NV-LIOM: LiDAR-Inertial Odometry and Mapping Using Normal Vectors Towards Robust SLAM in Multifloor Environments

Dongha Chung ⬙, *Member, IEEE*, and Jinwhan Kim ⬙, *Member, IEEE*

Abstract—Over the last few decades, numerous LiDAR-inertial odometry (LIO) algorithms have been developed, demonstrating satisfactory performance across diverse environments. Most of these algorithms have predominantly been validated in open outdoor environments; however, they often encounter challenges in confined indoor settings. In such indoor environments, reliable point cloud registration becomes problematic due to the rapid changes in LiDAR scans and repetitive structural features like walls and stairs, particularly in multifloor buildings. In this letter, we present NV-LIOM, a normal vector-based LiDAR-inertial odometry and mapping framework focused on robust point cloud registration designed for indoor multifloor environments. Our approach extracts the normal vectors from the LiDAR scans and utilizes them for correspondence search to enhance the point cloud registration performance. To ensure robust registration, the distribution of the normal vector directions is analyzed, and situations of degeneracy are examined to adjust the matching uncertainty. Additionally, a viewpoint-based loop closure module is implemented to avoid wrong correspondences that are blocked by the walls. The proposed method is tested through public datasets and our own dataset.

Index Terms—Localization, mapping, SLAM.

I. INTRODUCTION

THE field of mobile robotics has widened its operational environments through advances in sensors, actuators, and control algorithms. These advancements are reshaping the robotics landscape, extending their application from autonomous driving to a diverse array of environments, encompassing firefighting robots, delivery robots, and others that are required to overcome various obstacles in confined indoor environments. Reliable simultaneous localization and mapping (SLAM) techniques are required for these robots to navigate through unknown indoor environments.

LiDAR sensors directly measure the depth information of the surrounding environment, providing stable measurements regardless of the ambient brightness. Leveraging these capabilities, the 3-dimensional (3D) LiDARs are predominantly utilized for localization and mapping by registering consecutive LiDAR point clouds for mobile robotics applications. Over the last few decades, numerous public datasets for autonomous driving applications have been released, leading to significant advancements in LiDAR-inertial odometry algorithm technology through the development and validation of algorithms utilizing these datasets.

However, these algorithms are known to show degraded performance in indoor spaces and often fail significantly in multifloor environments [1], [2], [3]. Unlike outdoor environments, indoor environments are characterized by confined spaces and thin walls forming multiple segmented areas. In these areas, the scene captured by the LiDAR scans may change rapidly with repetitive structural elements such as walls and stairs. Due to these environmental factors, the existing algorithms frequently fail to find correct correspondences for point cloud registration. The nearest neighbor search may fail to provide correct correspondences in confined featureless spaces where the points are densely populated near the LiDAR [1], [2]. Additionally, it can result in incorrect correspondences between points on one side of a thin wall and those on the opposite side, which is also known as the double-sided issue [3].

In this letter, we present NV-LIOM, a normal vector-based LiDAR-inertial odometry and mapping framework focused on robust point cloud registration designed for indoor multifloor environments. Our approach introduces key front-end methods such as the use of points' normals for correspondence search, a degeneracy detection mechanism to handle challenging scenarios, and a view-based loop closure detection method to prevent incorrect correspondences. The main contributions of this letter can be summarized as follows:

- To address degeneracy situations in point cloud registration, especially in long corridors or stairwell scenarios, a degeneracy detection algorithm and corresponding registration uncertainty covariance matrix calculation method are proposed.
- We thoroughly tested the proposed algorithm in various datasets, including assessments on public datasets and our own collected dataset, as shown in Fig. 1. The validation
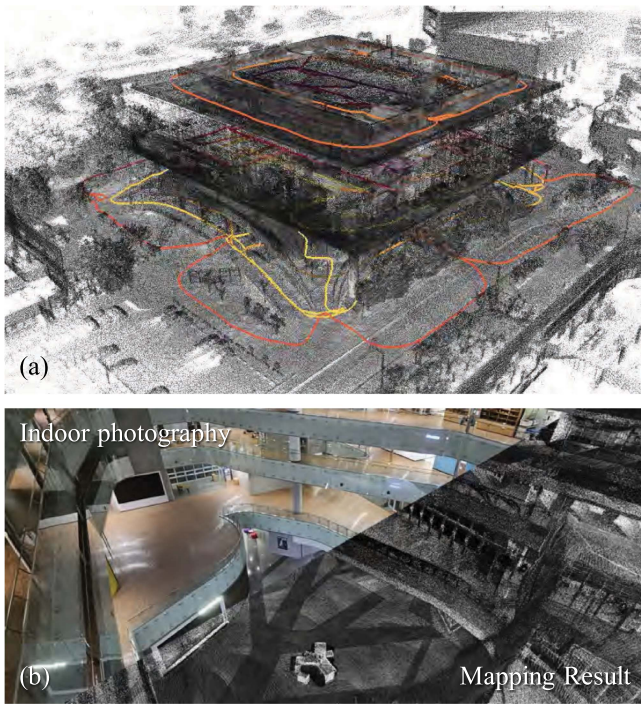
Fig. 1. Mapping result of the KI building, a five-story building at KAIST, using NV-LIOM. (a) Shows the overall mapping result and (b) shows a comparison between an indoor photo and the mapping result.

results demonstrate the effectiveness and versatility of the proposed approach.

II. RELATED WORK

The key component of LiDAR (-inertial) odometry relies on the effectiveness of point cloud registration. To handle the computational load of processing tens of thousands of points with each scan, various methods have been developed to reduce computation time. Feature-based methods extract key features from point clouds to perform matching, while direct methods involve down-sampling the point cloud and using techniques such as iterative closest point (ICP) [4] or generalized-ICP (GICP) [5].

Among feature-based algorithms, LOAM [6] utilizes 2D LiDAR scans with nodding motion, extracting corner points and surface points based on the relationships between adjacent points. LIO-SAM [7] employs a similar feature extraction method but introduces tightly-coupled LIO with IMU pre-integration [8], demonstrating high performance.

Leveraging its speed advantage, FAST-LIO2 [9] introduced a direct matching method using ikd-tree [10] for fast nearest neighbor search, enabling efficient scan-to-map matching instead of traditional scan-to-scan matching, resulting in more globally consistent performance in wide environments. Faster-LIO [11], on the other hand, replaces the tree structure of FAST-LIO2 with an incremental voxelization (iVox) system for even faster computational speeds.

While these direct methods exhibit excellent performance in outdoor environments, it is observed that they often fail in narrow indoor environments where the point clouds are densely packed in proximity. The fast computations of these algorithms rely on the down-sampling of the point clouds, which may fail to preserve meaningful points for reliable registration in cramped indoor environments. To tackle this issue, adaptive methods that adjust the parameters based on the analysis of LiDAR scans have been proposed. Based on Faster-LIO, AdaLIO [1] adaptively changes the voxelization size for down-sampling, search radius, and residual margin for plane selection by analyzing the given LiDAR scan for stable registration performance in degenerate cases. In [2] and DLIO [12], adaptive keyframe insertion methods are proposed to balance efficient computational time and reliable cloud registration. These methods insert keyframes based on scan analysis, resulting in dense keyframes in narrow spaces and sparse keyframes in wide areas.

When using only the nearest neighbor for correspondence search in registration, incorrect correspondences can occur in indoor environments, leading to inaccurate registration results. To address this issue, methods have been introduced that extract normal vectors from the LiDAR point cloud and perform correspondence searches considering the directions of these vectors. SuMa [13] employs a two-step process where the LiDAR point cloud is first projected into a depth image and normals are then extracted to form surfels. SLAM is conducted through matching between the surfel map and measured surfels, with surfel updates occurring based on the matching results. To expedite correspondence search, a rendering technique is utilized, projecting the surfel map onto the current frame to find correspondences in the image. LIPS [14] leverages the prevalence of planes in indoor scenarios, such as offices. It extracts planar primitives from LiDAR scans and utilizes a technique that matches these primitives. In [3], the approach begins by extracting the normal vector for each point and using clustering to identify planes. A method called forward ICP flow is introduced, utilizing the point-to-plane distance to find new scan points corresponding to the existing planes, instead of finding planes in every scan. During matching, if the angle difference between the normal vectors of the plane and the existing plane exceeds a certain threshold, the matching is avoided, effectively resolving the double-sided issue.

In this letter, similar to existing normal vector-based algorithms, we consider the directions of normal vectors to find correspondences and effectively eliminate incorrect correspondences. Additionally, since degeneracy can occur in indoor scenarios where the directions of normal vectors are not evenly distributed, we propose a degenerate detection method and relative pose uncertainty estimation based on the distribution of the normal vectors.

III. METHOD

A. System Overview

The state of the system $\mathbf{X}$ is expressed as follows:

$$\mathbf{X} = [R, \mathbf{t}, \mathbf{v}, \mathbf{b}_a, \mathbf{b}_g] \tag{1}$$

where $R, \mathbf{t}, \mathbf{v}$ are the rotation matrix, position, and velocity in the global reference frame and $\mathbf{b}_a$ and $\mathbf{b}_g$ are the IMU acceleration
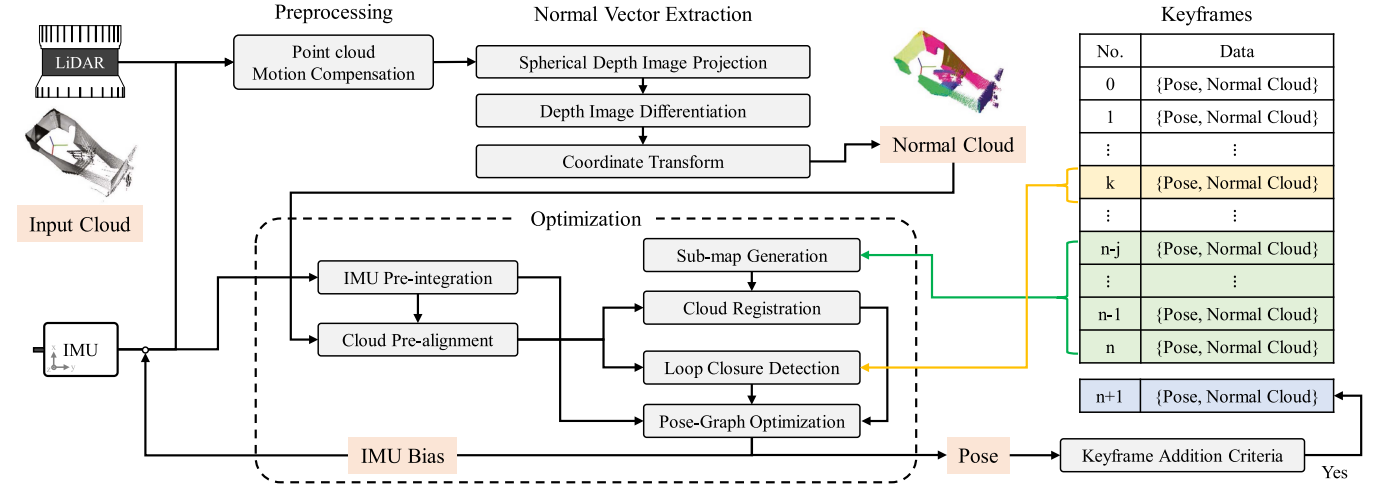
Fig. 2.   The overall framework of the proposed algorithm. The keyframe for loop closure is shown in yellow, the keyframes for sub-map generation are shown in green, and the newly added keyframe is shown in blue.

and gyroscope biases in IMU frame. When setting a body frame different from the IMU's coordinate system, transformations of IMU acceleration and angular velocity need to be performed according to the coordinate transformation. For convenience and to reduce additional computations, the body frame of the system is aligned with the IMU frame.

The keyframe $\mathbb{K}$ is expressed as follows:

$$\mathbb{K} = [\mathbf{x}(R, \mathbf{t}), N\{P_0(\mathbf{p}_0, \mathbf{n}_0), \ldots, P_{k-1}(\mathbf{p}_{k-1}, \mathbf{n}_{k-1})\}] \quad (2)$$

where $\mathbf{x}$ is the body pose in the global reference frame when the scan is made and N is the normal cloud containing normal points P_i that consist of the point coordinates $\mathbf{p}_i$ and the normal vector $\mathbf{n}_i$.

Unlike in outdoor environments, where a significant portion of the map is often within the LiDAR's range, the visibility of map points in the current scan may be extremely limited in narrow indoor spaces. Due to this characteristic, direct methods for matching scans directly to the map may result in drift, particularly in narrow corridors or during inter-floor transitions, making corrections difficult upon returning to the same location. Therefore, in this study, we adopt a keyframe-based pose-graph SLAM framework. The map is formed as a collection of keyframes, allowing it to dynamically adjust in response to corrections.

Fig. 2 illustrates the overview of the proposed algorithm. The in-scan motion is compensated by using the attitude calculated by integrating the IMU gyroscope measurements. The normal cloud is extracted by projecting the motion-compensated cloud using spherical projection. After aligning the extracted normal cloud using inertial measurements, we determine the relative pose through normal cloud registration between the submaps composed of preceding keyframes. Additionally, correction measurements are obtained through viewpoint-based loop-closure. These registration results are included in the graph as relative pose factors, and IMU measurements are also added to the graph via IMU preintegration. Through this pose-graph optimization, the current pose and IMU bias are estimated.

B. Spherical Projection & Normal Cloud Extraction

Normal vectors for each point in a dense LiDAR point cloud can be computed using the relationships with neighboring points, but this process can be computationally expensive. To address this, we adopted a technique similar to [15], converting the point cloud into a spherical range image for fast computation of normal vectors using neighboring pixel relationships. A spherical projection is performed on the motion-compensated LiDAR point cloud to generate a depth image. In this process, the size of the depth image is manually selected, taking into account the characteristics of the LiDAR point cloud, such as the number of LiDAR channels, horizontal resolution, and field of view (FoV). With given parameters of the maximum vertical FoV ($fov_{\max}$), minimum vertical FoV ($fov_{\min}$), depth image height (h) and width (w), the vertical resolution is as $ver_{res} = (fov_{\max} - fov_{\min})/h$, and the horizontal resolution is as $hor_{res} = 2\pi/w$. The image coordinates (u, v) of the each point ($\mathbf{p}(x, y, z)$) are as follows:

$$u = (\pi - atan2(y, x))/hor_{res}$$
$$v = (fov_{\max} - atan2(z, \sqrt{x^2 + y^2}))/ver_{res}. \quad (3)$$

The normal vector can be calculated by differentiating the depth value ($r(u, v)$) in horizontal direction ($\Delta r/\Delta \psi$) and vertical direction ($\Delta r/\Delta \theta$) of the range image as:

$$\mathbf{n}^s(u, v) = \begin{bmatrix} (\Delta r/\Delta \psi)/c \\ -(\Delta r/\Delta \theta)/c \\ 1/c \end{bmatrix} \quad (4)$$

where $\theta = \pi/2 - (fov_{\max} - v \times ver_{res})$ represents the polar angle, $\psi = \pi - u \times hor_{res}$ represents the azimuth angle, and c is a scaling variable making the normal vector as a unit vector.

In outdoor settings, adjacent pixels cover a larger area, reducing distance measurement noise in normal vector calculation. In contrast, indoor environments have smaller areas, increasing the noise impact. To address this, we use a window-based approach,
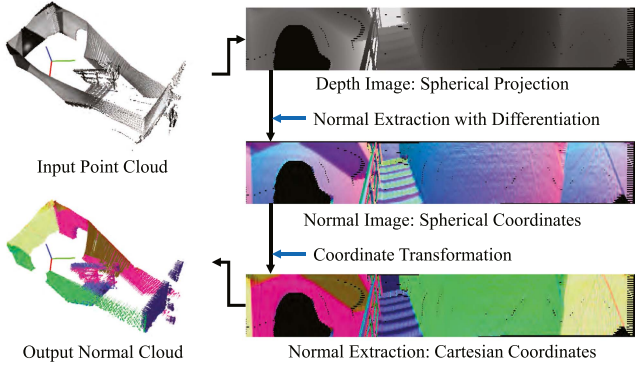
Fig. 3.   Normal cloud extraction process.

assuming similar derivative values within the window, rather than differentiating between neighboring pixels. The derivative values for pairs in each horizontal and vertical direction within the window are calculated and averaged to mitigate the effects of distance measurement noise. The normal vector represented in spherical coordinates, $\mathbf{n}^s$, is transformed into Cartesian coordinates, $\mathbf{n}^c$, based on the azimuth and elevation for each pixel, as $\mathbf{n}^c(u, v) = T(\theta, \psi)\mathbf{n}^s(u, v)$ where $T(\theta, \psi)$ is the transformation matrix as follows:

$$T(\theta, \psi) = \begin{bmatrix} -\sin(\psi) & \cos(\psi)\cos(\theta) & \cos(\psi)\sin(\theta) \\ \cos(\psi) & \sin(\psi)\cos(\theta) & \sin(\psi)\sin(\theta) \\ 0.0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}. \quad (5)$$

Since the transformation matrices for all pixels remain stationary, computational time is reduced by precomputing and storing the transformation matrices beforehand.

As the detected surface can not face parallel or in the direction of the ray ($\vec{\mathbf{r}} = \mathbf{p}/|\mathbf{p}|$), the normal vector is inversed when the dot product of the normal vector and the ray direction ($\mathbf{n}^c \cdot \vec{\mathbf{r}}$) is positive. Finally, to verify whether the normal vector forms a consensus with neighboring points in the window, the distances between the points and the planes formed by their normal vectors and neighboring points are calculated. If the number of points with a point-to-plane distance within 5 cm is less than one-third of the window size, it is considered invalid. Through this process, the normal cloud N containing the normal points P is obtained. Fig. 3 shows the overall process of normal cloud extraction from the input cloud.

C. Normal Cloud Registration

The scan-to-scan matching may lead to unreliable registration results especially for the LiDAR motion in roll or pitch rotation and translation in the scan rotation axis. Similar to LIO-SAM, we adopted the scan-to-submap matching method to address this issue. The submap is generated by accumulating the normal clouds of the preceding keyframes in the last keyframe coordinate system. For the last keyframe $\mathbb{K}_n$, the submap $\mathbf{N}_n$ augmenting previous j keyframes is as follows:

$$\mathbf{N}_n = N_n^n \cup N_{n-1}^n \cdots \cup N_{n-j}^n \quad (6)$$

where N_i^k indicates normal cloud in keyframe $\mathbb{K}_i$ transformed into coordinate system of keyframe $\mathbb{K}_k$, and $\cup$ indicates the normal cloud augmentation.

For accurate correspondence search and fast matching, the current query frame $\mathbb{K}_q$ is transformed from its last obtained pose to the initial pose through IMU integration. Knowing the world coordinate system of both the target and query frames allows us to determine the initial relative pose between the two frames. Utilizing this information, the target frame is transformed into the coordinate system of the query frame and proceeds with the matching process. Afterward, for faster matching speed, the current normal cloud N_q and the submap $\mathbf{N}_n$ are down-sampled using a voxel grid filter.

To achieve stable matching between the resulting normal clouds, correspondences are established between pairs that satisfy the following two criteria: First, pairs for which the point-to-point distances fall within the distance threshold. Second, pairs for which the difference in direction of the normal vectors falls within the angle threshold. To accomplish this, we first utilize a kd-tree to select submap points within the distance threshold for each query point in the current normal cloud. Then for the selected submap points in closest order, the angle difference in normal vector direction between the selected point and the query point is calculated. These two points are selected as correspondence pair if the angle difference falls within the angle threshold.

Afterwards, the relative pose is calculated using the Gauss-Newton method by utilizing the correspondence pairs. The residual cost function for each pair is calculated as the point-to-plane distance, and the relative pose of the target frame from the query frame can be calculated by solving the following optimization problem:

$$(R, \mathbf{t}) = \underset{(R, \mathbf{t})}{\operatorname{argmin}} \sum_{k=0}^{m-1} (\mathbf{n}_{qk} \cdot (R\mathbf{p}_{tk} + \mathbf{t} - \mathbf{p}_{qk}))^2. \quad (7)$$

The resulting relative pose is then converted to the relative pose factor and added to the factor graph.

D. Loop Closure Detection

Global loop detection algorithms often struggle in multifloor indoor environments characterized by repetitive structural features. This challenge is particularly evident in stairwells, where the recurring nature of their features can result in incorrect associations with clouds from different floors during inter-floor transitions. Therefore, in this study, we adopt a loop detection approach that focuses on matching keyframes within the vicinity of the current position rather than performing a global search.

While techniques like ICP or GICP, which use a radius search to find the closest points as correspondences, are frequently employed for local loop detection methods, they often lead to misalignment, especially in confined indoor spaces. This is mainly because indoor environments are typically composed of multiple-segmented areas, leading to significant variations in LiDAR scans even with slight changes in LiDAR position. To address this issue, we introduce a viewpoint-based loop closure
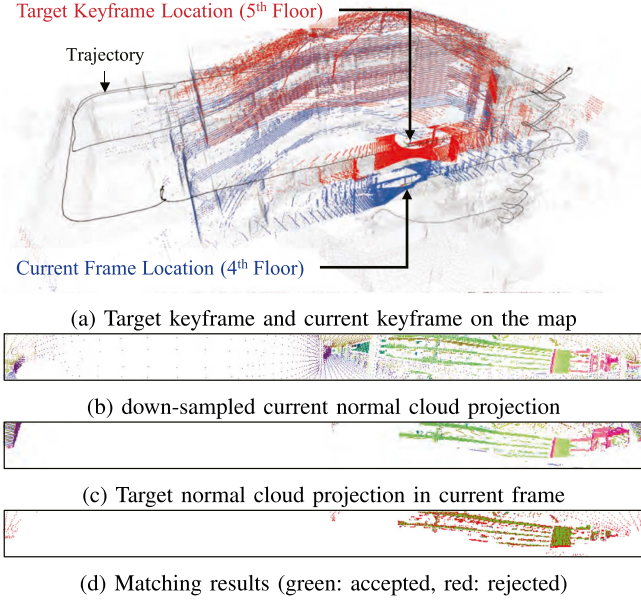
(a) Target keyframe and current keyframe on the map

(b) down-sampled current normal cloud projection

(c) Target normal cloud projection in current frame

(d) Matching results (green: accepted, red: rejected)

Fig. 4. Inter-floor loop closure detection example using viewpoint-based loop detection.

(a) Illustration of degenerate cases

(b) Calculated uncertainty using normal vector distributions

Fig. 5. Degenerate cases in the stairwell and in the long corridor environment.

detection method inspired by the projection techniques in [13] for better correspondence search.

Fig. 4 illustrates an example of viewpoint-based loop detection. A kd-tree is constructed using the positions of each keyframe, and then the closest keyframe to the current frame is selected. To avoid redundant computations and enhance efficiency, keyframes that are too recent (i.e., those created immediately before the current frame) are excluded from the kd-tree. Once a loop closure candidate keyframe is identified, the normal cloud of the candidate keyframe is transformed into the LiDAR pose of the current frame. Then, as shown in Fig. 4(c), it is projected to match the current LiDAR viewpoint. For the points that are assigned to the same pixels during spherical projection, only the point with the closest range is assigned.

This projection method, however, may not exclude points that should not be seen from the current viewpoint due to the sparse nature of the LiDAR point cloud. To address this issue, we utilize the normal points that lie within a 90-degree angle to the ray direction, denoted as N^+. As these points represent the opposite side of the wall, it is desirable to exclude any normal points that exceed a 90-degree angle to the ray direction, denoted as N^-, which pass through these points. Thus, for each pixel of the N^+ points, any nearby pixels of N^- points are excluded if the N^- point is further away than the N^+ point.

Subsequently, by filtering the projected points based on radial distance deviation and angular deviation compared to the down-sampled current normal cloud projection image (Fig. 4(b)), the correspondence pairs are obtained (Fig. 4(d)). Similar to the normal cloud registration method in Section III-C, the matched point correspondence pairs are optimized to estimate the relative pose. These are then inserted into the pose-graph as loop closure factors.
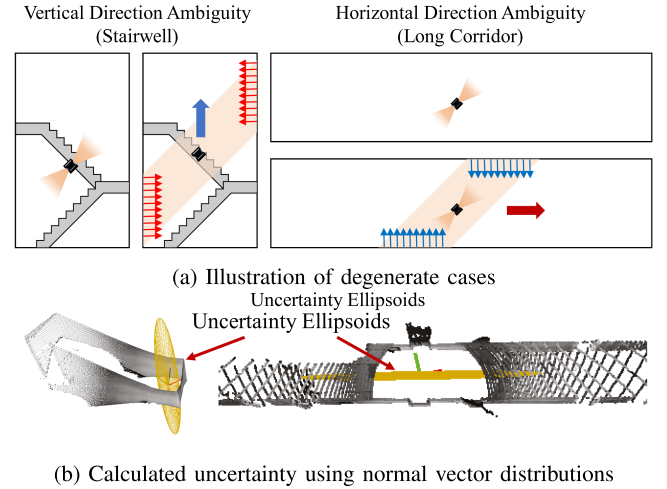
E. Degeneracy Detection

In indoor environments, many surfaces are often arranged parallel to each other, leading to cases of degeneracy. Environments such as stairwells or long corridors, as illustrated in Fig. 5(a), exemplify such cases, where the normal vectors of surfaces are distributed in only two directions, resulting in translation ambiguity along the remaining direction. For instance, in the case of a stairwell, the normal vectors of the walls forming the stairwell are distributed horizontally, leading to high localization accuracy in the horizontal direction but potential ambiguity in the vertical direction.

To calculate a suitable matching uncertainty for such degeneracy situations, the distribution of normal vectors among the matched normal points can be utilized. This can be obtained through principal component analysis of the normal vectors as follows: First, the covariance matrix C of the normal vectors is calculated as:

$$\mathbf{C} = \left(\frac{1}{m} \sum_{k=0}^{m-1} (\mathbf{n}_k \mathbf{n}_k^\top) \right). \tag{8}$$

Subsequently, the covariance matrix C is decomposed using eigenvalue decomposition as $C = V \Lambda V^{-1}$ where V is the matrix composed of eigenvectors and Λ is the matrix in which the diagonal elements are eigenvalues as:

$$V = [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2]$$
$$\Lambda = diag(\lambda_0, \lambda_1, \lambda_2) \tag{9}$$

where $\lambda_0 < \lambda_1 < \lambda_2$. The approximate distribution of normal vectors can be discerned using eigenvalues, wherein the smallest eigenvalue, λ_0, indicates a degenerate case if it falls below a specific threshold. Subsequently, each eigenvalue λ_i corresponds to an eigenvector $\mathbf{v}_i$, allowing the measurement covariance Q to be set as follows:

$$Q = s \, V diag \left(\frac{1}{\lambda_0}, \frac{1}{\lambda_1}, \frac{1}{\lambda_2} \right) V^\top \tag{10}$$

where s is a given constant. When matching with the immediately preceding keyframe, we inserted factors using this distribution-based measurement covariance in degenerate cases. However, in cases of loop closing where there is a high probability of incorrect matching, we refrained from inserting loop factors if degeneracy was detected to ensure stability. Fig. 5(b) illustrates the registration uncertainty, showing significant uncertainty in the vertical direction for stairwells and along the corridor direction for long corridors. This allows us to assign more weight to the IMU measurements in the directions where ambiguity arises during the graph optimization process while relying more on the LiDAR registration results in the other directions.

F. Pose-Graph SLAM Framework

The pose-graph consists of the prior factor given to the initial pose, relative pose factors obtained through the normal cloud registration, loop closure factors from the loop detection, and IMU factors and constant bias factors from the IMU-preintegration. The pose-graph was constructed and optimized using the iSAM2 framework [16].

G. Implementation Details

The number of pixels used in spherical image projection was set to the channel number by 1024. For stable normal vector extraction, a 3 by 3 window was used for LiDAR with 32 channels or fewer, while a 5 by 5 window was used for LiDAR with more than 32 channels to compute the normal vectors. The distance threshold for normal cloud registration was set to 0.5 m, and the down-sampling voxel size was set to 0.4 m or 0.2 m depending on the scenario. For keyframes, a new keyframe is added if the angle difference from the previous keyframe pose exceeds 30 degrees, or the distance difference is larger than a threshold. This distance threshold was set to be 1.0 m or 0.5 m depending on the environments. The loop closure distance threshold was set to 10 meters.

IV. EXPERIMENTS

To evaluate the performance of the proposed algorithm, we conducted tests using various datasets captured with different types of spinning LiDARs in diverse environments, including the SubT-MRS Dataset [17], the Newer College Dataset Extension [18], and our own dataset. The SubT-MRS Dataset encompasses narrow environments such as subterranean caves, a nuclear power plant, and a multifloor building. The dataset was acquired using a 16-channel LiDAR and an IMU. The Newer College Dataset Extension was captured in a campus environment using a 128-channel LiDAR and its built-in IMU. For our own dataset, the data was acquired in various buildings located on the KAIST campus using a handheld device. Our algorithm was compared against state-of-the-art algorithms, including LIO-SAM [7], FAST-LIO2 [9], LTA-OM [19], Faster-LIO [11], and DLIO [12] which are publicly available. The keyframe insertion distance and down-sampling voxel size of each algorithm were made identical for each scenario to ensure

TABLE I
RESULT OF SubT-MRS DATASET [17]

Algorithm	RMSE (m)				
	Final UGV1	Final UGV3	Urban UGV2	Laurel Caverns	Multifloor
LIO-SAM	0.3689	2.7598	0.1940	14.739	Failed
FAST-LIO2	0.2619	3.5930	0.3534	Failed	Failed
LTA-OM	0.3838	Failed	0.4872	Failed	Failed
Faster-LIO	0.3219	0.9616	0.2022	1.3551	1.4230
DLIO	0.3794	0.7569	0.2276	0.5947	2.2460
NV-LIOM	**0.1142**	**0.4141**	**0.1522**	**0.4045**	**0.2541**
w\o loop	0.8907	0.7728	1.5818	1.2346	0.5237
w\o deg.	0.2573	0.8369	0.2159	0.4265	1.0126

The bold values indicate the minimum value of each column, indicating the best performing algorithm for the same dataset.

a fair comparison. The absolute pose error was measured using the translation root mean square error (RMSE) after aligning the trajectory with Umeyama [20] alignment using evo [21].

To demonstrate the effectiveness of the proposed degeneracy detection algorithm and the view-based loop closure algorithm, we have added an ablation study. These are indicated as 'w\o deg.' and 'w\o loop' in the tables, respectively. The cases marked as 'Failed' indicate the instances where the odometry estimation during mapping is significantly off that further mapping becomes impossible due to severe drift. All tests were done online using a computer equipped with Intel i7-12700 CPU with 12 cores.

A. Results

1) SubT-MRS Dataset: Table I shows the results of the SubT-MRS Dataset. NV-LIOM showed lower RMSE across all datasets. Specifically, in the Multifloor scenario, LIO-SAM, FAST-LIO2, and LTA-OM failed to proceed mapping after experiencing significant drift in the stairwell. DLIO showed large errors due to incorrect keyframe matching across different floors.

2) Newer College Dataset Extension: The results for the Newer College Dataset Extension are shown in Table II . NV-LIOM demonstrates similar or lower errors compared to state-of-the-art methods in outdoor environments while showing greater robustness in very narrow environments as shown in the Stairs scenario.

3) Dataset of KAIST Buildings: Fig. 6 illustrates the results for the stairwell dataset of the 5-floor building, which was shown in Fig. 1. The result of LTA-OM was omitted as it showed almost identical result with FAST-LIO2. LIO-SAM failed to register as soon as it entered the stairwell, resulting in mapping failure to the extent that further progress was impossible (Fig. 6(a)). While FAST-LIO2 and Faster-LIO did not fail, but the errors accumulated gradually as ascending the stairs, resulting in curved mapping result of the stairwell (Fig. 6(b) and (c)). DLIO suffered wrong keyframe matching at the beginning, resulting in one floor missing from the whole stairwell. It also had the same problem of curved mapping due to the accumulation of the errors (Fig. 6(d)). On the other hand, the mapping result using NV-LIOM (Fig. 6(e)) shows a vertical structure, preserving the actual shape of the stairwell. Fig. 6(f) and (g) represent magnified

TABLE II
RESULT OF NEWER COLLEGE DATASET EXTENSION [18]

Algorithm	RMSE (m)												Avg. Comp. (ms)
	Quad Easy	Quad Medium	Quad Hard	Stairs	Cloister	Park	Math Easy	Math Medium	Math Hard	Under-Easy	Under-Medium	Under-Hard	
LIO-SAM	0.0743	**0.0668**	**0.1260**	Failed	**0.0741**	0.3683	0.0819	0.1259	**0.0852**	**0.0578**	**0.0645**	0.4188	113.7
FAST-LIO2	0.0830	0.0823	0.1843	0.1130	0.1229	0.3527	0.1015	0.1249	0.1336	0.1066	0.1050	0.1040	**22.74**
LTA-OM	0.0827	0.0829	0.1329	0.3619	0.1403	0.4073	0.0947	0.1346	0.1866	0.1075	0.1263	0.1175	27.13
Faster-LIO	0.2631	0.1624	0.1756	0.1171	0.1711	0.4086	0.1740	0.1899	0.0937	0.0597	0.0805	0.0849	22.92
DLIO	**0.0734**	0.0691	0.1655	0.1531	0.0935	0.3185	0.1141	0.1066	0.0853	0.0642	0.0655	0.0828	34.52
NV-LIOM	0.0762	0.0757	0.1780	**0.0876**	0.0767	**0.2896**	**0.0616**	**0.0952**	0.0936	0.0622	0.0718	**0.0780**	54.25
w\o loop	0.6936	0.5198	0.4606	0.1023	0.5001	2.5086	0.2420	0.8299	0.6169	0.1420	0.1349	0.1507	46.71
w\o deg	0.0773	0.0774	0.1953	0.1341	0.0858	0.3497	0.0755	0.0979	0.1098	0.0773	0.0813	0.0907	52.89

The bold values indicate the minimum value of each column, indicating the best performing algorithm for the same dataset.
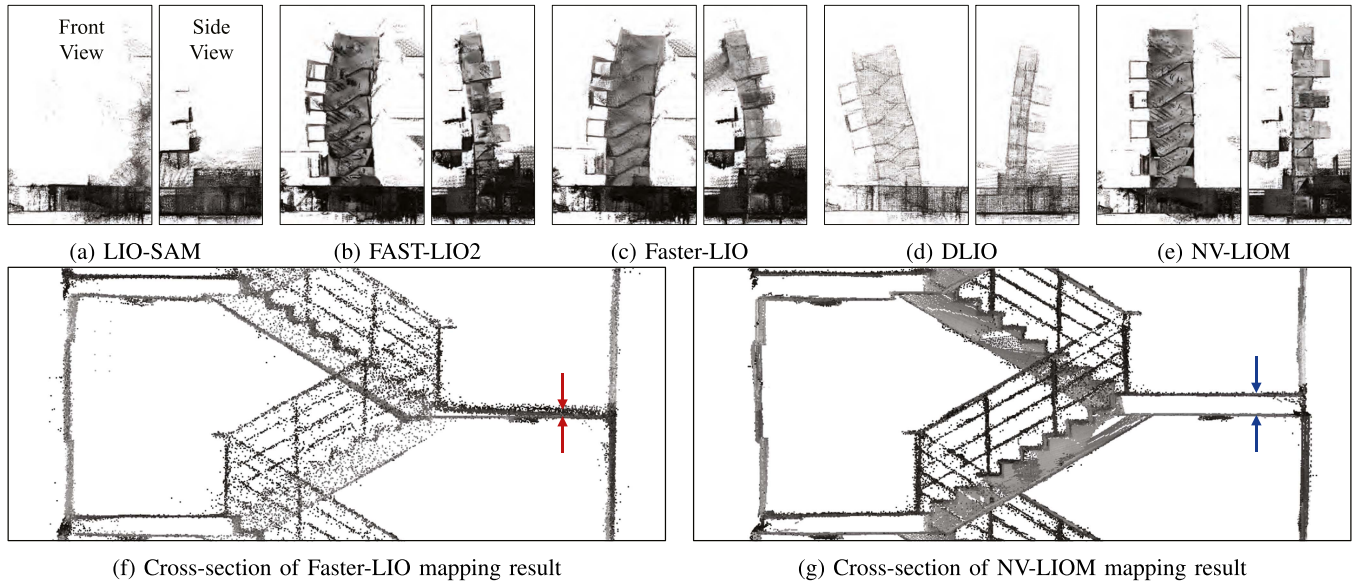

(a) LIO-SAM   (b) FAST-LIO2   (c) Faster-LIO   (d) DLIO   (e) NV-LIOM

(f) Cross-section of Faster-LIO mapping result   (g) Cross-section of NV-LIOM mapping result

Fig. 6. Mapping results of the own dataset (stairwell).

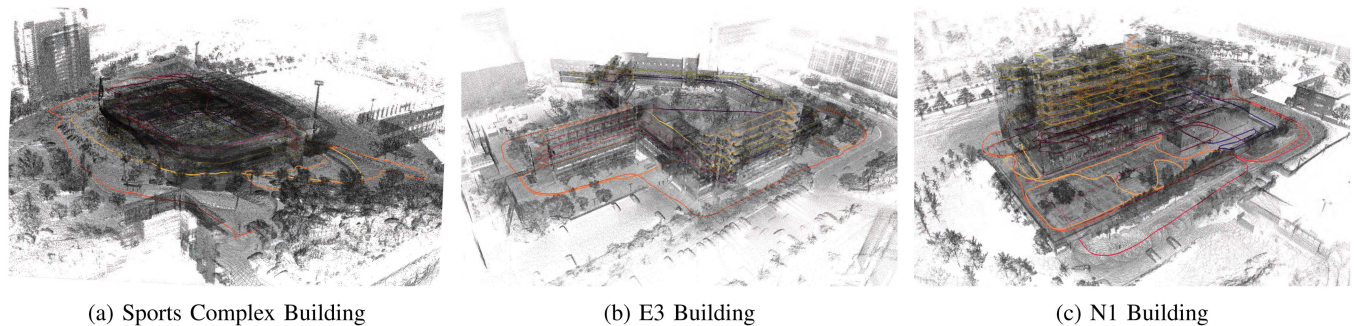(a) Sports Complex Building   (b) E3 Building   (c) N1 Building

Fig. 7. Mapping result of buildings with different characteristics.

views of the results obtained from Faster-LIO and NV-LIOM, respectively. As indicated by the arrows, while Faster-LIO failed to map properly due to incorrect correspondence between the ceiling of the lower floor and the floor of the upper floor, NV-LIOM considers the directions of the normal vectors, avoiding mismatches and leading to correct results.

Additionally, NV-LIOM was assessed in buildings with varying characteristics. Fig. 7(a) depicts a stadium-shaped building with an underground parking lot and a three-story structure with an open center. Fig. 7(b) shows a building comprising research labs and classrooms across five and six stories, connected to the three-story building. Fig. 7(c) represents a research facility consisting of an underground parking lot and nine above-ground stories. NV-LIOM conducted online SLAM without failing for each sequence, lasting approximately one hour, allowing for qualitative evaluation of the mapping results.

TABLE III
COMPUTATIONAL TIME FOR EACH PROCESS [MS]

Process	Spherical Projection	Normal Extraction	Cloud Reg.	Loop Closure	Graph Opt.
Time [ms]	0.695	2.256	10.07	15.76	18.56

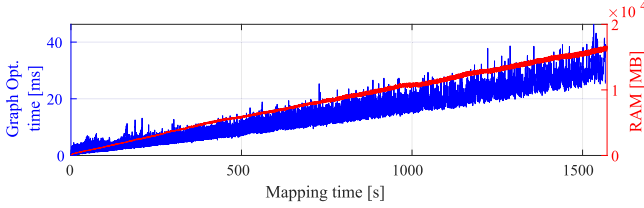Fig. 8. Graph optimization time and RAM usage over elapsed mapping time.

All mapping sequences and results are available in our code repository.

B. Ablation Studies

Our proposed algorithm, being keyframe-based, exhibited relatively larger errors without a loop closure module. This phenomenon was observed in sequences with long trajectories and was particularly noticeable in environments like the Park scenario in Table II, which covers a wide area.

In environments where the directions of the normal vectors of the point cloud are diverse (i.e. outdoor environments), the presence or absence of the degeneracy module did not significantly affect performance. However, in featureless, narrow environments like the SubT-MRS dataset, there was a noticeable difference. This difference was particularly evident in stairwell environments prone to degeneracy, as seen in the Multifloor sequence in Table I and the Stairs sequence in Table II.

C. Computational Time and Memory Usage

The result in Table II indicates that NV-LIOM requires additional computational time. Table III breaks down this computational time for each module, showing that cloud registration, loop closure, and graph optimization are the primary contributors to the increased computational load. Fig. 8 shows the keyframe number, graph optimization time, and memory usage over elapsed mapping time. While the time required for other modules remains fairly constant, graph optimization time increases with the number of keyframes, reaching approximately 30 ms for around 2000 keyframes.

Additionally, for the same dataset, our proposed algorithm's memory usage increases by approximately 7.39 MB with each added keyframe. Each keyframe needs to include a normal cloud for stable registration and loop detection, resulting in a memory usage increase proportional to the number of keyframes. This can vary depending on parameters such as keyframe distance, depth image size, and down-sampling voxel size.

V. CONCLUSION

This letter introduces NV-LIOM, a normal vector-based tightly-coupled LiDAR-inertial odometry and mapping framework designed for narrow indoor environments. NV-LIOM utilizes the normal vectors extracted from the LiDAR scans for cloud registration, degeneracy detection, and loop closure detection to ensure robust SLAM performance in narrow indoor environments. The proposed method was evaluated through public datasets and our own dataset encompassing various types of buildings. The experimental results demonstrate that NV-LIOM outperforms existing methods in terms of accuracy and robustness, particularly in narrow multifloor indoor environments.

REFERENCES

[1] H. Lim, D. Kim, B. Kim, and H. Myung, "AdaLIO: Robust adaptive LiDAR-inertial odometry in degenerate indoor environments," in *2023 20th Int. Conf. Ubiquitous Robots*, 2023, pp. 48–53.

[2] B. Kim, C. Jung, D. H. Shim, and A. Agha-mohammadi, "Adaptive keyframe generation based LiDAR inertial odometry for complex underground environments," in *2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3332–3338.

[3] L. Zhou, D. Koppel, and M. Kaess, "LiDAR SLAM with plane adjustment for indoor environment," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7073–7080, Oct. 2021.

[4] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[5] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot.: Sci. Syst.*, Seattle, WA, USA, Jun. 2009.

[6] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in realtime," in *Proc. Robot.: Sci. Syst.*, Berkeley, CA, USA, Jul. 2014, pp. 1–9.

[7] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

[8] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proc. Robot.: Sci. Syst.*, Rome, Italy, Jul. 2015.

[9] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

[10] Y. Cai, W. Xu, and F. Zhang, "ikd-Tree: An incremental K-D tree for robotic applications," 2021. [Online]. Available: https://github.com/hku-mars/ikd-Tree

[11] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-LIO: Lightweight tightly coupled LiDAR-inertial odometry using parallel sparse incremental voxels," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4861–4868, Apr. 2022.

[12] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct LiDAR-inertial odometry: Lightweight LIO with continuous-time motion correction," in *2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3983–3989.

[13] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Proc. Robot.: Sci. Syst.*, Pittsburgh, PA, USA, Jun. 2018.

[14] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: LiDAR-inertial 3D plane SLAM," in *2018 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 123–130.

[15] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *2011 IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3084–3091.

[16] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, 2012.

[17] S. Zhao et al., "SubT-MRS dataset: Pushing SLAM towards all-weather environments," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2024, pp. 22647–22657.

[18] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-camera LiDAR inertial extension to the newer college dataset," 2021. [Online]. Available: https://ori-drs.github.io/newer-college-dataset/

[19] Z. Zou et al., "LTA-OM: Long-term association LiDAR–IMU odometry and mapping," *J. Field Robot.*, early access, Apr. 15, 2024, doi: 10.1002/rob.22337.

[20] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, Apr. 1991.

[21] M. Grupp, "EVO: Python package for the evaluation of odometry and SLAM," 2017. [Online]. Available: https://github.com/MichaelGrupp/evo