Performance Improvement of Loop Closure Detection Using Semantic Segmentation in LiDAR SLAM

Shuya Muramatsu
Graduate School of Science and Engineering
Doshisha University
Kyotanabe, Kyoto, Japan
e-mail: ctwk0138@mail4.doshisha.ac.jp

Masafumi Hashimoto, Kazuhiko Takahashi
Faculty of Science and Engineering
Doshisha University
Kyotanabe, Kyoto, Japan
e-mail: {mhashimo, katakaha}@mail.doshisha.ac.jp

***Abstract*—This paper proposes a method to improve loop closure detection performance in Normal Distributions Transform (NDT) Graph Simultaneous Localization And Mapping (SLAM) using Light Detection And Ranging (LiDAR). Candidates of revisit places (loops) are detected from semantic information, such as buildings, fences, vegetation, trunks, poles, and traffic signs, based on semantic segmentation of LiDAR point cloud data. Loops are determined from loop candidates using semantic information-based point feature histograms in conjunction with conventional geometric information-based point feature histograms. Then, vehicle poses relative to loops are calculated based on the matching of point features and are applied to correct errors accumulated by NDT SLAM in the graph optimization framework. The projected-based semantic segmentation RangeNet++ is used to obtain semantic information about the surrounding environment. The experimental results obtained using the Semantic KITTI dataset demonstrate the performance of the proposed method.**

Keywords—LiDAR SLAM; NDT-Graph SLAM; loop closure detection; semantic segmentation.

I. INTRODUCTION

Recently, in the fields of Intelligent Transportation Systems (ITS) and mobile robotics, many studies on Autonomous Driving Systems (AVS) and Advanced Driver Assistant Systems (ADAS) have been reported [1][2]. Simultaneous Localization And Mapping (SLAM) and object recognition using onboard sensors, such as cameras and Light Detection And Ranging sensors (LiDARs), are important technologies for AVS and ADAS [3][4]. In this study, we focus on LiDAR SLAM.

Scan-matching SLAM using Normal Distributions Transform (NDT) and iterative closest point techniques is typically used in LiDAR SLAM [5]. However, the scan-matching SLAM causes accumulation errors as the travel distance of the vehicle increases. To reduce accumulation errors, Graph SLAM is employed in conjunction with scan-matching SLAM. In Graph SLAM, the detection of revisit places (referred to as loops) is a crucial issue, and many loop detection methods have been proposed [6].

A conventional loop detection method is based on geometric surface shape features of surrounding objects, such as poles and plains [7][8]. Loops are extracted based on the shape of the distribution of LiDAR point cloud data for objects, and loops are detected by checking the similarity of the surface shape features between two places. Another conventional geometric loop detection method is based on point features, such as Fast Point Feature Histograms (FPFH) [9] and normal-aligned radial features [10]. However, geometric surface and point features cannot distinguish objects with similar shapes, such as utility poles and trees; thus, false loop detection typically occurs.

In our previous works [11][12], to reduce false and miss detections of loops using geometric features, a two-stage loop detection method was proposed. First, loop candidates were detected based on the surface shape features. Next, loops were detected from loop candidates based on point features using FPFH and the three-Points Congruent Sets (3PCS) method. However, improvements are required to reduce miss and false loop detection.

Recently, in the ITS and mobile robotics domains, deep learning-based SLAM and object recognition methods have been actively discussed [13]. In SLAM, semantic segmentation improves the representation of surface and point features and the accuracy and robustness of loop detection. Many semantic segmentation-based loop detection methods have then been proposed [14]–[16]. However, applying semantic information to loop detection remains a challenge.

From this viewpoint, this paper improves the performance of our previous geometric loop detection method [12] by introducing semantic information from the RangeNet++ semantic segmentation method [17]. RangeNet++ is a projection-based semantic segmentation method with lower computation overhead and memory requirements than point- and voxel-based methods [18]. For this reason, RangeNet++ is used in this study.

The remainder of this paper is organized as follows. Section II describes LiDAR SLAM. Section III explains loop detection method. Section IV presents the experimental results obtained from the KITII dataset [19] to demonstrate the performance of the proposed method, followed by the conclusions in Section V.

II. OVERVIEW OF LIDAR SLAM

LiDAR SLAM is based on NDT-Graph SLAM. First, the point cloud data captured from a vehicle-mounted LiDAR are mapped onto a three-dimensional (3D) grid map (voxel map) represented in the LiDAR coordinate system attached to the LiDAR. Then, a voxel grid filter is applied to downsize the LiDAR point cloud data. The block used for the voxel grid filter is a cube with a side length of 0.2 m.

In the world coordinate system, a voxel map with a voxel size of 1 m is used for NDT scan matching. For the i-th ($i = 1, 2, \ldots n$) measurement in the LiDAR point cloud data, the position vector in the LiDAR coordinate system is denoted as $\boldsymbol{p}_{bi}$ and

that in the world coordinate system is denoted as $\boldsymbol{p}_i$. The following relation is obtained:

$$\begin{pmatrix} \boldsymbol{p}_i \\ 1 \end{pmatrix} = \boldsymbol{T}(\boldsymbol{x}) \begin{pmatrix} \boldsymbol{p}_{bi} \\ 1 \end{pmatrix} \quad (1)$$

where $\boldsymbol{x} = (x, y, z, \phi, \theta, \psi)^T$ denotes the pose of the vehicle. (x, y, z) and (ϕ, θ, ψ) denote the 3D position and attitude angle (roll, pitch, and yaw angles) of the vehicle, respectively, in the world coordinate system. $\boldsymbol{T}(\boldsymbol{x})$ denotes the homogeneous transformation matrix.

The LiDAR point cloud data obtained at the current time step are referred to as the current point cloud data, and the point cloud data obtained up to the previous time step are referred to as the reference map. The vehicle pose $\boldsymbol{x}$ at the current time step is determined by matching the current point cloud data with the reference map. The vehicle pose is used for coordinate transform using (1). Then, the current point cloud can be mapped to the world coordinate system, and the reference map is updated.

The LiDAR obtains point cloud data by scanning laser beams. Thus, when a vehicle moves, the entire point cloud data in the LiDAR sampling period cannot be acquired at a single point in time. Therefore, if the entire point cloud data at the current time step are mapped onto the world coordinate system using vehicle-pose information at a single point in time, motion artifact (distortion) arises in the LiDAR point cloud data and degrades SLAM accuracy. The motion artifact is corrected using an unscented Kalman filter-based algorithm [20].

The point cloud data captured by the LiDAR is processed into semantic segmentation using RageNet++, and semantic information is obtained for each LiDAR measurement. In this study, 12 semantic classes are considered: cars, two-wheelers, other vehicles, people, roads, lawns, buildings, fences, vegetation, trunks, poles, and traffic signs. As shown in Figure 1, LiDAR point cloud data with semantic information are mapped onto a voxel map with a voxel size of 1 m. Then, in each voxel, majority voting of the semantic class of the LiDAR point cloud data is selected, and the voxel semantic class is determined. Figure 2 presents an example of voxel semantic classification.

In SLAM, the use of LiDAR point cloud data related to moving objects (referred to as moving point cloud data), such as cars, two-wheelers, and pedestrians, degrades accuracy. Thus, these data are removed using an occupancy grid method, and those related to stationary objects (stationary point cloud data) are used in NDT SLAM. Semantic segmentation can detect moving objects. However, this paper focuses on loop detection using semantic information; thus, moving point cloud data are removed using the occupancy grid method.

The accuracy of NDT SLAM deteriorates over time due to accumulation errors. To reduce the error, Graph SLAM is employed [21]. Figure 3 shows a pose graph for Graph SLAM. Here, the relative poses of the vehicle, which are calculated by NDT SLAM for every LiDAR sampling period, are set to a pose graph as graph edges (the black arrows depicted in Figure 3). When loops previously visited by the vehicle are detected using the method described in Section III, the current pose of the vehicle relative to its pose at the revisit node is set to the pose

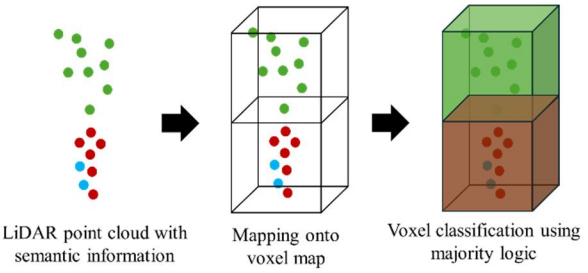


Figure 1. Flow of voxel classification using semantic information.

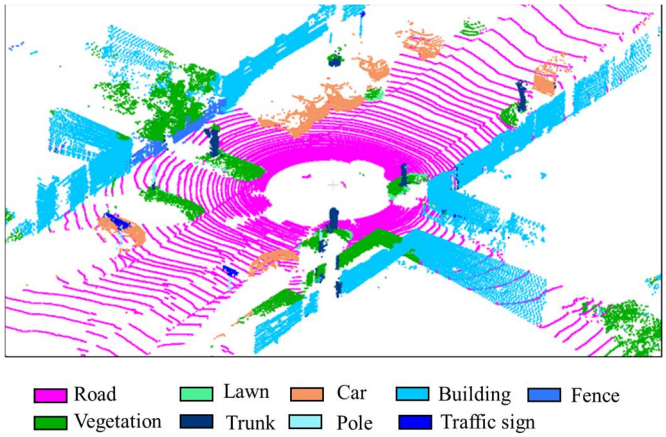


Figure 2. Example of voxel semantic classification using RangeNet++.

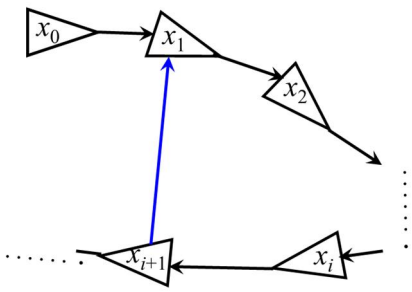


Figure 3. Pose graph. The black triangle indicates the graph node (vehicle pose), and the black and blue arrows indicate graph edges (relative poses between two neighboring nodes and loop constraint, respectively).

graph as a loop constraint (the blue arrow in Figure 3). Using the least squares method, Eq. (2) is then minimized to improve the accuracy of NDT SLAM as follows:

$$J(\boldsymbol{\chi}) = \sum_i \{(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) - \boldsymbol{\delta}_{i+1,i}\}^T \boldsymbol{\Omega}^{pose} \{(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) - \boldsymbol{\delta}_{i+1,i}\}$$
$$+ \sum_{x_A, x_B \in \text{loop}} \{(\boldsymbol{x}_B - \boldsymbol{x}_A) - \boldsymbol{\delta}_{A,B}\}^T \boldsymbol{\Omega}^{loop} \{(\boldsymbol{x}_B - \boldsymbol{x}_A) - \boldsymbol{\delta}_{A,B}\} \quad (2)$$

where the first and second terms on the right side indicate the constraints on NDT SLAM and loops, respectively; $\boldsymbol{\chi} = (\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \cdots, \boldsymbol{x}_i^T, \cdots)^T$; $\boldsymbol{x}_i$ denotes the vehicle pose at the i-th time step; $\boldsymbol{\delta}_{i+1,i}$ denotes the relative pose of the vehicle between the i-th and $(i+1)$th time steps, which is calculated from NDT SLAM; $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ denote the vehicle poses at the loop and current nodes, respectively; $\boldsymbol{\delta}_{A,B}$ denotes the relative pose of

the vehicle at the two nodes, which is calculated from the LiDAR point cloud using NDT scan matching; $\boldsymbol{\Omega}^{pose}$ and $\boldsymbol{\Omega}^{loop}$ denote the weighting matrices.

III. Loop Detection Method

A. Detection of Loop Candidates

A key component of Graph SLAM is loop detection. To this end, loop candidates are first obtained using vehicle self-location information via NDT SLAM. If the distance between an old node and the current node is less than 15 m, the old node is recognized as a loop candidate.

In our previous work [11][12], geometric surface shape features of surrounding objects were used to identify loop candidates (see Appendix A). In this paper, semantic information is used to improve the performance of selecting loop candidates. The similarity indicator (referred to as Semantic Similarity Indicator, SSI) is calculated to select loop candidates. For the SSI, six voxel semantic classes are considered: buildings, fences, vegetation, trunks, poles, and traffic signs.

Based on these classes of voxels, two feature descriptors, $Fs = (f_1, f_2, \ldots, f_6)$ and $Gs = (g_1, g_2, \ldots, g_6)$, are defined, where Fs denotes the feature descriptor, which is calculated from LiDAR point cloud data captured at loop candidates, and Gs denotes the feature descriptor, which is calculated from LiDAR point cloud data captured at the current place; f_i and g_i, $i = 1, 2, \ldots, 6$, denotes the number of the i-th semantic class. The SSI is then defined using the feature descriptors as follows:

$$SSI = \frac{\sum_{i=1}^{6}\{\max(f_i, g_i) - |f_i - g_i|\}}{\sum_{i=1}^{6}\max(f_i, g_i)} \times 100 \quad (3)$$

A higher degree of similarity between the LiDAR point cloud data at loop candidates and current places leads to a larger SSI. Thus, if the SSI is 60% or higher, the loop candidates are accepted.

B. Loop Detection and Relative Pose Calculation

Loops are determined from the loop candidates using a Point cloud Overlap Indicator (POI). From two LiDAR point cloud data captured at the current place and each loop candidate, the relative pose of the vehicle is calculated using NDT scan matching. The POI is then given by:

$$POI = \frac{1}{n}\sum_{i=1}^{n}\text{overlap}(d_i) \times 100 \quad (4)$$

where n represents the number of measurements in the LiDAR point cloud data captured at the current place of the vehicle; d_i denotes the nearest neighbor distance between the measurements in the LiDAR point cloud data captured at the current place and each loop candidate. The function overlap (d_i) is defined by

$$\text{overlap}(d_i) = \begin{cases} 1 & \text{for } d_i \le \text{threshold} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

A higher similarity between the LiDAR point cloud data captured at the current place and loop candidate leads to a smaller POI value. Thus, if the POI value is 80% or higher, the loop can be detected from the loop candidates.

In NDT scan matching, if the initial relative pose is incorrect, both the relative pose estimate and POI become inaccurate due to local minima issues. In our previous work [12], to accurately set the initial relative pose, geometric point feature histograms, FPFH, were used. In this paper, geometric and semantic point feature histograms (referred to as Semantic Point Feature Histograms, SPFH) are presented to improve the performance of setting the initial relative pose.

First, LiDAR point cloud data captured at the current place are mapped onto a voxel map (grid size of 0.2 m) in the LiDAR coordinate system and downsampled using a voxel grid filter. The centroid of the point cloud data in the i-th voxel ($i = 1, 2, \ldots$) on the voxel map is then obtained. The centroid is referred to as the feature point A_i in this paper. The geometric three-angle feature in the feature point is calculated, and geometric point feature histograms (33 dimensions in this study) are then calculated based on FPFH (see Appendix B).

In addition, a semantic feature histogram is calculated using the semantic information of the six semantic classes: buildings, fences, vegetation, trunks, poles, and traffic signs (see Section III-A).

Let A_i be the feature point in the i-th voxel and A_j be the feature point in the j-th voxel ($j = 1, 2, ..., n$) located around the i-th voxel. Here, semantic information is assigned to each feature point, which is classified as a voxel semantic class using the voxel classification method shown in Figure 1.

By obtaining the semantic class of the feature point A_i and the semantic classes of the n feature points ($A_1, A_2, \cdots, A_n$) around A_i, a six-dimensional semantic feature histogram, which is the ratio of each semantic class of A_i to the n feature points, is calculated. Here, n represents the number of voxels within a radius of 2 m from the i-th voxel. The semantic feature histogram of A_i is referred to as Simplified Semantic Histograms (SSH) of A_i, $\boldsymbol{SSH}(A_i)$. Similarly, $\boldsymbol{SSH}(A_j)$ is calculated for the feature point A_j ($j = 1, 2, ..., n$) around A_i.

The six-dimensional Semantic Histogram (SH) related to A_i is then given by

$$\boldsymbol{SH}(A_i) = \boldsymbol{SSH}(A_i) + \frac{1}{n}\sum_{j=1}^{n} w_j \boldsymbol{SSH}(A_j) \quad (6)$$

where the weight w_i is given as the inverse number of the distance between A_i and A_j.

The six-dimensional semantic feature histogram is combined with the 33-dimensional geometric feature histogram using FPFH. The 39-dimensional point feature histogram (referred to as Semantic Fast Point Feature Histogram, SFPFH) is then used to accurately set the initial relative pose.

From the LiDAR point cloud data captured at each loop candidate, the feature point B_i and the related SFPFH are obtained in the same way. Their feature points A_i and B_i are matched using the 3PCS method [12] as follows:

Step 1: Three feature points A_i ($i = 1, 2, 3$) are randomly extracted from the set of feature points obtained at the current location, and a triangle $\tilde{A}$ is composed of the three feature points A_1, A_2, and A_3. Then, 100 feature points B_j ($j = 1, 2, \ldots,$

100) with similar SFPFH as those of A_i (i = 1, 2, 3) are extracted from the set of feature points obtained at each loop candidate using the k-nearest neighbor method. A triangle $\tilde{B}$ is composed of any three feature points from 100 feature points B_j. The three feature points B_1, B_2, and B_3 are selected such that the two triangles $\tilde{A}$ and $\tilde{B}$ are congruent.

Step 2: The pose of the loop candidate relative to the current place is denoted by $\delta \boldsymbol{x} = (\delta x, \delta y, \delta z, \delta\phi, \delta\theta, \delta\psi)^T$, where $(\delta x, \delta y, \delta z)$ and $(\delta\phi, \delta\theta, \delta\psi)$ denote the relative position and attitude angle, respectively.

In the matched triangles $\tilde{A}$ and $\tilde{B}$, the centroid positions of the three-feature point sets (A_1, A_2, A_3) and (B_1, B_2, B_3) are denoted by $\bar{\boldsymbol{a}}$ and $\bar{\boldsymbol{b}}$, respectively. The positions of the feature points A_i and B_i, where i = 1, 2, 3, are denoted by $\boldsymbol{a}_i^*$ and $\boldsymbol{b}_i^*$, respectively. The feature point matrices are then given by $\delta \boldsymbol{a} = (\delta \boldsymbol{a}_1, \delta \boldsymbol{a}_2, \delta \boldsymbol{a}_3)^T$ and $\delta \boldsymbol{b} = (\delta \boldsymbol{b}_1, \delta \boldsymbol{b}_2, \delta \boldsymbol{b}_3)^T$, where $\delta \boldsymbol{a}_i \triangleq \boldsymbol{a}_i^* - \bar{\boldsymbol{a}}$ and $\delta \boldsymbol{b}_i \triangleq \boldsymbol{b}_i^* - \bar{\boldsymbol{b}}$. Based on the matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, which are defined by the singular value decomposition ($\boldsymbol{H} = \boldsymbol{W}_1 \boldsymbol{\Sigma} \boldsymbol{W}_2^T$) of the matrix $\boldsymbol{H} = \delta \boldsymbol{b} \cdot \delta \boldsymbol{a}^T$, the homogeneous matrix related to the relative pose $\delta \boldsymbol{x}$ is given by

$$\boldsymbol{T}(\delta \boldsymbol{x}) = \begin{pmatrix} \boldsymbol{W}_2 \boldsymbol{W}_1^T & \bar{\boldsymbol{a}} - \boldsymbol{R}\bar{\boldsymbol{b}} \\ 0 & 1 \end{pmatrix} \tag{7}$$

where $\boldsymbol{R}$ represents the rotational matrix related to $(\delta\phi, \delta\theta, \delta\psi)$.

The position of the i-th feature point A_i (i = 1, ..., n_A) in the feature point set A is denoted by $\boldsymbol{a}_i$, and that of the j-th feature point B_j (j = 1, ..., n_B) in the feature point set B is denoted by $\boldsymbol{b}_j$. The distance r_j between $\boldsymbol{b}_j$ and its nearest neighbor feature point $\boldsymbol{a}_i$ is then calculated by

$$r_j = \left\{ \begin{pmatrix} \boldsymbol{a}_i \\ 1 \end{pmatrix} - \boldsymbol{T}(\delta \boldsymbol{x}) \begin{pmatrix} \boldsymbol{b}_j \\ 1 \end{pmatrix} \right\}^T \left\{ \begin{pmatrix} \boldsymbol{a}_i \\ 1 \end{pmatrix} - \boldsymbol{T}(\delta \boldsymbol{x}) \begin{pmatrix} \boldsymbol{b}_j \\ 1 \end{pmatrix} \right\} \tag{8}$$

Step 3: Steps 1 and 2 are repeated 100 times to find the relative pose $\delta \boldsymbol{x}$ with the largest value in the following cost function:

$$J = \frac{1}{n_B} \sum_{j=1}^{n_B} \mathrm{overlap}(r_j) \tag{9}$$

where overlap (r_i) denotes the overlap function defined in (5).

The relative pose $\delta \boldsymbol{x}$ is then obtained. In NDT scan matching, the relative pose $\delta \boldsymbol{x}$ is used as the initial value, and an iterative calculation is performed. Therefore, the accurate relative pose is calculated, and the POI in (4) can be obtained accurately.

IV. Fundamental Experiments

A. Dataset

The Semantic KITTI dataset [19] is used to evaluate the performance of the proposed method. In the KITTI dataset, a mechanical 64-layer LiDAR (Velodyne HDL-64E) is mounted on a vehicle. The horizontal field of view of the LiDAR is 360° with an angular resolution of 0.08°, and the vertical field of view is 26.8° with an angular resolution of 0.42°. The LiDAR sampling period is 0.1 s, where the laser beam makes one rotation in the horizontal direction. The LiDAR obtains approximately 120,000 point cloud data points during the sampling period. Each observation point has information about its 3D position and reflection intensity.

In the KITTI dataset, 11 sequences (sequences 00–10) containing true semantic information, are used in this study. Four sequences (sequences 00, 02, 05, and 08), which have many loops, are used to evaluate the proposed method. Therefore, cross-validation is performed for the four sequences; thus, when sequence 00 is used for evaluation data, sequences 01–10 are used as training data, when sequence 02 is used for evaluation data, sequences 00, 01, and 03–10 are used as training data, and so on. The distances traveled by the vehicle in sequences 00, 02, 05, and 08 are 3.7, 5.0, 2.2, and 3.2 km, respectively.

B. Performance of Object Recognition by RangeNet++

The recognition performance of RangeNet++ is evaluated. Table I shows the accuracy (Intersection over Union (IoU), precision, and recall) for 12 semantic classes. As shown in the table, accuracy tends to be high for large objects, such as cars, roads, buildings, and vegetation, whereas it decreases for small objects, such as people and traffic signs.

For the six sematic voxel classes used in the loop detection method, including buildings, fences, vegetation, trunks, poles, and traffic signs, the average IoU, precision, and recall aree 58.6%, 73.7%, and 72.1%, respectively.

C. Performance of Loop Detection

The performance of SSI and SFPFH in loop detection is evaluated. First, a dataset of pairs of loops and current places with a relative distance of α m is generated from the KITTI dataset; these pairs should be detected accurately as true loops. The relative distance is set at $0 \le \alpha \le 5$ m and $5 < \alpha \le 10$ m. In addition, a dataset of pairs of loops and current places with a relative distance $\alpha \ge 300$ m is generated; these pairs should be detected as false loops.

The performance of loop detection is then evaluated using the true and false loop datasets. If a true loop is determined to be a loop, it is considered a correct loop detection; if a true loop is determined not to be a loop, it is considered a miss loop detection; if a false loop is considered a loop, it is considered a false loop detection.

First, the performance of loop candidate detection is evaluated in terms of the SSI. The precision and recall results of loop candidate detection are shown in Table II. For comparison, the results of loop candidate detection using the conventional geometric method (Geometric Similarity Indicator, GSI) [12]

TABLE I. Performace of Object Classification using RangeNet++

Class	IoU [%]	Precision [%]	Recall [%]
Car	89.5	91.4	97.7
Two-wheeler	43.6	63.1	58.6
Other vehicles	40.8	69.1	49.9
Person	22.8	52.8	28.7
Road	94.2	97.9	96.1
Lawn	65.4	81.0	77.4
Building	84.5	90.9	92.5
Fence	56.1	73.2	70.6
Vegetation	80.5	88.1	90.3
Trunk	53.1	68.5	70.2
Pole	40.1	51.5	64.5
Traffic sign	37.4	69.9	44.5

TABLE II. PERFORMANCE OF LOOP CANDIDATE DETECTION IN TERMS OF SSI AND GSI

Sequence	Recall		Precision
	$0 \le \alpha \le 5$ m	$5 < \alpha \le 10$ m	
00	99.7 (82.6)	98.8 (75.6)	64.0 (54.1)
02	99.9 (67.9)	99.4 (62.8)	62.5 (52.9)
05	98.2 (80.7)	98.9 (79.7)	53.5 (49.9)
08	100 (78.3)	99.4 (72.9)	61.3 (52.0)

The performance evaluation in terms of conventional GSI is shown in brackets.

TABLE III. PERFORMANCE OF LOOP DETECTION IN TERMS OF SFPFH AND FPFH

Sequence	$0 \le \alpha \le 5$ m	$5 < \alpha \le 10$ m
00	96.2 (94.3)	71.0 (65.7)
02	89.9 (85.3)	59.8 (58.0)
05	92.5 (92.7)	67.0 (65.2)
08	89.2 (77.8)	74.8 (57.1)

The performance evaluation in terms of conventional FPFH is shown in brackets.

(Appendix A) are also shown in the table. The higher the precision value, the fewer the incorrectly detected loop candidates, and the higher the recall value, the fewer the falsely extracted loop candidates. As shown in the table, the proposed SSI provides higher loop candidate extraction performance than the conventional GSI.

Next, the performance of loop detection is evaluated in terms of using SFPFH. The Matching Success Score (MSS) is defined as follows:

$$MSS = \frac{1}{n}\sum_{i=1}^{n} f(POI_i) \times 100 \qquad (10)$$

where n represents the number of true loop pair; POI_i denotes the POI of the i-th pair. The function $f(POI_i)$ is defined by

$$f(POI_i) = \begin{cases} 1 & \text{for } POI_i \le \text{threshold} \\ 0 & \text{otherwise} \end{cases} \qquad (11)$$

Higher loop detection accuracy from loop candidates leads to a larger *MSS*. The result is shown in Table III. For comparison, the results obtained using the conventional geometric method (FPFH) are also shown in the table. The results show that the proposed method (SFPFH) provides higher loop detection performance than the conventional method (FPFH).

D. SLAM Performance

Figures 4 and 5 show environment maps of sequence 00 constructed by SLAM using the proposed and conventional geometric methods, respectively. Figure 6 shows the vehicle movement path estimated by SLAM. As an enlarged map shown in Figure 5, the conventional method distorts the environment map because of miss detections of loops, whereas the proposed method does not cause distortion, as shown in Figure 4. In addition, in the area surrounded by the pink frame in Figure 6, the conventional method has a large error in vehicle self-pose estimation, whereas the proposed method achieves accurate estimation of vehicle self-pose.

Figure 7 shows an image of a location where distortion in the environment map occurs using the conventional method. The image contains many parked cars and the surface features of the cars are all similar; thus, incorrect loops are detected using only the conventional FPFH. This is why distortion occurs when using the conventional method. However, in such an environment, the proposed SFPFH can accurately detect loops.

Next, the following three evaluation values are used to evaluate SLAM performance in sequences 00, 02, 05, and 08:

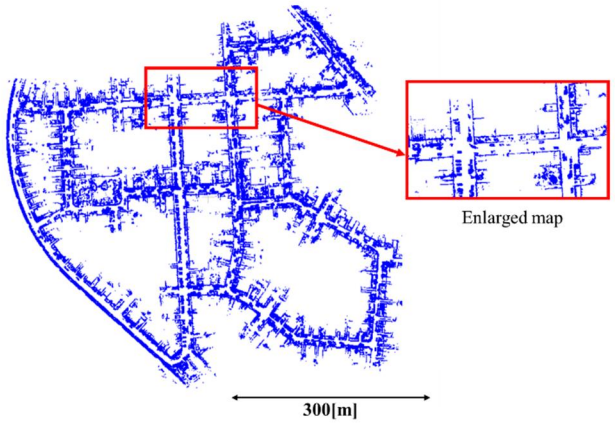


Figure 4. Environment map built using proposed method.

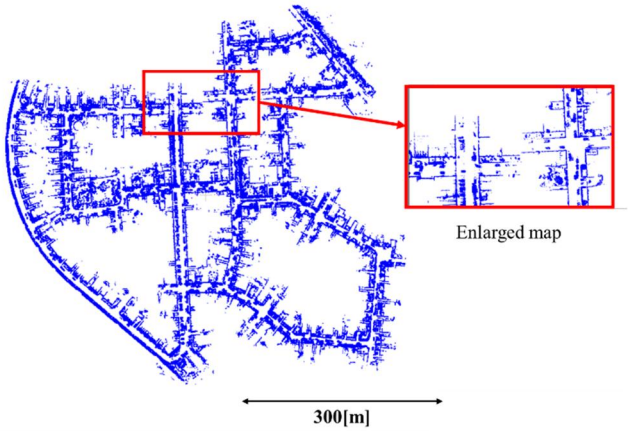


Figure 5. Environment map built using conventional method.

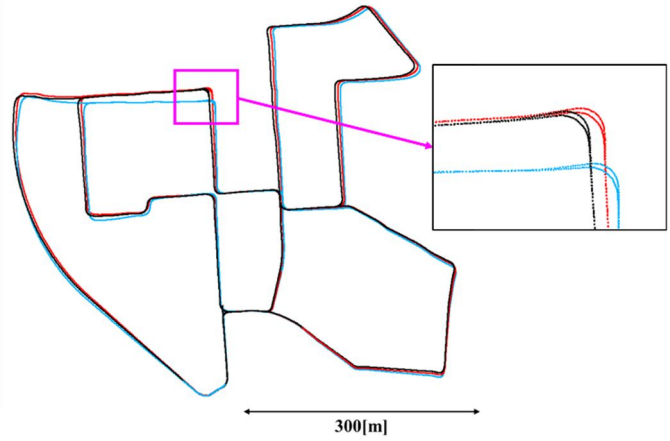


Figure 6. Vehicle movement path estimated using SLAM. The black line indicates the ground truth. The red and light blue lines indicate the results obtained using proposed and conventional methods, respectively.

Figure 7. Photo of area surrounded by the red frame in Figures 4 and 5.

TABLE IV. SLAM PERFORMANCE

Sequence	ATE [m]	NLD	MELC [%]
00	3.19 (6.92)	1030 (889)	9.6 (18.0)
02	38.62 (94.14)	334 (213)	51.6 (68.0)
05	3.75 (4.55)	594 (575)	23.5 (24.3)
08	11.05 (11.63)	458 (424)	12.1 (17.5)

The performance evaluation using the conventional geometric method is shown in brackets.

- Absolute Trajectory Error (ATE): Root-Mean-Square (RMS) error of estimate of vehicle self-position
- Number of Loop Detection (NLD)
- Miss Extraction rate of Loop Candidates (MELC)

The results are shown in Table IV. The proposed method achieves a smaller ATE, larger NLD, and smaller MELC than the conventional method, which demonstrates its superior SLAM performance improvement.

V. CONCLUSION AND FUTURE WORK

This paper presented a method for improving loop detection performance using semantic information in LiDAR SLAM. Semantic information about the surrounding environment was recognized using RangeNet++, and the SSI and SFPFH were introduced to improve the accuracy of loop detection in NDT-Graph SLAM. Experiments using the Semantic KITTI dataset were conducted to evaluate the performance of the proposed method compared with our previous geometry-based loop detection method.

This paper focused on applying semantic information to loop detection in Graph SLAM. Currently, we are applying semantic information to NDT SLAM in conjunction with Graph SLAM. In addition, semantic information will be applied to map updates and maintenance in the future.

ACKNOWLEDGMENT

This study was partially supported by the KAKENHI Grant #23K03781, the Japan Society for the Promotion of Science (JSPS).

APPENDIX A: GEOMETRY BASED DETECTION OF LOOP CANDIDATES

Loop candidates are detected based on a Geometric Similarity Indicator (GSI) [11][12], which is calculated using LiDAR point cloud data captured at the loop candidate and current place of the vehicle. LiDAR point cloud data are mapped onto the voxel map (grid size of 1 m). Each grid of the voxel map is first classified into three types: line, plane, or other voxels (Figure A). Three eigenvalues ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$) are calculated from LiDAR point cloud data in voxels based on principal component analysis, and the following features are calculated:

$$r_1 = \frac{\sqrt{\lambda_1} - \sqrt{\lambda_2}}{\sqrt{\lambda_1}},\ r_2 = \frac{\sqrt{\lambda_2} - \sqrt{\lambda_3}}{\sqrt{\lambda_1}},\ r_3 = \frac{\sqrt{\lambda_3}}{\sqrt{\lambda_1}} \quad \text{(A)}$$

When the maximum values are r_1, r_2, and r_3, the voxel is determined as being of line, plane, or other types. Based on the surface normal vector of the plane voxels, the plane voxels are further divided into nine classes which are different directions of normal vectors.

The two feature descriptors $\boldsymbol{A} = (a_1, a_2, \cdots, a_{11})^T$ and $\boldsymbol{B} = (b_1, b_2, \cdots, b_{11})^T$ are defined. $\boldsymbol{A}$ is calculated from LiDAR point cloud data captured at loop candidates, and $\boldsymbol{B}$ is calculated from the LiDAR point cloud data at the current place. a_1 and b_1 denote the numbers of line voxels in the voxel map. $a_2 - a_{10}$ and $b_2 - b_{10}$ denote the numbers of plane voxels divided into nine classes. a_{11} and b_{11} denote the numbers of other voxels.

From the feature descriptors $\boldsymbol{A}$ and $\boldsymbol{B}$, the GSI is given by

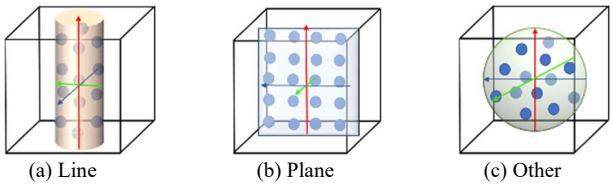
$$GSI = \frac{\sum_{i=1}^{11} \{\max(a_i, b_i) - |a_i - b_i|\}}{\sum_{i=1}^{11} \max(a_i, b_i)} \quad \text{(B)}$$

Among loop candidates, loops with large GSI values can be accepted.

LiDAR point cloud data are sparse in the vertical direction of the mechanical LiDAR; thus, voxels located far from LiDAR have fewer points in the voxel. This reduces the classification accuracy of surface shape features. To solve this problem, when voxels are located far from LiDAR (35 m or more in this study), adjacent voxels are combined, and all point cloud data of the combined voxels are used for principal component analysis.

APPENDIX B: FPFH FOR GEOMETRIY BASED LOOP DETECTION

Let $\boldsymbol{a}_i$ be the position of feature point A_i in the i-th voxel and $\boldsymbol{a}_j$ be the position of feature point A_j in the j-th voxel around the i-th voxel. As shown in Figure B, Let $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ be the normal vectors of these feature points and ($\boldsymbol{a}_j$ - $\boldsymbol{a}_i$) be the difference vector of feature point positions. Using these vectors, the



(a) Line (b) Plane (c) Other

Figure A. Voxel classification.

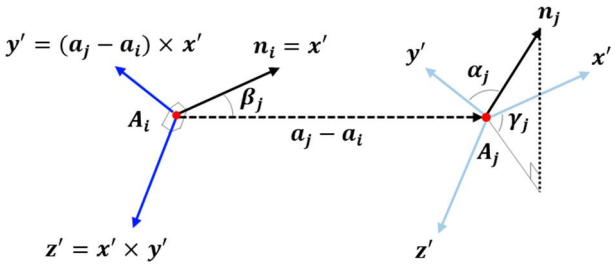


Figure B. Angle features.

angular feature (α_j, β_j, γ_j) is defined by $\alpha_j = \boldsymbol{y}' \cdot \boldsymbol{n}_j$, $\beta_j = \boldsymbol{x}' \cdot (\boldsymbol{a}_j - \boldsymbol{a}_i)$,and $\gamma_j = \arctan(\boldsymbol{z}' \cdot \boldsymbol{n}_j / \boldsymbol{x}' \cdot \boldsymbol{n}_j)$.

By obtaining the angle features of the n feature points (A_1, A_2,⋯, A_n) around A_i, a set of angle features is defined. The set is referred to as the Simplified Point Feature Histograms of A_i (***SPFH***(A_i)). Here, n represents the number of voxels that exist within a radius of 2 m from the i-th voxel. Similarly, ***SPFH***(A_j) is calculated for the feature point A_j (j = 1, 2, ..., n). The 33-dimensional point feature histograms related to A_i (***FPFH***(A_i)) are then given by [9]

$$\boldsymbol{FPFH}(A_i) = \boldsymbol{SPFH}(A_i) + \frac{1}{n}\sum_{j=1}^{n} w_j \boldsymbol{SPFH}(A_j) \qquad \text{(C)}$$

where the weight w_i denotes the inverse number of the distance between A_i, and A_j.

REFERENCES

[1] J. Zhao et al., “Autonomous Driving System: A Comprehensive Survey,” Expert Systems with Applications 242, pp. 1–27, 2023.

[2] Y. Fu, C. Li, F. R. Yu, T. H. Luan, and Y. Zhang, “A Survey of Driving Safety with Sensing, Vehicular Communications, and Artificial Intelligence-Based Collision Avoidance,” IEEE Trans. on Intelligent Transportation Systems, Vol. 23, Issue 7, pp. 6142–6163, 2022.

[3] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, “Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving,” IEEE Trans. on Intelligent Vehicles, Vol. 2, Issue 3, pp. 194–220, 2017.

[4] E. Arnold, et al., “A Survey on 3D Object Detection Methods for Autonomous Driving Applications,” IEEE Trans. on Intelligent Transportation Systems, Vol. 20, Issue 10, pp. 3782–3795, 2019.

[5] Y. Li, et al., “A Review of Simultaneous Localization and Mapping Algorithms Based on Lidar,” World Electric Vehicle J., 2025, 16(2), pp. 1–29, 2025.

[6] Z. Wang, Y. Shen, B. Cai, and M. T. Saleem, “A Brief Review on Loop Closure Detection with 3D Point Cloud,” Proc. 2019 IEEE Int. Conf. on Real-time Computing and Robotics, pp. 929–934, 2019.

[7] F. Martín, R. Triebel, L. Moreno, and R. Siegwart, “Two Different Tools for Three-Dimensional Mapping: DE-based Scan Matching and Feature-Based Loop Detection,” Robotica, Vol. 32, pp. 19–41, 2017.

[8] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Automatic Appearance-Based Loop Detection from 3d Laser Data Using the Normal Distributions Transform," J. of Field Robotics, Vol. 26, Issue 11-12, pp. 892–914, 2009.

[9] R. B. Rusu, N. Blodow, and M. Beetz, “Fast Point Feature Histograms (FPFH) for 3D Registration,” Proc. IEEE Int. Conf. on Robotics and Automation, pp. 3212–3217, 2009.

[10] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, “Point Feature Extraction on 3D Range Scans Taking into Account Object Boundaries,” Proc. 2011 IEEE Int. Conf. on Robotics and Automation, pp. 1–8, 2011.

[11] S. Tanaka, C. Koshiro, M. Yamaji, M. Hashimoto, and K. Takahashi, “Point Cloud Mapping and Merging in GNSS-Denied and Dynamic Environments Using Only Onboard Scanning LiDAR,” Int. J. on Advances in Systems and Measurements, Vol. 13, No. 3&4, pp. 275–288, 2020.

[12] R. Nakamura, T. Kambe, M. Hashimoto, and K. Takahashi, “SLAM-based Mapping in Truck-and-Robot System for Last-Mile Delivery Automation,” Proc. the 2023 IARIA Annual Congress on Frontiers in Science, Technology, Services, and Applications, pp. 31–37, 2023.

[13] L. Wang and Y. Huang, “A Survey of 3D Point Cloud and Deep Learning-Based Approaches for Scene Understanding in Autonomous Driving,” IEEE Intelligent Transportation Systems Magazine, Vol. 14, Issue 6, pp. 135–154, 2022.

[14] S. Arshad and G. W. Kim, “Role of Deep Learning in Loop Closure Detection for Visual and Lidar SLAM: A Survey,” Sensors, 2021, 21, 1243, 2021.

[15] L. Li, X. Kong, X. Zhao, T. Huang, and Y. Liu, “Semantic Scan Context: A Novel Semantic-Based Loop-Closure Method for LiDAR SLAM,” Autonomous Robots, Vol. 46, pp. 535–551, 2022.

[16] Y. Fan, et al., “A Semantic-Based Loop Closure Detection of 3D Point Cloud,” Proc. 2021 IEEE Int. Conf. on Robotics and Biomimetics, 2021.

[17] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “RangeNet++: Fast and Accurate LiDAR Semantic Segmentation,” Proc. 2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2019.

[18] A. Jhaldiyal and N. Chaudhary, “Semantic Segmentation of 3D LiDAR Data Using Deep Learning: A Review of Projection-Based Methods,” Applied Intelligence, 2022.

[19] J. Behley et al., “SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences,” Proc. the IEEE/CVF Int. Conf. on Computer Vision (ICCV), pp. 9297–9307, 2019.

[20] R. Nakamura, I. Inaga, M. Hashimoto, and K. Takahashi, “SLAM-Based Mapping Using Micromobility-Mounted Solid-State LiDAR,” Proc. the 11th IIAE Int. Conf. on Intelligent Systems and Image Processing, pp. 93–100, 2024.

[21] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, “A Tutorial on Graph-Based SLAM,” IEEE Intelligent Transportation Systems Magazine, Vol. 2, Issue 4, pp. 31–43, 2010.