Lightweight Semantic-aided Localization with Spinning LiDAR Sensor

Yuan Ren, Bingbing Liu, Ran Cheng, and Christopher Agia

***Abstract*—Autonomous driving demands robust and precise vehicle localization in complex environments with limited on-board computational resources. Incorporating reliable semantic information with localization algorithms can increase accuracy remarkably, however, the process of extracting semantic information from LiDAR point clouds and matching it to semantic maps is computationally intensive. Moreover, pure semantic localization cannot achieve the robustness requirements for safe self-driving as the necessary quantity of semantic landmarks cannot be guaranteed under extreme conditions. In this paper, we present a lightweight semantic-aided localization method that improves upon traditional techniques in two ways. First, we propose a highly efficient pipeline to extract three semantic classes from a LiDAR scan. Second, instead of semantic 3D point cloud registration, map matching is performed through 2D key point matching. We then integrate these two functions into a dynamic semantic aided localization framework. Our on-road experiments demonstrate that the proposed method achieves both the high accuracy of semantic localization and the robustness of non-semantic localization. With our algorithm consuming under 10% of CPU resources, we observe reduced positioning error, especially peak error, when comparing to non-semantic counterparts.**

***Index Terms*—Autonomous vehicle, vehicle localization, semantic localization, 3D-LiDAR**

I. INTRODUCTION

AUTONOMOUS driving technology has rapidly grown in the past decade. In order to improve the intelligence of autonomous vehicles, highly precise vehicle positioning systems are required. Traditional localization systems based on GNSS+IMU data cannot meet accuracy demands due to the insufficient number of visible satellites and multi-path signal reflections. And when using a low-cost IMU, substantial accuracy degradation can be expected during GNSS outage. The use of monocular or stereo camera systems are another solution for autonomous vehicle localization. Yet, while environmental features (e.g road painting markers) can be extracted and matched to prior feature maps, feature extraction remains susceptible to noise and poor lighting conditions. In contrast, the LiDAR sensor is insensitive to illumination changes and captures the reflective intensities and 3D geometry of the environment. Modern spinning LiDAR sensors used on autonomous vehicles can provide sparse point cloud outputs at above 10 Hz, at a range accuracy within a few centimeters. Thus, matching observations from 3D LiDAR scans with prior maps has become an effective and well-used method for vehicle localization [1], [2].

In theory, leveraging semantic information obtained through semantic segmentation can improve accuracy and enable intelligent behaviour for navigation [3]. There are however two practical drawbacks associated with applying semantic localization systems on mass-produced vehicles. First, the computing resources on mass-produced vehicles are limited. With the majority of the computing power reserved for perception and planning tasks, only very limited computing resources can be left for localization. Many reported methods cannot be used for online operation as scarce computational resources yield slower processing rates. Some methods propose the use of a GPU for semantic feature extraction (e.g. semantic segmentation) and complete CPU usage for matching; this is infeasible from a practical perspective. Second, the robustness of pure semantic localization is not comparable to non-semantic methods. Errors that occur in semantic segmentation will be propagated to map matching and ultimately affect the localization result. A lack of available semantic landmarks in extreme environments has the capacity to crash pure semantic algorithms. Moreover, map matching methods based on semantic 3D point cloud registration are unable to self-correct. Any errors from improper matching will be propagated and increase over time, eventually leading to a diverged localization result. While particle based matching methods can remedy this problem, they come with an exponential growth in computation.

In this paper, we introduce a lightweight semantic-aided localization system that operates directly on point clouds from a spinning LiDAR sensor, with all computation (i.e. semantic segmentation and map matching) accomplished with less than 10% of the computing resources of a desktop CPU. The proposed method combines the advantages of pure semantic and non-semantic methods by incorporating semantic landmarks - pole-like objects, vertical planes and road markers into a non-semantic system backbone. The algorithm can dynamically adjust estimation dependency over the various landmark types based on the quantity of landmarks, the spatial distribution of landmarks, and the quality of matching. With this unique strategy, the accuracy of pure semantic positioning and robustness of non-semantic positioning can be fully realized. Furthermore, our design takes into consideration the actual needs of mass-produced vehicles in terms of map size and

(Corresponding Author: Bingbing Liu)

Y. Ren is with Huawei Noah's Ark Lab, Toronto, Canada (email: yuan.ren3@huawei.com)

B. Liu is with Huawei Noah's Ark Lab, Toronto, Canada (email: liu.bingbing@huawei.com)

R. Cheng was with Huawei Noah's Ark Lab, Toronto, Canada (email: ran.cheng1@huawei.com)

C. Agia was with Huawei Noah's Ark Lab, Toronto, Canada (email: christopheragia@gmail.com)

LiDAR field of view (FOV). The semantic tags and the spatial distribution of the point cloud are compressed into a low-memory single channel 8-bit image. The proposed algorithm is tested on a LiDAR embedded in the vehicle's bumper with a FOV of slightly greater than 180 degrees.

II. Related work

Current LiDAR-based localization algorithms can be divided into two categories. The first category of methods do not extract any semantic information from the point cloud. They only build the map and determine the location of the vehicle based on the spatial distribution characteristics of the point cloud. The works of [2], [4] show that this type of methods can be successfully used in both indoor and urban environments. The second category of methods extract landmarks with semantic information from the point cloud. Due to the ubiquity, long-term stability and geometric features that can be easily extracted by 3D LiDAR, pole-like objects are the most commonly used landmark for vehicle localization in urban environments. Wu et al. [5] use street light poles as the landmarks in their localization algorithm. Sefati et al. [6] practice the pole-based localization in urban area with high rate of dynamic objects. Schaefer [7] investigates the feasibility of long-term urban localization by using different types of pole-like objects, such as traffic signs, street lamps, and tree trunks. Certainly, pole-like objects are not the only landmark suitable for vehicle localization. Lane markers are also very ideal 2D landmarks for vehicle localization. The works of Tao [8] and Schreiber [9] demonstrated that the lane markers can help to achieve high localization accuracy. Im et al. [10] use both lane markers and building outer wall together in localization. Similar to the lane marker, curb is also a usable 2D marker for localization in urban environment [11], [12]. While, its main disadvantage is that it is usually blocked by vehicles parked on the side of the road. In addition, Im et al. [13] explore the urban localization based on the corner features of buildings.

In recent years, deep learning-based semantic segmentation methods have been applied in SLAM, mapping and localization. There is a rich body of scientific works focused on camera-based semantic localization. Semantic SLAM, mapping and localization techniques have been proposed for use with monocular cameras [14], [15], [16], stereo cameras [17], [18], [19], [20], and RGB-D sensors [21], [22], [23], [24], [25]. Most of these methods are developed for indoor environments, and some display intelligent behaviours such as filtering dynamic objects prior to map matching. Some studies attempt to improve the performance of semantic SLAM, mapping, or localization with a dual 3D LiDAR sensor and camera setup [26], [27], [28], [29]. In these works, semantic information is mainly extracted from the camera while the 3D LiDAR sensor play a secondary role in providing geometrical information. Learning-based semantic segmentation is also used in pure LiDAR solutions [30], [31], [32]. Sun et al. [33] propose a learned probabilistic model to speed up the LiDAR-based localization using particles. Chen et al. [34] develop the OverlapNet, which uses semantic information of LiDAR scans to find loop closures. Chen et al. [35] propose a learned observation model for LiDAR-based Monte Carlo localization with a particle filter. However, the advantage of acquiring more detailed classification results with learning-based methods comes at a computational cost, only these semantic segmentation algorithms with real-time performance [36], [37] are suitable to be used in localization.

III. Our approach

Our method is based on a particle filter. When the system runs for the first time, these particles are initialized with a low-cost GPS receiver. In daily operation, the system will record the last position and heading results, which are obtained by the LiDAR based localization algorithm, before closing. When the system restarts, that record will be used to initialize particles. The states of particles are predicted by using IMU+wheel odometry, and are updated by matching the LiDAR observation with a prior 2D semantic map. The update process entails three steps: semantic landmark extraction, 2D map compression, and 2D map matching. The semantic extraction pipeline takes a raw LiDAR point cloud and outputs a point cloud segmented by semantic class categories. With each point in the point cloud assigned a semantic tag, the point cloud is efficiently converted into a 2D feature image. Key points are then located on the 2D feature image and matched with the prior map to update the weights of particles. The prior 2D semantic map is managed with the method introduced in [2]. The entire map is divided into square map tiles of 100 meters by 100 meters. The vehicle dynamically loads 9 map tiles according to its current position, and stitches them into a local map covering the area of 300 meters by 300 meters around the vehicle.

A. Semantic extraction from LiDAR point cloud

In this step, we explain the process of extracting semantic landmarks from a raw LiDAR point cloud. Semantic landmarks include pole-like objects, vertical planes, and road marks; all remaining points that not fall into one of these three categories are tagged with a non-semantic label. Our entire system is intended to run in real-time on an on-board computer without a GPU, motivating a design low in computational complexity.

Currently, most of the autonomous vehicles are equipped with a spinning LiDAR sensor that emits $8 \sim 128$ lasers per measurement, which is called a sector, and there are approximately 2000 sectors in each revolution. The semantic extraction can be accelerated by utilizing the geometric descriptors of the spinning LiDAR. The semantic extraction procedure is shown in Fig. 1. First, the curvature of each point is computed and the point cloud is segmented into ground/ceiling and non-horizontal objects. At the same time, vertical seeds with a high likelihood of lying on a vertical object are extracted. Then, road marks are extracted with an intensity filter, and non-horizontal points are clustered with a region growing algorithm. The region growth starts from the vertical seeds, hence, most of the vertical objects such as the poles and walls are clustered. Finally, these clusters are classified according to the size of their bounding box and average curvature. There are three categories: pole-like objects, vertical planes and non-semantic clusters. Pole-like objects, vertical planes and road marks are used as the semantic landmarks. The non-semantic clusters and the unclustered points are merged and used as non-semantic landmarks.

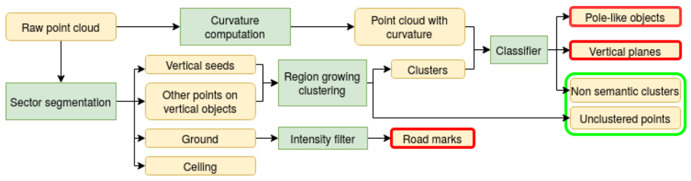


Fig. 1: Flow chart of the proposed algorithm. Red box: semantic features for localization; Green box: non-semantic features for localization.

1) LiDAR point cloud pre-processing: The z axis is defined as the spinning axis of the LiDAR. The elevation and azimuth of each laser point can be written as:

$$\lambda = \arctan \frac{z}{\sqrt{x^2+y^2}} \tag{1}$$

$$\psi = \arctan \frac{y}{x} \tag{2}$$

Using the known mounting angles of the laser emitters, the relationship between the elevation and the laser emitter number (ring number) can be established, and the 3D point cloud can be projected into a range image. Each row of the range image corresponds to a ring of the LiDAR scan, and each column corresponds to a sector. Several elements of the range image may be empty as not all emitted lasers receive echoes. We then need to compute the horizontal distance and curvature of each non-empty range image element to be used for region growth clustering and classification.

2) Horizontal scan - curvature computation: The curvature of a scan line encodes characteristics of an object's surface. The smoothly constructed surfaces of artificial objects are normally associated with smaller curvatures, while the scan lines on natural objects usually have a larger curvature. Each row of the range image corresponds to a scan line. Assuming that there are $2k+1$ continuous elements in a row of the range image, the coordinates of the point is denoted by $[x_j, y_j, z_j]^T$. i is the middle point. The curvature of point i can be approximated by [38]:

$$c = \frac{1}{2k\|\mathbf{r}_i\|} \| \sum_{i-k \le j \le i+k, j \ne i} (\mathbf{r}_i - \mathbf{r}_j) \| \tag{3}$$

where $\mathbf{r}_i = \sqrt{x_i^2+y_i^2+z_i^2}$ and $\mathbf{r}_j = \sqrt{x_j^2+y_j^2+z_j^2}$.

In this paper, k is set to 5, and the result is stored in a matrix with the same dimensions of the range image.

3) Vertical scan - sector segmentation: As depicted in Fig. 2, laser beams in a sector are numbered in ascending order of elevation. The inclination of the segment between two adjacent laser points can be written as:

$$\vartheta = \arctan \left(\frac{z_{i+1} - z_i}{\sqrt{x_{i+1}^2 + y_{i+1}^2} - \sqrt{x_i^2+y_i^2}} \right) \tag{4}$$

where i and $i+1$ are the ring numbers of the adjacent laser points. The laser points can be classified into four categories by using the inclination ϑ and their coordinates.

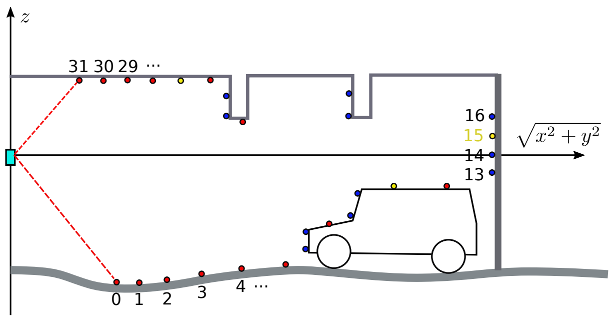


Fig. 2: Laser beams in a sector. Red points: laser points on the ground or ceiling; Blue points: laser points on vertical objects; Yellow points: missed points

The four classes are the ground points $\mathcal{PC}_\mathcal{G}$, ceiling points $\mathcal{PC}_\mathcal{C}$, vertical seeds $\mathcal{PC}_\mathcal{V}$, and the point cloud without ground and ceiling $\mathcal{PC}_\mathcal{D}$.

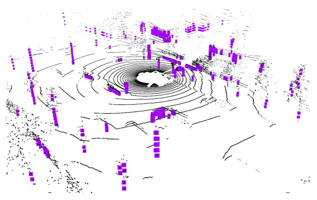
1) $\mathcal{PC}_\mathcal{G}$: $|\vartheta| < \epsilon$ and $\sqrt{x_{i+1}^2+y_{i+1}^2} > \sqrt{x_i^2+y_i^2}$;
2) $\mathcal{PC}_\mathcal{C}$: $|\vartheta| < \epsilon$ and $\sqrt{x_{i+1}^2+y_{i+1}^2} < \sqrt{x_i^2+y_i^2}$;
3) $\mathcal{PC}_\mathcal{V}$: $|\vartheta - \pi/2| < \epsilon$;
4) $\mathcal{PC}_\mathcal{D}$: $\mathcal{PC}_\mathcal{D} = \mathcal{PC} \setminus (\mathcal{PC}_\mathcal{G} \cup \mathcal{PC}_\mathcal{C})$.

Here, $\mathcal{PC}$ denotes the raw point cloud and ϵ is an inclination threshold. Based on the maximum ground inclination in an ordinary environment, it is empirically set to $15°$. Due to long distance and low reflectivity, some points may be missed. If adjacent points on both sides are lost, we classify the center point as $\mathcal{PC}_\mathcal{D}$.

Ceiling points $\mathcal{PC}_\mathcal{C}$ are not used in the following steps. High intensity points in $\mathcal{PC}_\mathcal{G}$ are classified as road marks. $\mathcal{PC}_\mathcal{D}$ includes all non-horizontal objects and will be compressed into 2D images to be used as the non-semantic features. At the same time, the semantic features for localization will also be extracted from a subset of $\mathcal{PC}_\mathcal{D}$ with region growing algorithm. This subset, $\mathcal{PC}_\mathcal{V}$, contains all vertical seeds of potential pole-like objects and vertical planes.

4) Region growing clustering: In Fig. 3a, the black points are the raw LiDAR point cloud, and the purple points are vertical seeds obtained by vertical scan. It can be seen that the majority of vertical planes and pole-like objects are covered by the vertical seeds. The vertical seeds are subject to noise, and few are not on any vertical object; this occurs when two unrelated points happen to lie on the same vertical line.

Region growing clustering started from the vertical seeds is used to get more clear and complete vertical objects. On the range image, all the adjacent points of a vertical seed are checked. If an adjacent point belongs to $\mathcal{PC}_\mathcal{D}$ and the difference of the horizontal distance is less than 20 cm, it is merged into the current cluster. A finalized cluster is obtained by repeating this procedure until no new adjacent point can be merged. After applying this process to all vertical seeds, we acquire a list of clusters, which is denoted by $\{\mathbb{C}|\mathcal{C}_i,\ i = 1,\dots,N\}$.


(a) Vertical seeds

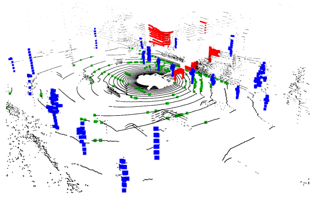

(b) Region growing clustering and classification result

Fig. 3: Vertical seeds and region growing clustering and classification result (Black: original point cloud; Purple: vertical seeds; Red: vertical plane; Blue: pole-like object; Green: road mark)

Fig. 3b shows the results of region growing clustering. The red points show the vertical planes, and the blue points are the pole-like objects. It can be found that the result of region growing clustering can better reflect the true shape of vertical objects. For instance, the vertical seeds of a tree only cover the trunk, but after region growing clustering the complete shape is inferred.

5) Fast filtering and classification: A series of clusters were obtained by region growing clustering from vertical seeds that have a high probability of being a vertical object of interest. Most of these clusters are pole-like objects and vertical planes. While, some of them are not a part of any vertical object. Usually, they are background objects with irregular shape. A fast filtering and classification algorithm is developed to remove these redundant clusters and assign semantic tags to the remaining clusters. The algorithm is shown below.

```
Algorithm 1. Fast filtering and classification
Input: A list of clusters. $\{\mathbb{C}|\mathcal{C}_i,\ i=1,\dots,N\}$.
Output: A list of pole-like objects $\mathbb{P}$; A list of vertical planes $\mathbb{V}$.
        A list of non-semantic objects $\mathbb{O}$
1: for $\mathcal{C}_i$ in $\mathbb{C}$ do
2:   if $\mathcal{C}_i$ has more than $N_{\min}$ points then
3:     create the bounding box of $\mathcal{C}_i$, whose dimension is $l \times w \times h$;
4:     if $w < w_{pole}$ and $l < w_{pole}$ and $h > h_{pole}$ then
5:       pushback $\mathcal{C}_i$ into $\mathbb{P}$;
6:     else if $\sqrt{w^2+l^2} > m_{plane}$ or $h > h_{plane}$ then
7:       compute the average curvature of $\mathcal{C}_i$, and denote it by $\rho_A$;
8:       if $\rho_A < \rho_p$ then
9:         pushback $\mathcal{C}_i$ into $\mathbb{V}$;
10:      else if
11:        pushback $\mathcal{C}_i$ into $\mathbb{O}$;
12:      end if
13:    end if
14:  end if
15: end for
16: return $\mathbb{P}$, $\mathbb{V}$, $\mathbb{O}$.
```

Six thresholds are used in this algorithm. In step 2, $N_{\min}$ is used to remove undersized clusters. In step 4, w_{pole} and h_{pole} are the maximum width and minimal height of a pole. m_{plane} and h_{plane} in step 6 are used to extract the vertical plane. ρ_p in step 8 gives the maximum mean curvature of the man-made surface. The classification results are shown in Fig. 3b.

B. 2D compression

We acquire a semantic 3D point cloud through the process outlined in Section III-A. Since the demand for vehicle positioning in autonomous driving is mainly 2D (x, y and heading angle), converting the 3D semantic point cloud into a 2D feature image serves to both reduce map size and increase the speed of map matching, meeting the functional and computational localization requirements.

To convert a point cloud into a 2D feature image, we begin by voxelizing it in the $x-y$ plane at a 10 cm resolution, yielding equal sized columns along with height and width of the map. The vertical extent of each column is from 0 m to 4.2 m, and is divided into 6 equal parts. As shown in Fig. 4, each 3D voxel occupies a 10 cm $\times$ 10 cm $\times$ 70 cm cuboid. If a cuboid is occupied by laser point, it is denoted by 1, otherwise, it is 0. With this method, the vertical distribution of laser points within a column can be represented by 6 binary bits. The other two bits are used to store the semantic tag. Note that, the space above the road mark is usually not occupied by any point, so the 6 binary bits are used to save the reflection intensity of the road mark.

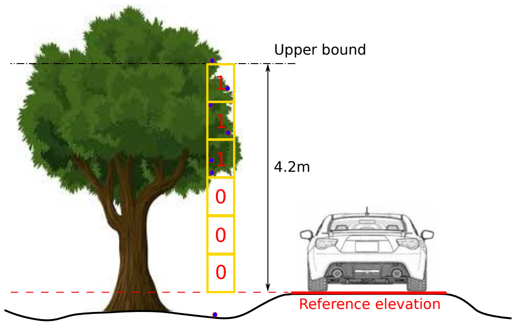


Fig. 4: Converting from 3D point cloud to 2D feature image

Table I shows the encoding format, and Fig. 5 shows a typical 2D feature image which has been converted from the single-channel 8-bit form to a color image for visual clarity. The tags of pole-like objects are denoted by light green, vertical planes by dark green, road markings by blue, and non-semantic vertical distribution of the point cloud by red. The intensity of the red channel corresponds the height of the object that occupies the cell.

TABLE I: Encoding format

bit	b7	b6	b5	b4	b3	b2	b1	b0
Pole-like obj	1	0	0/1	0/1	0/1	0/1	0/1	0/1
Vertical planes	0	1	0/1	0/1	0/1	0/1	0/1	0/1
Road marks	1	1	0/1	0/1	0/1	0/1	0/1	0/1
Non-semantic	0	0	0/1	0/1	0/1	0/1	0/1	0/1

This 2D conversion method is used in both mapping and localization. In mapping, the city-scale 3D semantic point cloud will be compressed into a 2D feature map. In localization, a LiDAR frame will be converted into a 2D feature image in real-time, and then matched with the map.

C. Map matching and weights update

Localization can be accomplished with a particle filter, with the wheel or LiDAR odometry used for prediction, and the update done by matching the observation with the map in

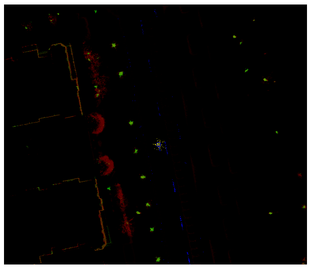


Fig. 5: 2D feature image

real-time. The map loading method used in this paper is well described in [2]. Here, we primarily focus on the map matching method.

1) Key points selection: As expected, the 2D feature image converted from the segmented and filtered LiDAR point cloud is typically very sparse. However, we observe on the order of a thousand non-zero pixels per frame, and matching all non-zero pixels with the prior map does meet real-time requirements. Hence, we select a set of key points from the non-zero feature image pixels. For the pole-like objects and non-semantic objects, we prefer to select pixels with larger vertical feature codes as key points, which denote the 2D positions of tall poles and non-semantic objects. For road marks, these pixels with larger reflectivity are selected first. The pixels corresponding to vertical plane are randomly selected. With this method, the building outline in the bird's eye view can be effectively expressed by key points which are evenly distributed on the wall-ground intersection line. The details of the key points selection method are shown in algorithm 2.

Algorithm 2. Key points selection

```
Input: 2D feature image; Key points number: $N_p$, $N_v$, $N_r$, $N$.
       $\mathbb{P}$ is the set of all pixels in the 2D feature image.
Output: Pole-like KP: $\mathbb{K}_p$; Vertical plane KP: $\mathbb{K}_v$;
        Road mark KP: $\mathbb{K}_r$; Non-semantic KP: $\mathbb{K}_n$ .
1: for $\mathcal{P}_i$ in $\mathbb{P}$ do
2:   if $\mathcal{P}_i \neq 0$ then
3:     if $(\mathcal{P}_i[b7] == 1)$ and $(\mathcal{P}_i[b6] == 0)$ then
4:       push back $\mathcal{P}_i$ into $\mathbb{P}_p$;
5:     else if $(\mathcal{P}_i[b7] == 0)$ and $(\mathcal{P}_i[b6] == 1)$ then
6:       push back $\mathcal{P}_i$ into $\mathbb{P}_v$;
7:     else if $(\mathcal{P}_i[b7] == 1)$ and $(\mathcal{P}_i[b6] == 1)$ then
8:       push back $\mathcal{P}_i$ into $\mathbb{P}_r$;
9:     else
10:      push back $\mathcal{P}_i$ into $\mathbb{P}_n$;
11:    end if
12:  end if
13: end for
14: sort $\mathbb{P}_p$ in descending order according to $[b5 \sim b0]$ bits;
15: if the size of $\mathbb{P}_p > N_p$ then
16:   push back the first $N_p$ elements of $\mathbb{P}_p$ into $\mathbb{K}_p$;
17: else
18:   $\mathbb{K}_p = \mathbb{P}_p$;
19: end if
20: random shuffle $\mathbb{P}_v$;
21: if the size of $\mathbb{P}_v > N_v$ then
22:   push back the first $N_v$ elements of $\mathbb{P}_v$ into $\mathbb{K}_v$;
23: else
24:   $\mathbb{K}_v = \mathbb{P}_v$;
25: end if
26: sort $\mathbb{P}_r$ in descending order according to $[b5 \sim b0]$ bits;
27: if the size of $\mathbb{P}_r > N_r$ then
28:   push back the first $N_r$ elements of $\mathbb{P}_r$ into $\mathbb{K}_r$;
29: else
30:   $\mathbb{K}_r = \mathbb{P}_r$;
31: end if
32: sort $\mathbb{P}_n$ in descending order according to $[b5 \sim b0]$ bits;
33: push back the first $N - N_p - N_v - N_r$ elements of $\mathbb{P}_n$ into $\mathbb{K}_n$;
34: return $\mathbb{K}_p$, $\mathbb{K}_v$, $\mathbb{K}_r$ and $\mathbb{K}_n$.
```

In algorithm 2, N_p, N_v and N_r are the limits on the quantity of pole-like, vertical plane and road mark key points, and N is the total number of key points. In our experiments, we set the values of these parameters to 50, 50, 20 and 300, respectively. Since the distribution of semantic key points changes with the environment, we always guarantee that the total number of key points, N, will be selected for map matching. In this case where no semantic-key points are available, our algorithm dynamically degenerates into a non-semantic method.

2) Matching of 2D feature images: Each particle gives a potential 2D pose of the vehicle. With this 2D pose and the installation matrix of LiDAR, the selected key points can be transformed into the map frame. A similarity metric can then be computed between the key point and the corresponding pixel on the map. The similarity metric is composed of two parts. The first being the consistency of the semantic tag, and the second is the similarity of vertical distribution (vertical objects) or reflection intensity values (road markings). Equation 5 shows how a particle weight is computed with key points of pole-like objects, vertical planes or non-semantic objects.

$$w_x = \frac{1}{N_x} \sum_{i=1}^{N_x} \frac{\alpha_i \cdot \mu_i}{6} \tag{5}$$

N_x is the number of key points for class x, α_i and μ_i are the semantic tag similarity and vertical distribution similarity of the i-th key point. The value of α_i can be obtained from table II, and μ_i is the dot product of bits b0-b5 between the key point and map.

TABLE II: Similarity of semantic tags

LiDAR \ Map	Pole	Plane	Road mark	Non-sem
Pole	1	0.64	0	0.21
Plane	0.72	1	0	0.24
Road mark	0	0	1	0
Non-sem	0.29	0.22	0	1

For example, a key point with value "10001111" corresponds to a map pixel whose value of "01001011". It means that the semantic tag of the key point is pole-like object, while the semantic tag of the corresponding map pixel is vertical plane. From the table, we obtain α_i equals 0.64. From b0 to b5, there are four bits that have the same value, so μ_i equals 4. It is clear that $\mu_i \in [0, 6]$, and $w_x \in [0, 1]$. w_x equal to 1 indicates a complete match of semantic similarity and vertical distribution between all key points and the map.

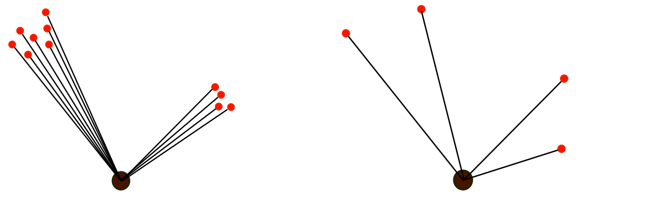
Given the key points of road marks, the particle weight can be updated by:

$$w_r = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{\alpha_i \cdot \xi_i \cdot \gamma_i}{63 \times 63} \tag{6}$$

where ξ_i is the intensity of the i-th key point (converting the binary numbers of b0-b5 to decimal). Similarly, γ_i is the intensity of the pixel in the map, and α_i is obtained from Table II. Note that, the semantic tag mismatch of vertical objects, for example matching a pole to a vertical plane, only attenuates similarity. The semantic tag mismatch of the road marks causes α_i to return 0. In simple terms, there is no similarity between road marks and vertical objects.

These values in Table II are obtained by counting all the pixels of the map. In the map, each pixel is observed many times and the semantic tag is determined by voting. For example, a pixel is observed N times, t_p votes are pole-like objects, t_v votes are vertical plane and t_n votes are non-semantic objects. The final semantic tag is pole because $t_p > t_v > t_n$. The probability that this pixel is a vertical plane is t_v/t_p. We extract all the pixels with the tag of pole-like object from the map, and computing the mean value of their t_v/t_p. The motivation for this matching strategy is robustness, as it accounts for potential error in the clustering and classification stages by enabling landmarks with different semantic tags to contribute updates on particle weights given a similar vertical structure.

3) Fusion of localization results: So far, four sets of weights have been computed per particle. In an ideal environment, which is rich in semantic landmarks and has minimal occlusion from dynamic objects, each weight can be used to update the particle filter and locate the vehicle independently. However, the accuracy of independent localization results depend heavily on (1) the number of available key points, (2) the quality of matching, (3) spatial distribution of key points, and (4) environmental characteristics. As illustrated in Fig. 6a, a large number of key points are clustered together, and Fig. 6b, few key points widely in the space - which scenario is more beneficial to positioning accuracy?



(a) A large number of key points are clustered together (b) Few widely separated key points

Fig. 6: Key points distribution

Augmented dilution of precision [39] is used to quantify the effect of the first three factors.

$$ADOP = \sqrt{trace\left((\mathbf{A}^T\mathbf{w}\mathbf{A})^{-1}\right)} \tag{7}$$

where $\mathbf{A}$ is a $n \times 2$ matrix, and n is the number of key points in the current LiDAR frame. Each row of $\mathbf{A}$ contains the $[x, y]$ coordinates of the i-th key point in the LiDAR frame. $\mathbf{w}$ is a $n \times n$ diagonal weighting matrix. The diagonal elements are the matching qualities of key points. For pole-like objects, vertical planes and non-semantic objects.

$$\mathbf{w} = \frac{1}{6}\begin{bmatrix} \alpha_1 \cdot \mu_1 & & 0 \\ & \ddots & \\ 0 & & \alpha_{N_x} \cdot \mu_{N_x} \end{bmatrix}$$

For road marks

$$\mathbf{w} = \frac{1}{63 \times 63}\begin{bmatrix} \alpha_1 \cdot \xi_1 \cdot \gamma_1 & & 0 \\ & \ddots & \\ 0 & & \alpha_{N_r} \cdot \xi_{N_r} \cdot \gamma_{N_r} \end{bmatrix}$$

The value of $ADOP$ lies in $(0, +\infty)$. A small $ADOP$ value (near zero) means that the map matching provides high-confidence positioning information and vice versa. Note that, one $ADOP$ per class is needed for a set of particles. Since each particle has diagonal weight matrices $\mathbf{w}_x$ for $x \in \{p, v, r, n\}$, the $\mathbf{w}_x$ with the greatest 2-norm is defined as $\mathbf{w}_x^*$ and substituted into Eq. 7.

While $ADOP$ describes the distribution of key points in the current LiDAR frame, it does not reflect the underlying geometry of the environment. For instance, on a highway, the result of pole-like object matching should have a larger weight than that of the road mark matching. Although the road marks might produce a small $ADOP$ value, they are unable to provide sufficient longitudinal constraint as they are parallel to the longitudinal direction of the vehicle. Therefore, we leverage the information entropy to measure the environmental influence on map matching. The information entropy is computed with:

$$H = 1 + \frac{1}{\log_{10} N}\sum_{i=1}^{N} w_i \cdot \log_{10} w_i \tag{8}$$

where N is the number of particles, and w_i is the weight of the i-th particle. The value of H is between 0 and 1. When H is close to 0, these weights do not carry much information, and when H is close to 1, these weights are information rich. Considering the $ADOP$ and information entropy together, the merging weights are computed by:

$$G = \kappa \cdot \exp(-H \cdot ADOP \cdot \rho) \tag{9}$$

where κ and ρ are correction parameters. The value of G is also contained between 0 and 1. The merging weights for poles, vertical planes, road marks and non-semantic landmarks are denoted by G_p, G_v, G_r and G_n, respectively. The merged weights are thus defined as:

$$w_i = \frac{\sum_{x=p,v,r,n} G_x(\kappa_x, \rho) \cdot w_i^x}{\sum_{x=p,v,r,n} G_x(\kappa_x, \rho)} \tag{10}$$

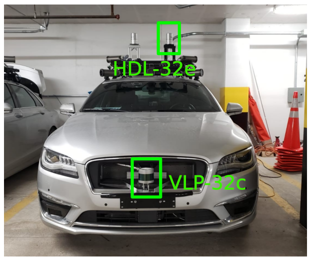
The particles will be updated with the merged weights.

IV. Experiment Results and Discussions

A. Hardware Platform

In accordance to vehicle appearance standards, most car manufacturers cannot accept a protruding LiDAR installed on the top of the car. They are more inclined to embed a LiDAR in the front bumper or behind the windshield. This installation method significantly limits the LiDAR's field of view. Furthermore, the bumper LiDAR is much lower than the top LiDAR, so its scans are more likely to be occluded by other vehicles. All of these factors may adversely affect the accuracy and robustness of the localization algorithm.

Our platform has a Velodyne HDL-32e LiDAR installed horizontally on top of the vehicle and a Velodyne VLP-32c LiDAR embedded in the bumper horizontally (Fig. 7a). The HDL-32e has 360-degree field of view and is used for mapping. The point cloud of the VLP-32c is shown in Fig. 7b. Its field of view is slightly greater than 180 degrees, and it is used for localization. The platform also equips dual antenna GNSS+INS system with RTK receiver which provides the ground truth position and orientation. The angular velocity of z-axis from INS is also used in wheel+IMU odometry. By integrating the z-axis angular velocity and rear wheel speed, the relative position in the initial vehicle body-fixed frame can be obtained. In addition, the computational platform used in the test is a computer with 8 logical cores Intel Xeon E3-1505MV5 2.8 GHz CPU.



(a) Installation of LiDARs

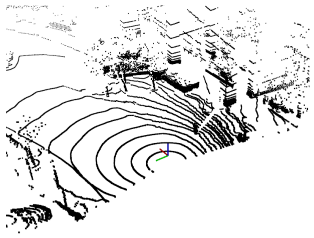

(b) Field of view

Fig. 7: Vehicle platform

B. Software Architecture

The software was developed based on Robot Operating System (ROS). It includes four ROS nodes, which are wheel+IMU odometry node, map management node, semantic segmentation node and particle filter node. The wheel+IMU odometry node and map management node are very lightweight. Semantic segmentation node receives raw LiDAR point cloud, and outputs 2D key points with semantic tags. The "untwist" operation of point cloud is done in this node [38]. Particle filter node is mainly responsible for map matching (update), and most of the auxiliary tasks such as status recording and zero velocity update are also completed by this node.

C. Test routes

The proposed localization algorithm has been tested on two test routes. The key parameters of these routes are shown in Table III. Each route was driven twice, one for mapping, the other for localization. Test route I contains highway and local roads. It also briefly crossed the office zone. The environment of test route II is more diverse. The vehicle was mainly driven on very busy local roads, and it also went into industrial zone, office zone and business plaza.

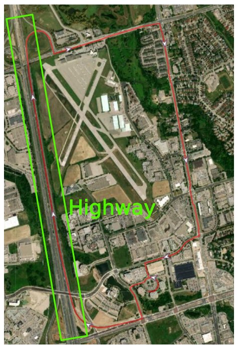
The maps of route I and II are shown in Fig. 8.

Fig. 9 shows the variation in landmark frequencies of different semantic tags along the test route I. We detect pole-like objects in almost every LiDAR frame, with slightly fewer on the highway. Observations of vertical plane features are far less frequent. In some sections of the highway no vertical planes are detected. Road marks and non-semantic features can be found in almost every frame. As expected, the observations of non-semantic features are more dense. Due to limited space, the landmark frequencies of test route II is not shown here. But for route II, we know that the landmark frequencies of the industrial zone are very similar to that of the local roads, and in office zone and business plaza more vertical plane features can be observed. We can concluded that pole-like objects, road marks and non-semantic objects are all reliable positioning features, and vertical planes only provide assistance on some road segments.

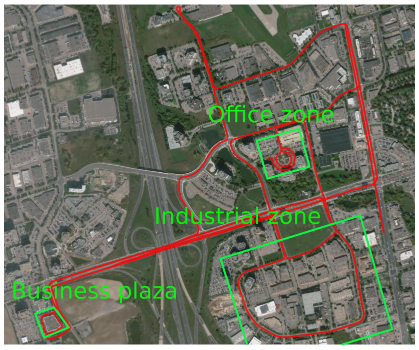
TABLE III: Key parameters of test routes

	Test route I	Test route II
Distance (km)	6.47	15.8
Duration (sec)	996	2331
Average speed (km/h)	23.4	24.4
Num of LiDAR frames	9945	23313
Loaded map size (MB)	3.7	10.2

Note that, Loaded map size denotes the total size of these tiles which are loaded dynamically during the test. It is not the size of the map database.



(a) Test route I



(b) Test route II

Fig. 8: Maps of test routes

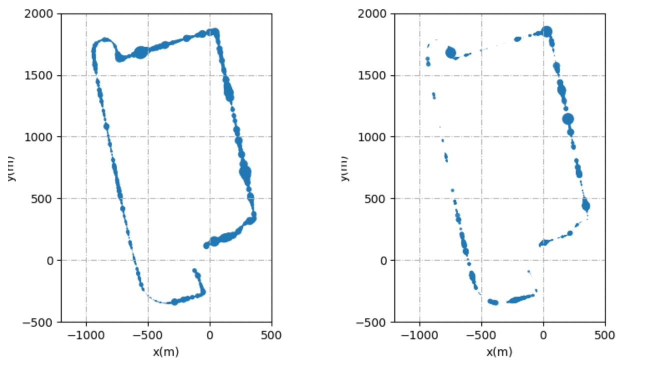
D. Road Test Results

Non-semantic localization is used as the baseline method, in which 500 particles and 300 key points are used. These key points are the first 300 pixels with the largest vertical distribution encoding, regardless of their semantic tag. The baseline method is very similar to the method introduced in Ref. [2]. In the tests of the proposed method, we always used 500 particles and 300 key points, in which up to 120 can be semantic key points.

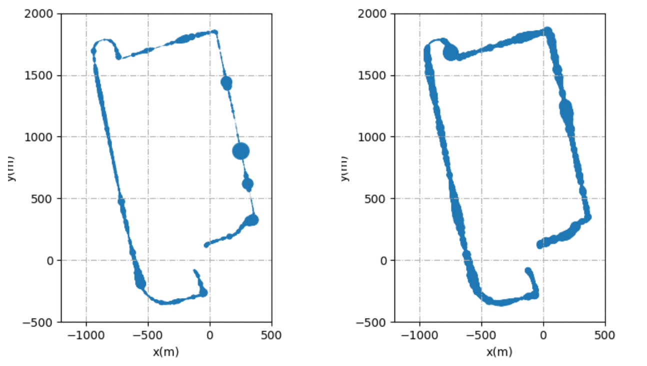
The proposed semantic extraction algorithm has six thresholds. In road testing, two sets of thresholds are used that represent different pole-like object selection strategies (see Table IV). In "Sem_1", w_{pole} equals 0.5. The standard of pole-like object selection is very strict, hence, the number of pole-like objects in each frame will be small. The standard of "Sem_2" is not so strict. Some imperfect poles, such as trees with a small crown, are used as poles in localization.

We also test three different sets of merge weights which are shown in Table V.

The semantic extraction thresholds in Table IV and the merge weights in Table V can create six parameter combinations. The localization algorithm runs with these six



(a) Number of poles (b) Number of vertical planes



(c) Number of road marks (d) Number of non-semantic landmarks

Fig. 9: Landmark quantities along the route

TABLE IV: Thresholds in semantic extraction

	$N_{\min}$	w_{pole}	h_{pole}	m_{plane}	h_{plane}	ρ_p
Sem_1	10	0.5	1.0	1.0	2.0	0.03
Sem_2	10	1.5	1.0	1.0	2.0	0.03

TABLE V: Merge weights

	κ_p	κ_v	κ_r	κ_n	ρ
Weight_1	1	1	1	1	2
Weight_2	1	2	1	1	2
Weight_3	2	1	1	1	2

parameter combinations by using the data sets of test route I and II. The mean and maximum errors (between LiDAR-based localization and RTK GPS) are listed in Table VI and Table VII, respectively. For both test routes, the combination "Sem_1+Weight_3" achieves the best accuracy. It means that the strict standard of semantic extraction ("Sem_1") and the higher merge weight of the pole-like objects ("Weight_3") help to achieve a better localization accuracy. It also can be found that comparing with the baseline method (non-semantic localization), the average error of the proposed method has not decreased much. While, the maximum error is almost reduced to 50% of the original value. In fact, the particle filter is very robust. Even in the case of continuous failed map matching, it will hardly diverge completely. However, the algorithm will not completely diverge does not mean that the localization results are always usable. A large positioning error will cause the failure of other system. Hence, reducing the maximum error of localization is equivalent to enhancing the robustness of the system. In other words, compared with increasing localization accuracy, the proposed semantic-aided localization method plays a greater role in increasing robustness. Fig. 10 show the longitudinal, lateral and heading error of the non-semantic localization and semantic-aided localization (test ID 4 in Table VI). It is clear to see the reduction of the maximum error. In Table VII, the positioning errors of the office zone/business plaza and local road/industrial zone are listed separately. For both non-semantic and semantic-aided methods, the positioning error in office zone/business plaza is much smaller than that of the local road/industrial zone, which is due to the lower speed in the office and business zone (the "untwist" error is lower) and the well structured environment.

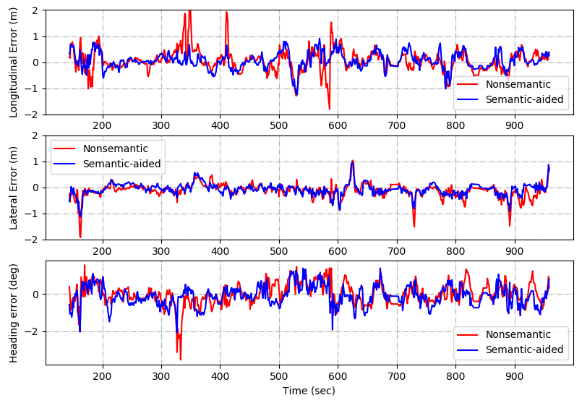


Fig. 10: Longitudinal, lateral and heading error of test route I

Fig. 11 depicts the merging weights G_p, G_v, G_r and G_n of test route I. The original weights histories are very noisy, so the trend of the weights in Fig. 11a are computed as a moving average with a window size of 10 seconds. It is clear that the non-semantic landmarks dominate the localization on most road segments. On the highway (from 200 to 500 seconds), semantic assistance mainly comes from road marks. While on local roads, semantic assistance information is based mostly on pole-like objects. Fig. 11b shows a part of the original weights history on the highway. It is visible that the number of pole-like objects are very limited, so the associated pole-like object weight is usually close to zero and increases rapidly when the vehicle passes by a few poles. This demonstrates that our combined $ADOP$ plus information entropy merging technique yields resourceful and effective behaviour based on available semantic landmarks for positioning.

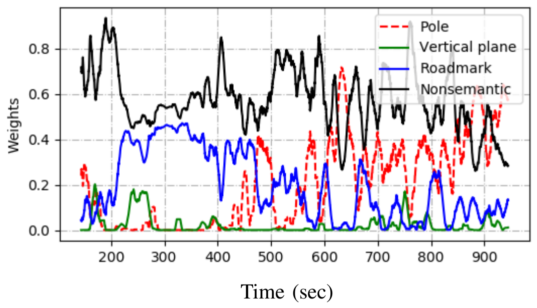
In Fig. 12a, the longitudinal error is denoted by the radius of these circles on the test route I. At some locations on highway, for example at point "A", the non-semantic localization method produces spikes in longitudinal error, and semantic-aided localization method can effectively suppress this error. Fig. 13a shows the point cloud at point "A". Most of the field of view is blocked by dynamic objects, and only the highway

TABLE VI: Positioning Accuracy of test route I

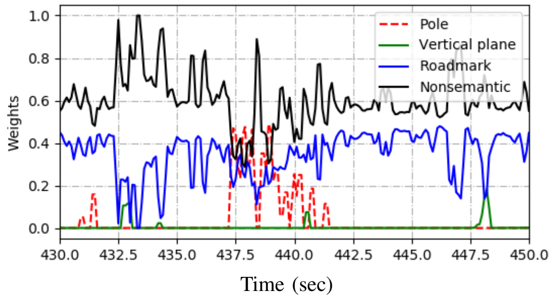
ID		Merge weights	Highway (mean)			Local (mean)			Whole test route (mean)			Whole test route (max)		
			lat	lon	heading	lat	lon	heading	lat	lon	heading	lat	lon	heading
1	Nonsem	–	0.151	0.345	0.482	0.236	0.231	0.437	0.198	0.300	0.445	1.983	2.020	3.892
2	Sem_1	Weight_1	0.158	0.251	0.469	0.170	0.223	0.431	0.171	0.241	0.435	1.359	1.617	2.707
3		Weight_2	0.159	0.471	0.492	0.171	0.263	0.446	0.173	0.379	0.453	1.762	2.371	3.655
4		Weight_3	**0.139**	**0.225**	**0.438**	**0.158**	**0.245**	**0.417**	**0.151**	**0.239**	**0.421**	**1.083**	**1.173**	**2.002**
5	Sem_2	Weight_1	0.163	0.271	0.467	0.201	0.243	0.437	0.199	0.248	0.448	1.832	1.759	3.003
6		Weight_2	0.167	0.481	0.481	0.212	0.281	0.472	0.195	0.354	0.478	1.855	2.586	3.282
7		Weight_3	0.148	0.285	0.446	0.187	0.257	0.436	0.182	0.246	0.442	1.637	1.995	2.896

TABLE VII: Positioning Accuracy of test route II

ID		Merge weights	Office/Business (mean)			Local/Industrial (mean)			Whole test route (mean)			Whole test route (max)		
			lat	lon	heading	lat	lon	heading	lat	lon	heading	lat	lon	heading
1	Nonsem	–	0.127	0.180	0.317	0.242	0.246	0.428	0.235	0.238	0.421	2.351	2.751	3.518
2	Sem_1	Weight_1	0.121	0.174	0.293	0.178	0.235	0.442	0.176	0.233	0.439	1.422	1.693	2.907
3		Weight_2	0.127	0.224	0.315	0.181	0.264	0.451	0.179	0.258	0.448	1.974	2.511	3.480
4		Weight_3	**0.089**	**0.123**	**0.276**	**0.155**	**0.251**	**0.433**	**0.147**	**0.236**	**0.417**	**1.223**	**1.324**	**2.253**
5	Sem_2	Weight_1	0.147	0.201	0.305	0.190	0.229	0.439	0.178	0.217	0.431	1.962	1.873	3.283
6		Weight_2	0.157	0.233	0.337	0.224	0.273	0.455	0.221	0.269	0.453	1.993	2.480	3.437
7		Weight_3	0.139	0.217	0.312	0.205	0.246	0.417	0.199	0.243	0.411	1.781	2.105	3.021



(a) Moving average of weights

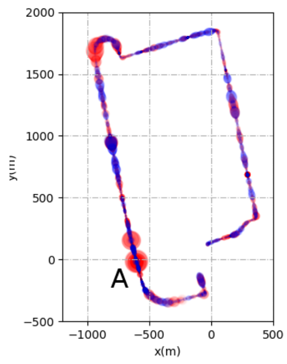


(b) Local history of weights

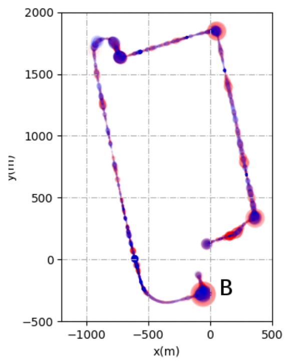
Fig. 11: Time history of weights (test route I)

fence on the left and two pole-like objects on the right can be used as stable landmarks. Since the highway barrier and one of the pole-like objects are much lower than the vehicle on the right, and the non-semantic localization method is inclined to select tall objects as landmarks, it ignores these stable features. Semantic-aided localization solves these situations with ease - when the LiDAR is blocked by dynamic objects, the longitudinal error does not increase significantly.

Fig. 12b shows the lateral errors on the route. The non-semantic localization method produces large lateral errors at open intersections (see point "B"). This is in part attributed to using a front bumper LiDAR. As shown in Fig. 13b, the generated point cloud at point "B" only captures the large open intersection and does not see the narrow road behind the vehicle. In this case, the environment does not provide enough lateral constraint. The semantic-aided method can reduce this error remarkably with semantic landmark constraints, but the lateral error at large intersections is still slightly greater than at other road segments.



(a) Longitudinal error

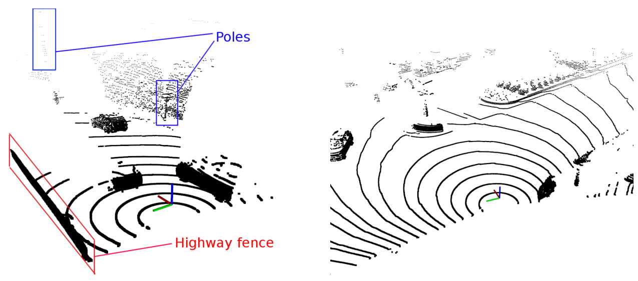


(b) Lateral error

Fig. 12: Localization error shown on trajectory. Curve indicates the localization error. Red: Non-semantic method; Blue: Semantic-aided method

E. Computation time

Table VIII and IX show the computation time of each ROS nodes (to deal with one frame of LiDAR point cloud) in non-semantic and semantic-aided localization methods, respectively. The map management node checks the current position of the vehicle every 0.5 second, so most of the time its CPU usage is zero. The segmentation node of the semantic-aided method needs more CPU resource than the feature extraction node of the non-semantic method becasue it contains the region growing clustering algorithm. The particle



(a) Point cloud blocked by dynamic objects (b) Point cloud at intersection

Fig. 13: Point cloud of the bumper LiDAR

filter node also consumes 5.9 ms more CPU time in the semantic-aided method because of the more complex map matching and weights merging computation.

TABLE VIII: Computation time of each ROS node (Non-semantic localization)

Node	avg time	min time	max time
Map management node	4.8 ms	0 ms	24 ms
Odometry node	3.6 ms	2 ms	5 ms
Feature extraction node	19.6 ms	12 ms	27 ms
Particle filter node	22.8 ms	7 ms	31 ms
Total	50.8 ms	21 ms	87 ms

TABLE IX: Computation time of each ROS node (Semantic-aided localization)

Node	avg time	min time	max time
Map management node	5.1 ms	0 ms	27 ms
Odometry node	3.5 ms	2 ms	5 ms
Segmentation node	42.3 ms	17 ms	46 ms
Particle filter node	28.7 ms	9 ms	37 ms
Total	79.6 ms	28 ms	115 ms

The average CPU time of the proposed method is 79.6 ms (using one of the 8 logical cores). For a 10 Hz LiDAR, the average CPU usage is 9.95%. The maximum processing time is 115 ms, which exceeds the time interval between two LiDAR frames. While, only the operations of the segmentation node and particle filter node are sequential, which take 83 ms in total. Other ROS nodes are running in parallel on other logical cores. So whole system can still run in real time .

V. CONCLUSIONS

This paper develops an algorithm to locate an autonomous vehicle with a 3D spinning LiDAR sensor. By exploiting the geometric properties of a point cloud with robust rule-based scan principles, three classes of semantic landmarks are extracted from the raw LiDAR point cloud in real-time and are used to improve the accuracy of the traditional non-semantic localization method. Augmented dilution of precision and information entropy are used to quantify the credibility of map matching results from different landmark types. The final localization result is obtained by a weighted average of the three semantic and one non-semantic localization results. In road testing, a front bumper LiDAR with an approximate 180 degrees field of view is used to simulate the sensor installment on production vehicles. The results demonstrate that our semantic-aided localization method can substantially reduce longitudinal error, particularly at its maxima. The proposed method also reduces the large lateral error experienced at open intersections, which is a prominent disadvantage of front bumper LiDAR localization. All the while, the efficient design of our system consumes no more than 10% computational power of a desktop CPU.

REFERENCES

[1] A. Das and S. L. Waslander, "Scan registration using segmented region growing ndt," *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1645–1663, 2014.
[2] H. Kim, B. Liu, C. Y. Goh, S. Lee, and H. Myung, "Robust vehicle localization using entropy-weighted particle filter-based data fusion of vertical and road intensity information for a large scale urban area," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1518–1524, 2017.
[3] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-net: Towards learning based lidar localization for autonomous driving," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6382–6391.
[4] A. Schlichting and C. Brenner, "Localization using automotive laser scanners and local pattern matching," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 414–419.
[5] F. Wu, C. Wen, Y. Guo, J. Wang, Y. Yu, C. Wang, and J. Li, "Rapid localization and extraction of street light poles in mobile lidar point clouds: A supervoxel-based approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 292–305, 2017.
[6] M. Sefati, M. Daum, B. Sondermann, K. D. Kreisköther, and A. Kampker, "Improving vehicle localization using semantic and pole-like landmarks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 13–19.
[7] A. Schaefer, D. Bscher, J. Vertens, L. Luft, and W. Burgard, "Long-term urban vehicle localization using pole landmarks extracted from 3-d lidar scans," in *2019 European Conference on Mobile Robots (ECMR)*, 2019, pp. 1–7.
[8] Z. Tao, P. Bonnifait, V. Frmont, and J. Ibaez-Guzman, "Lane marking aided vehicle localization," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013, pp. 1509–1515.
[9] M. Schreiber, C. Knöppel, and U. Franke, "Laneloc: Lane marking based localization using highly accurate maps," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 449–454.
[10] G. I. J. J. H. Im, S. H. Im, "Extended line map-based precise vehicle localization using 3d lidar," *Sensors*, vol. 18, p. 3179, 2018.
[11] A. Y. Hata, F. S. Osorio, and D. F. Wolf, "Robust curb detection and vehicle localization in urban environments," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 1257–1262.
[12] Y. Zhang, J. Wang, X. Wang, C. Li, and L. Wang, "A real-time curb detection and tracking method for ugvs by using a 3d-lidar sensor," in *2015 IEEE Conference on Control Applications (CCA)*, 2015, pp. 1020–1025.
[13] G. I. J. J. H. Im, S. H. Im, "Vertical corner feature based precise vehicle localization using 3d lidar in urban area," *Sensors*, vol. 16, no. 8, p. 1268, 2016.
[14] N. Brasch, A. Bozic, J. Lallemand, and F. Tombari, "Semantic monocular slam for highly dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 393–400.
[15] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
[16] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic slam," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1722–1729.
[17] V. Vineet, O. Miksik, M. Lidegaard, M. Niener, S. Golodetz, V. A. Prisacariu, O. Khler, D. W. Murray, S. Izadi, P. Prez, and P. H. S. Torr, "Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 75–82.

[18] S. Yang, Y. Huang, and S. Scherer, “Semantic 3d occupancy mapping through efficient high order crfs,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 590–597.

[19] P. Li, T. Qin, and a. Shen, “Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[20] K.-N. Lianos, J. L. Schonberger, M. Pollefeys, and T. Sattler, “Vso: Visual semantic odometry,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[21] B. Bescos, J. M. Fcil, J. Civera, and J. Neira, “Dynaslam: Tracking, mapping, and inpainting in dynamic scenes,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[22] J. Mccormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, “Fusion++: Volumetric object-level slam,” in *2018 International Conference on 3D Vision (3DV)*, 2018, pp. 32–41.

[23] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, “Semanticfusion: Dense 3d semantic mapping with convolutional neural networks,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4628–4635.

[24] M. Runz, M. Buffier, and L. Agapito, “Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects,” in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2018, pp. 10–20.

[25] C. Yu, Z. Liu, X. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, “Ds-slam: A semantic visual slam towards dynamic environments,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1168–1174.

[26] J. Yan, D. Chen, H. Myeong, T. Shiratori, and Y. Ma, “Automatic extraction of moving objects from image and lidar sequences,” in *2014 2nd International Conference on 3D Vision*, vol. 1, 2014, pp. 673–680.

[27] J. Wang and J. Kim, “Semantic segmentation of urban scenes with a location prior map using lidar measurements,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 661–666.

[28] J. Jeong, T. S. Yoon, and J. B. Park, “Multimodal sensor-based semantic 3d mapping for a large-scale environment,” *Expert Systems with Applications*, vol. 105, pp. 1 – 10, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418302082

[29] J. Jeong, T. S. Yoon, and J. B. Park, “Towards a meaningful 3d map using a 3d lidar and a camera,” *Sensors*, vol. 18, p. 2571, 2018.

[30] R. Dudé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, “Segmap: 3d segment mapping using data-driven descriptors,” in *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[31] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, “Recurrent-octomap: Learning state-based map refinement for long-term semantic mapping with 3-d-lidar data,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3749–3756, 2018.

[32] X. Chen, A. Milioto, E. Palazzolo, P. Gigure, J. Behley, and C. Stachniss, “Suma++: Efficient lidar-based semantic slam,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4530–4537.

[33] L. Sun, D. Adolfsson, M. Magnusson, H. Andreasson, I. Posner, and T. Duckett, “Localising faster: Efficient and precise lidar-based robot localisation in large-scale environments,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4386–4392.

[34] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss, “Overlapnet: Loop closing for lidar-based slam,” in *RSS 2020*, 2020.

[35] X. Chen, T. Läbe, L. Nardi, J. Behley, and C. Stachniss, “Learning an overlap-based observation model for 3d lidar localization,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4602–4608.

[36] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “Rangenet ++: Fast and accurate lidar semantic segmentation,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 11 2019, pp. 4213–4220.

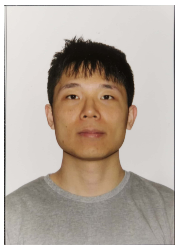
[37] E. E. Aksoy, S. Baci, and S. Cavdar, “Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving,” in *IEEE Intelligent Vehicles Symposium (IV) 2020*, 2020.

[38] J. Zhang and S. Singh, “Loam: Lidar odometry and mapping in real-time,” *Robotics: Science and Systems X*, 2014.

[39] C. Chen, “Weighted geometric dilution of precision calculations with matrix multiplication,” *Sensors*, vol. 15, pp. 803–817, 2015.

Yuan Ren is a Principal Engineer with Noahs Ark Lab, Huawei. Previously he worked as a Research Fellow with York University, Canada, and Marie Curie experienced researcher with Polytechnic University of Catalonia, Spain. He received his Bachelor, Master and Ph.D degree from Harbin Institute of Technology, China. His research interests focus on autonomous vehicle self-localization, inertial navigation and dynamical systems theory.

Bingbing Liu is a Senior Principal Engineer with Noah’s Ark Lab, Huawei. Previously he worked as a Research Scientist with Institute for Infocomm Research, A*STAR Singapore. He received his Bachelor degree from Harbin Institute of Technology, China and the Ph.D degree from Nanyang Technological University, Singapore respectively. His research interests include perception technologies for autonomous driving, SLAM, inertial navigation, etc.



Ran Cheng is working on visual SLAM on mobile robots, reinforcement learning and deep learning for visual based navigation. He obtained his master degree from McGill University under the supervision of Gregory Dudek. Previously he worked with Professor Jianwei Lu and Professor Laurent Itti in Tongji University where he conducted various research on LiDAR based SLAM and multi-sensor fusion on mobile robots.

Christopher Agia is a graduate researcher at Stanford University where he works on task-driven perception, planning and control for robot systems with a variety of deep learning techniques. He completed his Bachelor’s degree in Engineering Science, Robotics and Artificial Intelligence from the University of Toronto in 2021. He spent 2019 to 2020 as an Autonomous System Researcher at Huawei Noah’s Ark Research Lab before shortly after joining Google and Microsoft Mixed Reality and Robotics for several months each.