Robust Virtual Scan for Obstacle Detection in Urban Environments*

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

***Abstract*— Obstacle detection is an essential technique for intelligent vehicles. Environmental sensing especially plays a vital role to achieve accurate obstacle detection. Unlike classical 2D scan, emerging 3D Light Detection and Ranging (LiDAR) sensors can scan dense point cloud at one time, which represents detailed information of urban environments. The downside of obstacle detection using 3D LiDAR, on the other hand, is its computational cost posed by a large amount of 3D data. The *virtual scan* (VScan), first introduced by Petrovskaya et al. [1] for efficient vehicle detection and tracking, is a 2D compression of 3D point cloud to represent free space, obstacles and unknown areas. To overcome the computational problem of obstacle detection using 3D LiDAR, therefore, VScan is suitable. In addition, it can bridge across new-born 3D LiDAR sensors and many matured applications based on 2D scan, including occupancy grid map, SLAM, planning, detection, and tracking, due to its 2D representation of 3D point cloud.**

A key challenge to VScan for intelligent vehicles is that we must improve robustness of VScan in complex urban environments. For example, steep ramps with large slope, low curbs along the road, and overhung barrier gates at the entrance often make VScan mis-behave.

In this paper, we present a robust VScan generation method for intelligent vehicles driving in complex urban environments. Our method uses a new data structure, called *basic VScan matrix* (BVSM), to represent 3D point cloud around the own vehicle. We also develop (i) a *simultaneous road filtering and obstacle detection method* that works on top of BVSM to generate robust VScan generation, and (ii) a *sorted array based acceleration method* to perform the VScan generation in real-time. Experimental results from field testing, where steep ramps, barrier gates, and curbs are contained, demonstrate that our method improves the robustness and the speed of VScan generation as compared to traditional methods. The computational cost of generating VScan with 2000 beams is limited within 15ms. The result of obstacle detection on the road with large slopes is stable in the range of 60m with a few number of false positives.

I. INTRODUCTION

Obstacle detection is an essential part of the perception capabilities of intelligent vehicles. The currently deployed technologies such as platooning, adaptive cruise control, and crash-avoidance systems, rely on accurate obstacle detection. In particular, a recent paradigm shift toward autonomous driving has highlighted the importance of obstacle detection in urban environments. Results of obstacle detection, therefore, must be robust against traffic features to guarantee safety, including steep ramps with large slopes, small curbs along the road, and overhung barrier gates at the entrance, as shown in Fig. 1.

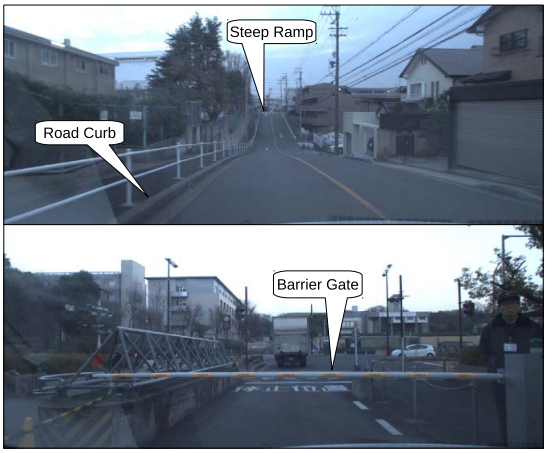

Fig. 1. A ramp with a large slope (upper) and a barrier gate at the entrance (lower). These scenes are photographed by the vehicle driving around our university campus.

Many intelligent vehicles, including all entrants to the DARPA Urban Challenge [2], rely on Light Detection and Ranging (LiDAR) sensors for obstacle detection as well as localization and mapping. Among these LiDAR sensors, for example, Velodyne HDL sensors can scan the real world by 3D point cloud 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 sensing solutions for obstacle detection in complex urban environments.

Data preprocessing or clean-up processing is a useful technique to augment physical sensors. For range data, it is common to sub-sample the rays, projecting from 3D point cloud to 2D scan, and reject outliers [4]. An example of preprocessing 3D LiDAR data includes a 2D compaction to *virtual scan* (VScan) [1], which is known to be effective for vehicle detection and tracking.

VScan is a set of beams emitted from a center to conserve information with respect to free space, obstacles and unknown areas, which is corresponding to 2D scan's range data except that VScan is flexible to obstacles with different heights. In terms of robustness and real-time processing, VScan is suitable for obstacle detection in urban environments.

A key challenge to VScan for intelligent vehicles is that we must improve the robustness of VScan in complex urban environments. The robustness of VScan is dominated

*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, {e_takeuchi, shinpei}@is.nagoya-u.ac.jp

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

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

by an algorithm of transforming 3D point cloud to 2D scan. As a case study, we have applied the basic VScan generation module "*pointcloud_to_laserscan*" [5] provided by Robot Operating System (ROS), a popular intelligent robot middleware, in different setups of field testing. It works well for such an urban environment that only contains flat roads, but it is not robust against such a complex urban environment that contains sloped road surfaces in a sense that these sloped road surfaces are mis-detected as obstacles. This means that robust VScan needs to filter out ground readings separating real obstacles and roads.

Many ground filter methods have been studied recently for intelligent vehicles [1], [6], [7], [8], [9], [10], [11], [12]. These methods can detect objects and obstacles even on sloped roads, using various techniques such as the normal vector, range images, occupancy grid maps, the measurement of consecutive readings. However, the following problems still remains open to practical VScan generation methods:

1) **Robustness:** First, it must be able to cope with sloped roads and frequent changes of vehicle's roll and pitch in urban environments. Second, it must be able to detect low obstacles such as road curbs and overhung barrier gates. Third, the result of obstacle detection must be stable within a certain range of distance.
2) **Speed:** It is preferred to set as many beams as possible in VScan to achieve high robustness. Meanwhile, it is also preferred to limit the computational cost of VScan generation to execute in real-time.

Contribution: In this paper, we present a robust and fast VScan generation method for obstacle detection in complex urban environments. Using our method, for Velodyne LiDAR sensor's rotational resolution, where the number of beams in VScan is set to 2000, the computational cost of VScan generation is limited within 15ms and the result of obstacle detection on sloped roads is stable in the range of 60m with a few number of false positives. To summarize, the main contributions of this paper are the following:

1) A new data structure called *basic VScan matrix* (BVSM) representing point cloud around the vehicle.
2) A *simultaneous road filtering and obstacle detection* method that works on top of BVSM for robust VScan generation.
3) A *sorted array based acceleration method* for fast VScan generation.

Organization: The rest of this paper is organized as follows. Section II introduces related work. Section III presents our obstacle detection methods using VScan. Section IV provides experimental results conducted with the real vehicle as shown in Fig. 7 around the university campus, where ramp ways, barrier gates and curbs exist, to demonstrate the robustness and the speed of our VScan generation method. This paper concludes in Section V.

II. Related Work

Obstacle detection is a core task of intelligent vehicles. The most popular sensors used in obstacle detection are cameras and LiDAR sensors (laser range finders).

Vision-based obstacle detection for driver assistance has received considerable attention over the past decade [13]. A variety of vision-based object detection methods have been developed [14], [15], [16], [17], [18] using monocular and stereo cameras. Especially, for [18], they provide a heuristic segmentation method to separate moving objects from the static background like road surfaces, while the vision-based depth and motion information is transferred into stick pixels often called *Stixels*, 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 coordinates, but also reserve their height information as *Stixel* with minimum and maximum heights. As a consequence, our VScan generation method is a form of 3D obstacle detection method.

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

Although some of these methods can work on sloped roads, they cannot fully satisfy the requirements of practical use, i.e., the robustness and the speed of VScan generation are inadequate. For example, the normal vector based method requires a normal calculation for every point, which is often a time-consuming process. [6] reported that the average computational cost of normal calculation is 352ms on a desktop-class computer (Pentium M with 2.1GHz). According to [6], the occupancy grid map based methods are not suitable for obstacle detection in complex urban environments with vegetation, hills, and curbs. [1], [11] filter out roads by the measurement of consecutive readings for each bearing, and thus they cannot always detect overhung obstacles. In [12]'s polar coordinates, the resolution along a radius dimension is set to 3m, which is used to produce candidates of ground points for road slope evaluation. However, with such a low resolution along a radius dimension, this method could not get accurate length estimates of VScan beams and also not be sensitive to low obstacles such as road curbs and overhung barrier gates.

Our VScan generation method is originated from [1] while holding their two features: 1) road filtering is independently conducted on each beam in VScan; 2) obstacles are detected by calculating a road slope and comparing it with a preset slope threshold. Meanwhile, in order to detect overhung obstacles, we develop a new data structure called *basic VScan matrix* (BVSM). Combined with the two features of [1]

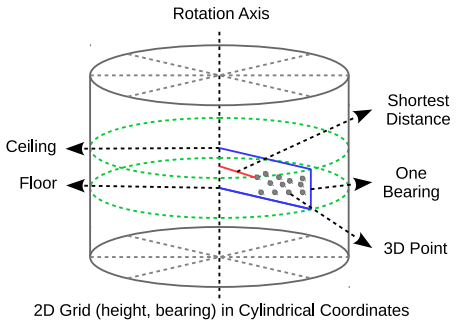

Fig. 2. 2D grid (bearing, height) in cylindrical coordinates and the illustration of basic VScan generation method.

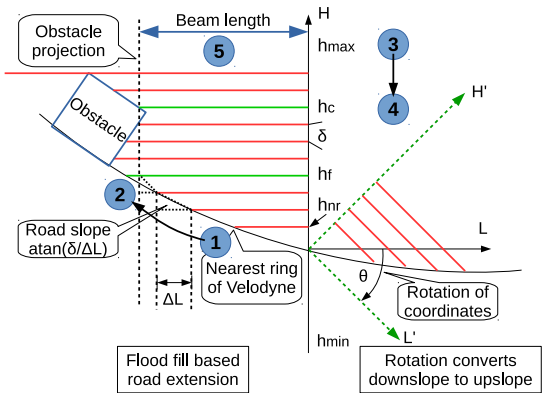

Fig. 3. Illustration of VScan generation method. The left part shows simultaneous road surface extension and obstacle detection. The right part shows the rotation method to cope with downslope roads.

mentioned above, a *simultaneous road filtering and obstacle detection method* that works on top of BVSM is developed. Finally, in order to achieve fast VScan generation with a large number of beams, a *sorted array based acceleration method* is developed.

In conclusion, our VScan generation method is supposed to outperform previous work in the following. First, it is robust against steep ramps with large slopes, and can detect low and overhung obstacles such as curbs and barrier gates Second, it is fast enough to generate 2000 beams in VScan within 15ms. Finally, it is based on a 3D obstacle detection method by using the *Stixel* to conserve the minimum and maximum heights of detected obstacles.

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

The measurement of consecutive readings in [1] cannot detect overhung obstacles directly[1] due to that they only focused on local features derived from three adjacent readings. In order to detect overhung obstacles, more global features need to be considered.

In our method, we project all points into a 2D grid (bearing, height) in cylindrical coordinates as shown in Fig. 2. 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 the *basic VScan generation* [5] to determine the beam length of **one bearing** in the basic VScan as the **shortest distance** between the **rotation axis** and the nearest **3D 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 results with different height ranges $[h_f, h_c)$, we develop a new data structure called *basic VScan matrix* (BVSM) as shown in Fig. 4. The detail of BVSM will be discussed in Section III-B.

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

As shown in the left part of Fig. 3, SRFOD starts from $[h_{min}, h_{max})$ (manually set accordingly from the road slope and the height of overhung obstacle) to find the best height range $[h_f, h_c)$. We first need to find the height of the nearest ring of 3D LiDAR h_{nr} ①, and then we use a flood fill method to extend the road surface toward an obstacle with a preset slope threshold to get the floor h_f ② (named as the road surface filtering stage). After touching the possible obstacle, we start from h_{max} ③ and then decrease the ceiling to h_c ④ to detect the obstacle (named as the obstacle detection stage). Finally, after several alternations between these two stages, we get the beam length from the best height range $[h_f, h_c)$ ⑤ for the basic VScan generation.

Moreover, our method is based on a flood fill method along a height dimension; therefore, this method could only handle flat and upslope roads, but not for downslope roads, because the obstacle is shaded by the downslope road surface. To solve this problem, we use a rotational transformation to enable our method to work with downslope roads. As shown in the right part of Fig. 3, after the rotation of the length-height coordinates, a downslope road is converted to a upslope road in a new $L' - H'$ coordinates, and its geometry property is still conserved. Therefore, we can generate VScan in the rotated coordinates first, and then we inversely rotate the generated VScan to get the final obstacle detection result.

B. Basic VScan Matrix Construction

We define $L_i(h_f, h_c)$ as the beam length in ith bearing of a basic VScan from the height interval $[h_f, h_c)$ (in the

[1]They implement an additional process for the overhung obstacle detection based on the "elevation method" and "flat ground assumption" after filtering out the road surface.

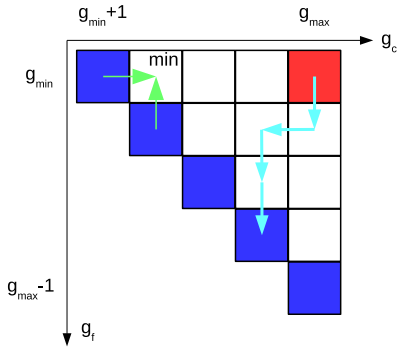

Fig. 4. Basic VScan matrix to store all possible basic VScan. Green arrows illustrate the way to calculate non-diagonal elements. Cyan path illustrates the way to extend road surfaces and detect obstacles.

following, we omit the notation i for convenience), and create a grid along the height dimension with a step δ in the height interval $[h_{min}, h_{max})$. Therefore, $L(h_f, h_c)$ can be discretized as $L'(g_f, g_c)$, where $g(h) = \lfloor (h - h_{min})/\delta \rfloor$, and then we can define all possible basic VScan results from different height ranges as the set expressed by (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 this set (1) as well as the corresponding BVSM, and this order guarantees the correctness of our obstacle detection method.

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

It is time-consuming to directly calculate all elements in BVSM. However, the property B (see Appendix) enables us to conduct the dynamic programming on the BVSM. As shown in Fig. 4, the diagonal elements in the blue color are the basic VScan results from the corresponding smallest height intervals, and they can be directly and individually calculated by projecting the corresponding 3D points into their own 2D grid cells (the cylindrical coordinates in Fig. 2). For the other elements in the BVSM, they equal to the smaller value of their left and lower elements illustrated by the green arrows in Fig. 4.

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

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

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

Assuming that the method is at a given BVSM's cell (g_f, g_c) and it has a beam length $L'(g_f, g_c)$, which represents the beam length of the basic VScan generated from the height range $[h_f, h_c)$, our method needs to distinguish roads and obstacles at height h_f. The decision tree is shown as below, and there are three possible results: road, unknown area, and obstacle.

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

$$\Delta L = L'(g_f + 1, g_c) - L'(g_f, g_c) \geq 0 \text{ and } \Delta h = h_c - h_f$$

The first layer of the decision tree utilizes a flood fill method with a slope threshold to detect road. According to the definition of $L'(g_f, g_c)$ and the property A, if $\Delta L > 0$, $\Delta L \leq L'(g_f + 1, g_f + 2) - L'(g_f, g_f + 1)$ must be satisfied; 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 a road if $atan(\delta/\Delta L) \leq \phi$.

The second layer of the decision tree detects whether there is an obstacle in the condition of $atan(\delta/\Delta L) > \phi$, which indicates that a large slope value exists in the height interval $[h_f, h_c)$. We use the height difference $\Delta h = h_c - h_f$ to filter out fake obstacles highly elevated above the ground, such as tree tops and overpasses; therefore, if $\Delta h > \Delta H$, there may exist a fake obstacle. Otherwise, we know that there is an obstacle in the height interval $[h_f, h_c)$, and it could be an overhung obstacle because of the relatively global feature.

Based on the decision result, our method will move on the BVSM or stop for the 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 $L'(g_f, g_c)$ (⑤ in Fig. 3).

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

D. Sorted Array Based Acceleration Method

We have presented our VScan generation method for the obstacle detection, and it consists of two consecutive but independent steps: 1) the BVSM construction, 2) the SRFOD method on the BVSM. In this section, we will analyze its computational complexity for real-time implementation.

Assume that the number of 3D points is P, the required 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 the diagonal elements in the BVSM, and $O(NM^2)$ is to calculate the other elements in the BVSM using the dynamic programming. The computational complexity of the second step is $O(NM)$, which is to roam on the BVSM. Therefore, its final computational complexity is $O(P+NM^2)$, and the memory cost is $O(NM^2)$ (BVSM).

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

From the property C, we need to sort all BVSM's diagonal elements according to their length $L'(g_f, g_f + 1)$ in an ascending order, and for the elements have the same length, we need to sort them according to their floor height g_f in a descending order. After sorting, we get two associated arrays: beam length array $L'[j]$ and height grid index array $g_f[j]$. From the property A, we know that $L'[0]$ 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 the BVSM's non-diagonal elements $L'(g_f, g_c)$, where $g_f \geq g_{nr}$, from two elements of the sorted arrays. Therefore, we can only keep the BVSM's diagonal elements and thus reduce the memory cost from the matrix form $O(NM^2)$ to the array form $O(NM)$. For the computational complexity, instead of constructing the 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 the BVSM-based method.

For the second step of the BVSM-based method, every non-diagonal element checked by the SRFOD method can be directly calculated by two corresponding diagonal elements in the sorted array. Instead of moving down or left from the BVSM's red corner cell to the diagonal element (Fig. 4), this method starts from the first two elements in the sorted array, and then move through the sorted array to simultaneously filter out roads and detect obstacles. 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 $j_1 = 1$. Assuming that this method is given two indices j_0 and j_1, it needs to distinguish roads and obstacles at height $h_f[j_0]$. The decision tree is shown as below and there are four possible results: road, unknown area, obstacle, and obstacle*.

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

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

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

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

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

Based on the decision result, our method will move through the sorted array or stop for the 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 equals to $L'[j_0]$.
4) Obstacle*: the beam length of VScan equals to $L'[j_1]$.

In conclusion, the computational complexity of the SAAM is $O(P + NMlog(M))$ and the memory cost is $O(NM)$ (the sorted array). As shown in Fig. 5, we compared the computational cost to generate a VScan with 2000 beams between the BVSM-based method and the SAAM. We find that the BVSM-based method is very slow for the small height step, while the computational cost of the SAAM method is capped around 10ms (Intel Xeon ES-1660 3.70GHz) in our experiment. For [1], it reports that it can generate 720 beams (0.5 degree for angular resolution) at 40Hz. Therefore, our method with additional overhung obstacle detection is fast enough for real-time applications.

E. 3D Obstacle Detection with Stixel

Our 3D obstacle detection method uses *Stixel* to conserve the minimum and maximum height information of detected obstacles (Fig. 6). For the minimum height, it is determined when the obstacle is detected. For the maximum height, the method will keep moving on the BVSM (or the sorted array) after the obstacle detection, and it is determined when the "road" is detected again according to (2) (or (3,4)). Since the method simultaneously filters out road surfaces and get VScan, the minimum height of overhung obstacles is actually the height of their projection on roads as shown in Fig. 6 (the barrier gate example).

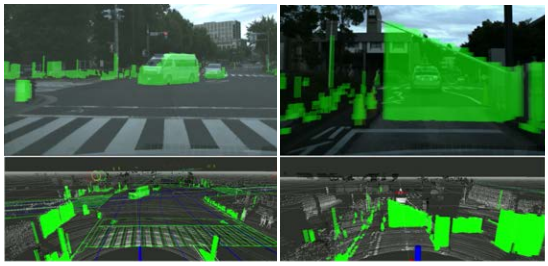

Fig. 6. An example of Stixel with the minimum and maximum height information from our VScan generation method. The left shows vehicle detection and the right shows barrier gate detection.

Fig. 7. The experimental vechile equipped with a Velodyne 64E-S2 sensor and a Point Grey Grasshopper 3 camera (also with other sensors not used in this experiment).

IV. Experimental Evaluation

In this section, we demonstrate how our VScan generation method works for complex urban environments including sloped roads, steep ramps, low curbs, and overhung barrier gates. We use a vehicle equipped with a Velodyne HDL-64E S2 sensor and a Point Grey Grasshopper 3 camera, as shown in Fig. 7. The total traveling distance for this evaluation is about $10km$.

A. Obstacle Detection on Steep Ramp

As mentioned in Section III-A, a rotation of length-height coordinates is required to convert a downslope road to an upslope road. We show the obstacle detection results on upslope and downslope roads respectively. The resolution step along a height dimension is set to $\delta = 0.2m$, so that the low obstacles like road curbs are not always detected in the results.

1) Upslope Road: Fig. 8 presents an upslope road scene. Our method successfully filters out the entire road and keep the road fence in the near place as shown in Fig. 9. Note that for the view angle of Velodyne, there are not a sufficient number of points to form obstacles at a far place on this upslope road. It is intriguing to see that our method also successfully detects obstacles within the nearest ring (the road fence on the left side of our vehicle). Furthermore, our method successfully detects vehicle obstacles behind our vehicle within 60m on a flat road, as shown in the lower part of Fig. 9, where points are very sparse.

2) Downslope Road: Fig. 10 presents the result of obstacle detection from a downslope road scene (see Fig. 1 upper image). This scene contains a downslope road as well as an upslope road in front of our vehicle. Therefore, we can

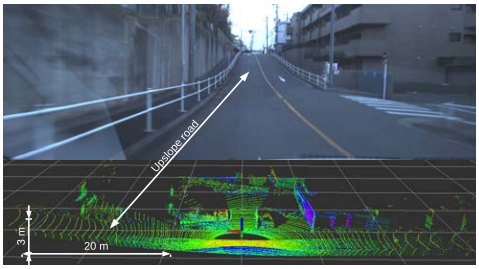

Fig. 8. A combined image and point cloud of an upslope road. The average slope is about 8.5 degree.

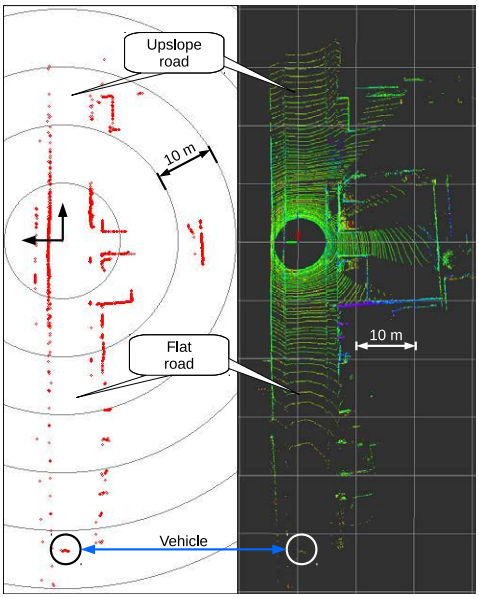

Fig. 9. The generated VScan on an upslope road (left) and the corresponding Velodyne data (right). At the bottom, a vehicle is detected behind our vehicle within 60m.

demonstrate not only that the rotation's effect on our method works on a downslope road, but also that the rotation will not affect road surface filters on both flat and upslope roads.

There are two different results of obstacle detection shown in Fig. 10. One in the left uses a rotation while the other in the right does not. We find that with the rotation, our method successfully filters out all the road surfaces, which results in being able to detect road fences and vehicles behind our vehicle within 50m. On the other hand, the result without the rotation shows that it has two shaded areas, where we find that walls are detected instead of road fences, and the vehicle behind our vehicle is also not detected.

B. Low Obstacle Detection

To detect low obstacles, δ should be small. In this experiment, we choose a road curb as a low obstacle to be detected. The maximum height of the road curb in our dataset is about 20cm, while the minimum is about 5cm. Fig. 11 shows the result of curb detection. We find that with a large height grid step ($\delta = 0.2m$), the curb cannot be detected well, while with a small height grid step ($\delta = 0.05m$), the curb can be clearly detected. Therefore, with a small height grid

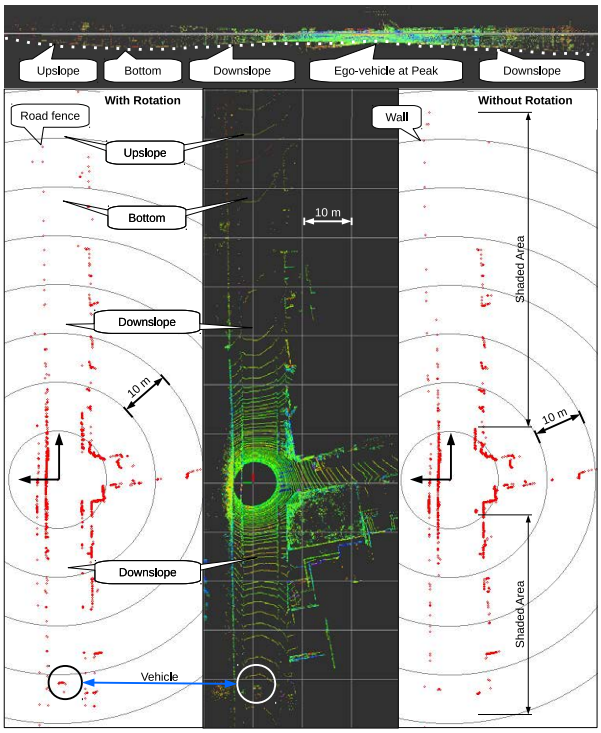

Fig. 10. The result of obstacle detection and the Velodyne data on a downslope road. On the left side, we show the detection result with a rotation. On the right side, we show the detection result without a rotation.

step, our method works better; however, we also find that with a small height grid step, the result of obstacle detection is more sensitive to the sensor noise. For example, in Fig. 11, the obstacle "wall2" with the 0.05m height grid step has many noise points and is not smooth like that with the 0.2m height grid step.

C. Obstacle Detection Stableness

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

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

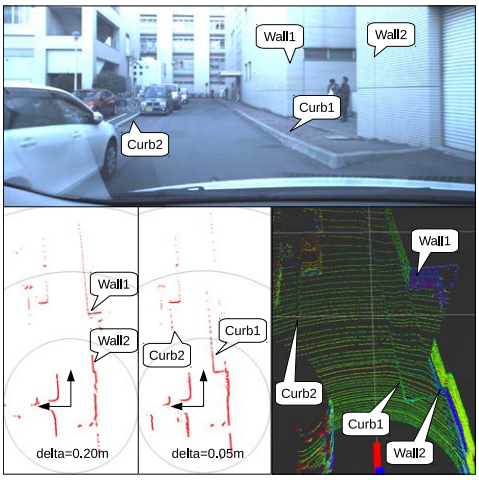

Fig. 11. The result of curb detection on a flat road. Two curbs exist in this scene, and our obstacle detection method with $\delta = 0.05m$ successfully detects them.

For the depth image based method, the calculation of the normal vector is sensitive to the sensor noise and the data density. The point cloud from the Velodyne sensor is noisy at the near place and is sparse at the far place; therefore, the precise calculation of the normal vector is hard to achieve. In this experiment, the road points from 30m to 70m in front of our vehicle are sparse. Many of them are falsely detected as obstacles for the wrong normal vector calculation. The moving vehicle behind our vehicle is detected from 60m to 10m. However, we find that the shapes of the detected vehicles are 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 scan. For the upslope road, the basic VScan was shaded by road surfaces in front and it can only detect vehicles from 35m to 10m. This is also a limitation of 2D scan as well as 2D applications based on it, and our VScan generation method provides a way to allow many 2D applications to work in such a situation.

V. Conclusion

In this paper, we have presented a robust and fast VScan generation method for the obstacle detection in complex urban environments. With the development of BVSM and the corresponding SRFOD method, this method sucessfully works on steep ramps with large slopes, and detects low and overhung obstacles such as curbs and barrier gates. Furthermore, with the development of SAAM, we can generate a VScan with 2000 beams in 15ms. Additional applications of VScan presented in this paper include *Stixel*, which conserves minimum and maximum heights of detected obstacles to represent their shapes. The video of our field testing with VScan has been uploaded to YouTube[2]. The source code is integrated into the Autoware[3] [19] developed by Nagoya University.

[2]https://www.youtube.com/watch?v=d6l31owNs2U
[3]https://github.com/CPFL/Autoware

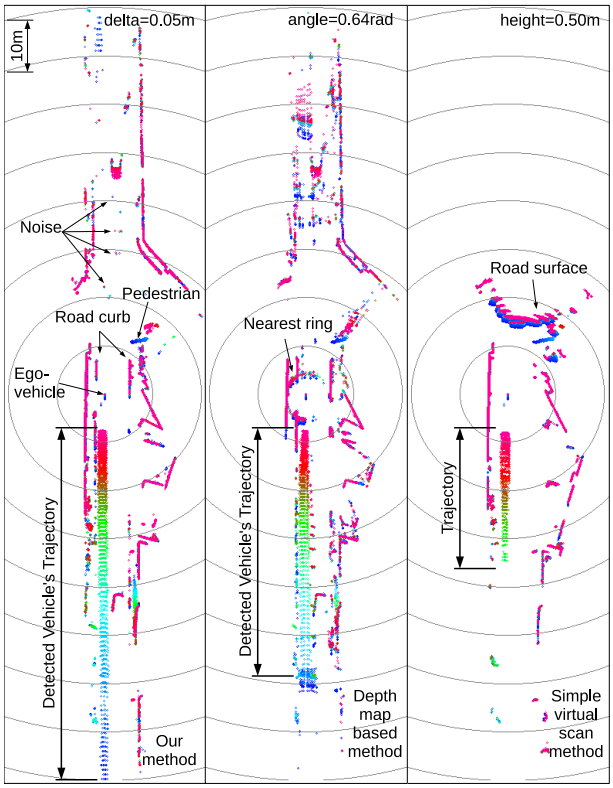

Fig. 12. The result of our method (left), the depth map based method (middle) and the basic VScan method (right). The blue represents the oldest frame and the red represents the latest frame. In total, 100 frames (10s) are accumulated.

In future work, we would like to extend the presented VScan generation method to detect more obstacles in a far range. We also plan to use the generated VScan to track moving objects such as vehicles and pedstrians.

APPENDIX

Property A

If $g_f + 1 < g_c$, then

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

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

Property B

If $g_f + 1 < g_c$, then

$$L'(g_f, g_c) = min(L'(g_f + 1, g_c), L'(g_f, g_c - 1)) \tag{6}$$

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 $\{L'(g_f, g_f + 1)\}$ of BVSM. Then sort them with their length L' 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 $L'[j]$ 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 $L'(g_f, g_c)$, 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.

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

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, $L'[j_0] < L'[j_1]$ and $L'[j_1]$ is the minimum beam length in diagonal elements subset $\{L'(g_f, g_f + 1) | g_f[j_0] < g_f \leq g_f[j_1]\}$. 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.

[19] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An Open Approach to Autonomous Vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 66–69, 2015.