Precise and Efficient Model-Based Vehicle Tracking Method Using Rao-Blackwellized and Scaling Series Particle Filters

Mengwen He[1*,3*,5], Eijiro Takeuchi[1,3], Yoshiki Ninomiy[2,3], and Shinpei Kato[3,4]

[1]Graduate School of Information Science, Nagoya University (* left since April, 2016)
[2]Institute of Innovation for Future Society (MIRAI), Nagoya University
[3]JST/COI, Nagoya (* left since April, 2016)
[4]Graduate School of Information Science and Engineering, the University of Tokyo
[5]Department of Electrical and Computer Engineering, Carnegie Mellon University

Objective

- Develop a precise and efficient high-dimensional particle filter based vehicle tracking method for the intelligent vehicle to interact with the surrounding vehicles on the road (Fig. 1).

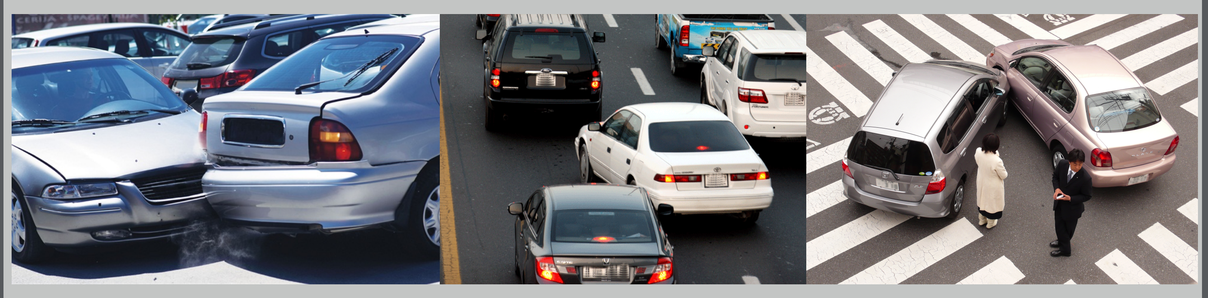

Figure 1: Precise and efficient tracking of a target vehicle can sensitively detect the driver's intension and will lead to reliable predictions for ensuring safe driving. For example: (Left) in the parking-lot, when a parked vehicle starts moving out, its small position change should be tracked for collision avoidance; (Middle) on the road, when a running vehicle tends to change lane, its small orientation change should be tracked for safe overtaking; (Right) at the intersection without traffic light, when a speedy vehicle comes out, its speed change should be tracked for deciding whether to pass the intersection.

Introduction

- Vehicle tracking is an important technique for a smart vehicle to interact with the surrounding vehicles.
- The Bayesian approach is a common and efficient method to solve object-tracking problems.
- The particle filter is a type of non-parametric Bayesian filter which has the advantage of being able to represent complex beliefs.
- However,
- Many intelligent vehicles rely on LiDARs for obstacle detection as well as localization and mapping.
- The 3D LiDAR, e.g. Velodyne, can fully scan the real world at 10 Hz producing nearly 100,000 points per frame; however, directly processing a frame of point-cloud is always time-consuming.
- The 2D LiDAR, e.g. SICK or Hokuyo, can horizontally scan the surrounding and briefly represent the obstacles as an array of range values; however, it may falsely detect sloped road surfaces as obstacles, and it cannot properly detect low or overhung obstacles.
- The VScan is also an array of range values from a 2D compact transformation of point-cloud. Meanwhile, we developed a robust and real-time VScan algorithm to handle the sloped road, low objects and overhung obstacles. Therefore, the VScan is suitable for rapid further processing in a complex urban environment.

Contributions

- A new data structure called *Basic VScan Matrix* (BVSM) represents point-cloud around the ego-vehicle.
- A *Simultaneous Road Filtering and Obstacle Detection* (SRFOD) algorithm works on top of BVSM for robust VScan generation (mainly focuses on slope roads, low objects, and overhung obstacles).
- A *Sorted Array based Acceleration Method* (SAAM) enables real-time VScan generation.

Method Description

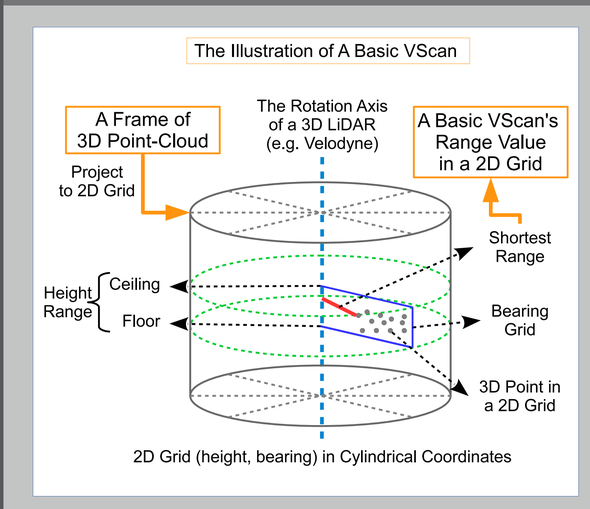

Figure 2: The basic VScan is normally extracted from a chosen set of points within a height range (Floor → Ceiling), and its range value in each bearing grid is calculated as the shortest range between the axis and the points.

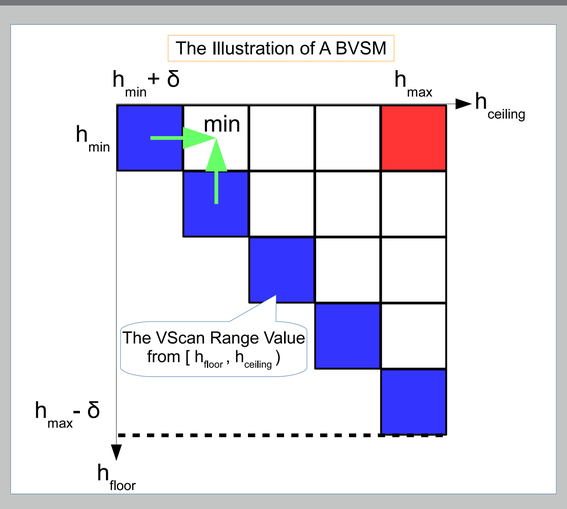

Figure 3: A BVSM collects all possible Basic VScans' range values in the same bearing grid. The blue diagonal cells correspond to the minimum height range δ, and the off-diagonal cells can be calculated via **min** operation (the green arrows).

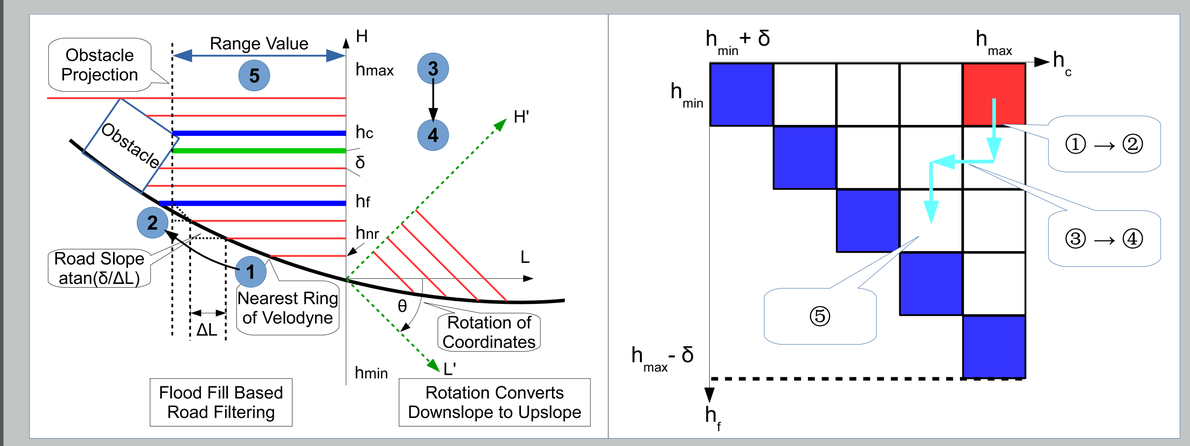

Figure 4: The SRFOD is to find a proper height range ($\mathbf{h_f} \rightarrow \mathbf{h_c}$) starting from the max height range ($\mathbf{h_{min}} \rightarrow \mathbf{h_{max}}$) for basic VScan calculation as well as obstacle detection in each bearing grid. [Left] ① → ② (road filtering stage): use the flood fill method with slope threshold to filter road surface. ③ → ④ (obstacle detection stage): lower ceiling to filter fake obstacles highly elevated above the ground. ⑤: after several alterations between these two stages, the proper height range is determined, and the range value is from the corresponding basic VScan. [Right] The SRFOD process can be represented as the movement (the cyan path) from the red corner cell toward the blue diagonal cells on the BVSM. The movement decision is presented in the following "Algorithm Description" section. Additionally, the SRFOD cannot work on downslope roads; therefore, we use a temporary rotation, which keeps geometry consistency, to convert a downslope road to an upslope road for obstacle detection, and then revert this rotation to determine the final range value.

Algorithm Description

The BVSM Based SRFOD

- Define $\mathbf{L_i(h_f, h_c)}$ as the range value in **i**th bearing of a basic VScan from the height interval $\mathbf{[h_f, h_c)}$
- Index $\mathbf{L_i(h_f, h_c)}$ as $\mathbf{L'_i(g_f, g_c)}$:
  $\{\mathbf{L'_i(g_f, g_c)} | \mathbf{g_f} \in \mathbb{N}, \mathbf{g_c} \in \mathbb{N}, \mathbf{g_{min}} \leq \mathbf{g_f} < \mathbf{g_c} \leq \mathbf{g_{max}}\}$
  where $\mathbf{g} = \lfloor (\mathbf{h} - \mathbf{h_{min}})/\delta \rfloor$
- Parameters:
  - ϕ: the maximum road slope.
  - $\mathbf{\Delta H}$: the passable height for ego-vehicle.

Algorithm:

1. Construct the BVSM of **i**th bearing grid.
2. Start from $\mathbf{g_f = g_{min}}$ and $\mathbf{g_c = g_{max}}$.
3. The rule for road or obstacle detection:

$$\mathbf{\Delta L} \cdot \tan(\phi) \begin{cases} \geq \delta \Rightarrow \textbf{Road} \\ < \delta \begin{cases} \mathbf{\Delta h} > \mathbf{\Delta H} \Rightarrow \textbf{Unknown} \\ \mathbf{\Delta h} \leq \mathbf{\Delta H} \Rightarrow \textbf{Obstacle} \end{cases} \end{cases} \quad (1)$$

   where
   $\mathbf{\Delta L} = \mathbf{L'_i(g_f + 1, g_c)} - \mathbf{L'_i(g_f, g_c)} \geq \mathbf{0}$, and
   $\mathbf{\Delta h} = \mathbf{h_c} - \mathbf{h_f}$
4. The decision of next movement on the BVSM:
   - Road: move to cell $\mathbf{(g_f + 1, g_c)}$ (down, ① → ②).
   - Unknown: move to cell $\mathbf{(g_f, g_c - 1)}$ (left, ③ → ④).
   - Obstacle: stop, and $\mathbf{L'_i(g_f, g_c)}$ is the final result (⑤).
5. Goto step 3 until $\mathbf{g_f + 1 = g_c}$.

Complexity Analysis:

- Parameters:
  - **P**: the number of 3D points.
  - **N**: bearing grid number (range array length).
  - **M**: height grid number ($\mathbf{(h_{max} - h_{min})/\delta}$).
- Computational Complexity: $\mathbf{O(P + NM^2)}$
  - BVSMs' diagonal cells: $\mathbf{O(P)}$
  - BVSMs' off-diagonal cells: $\mathbf{O(NM^2)}$
  - SRFOD on BVSMs: $\mathbf{O(NM)}$

Problem: If the high resolution of height grid (small δ) is required for low objects detection, the large **M** makes this algorithm time-consuming.

The SAAM

- The construction of BVSMs costs too much time and memory space; however, the BVSMs' off-diagonal cells can be directly derived from their sorted diagonal cells (see the property C in this paper's appendix).
- The SAAM will only use the BVSMs' sorted diagonal cells to realize SRFOD; therefore, it can significantly accelerate the SRFOD when the height grid number (**M**) is large.

Algorithm:

1. Calculate the diagonal cells ($\{\mathbf{L'_i(g_f, g_f + 1)}\}$) of the BVSM of **i**th bearing grid.
2. Sort diagonal cells in ascending order and for the cells have same $\mathbf{L'_i}$, sort them with their floor height index $\mathbf{g_f}$ in descending order. After sorting, we get two associated arrays: $\mathbf{L'_i[j]}$ and $\mathbf{g_f[j]}$.
3. Start from $\mathbf{j_0 = 0}$ and $\mathbf{j_1 = 1}$.
4. The rule for road or obstacle detection:

$$\mathbf{\Delta g} \begin{cases} < 0 \Rightarrow \textbf{Unknown} \\ = 1 \Rightarrow \textbf{(1)} \\ > 1 \begin{cases} \mathbf{\Delta h} > \mathbf{\Delta H} \Rightarrow \textbf{Unknown} \\ \mathbf{\Delta h} \leq \mathbf{\Delta H} \Rightarrow \textbf{(2)} \end{cases} \end{cases}$$

$$\mathbf{\Delta L} \cdot \tan(\phi) \begin{cases} < \delta \ \textbf{Obstacle} \\ \geq \delta \ \textbf{Obstacle*} \end{cases} \quad (2)$$

   $\mathbf{\Delta L} = \mathbf{L'_i[j_1]} - \mathbf{L'_i[j_0]}$, $\mathbf{\Delta h} = \mathbf{h_f[j_1]} - \mathbf{h_f[j_0]}$, and
   $\mathbf{\Delta g} = \mathbf{g_f[j_1]} - \mathbf{g_f[j_0]}$
5. The decision of next movement on the sorted array:
   - Road: $\mathbf{j_0 \leftarrow j_1}$ and $\mathbf{j_1 \leftarrow j_1 + 1}$.
   - Unknown: $\mathbf{j_1 \leftarrow j_1 + 1}$.
   - Obstacle: stop, and $\mathbf{L'_i[j_0]}$ is the final result.
   - Obstacle*: stop, and $\mathbf{L'_i[j_1]}$ is the final result.
6. Goto step 4 until $\mathbf{j_1}$ exceeds the sorted array range.

Computational Complexity: $\mathbf{O(P + NM \log(M))}$

- BVSMs' diagonal cells: $\mathbf{O(P)}$
- Sort diagonal cells of each BVSM: $\mathbf{O(NM \log(M))}$
- SRFOD on the sorted arrays: $\mathbf{O(NM)}$

- Speed test with $\mathbf{P \approx 100,000}$, $\mathbf{N = 2000}$, and $\mathbf{\delta = 0.05m \Rightarrow M = 100}$ (Intel Xeon ES-1660 3.70GHz):
  - The BVSM based SRFOD costs **113ms** > **100ms**, and Velodyne's working frequency is **10Hz**.
  - The SAAM costs **13ms**, and it is suitable for real-time applications.

Further Development: 3D VScan as "Stixels"

- Our SRFOD method can detect the obstacle's vertical height as "Stixels" containing min. and max. altitudes.

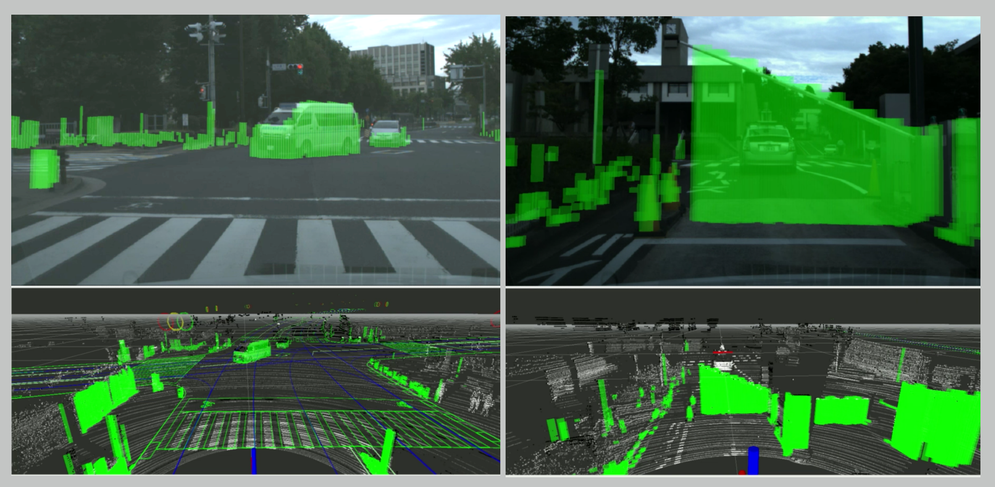

Figure 5: The visualization of "Stixels" (the green vertical bars) on images (first row) and in point-clouds (second row). Especially, the right column shows the detection of overhung barrier gate.

Experiment Results[1]

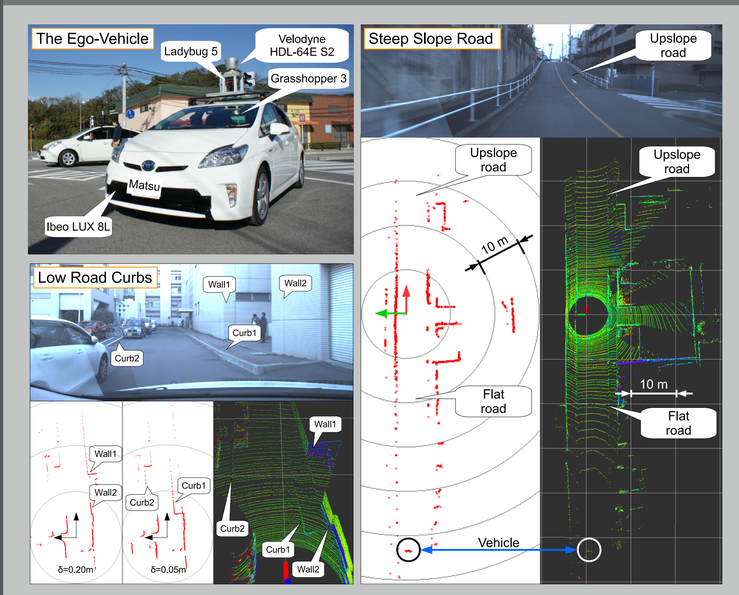

Figure 6: [Top-Left] Our experiment vehicle Matsu equipped with a Velodyne and cameras. [Bottom-Left] The detection of low road curbs with high resolution (small δ). Because the curb's height is less than **0.2m**, the road curb cannot be detected with δ = **0.2m**. [Right] The VScan result on a steep slope road (20m long and 3m high). Our algorithm successfully filtered the slope road surface, and additionally detected a Vehicle 50m behind the ego-vehicle on a flat road.

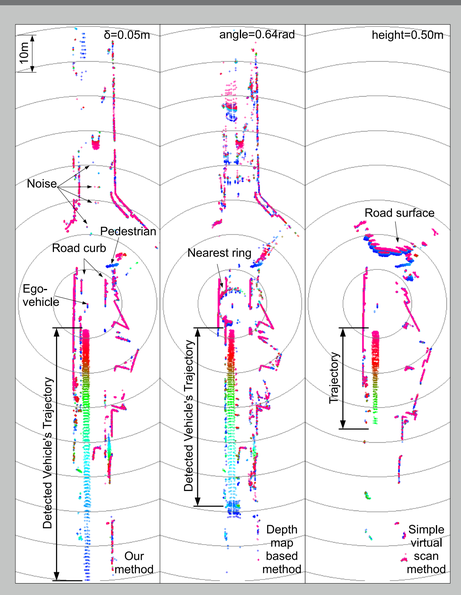

Figure 7: We compared our method (left) with other two VScan methods[2]: a basic implementation of F. Moosmann's depth map based method[3] (middle), and T. Foote's method[4] from ROS (right). The ego-vehicle is static, and the result accumulates 100 frames of VScan (blue→red) to test its stableness.

1. More video results can be found on Youtube `https://youtu.be/d6l31owNs2U`.
2. The video can be found on Youtube `https://youtu.be/0h0n9W6RIAQ`.
3. 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. 215220.
4. T. Foote, pointcloud to laserscan ROS node. [Online]. Available: `http://wiki.ros.org/pointcloudtolaserscan`.

Acknowledgments