A Computationally Efficient Solution for LiDAR-based Obstacle Detection in Autonomous Driving

Shengjie Chen, Rihui Song, Shixiong Chen, Wenjun Li and Kai Huang

***Abstract*— Precise perception is essential for autonomous driving, where obstacle detection is a very important part. The vision-based approaches are susceptible to light, and other commonly used sensors in autonomous driving such as millimeter-ware radars are not satisfied in distance and range, while the LiDAR (Light Detection and Ranging) has strong adaptability to the environment, becoming a standard sensor on an autonomous vehicle. In this paper, we propose a computationally efficient obstacle detection method based on Velodyne HDL-32E, a 3D LiDAR which can generates 700,000 points per second. To overcome this challenge from such huge amount of data, dimensionality reduction through the projection is introduced, combined with efficient and highly available detection methods, to meet the accuracy as well as real-time requirements. Experiments show that our obstacle detection method can meet the real-time requirement on the vehicle devices with limited resources and provide reliable obstacles information for our autonomous driving vehicle.**

***Keywords*— LiDAR, Cartesian/Polar Projection Model, Obstacle Detection, Autonomous Driving**

I. Introduction

This paper presents a computationally efficient solution for obstacle detection using 3D LiDAR data. Obstacle detection is a critical pre-processing step in a number of autonomous perception tasks. Douillard et al. [1] made a good summary of methods based on 3D point clouds' grid-based statistical characteristics. Mean elevation can be chosen as the basis for the ground extraction from 3D LiDAR data but it cannot capture overhanging structures. While Min-Max elevation, proposing by bias between maximum and minimum height, can be chosen as the basis for the obstacle detection, but sensitive to noise.

Some other methods based on 3D point clouds' geometric features were introduced in the past several years. Line-fit ground plane segmentation was put forth by Himmelsbach et al. [2] They treated the point clouds in polar model, instead of cartesian model mentioned above, and took advantage of the distribution of the points. On the other hand, a typical plane-fit technique was introduced from Zermas et al. [3] for the fast extraction of the ground points.

Recently, some deep learning techniques are applying to LiDAR-based detection tasks. BoLi used a single 2D end-to-end fully convolutional network, taking LiDAR bird view as input, to detect vehicle, leading to an excellent performance [4], followed by his new approach using 3D end-to-end fully convolutional network [5]. Xiaozhi et al. [6] proposed a Multi-View 3D object detection network (MV3D) which takes multimodal data, both LIDAR point cloud and RGB images as input, and predicts the full 3D extent of objects in 3D space.

All state-of-art obstacle detection methods using 3D LiDAR data mentioned so far, some cannot reach the real-time requirements even offline testing, some lack of challenges from online testing. Besides, most of them only consider about features in current scan, ignoring features in historical scans. With work presented in this paper, we put forward a computationally efficient solution for obstacle detection using 3D LiDAR data. In order to pull up the detection accuracy, we do fuse the results from historical scans. Experiments have been conducted to find the effect of two kinds of projection, both cartesian model and polar model, and several kinds of grid size, from small to large, on our approach. Optimal projection as well as grid size has been applied to our autonomous driving vehicle. Over 2,000 miles of road test shows that our obstacle detection approach is highly available for autonomous driving.

The remainder of the paper is organized as follows. We give a detailed description of our approach in section II and III, from obstacle detection in current scan to fusion with historical detection results. In section IV we provide experimental results and conclude the paper in section V.

II. Current Scan Processing

To meet the accuracy as well as real-time requirements, both spatial and temporal features of point clouds are considered. In spatial features, we need to put forth an efficient detection method in current scan processing.

A. Dimensionality Reduction through the Projection

Formally, current scan at time t will be denoted by an unordered set $P_t = \{p_1, p_2, ..., p_{N_t}\}$ with 3D points $p_i = (x_i, y_i, z_i)$ given by LiDAR's cartesian coordinate and the number of points, N_t, in current scan.

For a greater flexibility in controlling the run-time of the algorithm, we project each 3D point p_i into corresponding grid cell in Cartesian Model or Polar Model. Thus, the complexity of our obstacle detection approach will not depend on the size of the point clouds, but our selected parametes.

a) Cartesian Model Projection: This Cartesian Model projection has been applied in many grid-based segmentation since Stanley, an autonomous car from Standford, that utilized this strategy to build his 2D terrain occupancy map and won

Shengjie Chen, Rihui Song, Shixiong Chen, Wenjun Li and Kai Huang are with the School of Data and Computer Science, Sun Yat-sen University, China.

Shengjie Chen, Rihui Song, Wenjun Li and Kai Huang are with the Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education, China.

the DARPA Grand Challenge in 2005 [7]. As shown in Fig. 1, the xy-plane is divided into number of $\Delta\alpha \times \Delta\alpha$ squares, with number of front rows $H(ead)$, number of rear rows $B(ack)$, number of columns on both sides $2\times W(idth)$ and W columns for each side.

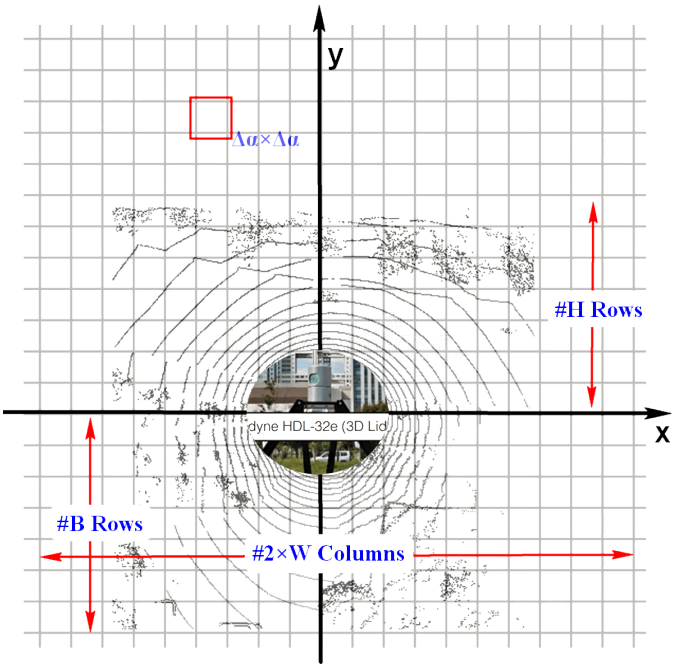


Fig. 1: Cartesian Model Projection

H, B, W are configured for the focus field of view, influenced by navigation planner, like maximum control spedd, etc. The focus field of view is -$W\Delta\alpha \leq x \leq W\Delta\alpha$, -$B\Delta\alpha \leq y \leq H\Delta\alpha$.

Given a 3D points $p_i = (x_i, y_i, z_i)$, the corresponding grid cell is (x_{idx}, y_{idx}), where,

$$x_{idx} = \frac{W\Delta\alpha - x_i}{\Delta\alpha}; \tag{1}$$

$$y_{idx} = \frac{y_i}{\Delta\alpha} + B, \tag{2}$$

from the lower left to the upper right.

b) Polar Model Projection: As shown in Fig. 2, the grid size is different, with inner ones smaller than outer ones. By contract, the Cartesian Model shares the same grid size.

The Polar Model is closer to LiDAR's physical model, and it treats distant and near objects with different resolution while reducing computations. Using smaller grid size, this Polar Model can capture close obstacles even little objects.

As shown in Fig. 2, the xy-plane is represented as a circle of radius between R_{min} and R_{max}, the focus field of view. You can even only care about the forward vision, [-10°, 190°] in polar coordinate, or we say a 200° perspective. We introduce parameter $\Delta\theta$ to describe the angle one segment covers, so we come up with $M = \frac{2\pi}{\Delta\theta}$ segments. The index to a segment that a point $p_i = (x_i, y_i, z_i)$ maps to is,

$$segment_{idx} = \frac{atan2(y_i, x_i)}{\Delta\theta}. \tag{3}$$

Besides, we introduce parameter Δr to describe the radius one bin covers, also come up with $B = \frac{R_{max} - R_{min}}{\Delta r}$ bins. The index to its corresponding bin is,

$$bin_{idx} = \begin{cases} \frac{\sqrt{x_i^2+y_i^2} - R_{min}}{\Delta r}, & R_{min} \leq \sqrt{x_i^2 + y_i^2} \leq R_{max} \\ \text{not exists}, & otherwise \end{cases}. \tag{4}$$

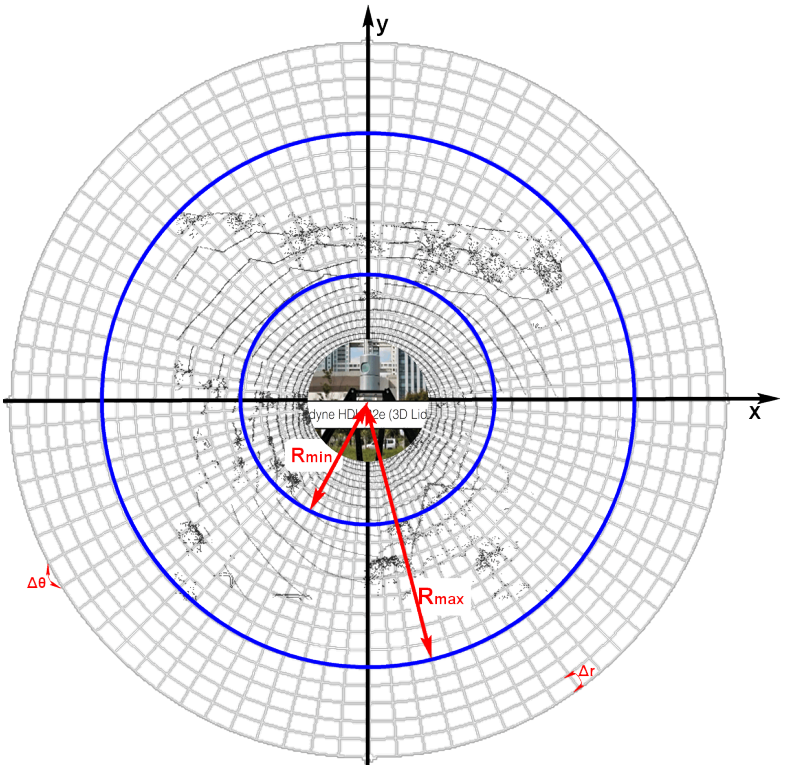


Fig. 2: Polar Model Projection

By these two models projection mentioned above, the LiDAR's 3D point clouds are dimensionally reducted to $2\frac{1}{2}$D, with $\frac{1}{2}$ representing for following statistical features.

B. Obstacle Detection Based on Hybrid Statistical Features

Our obstacle detection approach consists of following three steps: firstly remove ground points; next, non-ground points are used to capture obstacles; last, some overhanging obstacles, which our autonomous vehicle can pass through, should be excluded.

a) Remove Ground Points: Points that belong to the ground surface constitute the majority of the point clouds and their removal significantly reduces the number of points involved in the proceeding computations.

After projection using Cartesian Model or Polar Model, statistical features in each grid cell such as elevation mean,

elevation standard deviation, elevation variance, elevation Min-Max deviation are calculated. We use a simple but efficient way, the vertical mean, to extract the ground plane for removing ground points. Outliers should be eliminated as following way: where there are $N_{(x_{idx},y_{idx})}$ points in grid (x_{idx}, y_{idx}) with maximum elevation z_{max}, minimum elevation z_{min}, its vertical mean is,

$$z_{mean} = \frac{(\sum_{j=0}^{N_{(x_{idx},y_{idx})}-1} z_j) - z_{max} - z_{min}}{N_{(x_{idx},y_{idx})} - 2} \quad (5)$$

In commom road scene, road curbs are little obstacles compared with cones, street lights, fences, greenbelt, street trees and so on. The typical height for road curbs is from 5cm to 10cm [8] and we use a critical vertical mean threshold $\delta = 5$ (cm) to avoid false positives but remove great majority of ground points.

b) Capture Obstacles: The Min-Max vertical distance is a widely used feature in many LiDAR-based segmentation tasks, like [7] [1] [2] [9] [10]. Similar to them, the maximum absolute difference in z coordinate of all points falling into the respective grid cell is calculated and a vertical distance threshold $\sigma = 10$ (cm) works well in our road scenarios. In addition, as mentioned above, z_{max} or z_{min} which has a large bias from the z_{mean} should be eliminated.

c) Exclude Overhanging Obstacles: Street trees are seen everywhere inside the campus and they always grow over the road. Some other overhanging structures like an open-style door, overbridge and so on that our autonomous vehicle can pass through and these should not be captured as obstacles using Min-Max feature.

Fig. 3: The Autonomous Vehicle from Sun Yat-sen University, China

Our autonomous vehicle (Fig. 3) is $H_v = 172$ (cm) high, hence point clouds refered to higher objects can be ignored. We count the point clouds $\{p_i \in P | \ \delta \leq z_i \leq H_v\}$ and calculate the proportion of P for each grid cell,

$$\phi(x_{idx}, y_{idx}) = \frac{|P|}{N_{(x_{idx},y_{idx})}}, \quad (6)$$

and 78% works well in our road scenarios.

Processing strategy	Input	Output
Current-scan processing	1	1
	1 2	2
Multi-scan fusion	1 2 3	3
	1 2 3 4	4
	1 2 3 4 5	5
	...	...

Fig. 4: Multi-scan Fusion Using Sliding Window with Size 3

III. Multi-scan Fusion

A. Historical Pose Transformation

For fusing historical scans, what have to do first is to transform those historical scans to current LiDAR base. Since the LiDAR scanner is mounted on our autonomous vehicle, the physical constraints like momentum are bound to the vehicle odometry.

Before transformation, we analyze the LiDAR data offline. The data was captured when the vehicle was driven at 20kph or slower. Because the highest speed for our autonomous vehicle is 20 kph in auto mode. Using ICP (Iterative Closest Point) [11] algorithm to calculate the RT (Rotation matrix and Translation vector) from the current scan to the previous scan, we found that the rotation matrix are approximately equal to unit matrix and the translation vector are approximately equal to zero vector. "Approximately equal to" means that the differences between elements which are in the same position of matrix (or vector) are less than $1e$-5.

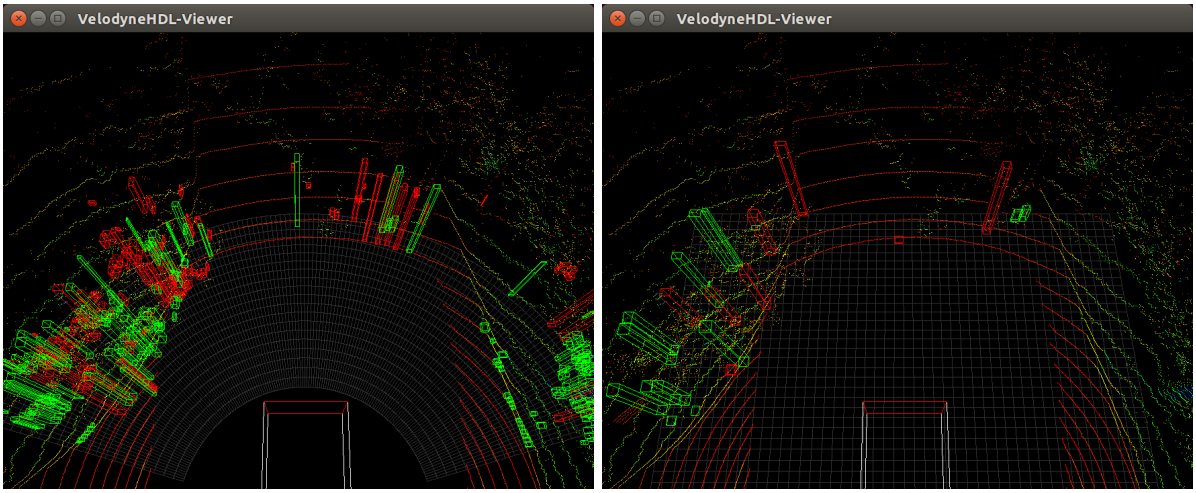
Since the difference is little, the points from the current scan and the previous scan which can be matched will be projected into the same grid cell, which means the historical pose transform can be ignored. For example, *Point* a is one point of the current scan and it is projected into grid cell A. *Point* a' is a point of previous scan and it is the point matches to *Point* a which means *Point* a' is very close to *Point* a. *Point* a' will be projected into grid cell A as we expected.

The validation of this hypothesis depends on the low speed of our autonomous vehicle (less than 20kph) and high frequency of LiDAR (10Hz). In other words, If the speed of vehicle is high, this hypothesis will be wrong.

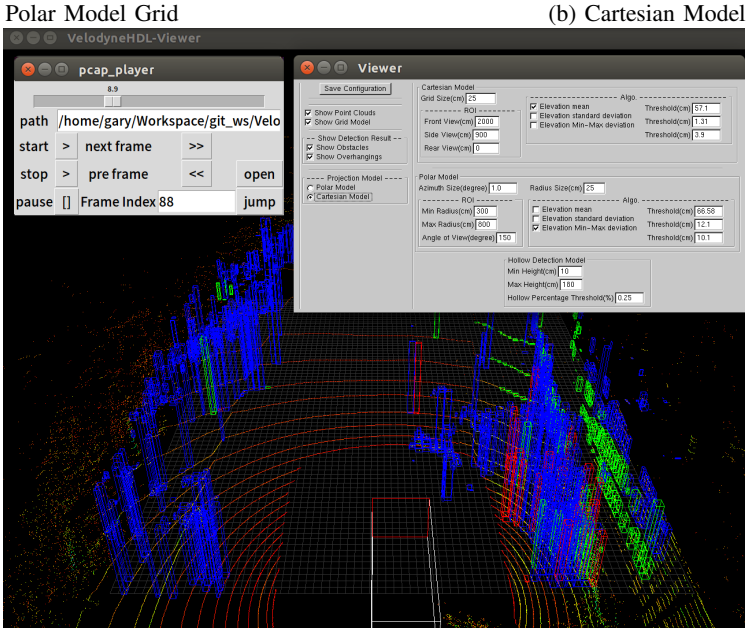
B. Current-Historical Scans Fusion

Although there is little difference between two scans as we have discussed above, we can not fuse too many scans together. If we do, the differences between each two scans will product together and then it will become too large to be ignored. So we only fuse three scans: the current scan and two previous scans together.

Since the frequency of LiDAR is 10Hz, LiDAR will output one scan every 0.1s. Numbering the scan in order, we get a sequence of scans. And then use a sliding window with size three to select the data we want to fuse together, as shown in Fig. 4.



(a) Polar Model Grid (b) Cartesian Model Grid



(c)

Fig. 5: ***(a)(b)*** is Grid Visualization of Corresponding Cell Size for Either Model. Grid of Detection Area is Used for Verification with Ground Truth. ***(c)*** is Our Visualization Tool with Algorithm Configuration Panel and Replay Tool. With *red* for obstacles higher than LiDAR, *green* for obstacles lower than LiDAR; *blues* are overhanging obstacles.

For each scan, we firstly apply current scan processing to it, and store the 0-1 occupancy grid map. And then merge three 0-1 occupancy grid maps to get one probability occupancy grid map as following:

$$grid^{t}_{(x_{idx},y_{idx})} = \sum_{i=1}^{3} \Omega_i \cdot grid^{t-i}_{(x_{idx},y_{idx})}, \tag{7}$$

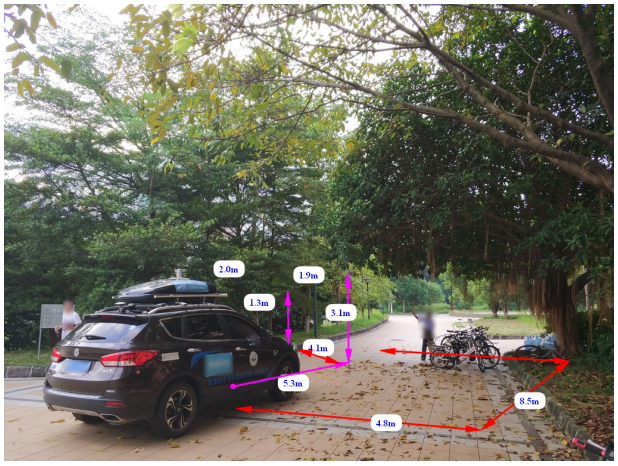
giving more accurate obstacles information for path planning.

IV. Experiments

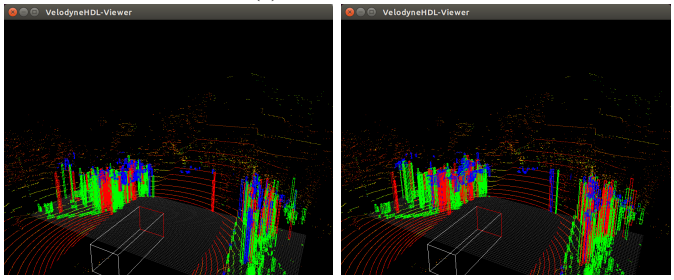
We analyze several typical scenarios offline to learn about the impact on different grid size in either projection model. Some available sizes are chosen in further online test and we analyze the detecting time inside both on our vehicle ECU (Electronic Control Unit) and raspberry pi 3. So far, over 2,000 miles of road tests for our autonomous vehicle also verify that approaches mentioned above can efficiently detect obstacles with high availability.

A. Offline Multi-Scenarios Analysis

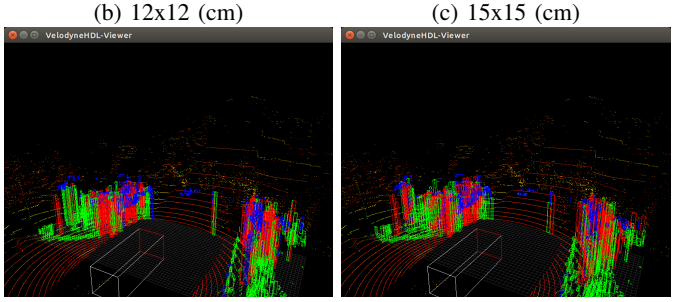
In this paper, we implement both models of projection and compare their detection results in following scenarios using different grid size. We record the LiDAR data in each scenario as PCAP (Packet CAPture) files and replay offline using our self-implemented tools (Fig. 5). Furthermore, landmarks in each scenario are manually marked and measured as ground truth.



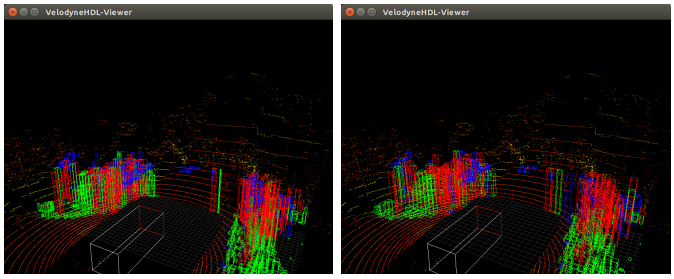
(a) Scenario: tree



(b) 12x12 (cm) (c) 15x15 (cm)

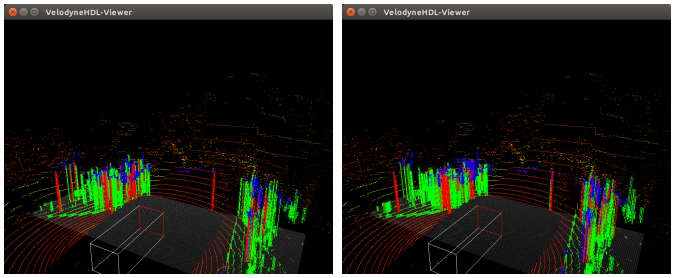


(d) 20x20 (cm) (e) 25x25 (cm)

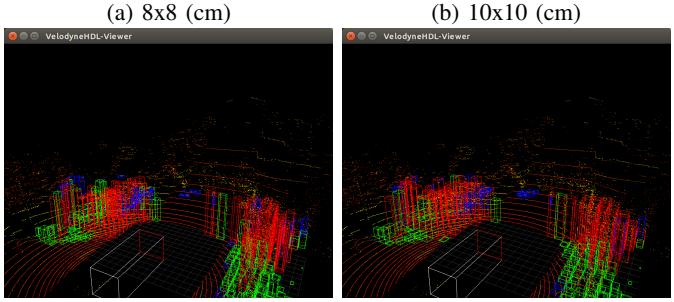


(f) 30x30 (cm) (g) 40x40 (cm)

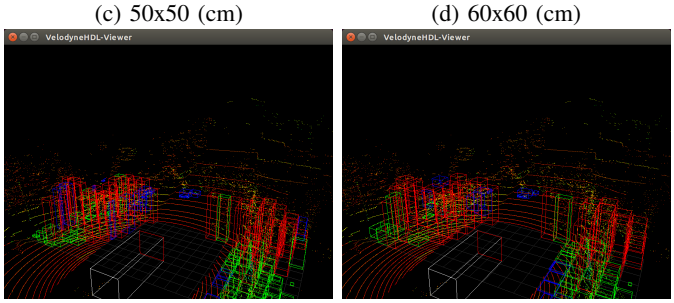
Fig. 6: Cartesian Projection Model Using Suitable Grid Size for Tree Overhanging Detection



(a) 8x8 (cm) (b) 10x10 (cm)



(c) 50x50 (cm) (d) 60x60 (cm)



(e) 75x75 (cm) (f) 100x100 (cm)

Fig. 7: Cartesian Projection Model Using too Small/Large Grid Size for Tree Overhanging Detection

a) Scenario I: ***tree***

As illustrated in Fig. 6, our approach using Cartesian Model with grid size from $12 \times 12(cm)$ to $40 \times 40(cm)$ can detect well for tree overhanging structures as well as road curbs. Moreover, grid size using $25 \times 25(cm)$ and $30 \times 30(cm)$ perform best for the overhanging tree detection in the front of our autonomous vehicle in this scenario.

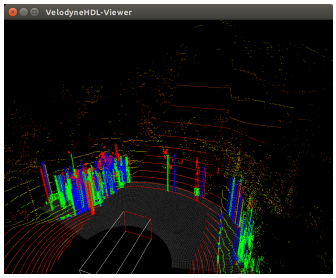
Compared with Fig. 7, smaller grid sizes not only upset the computations, but also lose the fineness of detection, missing some overhanging structures. As for larger grid sizes, the results, with obvious false positives, cannot be used for autonomous driving to pass through this road section.

Polar Model shares similar results as illustrated in Fig. 8 & Fig. 9, and azimuth with 1° resolution as well as $25(cm)$ and $30(cm)$ radius size are most suitable. Besides, Polar Model can capture more details of nearby road curbs than Cartesian Model, which is more available for nearby small obstacles.

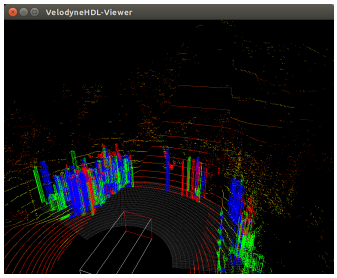
b) Scenario II-IV: ***flat road***, ***"toll gate"*** and ***intersection***

We also analyze our detection results on following scenarios: Fig. 10. Both Cartesian Model and Polar Model with different grid size are evaluated following the way mentioned above. Grid size using $25 \times 25(cm)$, $30 \times 30(cm)$ and $1.0^\circ \times 25(cm)$, $1.0^\circ \times 30(cm)$ are most tolerant in these scenarios.

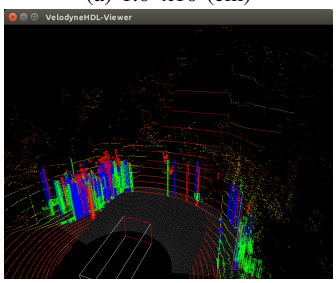
Detection results of grid size $25 \times 25(cm)$ and $1.0^\circ \times 25(cm)$ are compared in Fig. 11. Once again, for nearby road curbs, Polar Model performs better. However, Polar Model cannot detect small obstacles far away, even false check obstacles for



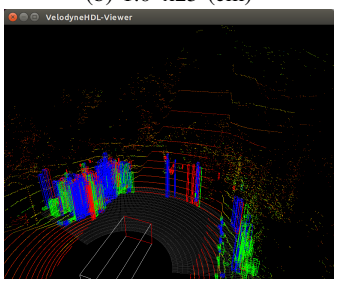
(a) 1.0°x10 (cm)



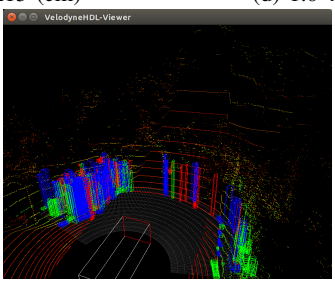
(b) 1.0°x25 (cm)



(c) 1.0°x15 (cm)

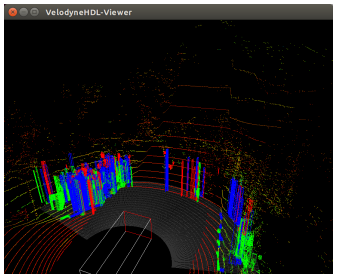


(d) 1.0°x30 (cm)

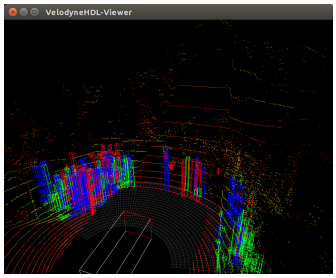


(e) 1.0°x40 (cm)

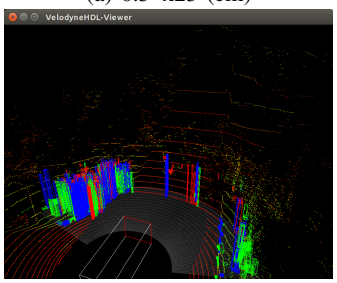
Fig. 8: Polar Projection Model Using Different Grid Size for Tree Overhanging Detection



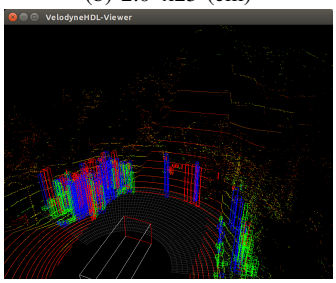
(a) 0.5°x25 (cm)



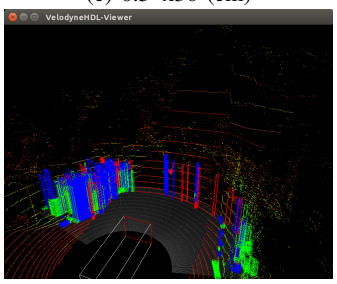
(b) 2.0°x25 (cm)



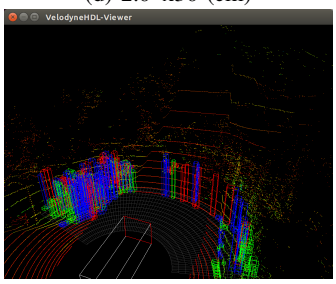
(c) 0.5°x30 (cm)



(d) 2.0°x30 (cm)



(e) 0.5°x40 (cm)



(f) 2.0°x40 (cm)

Fig. 9: Polar Projection Model Using too Small/Large Grid Size for Tree Overhanging Detection

overhanging structures in toll gate scenario. In comparision, Cartesian Model always performs normally, though without too much detail.

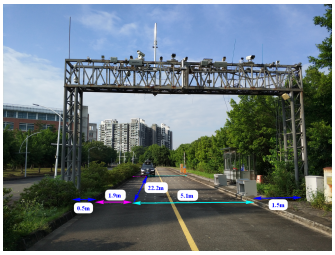
c) Scenario V: ***tilted ground***

Tilted ground detection is a huge challenge for obstacle detection in autonomous driving, and many papers (including those mentioned above) chose to keep from talking about it. As shown in Fig. 12 , Our approach can filter tilted ground during detecting, providing drivable messages for autonomous driving.

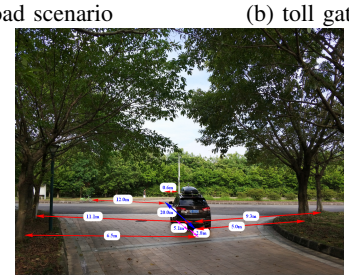
d) A brief sum-up: Based on the Cartesian Model projection, the whole space is divided into the same size of the grid cell, imparitial for obstacles both nearby and far away.

Differently, in Polar Model projection, starting from the center to the outward, inner grids are small and dense, while the outsides become larger and larger. The advantage of this Polar Model is that it has a good precision for nearby objects, which can well distinguish small obstacles when reducing the amount of computations. Focus on some, we say. The disadvantage is that it may not be possible to identify small obstacles far away.

(a) flat road scenario

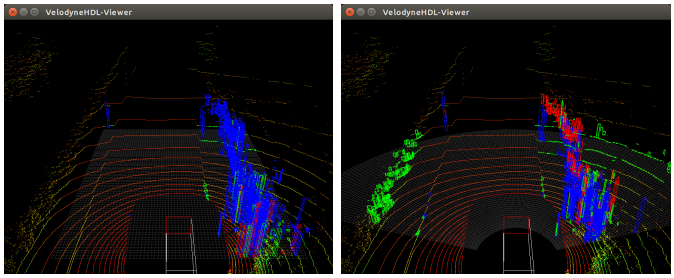


(b) toll gate scenario

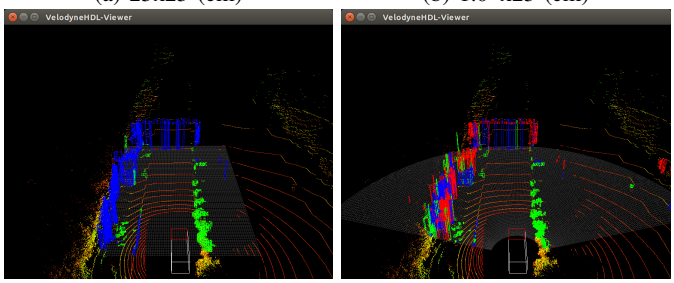


(c) intersection scenario

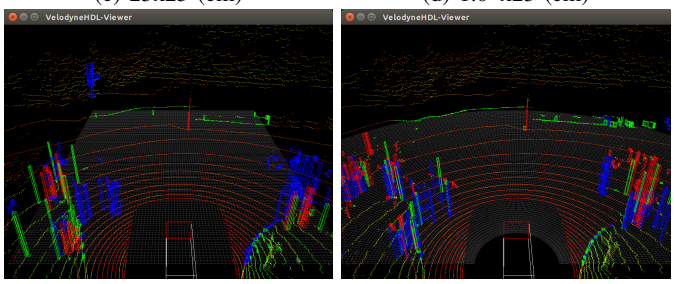
Fig. 10: Scenarios II-IV



(a) 25x25 (cm) (b) 1.0°x25 (cm)

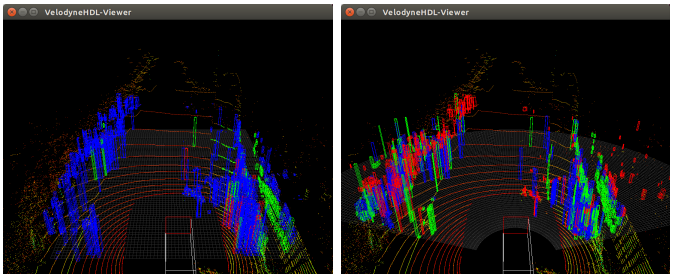


(c) 25x25 (cm) (d) 1.0°x25 (cm)

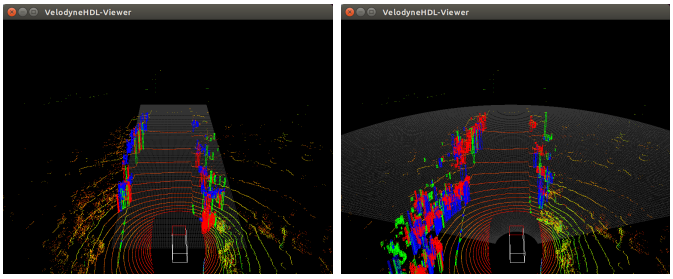


(e) 25x25 (cm) (f) 1.0°x25 (cm)

Fig. 11: Obstace Detection in Scenarios II-IV. ***(a)(b)*** is in flat road scenario, ***(c)(d)*** in toll gate scenario and ***(e)(f)*** in intersection scenario.



(a) 25x25 (cm) (b) 1.0°x25 (cm)



(c) 25x25 (cm) (d) 1.0°x25 (cm)

Fig. 12: Obstace Detection in Tilted Ground Scenario

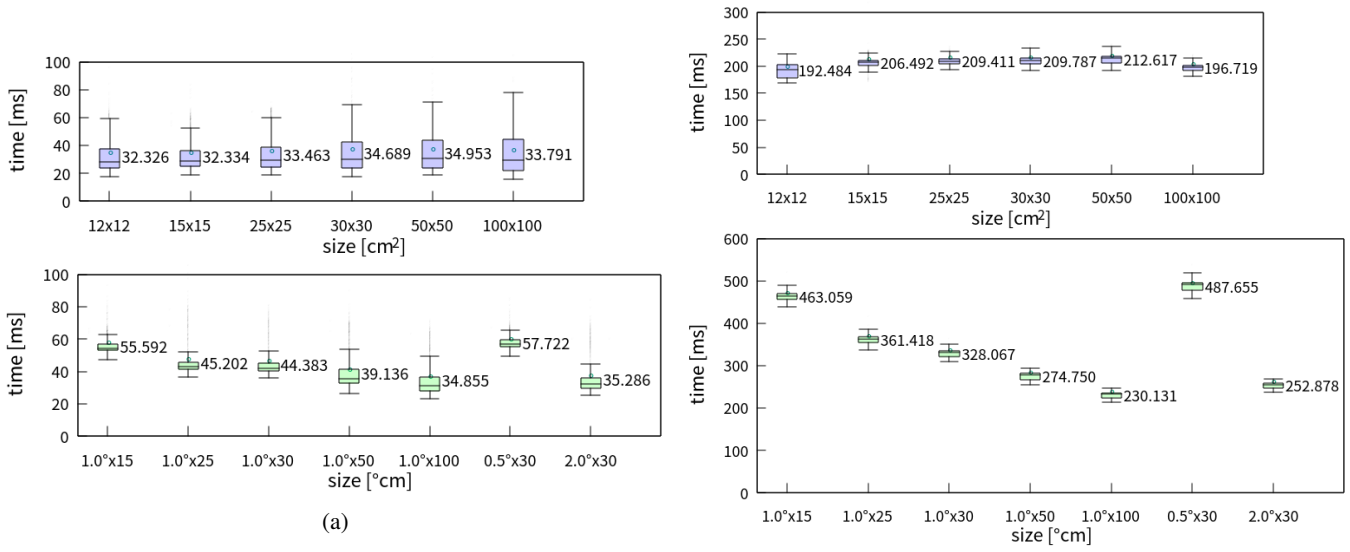
B. Online Running Results

Our autonomous vehicle is qualified with way points following function, based on our obstacle detection approaches mentioned in this paper. Using available grid sizes learning from Multi-Scenarios Analysis offline in either projection model, the following autonomous driving experiments in the track (Fig. 13) are carried out on 2 embedded platforms: (1) Our vehicle ECU configures with an 2.5Ghz Intel core-i7 6500u processor and Ubuntu 16.04; (2) The raspberry pi 3 also runs Ubuntu 16.04, but has a A53 (BCM2837) processor, the basic frequency of which is only 1.2Ghz. Autonomous performances are evaluated referring to the degree of human intervention, comfort of passengers, etc. Besides, detecting time inside are analyzed on both embedded platform, especially in term of restrict-computation raspberry pi 3.

a) time analysis: As illustrated in Fig. 14, raspberry pi 3 needs greatly more time, typically 8 times, than high-performing vehicle ECU. Besides, detecting time is almost the same in Cartesian Model, even using different grid sizes on both platform, which differs a lot in Polar Model. It costs more time using dense cells, that better matches the way we thought. The reason, we find that this Cartesian Model is more in line with this road section, filtering out the vast majority of obstacles except for curbs on both sides of the road. In addition, Polar Model is more stable due to its centralized detecting time.

b) autonomous driving performances analysis: Autonomous driving performances are evaluated referring to the degree of human intervention (45%), frequency of antics

Fig. 13: Test Section Environments in East Campus, Sun Yat-sen University, China. Our autonomous vehicle starts at 1, drives around 2 U-turns at 2 and 3, changes to left driveaway to pass the cones 4 then changes back to right driveaway, finally through the "toll gate".



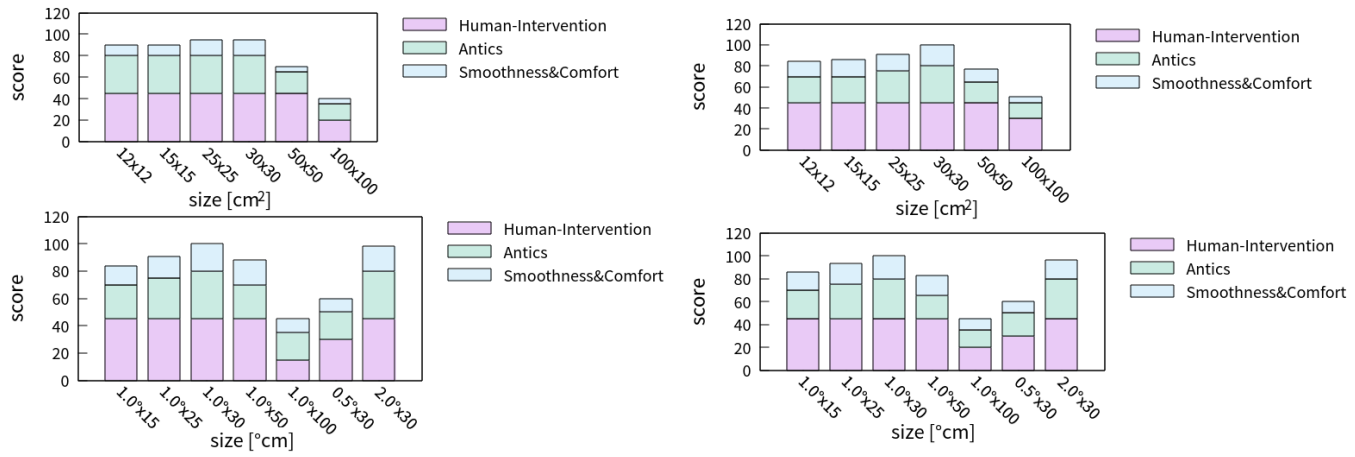
(a)

(b)

Fig. 14: Online Obstacle Detection Time both on Our Vehicle ECU and Raspberry pi 3. ***(a)*** is on our vehicle ECU and ***(b)*** on raspberry pi 3; *blues* are for Cartesian Model and *greens* for Polar Model.

(35%) and control smoothness as well as comfort of passengers (totally accounting for 20%). According to Fig. 15, when the obstacle detection runs on raspberry pi 3, the autonomous driving performances are acceptable and very close to the situation when running on our vehicle ECU.

What's more, when using grid size 25×25, 30×30



(a) (b)

Fig. 15: Online Obstacle Detection Performance both on Our Vehicle ECU and Raspberry pi 3. ***(a)*** is on our vehicle ECU and ***(b)*** on raspberry pi 3.

and cell size $1^\circ \times 30$, the system performs best, no matter running on ECU or raspberry pi 3. We can also see that the Cartesian Model causes less antics than the Polar Model but the control smoothness is worse. That is because the Polar Model holds more details of the point clouds than Cartesian Model especially when the obstacles are nearby.

In WRC (World Robot Conference) 2017 (Fig. 3), Beijing, we use the ECU mentioned above to finish the detection and planning computations for our autonomous vehicle, smoothly tracking to complete the entire journey.

V. Conclusion

In this paper, we put forward a computationally efficient solution for LiDAR-based obstacle detection. Two projection models, Cartesian Model and Polar Model are proposed, providing the basis for high-performance computations of grid-based statistical features. Each scan is processed and moreover, 2 more nearest historical detection results are fused, giving more accurate obstacles information for path planning. A considerable amount of experimental work is carried out on our autonomous vehicle using our approach. The autonomous driving performances are still desirable even on a restrict-computation raspberry pi 3.

Acknowledgment

This work was supported by the National Key Research and Development Program of China under grant No.2017YFB1001703 and the Fundamental Research Funds for the Central Universities under grant No.17lgjc40.

References

[1] B. Douillard, J. Underwood, N. Melkumyan, S. Singh, S. Vasudevan, C. Brunner, and A. Quadros, "Hybrid elevation maps: 3d surface models for segmentation," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1532–1538.

[2] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 560–565.

[3] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5067–5073.

[4] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[5] B. Li, "3d fully convolutional network for vehicle detection in point cloud," *arXiv preprint arXiv:1611.08069*, 2016.

[6] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," *arXiv preprint arXiv:1611.07759*, 2016.

[7] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[8] C. Fernández, R. Izquierdo, D. F. Llorca, and M. Sotelo, "Road curb and lanes detection for autonomous driving on urban scenarios," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 1964–1969.

[9] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3d lidar point clouds," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2798–2805.

[10] M. Himmelsbach, A. Mueller, T. Lüttel, and H.-J. Wünsche, "Lidar-based 3d object perception," in *Proceedings of 1st international workshop on cognition for technical systems*, vol. 1, 2008.

[11] P. J. Besl, N. D. McKay *et al.*, "A method for registration of 3-d shapes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 239–256, 1992.