2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

LiDAR-Inertial SLAM with Efficiently Extracted Planes

Chao Chen[1*], Hangyu Wu[1*], Yukai Ma[1], Jiajun Lv[1], Laijian Li[1], Yong Liu[1]

***Abstract*— This paper proposes a LiDAR-Inertial SLAM with efficiently extracted planes, which couples explicit planes in the odometry to improve accuracy and in the mapping for consistency. The proposed method consists of three parts: an efficient Point→Line→Plane extraction algorithm, a LiDAR-Inertial-Plane tightly coupled odometry, and a global plane-aided mapping. Specifically, we leverage the ring field of the LiDAR point cloud to accelerate the region-growing-based plane extraction algorithm. Then we tightly coupled IMU pre-integration factors, LiDAR odometry factors, and explicit plane factors in the sliding window to obtain a more accurate initial pose for mapping. Finally, we maintain explicit planes in the global map, and enhance system consistency by optimizing the factor graph of optimized odometry factors and plane observation factors. Experimental results show that our plane extraction method is efficient, and the proposed plane-aided LiDAR-Inertial SLAM significantly improves the accuracy and consistency compared to the other state-of-the-art algorithms with only a small increase in time consumption.**

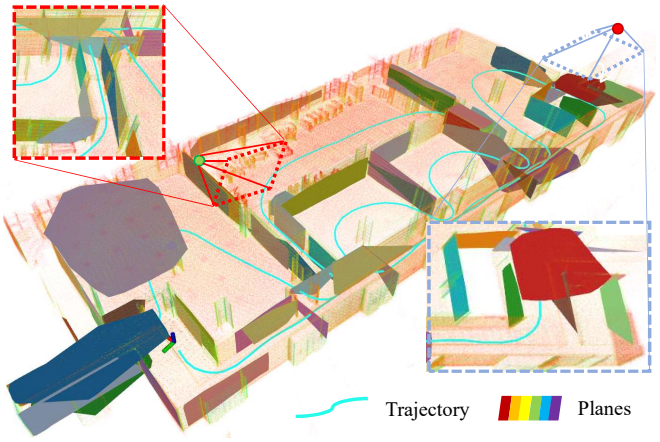


Fig. 1: The hybrid map constructed by the proposed method on *units-dolly* sequence of VECtor [7] dataset. We have removed some ceiling planes for better visualization.

I. Introduction

With the ability of real-time 3D reconstruction and self-localization, LiDAR-based simultaneous localization and mapping (SLAM) has been widely used in many indoor robotic applications, such as autonomous navigation, control, and motion planning. However, due to the narrow indoor space and repetitive structures, traditional LiDAR-based SLAM suffers from significant cumulative errors. On the other hand, planes ubiquitously exist in the indoor environment, and how to quickly extract and apply them reasonably in SLAM gains extensive attention recently.

Some works utilized locally fitted planes in scan registration, such as surfel-based methods [1, 2] and local plane feature methods [3, 4], while explicitly existing planes in the environment are ignored. Other works introduced explicit planes, such as BALM [5] and π-LSAM [6]. However, the plane extraction algorithms are computationally intensive which limits their real-time applications.

To address this issue, we propose an efficient LiDAR ring-based plane extraction algorithm and tightly couple explicit planes in the odometry and mapping of LiDAR-Inertial SLAM (LI-SLAM). The main contributions of this paper are as follows:

- We propose a ring-based Point→Line→Plane extraction algorithm, which extracts explicit planes in point cloud efficiently and robustly.
- We develop a LI-SLAM system with tightly coupled explicit planes both in the odometry and mapping, which improves the accuracy and consistency of the system.
- The proposed method is validated on both the public dataset (i.e., VECtor [7]) and the self-collected dataset. Results show that our system achieves better performance with little increase in time cost compared to the other state-of-the-art LiDAR-Inertial SLAM algorithms.

This work is partially supported by the National Natural Science Foundation of China under grant NSFC 62088101.

*These two authors contribute equally to this work.

[1]The authors are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China. (Yong Liu is the corresponding author, email: yongliu@iipc.zju.edu.cn)

II. Related Works

A. LiDAR-Inertial SLAM

LiDAR-Inertial SLAM is broadly divided into two categories: loosely coupled methods and tightly coupled methods. Loosely coupled methods rely on prior predictions, using IMU data to remove the distortion of point cloud and provide initial pose for point cloud registration. For example, earlier methods like LOAM [3] and recent improvements like LeGO-LOAM [8] are loosely coupled systems. However, the accuracy of loosely coupled systems is low due to the lack of using the raw data from two sensors.

In contrast, tightly coupled systems jointly process LiDAR and IMU data, considering their internal relationship, which can usually provide better accuracy and robustness. LIO-mapping [9] adopts an optimization framework [10] to minimize the residual error of IMU and LiDAR measurements. LIO-SAM [4] limits computational complexity by introducing LiDAR key-frames and using factor graph to optimize IMU and LiDAR jointly. It is currently one of the most popular open-source LI-SLAM frameworks. FAST-LIO [11]

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10342325

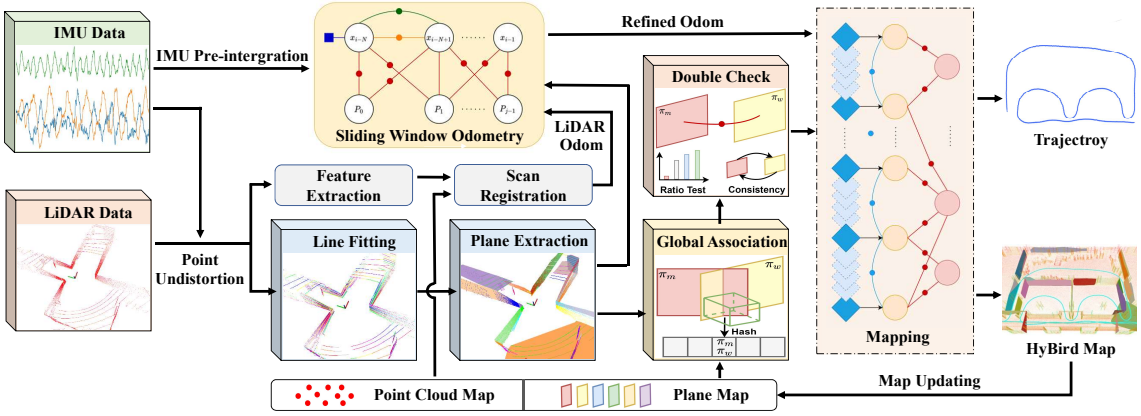


Fig. 2: System overview of our algorithm.

uses the Error-state Iterated Kalman Filter to couple IMU and LiDAR measurements tightly. FAST-LIO2 [12] proposed an incremental kd-tree to manage maps more efficiently.

B. LiDAR-based SLAM With Planes

In LiDAR-based SLAM algorithms, most algorithms use point cloud maps composed of edge or plane features, such as LOAM [3], LINS [13], LIO-SAM [4] and Lili-OM [14]. When registering a new scan, each point in the scan is registered to a local fitted plane by iterated closet point (ICP) [15] or its variants (e.g., generalized ICP [16]). The recent work VoxelMap [17] manages local planes using voxel maps and estimates plane parameters and confidences simultaneously.

Other works try to explicitly extract planes in the environment and add them to SLAM as landmarks. Kaess et al. [18] propose a quaternion-based plane parameterization method and improve the convergence speed by constructing a relative plane model. Geneva et al. [19] utilize the closest point (CP) to parameterize the plane, which outperforms the plane parameterization method introduced in [18]. Liu et al. derive the first and second order derivatives of the point-to-plane cost function in BALM [5], enabling efficient iterative optimization by the Levenberg-Marquardt algorithm [20]. Subsequently, Zhou et al. propose π-LSAM [6], which refers to the concept of Bundle Adjustment (BA) in visual SLAM and jointly optimizes key-frames and planes in the sliding window using a Plane Adjustment (PA) method.

However, most of these works extract planes with RANSAC [21], which is very slow. Moreover, the complex plane utilization in the SLAM framework increases the calculation, which brings significant challenges to the real-time performance of the system.

III. System Overview

The overview of our system is shown in Fig. 2. We integrate IMU data to get the predicted pose for LiDAR point cloud distortion and registration. For the input LiDAR point cloud data, we extract local edge and plane features (Feature Extraction in Fig. 2) [3] and register them with the feature map (Scan Registration in Fig. 2) to obtain the initial pose (LiDAR Odom in Fig. 2). Then we extract explicit planes (Plane Extraction in Fig. 2, Sec IV) for later use. We construct a factor graph consisting of IMU pre-integration factors [22], LiDAR odometry factors, and explicit plane factors within the sliding window (Sliding Window Odometry in Fig. 2, Sec. V-C), and solve the problem to obtain a more accurate odometry (Refined Odom in Fig. 2). Finally, we perform a key-frame-based global optimization with optimized odometry factors and global plane factors for consistent mapping (Mapping in Fig. 2, Sec. V-D).

IV. Efficient Plane Extraction

In this section, we will introduce the proposed efficient plane extraction method which groups points into lines followed by lines into planes, i.e. Point→Line→Plane. Before diving into details, we define the plane $\pi = [\mathbf{n}\ d] \in \mathbb{R}^4$ and line $\mathbf{l} = [\mathbf{n}\ \mathbf{m}] \in \mathbb{R}^6$, where $\mathbf{n} \in \mathbb{R}^3$ ($\|\mathbf{n}\|_2 = 1$) is the unit direction vector of the plane or line, $d \in \mathbb{R}^1$ is the signed distance from frame origin to the plane, and $\mathbf{m} \in \mathbb{R}^3$ is the midpoint of the line.

A. Ring-based Line Fitting

The basic idea of line fitting based on rings is similar to the *Region Growing* [23]. Firstly, we classify the scan by ring field and sort the points by increasing timestamps. Then we randomly select a point in each ring as a seed and traverse the points one by one in time forward and backward to fit the line segment. If the distance d_{pl} (Eq. 1) from one point $\mathbf{p}_i$ to the fitting line $\mathbf{l} = [\mathbf{n}\ \mathbf{m}]$ is less than the threshold ϵ_{pl}, we group the point into the line and update the parameter of the line. Otherwise, we select $\mathbf{p}_i$ as a new seed to grow a new line.

$$d_{pl} = \frac{\|(\mathbf{p}_i - \mathbf{m}) \times \mathbf{n}\|}{\|\mathbf{n}\|} < \epsilon_{pl}. \tag{1}$$

Given a set of points of a line, we update the midpoint of the line as the mean of all points and the direction as the eigenvector corresponding to the largest eigenvalue of the covariance matrix of all points. The eigen-decomposition in direction update is the most time-consuming step in line fitting. For more efficient calculation, we use an analytical solution [24] specifically for the symmetric 3×3 matrix to accelerate eigenvalue decomposition.

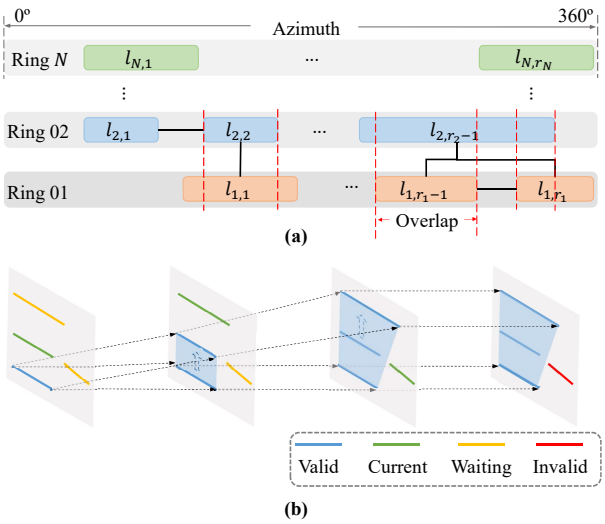


Fig. 3: (a) Line graph constructed with fitted lines. (b) Plane extraction based on the line graph.

B. Line-based Plane Extraction

In order to further fit the plane based on fitted lines, we construct a *Line Graph* as shown in Fig. 3(a). We denote all line segments as *Nodes* and the spatial adjacent relationship between line segments as *Edges*. Let $l_{i,j}$ denotes the j-th line segment detected in the i-th ring of a scan. There are two types of edges in a line graph: (1) edges between lines in the same ring, i.e. the edge between $l_{i,j}$ and $l_{i,j+1}$; (2) edges between lines in adjacent rings with overlap, i.e. the edge between l_{i,r_1-1} and l_{i+1,r_2-1}. The nodes connected with edges are donated as *Neighbor Nodes*. We apply the classic Breath-First Search (BFS) method to fit planes in the line graph. As shown in Fig. 3(b), specific steps are as follows:

(a) Start from the first node of the first ring and traverse nodes of the entire line graph.
(b) Search all *Neighbor Nodes* of the root node, try to initialize a plane given two nodes (lines).
(c) After the plane is initialized, traverse all of its *Neighbor Nodes*. If the line is on the plane, add it to the plane and update the plane parameters.
(d) After all nodes have been traversed, the algorithm ends.

We update the direction vector of a plane by fast eigenvalue decomposition on the covariance matrix of the points proposed in Sec. IV-A. The eigenvector corresponding to the smallest eigenvalue is the direction vector of the plane, and the parameter d of the plane is the signed distance from the frame origin to the midpoint of the plane.

Finally, we remove planes with small area and merge planes with similar parameters.

V. Plane-aided LiDAR-Inertial SLAM

In this section, we will introduce the proposed local plane-aided sliding window odometry to improve the accuracy of the odometry, and the global plane-aided mapping for establishing a more consistent map. In the following, we denote the world frame as W and the LiDAR frame as L.

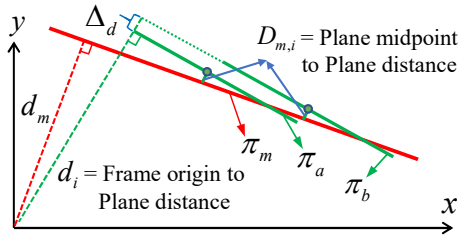


Fig. 4: The top view of the same map plane π_m (solid red line) and current observed plane $\pi_i, i \in \{a, b\}$ (solid green line).

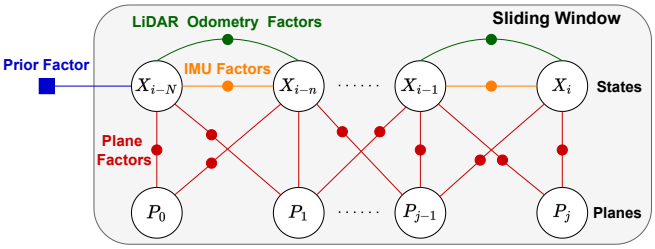


Fig. 5: The factor graph of sliding window optimization.

A. Plane Association

Given a plane $\pi^L = \begin{bmatrix}\mathbf{n}^L & d^L\end{bmatrix}^T$ in LiDAR frame L, we can transform it to the world fram W by

$$\pi^W = \begin{bmatrix}\mathbf{n}^W \\ d^W\end{bmatrix} = \begin{bmatrix}{}^W_L\mathbf{R} & \mathbf{0} \\ ({}^M_L\mathbf{t})^T\,{}^W_L\mathbf{R} & 1\end{bmatrix}\begin{bmatrix}\mathbf{n}^L \\ d^L\end{bmatrix} \tag{2}$$

where ${}^W_L\mathbf{R}$ and ${}^W_L\mathbf{t}$ are the rotation and translation from LiDAR frame to world frame respectively.

Suppose we have transferred planes of the new scan from the LiDAR frame to the world frame through Eq. 2. Let $\pi_i = \begin{bmatrix}\mathbf{n}_i & d_i\end{bmatrix}^T$ be a plane of the new scan to be associated and $\pi_m = \begin{bmatrix}\mathbf{n}_m & d_m\end{bmatrix}^T$ a plane in the map. We design a plane-distance-insensitive metric as follows:

$$\theta_{i,m} = \arccos\left(\mathbf{n}_i^T\mathbf{n}_m\right) \tag{3}$$

$$d_{i,m} = \mathbf{n}_m^T\mathbf{p}_i + d_m \tag{4}$$

where $\theta_{i,m}$ is the angle between the normal vector $\mathbf{n}_i$ and $\mathbf{n}_m$, and $d_{i,m}$ is the distance from the center $\mathbf{p}_i$ of plane π_i to the map plane π_m. If $\theta_{i,m} < \theta_{thre}$ and $d_{i,m} < d_{thre}$, we consider them to be the same plane. In practice, θ_{thre} is set as 15° and d_{thre} as 0.2 m. Most works [25–27] use the distance $d_{i,m} = |d_i - d_m|$ to measure the distance of two planes. However, we found it is sensitive to the plane's position. As shown in Fig. 4, there is a small angle error between the normal vector $\mathbf{n}_a$ and $\mathbf{n}_m$, resulting in the distance $d_{a,m}$ larger than $d_{b,m}$. In contrast, the distances from the center point of π_a and π_b to plane π_m are nearly identical, making it a more suitable choice for plane associations.

B. Scan Registration

For a point scanned at time $t \in [t_i, t_j)$, we linearly interpolate the pose between t_i and t_j using IMU integration, thus aligning all LiDAR points of the scan to the starting time t_i. For computational efficiency, we use the method proposed

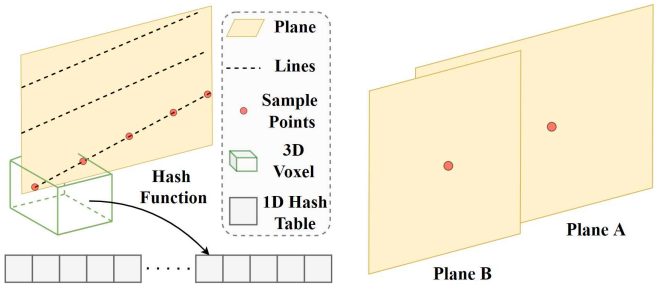


Fig. 6: (a) Plane map management using spatial voxel hash of planes. (b) Plane associations using hash indexing.

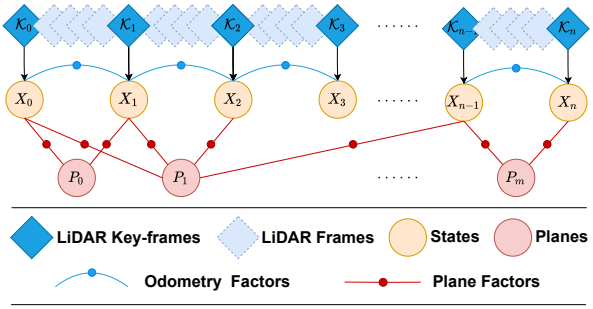


Fig. 7: The factor graph of plane-aided mapping.

by LOAM [3] to extract local edge and plane features for scan registration. To register a LiDAR scan at time t_i, we first select some nearest key-frames to construct a local map, consisting of a local edge map and a local plane map. Then we match edge features and plane features of the current frame with the corresponding feature map. With point-to-edge and point-to-plane residuals [28], we solve the problem iteratively by the Levenberg-Marquardt algorithm [20] and get the initial pose of this scan.

C. Plane-aided Odometry

In order to give a more accurate initial pose to mapping for plane association, we further optimize the pose in a sliding window with the aid of explicit planes. Since the planes in the sliding window are spatially adjacent, we simply use the method in Sec. V-A to associate the same planes. The factor graph of the sliding window is shown in Fig. 5, which consists of IMU pre-integration factors [22], LiDAR odometry factors, explict plane observation factors, and prior factors from marginalization. Here, we choose to include LiDAR odometry factors rather than LiDAR local feature factors in the sliding window optimization. It is a trade-off between accuracy and real-time.

Given two planes π_i and π_m in the world frame, the plane factor is defined as follows:

$$\pi_i \ominus \pi_m = \begin{bmatrix} \xi^\top \mathbf{B}_p & d_i - d_m \end{bmatrix} \in \mathbb{R}^3 \tag{5}$$

The operation $\ominus$ represents the error between two planes, where $\mathbf{B}_p \in \mathbb{R}^{3\times 2}$ is a set of basis on tangent plane of $\mathbf{n}_i$, and $\xi \in \mathbb{R}^3$ is defined as

$$\xi = -\frac{\arccos\left(\mathbf{n}_i^T \mathbf{n}_m\right)}{1-\left(\mathbf{n}_i^T \mathbf{n}_m\right)^2}\left(\mathbf{n}_m - \left(\mathbf{n}_i^T \mathbf{n}_m\right)\mathbf{n}_i\right) \in \mathbb{R}^3 \tag{6}$$

We recommend readers to refer to [29] for more details.

D. Plane-aided Mapping

1) Hash-based Map Plane Indexing: To improve the speed and robustness of plane associations between scan and global plane map, we propose a 3D voxel-based plane indexing method to detect if there are overlaps among planes. Besides, we map the 3D voxel to 1D variables via a hash function for fast indexing. As shown in Fig. 6(a), we first sample the lines in the plane and get the point sets $\mathbb{P}$. Then for any point $\mathbf{p}_i \in \mathbb{P}$, we calculate its voxel coordinates,

$$\mathbf{p}_i = [p_x, p_y, p_z]^T, \quad \mathbf{v} = \frac{1}{s}\left[[p_x], [p_y], [p_z]\right]^T \tag{7}$$

where $\mathbf{v}$ is the integer coordinate and s is the voxel size. Given a voxel $\mathbf{v} = (v_x, v_y, v_z)$, we can get the hash value of it using the hash algorithm [30]:

$$\text{hash}(\mathbf{v}) = (v_x n_x \text{ xor } v_y n_y \text{ xor } v_z n_z) \bmod \mathbb{N} \tag{8}$$

where n_x, n_y, n_z are three large prime numbers, and $\mathbb{N}$ is the size of the hash table. Finally, we maintain a hash table, each element of which contains an array with plane ids that intersect with this element's corresponding voxel.

2) Global Map Plane Association: Given a plane π_i of current scan, we first calculate its voxel coordinates and hash index. Then we query all plane ids with the same hash index as π_i in the hash table to get candidate planes in the map. Finally, we use the plane association method in Sec. V-A to determine which plane in the map is same as plane π_i. As shown in Fig. 6(b), plane *B* associates with plane *A* because of not only the similar parameter, but also the geometric overlap. Besides, we also propose two *Double Check* criteria to improve the accuracy of plane associations:

- Ratio Test Check: The optimal associated plane error should be less than 70% of the suboptimal.
- Association Consistency Check: The plane of current scan finds the closest plane in the global plane map, and that plane finds the closest plane in the current scan. The results need to be consistent.

3) Map Plane Updating: To improve mapping consistency, we merge planes of current scan into the plane map. When the current plane finds an associated plane in the map, we add its id to the corresponding hash table and merge it into the plane map. Subsequently, we incrementally update the 3D voxels and parameters of the global plane. Otherwise, we add a new voxel to the plane map and append it to the hash table.

4) Key-frame-based Global Optimization: We perform global pose graph optimization based on key-frame poses and associated global planes for consistent mapping. As shown in Fig. 7, the global factor graph only contains optimized odometry factors that come from the sliding window and plane factors from the global plane associations. We use

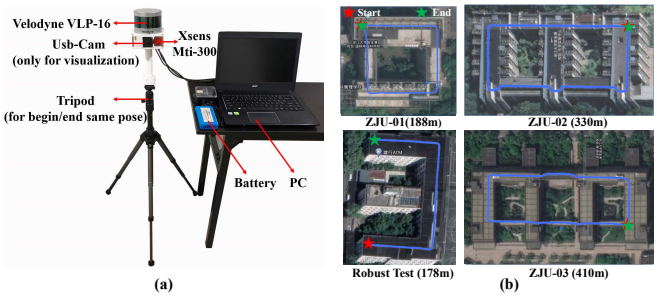


Fig. 8: (a) Our sensor suite for data collection. (b) The trajectories generated from our algorithm on ZJU sequences aligned with Google Earth.

TABLE I: The average plane extraction time cost (ms) of different algorithms.

Dataset	RANSAC Method [21]	π-LSAM [6]	Ours (Sec. IV)
ZJU-01	69.27	30.03	4.89
ZJU-02	75.53	26.89	5.41
ZJU-03	133.60	28.58	5.13
Average	92.8	28.50	5.14

iSAM2 [31] for factor graph optimization, which uses the Bayesian tree to transform the factor graph, and only needs to modify the variables related to current scan each time.

VI. Experiments

In this section, we evaluate our method on both public and self-collected datasets. All experiments are run on a computer with an Intel Core i5-10400f CPU and 16GB RAM.

A. Dataset

1) VECtor [7] Dataset: We conduct quantitative experiments on four indoor sequences of VECtor [7] dataset, which equips with an Ouster OS0-128 LiDAR and an XSens MTi-30 IMU. The *school* sequences are captured on a single floor of a school building and the *units* sequences combine empty rooms, cluttered rooms and a long straight hallway.

2) Self-collected Dataset: Our sensor suite for data collection is shown in Fig. 8(a), which consists of a Velodyne VLP-16 LiDAR at 10Hz, an Xsens Mti-300 IMU at 400Hz, a USB camera for visualization and a tripod for ensuring same poses at the starting and ending point.

We collected three representative sequences with the same starting and ending poses, named *ZJU-01*, *ZJU-02* and *ZJU-03*. The trajectories of *ZJU* sequences are shown in Fig. 8. *ZJU-01* is a clean indoor environment. One side of the corridor of *ZJU-02* is windows so that LiDAR can scan part of the outdoor scene. *ZJU-03* is a combination of indoor scenes and outdoor scenes, which is more challenging. Besides, we also collected a *Robust Test* dataset of aggressive motion for robustness evaluation.

B. Plane Extraction Evaluation

To the best of our knowledge, there are currently no open-source plane extraction algorithms specifically for LiDAR sensor. We reproduce the plane extraction method for LiDAR of π-LSAM [6], which is adapted from the open-source method plane_detector [32] with the similar region growing algorithm as ours. In addition, we also compare with plane extraction using RANSAC [21].

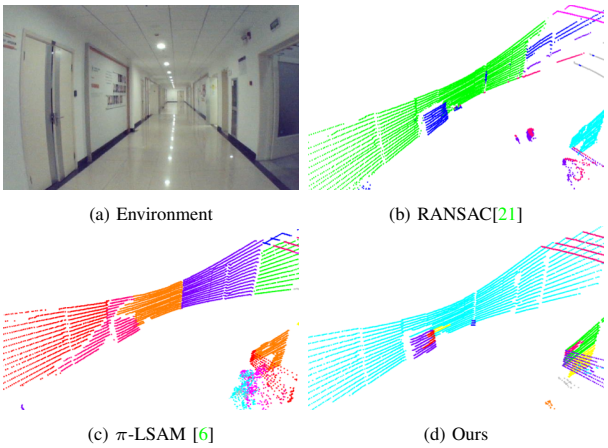


Fig. 9: Visualization results of different plane extraction algorithms. Point clouds of different colors represent different planes.

TABLE II: The translational RMSE (meters) on VECtor [7] dataset.

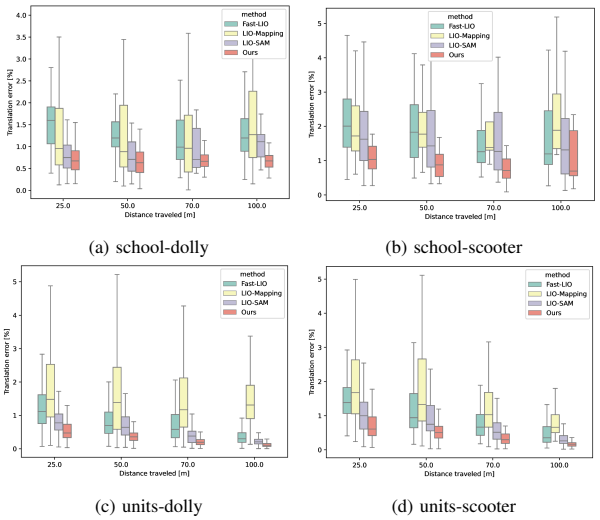
Methods	school dolly	school scooter	units dolly	units scooter
LIO-Mapping [9]	0.142	0.375	0.969	0.332
LIO-SAM [4]	0.125	0.186	0.142	0.192
FAST-LIO2 [12]	0.131	0.202	0.161	0.244
Ours	**0.112**	**0.176**	**0.080**	**0.131**

1) Qualitative Evaluation: The results of different plane extraction algorithms in a typical indoor environment are shown in Fig. 9. It can be seen that the extracted planes of our method are more complete and less mis-extracted. Because the line extraction based on LiDAR rings effectively filters out noise, while the *Line Graph* further consolidates adjacent lines into planes, resulting in more complete planes. π-LSAM [6] utilizes range image points for region growing, but growth is halted when encountering noise, leading to the segmentation of large planes. The planes obtained by the RANSAC [21] method are complete, yet this approach is time-consuming.

2) Efficiency Evaluation: We evaluate the efficiency, especially the average time consumption of different plane extraction algorithms on *ZJU* sequences. Results are listed in Table. I. The time cost of our method is 18.0% of π-LSAM [6], and 5.5% of the RANSAC [21] method. It is worth noting that the RANSAC [21] method needs more time on *ZJU-03* than other sequences. Because *ZJU-03* combines indoor and outdoor scenes, RANSAC [21] method needs more random sampling to fit planes due to fewer planes outdoors. While our method is based on region growing, which will not be affected by scenes.

C. Plane-aided SLAM Evaluation

Since recent works of fusing explicit planes in LiDAR-based SLAM [6, 19] are not open-sourced and have no ex-



(a) school-dolly (b) school-scooter

(c) units-dolly (d) units-scooter

Fig. 10: Percentage of relative translation errors of different methods on four sequences of VECtor [7] dataset.

TABLE III: The start-to-end drift error on the *ZJU* sequences.

Dataset	Error	LIO Mapping	LIO SAM	FAST LIO2	Ours
ZJU-01	Trans.(m)	1.851	3.376	3.751	**0.039**
	Rot.(°)	19.728	5.064	4.491	**0.558**
ZJU-02	Trans.(m)	4.148	3.376	6.297	**0.267**
	Rot.(°)	6.156	5.064	4.693	**0.292**
ZJU-03	Trans.(m)	3.979	7.342	4.434	**0.007**
	Rot.(°)	5.982	9.094	6.522	**0.118**

perimental results on public datasets, or do not support point clouds with distortion [17], we compare the proposed method with excellent open-source LI-SLAM algorithms, including LIO-Mapping [9], LIO-SAM [4], and FAST-LIO2 [12]. For all of these methods, we use their recommended parameters. Note that, we disable the loop closure module of all the methods for a fair comparison.

1) Results on VECtor [7] Dataset : The Absolute Position Error (APE) results are shown in Table II, in which our method achieves the best absolute accuracy among all sequences. The Relative Position Error (RPE) shown in Fig. 10 also illustrate the superiority of our algorithm. The trajectory and mapping results on the *units-dolly* sequence are shown in Fig. 1. The results are expected since our method benefits from explicit plane extractions and associations, while VECtor [7] dataset has rich planes. The plane constraint is equivalent to a global constraint, which can establish observations between planes of the current scan and historical planes.

The observed enhancement in our method within the *school* environment is not highly significant. This might be attributed to the presence of noise in the form of wall paintings, which affects our plane extraction algorithm. Furthermore, the limited range of the indoor environment in this sequence does not fully demonstrate the benefits of fusing explicit planes.

TABLE IV: The average running time (ms) of the proposed method.

Dataset	Plane Detection Sec. IV	Plane-aided Odometry Sec. V-C	Plane-aided Mapping Sec. V-D
ZJU-01	4.89	1.23	0.86
ZJU-02	5.41	1.22	0.87
ZJU-03	5.13	1.47	1.28
Average	5.14	1.31	1.00

2) Results on ZJU Sequences: Due to the unavailability of ground truth of *ZJU* dataset, we maintain the poses of the starting and ending points unchanged. Instead, we evaluate the algorithm's performance by measuring the pose difference between the starting and ending points. The start-to-end drift error of four algorithms is summarized in Table III. Our method achieves the minimum error across all sequences. Fig. 11 presents the mapping results and details, focusing on two large-scale sequences, *ZJU-02* and *ZJU-03*.

The *ZJU* sequences pose a challenge for LiDAR-based SLAM due to their long corridors. In such corridors, LiDAR has limited observations in certain directions (e.g., ceilings), resulting in insufficient data to accurately estimate the robot's pose. In contrast, our algorithm addresses this limitation by incorporating explicit planes, which effectively extend observations from a few points to an infinite number of points on the extracted planes. This augmentation provides additional information for pose estimation. Furthermore, our method employs a global plane association, which extends the matching scope beyond traditional point cloud registration. These advantages significantly reduce the drift in our SLAM system.

Notably, our method achieves a loop closure effect when reaching the end of the trajectory, utilizing explicit plane matching, resulting in nearly returning to the starting point. On the other hand, other methods that solely rely on the original point cloud struggle to achieve such loop closure detection.

3) Robustness Evaluation with Aggressive Motion: In this experiment, we assess the robustness of our system under aggressive motion using the *Robust Test* sequence, where the maximum angular velocity reaches 250 $^\circ/s$ (see Fig. 12). We qualitatively evaluate the performance of our algorithm and compare it with other methods using the mapping results, which are presented in Fig. 13. During this evaluation, FAST-LIO2 [12] fails to run on the *Robust Test* sequence. This failure can be attributed to the fact that in narrow indoor environments, relying solely on raw point clouds leads to insufficient constraints, making the system more susceptible to failure. In contrast, our method demonstrates robust performance and achieves better results than other methods under the same challenging conditions.

D. System Run Time Analysis

Since our algorithm is adapted from LIO-SAM [4], we mainly focus on the additional time consumption of our adaptations. The average single-frame running times of LIO-SAM [4] on three *ZJU* sequences are 20.84ms, 18.98ms,

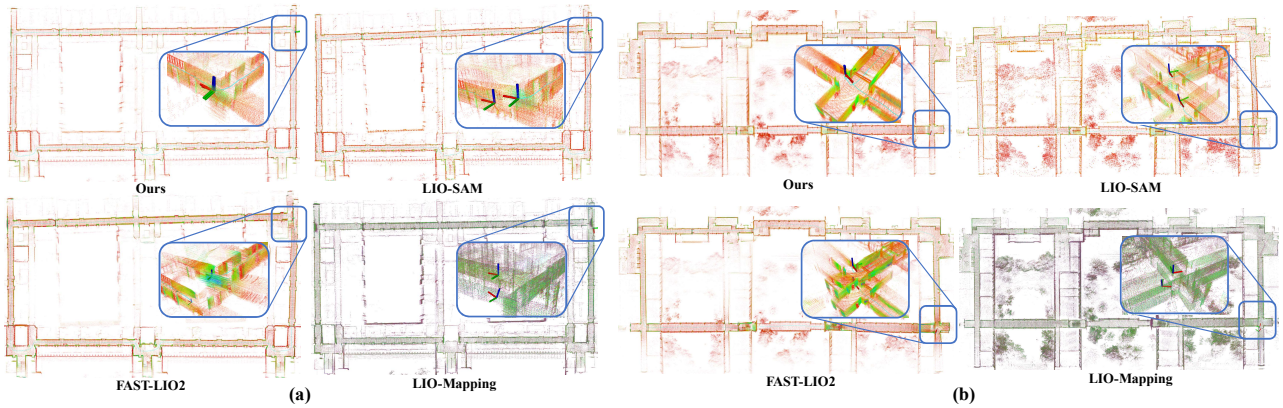


Fig. 11: (a) Mapping results on *ZJU-02* dataset. (b) Mapping results on *ZJU-03* dataset.

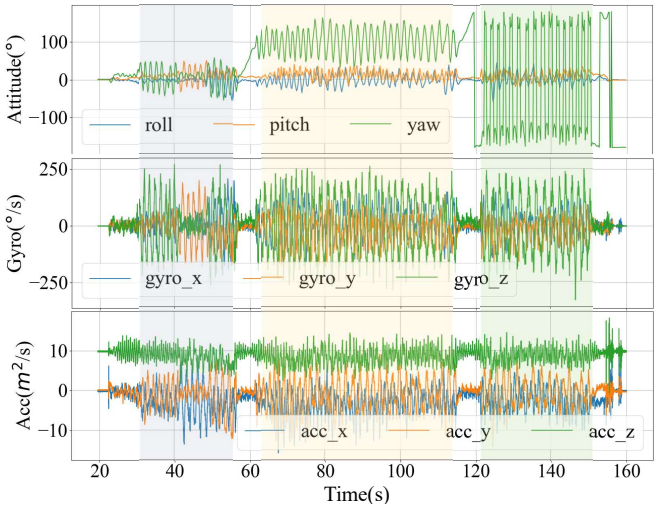


Fig. 12: The attitude, angular velocity, and acceleration of motion. The movement is usually very aggressive and can be divided into three different motion phases.

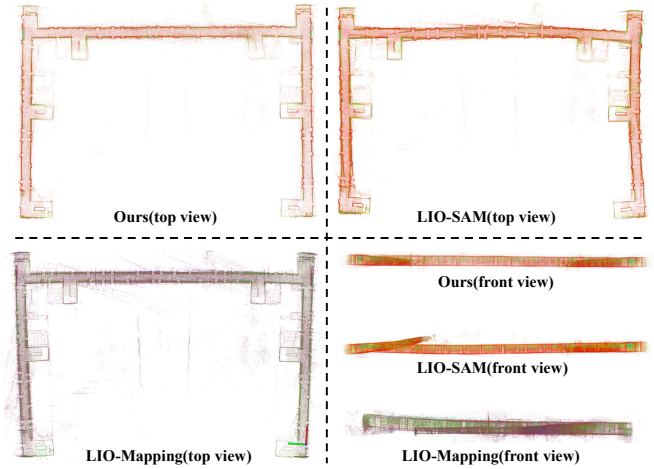


Fig. 13: Mapping results of LIO-SAM [4], LIO-Mapping [9], and our system on the *Robust Test* sequence with aggressive motions.

and 35.98ms, respectively. Table IV lists the average time consumption of our adaptations on *ZJU* sequences. It can be seen that our adaptations do not increase too much time compared to the original LIO-SAM [4], which ensures the real-time performance.

E. More Discussion

Although our method achieves better performance in planar-rich environments, it suffers from generalizability issues. We therefore only conduct experiments on indoor environment datasets containing rich planar features to demonstrate the benefits of our method, but not on more general datasets. In environments where planes are scarce, such as outdoor environments, it may not benefit from explicit planes. Under such circumstances, our method degenerates to LIO-SAM [4], although it still works fine. In addition, our plane extraction relies on the ring field, so it is only suitable for mechanical LiDAR.

VII. Conclusion and Feature Work

In this paper, we propose a plane-aided LiDAR-Inertial SLAM system, including an efficient Point→Line→Plane extraction algorithm, an accurate odometry of tightly coupled IMU, LiDAR, and explicit planes, and a global plane-aided mapping for consistency. The efficient plane extraction algorithm and key-frame-based factor graph optimization enhance the real-time performance of our system. Extensive experimental results demonstrate that our system can achieve the best overall performance among the state-of-the-art counterparts.

In the future, we will investigate methods to extract explicit lines and incorporate them into SLAM to enhance the accuracy and robustness of the system. In addition, the deep-learning-based explicit line and plane extraction method is also worth studying.

References

[1] J. Behley and C. Stachniss. “Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments.” In: *Robotics: Science and Systems*. Vol. 2018. 2018, p. 59.

[2] D. Droeschel and S. Behnke. "Efficient continuous-time SLAM for 3D lidar-based online mapping". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 5000–5007.

[3] J. Zhang and S. Singh. "LOAM: Lidar odometry and mapping in real-time." In: *Robotics: Science and Systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9.

[4] 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 international conference on intelligent robots and systems (IROS)*. IEEE. 2020, pp. 5135–5142.

[5] Z. Liu and F. Zhang. "Balm: Bundle adjustment for lidar mapping". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3184–3191.

[6] L. Zhou, S. Wang, and M. Kaess. "π-LSAM: LiDAR smoothing and mapping with planes". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 5751–5757.

[7] L. Gao, Y. Liang, J. Yang, S. Wu, C. Wang, J. Chen, and L. Kneip. "VECtor: A Versatile Event-Centric Benchmark for Multi-Sensor SLAM". In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 8217–8224.

[8] T. Shan and B. Englot. "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4758–4765.

[9] H. Ye, Y. Chen, and M. Liu. "Tightly coupled 3d lidar inertial odometry and mapping". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3144–3150.

[10] T. Qin, P. Li, and S. Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[11] W. Xu and F. Zhang. "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3317–3324.

[12] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. "Fast-lio2: Fast direct lidar-inertial odometry". In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2053–2073.

[13] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu. "Lins: A lidar-inertial state estimator for robust and efficient navigation". In: *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2020, pp. 8899–8906.

[14] K. Li, M. Li, and U. D. Hanebeck. "Towards high-performance solid-state-lidar-inertial odometry and mapping". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5167–5174.

[15] G. C. Sharp, S. W. Lee, and D. K. Wehe. "ICP registration using invariant features". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.1 (2002), pp. 90–102.

[16] J. Servos and S. L. Waslander. "Multi channel generalized-ICP". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 3644–3649.

[17] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang. "Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry". In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 8518–8525.

[18] M. Kaess. "Simultaneous localization and mapping with infinite planes". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 4605–4611.

[19] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang. "Lips: Lidar-inertial 3d plane slam". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 123–130.

[20] A. Ranganathan. "The levenberg-marquardt algorithm". In: *Tutoral on LM algorithm* 11.1 (2004), pp. 101–110.

[21] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[22] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. "On-manifold preintegration for real-time visual–inertial odometry". In: *IEEE Transactions on Robotics* 33.1 (2016), pp. 1–21.

[23] S. Hojjatoleslami and J. Kittler. "Region growing: a new approach". In: *IEEE Transactions on Image processing* 7.7 (1998), pp. 1079–1084.

[24] O. K. Smith. "Eigenvalues of a symmetric $3\times$ 3 matrix". In: *Communications of the ACM* 4.4 (1961), p. 168.

[25] M. Hsiao, E. Westman, G. Zhang, and M. Kaess. "Keyframe-based dense planar SLAM". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Ieee. 2017, pp. 5110–5117.

[26] M. Hsiao, E. Westman, and M. Kaess. "Dense planar-inertial slam with structural constraints". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6521–6528.

[27] J. Zhang, C. Zhang, J. Wu, J. Jin, and Q. Zhu. "LiDAR-Inertial 3D SLAM with Plane Constraint for Multi-story Building". In: *arXiv preprint arXiv:2202.08487* (2022).

[28] J. J. Moré. "The Levenberg-Marquardt algorithm: implementation and theory". In: *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*. Springer. 2006, pp. 105–116.

[29] F Dellaert. *Derivatives and differentials*. 2020.

[30] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. H. Gross. "Optimized spatial hashing for collision detection of deformable objects." In: *Vmv*. Vol. 3. 2003, pp. 47–54.

[31] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. "iSAM2: Incremental smoothing and mapping using the Bayes tree". In: *The International Journal of Robotics Research* 31.2 (2012), pp. 216–235.

[32] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak. "Fast plane detection and polygonalization in noisy 3D range images". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 3378–3383.