Interactive 3D Graph SLAM for Map Correction

Kenji Koide, Jun Miura, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno

***Abstract*—This letter presents an interactive graph SLAM framework with a 3D LIDAR. This framework allows the user to interactively correct a 3D environmental map generated by an automatic SLAM system. By optimizing a pose graph consisting of pose constraints created by the automatic SLAM and map correction constraints, which are created by the user through a graphical user interface, we obtain a large and globally consistent 3D environmental map. We propose semi-automatic loop closing and plane-based map correction techniques for creating map correction constraints. We also devise a pose constraint update approach to refine the pose constraints given by the automatic SLAM. The evaluation results demonstrate that the proposed system enables us to improve the consistency of mapping results and obtain a mapping accuracy that outperforms state-of-the-art automatic SLAM frameworks with minimal human effort.**

***Index Terms*—SLAM, Mapping.**

I. Introduction

ENVIRONMENTAL mapping is essential for many functions in intelligent mobile systems, such as localization, place recognition, and navigation. One way to create a three-dimensional (3D) environmental map is so-called simultaneous localization and mapping (SLAM) with a mobile 3D light detection and ranging sensor (LIDAR). This approach requires a relatively affordable sensor and is less time-consuming compared with methods based on an aerial camera or static high-definition LIDAR. Many SLAM frameworks have been proposed, and some of them are publicly available as open source codes [1]–[3]. However, large-scale consistent mapping is still challenging, even for state-of-the-art SLAM frameworks. They require careful tuning of hyper-parameters depending on the sensor and environment properties, which is difficult for, in particular, users who are not SLAM experts. Without good hyper-parameter selection, we often face mapping failures that result in local and global inconsistency of the map (e.g., doubled and bent walls and floors).

Global consistency of mapping results can be greatly improved by introducing prior knowledge of the environment, such as flat floor [4] or Manhattan world [5], [6] assumptions. However, such assumptions make the system applicable to only limited environments. Environment assumptions should be carefully and selectively added to the mapping process to retain generality.

In this letter, we propose a framework that allows us to interactively correct mapping failures and add prior information of the environment as necessary through a graphical user interface (GUI). To create a consistent environmental map with minimal human effort, the proposed framework optimizes a pose graph with pose constraints $\mathcal{C}^A$, which are created by an automatic SLAM, and map correction and prior knowledge constraints $\mathcal{C}^M$ created semi-automatically through the GUI. For creating $\mathcal{C}^M$, we propose semi-automatic loop closing and plane detection-based map correction techniques that enable us to efficiently improve the local and global consistency of mapping results. We also propose a pose constraint update mechanism to refine the given pose constraints $\mathcal{C}^A$.

The contribution of this letter is three-fold. First, we propose an interactive framework for map correction with a complete GUI. Second, we propose pose edge refinement and plane-based map correction techniques that enable us to improve the local and global consistency of an environmental map. Third, the source code for the proposed framework is available on a public repository.[1]

II. Related Work

A. Loop Detection and Closing

Loop closing is an essential technique in SLAM to correct odometry drift. By optimizing the sensor trajectory such that observations obtained in a common place become identical (i.e., closing the loop), we can correct accumulated odometry errors.

The most common way to find loops is to compare nearby keyframes and test if they satisfy a loop detection criterion (e.g., low matching error). While many SLAM frameworks employ this simple yet general approach [2], [4], [7], it becomes unfeasible when the trajectory deteriorates significantly as a result of the drift in the odometry data, making it difficult to extract relevant loop candidates.

Many loop detection methods to find loop candidates under deteriorated odometry data have been proposed. Many

Manuscript received June 18, 2020; accepted September 29, 2020. Date of publication October 5, 2020; date of current version October 16, 2020. This letter was recommended for publication by the Associate Editor A. Nuechter and S. Behnke upon evaluation of the Reviewers' comments. This work was supported in part by JSPS KAKENHI under Grant 18K18072 and in part by the Project Commissioned by the New Energy and Industrial Technology Development Organization (NEDO). *(Corresponding author: Kenji Koide.)*

Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno are with the Department of Information Technology and Human Factors, the National Institute of Advanced Industrial Science and Technology, Tsukuba 3050061, Ibaraki, Japan (e-mail: k.koide@aist.go.jp; yokotsuka-masashi@aist.go.jp; shuji.oishi@aist.go.jp; atsuhiko.banno@aist.go.jp).

Jun Miura is with the Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi, Aichi 4418580, Japan (e-mail: jun.miura@tut.jp).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3028828

[1]https://github.com/koide3/interactive_slam

2377-3766 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: FH Dortmund. Downloaded on November 21,2020 at 23:26:36 UTC from IEEE Xplore. Restrictions apply.

traditional studies proposed point cloud descriptors, such as distance [8] and local surface feature [9] histograms, to calculate the similarity between point clouds. ScanContext [10] is a recent point cloud descriptor that encodes the point height information in the polar grid coordinate. This representation enables quick loop candidate extraction with the ring key comparison technique and robust similarity calculation between point clouds. LiDAR-Iris proposed by Wang *et al.* [11] calculates a discriminative binary feature map from a ScanContext image using LoG-Gabor filters, and it uses a Fourier transform-based comparison method to avoid brute-force rotation evaluation. Although these methods have shown high loop detection accuracy in open and feature-rich places like outdoor urban environments, their accuracy can deteriorate significantly in featureless and smaller environments.

Recently, several deep convolutional neural network-based loop detectors have been proposed. They typically project point clouds into a 2D image form (e.g., spherical image [12], [13], histogram image [14], and ScanContext image [15]) and apply convolutional filters to extract discriminative features. Then, the following comparison network takes two point cloud features and outputs some metrics (e.g., relative pose [12] and overlap [13] between point clouds) to judge if they are a correct loop pair. While they show impressive performance in the environment and with the sensor that the network was trained on, they often require costly data collection and re-training to adapt the network to new environments and sensors.

Generality is still an open problem for hand-crafted and deep CNN-based loop detection methods, and it remains difficult to fully automatically find loops across different environments.

B. Manual Correction for SLAM

The human-in-the-loop (HitL) SLAM proposed by Nashed and Biswas [16] allows the user to interactively add 2D line (wall) constraints (collocation, colinear, and parallel) to the 2D pose graph generated by an automatic episodic non-Markov localization SLAM [17]. HitL SLAM provides a GUI to draw lines on a 2D map, and it associates the drawn lines with corresponding map points using an expectation and maximization algorithm. By optimizing the pose graph consisting of the constraints created automatically and manually, HitL SLAM constructs a large and consistent map. Milijas *et al.* proposed another pose graph-based interactive SLAM approach for 2D mapping [18]. Through a GUI, they manually inserted pose constraints in a pose graph created using Google Cartographer [3] and performed pose graph optimization to obtain a consistent mapping result. There are also a few 2D mapping frameworks that allow the user to manually manipulate submap nodes to correct mapping failures [19].

While these frameworks work well for 2D mapping, it is difficult for several reasons to efficiently correct 3D maps by directly extending them to 3D. First, the complexity of the problem significantly increases in 3D mapping. Although these frameworks provide basic map correction features (e.g., line and pose constraints), we need much more effort to correct 6 DoF trajectories using only these features. To effectively correct 3D mapping failures, it is important to have the assistance of semi-automatic features guided by a few manual user operations. Second, their problem formulation allows only for inserting new constraints but not for refining existing constraints. In 3D SLAM, odometry errors can be larger than in 2D SLAM, and such errors can have a negative effect on the final mapping result. It is thus necessary to eliminate measurement and estimation errors existing in the automatic mapping result by refining the existing constraints.

There are also several studies that takes the help of manual operations to efficiently perform map segmentation and semantic mapping [20]–[23]. They assume consistency in annotating the 2D map and do not consider correcting mapping failures.

III. Proposed Framework

A. System Overview

To obtain a consistent 3D environmental map, we optimize a parameter set $\mathcal{X}$ that includes sensor poses under constraints $\mathcal{C}^A = \{z_i^A|_{i=1,\dots,N}\}$ and $\mathcal{C}^M = \{z_i^M|_{i=1,\dots,M}\}$ created by, respectively, an automatic SLAM and the proposed framework for map refinement. The objective function is defined as follows:

$$F(\mathcal{X}) = E(\mathcal{C}^A) + E(\mathcal{C}^M), \tag{1}$$

$$E(\mathcal{C}) = \sum_{z_k \in \mathcal{C}} e_k\left(x_k, z_k\right)^T \Omega_k e_k(x_k, z_k), \tag{2}$$

where x_k, e_k, and Ω_k are the parameter block, error function, and information matrix, respectively, corresponding to a constraint z_k. We can efficiently optimize the parameter set $\mathcal{X}$ such that Eq. (1) is minimized by using the pose graph optimization approach [24].

Fig. 1 shows the workflow of the proposed framework. In this workflow, the user creates map correction constraints $\mathcal{C}^M$ and refines given constraints $\mathcal{C}^A$ using several automatic and semi-automatic functions on a GUI until they are satisfied with the mapping results. To improve local and global mapping consistency, we propose the following map correction functions:

1) Automatic/semi-automatic loop closing
2) Automatic pose constraint refinement
3) Semi-automatic plane-based constraints

Every time a correction is made by the user, the pose graph is optimized by the Levenberg-Marquardt optimizer [25] in g2o [24], a hyper-graph optimization library, and the updated map with the latest correction is displayed to the user immediately.

The proposed framework is built on top of the Robot Operating System (ROS) ecosystem, and it uses mapping data generated by any ROS SLAM framework as an initial pose graph. If the ROS package can output a pose graph, our framework directly uses that pose graph as an initial graph. Otherwise, it extracts keyframes at certain intervals from the odometry estimated by the SLAM and creates a strip of pose vertices as an initial graph by connecting consecutive keyframes.

Fig. 2 shows an example of maps generated before and after the map correction process. We used an odometry sequence generated by LeGO-LOAM [2] for this example. Although the

Authorized licensed use limited to: FH Dortmund. Downloaded on November 21,2020 at 23:26:36 UTC from IEEE Xplore. Restrictions apply.

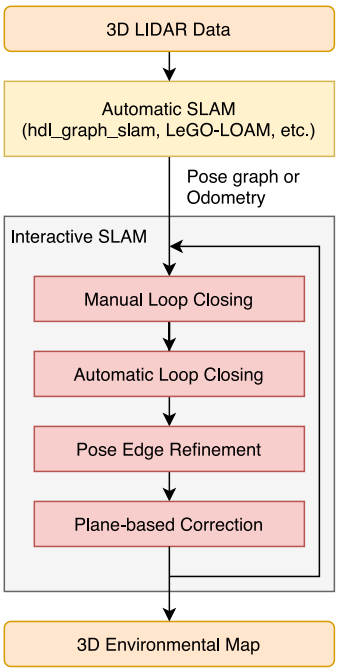
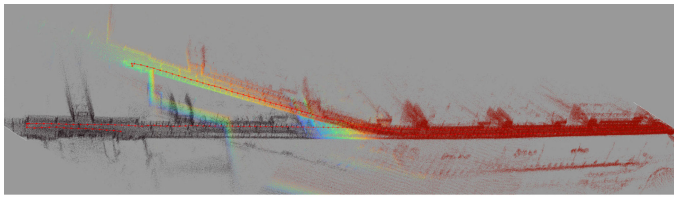
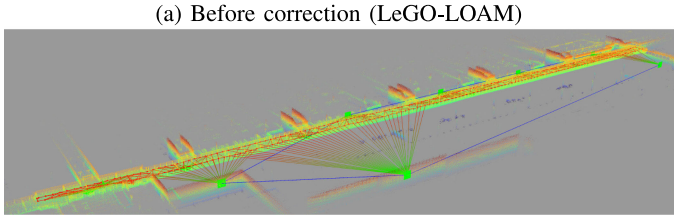


Fig. 1. Workflow of the proposed framework.



(a) Before correction (LeGO-LOAM)



(b) After correction

Fig. 2. 3D environmental maps generated before and after map correction. Point color encodes the altitude. The corridor is approximately 450 m in length.

initial map is largely bent and corrupted because of odometry errors, we can see that the map becomes flat and globally consistent after map correction. Fig. 3 shows the pose graph correction process. The red points and green planes respectively indicate pose and plane vertices, and the lines between the vertices represent relative pose constraint edges. In the following subsections, we explain the correction process in detail.

B. Loop Closing

We create loop constraints by letting the user explicitly select loop candidate keyframes through the GUI. Once the user selects two keyframes to be the beginning and end of a loop, we perform relative pose estimation using fast point feature histograms (FPFH) [8] to obtain an initial guess of the transformation between them. The GUI allows the user to inspect and fine-tune the initial guess, and then scan matching is applied to the keyframes for fine registration. The estimated relative pose is added to the pose graph as a loop constraint edge. In the proposed framework, ICP [26], NDT [27], and GICP [28] can be used for fine registration, and we use GICP in this letter. In Fig. 3(b), we can see that a manually added loop constraint greatly improves the global consistency of the map.

Once the sensor trajectory is roughly corrected by the manual loop closing, we can perform scan matching-based automatic loop closing. The automatic loop closing process is similar to that in [4]. We find a keyframe pair (k_i, k_j) that satisfies the following conditions, and we add a loop constraint between them:

$$\|k_i.p - k_j.p\| < th_{dist}, \tag{3}$$

$$\|path(k_i, k_j)\| > th_{path}, \tag{4}$$

$$matching\_score(k_i, k_j) < th_{score}, \tag{5}$$

where $k_i.p$ is the estimated position of the i-th keyframe, *path* is the minimum graph path length between the keyframes, *matching_score* is a metric to evaluate the scan matching result (e.g., sum of the distances between the corresponding points), and th_* are threshold values. The first condition extracts neighboring keyframes, while the second condition prevents loop constraints being created between consecutive frames and too many constraints being created in a small section. We apply a Huber robust kernel to each loop constraint to filter out wrong loop constraints.

Scan matching-based loop detection is inherently sensitive to the initial guess. In the context of interactive SLAM, however, keyframes are roughly aligned by the preceded manual loop closing, and we can expect a good initial guess for scan matching. Thus, this simple yet general loop closing approach works well in most cases.

In Fig. 3(c), loop constraints between neighboring keyframes are added by the automatic loop closing process. We can see that the distance between the outward and inward trajectories becomes consistent, and this result implies that local consistency is improved.

C. Pose Constraint Refinement

Scan matching odometry can be inaccurate when the sensor moves quickly or there are fewer features in the environment. We correct such errors in odometry by updating the pose constraints. We first extract a pose constraint that is inconsistent with its neighboring constraints. The inconsistency of a pose constraint is defined as the χ^2 distance:

$$\mathbf{e}_{i,j} = manifold(P_i^{-1} P_j R_{i,j}^{-1}), \tag{6}$$

$$error_{i,j} = \mathbf{e}_{i,j}^T \Omega_{i,j} \mathbf{e}_{i,j}, \tag{7}$$

where P_i and P_j are the estimated poses of the keyframes i and j, respectively; $R_{i,j}$ and $\Omega_{i,j}$ are the relative pose constraint and the information matrix assigned to the keyframes, respectively; and *manifold* is a function to convert an SE3 pose into

Authorized licensed use limited to: FH Dortmund. Downloaded on November 21,2020 at 23:26:36 UTC from IEEE Xplore. Restrictions apply.

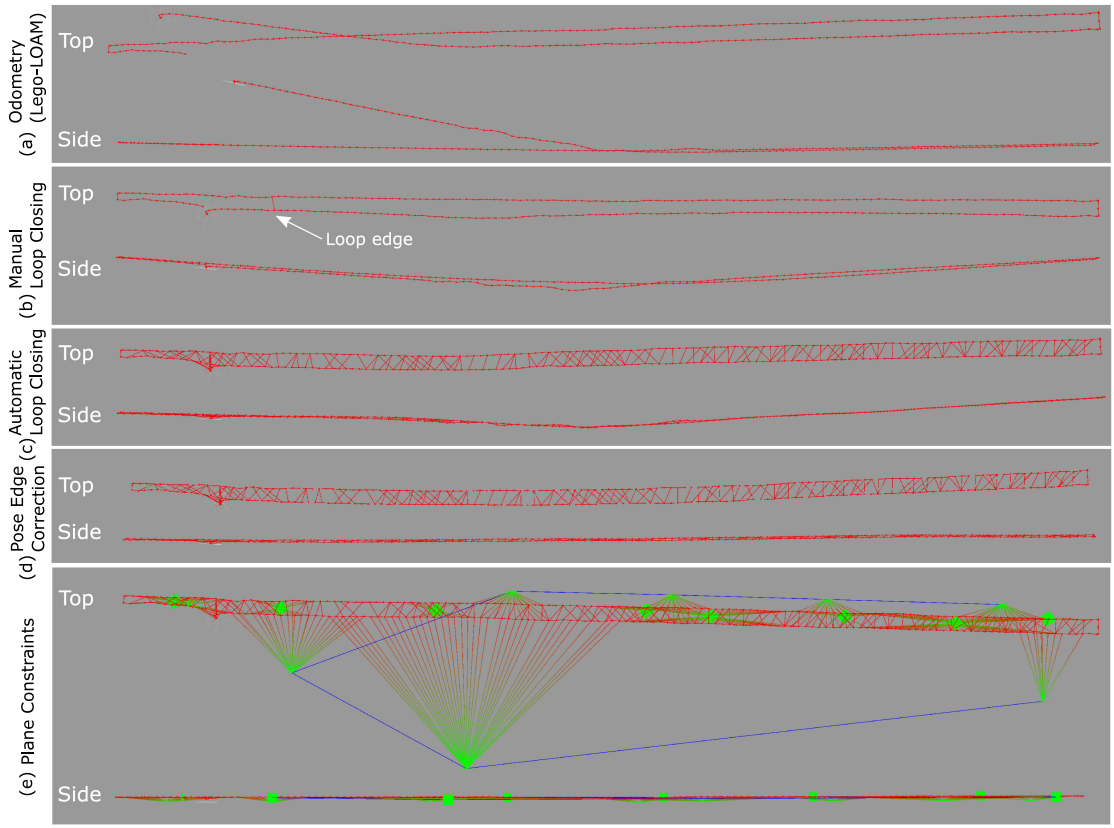


Fig. 3. Pose graph correction process (top to bottom).

a six dimensional parameter vector. We use the χ^2 distances as sampling weights and randomly sample a pose constraint for refinement.

We apply scan matching between the keyframes, which are connected by the sampled constraint, with the relative pose estimated by the graph optimization $P_i^{-1}P_j$ as an initial guess. The estimated relative pose would be close to the true relative pose because it is optimized with loop closing and other constraints. We can, thus, expect that the scan matching converges to a good estimate. If the new estimated relative pose improves the *matching_score* between the keyframes, we update the relative pose assigned to the constraint. Then, we perform pose graph optimization and repeat this pose refinement process.

In Fig. 3(c), the map is significantly bent as a result of odometry errors, and a large altitude error is observed. After the pose constraint refinement, the odometry errors are corrected, and the map becomes flat (see Fig. 3(d)). In this example, the χ^2 distance (the sum of the weighted constraint errors) of the pose graph decreased from 0.447 to 0.061 during the refinement process. This indicates that the consistency of the pose graph is improved.

D. Plane-Based Semi-Global Constraints

Although the loop closing and the pose constraint refinement improve the local consistency of a map, the global inconsistency, such as odometry bias, is difficult to correct. To improve global consistency, we exploit plane constraints. In environments where buildings can be observed, we can easily find parallel and perpendicular planes in the distance. We use the relationship between such distant planes as semi-global constraints to correct global inconsistencies in a map.

To insert a plane vertex into the pose graph, the user needs to pick a point on the plane using the GUI. We extract points with normals similar to the selected one using region growing [29], and we estimate the plane coefficients using a random sample consensus (RANSAC) plane detector [30]. Then, a new vertex representing the detected plane is added to the pose graph. Following the method in [31], we use a tuple of the azimuth angle ϕ, elevation angle ψ, and distance from the origin d as a compact representation of a plane $\pi = [n_x, n_y, n_z, d]^T = [\mathbf{n}, d]^T$ for optimization.

$$\tau(\pi) = [\phi, \psi, d] = \left[\tan^{-1}\left(\frac{n_y}{n_x}\right), \tan^{-1}\left(\frac{n_z}{\|\mathbf{n}\|}\right), d\right]. \quad (8)$$

Through the GUI, the user can choose two planes π_0 and π_1 among the detected ones and add an identity, parallel, or perpendicular constraint between them. The error functions of the plane constraints are defined as follows:

$$\textit{identity\_error}(\pi_0, \pi_1) = \pi_0 - \pi_1, \quad (9)$$

$$\textit{parallel\_error}(\pi_0, \pi_1) = \mathbf{n}_{\pi_0} - \mathbf{n}_{\pi_1}, \quad (10)$$

$$\textit{perpendicular\_error}(\pi_0, \pi_1) = \mathbf{n}_{\pi_0} \cdot \mathbf{n}_{\pi_1}. \quad (11)$$

Authorized licensed use limited to: FH Dortmund. Downloaded on November 21,2020 at 23:26:36 UTC from IEEE Xplore. Restrictions apply.

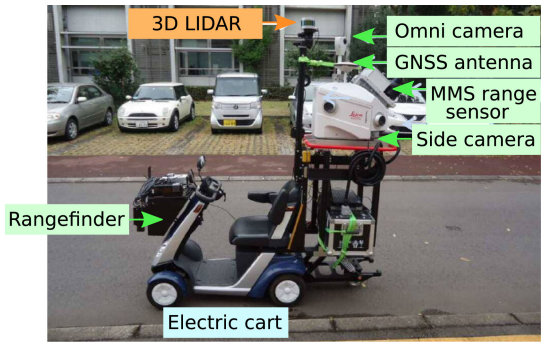


Fig. 4. Mobile mapping system.

If $\mathbf{n}_{\pi_0} \cdot \mathbf{n}_{\pi_1}$ is less than 0, this indicates that the directions of the plane normals are not matched, and we use a flipped plane $\pi_1' = -\pi_1$ for error calculation. We consider that the manually created plane constraints are a reliable information source and set a large information matrix (e.g., $10^3\mathbf{I}$) so that the pose graph optimization puts emphasis on these constraints.

In Fig. 3(e), several plane constraints placed between the floor plane and distant walls effectively correct the global error of the map.

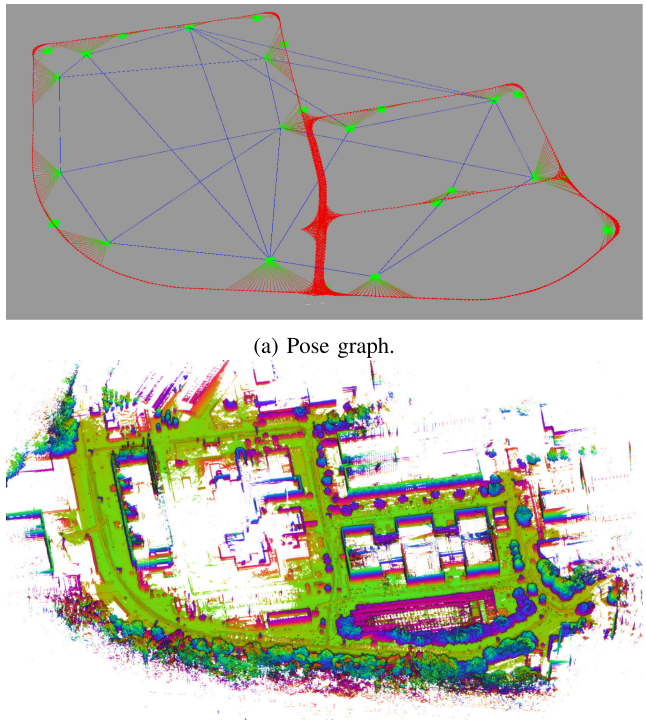
IV. Evaluation

A. Mapping in an Outdoor Environment

To demonstrate that the proposed framework allows us to create a large consistent environmental map, we evaluated it on a real LIDAR dataset.[2] For this evaluation, we recorded a 3D LIDAR data sequence with the Mobile Measurement System developed by the PASCO Corporation shown in Fig. 4. The measurement system consists of several static total stations placed in the environment and a mobile platform equipped with a 3D LIDAR, rangefinders, global navigation satellite system (GNSS), and cameras. It enables the measurement of the 3D LIDAR position to an accuracy of a few millimeters. We used the measured 3D LIDAR trajectory as the ground truth and evaluated the accuracy of the mapping results by comparing the estimated sensor trajectory with the ground truth. As evaluation metrics, we calculated absolute and relative trajectory errors (ATE and RTE) [32]. Note that, because the ground truth provides only the sensor translation but not orientation, we used a modified RTE calculation routine that aligns the sub-trajectory in an evaluation window and then calculates the errors between it and the ground truth.

To show that large-scale environmental mapping is difficult even for state-of-the-art automatic SLAM frameworks, we ran LOAM [1],[3] A-LOAM,[4] LeGO-LOAM [2], SuMa [7], ethzasl_icp_mapping [33], and hdl_graph_slam [4] on this dataset.

[2]The dataset is available at https://github.com/SMRT-AIST
[3]https://github.com/laboshinl/loam_velodyne
[4]https://github.com/HKUST-Aerial-Robotics/A-LOAM



(a) Pose graph.

(b) Environmental map. The color encodes the altitude.

Fig. 5. Generated pose graph and 3D environmental map. The environment is approximately 200 m × 400 m.

We extracted keyframes from the odometry sequence estimated by LOAM to generate an initial pose graph for the proposed framework. The keyframe interval was set to 5 m. Because the result of LOAM was already accurate, we manually added only one new loop constraint and then performed automatic loop closing. To further improve the global mapping accuracy, we applied edge refinement and then added 15 planes to the pose graph. The map correction process took approximately 15 minutes in total to correct the trajectory of the 20-minute dataset. Fig. 5 shows the pose graph and 3D map created with the proposed framework. We can see that a highly accurate and globally consistent map was obtained.

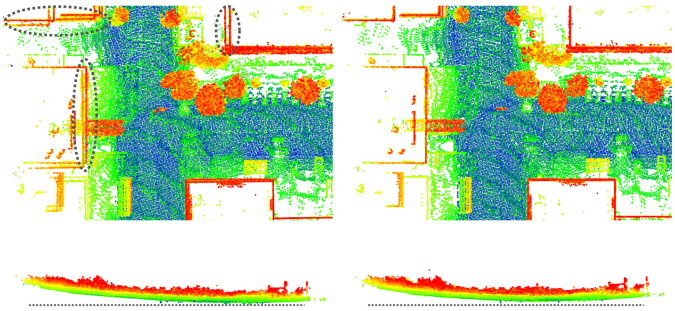
Table I shows the trajectory errors for the proposed framework and the automatic SLAM frameworks. While the RTEs of the automatic SLAM frameworks are small, they show large ATEs (1.731–3.074 m). It is worth mentioning that the ATE of LeGO-LOAM deteriorated after loop closing. Fig. 6 shows the mapping results of LeGO-LOAM with and without loop closing. From the top view, we can see that loop closing improved the local consistency of the map. However, the side view shows that the added loops slightly bent the map and increased the error in the altitude direction, resulting in the ATE deteriorating. This result suggests that loop closing is not always effective in correcting global errors in a trajectory, and we need global constraints such as plane constraints and gravity vector [3] constraints to improve the global accuracy of a map.

The proposed map correction functions effectively improved the global consistency of the base trajectory estimated by LOAM

Authorized licensed use limited to: FH Dortmund. Downloaded on November 21,2020 at 23:26:36 UTC from IEEE Xplore. Restrictions apply.

TABLE I
TRAJECTORY ERRORS

	Method	Loop	ATE [m]	RTE(5m) [m]	RTE(50m) [m]	RTE(500m) [m]
Baseline	ethzasl_icp_mapping [33]		5.248 ± 5.313	0.038 ± 0.094	0.433 ± 1.279	2.168 ± 2.113
	LOAM [1]		1.731 ± 1.015	0.045 ± 0.021	0.092 ± 0.060	0.701 ± 0.172
	A-LOAM		2.168 ± 1.284	0.043 ± 0.024	0.094 ± 0.051	0.820 ± 0.276
	LeGO-LOAM [2]		1.846 ± 1.112	0.027 ± 0.029	0.105 ± 0.141	0.680 ± 0.194
		✓	2.418 ± 1.490	0.028 ± 0.027	0.103 ± 0.143	0.859 ± 0.326
	SuMa [7]		4.352 ± 1.927	0.022 ± 0.016	0.155 ± 0.097	2.533 ± 0.549
		✓	3.074 ± 1.637	0.021 ± 0.016	0.123 ± 0.085	1.321 ± 0.239
	hdl_graph_slam [4]		4.018 ± 2.078	0.044 ± 0.034	0.389 ± 0.306	3.012 ± 0.782
		✓	1.797 ± 0.992	0.034 ± 0.044	0.206 ± 0.178	1.337 ± 0.443
Proposed	Base (LOAM)		1.731 ± 1.015	0.045 ± 0.021	0.092 ± 0.060	0.701 ± 0.172
	Loop closing	✓	1.683 ± 1.027	0.045 ± 0.022	0.091 ± 0.058	0.693 ± 0.174
	Edge refinement	✓	1.340 ± 0.826	0.044 ± 0.017	0.082 ± 0.043	0.590 ± 0.155
	Plane constraints	✓	0.517 ± 0.151	0.044 ± 0.017	0.081 ± 0.043	0.379 ± 0.034



(a) Without loop closing (b) With loop closing

Fig. 6. Top and side view of the mapping results of LeGO-LOAM. Top view is cropped for the best view. While local consistency has improved with loop closing (top view), the global altitude error has increased.

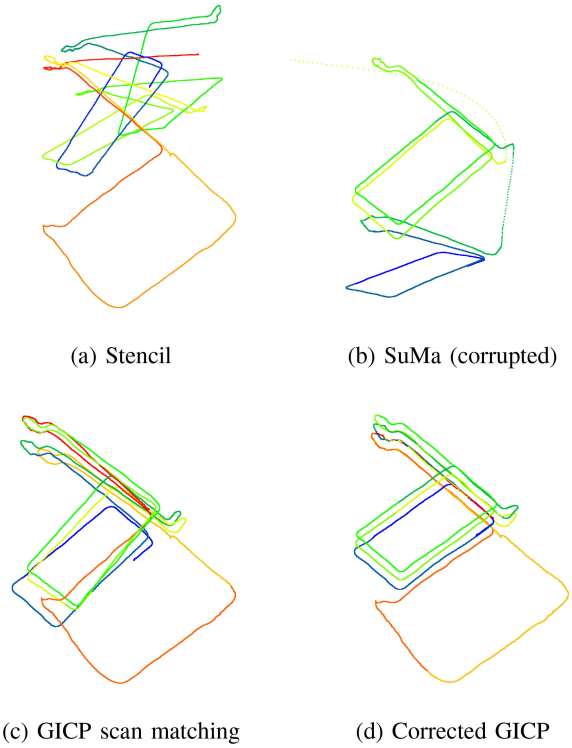
while retaining its good local consistency. After applying the loop closing, the ATE was decreased from 1.731 m to 1.683 m, and then the edge refinement process improved the ATE to 1.340 m. The best ATE we obtained after adding plane constraints outperforms the results of the automatic SLAM frameworks (0.517 m).

B. Mapping Across Indoor and Outdoor Environments

To emphasize the necessity of the mechanism to correct online mapping failures, we recorded another LIDAR sequence across indoor and outdoor environments. Such a situation, in which the sensor moves across very different environments, is extremely difficult for both the SLAM frontend and backend. The length of the sequence is approximately 20 minutes. We used Kaarta Stencil, a state-of-the-art commercial Visual-LIDAR-IMU mapping system, to record the dataset.

1) Frontend: Fig. 7(a) shows the trajectory estimated by Stencil. While the estimated trajectory in the outdoor environment is accurate, the trajectory estimation often became corrupted in the indoor environment. Even for the commercial SLAM system, it is still difficult to create a consistent map across different environments that require very different parameter settings for environment scale and structuredness changes.

We also ran the SLAM frameworks used in Section IV-A; however, all the methods yielded corrupted trajectories with



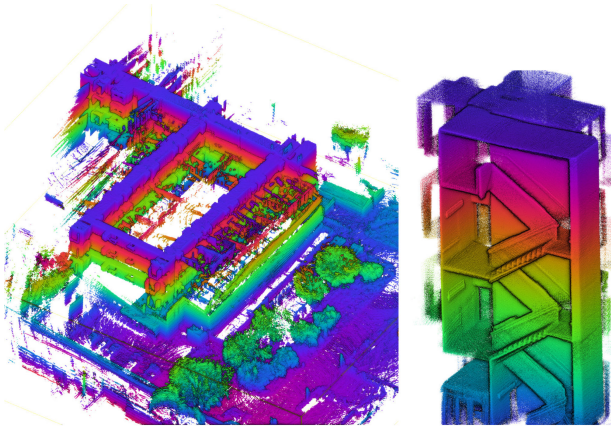
(a) Stencil (b) SuMa (corrupted)

(c) GICP scan matching (d) Corrected GICP

Fig. 7. Estimated trajectories in the indoor and outdoor environment.

large drift. In particular, the LOAM-variants (Stencil, LOAM, A-LOAM, LeGO-LOAM) tend to get corrupted at corners and staircases. Because their odometry estimation rely on the matching of plane and edge features, they can be unstable in featureless indoor environments. Once a wrong data association is made when turning a corner or climbing stairs, the odometry estimation becomes corrupted. SuMa, ethzasl_icp_mapping, and hdl_graph_slam take ICP-based approaches that use many points to estimate the odometry and thus are more robust for featureless indoor environments. However, the frame-to-model matching approach of SuMa and ethzasl_icp_mapping makes the estimation become totally corrupted (Fig. 7(b)) once the map model gets broken. Furthermore, they suffer from large drift because, in the indoor environment, the common view between frames is small, and estimation errors rapidly accumulate.

Authorized licensed use limited to: FH Dortmund. Downloaded on November 21,2020 at 23:26:36 UTC from IEEE Xplore. Restrictions apply.



(a) Generated environmental map (b) Staircase

Fig. 8. Mapping result in the indoor and outdoor environment.

2) Backend: To see how loop detection methods deteriorate in indoor environments, we excluded outdoor frames from the dataset and evaluated several loop detection methods (distance histogram-based method [9], ScanContext [10], LiDAR-Iris [11], and OverlapNet [13]) on the dataset. We also evaluated them on the outdoor dataset used in Section IV-A. The total number of frames were 332 and 519 in the indoor and the outdoor environments, respectively. For each frame combination, we calculated the overlap rate using the final mapping result, and a frame pair with an overlap rate larger than 30% was considered as loop beginning and end. To apply OverlapNet to the 16-line LIDAR data, we interpolated point cloud data on both the horizontal and vertical axes to generate 64-line LIDAR-like range images. Although the network may not show the best performance without re-training, we can see how its accuracy deteriorates in a new environment.

Fig. 9 shows ROC curves of the loop detection algorithms in the outdoor and indoor environments. We can see that, while they show high detection accuracy in the outdoor environment, all the algorithms are significantly degraded in the indoor environment.

There are two points that make loop detection difficult in indoor environments. First, the scale and structure of the indoor environment is significantly different from the outdoor environment. Fig. 10 shows ScanContext images in the outdoor and indoor environments. While the valid pixels are widely distributed in the ScanContext of the outdoor environment, in the indoor environment, most of the points fall in the inner rings of the ScanContext, and most of the pixels have no height information. This makes the ScanContext less discriminative in the indoor environment. Second, we often see highly repeated structures in indoor environments. In Fig. 11, red and green points respectively show correct loop beginning and end at a corner, and blue points show a wrong loop end at another floor. In this case, [9] shows very similar distances for the correct loop pair (red vs green: 6.89×10^{-3}) and the wrong loop pair (red vs blue: 6.83×10^{-3}). It is inherently difficult, if not impossible, to correctly find loops without accurate odometry in such environments.

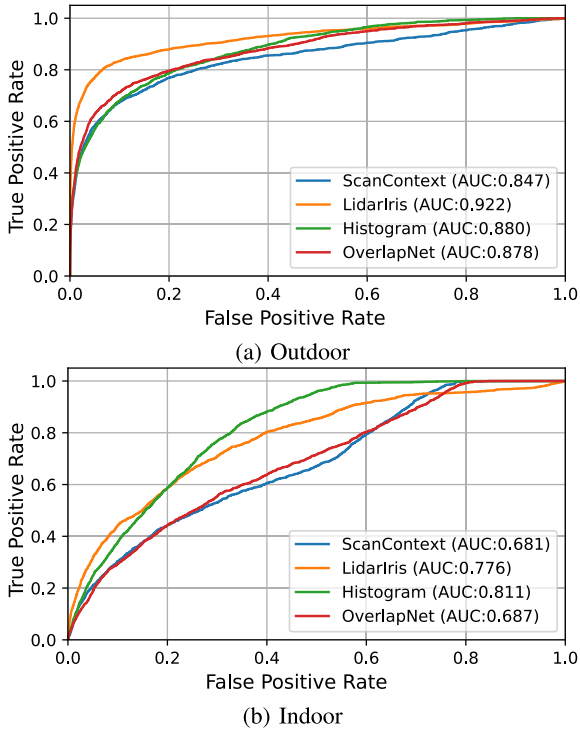


Fig. 9. ROC curves of loop detection methods in indoor and outdoor environments.

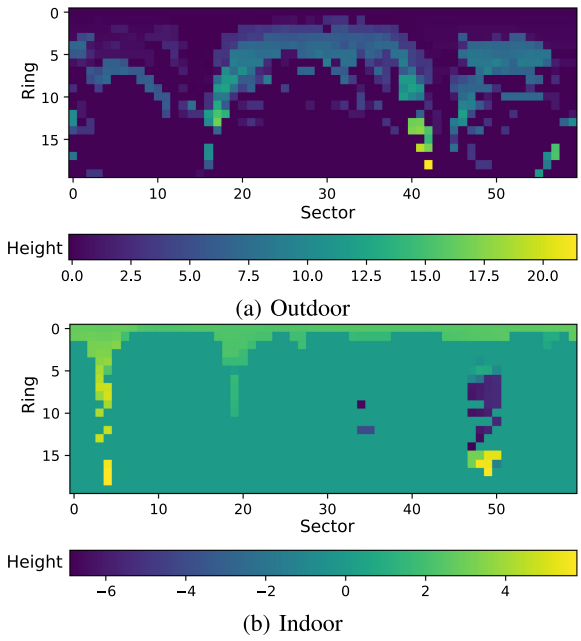


Fig. 10. ScanContext images in indoor and outdoor environments.

3) Proposed framework: We estimated the sensor odometry using GICP scan matching with conservative parameter settings (frame-by-frame matching and a larger matching distance threshold) and extracted keyframes for every 3 m of the trajectory. To correct the estimated trajectory, we manually added 19 loops and 17 planes to the pose graph and then performed

Authorized licensed use limited to: FH Dortmund. Downloaded on November 21,2020 at 23:26:36 UTC from IEEE Xplore. Restrictions apply.

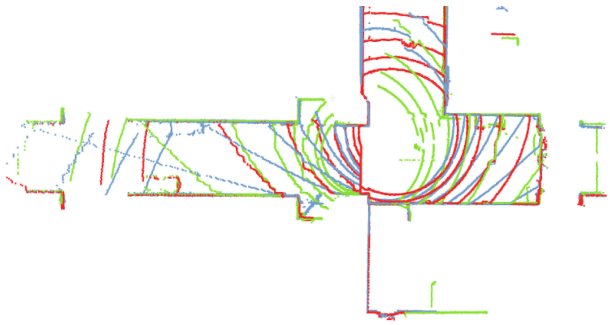


Fig. 11. Example of repeated environment structure in the indoor environment. Red: loop beginning, Green: correct loop end, Blue: wrong loop end at another floor.

automatic loop closing and pose edge refinement. The correction process took approximately 20 minutes. Fig. 8(a) and (b) respectively show the estimated trajectories before and after the correction. Fig. 7(d) and (f) show the generated environmental map with the proposed framework and the point cloud of a staircase in the environment, respectively. We can see that the generated map is consistent across multi-floor indoor and outdoor environments.

V. Conclusion

This letter proposed an interactive SLAM framework that allows the user to correct mapping failures of an automatic SLAM semi-automatically and improve the quality of a 3D environmental map. To correct a mapping result, the proposed framework optimizes a pose graph consisting of constraints created by the automatic SLAM and additional map correction constraints. We proposed loop closing and plane-based constraints to semi-automatically create map correction constraints. We also proposed an approach to refine the initial pose constraints. The evaluation results demonstrate that the proposed method enabled us to achieve a mapping quality that outperformed state-of-the-art automatic SLAM methods with minimal human effort.

References

[1] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst.* Robotics: Science and Systems Foundation, Jul. 2014.

[2] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 4758–4765.

[3] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d LIDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 1271–1278.

[4] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *Int. J. Adv. Robotic Syst.*, vol. 16, no. 2, Mar. 2019.

[5] P. Kim, B. Coltin, and H. J. Kim, "Linear RGB-d SLAM for planar environments," in *Proc. Eur. Conf. Comput. Vis.*, Springer, Aug. 2018, pp. 350–366.

[6] H. Li, J. Yao, J.-C. Bazin, X. Lu, Y. Xing, and K. Liu, "A monocular SLAM system leveraging structural regularity in manhattan world," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 2518–2525.

[7] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3d laser range data in urban environments," in *Proc. Robot.: Sci. Syst.*. Robotics: Science and Systems Foundation, Jun. 2018.

[8] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3d registration," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 3212–3217.

[9] T. Rohling, J. Mack, and D. Schulz, "A fast histogram-based similarity measure for detecting loop closures in 3-d LIDAR data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2015, pp. 736–741.

[10] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 4802–4809.

[11] Y. Wang, Z. Sun, C.-Z. Xu, S. Sarma, J. Yang, and H. Kong, "Lidar iris for loop-closure detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2020.

[12] L. Schaupp, M. Burki, R. Dube, R. Siegwart, and C. Cadena, "OREOS: Oriented recognition of 3d point clouds in outdoor scenarios," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2019, pp. 3255–3261.

[13] X. Chen *et al.*, "OverlapNet: Loop closing for LiDAR-based SLAM," in *Proc. Robot.: Sci. Syst.* Robotics: Science and Systems Foundation, Jul. 2020.

[14] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong, "LocNet: Global localization in 3d point clouds for mobile vehicles," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2018, pp. 728–733.

[15] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term LiDAR localization using scan context image," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1948–1955, Apr. 2019.

[16] S. B. Nashed and J. Biswas, "Human-in-the-loop slam," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2018.

[17] J. Biswas and M. M. Veloso, "Episodic non-markov localization," *Robot. Auton. Syst.*, vol. 87, Jan. 2017.

[18] R. Milijas, J. Orsulic, and S. Bogdan, "When measurements fail: Using an interactive SLAM solution to fight bad odometry," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, May 2020, pp. 1–6.

[19] S. Macenski, "On use of the slam toolbox: A fresh(er) look at mapping and localization for the dynamic world," ROSCon 2019, Oct. 2019.

[20] A. Diosi, G. Taylor, and L. Kleeman, "Interactive SLAM using laser and advanced sonar," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2005, pp. 1103–1108.

[21] C. Nieto-Granda, J. G. Rogers, A. J. B. Trevor, and H. I. Christensen, "Semantic map partitioning in indoor environments using regional analysis," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 1451–1456.

[22] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3515–3522.

[23] A. Sidaoui, M. K. Zein, I. H. Elhajj, and D. Asmar, "A-SLAM: Human in-the-loop augmented SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2019, pp. 5245–5251.

[24] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3607–3613.

[25] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math.*, vol. 2, no. 2, pp. 164–168, Jul. 1944.

[26] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Proc. Object Recognit. Supported User Interact. Service Robots*, Aug. 2002, pp. 545–548.

[27] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Jan. 2003, pp. 2743–2748.

[28] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot.: Sci. Syst.*. Robotics: Science and Systems Foundation, Jun. 2009.

[29] G. V. T. Rabbani and F.A. van den Heuvel, "Segmentation of point clouds using smoothness constraints," in *Proc. ISPRS Commision V Symp.*, Sep. 2006.

[30] M. A. Fischler and R. C. 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.

[31] L. Ma, C. Kerl, J. Stuckler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-d SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 1285–1291.

[32] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 7244–7251.

[33] F. Pomerleau, P. Krusi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3d map maintenance in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2014, pp. 3712–3719.

Authorized licensed use limited to: FH Dortmund. Downloaded on November 21,2020 at 23:26:36 UTC from IEEE Xplore. Restrictions apply.