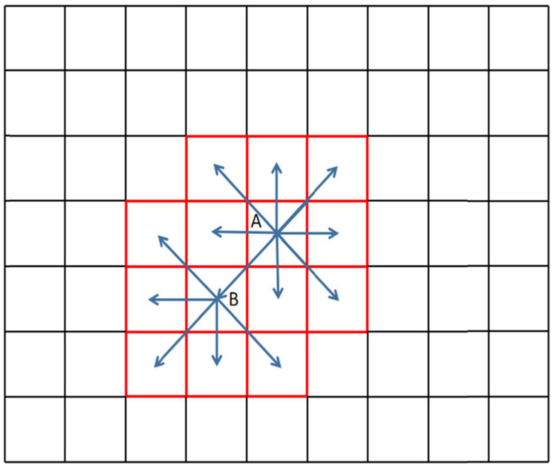


**FIGURE 6.** 8-neighbor grids extended clustering diagram.

and seed grid A is less than the height difference threshold of the same obstacle $T_{ch}(T_{ch} = 0.3)$, grid A and grid B belong to the same obstacle. They are thus added to the same obstacle list and $F_{visit}$ of grid B is 1. Otherwise, grid B is skipped and $F_{visit}$ is unchanged.

2) If $\Delta h < T_{\Delta h}$ in grid B, $F_{visit}$ of grid B is 1.

After traversing the 8-neighbor grids in turn, the succeeding grid of the current seed grid in the obstacle list is used as the new seed grid for 8-neighbor search. All the grids are traversed to complete the obstacle detection.

*Step 3:* Obstacle refinement. If the clustered obstacles have large volume and multi-obstacles marking grids to satisfy the requirements of equation (6), 20cm×20cm grid and $T_{ch} = 0.15$ are used to perform Step 2 again to refine them.

$$\begin{cases} ob_{g\_n} \geq T_{g\_n} \\ ob_{m\_n} \geq T_{m\_n} \end{cases} \tag{6}$$

where, $ob_{g\_n}$, $ob_{m\_n}$ are the number of grids and the number of grids marked as multi-obstacles, respectively. $T_{g\_n}$, $T_{m\_n}$ are thresholds for number of grids requiring second clustering and number of multi-obstacle grids ($T_{g\_n} = 15$, $T_{m\_n} = 4$), respectively

## B. GRID-SECOND DETECTOR

SECOND detector is a derivative of VoxelNet, which can solve the problem of sparse point cloud of remote obstacles, and realize the stable detection of obstacles without splitting. Inspired by the SECOND Detector, a Grid-SECONG detector is proposed in this paper on the basis of its main structure. The Grid-SECONG uses multi-dimensional features and replaces voxel grids with 2D grids to improve SECOND detector, thus improving the original detection accuracy and reducing the time consumption.

The network structure of the Grid-SECONG detector is shown in Figure 7, which consists of four components: (1) Rastering and multi-dimensional Features; (2) Grid Feature Extractor; (3) Sparse Conv Layers; (4) RPN.

*Step 1:* We first adopt the method of data preprocessing to divide the point clouds into grids with 40 cm. Then, mean heigh $h_{\text{mean}}$, maximum height $h_{\max}$, mean intensity $I_{\text{mean}}$, maximum intensity $I_{\max}$, the number of point clouds $N_{\text{count}}$, the angle $A_{\text{dir}}$ of the center point relative to the origin, the distance $D_{\text{ctr}}$ between the center point and the origin, and the occupation flag $F_{\text{non}}$ per grid are calculated. At last, these multi-dimensional features of the grid are input to the network as 8 channels for gridlwise feature extraction.

*Step 2:* We treat the 2D grid as a voxel grid with the height of 1. Based on the voxel feature encoding (VFE) layer as described in reference 17, we adjust the form of input tensor to construct grid feature encoding (GFE) for extracting gridwise features. A GFE layer takes only one tensor with 8 channels in the same grid as the input point, and uses a fully connected network (FCN) consisting of a linear layer, a batch normalization (BatchNorm) layer and a rectified linear unit (ReLU) layer to extract gridwise features. Then, it uses elementwise max pooling to obtain the locally aggregated features for each grid. As a whole, the gridwise feature extractor consists of several GFE layers and an FCN layer.

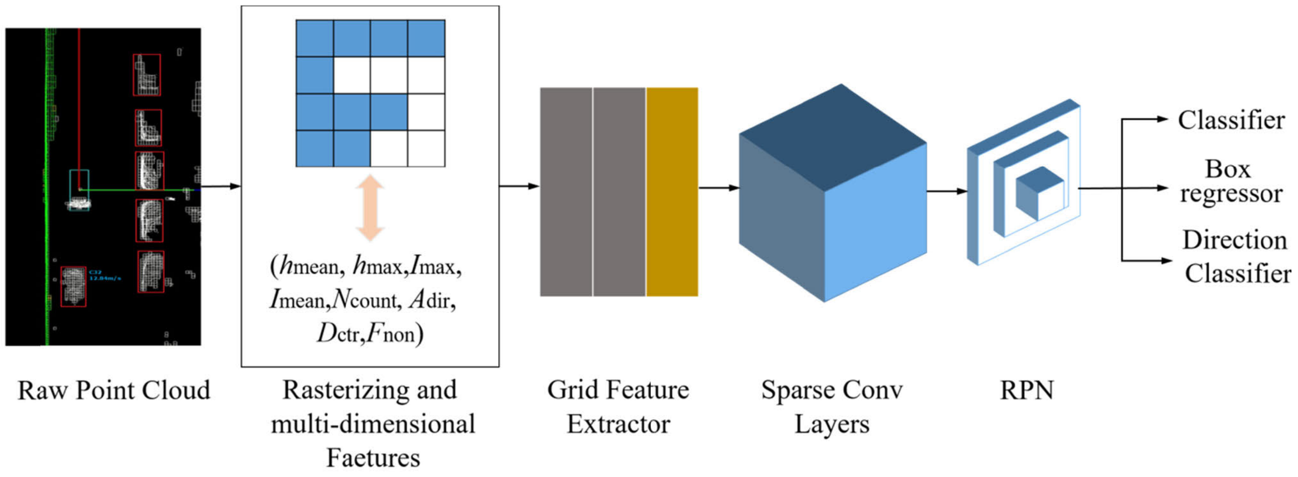
*Step 3:* Sparse Conv Layers, as described in reference 18 is adopted in this paper to increase the speed of both training and inference. It consists of two phases of sparse convolution. Each phase contains several submanifold convolutional layers and one normal sparse convolution to perform downsampling in the z-axis. When the z-dimensionality is downsampled to one or two, the sparse 3D data will be converted into a 2D BEV image.

*Step 4:* We adopt an RPN architecture composed of three stages in reference 18. Each stage is followed by several downsampled convolutional layers. After each convolutional layer, BatchNorm and ReLU layers are applied. Then, we upsample the output of each stage to a feature map of the same size and concatenate these feature maps into one feature map. Finally, three $1 \times 1$ convolutions are applied for the prediction of class, regression offsets and direction.

## C. A FAST DETECTION METHOD BY FUSION OF DOUBLE-LAYER REGION GROWING AND GRID-SECOND DETECTOR

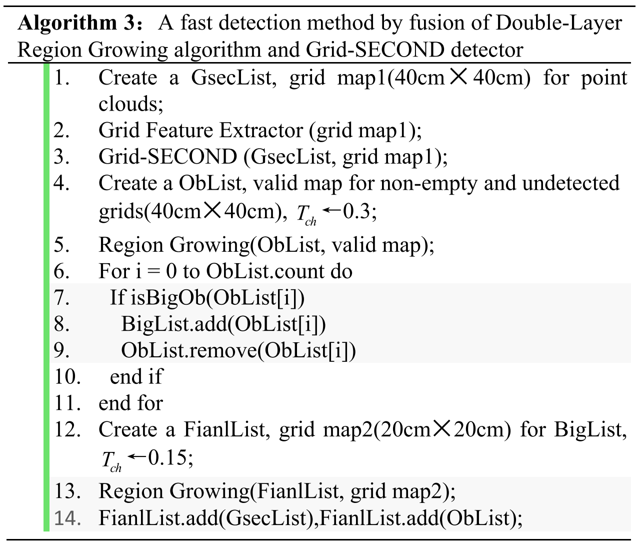
Though Double-Layer Region Growing algorithm can quickly distinguish adjacent obstacles, it is easy to detect splitting for remote obstacles. Grid-SECOND detector adopts the sparse distribution of point clouds for training, and can solve the problem of remote obstacle detection splitting. However, due to the limitation of training obstacle types, when the obstacles are close or not involved in training set, these obstacles are easily missed. This paper combines the above two detection methods, and proposes a more robust and faster detection method. The structure of the method is shown in Figure 8. In addition, the specific steps are as follows.

*Step 1:* To start, the preprocessed grids are processed as follows according to the steps of the Grid-SECOND detector: Calculate 8 channels features per grid; Extract grid features; Convert sparse 3D data into a 2D BEV image; Apply RPN for



Raw Point Cloud
($h$mean, $h$max, $I$max, $I$mean, $N$count, $A$dir, $D$ctr, $F$non)
Rasterizing and multi-dimensional Faetures
Grid Feature Extractor
Sparse Conv Layers
RPN
Classifier
Box regressor
Direction Classifier

**FIGURE 7.** The structure of our proposed Grid-SECOND detector.



**Algorithm 3:** A fast detection method by fusion of Double-Layer Region Growing algorithm and Grid-SECOND detector

```
1.  Create a GsecList, grid map1(40cm×40cm) for point clouds;
2.  Grid Feature Extractor (grid map1);
3.  Grid-SECOND (GsecList, grid map1);
4.  Create a ObList, valid map for non-empty and undetected grids(40cm×40cm), $T_{ch}$ ←0.3;
5.  Region Growing(ObList, valid map);
6.  For i = 0 to ObList.count do
7.    If isBigOb(ObList[i])
8.      BigList.add(ObList[i])
9.      ObList.remove(ObList[i])
10.   end if
11. end for
12. Create a FianlList, grid map2(20cm×20cm) for BigList, $T_{ch}$ ←0.15;
13. Region Growing(FianlList, grid map2);
14. FianlList.add(GsecList),FianlList.add(ObList);
```

**FIGURE 8.** A fast detection method by fusion of Double-Layer Region Growing algorithm and Grid-SECOND detector.

the prediction of class, regression offsets and direction. Then, the first obstacle detection is completed.

*Step 2:* Next, considering that there may be adjacent obstacles or obstacles that are not in the training set in urban environment, the Grid-SECOND detector may fail to detect them. Therefore, the non-empty and undetected grids are selected from the grid map in the Step1, and the Double-Layer Region Growing algorithm is used to cluster these grids according to the steps of III-A. Finally, the obstacles which aren't detected by Grid-SECOND is detected by twice Regional Growth algorithm.

## IV. EXPERIMENT

At first, according to the loss function definition and training method in reference 18, this paper uses KITTI data set which contains 7481 training point clouds to complete the Grid-SECOND detector model training. Then, combined with the Double-Layer Region Growing algorithm, the following experimental tests are carried out.

This experiment consists of two parts: Part A is the experiment on typical urban dynamic. LiDAR data is collected by Velodyne HDL-64E. It includes adjacent obstacles, remote obstacles and obstacles that are not involved in the training set. Part B is the experiment in the public test dataset provided by KITTI [23]. During both experiment parts, the methods for processing the LiDAR data are run on a computer with i7-8700 processor, RTX2070 graphics card and 16G RAM. The system environment is ubuntu16.04 and the development environment is QtCreator 5.8. Besides, the software is written by C++.

### *A. EXPERIMENT ON URBAN DYNAMIC SCENES*

In this paper, the obstacle with more than 15 grids (40cm×40cm) is defined as large obstacle. Then, 5000 frames are collected from the urban dynamic environment and are marked manually, including 15036 large obstacles. This paper uses Region Growing algorithm [12] and SECOND [18] detector to process the 5000 frames, respectively and compares the detection results with the proposed method.

#### 1) QUALITATIVE ANALYSIS

In order to illustrate the performance of the proposed method, we selected three typical urban dynamic scenes for the analysis. The raw point clouds of the three scenes are shown in Figure 9(a), Figure 11(a) and Figure 13(a). The point clouds after ROI filtering of the three scenes are shown in Figure 9(b), Figure 11(b) and Figure 13(b). The detection results of the three scenes corresponding to the three methods are shown in Figure 10, Figure 12 and Figure14 respectively. Different obstacles detected by the SECOND detector and the Grid-SECOND detector are represented by different colors, such as purple for vehicles, red for unknown, and blue for pedestrians. The obstacles detected by the Region Growing algorithm is randomly colored.

Scene A1 contains a remote vehicle as shown in Figure 9 (A box), and local 3D display of A box is shown in Figure 9(c). Since the vehicle is far from the LIDAR, the point

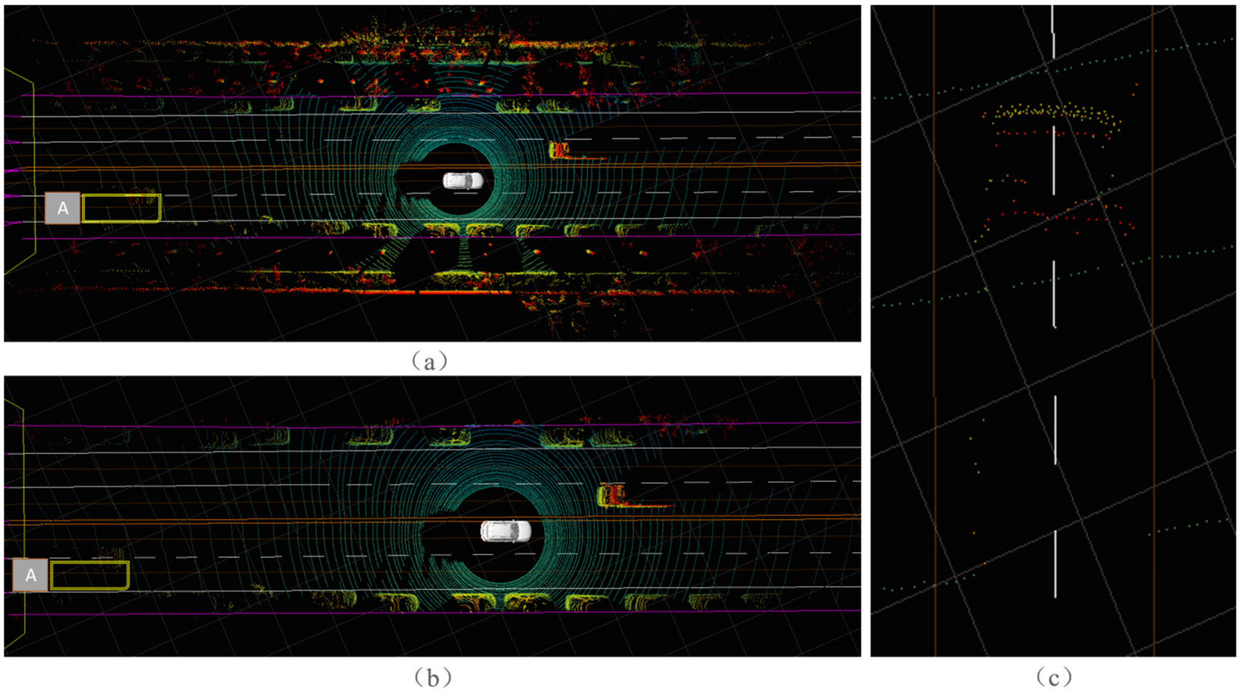


FIGURE 9. Scene A1. (a) Raw points. (b) Raw points after ROI filtering. (c) Local 3D display (A box).

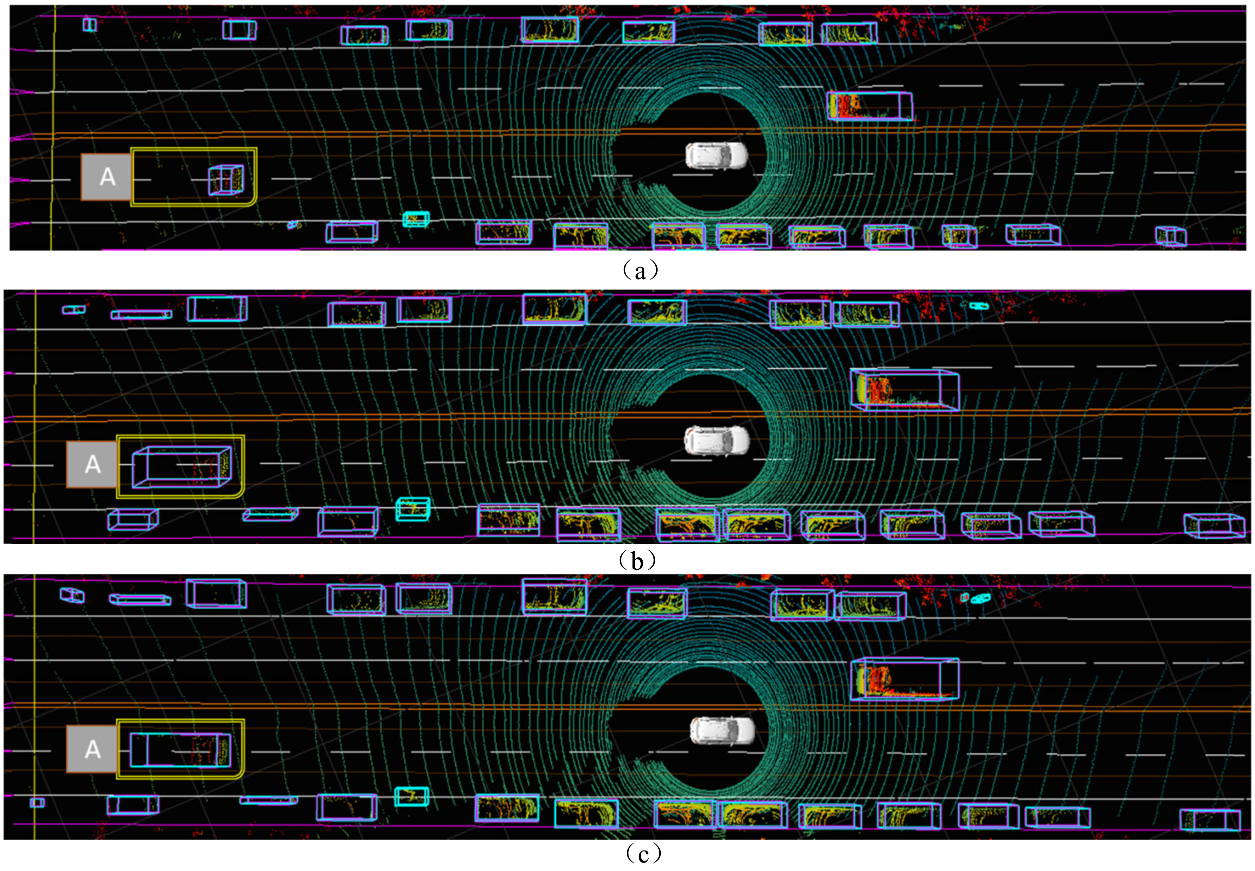


FIGURE 10. Comparison of the detection results in scene A1. (a) Method in reference 12. (b) Method in reference 18. (c) The proposed method.

clouds at the front of the vehicle are dense and the point clouds at the top of the vehicle are sparse. The SECOND [18] detector and the proposed method use sparse point clouds for training, and can accurately detect the complete vehicle shape. However, due to small height difference and small number of the point cloud at the top of the vehicle, the vehicle

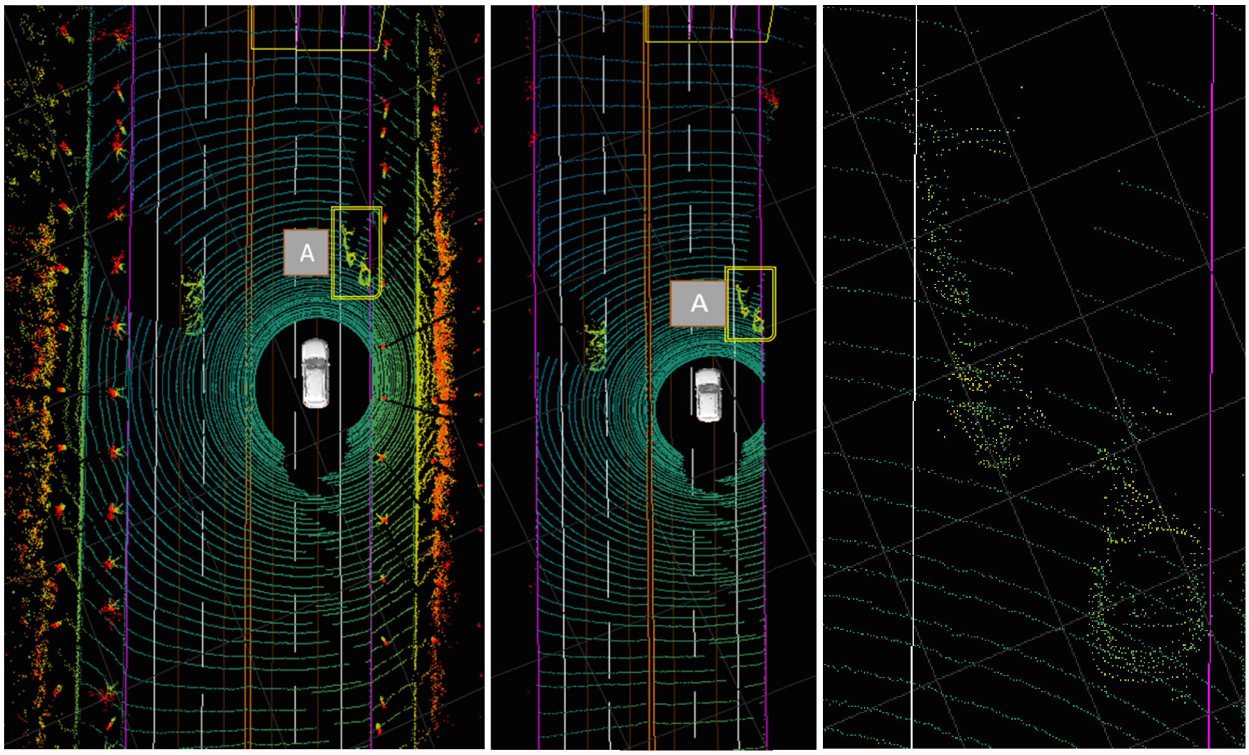


**FIGURE 11.** Scene A2. (a) Raw points. (b) Raw points after ROI filtering. (c) Local 3D display (A box).

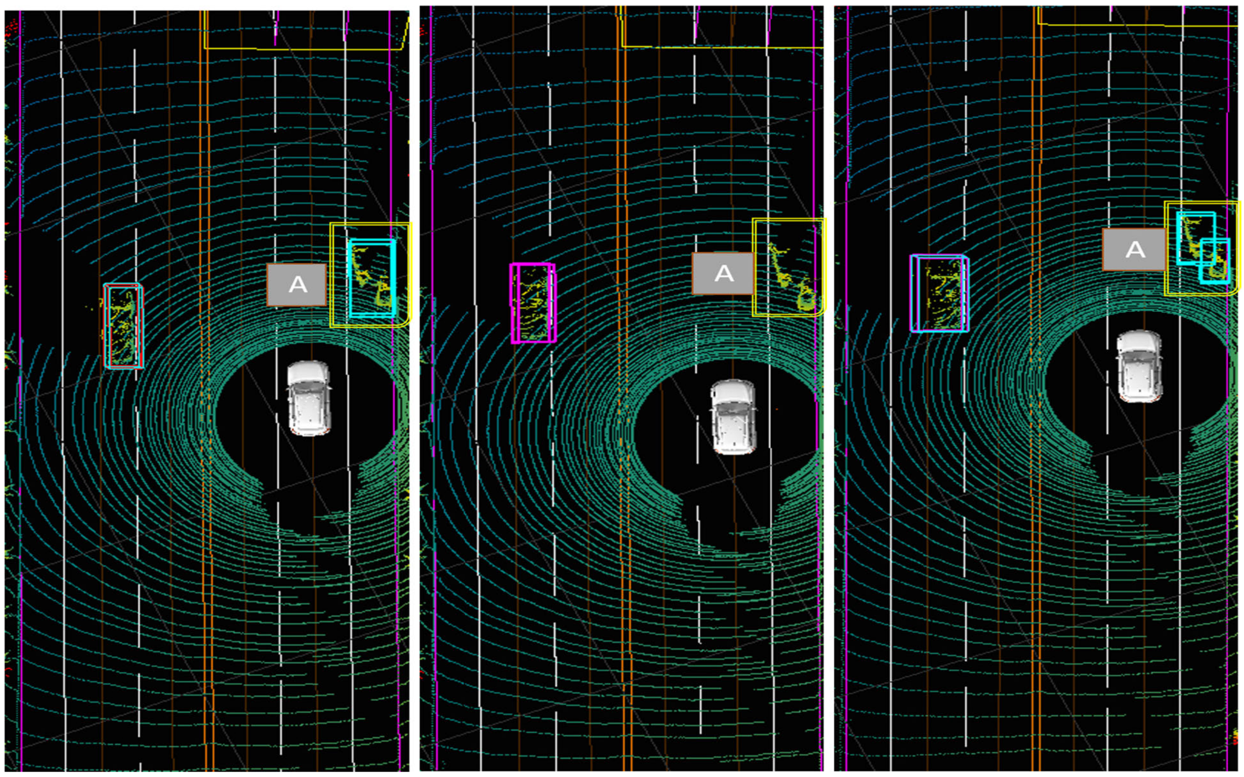


**FIGURE 12.** Comparison of the detection results in scene A2. (a) Method in reference 12. (b) Method in reference 18. (c) The proposed method.

is detected split by Region Growing algorithm [12] and only the front part is detected, as shown in Figure 10(a).

Scene A2 contains two adjacent bicycles, as shown in Figure 11(A box), and local 3D display of A box is shown

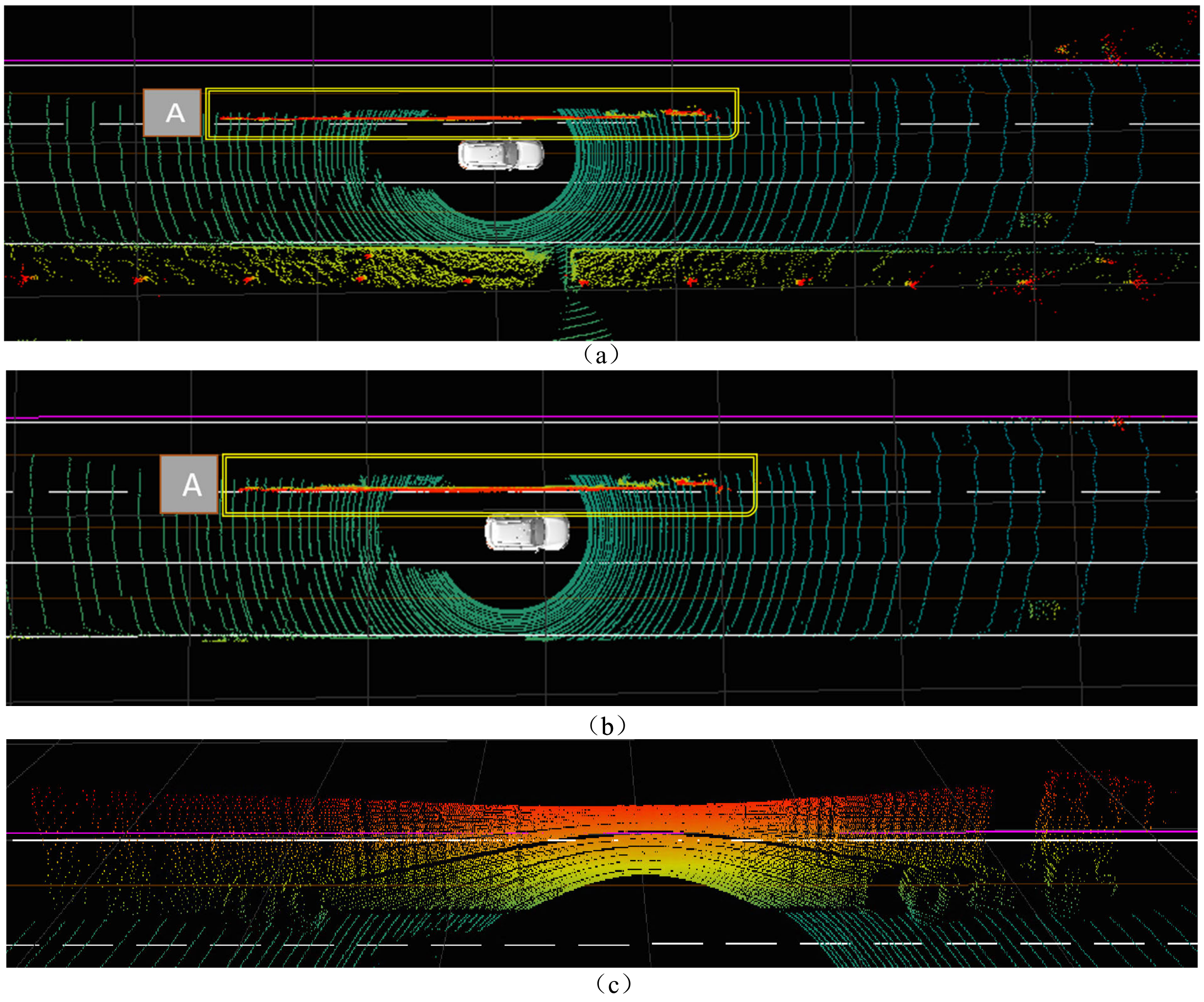


**FIGURE 13.** **Scene A3. (a) Raw points. (b) Raw points after ROI filtering. (c) Local 3D display (A box).**

in Figure 11(c). Since the two bicycles are similar in shape and close together, the features interfere with each other and the SECOND [18] detector cannot detect them as shown in Figure 12(b). Although Region Growing algorithm [12] can detect the adjacent obstacles, it cannot refine and distinguish them due to the large grid(40cm). The proposed method uses the second Region Growth algorithm to refine the adjacent obstacles detected by the first Region Growth algorithm, and successfully distinguishes them as shown in Figure 12(c).

Scene A3 contains a big truck. When it runs to the side of the Intelligent Vehicle, the LIDAR can only scan the side of the big truck. The shape of point cloud looks like a wall that is not involved in training set, as shown in Figure 13(A box), and local 3D display of A box is shown in Figure 13(c). At this time, the SECOND [18] detector cannot detect the big truck, as shown in Figure 14(b). In contrast, the Region Growing algorithm [12] with large grid (40cm) can detect the big truck as shown in Figure 14(a). The proposed method can also accurately detect the big trunk through the first Region Growth algorithm with large grid (40cm) as shown in Figure 14(c). Given that the point cloud at the junction of the front and body of the trunk is dense, the covariance of the grid is small. Then, the big trunk cannot meet the conditions of using the second Region Growth algorithm with small grid (20cm), avoided false detection.

### 2) QUANTITATIVE ANALYSIS

In order to further verify the accuracy and robustness of the method, the detection results of obstacles in the 5000 frames are evaluated by positive detection rate $P_{tp}$, false detection rate $P_{fp}$ and missed detection rate $P_{md}$ [24] according to equation (7).

$$\begin{cases} P_{tp} = N_{tp}/N_{\text{sum}} \\ P_{fp} = \left(N_{fg} + N_{fs}\right)/N_{\text{sum}} \\ P_{md} = N_{md}/N_{\text{sum}} \end{cases} \tag{7}$$

where, $N_{\text{sum}}$ is the total number of obstacles. $N_{tp}$ is the number of correctly detected obstacles. $N_{fg}$ is the number of ground point detected as obstacles. $N_{fs}$ is the number of detected split or incorrectly distinguished obstacles and $N_{md}$ is the number of undetected obstacles.

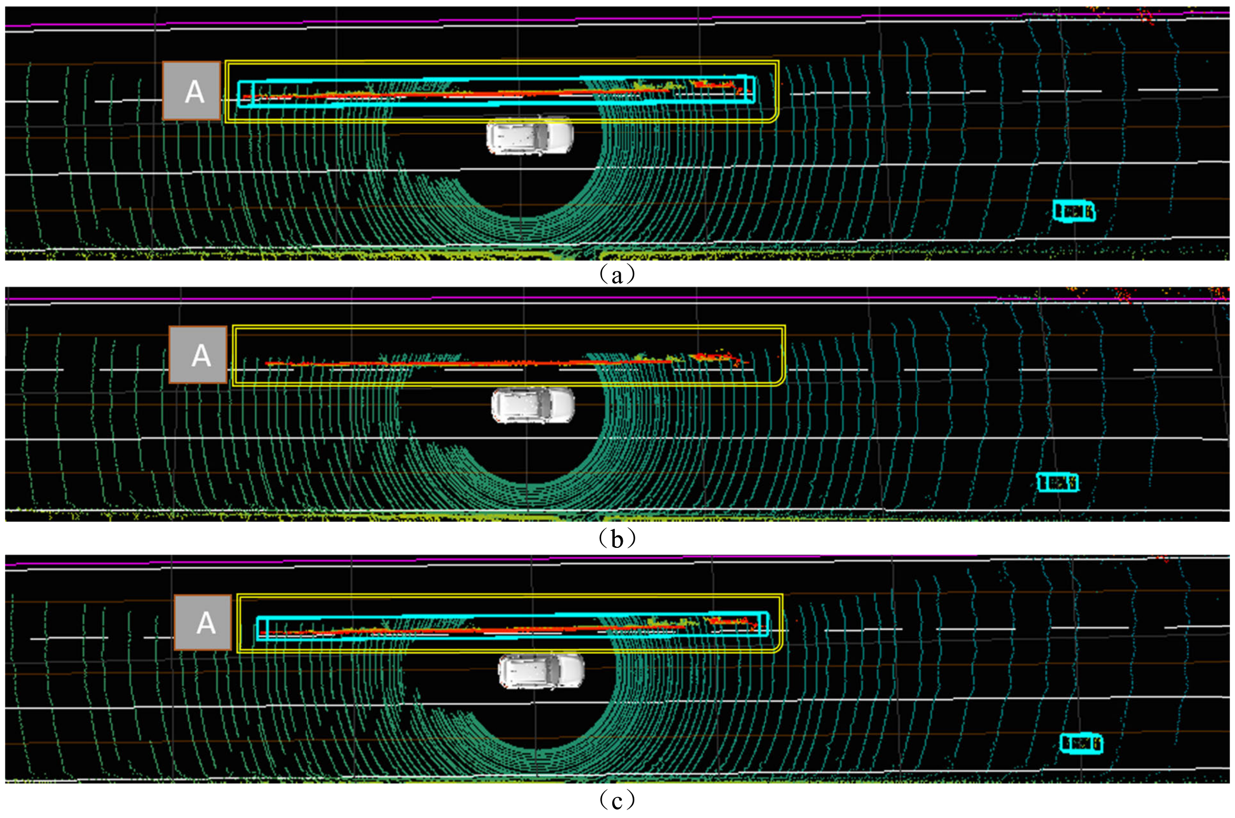


**FIGURE 14.** Comparison of the detection results in scene A2. (a) Method in reference 12. (b) Method in reference 18. (c) The proposed method.

Therefore, the higher $P_{tp}$, the better the detection effect. The higher the $P_{fp}$ and $P_{md}$, the poor the detection effect.

Table 1 lists respectively the overall detection results of the three methods. In the urban dynamic environment, the detection results of the proposed method for different obstacles are obviously better than those of the other two methods. Firstly, the method in reference 12 is difficult to distinguish adjacent objects, therefore, $P_{tp}$ is relatively low. Due to the fact that the method is also greatly affected by height difference, it is easy to detect splitting for remote obstacles, and the $P_{fp}$ is relatively high. Then, the method in reference 18 is not affected by the height difference, but it is still difficult to detect objects that are not in the training set. Thus, the $P_{md}$ is relatively high. The proposed method combines Double-Layer Region Growing algorithm and Grid-SECOND detector to effectively distinguish adjacent obstacles, solve the splitting in remote obstacle detection, and realize the detection of the obstacles that are not involved in training set. Compared with the methods in reference 12 and reference 18, the overall detection accuracy is increased by about 12%.

### *B. EXPERIMENT ON KITTI DATASET*

We evaluate the proposed method on the KITTI 3D object detection test set [23] which contains 7518 test point clouds, covering three categories: Car, Pedestrian, and Cyclist. For each class, detection outcomes are evaluated based on three difficulty levels: easy, moderate, and hard, which are determined according to the object size, occlusion state, and truncation level. We compare the proposed method with state-of-the-art methods of 3D object detection. Among them, for obstacles detected by the Region Growing algorithm under different types, this paper only need to manually judges whether the detection is correct to obtain the detection accuracy.

The performance of these 3D detectors on KITTI test set is shown in Table 2. Our method is superior to the state-of-the-art methods. Although PointNet++ uses a 2D detector that has been fine-tuned using ImageNet weights to achieve better results in Pedestrian detection, our network is trained from scratch and uses only LiDAR data. For car and cyclist detection, PointRCNN outperforms SECOND. While for the pedestrian detection, SECOND outperforms PointRCNN. In our method, SECOND detector is optimized based on the grid, and the Double-Layer Region Growth algorithm is used to solve the problem of the missed detection by Grid-SECOND detector. Therefore, our proposed method achieves better results than PointRCNN and SECOND.

The average time cost of these different methods on the KITTI test set is shown in Figure 15. Because VoxelNet operates directly on sparse 3D points and convolutional middle layers is too complicated, the time-consuming of VoxelNet is the longest, and the inference time is 230ms. However, the PointRCNN and VeloFCN are relatively less time-consuming, and take about 80ms. Since the

**TABLE 1. Results of obstacle detection in different methods.**

| Method | The number of large obstacles | $N_{\text{sum}}$ | $P_{tp}$ /% | $P_{fp}$ /% | $P_{md}$ /% | Average time cost (ms) |
|---|---|---|---|---|---|---|
| Method in reference 12 | 15036 | 40211 | 85.56 | 9.28 | 5.16 | 5.3 |
| Method in reference 18 | 15036 | 40211 | 87.68 | 4.5 | 7.82 | 48.1 |
| The proposed method | 15036 | 40211 | 97.83 | 1.47 | 1.28 | 49.3 |

**TABLE 2. Performance comparison in 3D detection: average precision (AP, in %) on KITTI test set.**

| Method | Car(IOU=0.7) | | | Pedestrian(IOU=0.5) | | | Cyclist(IOU=0.5) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Easy | Moderate | Hard | Easy | Moderate | Hard | Easy | Moderate | Hard |
| VeloFCN | 15.32 | 13.57 | 15.68 | N/A | N/A | N/A | N/A | N/A | N/A |
| PointNet++ | 81.72 | 70.31 | 60.68 | 52.23 | 44.38 | 40.31 | 71.86 | 55.67 | 50.36 |
| VoxelNet | 77.28 | 66.21 | 58.43 | 39.72 | 34.23 | 31.25 | 62.13 | 47.25 | 45.27 |
| SECOND | 83.23 | 72.35 | 65.28 | 52.11 | 42.14 | 38.53 | 70.65 | 54.17 | 46.83 |
| PointRCNN | 85.41 | 75.29 | 67.33 | 48.95 | 41.75 | 38.47 | 73.81 | 58.59 | 53.19 |
| Ours | 86.12 | 77.43 | 68.25 | 55.68 | 46.21 | 38.66 | 75.44 | 59.63 | 55.27 |

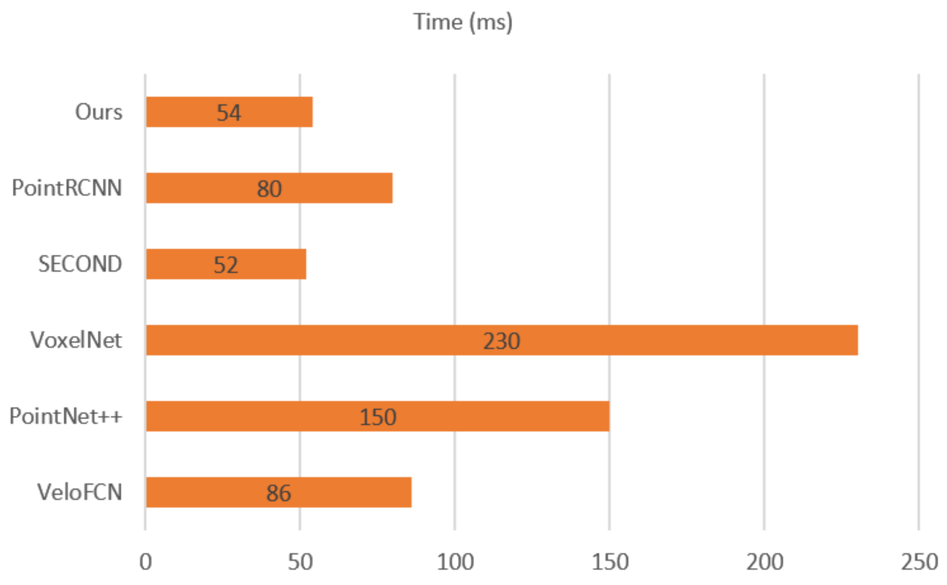


**FIGURE 15. Comparison of average time cost in different methods.**

SECOND detector uses improved sparse convolution, the time-consuming of the SECOND detector is the shortest, and the inference time is 50ms. We adopt 2D grids instead of voxel grids to reduce the time-consuming of the SECOND detector, and fuse the Double-Layer Region Growth algorithm to increase the overall detection accuracy of the proposed method, while ensuring that the total time remain is equivalent to that of the SECOND detector.

In order to further analyze the orientation accuracy of obstacle, we select Car category with higher orientation sensitivity for analysis. The performance comparison of average orientation similarity (AOS) on KITTI test set for the Car class is shown in Table 3. This paper extracts multiple features of different dimensions in the same grid as input, which is convenient for obtaining the 3D box proposals. Therefore, compared with SECOND detector that extracts point features

**TABLE 3. Performance comparison in 3D detection: average orientation similarity (AOS, in %) on KITTI test set for the Car class.**

| Method | Easy | Moderate | Hard |
|---|---|---|---|
| SECOND | 86.12 | 73.56 | 68.11 |
| Ours | 88.34 | 76.88 | 69.45 |

separately, our method can obtain the obstacle orientation more accurately.

## V. CONCLUSION

This paper proposes a fast and robust obstacle detection method that combines Double-Layer Region Growing algorithm and Grid-SECOND Detector. At first, this method adopts Grid-SECOND detector that replaces voxel grids with 2D grids and adopts multi-dimensional features to improve SECOND detector for obstacle detection. Then, the first Region Growing algorithm is used to cluster the remaining undetected and non-empty grids. Finally, the second Region Growth algorithm is used to refine the obstacles that are characterized by large volume and may contain multiple targets. It effectively distinguishes the adjacent obstacles and solves the splitting of remote obstacle detection, while making up for the missing detection problem caused by the limited training set.

The proposed method is verified in our extracted urban dataset and the KITTI dataset, respectively. The experiment results on urban dataset show that the proposed method is significantly better than Region Growing algorithm [12] and SECOND [18] detector for detecting remote obstacles, adjacent obstacles and obstacles outside the training set. The experiment results on KITTI dataset show that the proposed method outperforms other state-of-the-art methods in AP and AOS. Additionally, the proposed method adopts Grid-SECOND detector to reduce time-consuming and guarantees that the time cost of the proposed method after fusing Double-Layer Region Growing algorithm is similar to the SECOND detector, while increasing the overall obstacle detection accuracy. The overall method takes about 50ms and meets the real-time requirements of intelligent vehicle, which can provide more reliable obstacle information for the perception system.

Although the proposed method can distinguish adjacent obstacles by means of the second Region Growing algorithm with small grid size, it cannot accurately distinguish two obstacles with overlapping height. In the next step of work, we will divide the point cloud of obstacle with large volume into voxel grids, and use Region Growing algorithm for the voxel grids. It can save time cost while improving the refinement accuracy of large obstacles.

## REFERENCES

[1] K. Uehara, H. Saito, and K. Hara, "Line-based SLAM considering directional distribution of line features in an urban environment," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, vol. 6, 2017, pp. 255–264.

[2] L. Stanislas and M. Dunbabin, "Multimodal sensor fusion for robust obstacle detection and classification in the maritime RobotX challenge," *IEEE J. Ocean. Eng.*, vol. 44, no. 2, pp. 343–351, Apr. 2019.

[3] W. Shicai, "Obstacle detection and tracking for intelligent vehicle based on density characteristics of point cloud using 3D lidar," *J. HEF Univ. Technol.*, vol. 42, no. 10, pp. 1311–1317, May 2019.

[4] H. Yin, Y. Wang, X. Ding, L. Tang, S. Huang, and R. Xiong, "3D LiDAR-based global localization using siamese neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1380–1392, Apr. 2020.

[5] M. Labbé and F. Michaud, "RTAB-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *J. Field Robot.*, vol. 36, no. 2, pp. 416–446, Mar. 2019.

[6] P. Di, S. O. Automation, and H. D. University Kinect, "Obstacle detection algorithm based on DBSCAN and aradient partition," *Electronicence Technol.*, vol. 32, no. 1, pp. 86–90, 2019.

[7] J. Liu, "Negative obstacle detection in unstructured environment based on multiple LiDAR and compositional features," *Jiqiren/Robot*, vol. 39, no. 5, pp. 638–651, 2017.

[8] A. H. Lang, S. Vora, H. Caesar, and L. Zhou, "PointPillars: Fast encoders for object detection from point clouds," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, May 2019, pp. 12697–12705.

[9] Z. Wenkai, H. Fang, and K. Weijian, "Point-selective Geetest CAPTCHA Recognition Based on Faster-RCNN," *Electron. Sci. Technol.*, vol. 32, no. 9, pp. 42–45, 2019.

[10] N. A. Stanley, G. West, and D. Belton, "Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data," *Pattern Recognit.*, vol. 48, no. 4, pp. 1404–1419, 2015.

[11] L. Feng, C. Xihua, K. Liu, F. Tang, and Q. Meng, "GO-DBSCAN: Improvements of DBSCAN algorithm based on grid," *Int. J. Comput. Theory Eng.*, vol. 9, no. 3, pp. 151–155, 2017.

[12] X. Desheng, X. Youchun, and W. Rendong, "Obstacle detection and tracking for unmanned vehicles based on 3D laser radar," *J. Mil. Transp. Univ.*, vol. 2018, no. 8, pp. 952–959, 2018.

[13] S. Benyue, M. Jinyu, and P. Yusheng, "Algorithm for RGBD point cloud denoising and simplification based on K-means clustering," *J. Syst. Simul.*, 2016, vol. 28, no. 10, pp. 2329–2334.

[14] R. Q. Charles, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. Comput. Vis. Pattern Recognit. (CVPR*, 2017, vol. 1, no. 4, p. 7.

[15] C. Qi, L. Yi, H. Su, and L. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[16] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," *Robot., Sci. Syst.*, pp. 1–4, Apr. 2016.

[17] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," *CoRR*, vol. abs/1711.06396, pp. 1–4, Dec. 2017.

[18] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, Oct. 2018.

[19] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019.

[20] L. Jiong, Z. Kai, B. Rui, Z. Yuan, and X. Youchun, "Urban ground segmentation algorithm based on ray slope threshold," *Acta Opt. Sinica*, vol. 39, no. 9, pp. 352–360, 2019.

[21] D. Han, Y. Xu, H. Li, D. Xie, and W. Chen, "Calibration of extrinsic parameters for three-dimensional lidar based on hand-eye model," *Opto-Electron. Eng.*, vol. 44, no. 8, pp. 798–804, 2017.

[22] C. Y. W. Jingtao, "Three-dimensional reconstruction of point cloud based on graph optimization method," *J. Mil. Transp. Univ.*, vol. 2017, no. 9, pp. 85–90, 2017.

[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[24] L. Xuezhu, *Study on the Influence of DPM Model Parameters on Pedestrian Detection Results*. ShanDong, China: Qufu Normal Univ., 2017.