Hindawi
Journal of Sensors
Volume 2019, Article ID 8473980, 9 pages
https://doi.org/10.1155/2019/8473980

Research Article

Real-Time Vehicle Detection Algorithm Based on Vision and Lidar Point Cloud Fusion

Hai Wang,[1] **Xinyu Lou**,[1] **Yingfeng Cai**,[2] **Yicheng Li**,[2] **and Long Chen**[2]

[1]*School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang 212013, China*
[2]*Institute of Automotive Engineering, Jiangsu University, Zhenjiang 212013, China*

Correspondence should be addressed to Yingfeng Cai; caicaixiao0304@126.com

Received 11 November 2018; Revised 27 December 2018; Accepted 5 March 2019; Published 17 April 2019

Academic Editor: Jaime Lloret

Vehicle detection is one of the most important environment perception tasks for autonomous vehicles. The traditional vision-based vehicle detection methods are not accurate enough especially for small and occluded targets, while the light detection and ranging- (lidar-) based methods are good in detecting obstacles but they are time-consuming and have a low classification rate for different target types. Focusing on these shortcomings to make the full use of the advantages of the depth information of lidar and the obstacle classification ability of vision, this work proposes a real-time vehicle detection algorithm which fuses vision and lidar point cloud information. Firstly, the obstacles are detected by the grid projection method using the lidar point cloud information. Then, the obstacles are mapped to the image to get several separated regions of interest (ROIs). After that, the ROIs are expanded based on the dynamic threshold and merged to generate the final ROI. Finally, a deep learning method named You Only Look Once (YOLO) is applied on the ROI to detect vehicles. The experimental results on the KITTI dataset demonstrate that the proposed algorithm has high detection accuracy and good real-time performance. Compared with the detection method based only on the YOLO deep learning, the mean average precision (mAP) is increased by 17%.

1. Introduction

The core technology of the unmanned driving includes the environmental perception, precise positioning, and path planning. The complex road environments, especially the mixed traffic environments, cause great difficulties to the environment perception of the autonomous vehicles. Moreover, vehicle detection is an important part of the environment perception and plays a vital role in the safe driving of the unmanned ground vehicle (UGV).

Currently, the mainstream obstacle detection sensors are camera and lidar. Cameras have been widely used in the intelligent driving because of their low cost and the ability to obtain textures and colors of targets, which are especially important in recognition of the traffic lights and traffic signs. In [1], vehicles were detected based on the ground plane assumption. In [2, 3], the histogram of oriented gradient (HOG) and Haar features were, respectively, used to detect the vehicles. In [4], a scale-insensitive convolution neural network (SINet) was proposed to detect vehicles, and vehicles of different sizes were detected in the images. However, a single camera cannot obtain depth information and is greatly affected by lighting conditions.

On the other hand, lidar can acquire distance and three-dimensional information at a long detection distance. Lidar is not affected by the illumination conditions and has strong robustness, so it has been widely used in the environmental perception. In [5], the geometry-based target recognition was used, but that method was highly influenced by the environmental conditions, such as the presence of bushes, which could be misidentified. Moreover, in [6], lidar was used to detect and classify pedestrians. The point clouds acquired by the lidar were projected onto the grid and clustered by the global nearest neighbor (GNN), and for each candidate, its eigenvector was calculated and classified by the support vector machine (SVM) based on the radical basis function (RBF) kernel. In [7], the heuristic algorithm based on the shape and location of the road network was used, and the

clustered data were processed to select the objects that could be vehicles. In [8], pedestrians and cyclists on the road are classified by saliency network. In [9], the local features were extracted from 3D voxels and the targets were classified by the decision trees. In [10], the deep neural network model is used to classify objects in multiscale. On the one hand, in [11], the vehicles and pedestrians were classified using the Gaussian hybrid model classifier (GMM classifier). In [12], a 3D object recognition method that does not require segmentation was proposed. The 3D Haar-like local features were used instead of the global features, and the local features were combined with the AdaBoost training. In order to improve the efficiency of the operation, a 3D summation area table was used, and to reduce the false alarm, the classifier was retrained multiple times using the false detection as negative samples. However, the cost of the used lidar was very high, the amount of calculation was large, the color information could not be obtained, and the resolution decreased with the increase in the distance; besides, only very scarce information on the long-distance objects could be obtained.

The usage of the multisensor fusion scheme that combines camera and lidar in the UGV has been gradually increasing. There are three mainstream integration schemes: (1) target detection using lidar and camera separately and then merging the results [13, 14], (2) target detection using the images obtained by the camera and then using the lidar for the confirmation [15], and (3) target detection by using the lidar to determine the region of interest and the camera for target detection [16, 17]. In [14], the obstacles in the image and point cloud were processed separately to identify and classify the pedestrians. Then, the target matching was carried out, and the results were merged for pedestrians with the same detection and classification results. In [15], targets were extracted from the image obtained by the camera, and then vehicles were detected, and hypotheses were generated by the Haar-like features and AdaBoost, and after that, the hypotheses were confirmed by lidar. In [16], the authors used lidar to determine the region of interest and the SVM to classify the target in the image, but the proposed method could detect only the rear features of a vehicle, and the detection rate was low. In [17], vehicle features were extracted from the side view of a tire, so the detection rate for the vehicles parallel to the autonomous vehicle was low. The main idea of this paper is the same as that of [16, 17], which is to use lidar to extract the region of interest, and then to use the YOLO, which has high detection accuracy, to detect the objects in the ROI of the images.

In this paper, the map is constructed by the max-min elevation map from point cloud, which is clustered by eight connected region markers, and then, morphological expansion is applied to the clustering results. After that, the minimum rectangle of each connected domain is calculated and points from these rectangles are projected onto the image. After projection, the coordinate extremum of each rectangle on the image is obtained. The area is enlarged and merged to obtain the area of interest in the image, and finally, the YOLO is used to classify the obstacles. The process is shown in Figure 1.

2. Region of Interest Extraction

2.1. Build Grid Map. The Velodyne HDL-64 lidar used in this paper returns about 1.3 million points per second. The huge amount of data helps the environment perception of the UGV, but it poses a great challenge to the real-time algorithm performance. In response to this problem, this paper uses the max-min elevation map for environment creation. At the distance of d_f, in front of the vehicle, an $M \times N$ grid map with the bottom left corner as the origin is established where each cell is a square with the side length d_c. We set M to 320, N to 160, and d_c to 0.2 m. The projection of the lidar coordinate system onto the grid coordinate system is given as follows:

$$P_m = \left\lfloor \frac{(y_l - d_f)}{d_c} \right\rfloor + 1,$$
$$P_n = \left\lfloor \frac{(x_l + (N/2) * d_c)}{d_c} \right\rfloor + 1, \quad (1)$$

where (x_l, y_l) is the lidar returned point in the lidar coordinate system. The lidar coordinate system takes the vehicle direction as the y-axis, the vertical direction as the z-axis, and direction that is perpendicular to the forward direction to the right as the x-axis. P_m and P_n correspond to the grid coordinates of the point. When points P_m and P_n satisfy the conditions $M < P_m$ or $P_m < 0$ and $N < P_n$ or $P_n < 0$, the point is dropped. Example of grid map is shown in Figure 2.

After traversing all the points, the points within the grid map are projected onto the corresponding grid, and the height data is preserved. We consider that there is an obstacle when the difference between the maximum height and the minimum height of a single grid is greater than the set threshold. The grid map provides a computationally efficient approximation to the terrain gradient in a cell [18] and the amount of data we need to deal with is greatly reduced from p to $M * N$, where p is the amount of data returned from lidar.

Considering that the distance between the scanning lines of a multiline lidar increases with the increase in the scanning distance, an incomplete scanning of obstacles can be caused, especially in the vertical direction, and the distant obstacles can be scanned only by several scanning lines. Therefore, this paper uses the morphological expansion operation for obstacle cells in the grid map. The matrix we use is given as follows:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (2)$$

Taking the scene shown in Figure 3 as an example, the top view of the original point cloud, the created grid map, and the grid map after expansion operation shown in Figures 4, 5, and 6, respectively, are obtained.

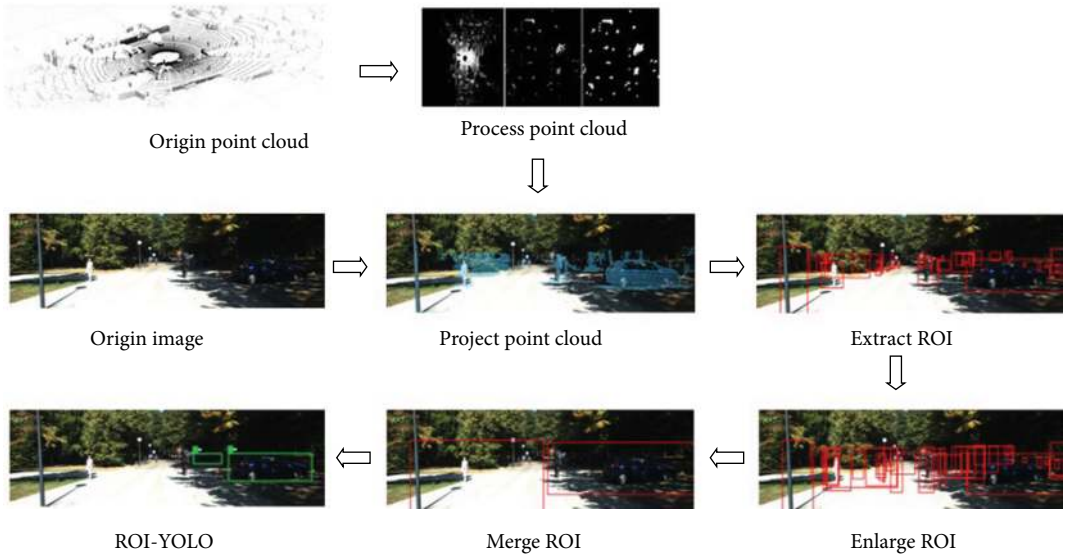

Figure 1: Algorithm flowchart.

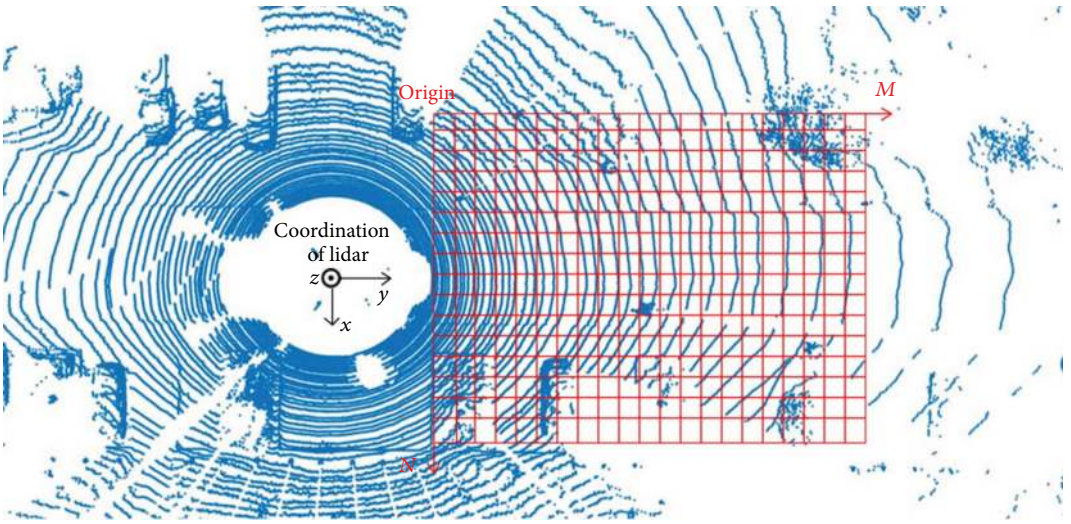

Figure 2: Example of grid map.

Figure 3: Example scenario.

2.2. Grid Clustering. In this paper, eight connected region markers are applied to clustering. If an obstacle cell is connected with the other obstacle cells in the upper, lower, left, right, upper-left, lower-left, upper-right, or lower-right corner, these cells are considered to belong to the same obstacle, so they are represented by the same number.

2.3. Projection from Grid Map to Image. We first convert the grid coordinate extremum $x_{g\ \min}, x_{g\ \max}, y_{g\ \min}, y_{g\ \max}$ of each obstacle to the lidar coordinate system $x_{l\ \min}, x_{l\ \max}, y_{l\ \min}, y_{l\ \max}$ and then extract all the obstacle points (x_{li}, y_{li}, z_{li}) from the point cloud as follows:

$$\begin{aligned} x_{l\ \min} < x_{li} < x_{l\ \max}, \\ y_{l\ \min} < y_{li} < y_{l\ \max}, \end{aligned} \tag{3}$$

where l denotes the obstacle number. After that, we project the extracted points to the image, where p_{in} is the input point cloud and E is the projection matrix of the point cloud to the image. Since the dimension of p_{in} is $n \times 3$ and the dimension of E is

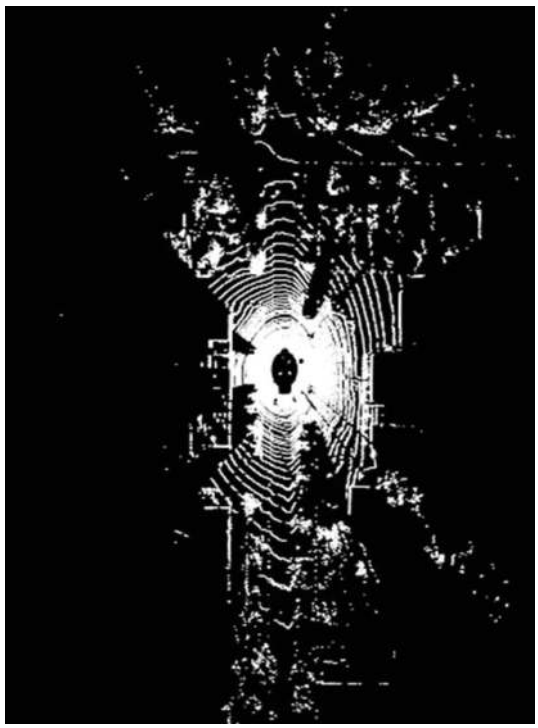

Figure 4: Original point cloud.

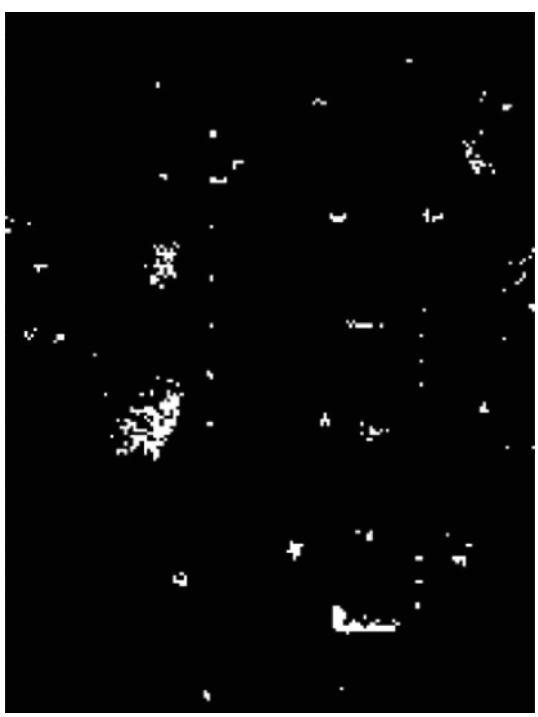

Figure 5: Grid map.

3×4, it is necessary to fill 1 in the last column of p_{in} to obtain p_{in1}. The projection formula is given as follows:

$$p_{\text{out1}} = \left(E * p_{\text{in1}}^{\text{T}}\right)^{\text{T}}, \tag{4}$$

Figure 6: Grid map after expansion.

Figure 7: Obstacle point cloud projection image.

where p_{in1} denotes the matrix of input point cloud after extension with 1 in the last column and p_{out1} denotes the nonnormalized homogeneous coordinates of obstacle points.

The normalized homogeneous coordinates are obtained as follows:

$$p_{\text{out}} = \frac{L(p_{\text{out1}})_{1,2}}{L(p_{\text{out1}})_3 * [1\ 1]}, \tag{5}$$

where $L(p_{\text{out1}})_{1,2}$ represents a new matrix consisting of the first and second column elements of p_{out1}, and its dimension is $n \times 2$; $L(p_{\text{out1}})_3$ represents a new matrix composed of elements of the third column of p_{out1}, and its dimension is $n \times 1$. The dimension of p_{out} is $n \times 2$ and it denotes a set of the projection coordinates of the point cloud on the image. The image coordinate system takes the upper-left corner of the image as the origin, the right direction as the x-axis positive direction, and the downward direction as the y-axis positive direction.

The projection effect is shown in Figure 7.

Figure 8: Original region of interest.

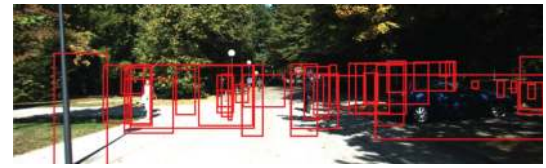

Figure 9: Enlarged region of interest.

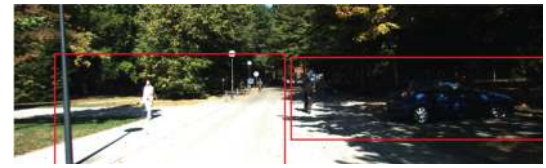

Figure 10: Merged region of interest.

2.4. Extract Regions of Interest. From the coordinates of the points of p_{out}, the extremum of the horizontal and vertical coordinates of each obstacle in the image coordinate system $x^i_{p\ \min}$, $x^i_{p\ \max}$, $y^i_{p\ \min}$, $y^i_{p\ \max}$ can be obtained. We use the rectangular region formed by the four extreme values that indicate the region of interest of the obstacle in the image, as shown in Figure 8. Since for the lidar, the farther the scanning distance is, the lower the density of the point cloud is, and when the obstacle is far away from the UGV, the lidar can only scan a part of the obstacle. At the same time, the scanning point of the lidar cannot penetrate the obstacle, so the occlusion between the obstacles can also cause the same problem. If only the abovementioned method for the region of interest extraction is used, often, only a part of the target obstacle is extracted, which affects obstacle classification. On this basis, this paper proposes a region of interest amplification method based on the dynamic threshold.

For obstacle i, the average distance between the lidar scanning point belonging to the obstacle and the UGV labeled as $d_{\text{mean}i}$ is calculated, and the threshold is given as follows:

$$h = \frac{M * d_c}{M * d_c - d_{\text{mean}i}}. \tag{6}$$

The parameters of the rectangular region of interest that the obstacle belongs to are given as follows:

$$\begin{aligned} x^i_{\text{roimin}} &= x^i_{p\ \min} - h * 3, \\ y^i_{\text{roimin}} &= y^i_{p\ \min} - h * 3, \\ \text{width}^i_{\text{roi}} &= \left(x^i_{p\ \max} - x^i_{p\ \min}\right) + 6 * h, \\ \text{height}^i_{\text{roi}} &= \left(y^i_{p\ \max} - y^i_{p\ \min}\right) + 6 * h, \end{aligned} \tag{7}$$

where $\text{width}^i_{\text{roi}}$ is the rectangle width, and $\text{height}^i_{\text{roi}}$ is the rectangle height. The rectangular frame in Figure 9 denotes the enlarged region of interest.

The region of interest will overlap after enlargement. In this paper, the overlapped rectangles are merged into one region of interest whose parameters are given as follows:

$$\begin{aligned} \text{width}^i_{\text{roi}} &= \max\left(x^{i1}_{p\ \min} + \text{width}^{i1}_{\text{roi}}, x^{i2}_{p\ \min} + \text{width}^{i2}_{\text{roi}}\right) - \min\left(x^{i1}_{p\ \min}, x^{i2}_{p\ \min}\right), \\ \text{height}^i_{\text{roi}} &= \max\left(y^{i1}_{p\ \min} + \text{height}^{i1}_{\text{roi}}, y^{i2}_{p\ \min} + \text{height}^{i2}_{\text{roi}}\right) - \min\left(y^{i1}_{p\ \min}, y^{i2}_{p\ \min}\right), \\ x^i_{\text{roimin}} &= \min\left(x^{i1}_{p\ \min}, x^{i2}_{p\ \min}\right), \\ y^i_{\text{roimin}} &= \min\left(y^{i1}_{p\ \min}, y^{i2}_{p\ \min}\right). \end{aligned} \tag{8}$$

The merged region of interest is shown in Figure 10.

3. Obstacle Classification by YOLO

The YOLO is a new target detection algorithm. It uses a single neural network to predict the bounding box and class probability directly from the complete image by only one evaluation. Moreover, it can directly optimize the detection performance end-to-end, so it has a high real-time performance. The basic YOLO model can process the image in real-time at the speed of 45 frames/sec while achieving twice larger mAP than the other common real-time detectors. However, the YOLO has a certain positioning error [19]. In this work, the YOLO v3 is used to classify the obstacles. The YOLO v3 combines various advanced methods to overcome the shortcomings (low detection accuracy, lousy performance at detecting small objects, etc.) of the previous two generations of the YOLO algorithm and improves the

detection accuracy guaranteeing excellent real-time performance. The YOLO v3 is based on the idea of the ResNet, and it combines the Darknet19 network of the YOLO2 to propose a new feature extraction network named the Darknet-53. Regarding the classification accuracy, the Darknet-53 is close to the ResNet-101 and ResNet-152, but much faster [20]. The running time of the YOLO v3 on 320 $\times$ 320 pixels images is 22.2 ms, reaching 28.2 mAP, which is as accurate as the single shot multibox detector (SSD), but three times faster [21]. However, the YOLO shows a certain decrease in the detection rate when the target is dense and when there is occlusion.

3.1. Dataset Selection and Processing. In this work, we used the KITTI [22] dataset to train and verify the YOLO network. The KITTI dataset was cofounded by the German Karlsruhe Institute of Technology (KIT) and the Toyota Institute of Technology to provide a dataset for the computer vision algorithms in an autonomous driving environment. The KITTI dataset contained eight obstacle types: cars, vans, trucks, pedestrians in standing and sitting positions, cyclists, trams, and others. In this work, cars, vans, and trucks were considered to be the same type of obstacles (vehicles), and the other types of obstacles were ignored.

3.2. Network Training. Before the training, we created the training set and verification set. We used the random selection method to divide all of the 7481 images in the KITTI dataset into the training set and verification set by the ratio of 7 : 3. Size of images provided by KITTI is 1242 $*$ 375.

The main parameters of the experimental platform were CPU—Intel Xeon E5-2687W V4 at 3.00 GHz, memory—128 G, and GPU—NVIDIA Quadro M4000. The deep learning used the Keras platform.

The training parameters of the YOLO v3 are given in Table 1.

Table 1: Training parameters of the YOLO v3.

Parameter	Value
Batch size	2
Learning rate	10^{-4}
Ignore thresh	0.5
Number of epochs	100

Table 2

(a) Result of the original YOLO v3

Benchmark	Easy	Moderate	Hard
Car (detection)	58.56%	43.38%	38.23%

(b) Result of the proposed method

Benchmark	Easy	Moderate	Hard
Car (detection)	70.58%	62.71%	55.17%

4. Results

4.1. Overall Results. In the test, the test set provided by the KITTI was used. The result of the original YOLO v3 algorithm is shown in Table 2(a). The result of vehicle detection using the proposed method is shown in Table 2(b). The precision and recall chart of the YOLO v3 algorithm and the proposed algorithm for vehicle detection on the KITTI test set are presented in Figures 11(a) and 11(b), respectively.

4.2. Comparison of Experimental Results in Sample Scenarios. Figures 12, 13, and 14 show some of the scenarios in the experiment. As it can be seen in these figures, there was a large number of vehicles in each scene, and the occlusion was severe. Figure 12(b) shows the results of vehicle detection using the YOLO v3 algorithm. Figure 12(c) shows the results of vehicle detection using the proposed algorithm.

5. Discussion

The experimental results showed that the proposed algorithm significantly improved the vehicle detection accuracy at different detection difficulty levels compared to the original YOLO v3 algorithm, especially for the vehicles with severe occlusion. Under easy, moderate, and hard difficulties, the average accuracy (AP) improvement was by nearly 12%, 20%, and 17%, respectively. We calculated the proportion of the area of all ROIs to the total area of images on the training dataset in KITTI, which is 55%, and the proportion of the number of ground truths contained in ROIs to the total number of ground truths in the training dataset, which is 96%. At the same time, because the use of grid map for the purpose of reducing dimensions of point clouds and the selected point clouds is only in the range of 64 m*32 m in front of the vehicle, the computational complexity of the algorithm is greatly reduced. Therefore, the time consumption of the proposed algorithm does not increase much compared to YOLO v3. We tested it on our experimental platform with 1000 frames of data which randomly selected from KITTI dataset, and the average processing time per frame of the proposed algorithm was about 0.09 s, while of the YOLO v3 algorithm was 0.05 s.

We selected 50 easy, moderate, and difficult images, respectively, from the KITTI test set and compared the obtained results with the results given in [16, 17] as shown in Table 3.

The experimental results show that the proposed algorithm has great advantages under various difficulty conditions compared with the algorithms proposed in [16, 17].

However, the proposed algorithm still has certain shortcomings. For instance, when the target vehicle is far away from the UGV, it cannot be scanned by the lidar, so the algorithm cannot detect it. At the same time, if a large vehicle is close to the autonomous vehicle, such as a container truck, it is possible that only a portion of that vehicle would be detected after enlarging the ROI area so that the algorithm could fail to identify it.

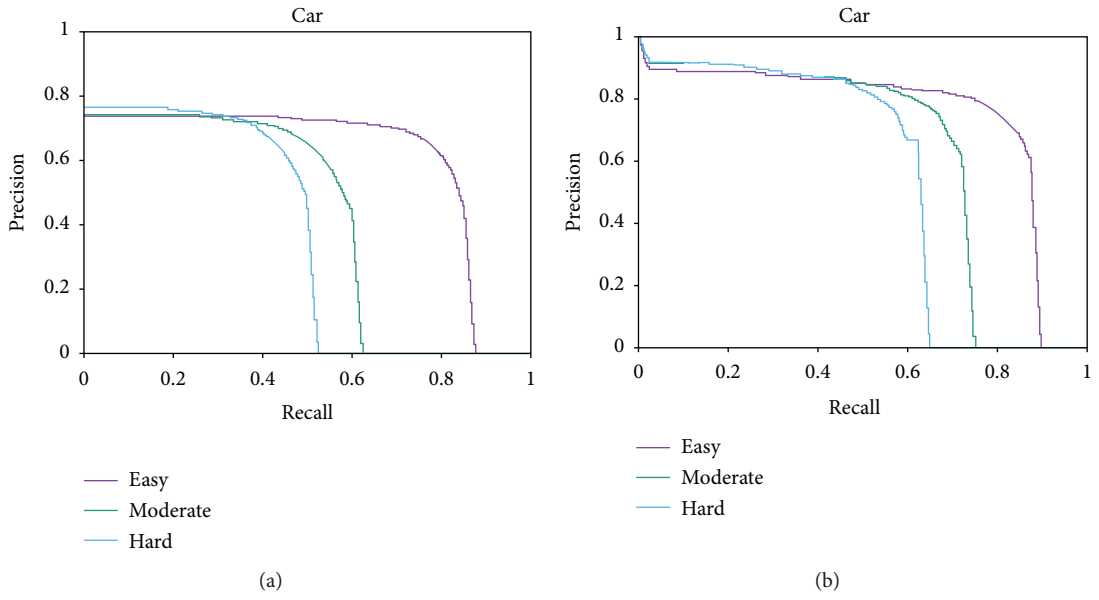

Figure 11: Test results on the KITTI dataset. (a) Precision and recall chart of vehicle detection on the KITTI test set of the YOLO v3 algorithm. (b) Precision and recall chart of vehicle detection on the KITTI test set of our algorithm.

Figure 12: The experimental results of Scenario 1. (a) Scenario 1. (b) Results of vehicle detection using the YOLO v3 algorithm. (c) Results of vehicle detection using our algorithm.

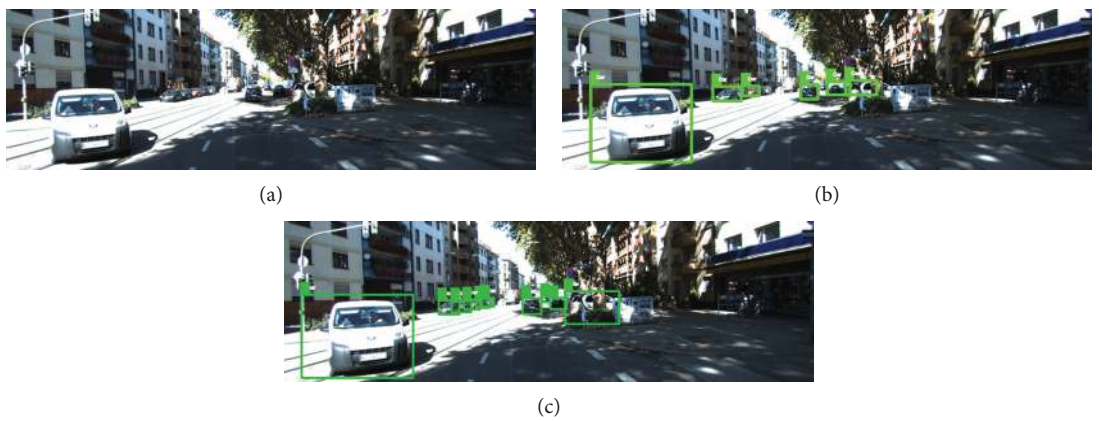

Figure 13: The experimental results of Scenario 2. (a) Scenario 2. (b) Results of vehicle detection using the YOLO v3 algorithm. (c) Results of vehicle detection using our algorithm.

(a) (b)

(c)

Figure 14: The experimental results of Scenario 3. (a) Scenario 3. (b) Results of vehicle detection using the YOLO v3 algorithm. (c) Results of vehicle detection using our algorithm.

Table 3: Comparison of the results.

Method	AP (%) Easy	Moderate	Hard
Reference [16]	46.28	39.87	24.32
Reference [17]	41.72	33.49	19.01
This work	69.36	61.33	46.27

6. Summary and Outlook

A vehicle detection method based on the multisensor fusion is proposed in this paper. Using the calibration relationship between the lidar and camera, the region of interest extracted by the lidar is projected into the image obtained by the camera, and the region of interest in the image is obtained and processed. Finally, the YOLO v3 algorithm is used to detect the vehicle in the region of interest. The effectiveness of the proposed algorithm is verified by experiments.

In our future work, we will optimize the extraction of the region of interest to achieve better target extraction. At the same time, different ROI sizes mean different input image sizes. Although they will be resized into images of the same size in the beginning of YOLO, the proportion of area of obstacles on the image will change greatly, which will make the detection accuracy of the model decrease. To solve the problem, we will improve the YOLO algorithm or use some other more targeted networks to achieve better detection accuracy.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

H.W. and Y.C. did the methodology, X.L. and Y.L. worked on the software, and L.C. worked on the project administration of the study.

Acknowledgments

This research was funded by the National Key Research and Development Program of China (2018YFB0105003), the National Natural Science Foundation of China (U1764264, 51875255, U1664258, U1764257, 61601203, and 61773184), the Key Research and Development Program of Jiangsu Province (BE2016149), Natural Science Foundation of Jiangsu Province (BK20180100) the Key Project of the Development of Strategic Emerging Industries of Jiangsu Province (2016-1094, 2015-1084), the Key Research and Development Program of Zhenjiang City (GY2017006), and the Overseas Training Program for Universities of Jiangsu Province.

References

[1] G. Kim and J. S. Cho, "Vision-based vehicle detection and inter-vehicle distance estimation," in *2012 12th International Conference on Control, Automation and Systems*, JeJu Island, South Korea, October 2012.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pp. 886–893, San Diego, CA, USA, June 2005.

[3] T. Mita, T. Kaneko, and O. Hori, "Joint Haar-like features for face detection," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pp. 1619–1626, Beijing, China, October 2005.

[4] X. Hu, X. Xu, Y. Xiao et al., "SINet: a scale-insensitive convolutional neural network for fast vehicle detection 2018," http://arxiv.org/abs/1804.00433.

[5] M. Darms, P. Rybski, and C. Urmson, "Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments," in *2008 IEEE Intelligent Vehicles Symposium*, pp. 1197–1202, Eindhoven, Netherlands, June 2008.

[6] H. Wang, B. Wang, B. Liu, X. Meng, and G. Yang, "Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle," *Robotics and Autonomous Systems*, vol. 88, pp. 71–78, 2017.

[7] S. Kammel and B. Pitzer, "Lidar-based lane marker detection and mapping," in *2008 IEEE Intelligent Vehicles Symposium*, pp. 1137–1142, Eindhoven, Netherlands, June 2008.

[8] Y. Cai, Z. Liu, H. Wang, and X. Sun, "Saliency-based pedestrian detection in far infrared images," *IEEE Access*, vol. 5, pp. 5013–5019, 2017.

[9] P. Babahajiani, L. Fan, and M. Gabbouj, "Object recognition in 3D point cloud of urban street scene," in *Computer Vision - ACCV 2014 Workshops. ACCV 2014. Lecture Notes in Computer Science, vol 9008*Springer, Cham.

[10] H. Wang, L. Dai, Y. Cai, X. Sun, and L. Chen, "Salient object detection based on multi-scale contrast," *Neural Networks*, vol. 101, pp. 47–56, 2018.

[11] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, "A lidar and vision-based approach for pedestrian and vehicle detection and tracking," in *2007 IEEE Intelligent Transportation Systems Conference*, pp. 1044–1049, Seattle, WA, USA, September-October 2007.

[12] G. Pang and U. Neumann, "Training-based object recognition in cluttered 3D point clouds," in *2013 International Conference on 3D Vision - 3DV 2013*, pp. 87–94, Seattle, WA, USA, June-July 2013.

[13] S. Hwang, N. Kim, Y. Choi, S. Lee, and I. S. Kweon, "Fast multiple objects detection and tracking fusing color camera and 3D LIDAR for intelligent vehicles," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 234–239, Xi'an, China, August 2016.

[14] T. E. Wu, C. C. Tsai, and J. I. Guo, "LiDAR/camera sensor fusion technology for pedestrian detection," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1675–1678, Kuala Lumpur, Malaysia, December 2017.

[15] F. Zhang, D. Clarke, and A. Knoll, "Vehicle detection based on LiDAR and camera fusion," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1620–1625, Qingdao, China, October 2014.

[16] J. P. Hwang, S. E. Cho, K. J. Ryu, and S. Park, "Multi-classifier based LIDAR and camera fusion," in *2007 IEEE Intelligent Transportation Systems Conference*, pp. 467–472, Seattle, WA, USA, September-October 2007.

[17] K. Cho, S. H. Baeg, K. Lee, H. S. Lee, and S. D. Park, "Pedestrian and car detection and classification for unmanned ground vehicle using 3D lidar and monocular camera," in *Proceedings Volume 8045, Unmanned Systems Technology XIII; 804507*, Orlando, FL, USA, May 2011.

[18] B. Douillard, J. Underwood, N. Melkumyan et al., "Hybrid elevation maps: 3D surface models for segmentation," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, Las Vegas, NV, USA, June 2016.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[21] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," 2018, http://arxiv.org/abs/1804.02767.

[22] J. Fritsch, T. Kuhnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 1693–1700, The Hague, Netherlands, October 2013.