A Robust Real-time 2D Virtual Scan Generation Method for Obstacle Detection in Complex Urban Environment*

He Mengwen[1,3], Eijiro Takeuchi[2,3], Yoshiki Ninomiya[2,3] and Shinpei Kato[1,3]

***Abstract*— Obstacle detection is an essential technique for many mobile robotics applications. Unlike 2D LiDAR, 3D LiDAR obtains point cloud at one time, which enables robust obstacle detection in complex urban environment. However it is also a challenge to process such huge 3D data in real time. The *virtual scan* (VScan), first introduced in [1] for efficient vehicle detection and tracking, is a 2D compression of 3D point cloud to represent free space, obstacles and unknown areas. Therefore, VScan is suitable for obstacle detection, and additionally it bridges the new-born 3D LiDAR and many matured applications based on 2D LiDAR, for example, occupancy grid map, SLAM, planning, detection and tracking.**

However, in complex urban environment, steep ramp with large slope, low curb along road and overhung barrier gate at entrance frequently exist. Therefore, in our practical development of autonomous vehicle, we put some strict requirements on the robustness and speed of VScan generation method, and based on these requirements, we developed a robust real-time 2D VScan generation method.

In our method, a new data structure called *basic VScan matrix* (BVSM) is firstly proposed to represent the 3D point cloud around vehicle. Then a *simultaneous road filtering and obstacle detection method* works on the BVSM is developed for robust VScan generation, and a *sorted array based acceleration method* is developed for real-time VScan generation.

We conducted several experiments around Nagoya University, where steep ramp, barrier gate and curb exist, to demonstrate the robustness and speed of our VScan generation method. The time cost of generating VScan with 2000 beams is limited within 15ms. For obstacle detection on road with large slope, it is stable within 60m and few number of false positive.

I. INTRODUCTION

Obstacle detection is an essential perception capability for mobile robotics applications in dynamic environments. Autonomous and semi-autonomous vehicles rely on real-time obstacle detection for a growing list of applications including: platooning, adaptive cruise control, and crash-avoidance systems. Meanwhile, the detection of obstacle should robustly work in complex urban environment includes steep ramp with large slope, low curb along road and overhung barrier gate at entrance (Fig. 1), to guarantee safety.

Most fully autonomous vehicles, includes all entrants in the DARPA Urban Challenge [2], rely on LiDARs (Light Detection and Ranging) for obstacle detection, localization and mapping. Among these LiDARs, Velodyne, a type of 3D LiDAR, can produce a 3D scan of the ego-vehicle's environment at a rate of 10Hz producing nearly 100,000 points per frame [3]. Despite the challenge of real-time processing, 3D LiDAR with a wealth of range data is one of the most robust currently available sensor solutions for obstacle detection in complex urban environment.

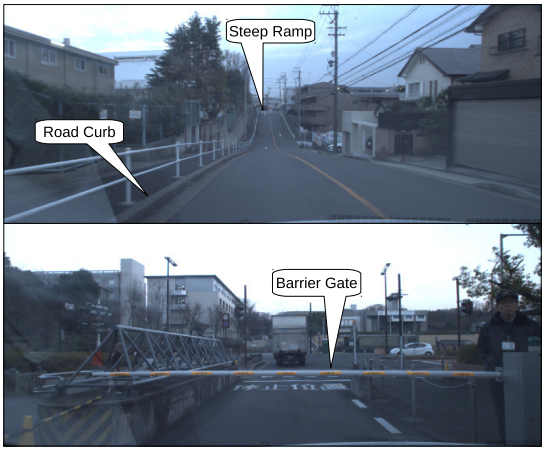

Fig. 1. Ramp with large slope (upper) and barrier gate at entrance (lower). These photos are taken by our autonomous vehicle in an experiment around Nagoya University.

Data preprocessing or clean-up techniques are useful to augment the physical sensor and method efficiency, therefore, for range data it is common to sub-sample the rays, project from 3D to 2D, and reject outliers [4]. An example of preprocessing Velodyne 3D data is to compact it to 2D *virtual scan* (VScan) for vehicle detection and tracking [1]. The VScan can be thought of as a set of beams emitted from a center to conserve information with respect to free space, obstacles and unknown areas, which is same to 2D LiDAR range data except that the VScan is often from different heights. Therefore, VScan is suitable for obstacle detection in urban environment.

ROS provides a real-time *basic VScan generation* method called "*pointcloud_to_laserscan*" [5] for easy urban environment with flat road. However, it is not robust to complex urban environment, especially the sloped road surface will be detected as fake obstacle. Thus, it is common to filter out ground readings firstly for robust VScan generation and obstacle detection.

Many ground filter methods have been developed in object (obstacle) detection methods [6], [7], [8], [9], [10], [1], [11],

*This research is supported by the Center of Innovation Program (Nagoya-COI: Mobility Society leading to an Active and Joyful Life for Elderly) from Japan Science and Technology Agency.

[1]Graduate School of information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan alexanderhmw@ertl.jp, shinpei@is.nagoya-u.ac.jp

[2]Institute of Innovation for Future Society (MIRAI), Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan {takeuchi,ninomiya}@coi.nagoya-u.ac.jp

[3]JST/COI, Nagoya 464-8603, Japan

[12] recently for autonomous vehicle applications. These methods can work on sloped road to detect object (obstacle) using various techniques like normal vector, range image, occupancy grid map, the measurement of consecutive readings, etc. However, in our practical development of autonomous vehicle, we put some strict requirements on the robustness and speed of VScan generation method as below:

1) Robustness: First, it can cope with sloped road and the frequent change of vehicle's roll and pitch in complex urban environment; Second, it can detect low obstacle like road curb or overhung obstacle like barrier gate; Third, the detection of obstacle is stable within a certain distance range.

2) Speed: It is desirable to set the number of beams in VScan as many as possible to guarantee the robustness of following process. Meanwhile, it is preferable to limit the time cost of VScan generation to guarantee the real-time performance of entire process. In our research, for Velodyne's rotational resolution, we set the number of beams in VScan to 2000 and the time cost of VScan generation is limited within 15ms by using CPU with one thread.

In this paper, we present a robust real-time VScan generation method in Section III: A new data structure called *basic VScan matrix* is firstly proposed to represent the 3D point cloud around vehicle. Then a *simultaneous road filtering and obstacle detection* method works on the *basic VScan matrix* is developed for robust VScan generation, and a *sorted array based acceleration method* is developed for real-time VScan generation.

Section VI presents several experiments conducted with our autonomous vehicle (Fig. 7) around Nagoya University, where ramp way, barrier gate and curb exist, to demonstrate the speed and robustness of our vscan generation method. Section VI summarizes our work.

II. Related Work

Obstacle/Object detection is a core task in mobile robotics as well as in the field of intelligent vehicles. The most popular sensors used in obstacle detection are camera and LiDAR.

The vision-based obstacle/object detection for driver assistance has received considerable attention over the last 15 years [13] and a variety of vision-based object detection methods have been developed [14], [15], [16], [17], [18] using monocular or stereo camera. Especially, for [18], they provide a heuristic segmentation method to separate moving objects from the static background like road surface, etc. Then the vision-based depth and motion information is transferred into so called *Stixels* (stick pixels), a very compact representation of 3D scenes that can also be applied to LiDAR or radar data. Inspired by this *Stixel* method, our VScan generation method not only represents detected obstacles in 2D coordinate, but also reserve their height information as *Stixel* with minimum and maximum height. Therfore, our VScan generation method is actually a 3D obstacle detection method.

Meanwhile, a number of 3D LiDAR (Velodyne) based obstacle detection methods have been developed over the past decade [6], [7], [8], [9], [10], [1], [11], [12]. Among these methods, road filtering is a very important preprocessing step and there are mainly 4 kinds of road filtering method used in obstacle detection: 1) [6], [7] project all points to depth map and then use normal vector to identify road points; 2) [8], [9], [10] project all points to an assumed or estimated ground plane, often combined with an occupancy grid map, and then use occupancy value to filter out ground, for example, [10] used the density of the points within a cell as occupancy value; 3) [1], [11] project all points to 3D grid in spherical coordinates and 4) [12] projects all points to 2D grid in polar coordinates. Then 3) and 4) identify road point by evaluating its road slope.

Some of these methods can work on sloped road in real-time. However, they cannot fully satisfy the requirements on the robustness and speed of VScan generation and obstacle detection mentioned in Section I. For example, the normal vector based method requires normal calculation for every point, which is often time-consuming, and [6] reported their time cost of normal calculation is 352ms. According to [6]'s practice, the occupancy grid map based methods are not suitable for reliable detection of sloped objects like vegetation, hills, or curbs. [1], [11] filter out road by the measurement of consecutive readings for each bearing, and thus they cannot always detect overhung obstacle. In [12]'s polar coordinates, the resolution along radius dimension is set to 3m, which is for producing candidate ground points for road slope evaluation. However, with such a low resolution along radius dimension, this method would not get accurate beam length and also not be sensitive to low obstacle like road curb.

Our VScan generation method is originated from [1] and we keep their two features: 1) road filtering is independently conducted on each beam in VScan; 2) obstacle is detected by calculating road slope and comparing it with a preset slope threshold. Meanwhile, in order to detect overhung obstacle, we developed a new data structure called *basic VScan matrix*. Combined with the two features of [1] mentioned above, a *simultaneous road filtering and obstacle detection method* works on the *basic VScan matrix* is developed. Moreover, in order to maintain the real-time performance of generating large number of beams in VScan, a *sorted array based acceleration method* is developed.

In conclusion, the contributions of our VScan generation method for obstacle detection are: 1) it is a robust method that can work on steep ramp with large slope, and detect low and overhung obstacle like curb and barrier gate; 2) it is a real-time method that can generate 2000 beams in VScan within 15ms by using CPU with one thread; 3) it is a 3D obstacle detection method by using the *Stixel* to conserve the minimum and maximum height of detected obstacle.

III. Obstacle Detection Method

A. Method Overview

As mentioned in Section II, our obstacle detection method independently works on each beam in VScan like [1],

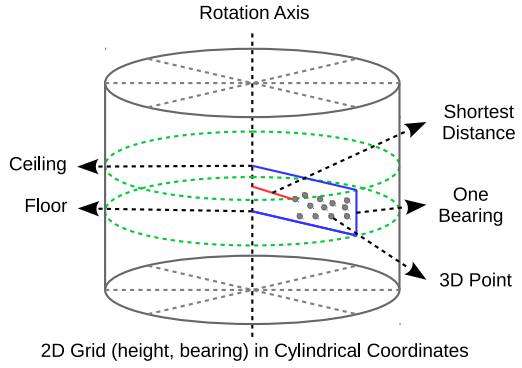

Fig. 2. 2D grid (bearing, height) in cylindrical coordinates and the illustration of basic VScan generation method.

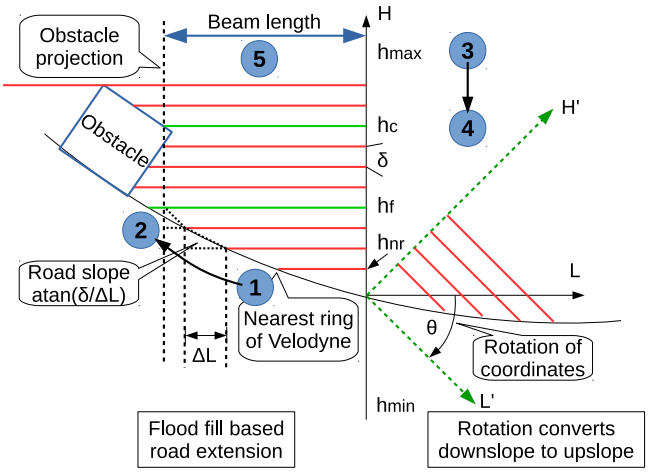

Fig. 3. Illustration of VScan generation method. The left part shows simultaneous road surface extension and obstacle detection. The right part shows the rotation method to cope with downslope road.

therefore, in the following, we will only take one bearing to explain our VScan generation method.

The measurement of consecutive readings in [1] cannot detect overhung obstacle is due to that they only focused on local feature derived from 3 adjacent readings. In order to detect overhung obstacle, more global feature should be considered.

In our method, we project all points into a 2D grid (bearing, height) in cylindrical coordinates as shown in Fig. 2. Then a certain pair of floor and ceiling (discrete height values along rotation axis and noted as h_f and h_c) can define a height interval for *basic VScan generation* [5] and the beam length in basic VScan equals to the shortest distance between rotation axis and a nearest point.

The basic VScan from small height interval can be regarded as a relatively local feature and the one from large height interval can be regarded as a relatively global feature. In order to calculate and store all possible basic VScan with different height range $[h_f, h_c)$, we develop a new data structure called *basic VScan matrix*. The detail of *basic VScan matrix* will be discussed in Section III.B.

Generally speaking, the local feature is suitable for road filtering, while the combination of local and global feature is robust for obstacle detection. Therefore, we take all possible pairs of floor and ceiling into account for VScan generation and thus develop a *simultaneous road filtering and obstacle detection method* to find the best height range $[h_f, h_c)$ and the corresponding basic VScan as final result. We present this method's basic steps as below and its details will be discussed in Section III.C.

Simultaneous road filtering and obstacle detection method starts from $[h_{min}, h_{max})$ to find the best height range $[h_f, h_c)$ as shown in Fig.3. We first need to find the height of nearest ring of Velodyne h_{nr} ①. Then we use flood fill method to extend road surface toward obstacle with a preset slope threshold to get floor h_f ②. Then we start from h_{max} ③ and decrease ceiling to h_c ④ to detect obstacle. Finally, we get the beam length from the height range $[h_f, h_c)$ ⑤.

Moreover, our method is based on flood fill method along height dimension. Therefore, this method could only handle flat and upslope road, but for downslope road, the obstacle is shaded by road surface. We use rotational transformation to make our method to work with downslope road. As shown in Fig.3 right part, after rotation of length-height coordinates, downslope road is converted to upslope road in new $L' - H'$ coordinates and its geometry property is still conserved. Therefore we can generate VScan in the rotated coordinates firstly. Then we rotate the generated VScan back to get obstacle detection result.

B. Basic VScan Matrix Construction

Define $L_i(h_f, h_c)$ as the beam length of basic VScan from height interval $[h_f, h_c)$ in ith bearing, and create grid along height dimension with step δ in height interval $[h_{min}, h_{max})$. Therefore, $L_j(h_f, h_c)$ can be discretized as $L'_i(g_f, g_c)$, where $g = \lfloor (h - h_{min})/\delta \rfloor$, and we define all possible basic VScans from different height ranges as set (1). This set can be stored by a $g_{max} \times g_{max}$ upper triangular matrix as shown in Fig.4. In this paper, we call it *basic VScan matrix* (BVSM). The property A (see appendix) builds an order over set (1) as well as BVSM. This order is the guarantee of our obstacle detection method's correctness.

$$\{L'_i(g_f, g_c) | g_f \in \mathbb{N}, g_c \in \mathbb{N}, g_{min} \leq g_f < g_c \leq g_{max}\} \quad (1)$$

It is time-consuming to directly calculate all elements in BVSM. However, the property B (see appendix) enables the dynamic programming on BVSM. As shown in Fig. 4, the diagonal element in blue color is the basic VScan from corresponding smallest height interval and it can be directly calculated by projecting corresponding points into its 2D grid cell. Then for other elements, they are assigned the smaller value of their left and lower elements illustrated as green arrows.

C. Simultaneous Road Filtering and Obstacle Detection

From the property A, we know that the red corner cell in BVSM (Fig. 4) corresponds to the shortest beam length

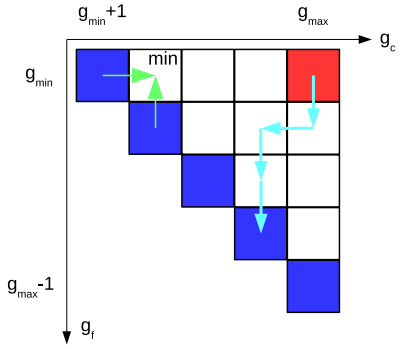

Fig. 4. basic VScan matrix to store all possible basic VScan. Green arrows illustrate the way to calculate non-diagonal elements. Cyan path illustrates the way to extend road surface and detect obstacle.

of all possible basic VScans. For Velodyne's property, we know that this shortest beam corresponds to either the nearest ring of Velodyne on road or obstacle located inside the nearest ring. Therefore, this cell in BVSM is the start point to simultaneously filter out road and detect obstacle in our method.

If the start point is on a obstacle, then the method will end and output the shortest beam as VScan as well as detected obstacle. If the start point is the nearest ring on road, we will present how our method uses BVSM to realize the basic steps mentioned in Section III.A (Fig. 3) as below.

Before the process of road surface filtering and obstacle detection, we first define an angular threshold ϕ as maximum road slope and a height threshold ΔH to determine the overhang obstacle's maximum height to the road (also regarded as passable height for vehicle).

Assume the method is at a random BVSM's cell (g_f, g_c) and it has a beam length $L'_i(g_f, g_c)$, which represents the beam length of the basic VScan generated from height range $[h_f, h_c)$. Then the method needs to distinguish road and obstacle at height h_f. The decision tree is shown as below and there are 3 possible results: road, unknown and obstacle.

$$\Delta L \cdot tan(\phi) \begin{cases} \geq \delta & \Rightarrow Road \\ < \delta & \begin{cases} \Delta h > \Delta H & \Rightarrow Unknown \\ \Delta h \leq \Delta H & \Rightarrow Obstacle \end{cases} \end{cases} \tag{2}$$

$$\Delta L = L'_i(g_f + 1, g_c) - L'_i(g_f, g_c) \text{ and } \Delta h = h_c - h_f$$

The first level of the decision tree utilizes flood fill method with a slope threshold to detect road. According to the definition of $L'_i(g_f, g_c)$ and the Property A, the $\Delta L \leq L'_i(g_f + 1, g_c) - L'_i(g_f, g_f + 1)$. Therefore, the slope value derived from $atan(\delta/\Delta L)$ is the maximum slope value at height h_f, and thus the decision result will be road if $atan(\delta/\Delta L) \leq \phi$.

The second level of the decision tree detect whether there is a obstacle in condition of $atan(\delta/\Delta L) > \phi$, which indicates a large slope value exist in height interval $[h_f, h_c)$. We use the height difference $\Delta h = h_c - h_f$ to filter out fake obstacle highly elevated above ground, for example, tree tops or overpasses. If $\Delta h > \Delta H$, there may exist a fake obstacle. Otherwise, we know there is an obstacle in height interval $[h_f, h_c)$ and it could be an overhung obstacle for the calculation of slope value is from relatively global feature.

Based on the decision result, our method will move on BVSM or stop for final result:

1) Road: it will move to cell $(g_f + 1, g_c)$ (down), which is to extend road surface (① → ② in Fig. 3).
2) Unknown: it will move to cell $(g_f, g_c - 1)$ (left), which is to further check whether this is a fake obstacle by decreasing ceiling (③ → ④ in Fig. 3).
3) Obstacle: it will stop and the beam length of VScan will equal to $L'_i(g_f, g_c)$ (⑤ in Fig. 3).

We could find that this method is actually to find a path from BVSM's red corner cell toward the blue diagonal cell by moving down or left step by step illustrated by cyan path in Fig.4, which may be stopped by detected obstacle.

D. Sorted Array Based Acceleration Method

We have present our VScan generation method for obstacle detection. It consists of 2 consecutive but independent steps: 1) basic VScan matrix construction, 2) simultaneous road filtering and obstacle detection on BVSM. Then, we will analyze its computational complexity for real-time application.

Assume the number of Velodyne points is P, the number of beams in VScan is N, and the grid number along height dimension is M. The computational complexity of the first step is $O(P + NM^2)$, where $O(P)$ is to calculate diagonal elements in BVSM, and $O(NM^2)$ is to calculate other elements in BVSM using dynamic programming. The second stepls computational complexity is $O(NM)$, which is to move on BVSM. Therefore, its final computational complexity is $O(P + NM^2)$ and memory cost is $O(NM^2)$ (BVSM).

In order to detect low obstacle, the grid step δ should be small and thus the grid number M would be large, which makes this method time-consuming. Therefore we use the property C (see appendix) to develop a *sorted array based acceleration method*. It is essentially same with BVSM based method, but its computational complexity is $O(P + NMlog(M))$ and memory cost is $O(NM)$.

From the property C, we need to sort all BVSM's diagonal elements with their length L'_i in ascending order and for the elements have same length, sort them with their floor height g_f in descending order. After sorting, we get two associated arrays: beam length array $L'_i[j]$ and height grid index array $g_f[j]$. From the property A, we know that $L'_i[0]$ corresponds to the nearest ring and $g_f[0] = g_{nr}$ as shown in Fig.3.

Meanwhile, the property C provides a direct way to calculate BVSM's non-diagonal elements $L'_i(g_f, g_c)$, where $g_f \geq g_{nr}$, from two elements of the sorted arrays. Therefore, we could only keep BVSM's diagonal elements and thus we reduce the memory cost from matrix form $O(NM^2)$ to array

form $O(NM)$. For the computational complexity, instead of constructing BVSM, we need to sort the diagonal elements and its computational complexity is $O(P + NMlog(M))$. This is the acceleration of the first step of BVSM based method.

For the second step of BVSM based method. Because, every non-diagonal element checked by BVSM based method can be directly calculated by two corresponding diagonal elements, which are stored in the sorted array. Instead of moving down or left from BVSM's red corner cell to diagonal element, this method starts from the first two elements in sorted array and then move through the sorted array to simultaneously filter out road and detect obstacle. Thus its computational complexity is still $O(NM)$.

In this method, two array indices j_0 and j_1 ($j_0 < j_1$) are needed and their initial values are $j_0 = 0$ and $j_1 = 1$. Assume this method is with indices j_0 and j_1, then it needs to distinguish road and obstacle at height $h_f[j_0]$. The decision tree is shown as below and there are 4 possible results: road, unknown, obstacle and obstacle*.

$$\Delta g \begin{cases} < 0 & \Rightarrow Unknown \\ = 1 & \Rightarrow (2) \\ > 1 & \begin{cases} \Delta h > \Delta H & \Rightarrow Unknown \\ \Delta h \leq \Delta H & \Rightarrow (4) \end{cases} \end{cases} \tag{3}$$

$$\Delta L \cdot tan(\phi) \begin{cases} < \delta & Obstacle \\ \geq \delta & Obstacle* \end{cases} \tag{4}$$

$\Delta L = L'_i[j_1] - L'_i[j_0]$, $\Delta h = h_f[j_1] - h_f[j_0]$ and $\Delta g = g_f[j_1] - g_f[j_0]$

The decision tree present above is completely derived from (2) with the Property C. For the limitation of space, we only briefly discuss about (4), whose results are obstacle and obstacle*, in condition of $\Delta g > 1$ and $\Delta h \leq \Delta H$. The length array $L'_i[j]$ is in ascending order and the height grid index array $g_f[j]$ is in descending order. Therefore, $\exists$ a $L'_i[j]$, where $j_0 < j < j_1$, satisfy $L'_i[j_0] \leq L'_i[j_1] \leq L'_i[j]$, which must lead to obstacle detection.

Based on the decision result, our method will move on the sorted array or stop for final result:

1) Road: $j_0 \leftarrow j_1$ and $j_1 \leftarrow j_1 + 1$.
2) Unknown: $j_1 \leftarrow j_1 + 1$.
3) Obstacle: the beam length of VScan will equal to $L'_i[j_0]$.
4) Obstacle*: the beam length of VScan will equal to $L'_i[j_1]$.

In conclusion, *sorted array based acceleration method*'s computational complexity is $O(P + NMlog(M))$ and the memory cost is $O(NM)$ (the sorted arrays). As shown in Fig.5, we compared the time cost to generate VScan with 2000 beams between BVSM based method and sorted array based method. We find that the former one is very slow for small height step, while the time cost of latter one is kept around 10ms. For [1], it reports that it can generate 720 beams (0.5 degree for angular resolution) at 40Hz. Therefore, our method is about 6 times faster with additional overhung obstacle detection.

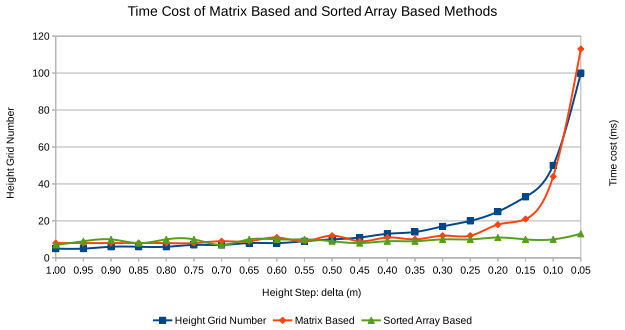

Fig. 5. Time cost to generate 2000 beams VScan of matrix based method and sorted array based method.

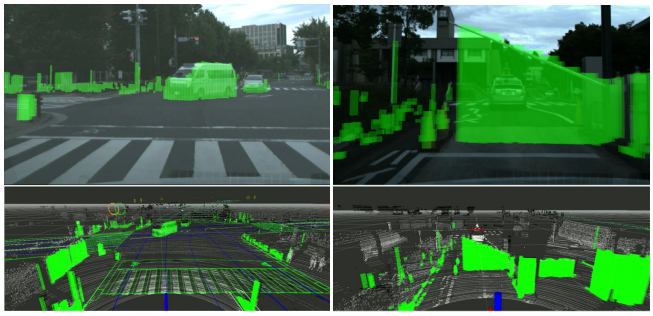

Fig. 6. Example of Stixel with minimum and maximum height information from our virtual scan generation method. Left shows a vehicle detection and right shows a barrier gate detection.

E. 3D Obstacle Detection with Stixel

Our 3D obstacle detection method uses *Stixel* to conserve the minimum and maximum height of detected obstacle (Fig. 6). For minimum height, it is determined when the obstacle is detected. For maximum height, the method keeps on moving on BVSM (or sorted array) after obstacle detection and it is determined when the "road" is detected again as (2). Because the method simultaneously filter out road surface and get virtual scan, the minimum height of overhung object is actually the height of its projection on road as shown in Fig.6.

IV. Experiment

In our research, we have an autonomous vehicle equipped with Velodyne HDL-64E S2 and Grasshopper 3 camera (Fig. 7). In order to demonstrate our obstacle detection method, We conducted an experiment around Nagoya University, where flat road, steep ramp, curb and barrier gate exist. The total distance traveled in this experiment is about $10km$.

Below, we will first visualize some obstacle detection results on steep ramp, then test the effect of the resolution along height dimension on curb detection (low obstacle), and finally, show the experiment to test our method's stableness and the comparison between other VScan generation methods. Readers can check more results on this paper's video.

Fig. 7. Autonomous Vehicle "Matsu" equipped with Velodyne 64E-S2 and Grasshopper 3 HD Camera.

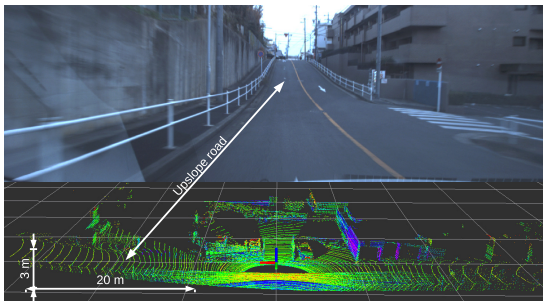

Fig. 8. The image and point cloud of an upslope road. The average slope is about 8.5 degree.

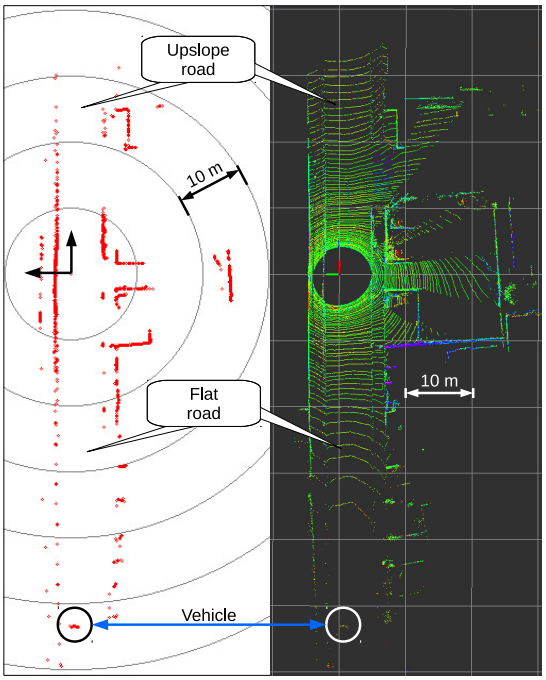

Fig. 9. The VScan generated on upslope road (left) and the corresponding Velodyne data (right). At bottom, a vehicle is detected behind our vehicle within 60m.

A. Obstacle Detection on Steep Ramp

As mentioned in Section IIIA, a rotation of length-height coordinates is needed to convert downslope road to upslope road. Below, we will separately show the obstacle detection results on upslope and downslope road. The resolution step along height dimension is set as $\delta = 0.2m$, so that the low obstacle road curb will not be always detected in the results.

1) Upslope Road: Fig. 8 presents an upslope road scene. Our method can successfully filter out the entire road and keep the road fence in near place (for the view angle of Velodyne, there is not enough points to form obstacle at far place on upslope road) as shown in Fig. 9.

Moreover, there are other two points to be noticed. First is that our method can successfully detect the obstacle within nearest ring (the road fence on the left of ego-vehicle). Second is that our method successfully detect a vehicle obstacle behind ego-vehicle within 60m on flat road as shown in Fig. 9 lower part, where points are very sparse.

2) Downslope Road: Fig. 10 presents the obstacle detection result from a downslope road scene (see Fig. 1). This scene contains a downslope and then upslope road in front of ego-vehicle, a downslope road behind ego-vehicle, and a flat road on the right of ego-vehicle. Therefore, we can demonstrate not only the rotation's effect on our method works on downslope road, but also that the rotation will not affect road surface filter on flat and upslope road.

There are two obstacle detection results in Fig. 10. the left one uses rotation and the right one does not. We find that with the rotation, our method successfully filter out all the road surface and detect the road fence and the vehicle behind ego-vehicle within 50m. While for the result without rotation, it has two shaded areas. In the shaded areas, we find that the wall is detected instead of road fence and the vehicle behind ego-vehicle is also not detected.

B. Low Obstacle Detection

In order to detect low obstacle, δ should be small. In our experiment, we choose road curb as the low obstacle to be detected. The maximum height of road curb in our dataset is around 20cm and the minimum one is around 5cm. Fig.11 shows the curb detection results. We find that with large height grid step ($\delta = 0.2m$), the curb can not be detected, while, with small height grid step ($\delta = 0.05m$), the curb can be clearly detected. Therefore, we demonstrated that with small height grid step, our method can detect low obstacle. However, we also find that with small height grid step, the obstacle detection result is more sensitive to the sensor noise. For example, in Fig.11, the obstacle "wall2" with 0.05cm height grid step has many noise points and is not smooth like that with 0.2cm height grid step.

C. Obstacle Detection Stableness

To demonstrate the method's stableness, we accumulate 100 frames (10s) together to check static obstacle's stableness and dynamic obstacle's continuity. We choose one scene in the dataset (Fig. 12), where the ego-vehicle stopped at a cross-road. The blue one represents oldest frame and the red one represents latest frame. The detection result of our method is shown in the left of Fig.12. We find that the static wall is detected stably and the road curb could be detected with $\delta = 0.05m$. However, some noise points are also detected. For the dynamic obstacle, we find that there

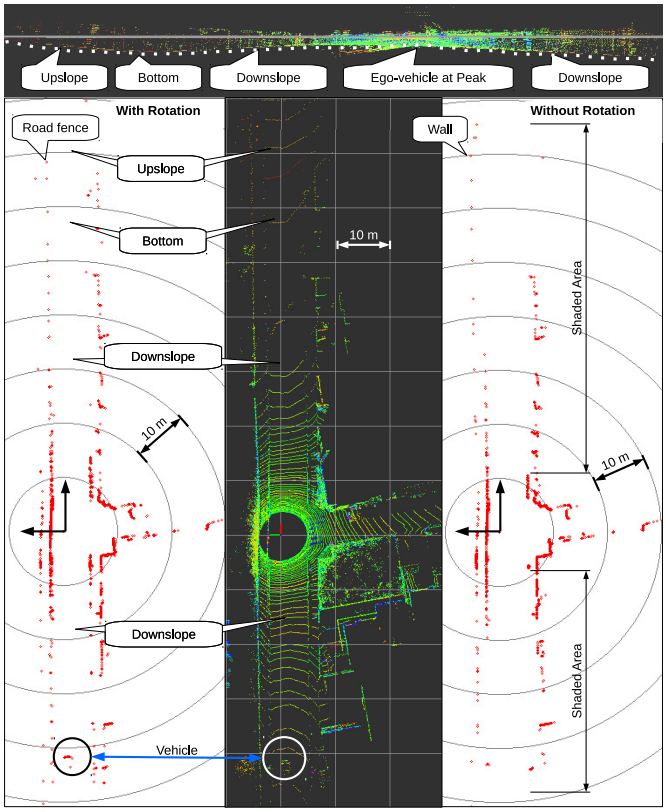

Fig. 10. The result of obstacle detection and the Velodyne data on downslope road. On the left, it is the detection result with rotation. On the right, it is the detection result without rotation.

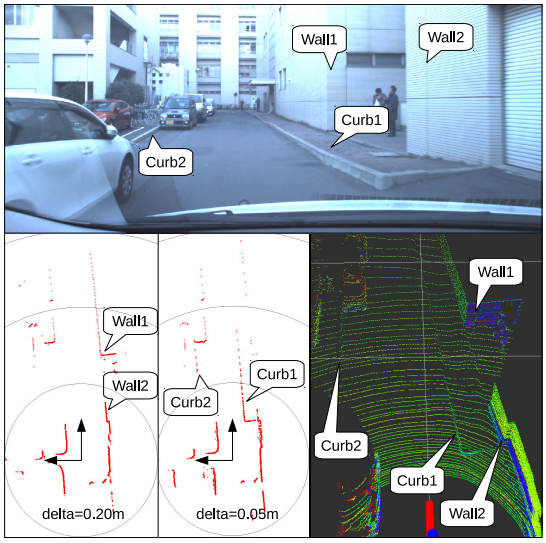

Fig. 11. Curb detection result on a flat road. Two curbs exist in this scene and our obstacle detection method with $\delta = 0.05m$ successfully detected them.

is a pedestrian appeared at the beginning of the accumulated frames (blue trajectory), and the most important thing is that a dynamic vehicle was continuously detected from 80m to 8m behind the ego-vehicle. This is a great evidence to demonstrate our method's stableness on obstacle detection.

To further demonstrate its stableness, we compared our method with other two methods. The first one is a naive implementation on depth image based VScan generation method from [6] (Fig. 12 middle). The second one is the basic VScan generation method from ROS [5] (Fig. 12 right).

For the depth image based method, the calculation of normal vector is sensitive to sensor noise and data density. Meanwhile, the point cloud from Velodyne is noisy at near palce and is sparse at far place. Therefore, the precise calculation of normal vector is hard to achieve. In our experiment, the road points from 30m to 70m in front of ego-vehicle are sparse, many of them are falsely detected as obstacle for the wrong normal vector calculation. The vehicle behind ego-vehicle was detected from 60m to 10m. However, we find that the shape of the detected vehicle is not stable.

For the basic VScan method, it is the easiest way to get VScan from 3D point cloud and it is also a simulation of 2D LiDAR. For the upslope road, the basic VScan was shaded by road surface in front and it can only detect the vehicle from 35m to 10m. This is also the limitation of 2D LiDAR as well as the 2D applications based on it, and our VScan generation method provides a way to enable many 2D applications to work in such situation.

V. Conclusion

In this paper, we proposed a robust real-time VScan generation method for obstacle detection in complex urban environment. With the development of BVSM and the corresponding *simultaneous road filtering and obstacle detection method*, this method can work on steep ramp with large slope, and can detect low and overhung obstacle like curb and barrier gate. Meanwhile, with the development of *sorted array based acceleration method*, this method can generate VScan with 2000 beams within 15ms. Additionally, by using *Stixel*, this method conserves minimum and maximum height information of detected obstacle, which has great potential application.

Moreover, the robust VScan bridges the new-born 3D LiDAR and matured applications based on 2D LiDAR, and enables these 2D applications to work in complex urban environment. For example, we have deployed our VScan generation method in occupancy grid map and vehicle tracking. The results are very stable for many kinds of urban environments (see attached video).

Appendix

Property A

if $g_c - g_f > 1$, then

$$\begin{aligned} L'_i(g_f, g_c) &\leq L'_i(g_f + 1, g_c) \\ and \quad L'_i(g_f, g_c) &\leq L'_i(g_f, g_c - 1) \end{aligned} \tag{5}$$

Proof: the point-set within $[h_f + \delta, h_c)$ and $[h_f, h_c - \delta)$ is the subset of the point-set within $[h_f, h_c)$.

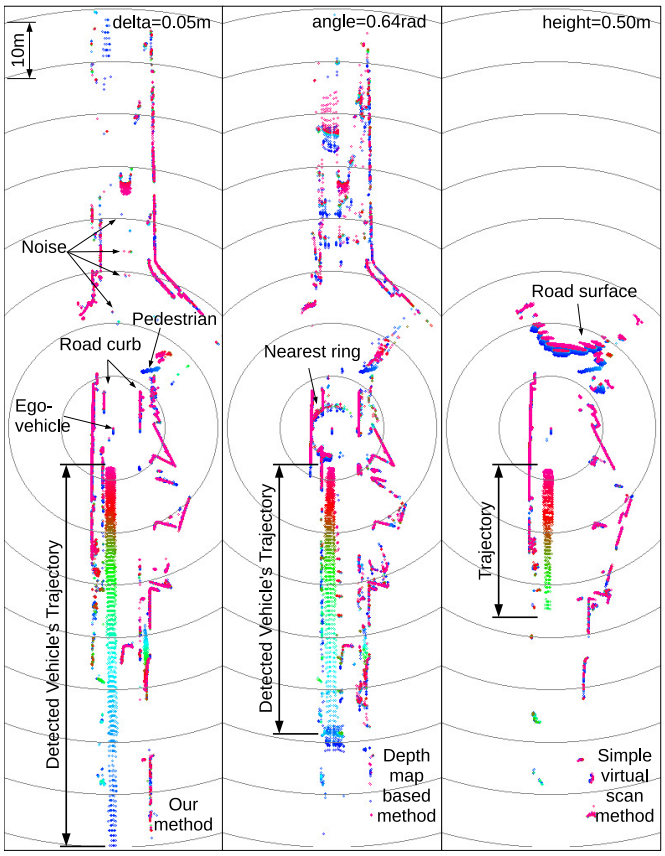

Fig. 12. 100 frames (10s) accumulated result of our method (left), depth map based based method (middle) and basic VScan method (right). Blue represents oldest frame and red represents latest frame.

Property B

If $g_f + 1 < g_c$, then

$$L'_i(g_f, g_c) = min(L'_i(g_f + 1, g_c), L'_i(g_f, g_c - 1)) \quad (6)$$

Proof: the point-set within $[h_f, h_c)$ equals the union of the point-sets within $[h_f + \delta, h_c)$ and $[h_f, h_c - \delta)$.

Property C

Take all the diagonal elements $\{L'_i(g_f, g_f+1)\}$ of BVSM. Then sort them with their length L'_i in ascending order and for the elements have same length, sort them with their floor height g_f in descending order. After sorting, we get two associated arrays: beam length array $L'_i[j]$ and height grid index array $g_f[j]$.

For any two array indices j_0 and j_1 ($j_0 < j_1$), if $g_f[0] \leq g_f[j_0] < g_f[j_1]$ and there does not exist a j, where $j_0 < j < j_1$ to make $g_f[j_0] < g_f[j] < g_f[j_1]$, then we could calculate BVSM's non-diagonal element $L'_i(g_f, g_c)$, where $g_f[j_0] \leq g_f < g_f[j_1]$ and $g_f + 1 < g_c \leq g_f[j_1]$ as below.

$$L'_i(g_f, g_c) = \begin{cases} L'_i[j_0] & (g_f = g_f[j_0]) \\ L'_i[j_1] & (g_f[j_0] < g_f < g_f[j_1]) \end{cases} \quad (7)$$

Proof: there does not exit a j, where $j_0 < j < j_1$ to make $g_f[j_0] < g_f[j] < g_f[j_1]$, then according to the sort rule, $L'_i[j_0] < L'_i[j_1]$ and $L'_i[j_1]$ is the minimum beam length in diagonal elements subset $\{L'_i(g_f, g_f + 1) | g_f[j_0] < g_f \leq g_f[j_1]\}$. With the property B, the property C is proved.

REFERENCES

[1] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.

[2] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. Springer Science & Business Media, 2009, vol. 56.

[3] Velodyne Lidar, Inc., *HDL-64E user's manual*, 2008. [Online]. Available: www.velodyne.com

[4] A. Petrovskaya, M. Perrollaz, L. Oliveira, L. Spinello, R. Triebel, A. Makris, J.-D. Yoder, C. Laugier, U. Nunes, and P. Bessiere, "Awareness of road scene participants for autonomous driving," in *Handbook of Intelligent Vehicles*. Springer, 2012, pp. 1383–1432.

[5] T. Foote, "pointcloud_to_laserscan ROS node." [Online]. Available: http://wiki.ros.org/pointcloud_to_laserscan

[6] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion," in *Intelligent Vehicles Symposium, 2009 IEEE*. IEEE, 2009, pp. 215–220.

[7] M. Li and Q. Li, "Real-time road detection in 3d point clouds using four directions scan line gradient criterion," *Future*, vol. 5, 2009.

[8] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[9] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[10] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, *et al.*, "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.

[11] N. Wojke and M. Haselich, "Moving vehicle detection and tracking in unstructured environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3082–3087.

[12] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[13] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 5, pp. 694–711, 2006.

[14] I. Ulrich and I. Nourbakhsh, "Appearance-based obstacle detection with monocular color vision," in *AAAI/IAAI*, 2000, pp. 866–871.

[15] K. Huh, J. Park, J. Hwang, and D. Hong, "A stereo vision-based obstacle detection system in vehicles," *Optics and Lasers in Engineering*, vol. 46, no. 2, pp. 168–178, 2008.

[16] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele, "Stereo vision-based vehicle detection," in *IEEE Intelligent Vehicles Symposium*. Citeseer, 2000, pp. 39–44.

[17] R. Labayrade, D. Aubert, and J.-P. Tarel, "Real time obstacle detection in stereovision on non flat road geometry through" v-disparity" representation," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2. IEEE, 2002, pp. 646–651.

[18] F. Erbs, A. Barth, and U. Franke, "Moving vehicle detection by optimal segmentation of the dynamic stixel world," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 951–956.