IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 10, 2022, accepted January 24, 2022, date of publication February 9, 2022, date of current version March 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3150387

Semi-Automatic Town-Scale 3D Mapping Using Building Information From Publicly Available Maps

SHUN NIIJIMA[1,2], (Member, IEEE), RYUSUKE UMEYAMA[1,2], YOKO SASAKI[2], AND HIROSHI MIZOGUCHI[1,2], (Member, IEEE)

[1]Department of Mechanical Engineering, Tokyo University of Science, Noda-shi, Chiba 278-8510, Japan
[2]National Institute of Advanced Industrial Science and Technology (AIST), Koto-ku, Tokyo 135-0064, Japan

Corresponding author: Shun Niijima (niijima-shun@aist.go.jp)

This work was supported in part by the JSPS KAKENHI under Grant 19J22783, and in part by the New Energy and Industrial Technology Development Organization (NEDO).

ABSTRACT The 3D maps are used for self-positioning estimation and path planning for the autonomous navigation of robots in urban areas. This paper presents a framework that generates globally consistent 3D maps from the pose graph of existing simultaneous localization and mapping (SLAM) methods. Our approach corrects a pose graph by performing a 3D alignment with building information on publicly available maps. The framework automatically finds an appropriate anchor pose for the alignment and optimizes the pose graph according to the constraints obtained by the alignment. However, there are situations where it is difficult to automate 3D alignment because of measurement errors as well as errors in publicly available maps. To minimize operational costs, the proposed framework incorporates a user interface (UI) that allows users to check the results of 3D map alignment and make simple corrections. The framework was evaluated by conducting 3D mapping experiments in an urban area in Japan, and 3D mapping was performed over a distance of approximately 15 kilometers. The experimental results showed that the framework could automatically select anchor poses with high probability and generate 3D city maps with an average of approximately five manual operations per km by the user. The accuracy of the 3D mapping was evaluated by comparing it with a manually corrected reference trajectory based on an accurate 3D map from a commercial mobile mapping system (MMS). The 3D maps had an average absolute position error of 5.5, which is the lowest error compared to the maps generated by other open source software (OSS) SLAM methods.

INDEX TERMS Global consistency 3D mapping, mobile robot, publicly available maps, urban area.

I. INTRODUCTION

Various mobile robots use a city 3D map as prior knowledge to perform autonomous navigation. In particular, 3D maps are useful for self-pose localization [1], [2] and path planning in mobile robot navigation [3], [4]. With accurate global 3D maps of urban areas it is possible for mobile robots to navigate autonomously in a wide city area. The goal of this study is generating globally consistent 3D maps for mobile robots operating in urban areas.

The simultaneous localization and mapping (SLAM) is often used for 3D mapping for mobile robots. It is to perform sequential scan matching on the data collected by a mobile robot's sensors. Numerous SLAM methods have been introduced in recent years [5]–[8]. The sequential scan matching results in cumulative position errors that increase with the number of observations. When mapping on a large scale, as in the case of a city, these cumulative errors increase, so that the maps produced by multiple observations do not overlap and cannot be used for navigation.

These errors can be corrected by pose graph optimization [9]. These method needs anchor poses that provides a global position and orientation as additional information for SLAM correction. Some of these methods include a loop closure function to obtain the anchor pose of returning to the same place. However, the loop closure function cannot correct the overall distortion and must include a loop in the driving trajectory. The accurate absolute coordinates

The associate editor coordinating the review of this manuscript and approving it for publication was Shun-Feng Su.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

are required to obtain accurate anchor poses. The global navigation satellite system (GNSS) is often used to acquire an absolute coordinate system. This system suffers from a few system errors owing to the loss of signal and multipath propagation, especially in urban areas.

This paper presents a framework for semi-automatic globally consistent 3D mapping for mobile robots. The proposed method is aimed at generating 3D maps in urban areas with many buildings where it is difficult to obtain GNSS. Our approach corrects 3D maps using the 3D alignment of the scanned data to a publicly available map. The resulting 3D alignment is added as a constraint to the pose graph and is subjected to an optimization process. However, the 3D alignment fails because of the errors between the measurements and the publicly available maps. An example of such error is an instance when the wall of a building measured by the sensor is not in publicly available maps. Such failures greatly reduce the accuracy of 3D maps. Although it is difficult to avoid such failures automatically, they are easily visible to the user. To reduce the operational cost, our framework automatically extracts anchor pose candidates for alignment and has a user interface (UI) to confirm or modify the 3D alignment results. The user can check and fix the anchor pose from the anchor pose candidates and perform pose-graph optimization by checking the alignment results on the user interface. The optimization eliminates cumulative errors and generates a globally consistent 3D map. Fig. 1 shows a globally consistent 3D map generated by the proposed framework. The 3D map (color point clouds) coincides with the publicly available maps (gray point clouds).

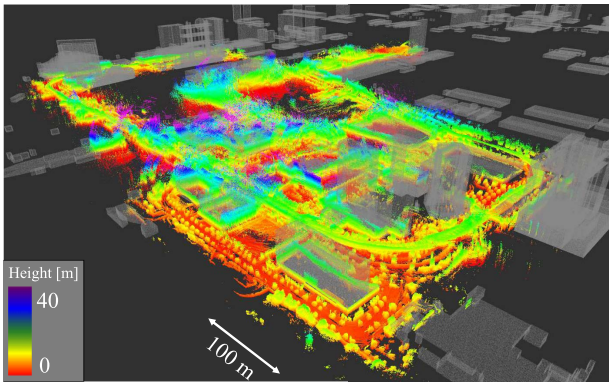


FIGURE 1. 3D map (color point clouds) that is globally consistent with buildings displayed on a publicly available map (gray point clouds).

II. RELATED WORK

The airborne laser scanning (ALS) [10] and mobile mapping systems (MMS) [11] are the major solutions for city mapping. The ALS technology has attracted significant interest in the field of mapping community mainly because of the increasing applications across a wide spectrum. An ALS survey is carried out by shining an active laser beam with a light detection and ranging (LiDAR) from a fixed-wing aircraft towards the ground. The integration with a GNSS and IMU determines the three-dimensional position of objects generating a cloud of points. It can generate a large-scale 3D model from scanning sequences in a single aerial survey. The ALS uses real-time kinematics (RTK) or post-processing kinematics (PPK) technique to observe a GNSS receiver positioned on a geodesic vertex in order to obtain good accuracy. The ALS cannot easily obtain information about objects on the ground that are under trees or along the facades of buildings.

The MMS measures roads using a car mounted with multiple sensors, such as lasers, cameras, and GNSS. The MMS can retrieve dense point clouds of ground objects along the roads. Because the point clouds are mainly around roads, the resulting 3D map will be sparse away from the road. For city robot navigation, mapping methods must meet the requirements of low cost and meet space restrictions.

The SLAM is often used to generate 3D maps for autonomous robot navigation. Recent SLAM methods such as LOAM [7] and Cartographer [8] can generate 3D maps using only LiDAR and inertial measurement unit (IMU) data. The errors accumulate during long observations and distort the resulting 3D map. It is difficult to guarantee the accuracy of 3D maps without access to absolute positions in a town-scale environment. Many studies have addressed these issues using the SLAM. The accumulated error is reduced by detecting loops using the Euclidean distance, feature points, or segmentation results [12]–[14]. Although these techniques reduce the local accumulated errors, they do not always have global consistency. For example, even if the loop of the trajectory is closed, the loop may be significantly distorted from the actual trajectory. Moreover, the driving route is restricted because it must include a loop.

To deal with these problems in urban areas, the use of publicly available maps has been proposed. The road network of publicly available maps has often been incorporated into SLAM for autonomous vehicles running on the road [15], [16]. The OpenStreetSLAM [17] uses publicly available maps to improve the accumulated error of visual odometry (VO) algorithms. The researchers used chamfer matching to align the detected path with the road network extracted from publicly available maps. [18] corrected the Monte Carlo filter localization results by alignment 3D laser range finder data with road network.

For mobile robots that move not only on roads but also on sidewalks and indoors, the building information is often used for correction. Hentschel *et al.* represented buildings as 2D line features. This line map was used to calculate the estimated locations of certain robots [19]. In [20], the accuracy of self-positioning was improved by using visual SLAM and building information. [21] improved the graph-based SLAM by using the 2D alignment of building information and a laser scanner. They used 2D iterative closest point-based (ICP-based) matching to align with the building information. Although these studies improved 2D SLAM for generating globally consistent maps in urban areas by using publicly available maps, they did not address the issue of 3D map

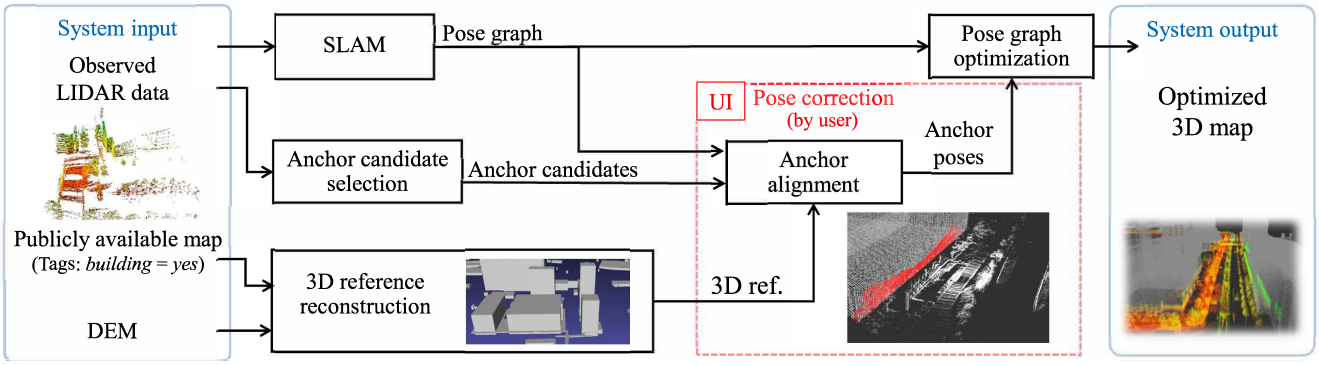


FIGURE 2. Overview of semi-automatic 3D mapping framework.

modification. [22] modifies the SLAM by alignment with building wall and 3D LiDAR data. This method is similar to our method, but it assumes that the 3D city model and sensor observations are accurate. The publicly available maps may contain errors. For instance, they may not include data on buildings that were constructed after the maps were produced. The pose graph optimization requires accurate anchor poses. If an anchor pose is substantially incorrect, the pose graph optimization will not converge.

To prevent anchor poses that contain such obvious errors from being selected, the proposed framework includes a UI that allows users to easily check and modify the 3D alignment. Our framework improves graph-based SLAM from the results of aligning 3D LiDAR data with the building walls of publicly available maps. We are able to construct 3D maps mainly from LiDAR observations in urban areas where many buildings are prone to GNSS inaccuracies.

III. SEMI-AUTOMATIC TOWN-SCALE 3D MAPPING

Our framework generates globally consistent city 3D maps that coincide with publicly available maps. The framework extracts anchor pose candidates automatically and performs pose graph optimization with the anchor poses, which are aligned with 3D building information of publicly available maps. It possesses a UI that can easily confirm and modify the alignment results to prevent the failure of pose graph optimization because of the errors in the publicly available maps. The UI guarantees the accuracy of the 3D map at minimal operational costs because it is easy for the user to correct obvious mistakes in the anchor pose alignment process.

An overview of the proposed 3D mapping framework is shown in Fig. 2. The UI was enclosed in a red box. First, a robot is driven through an urban area to obtain sensor data. The SLAM methods calculate the robot's trajectory at each time point in the form of a pose graph. Next, the anchor-candidate selection part determines the anchor candidates from the sensor scans. The anchor pose candidates are the anchor poses that perform the alignment with the walls of the building. The anchor alignment part then aligns the selected anchor poses with the building wall of the publicly available map using the ICP algorithm. The UI checks or modifies the alignment results such as the failure to detect walls or converge to a different place. The user can easily check and modify the anchor poses and execute an optimization in the UI. Finally, the 3D map is improved by pose graph optimization using the selected anchor poses and pose graph.

A. POSE GRAPH GENERATION

Many SLAM methods have been proposed for 3D mapping by mobile robots. The proposed framework can be used with any SLAM pose graph. In the SLAM process, a time series of the robot poses $\mathbf{X}$ is expressed as the nodes of a pose graph. The robot pose graph $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n, \cdots, \mathbf{x}_N\}$ has a state value vector $\mathbf{x}_n$ at time i. Each state value vector consists of a translation vector $\mathbf{t}_n = (t_{nx}, t_{ny}, t_{nz})$ and a rotational component consisting of a quaternion $\mathbf{q}_n = (q_{nx}, q_{ny}, q_{nz}, q_{nw})$.

B. 3D REFERENCE DATA ON PUBLICLY AVAILABLE MAP

As reference information for alignment, this module uses the building shape information from publicly available maps. Here, we used OpenStreetMap (OSM) [23] and the digital elevation model (DEM) as reference information.

The OSM is a free, editable map of the entire world that is released with an open-content license. The OSM data express points as *nodes* [latitude, longitude] and express *ways* as edges connecting nodes. The OSM data can be easily obtained and used through an overpass API[1]. The user can obtain building data by sending the query "tag: building=yes" via the API. The received data included the attributive tags and contours of buildings that were expressed in a closed way. A 3D mesh was constructed from the building data. The contours are mostly 2D outlines on the cartographic projection plane, but some contours possess 3D shapes of buildings. The building data also included tags that contained floor level and height. To represent the building as a 3D mesh, 3D contours were used directly, and 2D outlines were located either at the specified height or height from the floor level. In the OSM, the geodetic system is WGS84 and is represented by latitude and longitude data (EPSG:4326).

[1] https://overpass-turbo.eu/

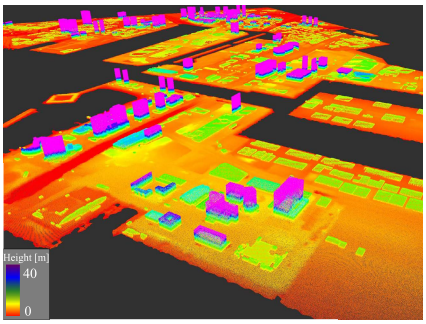


FIGURE 3. 3D reference information of publicly available maps depicted as 3D point clouds.

The DEM data is a data set that divides the ground surface into squares of equal spacing, and each square has an elevation value for the center point. The DEM is provided by the Geospatial Information Authority of Japan (GSI) [2]. The elevation information was provided through an API. [3] The elevation data form a dense point cloud and are used as the ground information. The point cloud of the ground is constructed by placing points at 0.1–m intervals on a delimited mesh. Fig.3 illustrates an example of reconstructed buildings and ground information of publicly available maps that are represented as 3 point clouds. For DEM point clouds, the geodetic system was JGD2011, and the projection was the planar orthogonal coordinate system 9 (EPSG:6677).

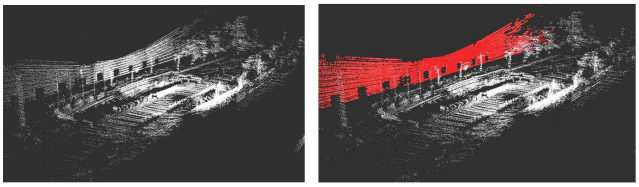
C. ANCHOR-CANDIDATE SELECTION

The anchor candidates are selected from the pose graph obtained by SLAM. This part evaluates the LiDAR scan at each time frame for selecting anchor candidates. The anchor pose candidates should include many vertical planes because the anchor pose is aligned to the walls of the building information.

The anchor pose candidate selection process performs plane detection on the scanned LiDAR point cloud at each time frame by using the RANSAC algorithm [24], as shown in Fig.4. Next, the anchor pose candidates that have a high ratio of detected plane points to the total number of observed points are selected. The anchor candidates were selected from frames with the highest ratios. The user can adjust the number of anchor candidates using two parameters: the plane ratio threshold r_{th} and frame interval h. The plane ratio threshold r_{th} determines the minimum plane ratio, whereas the frame interval determines the minimum number of frames between the anchor candidates. The plane ratio was set according to the number of buildings in the environment. It is necessary to set a higher value for environments with many buildings. The frame interval should be set according to the speed of the robot and the environment. If this interval is set to a low value, the robot will get many anchor candidates, but it will get multiple anchor candidates for the same wall of the building, which will require extra checking in the UI. The selected frames of the anchor candidates are then passed to the anchor-alignment UI.

[2]https://www.gsi.go.jp/ENGLISH/

[3] https://cyberjapandata2.gsi.go.jp/general/dem/scripts/getelevation.php



(a) Raw data (b) Plane detection

FIGURE 4. Example of building wall detection from LiDAR data: (a) raw point cloud; (b) plane detected in red point cloud.

D. ANCHOR ALIGNMENT IN THE UI

This process aims to align an anchor candidate with a publicly available map and discard alignment results with obvious mistakes. The anchor pose alignment and pose graph optimization are performed offline. Fig. 5 shows the screen of the UI, which displays the building information from publicly available maps, LiDAR data, pose graph, anchor candidates, and operation button. Fig. 6 shows an example of the use of UI for alignment. The process of alignment and optimization is shown in Algorithm 1.

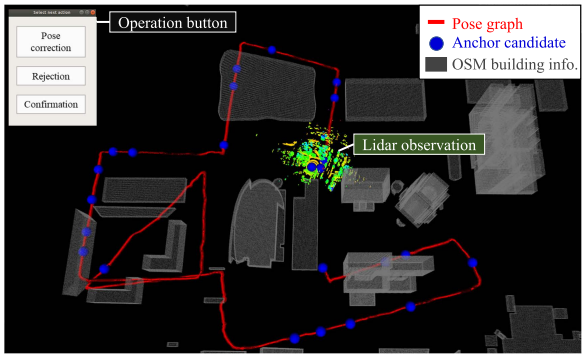
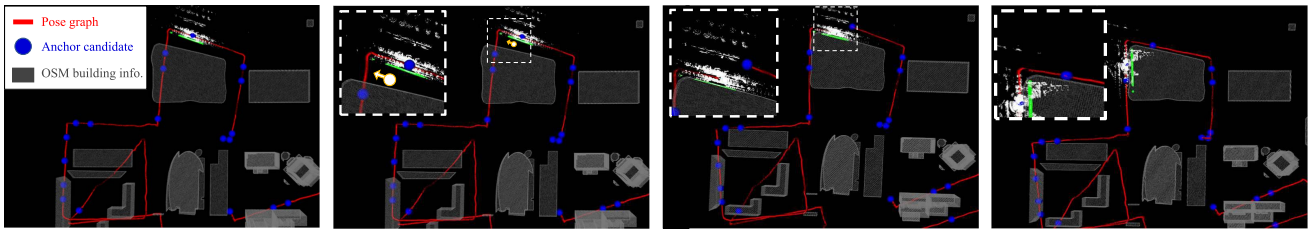


FIGURE 5. Screen of the UI.

The initial pose and LiDAR scan are shown in the UI. The user sets the start position and pose by referring to a publicly available map. It is fixed by ICP alignment with the 3D OSM model. The approximate initial pose can also be specified using the GNSS. In this study, we set the approximate pose using the mouse on the UI.

Next, the UI displays the robot's pose for the next anchor candidates, pose graph, LiDAR observation, and detected plane. The anchor pose aligns the detected plane with a 3D mesh of the building information using an ICP algorithm. The user can check the scanned data on the UI and observe if the data differ significantly from the publicly available maps or if the ICP alignment has failed. The user can decide what to do next by clicking on the three buttons "Confirmation," "Pose correction," and "Rejection"



(a) Checking current pose (b) User's pose correction (c) Automatic alignment with ICP (d) Pose graph optimization

FIGURE 6. Example of UI processing in anchor alignment: (a) shows a failure of alignment; (b) the user corrects the pose manually, where the orange arrow shows the setting pose; (c) Retry the ICP alignment; (d) The pose graph is corrected until the current pose is based on the anchor poses.

Algorithm 1 Process of Anchor Alignment Using the UI.

Input: pose graph: $\mathbf{X}$, anchor pose candidates: $\mathbf{Z}'$
Output: optimized pose graph: $\mathbf{X}^*$

1: anchor poses: $\mathbf{Z} = []$
2: $\mathbf{x}_0 \leftarrow$ set initial pose in the UI
3: **for** each $\mathbf{Z}'$ **do**
4: **while do**
5: $\mathbf{z}'_i \leftarrow$ align anchor pose with OSM using ICP
6: check alignment result and push the button in the UI
7: **if** button = *Pose correction* **then**
8: $\mathbf{z}'_i \leftarrow$ correct anchor pose in the UI
9: **end if**
10: **if** button = *Rejection* **then**
11: **break**
12: **end if**
13: **if** button = *Confirmation* **then**
14: $\mathbf{Z} \leftarrow \mathbf{z}'_i$ add anchor poses
15: $PoseGraphOptimization(\mathbf{X}, \mathbf{Z}) \rightarrow \mathbf{X}^*$
16: $\mathbf{X}^* = \mathbf{X}$
17: **break**
18: **end if**
19: **end while**
20: **end for**
21: **return** $\mathbf{X}^*$

When the anchor pose was located at a long distance from the building, as shown in Fig. 6 (a), the user pushes the ''Pose correction button.'' This process can correct the current anchor pose manually in the UI (Algorithm 1 line7-9). The user can freely specify the 6DoF pose of the anchor. In the UI, the user can modify the approximate position of the anchor by clicking and dragging with the mouse, as shown by the orange arrow in Figure 6(b). If more detailed coordinates are needed, the user can freely specify the 6DoF pose by setting the values with the keyboard. The UI retries the alignment using the initial pose. The UI displays the results of ICP alignment. Fig. 6 (c) shows the observed LiDAR data and anchor pose after the ICP alignment.

When the selected anchor pose candidate is unsuitable for optimization, the user pushes the ''Rejection'' button. The unsuitable anchor pose candidates are the ones wherein (a) the wall surface of the 3D buildings is not included in the LiDAR scan or (b) the wall surface of the 3D buildings in the LiDAR scan is not registered in the OSM. By excluding the alignment results with obvious mistakes, the accuracy of the overall pose-graph optimization process can be guaranteed(Algorithm 1 line10-12).

When the alignment result is correct, the user pushes the ''Confirm'' button, which adds the current anchor pose candidate to the anchor poses and performs graph-based optimization(Algorithm refalg1 line13-18). The optimization process is described in section III. Fig. 6 (d) shows that the pose graph is corrected by optimization process.

These process are repeated until the last anchor pose candidate has been processed. Here, we used a robot operating system visualization (Rviz) [4] for the implementation of the UI.

E. POSE GRAPH OPTIMIZATION

The pose graph consists of nodes and edges. The nodes represent robot poses. A node $\mathbf{x}_i$ represents the state value vector in the pose graph $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_i, \mathbf{x}_j, \cdots, \mathbf{x}_N\}$ obtained from the SLAM. An edge between two nodes $\mathbf{x}_i$, $\mathbf{x}_j$ represents a constraint $\mathbf{c}_{ij}$ that transforms $\mathbf{x}_i$ to $\mathbf{x}_j$. The constraint is calculated by $\ominus$, which is a standard motion composition operator [25].

$$\mathbf{c}_{ij} = \mathbf{x}_i \ominus \mathbf{x}_j. \tag{1}$$

The error function $\mathbf{e}(\mathbf{x})_{ij}$ is calculated from the difference between the constraint $\mathbf{c}_{ij}$ and the transformation of the one-step optimized pose graph$\hat{\mathbf{c}}_{ij}$:

$$\mathbf{e}(\mathbf{x})_{ij} = \mathbf{c}_{ij} - \hat{\mathbf{c}}_{ij}. \tag{2}$$

As the error functions are typically nonlinear, we linearize $\mathbf{e}(\mathbf{x})_{ij}$ around the current best estimate:

$$\mathbf{e}(\mathbf{x} + \Delta x)_{ij} = \mathbf{e}(\mathbf{x})_{ij} + J_{ij}\Delta x, \tag{3}$$

where J_{ij} represents the Jacobiaan matrix. The goal of the optimization step is to find a configuration of nodes $\mathbf{X}^*$ such that the overall graph is maximally consistent with the observations:

$$\mathbf{X}^* = \underset{\mathbf{X}}{argmin} \left(\sum_{ij}^{N} \mathbf{e}_{ij}(\mathbf{x})^{\mathrm{T}} \, \Omega_{ij} \, \mathbf{e}_{ij}(\mathbf{x}) \right). \tag{4}$$

[4] http://wiki.ros.org/rviz

Here, Ω_{ij} represents the information matrix of a constraint, which relaxes the differences in the space of the state value, such as the one between the translation and rotation vectors. Two static diagonal information matrices are used: straight Ω_{ij}^{st} and curved Ω_{ij}^{cv}. The SLAM tends to have possess fewer errors in the translational direction when the robot travels in a straight line and more errors in the rotational direction when the robot is in the middle of a curve. Thus, the value of the information matrix to be used is determined from the change in the pose Δrot_{ij}

$$\Omega_{ij} = \begin{cases} \Omega_{ij}^{st} & (\Delta rot_{ij} < rot_{thre}) \\ \Omega_{ij}^{cv} & (\Delta rot_{ij} \geq rot_{thre}), \end{cases} \tag{5}$$

where rot_thre is the threshold for detecting the curve from the attitude change. These values should be set according to the type of mobile robot used. If the mobile robot is moving in a complex way, the value should be high, and if it is moving in a linear manner, the value should be low.

The constraints of the anchor pose list $\mathbf{Z}^{map}$ were added to the SLAM pose graph optimization. The error function $\mathbf{e}_k^{map}(\mathbf{x})$ of the anchor pose is expressed as

$$\mathbf{e}_k^{map}(\mathbf{x}) = \mathbf{z}_k^{map} - \hat{\mathbf{z}}_k. \tag{6}$$

where $\hat{\mathbf{z}}_k$ represents the one-step optimized pose $\mathbf{x}_i$, as seen from the origin. The error function owing to the anchor pose is expressed as follows:

$$\mathbf{F}^{map}(\mathbf{X}) = \sum_{k}^{K} \mathbf{e}_k^{map}(\mathbf{x})^{\mathrm{T}} \, \Omega_k^m \, \mathbf{e}_k^{map}(\mathbf{x}). \tag{7}$$

The aim of the optimization problem is to find a configuration $\mathbf{X}^*$ of nodes that minimizes the sum of the two error functions:

$$\mathbf{X}^* = \underset{\mathbf{X}}{argmin} \left(\sum_{ij}^{N} \mathbf{e}_{ij}(\mathbf{x})^{\mathrm{T}} \, \Omega_{ij} \, \mathbf{e}_{ij}(\mathbf{x}) + \mathbf{F}^{map}(\mathbf{X}) \right). \tag{8}$$

The optimization process repeats and updates $\mathbf{X}^*$ until the end of the anchor pose. We implemented this optimization process with p2o [26], which is a compact and portable implementation for solving graph optimization problems.

IV. EVALUATION

Our framework was evaluated by 3D mapping in an urban area. The results of the 3D maps are shown below after showing the experimental setting. Anchor pose selection was evaluated by calculating the precision and recall. The total number of pose graphs and the number of anchor poses indicate that the optimization process is possible with a sufficiently small number of operations. Finally, the pose graphs were evaluated by calculating the absolute position error against the reference trajectory and by comparing the pose graph obtained by our framework with those of other open source software (OSS) SLAM methods.

A. EXPERIMENTAL SETTING

Fig. 7 shows the mobile robot (Whill Model CR) for collecting the data. Table 1 shows the key specification of used sensors. The robot mounted LiDAR (Velodyne VLP-16) and IMU (Xsens MTi-300 AHRS) sensors. The used Lidar scanner measures 360° of 3D information using 16 lasers. As the number of scanners increased, more distant walls could be detected. With this VLP-16, walls can be detected at a distance of approximately 20 meters. The collected 3D point clouds were not filtered and corrected only by the IMU data.

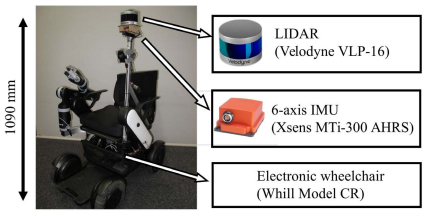


FIGURE 7. Robot and sensors.

TABLE 1. Key specifications of the LiDAR and IMU sensors.

LiDAR (Velodyne VLP-16)	
Observed cycle	15 [Hz]
Channel	16
Measurement Range	100 [m]
Field of View(vertical)	30 [deg]
6-axis IMU (Xsens MTi-300 AHRS)	
Observed cycle	200 [Hz]
Roll, pitch	0.2 [deg RMS]
Yaw	1.0 [deg RMS]

The experiments were performed in Aomi, Koto-ku, Tokyo, Japan. The running routes are shown in Fig. 8. The routes were on the sidewalk and a pedestrian crossing, and the average speed of the mobile robot was 1.6 m/s, including stops for up to approximately 100 s because of the halt at traffic signals. The total running time and distance were 3 h 0 min 36 secs and 14.36 km. The details of each route are summarized in Table 2.

The Cartographer, which is a state-of-the-art method, was used to obtain the pose graph. To compare the pose graph with those of the existing methods, the 3D maps were generated using the OSS SLAM methods, including normal distribution transform (NDT) mapping, LeGO-LOAM, and Cartographer. NDT mapping is implemented in the autoware project[5], which is an offline process and does not include a loop closure feature. The LeGO-LOAM and Cartographer run online and include a loop closure feature. The Cartographers also include a map optimizer function that operates on the

[5]https://github.com/Autoware-AI/autoware.ai

TABLE 2. Experimental run data.

Route	Distance [km]	Run time
1	0.99	11 min 53 sec
2	1.13	19 min 02 sec
3	2.54	29 min 04 sec
4	2.87	33 min 19 sec
5	3.03	39 min 26 sec
6	3.80	47 min 52 sec
Total	14.36	3 hour 0 min 36 sec

TABLE 3. Information matrix of optimization.

Information matrix	diagonal component
$\mathbf{\Omega}_{ij}^{st}$	[1.5, 1.5, 1.5, 4.0, 4.0, 4.0]
$\mathbf{\Omega}_{ij}^{cv}$	[1.0, 1.0, 1.0, 3.5, 3.5, 3.5]
$\mathbf{\Omega}_{ij}^{m}$	[1.0, 1.0, 1.0, 4.0, 4.0, 4.0]

submaps. Table 3 summarizes the parameters of the static information matrix in the pose graph optimization. In particular, the values represent the diagonal components of the matrix. Because our mobile robot travels straight on a paved road, the rotation threshold rot_{thre} was set to 10 °/s.

To evaluate the error of the optimized trajectory, the absolute position error (APE) was calculated from the reference trajectory. The APE is based on two absolute relative poses: the reference pose $x_{ref,n}$ and the estimated pose $x_{est,n}$ at timestamp n:

$$\begin{aligned}\mathbf{E}_n &= \mathbf{x}_{est,n} \ominus \mathbf{x}_{ref,n} \\ &= \mathbf{M}(\mathbf{x}_{ref,n})^{-1} \cdot \mathbf{M}(\mathbf{x}_{est,n}) \in \mathrm{SE}(3) \qquad (9) \\ APE &= ||\mathbf{E}_n - \mathbf{I}_{4\times4}||_F, \qquad (10)\end{aligned}$$

where M is a 4×4-transform matrix and $\mathbf{I}_{4\times4}$ is unit matrix. The APE value represents the magnitude of the distortion caused by the cumulative error of the SLAM. We used the evo [27] tool, which is a Python package for evaluating odometry and SLAM. The reference path was based on a 3D map built with a commercial MMS and optimized by manually setting anchor poses every 500 frames of LiDAR data. Manual anchor poses were recorded using the proposed UI tool. The reference path represents the trajectory in the ideal case when enough anchors are obtained at regular intervals. [6]

B. MAPPING RESULTS

The 3D mapping was performed for each route. Fig. 9 shows the final results. The colored point clouds are illustrated on the grey OSM building map. Each color is consistent with the line colors shown in Fig. 8. Because the optimized maps have a common coordinate system on the OSM, all 3D maps fit closely with other trials. Each 3D map has global consistency and does not witness significant distortions. Fig. 9 shows enlarged views of the optimized 3D maps in each area surrounded by the white squares in Fig. 9. The optimized 3D maps correspond to building information from the OSM. The shapes of the trees and sidewalks were

[6]The data set is released on github:
https://github.com/aistairc/City3DmapData

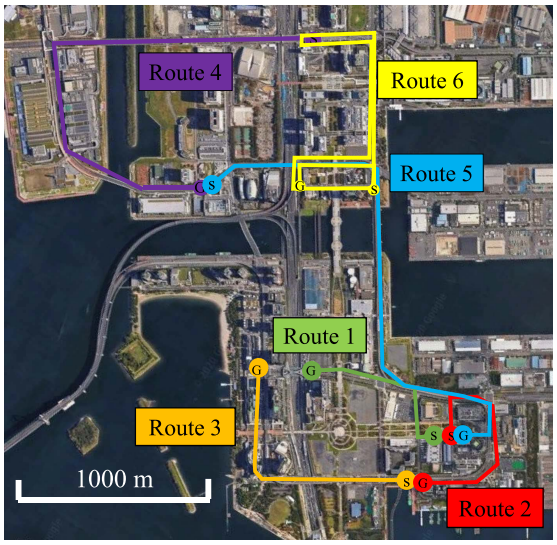
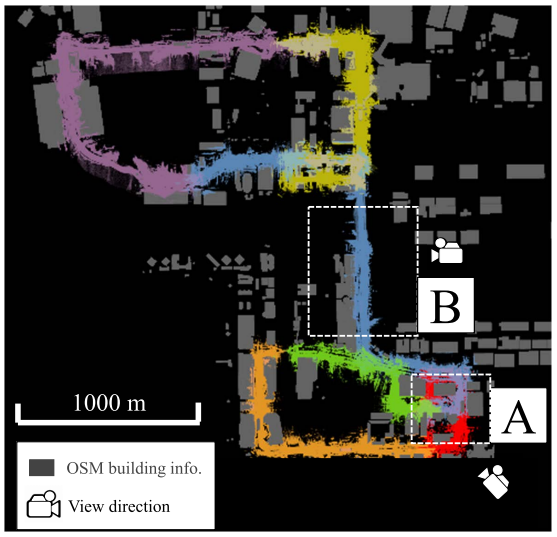
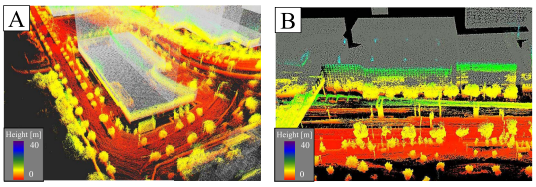


FIGURE 8. Running routes for collecting data.



(a) Birds-eye view of 3D maps



(b) Oblique view in area A (c) Oblique view in area B

FIGURE 9. Generated 3D maps generated by modifying cartographer with the proposed framework.

TABLE 4. Total number of selected anchor pose candidates and user operations.

pose graph	candidates	user operation (average [/km])	
		pose correction	rejection
62425	114	33 (2)	40 (3)

correct. This indicates that the 3D map is also locally correct and can be used for robot navigation.

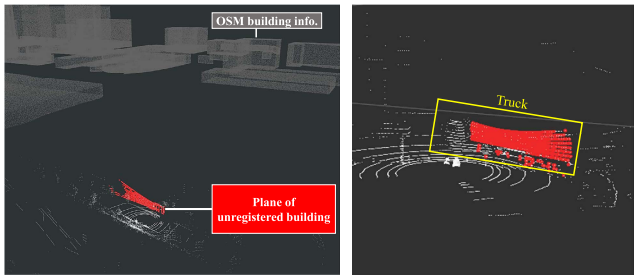
Table 4 shows the number of anchor pose candidates selected by our method and the number of user operations. These results reflect the number of manual corrections and rejections that were made. Our method selected 114 anchor pose candidates from a total of 62424 pose graphs. The number of anchor pose candidates that the user had to confirm was small enough for the size of the overall pose graph. Additionally, our method correctly selected scans containing many planes as anchor pose candidates. The user made 33 pose corrections to the selected anchor pose candidates and rejected 40 of them. The pose correction was necessary when the SLAM accumulation error was large and the ICP alignment did not converge. The user corrected the anchor pose by performing the ICP again with a manually input initial position aligned with the wall, such as in Fig. 6(b). The rejections occurred when the detected plane was not registered in OSM (Fig. 10(a)) or when the detected plane was not a plane of a building (Fig. 10(b)). These pose candidates were eliminated manually. The average number of operations by the user is approximately five per kilometer: two pose corrections and three rejections. These results show that a globally consistent 3D map can be generated with a small number of user operations.

C. ANCHOR POSE SELECTION

This evaluation confirmed whether the anchor pose candidate could be used for alignment. We used a pose graph from the same cartographer that we used for mapping. The proposed method is compared with the following methods: The equal division method (baseline) arranges the anchor pose candidates at equal intervals in the pose graph; the number of intervals in this case is the same as that in the proposed method. The other method is the ''beside-buildings'' method. This method simply selects the representative points of the anchor pose that pass through the side of the building as the anchor pose candidates. The representative anchor pose is the nearest to the center of the building. Table 5 lists the

TABLE 5. Results of anchor-pose candidate selection.

	equally divided	beside buildings	proposed
precision	0.37	0.52	0.86
recall	0.31	0.48	0.73
F measure	0.34	0.49	0.79



(a) Detected plane was not registered in OSM (b) False detection of building wall as side of truck

FIGURE 10. Examples of anchor pose candidates rejected by user.

precision, recall, and F-measure for all six routes. The walls to be detected are the ones within 20 m of the reference trajectory. The anchor pose can see these walls were used as ground truth, and each method was compared. It is up to the user to determine if the estimation result is incorrect. The proposed method achieves the highest results. In particular, because the proposed method considers whether the scan information includes the wall surface, the anchor pose can be estimated with higher results than in the beside-buildings method, which simply uses the pose graph at the side of the wall surface. Fig. 11 shows the trajectory before and after optimization. The light blue arrow represents the selected anchor poses, and the yellow line represents the building wall near the reference trajectory to be detected. The results show that the proposed method can select an effective anchor pose near the wall surface.

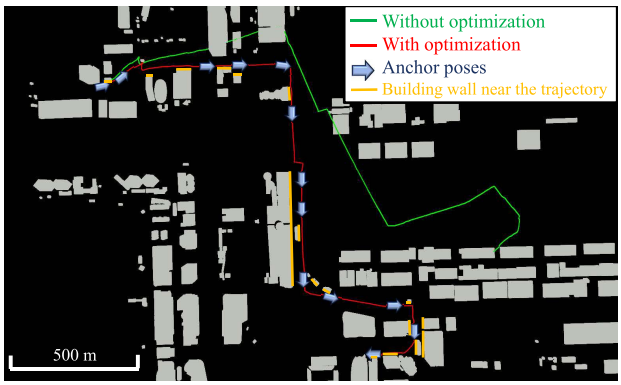


FIGURE 11. The modified pose graph in route 5 with selected anchor poses.

D. POSE GRAPH OPTIMIZATION

Fig. 12 summarizes the RMSE and maximum APEs (the number in parentheses represents the maximum value). The proposed framework had a lower cumulative error on all routes than those of the other methods. It produced a mapping within a certain error range regardless of the mileage or presence or absence of loops on the route. The average RMSE was 5.51, and the maximum value was 33.1. The average error, converted to a Euclidean distance, was 0.7 m, which

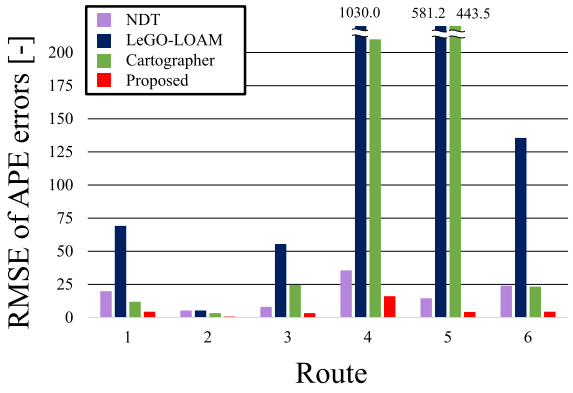
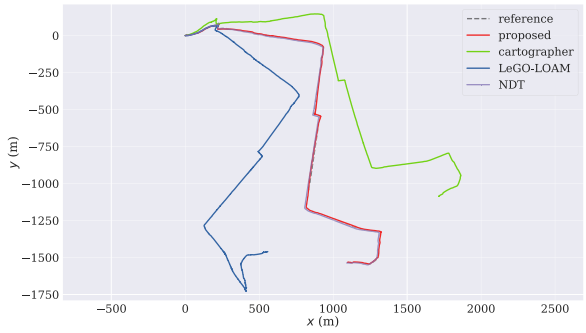
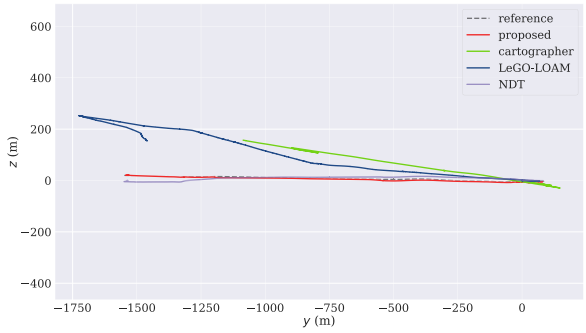


FIGURE 12. RMSE and (maximum value) of APE errors.



(a) top view

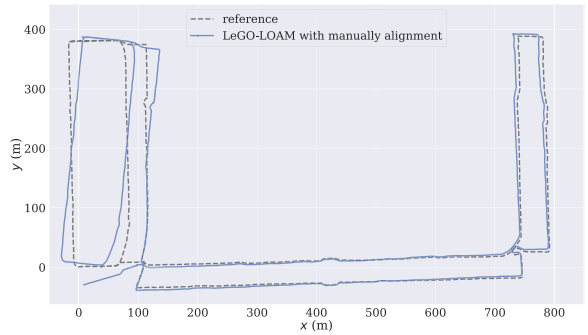


(b) longitudinal profile

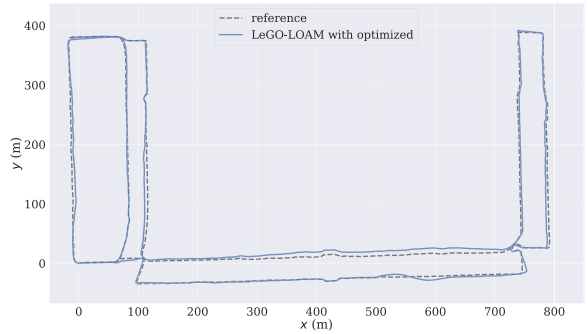
FIGURE 13. Comparison of pose graphs in route 5 generated with OSS SLAM methods.

indicates that our framework is sufficiently accurate and can use publicly available maps for the navigation task.

Fig. 13 shows the pose graphs for route 5; Fig. 13 (a) is a top view of the pose graphs and Fig. 13 (b) is a longitudinal profile. The other SLAMs show several deviations in their final positions because of a few degrees of deviation in the curve position. The NDT SLAM seems to overlap the trajectory of the reference in the top view, but the longitudinal profile shows that it is greatly deviated in the vertical direction. When the observed angle of incidence is large, the depth measured by LiDAR can be biased. As a result, when the mobile robot is moving on the ground surface, the observed



(a) Trajectory of LeGO-LOAM with manual alignment



(b) Trajectory of LeGO-LOAM optimized by proposed framework

FIGURE 14. Comparison of pose graphs in route 6 generated by proposed method and LeGO-LOAM method.

point from the ground may be slightly bent, and the estimated trajectory may drift along the vertical direction. The figure shows that the proposed optimization reduces these drifts along the vertical direction. These trends were similar for all the routes. These results of optimized pose graph are also included in the released data.

Fig.14(a) also shows the manual alignment of the LeGO-LOAM trajectory. Manual alignment means that the whole trajectory is manually translated or rotated so that the APE is minimized. Although the overall error of the LeGO-LOAM pose graph is reduced by the loop closure, the entire trajectory is distorted, so it does not overlap with the reference trajectory in manual alignment. This is an example where the loop closure is successful but the result does not correspond to the published map. Fig.14(b) shows the trajectory optimized by the proposed method. In the proposed method, the distortion of the whole trajectory is corrected by the anchor pose, so that the trajectory is consistent with the reference trajectory. This result shows that the proposed method can correct the distortion of the whole trajectory, which cannot be corrected by the conventional loop-closing function.

V. CONCLUSION

This paper proposes a semi-automatic 3D mapping framework for generating globally consistent 3D maps. The proposed framework uses the wall information of the building

as constraints and the 3D alignment results of the LiDAR observations, and thus, the existing SLAM pose graphs can be modified offline. To reduce the operational costs, it automatically extracts anchor pose candidates for alignment and has a user interface to confirm and modify the alignment results. The proposed framework was evaluated by generating 3D maps in a real urban environment in Japan. The experimental results show that 3D maps of the urban area with approximately five operations per kilometer were generated. The proposed method modifies the pose graph of the cartographer, which is a state-of-the-art SLAM method, to reduce the absolute position error to approximately 5.5, on average. These values were minimal compared to those of other OSS SLAMs, such as NDT and LeGO-LOAM.

Our future work will include the implementation of a town-scale autonomous navigation using globally consistent 3D maps. The globally consistent maps can be added to the information on publicly available maps, such as semantic maps. The publicly available maps have useful information for navigation, such as pointing out sidewalks and pedestrian crossings. The meaningful 3D maps will enable autonomous navigation. In this study, we did not deal with the information matrix of publicly available maps. The accuracy of the alignment depends on the shape of the surrounding buildings. To enhance the accuracy of 3D maps, it is necessary to design an information matrix that considers these error sources.

REFERENCES

[1] A. Ohsato, Y. Sasaki, and H. Mizoguchi, "Real-time 6DoF localization for a mobile robot using pre-computed 3D laser likelihood field," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Zhuhai, China, Dec. 2015, pp. 2359–2364.

[2] R. Hasegawa, Y. Sasaki, H. Takemura, and N. Uyama, "6-DoF Monte-Carlo localization for hand-held applications based on state vector verification," in *Proc. IEEE SENSORS*, Glasgow, U.K., Oct. 2017, pp. 1–3.

[3] S. Niijima, Y. Sasaki, and H. Mizoguchi, "Autonomous navigation of electric wheelchairs in urban areas on the basis of self-generated 2D drivable maps," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Auckland, New Zealand, Jul. 2018, pp. 1081–1086.

[4] S. Niijima, Y. Sasaki, and H. Mizoguchi, "Real-time autonomous navigation of an electric wheelchair in large-scale urban area with 3D map," *Adv. Robot.*, vol. 33, pp. 1–13, Jul. 2019.

[5] F. Moosmann and C. Stiller, "Velodyne SLAM," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Baden-Baden, Germany, Jun. 2011, pp. 393–398.

[6] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Porto, Portugal, Apr. 2018, pp. 287–296.

[7] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot. Sci. Syst.*, Berkeley, CA, USA, 2014, pp. 109–111.

[8] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Stockholm, Sweden, May 2016, pp. 1271–1278.

[9] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, Nov. 2010.

[10] A. Wehr and U. Lohr, "Airborne laser scanning an introduction and overview," *ISPRS J. Photogramm. Remote Sens.*, vol. 54, nos. 2–3, pp. 68–82, 1999.

[11] G. Hunter, C. Cox, and J. Kremer, "Development of a commercial laser scanning mobile mapping system–streetmapper," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, vol. 36, pp. 17–18, Jan. 2006.

[12] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robots*, vol. 4, no. 4, pp. 333–349, Oct. 1997.

[13] J. Sprickerhof, A. Nüchter, K. Lingemann, and J. Hertzberg, "A heuristic loop closing technique for large-scale 6D SLAM," *Automatika*, vol. 52, no. 3, pp. 199–222, Jan. 2011.

[14] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based place recognition in 3D point clouds," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Singapore, May 2017, pp. 5266–5272.

[15] P. Fleischmann, T. Pfister, M. Oswald, and K. Berns, "Using OpenStreetMap for autonomous mobile robot navigation," *Adv. Intell. Syst. Comput.*, vol. 531, pp. 883–895, Feb. 2017.

[16] S. Niijima, J. Nitta, Y. Sasaki, and H. Mizoguchi, "Generating 3D fundamental map by large-scale SLAM and graph-based optimization focused on road center line," in *Proc. 26th IEEE Int. Symp. Robot Hum. Interact. Commun. (RO-MAN)*, Lisbon, Portugal, Aug. 2017, pp. 1188–1193.

[17] G. Floros, B. van der Zander, and B. Leibe, "OpenStreetSLAM: Global vehicle localization using OpenStreetMaps," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Karlsruhe, Germany, May 2013, pp. 1054–1059.

[18] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard, "Localization on OpenStreetMap data using a 3D laser scanner," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Seattle, WA, USA, May 2015, pp. 5260–5265.

[19] M. Hentschel and B. Wagner, "Autonomous robot navigation based on OpenStreetMap geodata," in *Proc. 13th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Madeira Island, Portuga, Sep. 2010, pp. 1645–1650.

[20] R. Liu, J. Zhang, S. Chen, T. Yang, and C. Arth, "Accurate real-time visual SLAM combining building models and GPS for mobile robot," *J. Real-Time Image Process.*, vol. 18, no. 2, pp. 419–429, Apr. 2021.

[21] O. Vysotska and C. Stachniss, "Improving SLAM by exploiting building information from publicly available maps and localization priors," *PFG-J. Photogramm., Remote Sens. Geoinf. Sci.*, vol. 85, no. 1, pp. 53–65, Feb. 2017.

[22] L. Lucks, L. Klingbeil, L. Plümer, and Y. Dehbi, "Improving trajectory estimation using 3D city models and kinematic point clouds," *Trans. GIS*, vol. 25, no. 1, pp. 238–260, Feb. 2021.

[23] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

[24] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[25] R. Smith, M. Self, and P. Cheeseman, *Estimating Uncertain Spatial Relationships in Robotics*. New York, NY, USA: Springer, 1990, pp. 167–193.

[26] K. Irie and M. Tomono, "A compact and portable implementation of graph-based SLAM," in *Proc. Robot. Mechatronics Conf.*, Fukushima, Japan, May 2017, p. 2P2–B01.

[27] M. Grupp. (2017). *EVO: Python Package for the Evaluation of Odometry and SLAM*. [Online]. Available: https://github.com/MichaelGrupp/evo

SHUN NIIJIMA (Member, IEEE) received the M.S. degree in mechanical engineering from the Tokyo University of Science, Chiba, Japan, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Mechanical Engineering. His research interests include robot navigation, field robotics, and SLAM. He is a member of the RSJ and JSME. He is also a Research Fellow (DC1) at the Japan Society for the Promotion of Science. He was a recipient of the Young Researcher Fellow of the Japan Society of Mechanical Engineers, in 2019.

RYUSUKE UMEYAMA received the B.S. degree in mechanical engineering from the Tokyo University of Science, Chiba, Japan, in 2020, where he is currently pursuing the M.S. degree with the Department of Mechanical Engineering. His research interests include mobile robots and the SLAM. He is a member of the RSJ.

YOKO SASAKI received the Ph.D. degree in mechanical engineering from the Tokyo University of Science, in 2009. Her research in Ph.D. degree is on auditory systems for mobile robots. She joined the Digital Human Research Center, National Institute of Advanced Industrial Science and Technology (AIST). She is currently a Senior Researcher at the Artificial Intelligent Research Center, AIST. Her research interests include the development of autonomous mobile robots and sensing and navigation technology, especially in home environments.

HIROSHI MIZOGUCHI (Member, IEEE) received the D.E. degree from The University of Tokyo, in 1985. From 1985 to 1994, he was at the Research and Development Center, Toshiba Corporation. From 1994 to 1997, he worked at the RCAST, The University of Tokyo. In 1996, he had a chance to stay at the Professor Roberto Cipolla's Laboratory, Department of Engineering, Cambridge University, U.K., as an Academic Visitor supported by a Monbusho Grant. From 1997 to 2002, he was an Associate Professor with the Department of Information and Computer Science, Faculty of Engineering, Saitama University. He is currently a Professor with the Department of Mechanical Engineering, Faculty of Science and Technology, Tokyo University of Science. His research interest includes systems of human interfaces in the real world. He is a member of the Robotics Society of Japan, the Japan Society of Mechanical Engineers, and the IEEE Robotics and Automation Society.