A Plane-based LiDAR Odometry Method for Man-made Scene

Zihao Yan, Peng Li, Rui Wang, and Boli Chen

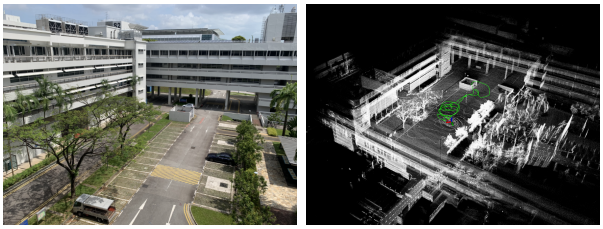
***Abstract*— In this paper, a plane-based LiDAR odometry method is proposed. SLAM is an essential part of the autonomous robotic design that provides estimated pose of a robot. Instead of using the point cloud map as in most existing works, the proposed method constructs a map consisting of a series of planes for estimating the pose in an efficient and accurate way. The plane map method reduces the number of objects processed in the map compared to point cloud map methods. Every time a LiDAR scan is received, the scan is voxelized and the planes included are extracted. The planes are matched with their counterparts in the plane map. Subsequently, the pose is optimized iteratively to get an accurate pose estimate. With the optimized pose, the plane map is updated. The effectiveness of the proposed method is verified by both public datasets and real-world experiments. The results show that the plane map-based method can achieve accurate SLAM with a processing rate of more than 20 Hz in both indoor and outdoor scenarios in comparisons with some recent LiDAR SLAM algorithms.**

I. Introduction

Simultaneous localization and mapping (SLAM) is one of the most fundamental problems in the field of autonomous robotics [1]. When a robot explores an unknown environment, SLAM is utilized to localize the robot. Comparing to visual sensors, LiDAR can achieve more reliable and accurate measurements and stay robust to illumination and weather change [2]. Therefore, LiDAR SLAM is widely used in autonomous robotic applications.

The matching of point clouds plays an essential role in LiDAR SLAM. Iterative closest points (ICP) [3] and its variants [4], [5] are the most classic method to calculate the transformation between two raw point clouds iteratively. The feature-based method introduced by LOAM [6] extract planes and edges and match the features, which enable real-time scan-to-scan registration. Some works such as LeGO-LOAM [7], LOAM LIVOX [8] and F-LOAM [9] improve LOAM and are widely used.

Although existing works on LiDAR SLAM have achieved good performance, the time cost of processing a LiDAR scan and of optimizing the pose remains high. This is because the number of points in a single LiDAR scan is large and the size of the point cloud map expands as the robot explores the surroundings.



(a) EEE environment (b) Result on eee_02 dataset

Fig. 1. Example of the proposed method on NTU VIRAL datasets. (a) shows the environment of the dataset. (b) is the mapping and pose estimation result on the eee_02 dataset.

In most man-made environments, there exist many planes available for environment reconstruction, such as the walls, the doors, the tables, and so on. In such cases, these planes can be used to build a plane map. The plane map represents the feature of the points that lie on an exact plane. Since the number of planes in a plane map is much smaller than the number of points in a point cloud map for the same scene, achieving SLAM by leveraging the plane map are prone to significantly reduce the computational cost. On the other hand, the plane-based SLAM is also memory saving for map storage. Therefore a plane map is suitable for resource-constrained robotics research when exploring an unknown man-made environment.

The extraction of the plane is the basis for constructing a plane map. The extraction algorithms can be roughly classified as model fitting-based method, clustering-based method, and region growing-based method. For the model fitting-based method, the Hough Transform [10] and the Random sample consensus (RANSAC) [11] are widely deployed, they find the best-fit plane of a set of points to realize the extraction. The clustering-based methods [12] are robust and capable for unstructured objects. For the region growing-based method [13], they choose a seed region and the clusters grow by checking the similarity of the normal vector.

In the past few years, plane-based SLAM has become more and more popular in visual SLAM. Specifically, CPA-SLAM [14] extracts the planar features by RANSAC and tracks the pose by these planes. SP-SLAM [15] achieves the estimation goal by adding parallel and perpendicular plane constraints to the system. ManhattanSLAM [16] uses structural regularities to improve the accuracy and robustness of the system. VIP-SLAM [17] utilizes the plane information and integrates it into the system to achieve accurate tracking.

In this paper, a lightweight LiDAR odometry method

Z. Yan, P. Li and R. Wang are with the School of Mechanical Engineering and Automation, Harbin Institute of Technology Shenzhen, Shenzhen, Guangdong 518055, China (e-mail: 929491173@163.com; lipeng2020@hit.edu.cn; r.wang@hit.edu.cn).

B. Chen is with the Dept. of Electronic and Electrical Engineering at University College London, UK (boli.chen@ucl.ac.uk).

Corresponding author: Peng Li. This work was supported in part by the National Natural Science Foundation of China under Grant (72101065), in part by the Guanedong Basic and Applied Basic Research Foundation (2021A1515110262. 2022A1515011274, 2021A1515110336), in part by the Shenzhen Fundamental Research Project (CYJ20210324120400003), and partially supported by the Royal Society grant, IES\R2\212041.

based on the plane map is introduced. The input LiDAR scan is firstly voxelized and the planes are extracted. With the planes, the corresponding planes in the plane map are matched. Subsequently, the cost function is constructed to optimize the pose iteratively. Afterwards, the new planes are utilized to update the plane map. The effectiveness of the proposed algorithm is examined by public datasets and experiments (Fig. 1). Main contributions of the proposed method can be summarized as follows:

- A real-time and accurate LiDAR odometry is designed without the necessity of GPU acceleration.
- A novel way of plane characterization is proposed that increases the efficiency in correspondence detection and plane updating.
- A series of planes are acting as the basic element for feature correspondence of the source and target set, which accelerates the overall process.

This paper is organized as follows: In Section II, preliminaries and the problem formulation are provided. The overview of the system pipeline is introduced with a detailed insight into its key components in Section III. The performance of the proposed method is evaluated in Section IV. Conclusion remarks are drawn in Section V.

II. Preliminaries

Given a scan $\mathcal{S}_k$ at frame k from 3D LiDAR, the task of the proposed method is to accurately estimate the pose $\mathbf{T}_k$ of the LiDAR in world frame $\{\mathrm{W}\}$ and build a geometric map $\mathcal{M}$ of the environment.

In this paper, the k-th scan of N points is denoted by $\mathcal{S}_k = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$, where $\mathbf{p}_i = (x_i, y_i, z_i)$ represents the 3D coordinates. The world frame and local frame of $\mathcal{S}_k$ is denoted by $\{\mathrm{W}\}$ and $\{\mathrm{L}_k\}$, respectively. The world frame $\{\mathrm{W}\}$ coincides with the first local frame $\{\mathrm{L}_0\}$.

The pose of the k-th scan is presented by a transformation matrix $\mathbf{T}_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$, where $\mathbf{R}_k \in SO(3)$ is a rotation matrix describing orientation and $\mathbf{t}_k \in \mathbb{R}^3$ is a translation vector describing the position of a scan.

The map is defined as a collection of m planes, i.e. $\mathcal{M} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_m\}$, where $\mathcal{P}_j$ represents a plane. A plane can be characterized by $\mathcal{P}_j = f(\mathbf{c}_j, \mathbf{n}_j, \mathbf{C}_j, N_j)$, where $\mathbf{c}_j$ is the plane center, $\mathbf{n}_j$ is the normal vector, $\mathbf{C}_j$ is the covariance matrix and N_j is the number of points. These characters are instrumental for plane matching and plane updating in the proposed algorithm.

III. Plane-based LiDAR SLAM

In this section, the proposed method is described and an overview is shown in Fig. 2. When a new scan is received, the motion distortion is compensated and the planes set $\mathcal{P}^k$ are extracted from the undistorted point cloud $\tilde{\mathcal{S}}_k$. With extracted planes and the initial pose $\bar{\mathbf{T}}_k$ given by pose prediction, the pose of this frame is estimated. Subsequently, the planes are included in the plane map. Meanwhile, the plane map is updated by merging similar planes and erasing outliers.

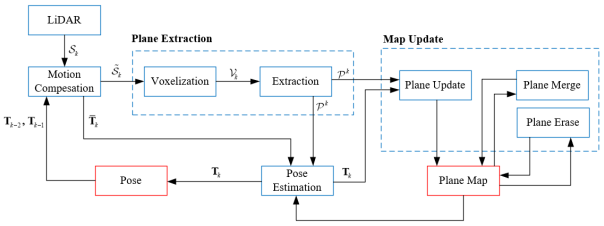


Fig. 2. System overview

A. Motion compensation and pose prediction

Due to the fact that the measurements of a LiDAR are not collected at the same time, the measurements suffer from motion distortion. For most of the existing 3D LiDAR, the sampling time between two consecutive scans is less than 10 ms. Therefore, it is reasonable to assume a constant angular velocity and a linear velocity during two consecutive scans to compensate the distortion and predict the motion.

Define the rigid transformation from frame $k-1$ to frame k as $\mathbf{T}_{k-1,k}$. Suppose $\boldsymbol{\omega}_{k-1,k}$ is the angle-axis representation of $\mathbf{R}_{k-1,k}$, $\mathbf{x}_{k-1,k} = [\boldsymbol{\omega}_{k-1,k}; \mathbf{t}_{k-1,k}]$ is used to characterize $\mathbf{T}_{k-1,k}$. The skew-symmetric matrix of $\boldsymbol{\omega}$ is defined as $[\boldsymbol{\omega}]_\times \in \mathfrak{so}(3)$ and the exponential map $exp(\cdot)$ can map an element of $\mathfrak{so}(3)$ into $SO(3)$.

The rigid transformation at the last scan S_k is denoted by $\mathbf{T}_k$. When the k-th scan $\mathcal{S}_k$ is obtained, the pose of the last two frames, i.e. $\mathbf{T}_{k-2}$ and $\mathbf{T}_{k-1}$, are used to estimate the transformation between last frame and current frame

$$\mathbf{T}_{k-1,k} = {\mathbf{T}_{k-1}}^{-1}\mathbf{T}_k = {\mathbf{T}_{k-2}}^{-1}\mathbf{T}_{k-1}. \tag{1}$$

The start time and end time of scan $\mathcal{S}_k$ are denoted by t_k^s and t_k^e, respectively. For the i-th point $\mathbf{p}_i$ in the scan $\mathcal{S}_k$ recorded within the time $t_i \in [t_k^s, t_k^e]$, a time ratio variable is defined as $s = \frac{t_i - t_k^s}{t_k^e - t_k^s}$. The rigid transformation $\mathbf{T}_{k-1,t_i}$ can be estimated by linear interpolation

$$\begin{aligned} \mathbf{R}_{k-1,t_i} &= exp(s[\boldsymbol{\omega}_{k-1,k}]_\times), \\ \mathbf{t}_{k-1,t_i} &= s\mathbf{t}_{k-1,k}. \end{aligned} \tag{2}$$

where $\mathbf{R}_{k-1,t_i}$ is the rotation matrix of $\mathbf{T}_{k-1,t_i}$ and $\mathbf{t}_{k-1,t_i}$ is the translation vector of $\mathbf{T}_{k-1,t_i}$. The distortion of current scan $\mathcal{S}_k$ can be corrected as

$$\tilde{\mathcal{S}}_k = \{{\mathbf{R}_{k-1,k}}^{-1}(\mathbf{T}_{k-1,t_i}\mathbf{p}_i - \mathbf{t}_{k-1,k}) \mid \mathbf{p}_i \in \mathcal{S}_k\}. \tag{3}$$

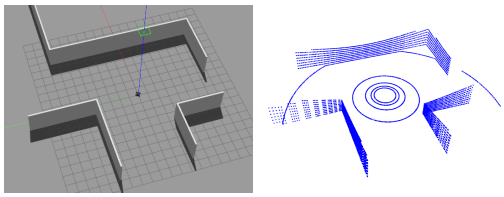
The pose of frame k is predicted as

$$\bar{\mathbf{T}}_k = \mathbf{T}_{k-1}\mathbf{T}_{k-1,k}. \tag{4}$$

B. Plane extraction

After compensating the distortion of the point cloud in the environment (Fig. 3a), the next step is to extract planar features from the undistorted point cloud $\tilde{\mathcal{S}}_k$ (Fig. 3b). The 3D point cloud is voxelized and the voxel containing a plane is captured(Fig. 3c). Then voxels with similar planar features are clustered through region growth (Fig. 3d).

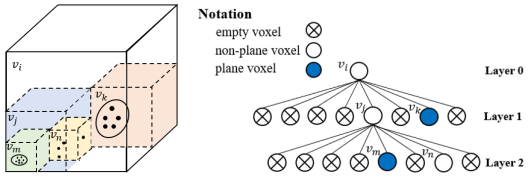
1) Voxelization: Inspired by VoxelMap [18], voxelization can be achieved in a coarse-to-fine manner. Instead of



(a) Environment (b) Point cloud

(c) Voxelization (d) Planes

Fig. 3. Plane extraction process. For a robot with 3D LiDAR in the environment (a), it can receive the point cloud (b). The point cloud is voxelized and transformed into voxel cloud (c). Cluster similar planes voxel, planes in the point cloud is extracted (d).



(a) Distribution of points (b) Octotree structure

Fig. 4. Illustration of voxelization. Assume the maximum number of layers is 2. Given a voxel v_i in layer 0 with points lying on different planes, divide the voxel into eight octants. In layer 1, points in v_k form a plane, so the planar characters are recorded. The non-plane voxel v_j is divided again and plane voxel v_m is recorded. Since layer 2 is the maximum number of layers, non-plane voxel v_n is not divided. (b) presents the structure of voxel v_i.

constructing a voxel map, an adaptive voxel point cloud is constructed for plane extraction. In the proposed method, the space of local frame $\{\mathrm{L}_k\}$ is decomposed into voxels with a coarse map resolution. In such case, all points in undistorted point cloud $\tilde{\mathcal{S}}_k$ are distributed into voxels according to their positions. Denoting a voxel as $v_i = \{\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2, \cdots, \tilde{\mathbf{p}}_{N_i}\}$, the covariance matrix $\mathbf{C}$ of the points in a voxel v_i is

$$\begin{aligned} \mathbf{C}_{v_i} &= \frac{1}{N_i}\sum_{k=1}^{N_i}\left(\tilde{\mathbf{p}}_k - \bar{\mathbf{p}}_{v_i}\right)\left(\tilde{\mathbf{p}}_k - \bar{\mathbf{p}}_{v_i}\right)^T, \\ \bar{\mathbf{p}}_{v_i} &= \frac{1}{N_i}\sum_{k=1}^{N_i}\tilde{\mathbf{p}}_k. \end{aligned} \tag{5}$$

For each voxel with points from the undistorted point cloud, points in the voxel form a plane if the minimum eigenvalue $\lambda_{\min}(\mathbf{C}_{v_i})$ of $\mathbf{C}_{v_i}$ is less than a predefined threshold. Otherwise, the voxel is regarded as a non-plane voxel. The covariance matrix is calculated and non-plane voxels are divided into eight octants. Such covariance matrix calculation and voxel division process are repeated until the number of layers reaches the maximum limit. If voxel v_i forms a plane, the planar characters covariance matrix $\mathbf{C}_{v_i}$, plane center $\mathbf{c}_{v_i} = \bar{\mathbf{p}}_{v_i}$, plane normal vector $\mathbf{n}_{v_i} = \mathbf{u}_{\min}(\mathbf{C}_{v_i})$, and number of points $N_{v_i} = N_i$ are recorded, where $\mathbf{u}_{\min}(\mathbf{C}_{v_i})$ is the eigenvector corresponding to the minimum eigenvalue $\lambda_{\min}(\mathbf{C}_{v_i})$. Fig. 4 illustrates the voxelization process. As a result, a voxel point cloud $\mathcal{V}_k$ is generated (Fig. 3c).

2) Cluster: Similar plane features are clustered according to their normal vector and locality through region growth. A cluster corresponds to a plane in the point cloud. In the proposed method, the voxel is considered as a seed. Due to the fact that the number of voxels in a point cloud is comparatively smaller than the number of points and the spatial structure of the voxel allows a rapid search of its adjoining voxels, the planes can be extracted efficiently.

The plane voxel v_s with the most number of points N_s is selected as the root to begin plane searching and its neighbor voxels are pushed into a search set S. The normal vector and the plane center of the plane voxel v_s are denoted as n_s and c_s, respectively. Every plane voxel v_t in the search set S with the normal vector n_t and the plane center c_t is regarded as a candidate voxel to be matched and removed from the search set S. The angle between the normal vector of v_s and v_t and the projections of the center vector $\mathbf{c}_s - \mathbf{c}_t$ on both normal vectors are calculated, which are used to cluster similar plane voxel. Specifically, the plane voxel v_t is classified into a cluster if the following conditions hold

$$\begin{cases} \mathbf{n}_s^T\mathbf{n}_t > \tau_\theta, \\ \mathbf{n}_i^T(\mathbf{c}_s - \mathbf{c}_t) < \tau_d, \ i \in \{s, t\}. \end{cases} \tag{6}$$

where τ_θ is the angular threshold and τ_d is the distance threshold that are properly chosen. If the conditions in (6) are satisfied, the voxel v_t is added to the current cluster. In the meantime, the adjoining voxels of v_t are added to the search set S. Otherwise, the voxel v_t is ignored. The calculation is repeated until the search set S is empty. The updating rules of the planar characters will be analyzed in Section III-D. Subsequently, a new unclustered seed voxel is chosen to repeat the clustering process until all voxels in $\mathcal{V}_k$ have been checked. After plane extraction, a set of planes $\mathcal{P}^k = \{\mathcal{P}_0^k, \mathcal{P}_1^k, \cdots, \mathcal{P}_n^k\}$ is generated (see Fig. 3d).

C. Pose estimation

In order to estimate the pose of the LiDAR correctly, the proposed method aims to calculate the transformation between the current plane set $\mathcal{P}^k$ and plane map $\mathcal{M}$ iteratively. At the i-th iteration, the pose of the k-th frame is defined as $\mathbf{T}_k^i$ and the best estimation $\mathbf{T}_k^{i\,*}$ is used to initialize the next iteration $\mathbf{T}_k^{i+1} = \mathbf{T}_k^{i\,*}$. To be noted, the pose of the 0-th iteration is initialized as $\mathbf{T}_k^0 = \bar{\mathbf{T}}_k$, where $\bar{\mathbf{T}}_k$ is from (4). At the beginning of each iteration, all the planes in $\mathcal{P}^k$ are transformed from the LiDAR frame $\{\mathrm{L}_k\}$ to the world frame $\{\mathrm{W}\}$ by $\mathbf{T}_k^i$. For an arbitrary plane $\mathcal{P}_a^k$ in $\mathcal{P}^k$, its planar characters after transformation are given by

$$\begin{aligned} \tilde{\mathbf{c}}_a &= \mathbf{R}_k^i\mathbf{c}_a + \mathbf{t}_k^i, \\ \tilde{N}_a &= N_a, \\ \tilde{\mathbf{n}}_a &= \mathbf{R}_k^i\mathbf{n}_a, \\ \tilde{\mathbf{C}}_a &= \mathbf{R}_k\mathbf{C}_a{\mathbf{R}_k}^T. \end{aligned} \tag{7}$$

The plane set after transformation is defined as $\tilde{\mathcal{P}}^k$. With these new planar characters, the next step is to characterize

the counterparts of the plane set $\tilde{\mathcal{P}}^k$ in plane map $\mathcal{M}$. For any nearby planes pair $\mathcal{P}_l$ and $\tilde{\mathcal{P}}_a^k$ in the plane map $\mathcal{M}$ and $\tilde{\mathcal{P}}^k$ meet the angular condition in (6), a similarity metric is defined to further determine the best match

$$m = \alpha\|\mathbf{c}_{\mathcal{P}_l} - \tilde{\mathbf{c}}_a\| + \beta[\mathbf{n}_{\mathcal{P}_l}^T(\mathbf{c}_{\mathcal{P}_l} - \tilde{\mathbf{c}}_a)] + \gamma(1 - \mathbf{n}_{\mathcal{P}_l}^T\tilde{\mathbf{n}}_a). \quad (8)$$

where α, β, and γ are user-defined scaling characters. Each plane in the plane set $\tilde{\mathcal{P}}^k$ corresponds to the plane in plane map $\mathcal{M}$ with the minimum value of the metric. The best match of plane $\tilde{\mathcal{P}}_a^k$ in plane map $\mathcal{M}$ is denoted as $\mathcal{P}_{\mathcal{P}_a^k}$ for convenience. It is worth noting that $\mathcal{P}_{\mathcal{P}_a^k}$ is also the match of plane $\mathcal{P}_a^k$. The metric in (8) makes it possible to find planes with similar characters and to exclude those outliers parallel to plane $\mathcal{P}_{\mathcal{P}_a^k}$ but on another object.

With the matched planes, the pose is estimated by minimizing the sum of the point-to-plane error

$$\mathbf{T}_k^{i\,*} = \arg\min_{\mathbf{T}_k^i} \sum_{\mathcal{P}_a^k \in \mathcal{P}^k} \sum_{j=1}^{N_a} \mathbf{n}_{\mathcal{P}_{\mathcal{P}_a^k}}^T (\mathbf{T}_k^i \mathbf{p}_{a,j} - \mathbf{c}_{\mathcal{P}_{\mathcal{P}_a^k}}). \quad (9)$$

where $\mathbf{p}_{a,j}$ is the j-th point of plane $\mathcal{P}_a^k$ from plane set $\mathcal{P}^k$, $\mathbf{c}_{\mathcal{P}_{\mathcal{P}_a^k}}$ and $\mathbf{n}_{\mathcal{P}_{\mathcal{P}_a^k}}$ are the center and normal vector of plane $\mathcal{P}_{\mathcal{P}_a^k}$ in plane map $\mathcal{M}$. The Jacobian matrix of the error can be calculated by applying left perturbation model [19]

$$\mathbf{J} = \mathbf{n}_{\mathcal{P}_{\mathcal{P}_a^k}}^T \begin{bmatrix} \mathbf{I}_{3\times3} & -[\mathbf{T}_k^i \mathbf{p}_{a,j}]_\times \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} \end{bmatrix}. \quad (10)$$

The nonlinear optimization in (9) can be solved by the Gauss-Newton method. In each iteration, the Jacobian matrix of each residual is computed and the pose of the LiDAR is estimated by the Ceres Solver library[1].

D. Map update

With the optimized pose $\mathbf{T}_k^{i\,*}$, the plane map is updated by including new observations, merging similar planes and erasing unstable planes. For a plane with new corresponding planes, the planar characters are updated.

1) Planar characters update: In the flow of plane extraction (III-B.2), plane update (III-D.2), and plane merge (III-D.3), it is necessary to update the planar characters. Instead of calculating the new covariance matrix $\mathbf{C}_{new}$ with all the points on both planes, the covariance matrices from both planes are utilized to avoid time-consuming calculations with massive points. Specifically, for similar pairs plane $\mathcal{P}_a$ and plane $\mathcal{P}_b$ with plane centers $\mathbf{c}_a$ and $\mathbf{c}_b$, number of points N_a and N_b and covariance matrices $\mathbf{C}_a$ and $\mathbf{C}_b$, respectively. The planar characters are updated as follows

$$\begin{aligned} \mathbf{c}_{new} =& \frac{1}{(N_a + N_b)}(N_a\mathbf{c}_a + N_b\mathbf{c}_b), \\ N_{new} =& N_a + N_b, \\ \mathbf{C}_{new} =& \frac{1}{N_{new}}[N_a(\mathbf{C}_a + \mathbf{c}_a\mathbf{c}_a^T) + N_b(\mathbf{C}_b + \mathbf{c}_b\mathbf{c}_b^T)] \\ & - \mathbf{c}_{new}\mathbf{c}_{new}^T, \\ \mathbf{n}_{new} =& \mathbf{u}_{\min}(\mathbf{C}_{new}). \end{aligned} \quad (11)$$

[1]https://github.com/ceres-solver/ceres-solver

2) Plane update: After pose estimation, the plane map is updated with optimized pose $\mathbf{T}_k^{i\,*}$ and the plane set $\mathcal{P}^k$ of frame k. Initialized by the first plane set $\mathcal{P}^0$, the plane map is updated with successive plane sets. All the planes in $\mathcal{P}^k$ are transformed from LiDAR frame $\{\mathrm{L}_k\}$ to world frame $\{\mathrm{W}\}$ by $\mathbf{T}_k^{i\,*}$ obtaining $\tilde{\mathcal{P}}^k$. To find the corresponding plane for an arbitrary plane $\tilde{\mathcal{P}}_a^k$ in $\tilde{\mathcal{P}}^k$, any plane in map $\mathcal{M}$ is considered as a candidate corresponding plane if the following similarity conditions are verified

- The angle between the normal vectors of the two planes is lower than a threshold.
- The projections of the center vector on both planar surfaces should below a certain threshold.
- The projections of the center vector between two centers on both normal vectors should below a certain threshold.

The center of candidates are projected to plane $\tilde{\mathcal{P}}_a^k$, among which the plane with shortest projection is chosen as the correspondence. The planar characters are updated as in (11). In addition, a plane without any candidate plane is regarded as a new plane and included in the plane map $\mathcal{M}$.

3) Plane merge: In the plane map, some tiny planes that lie on the edge of existing corresponding planes are detected in III-D.2. As the number of planes in the plane map increases, such tiny planes are to be merged into nearby planes according to the similarity conditions in III-D.2. As a result, tiny planes are removed and the plane map can be streamlined, which is beneficial for efficient calculation.

4) Plane erase: Considering the fact that the motion compensation and the similar plane cluster are not always perfect, the outliers are erased to increase the robustness of the proposed method and avoid unnecessary calculations. If the number of points on a plane is less than 100 and the plane is not observed by the past 10 frames, the plane is regarded as an outlier. As a result, the outliers are erased as they are considered to be inactive in the plane map.

E. Tightly-coupled LiDAR inertial odometry

To enhance the tracking accuracy of moving UAVs, the proposed method can be integrated into tightly-coupled SLAM frameworks such as LIO-SAM [20]. The point-to-plane error in (9) can be regarded as a LiDAR odometry factor that is jointly optimized with the IMU preintegration factor and the loop closure factor. Due to space limitations, readers are advised to refer to [20] for details.

IV. Experiments

In this section, the performance of the proposed method is evaluated in different real-world scenarios, including outdoor and indoor scenes with different types of LiDARs. All the experiments are conducted on a laptop equipped with Intel Core i7-7700HQ @ 2.8 GHz and 16 GB RAM.

A. Indoor & outdoor experiments

The datasets are collected with a handheld device, which is composed of a 16 channels RoboSense LiDAR. The datasets are collected in a narrow corridor (see Fig. 5(a)), a parking

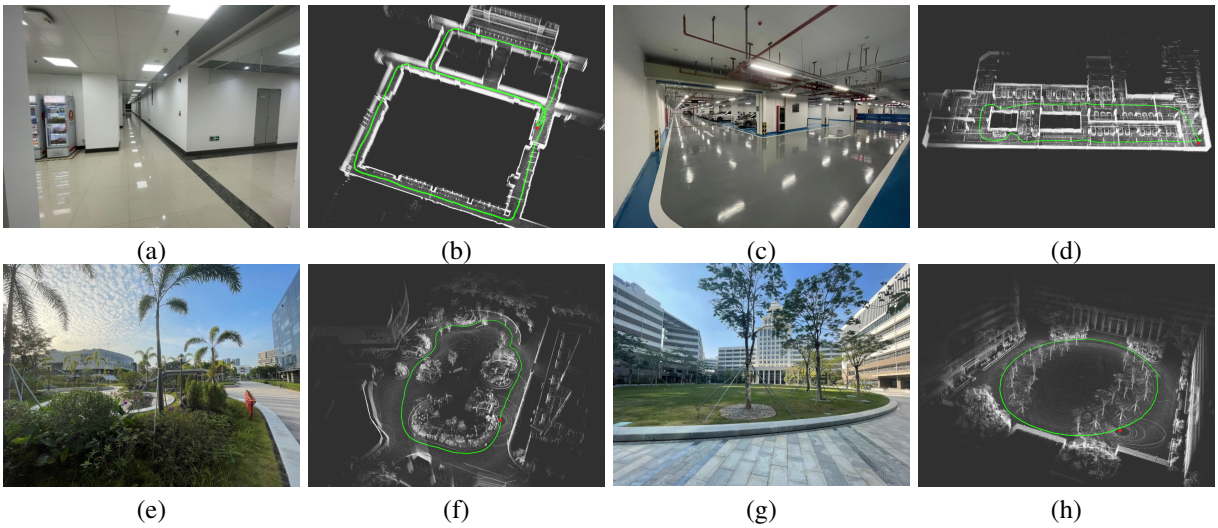


Fig. 5. Real environment and map generated by the proposed method, the green line is the estimated trajectory and the red point is the start and end point. (a)-(b) is a corridor. (c)-(d) is a parking lot. (e)-(f) is a garden. (g)-(h) is a square.

lot with moving vehicles (Fig. 5(c)), a garden with plants and stones (Fig. 5(e)), and a square in the outdoor (Fig. 5(g)).

With unknown ground truth of the trajectory, datasets are collected via closed trajectories. The results are compared with LOAM [6], BALM [21] and F-LOAM [9]. The End-to-End error and the average computation time are shown in Table I. The results show that the proposed method shows comprehensive advantages in accuracy and computation time in the indoor and outdoor experiments. Although BALM has a better average computation time in the Square experiment, the proposed method is more robust in various environments. The proposed method has a processing rate of higher than 20 Hz, which is comparable to BALM and F-LOAM. Taking the computation time of the proposed method in the Garden experiment as an example, the average runtimes of each module are shown in Table II. The preprocess module includes the point cloud input process and motion compensation process. The overall method has an average runtime of 30.586 ms and a standard deviation of 8.724 ms in the Garden experiment.

TABLE I

COMPARISON OF DIFFERENT ALGORITHMS ON THE INDOOR & OUTDOOR EXPERIMENTS (**END-TO-END ERROR (M)/AVERAGE COMPUTATION TIME (MS)**).

Experiment	A-LOAM	BALM	F-LOAM	Proposed
Corridor	0.19/31.52	0.27/30.88	0.11/47.67	**0.03/21.13**
Parking	0.11/44.28	0.29/44.28	0.06/84.38	**0.03/31.01**
Garden	0.13/64.08	-/-	-/-	**0.05/30.59**
Square	0.49/75.07	0.09/**21.38**	-/-	**0.06**/40.50

- The best result is highlighted in **bold**.
- "-" means fail.

B. Experiments with IMU

The NTU VIRAL is a sequence of datasets collected from an Unmanned Aerial Vehicles [22]. These datasets are collected in several challenging indoor and outdoor conditions. The metric used in this case is the Absolute Trajectory Error (ATE) [23], which compares the absolute distances between the estimated and the ground truth trajectory. In the experiments, the data collected by the horizontal LiDAR is regarded as the input LiDAR scans.

TABLE II

AVERAGE COMPUTATION COST FOR EACH MODULE IN THE GARDEN EXPERIMENT

Module	Average (ms)	Standard Deviation (ms)
Preprocess	1.627	0.337
Plane Extraction	7.399	3.138
Pose Estimation	15.338	3.749
Map Update	6.222	2.827
Total	30.586	8.724

The proposed method is compared with LOAM [6], LIO-SAM [20], and MILIOM [24]. Since LIO-SAM and MILIOM use IMU data to reduce the influence of UAVs' fast motion, both the results of the LiDAR-only version and the tightly-coupled LiDAR inertial version of the proposed method are shown for a fair comparison. Only the 1-LiDAR results are considered for all methods. For notation simplicity, the LiDAR-only version of the proposed method is denoted by "**Proposed-L**" and the tightly-coupled LiDAR inertial version of the proposed method is denoted by "**Proposed**". The results are summarized in Table III. As can be seen, the proposed method is able to achieve better performance in most of the NTU VIRAL datasets. It can be noticed that all the ATE of the proposed method is below 10cm when the IMU is integrated into the system.

The runtime of the proposed method in tightly-coupled LiDAR inertial version is compared with open-source method A-LOAM[1] and LIO-SAM[2]. The results are shown in Table IV. The proposed method is able to provide accurate and effective pose estimation at an average processing rate of more than 20Hz. For MILIOM, only the computation time

[1]https://github.com/HKUST-Aerial-Robotics/A-LOAM
[2]https://github.com/TixiaoShan/LIO-SAM

TABLE III

ATE (M) OF THE STATE-OF-THE-ART LIDAR-BASED SLAM METHODS OVER THE NTU VIRAL DATASETS.

Dataset	ALOAM	LIO-SAM	MILIOM	Proposed-L	Proposed
eee_01	0.224	0.092	0.104	0.245	**0.081**
eee_02	0.189	0.082	**0.065**	0.208	0.069
eee_03	0.152	0.118	**0.063**	0.217	0.090
nya_01	0.080	0.090	0.083	0.226	**0.080**
nya_02	0.093	0.107	0.072	0.227	**0.072**
nya_03	0.084	0.366	**0.058**	0.253	0.081
sbs_01	0.195	0.097	0.076	0.222	**0.074**
sbs_02	0.087	0.096	0.081	0.211	**0.076**
sbs_03	0.346	0.096	0.088	0.246	**0.083**

• The best result is highlighted in **bold**.

when running the eee_02 dataset is shown in [24]. The frontend processing time of MILIOM has a mean of 14.56 ms and the backend processing time of MILIOM has a mean of 62.99 ms. While the proposed method takes only 39.40 ms to get similar accuracy in the case of eee_02 dataset.

TABLE IV

RUNTIME COMPARISON ON NTU VIRAL DATASET.

Dataset	A-LOAM	LIO-SAM	Proposed
eee_01	221.78 ms	186.13 ms	**42.48 ms**
eee_02	183.04 ms	136.91 ms	**39.40 ms**
eee_03	162.69 ms	117.76 ms	**42.77 ms**
nya_01	115.51 ms	101.01 ms	**33.73 ms**
nya_02	118.64 ms	105.78 ms	**33.22 ms**
nya_03	121.66 ms	107.78 ms	**33.91 ms**
sbs_01	186.80 ms	135.48 ms	**35.02 ms**
sbs_02	188.34 ms	146.64 ms	**35.60 ms**
sbs_03	195.52 ms	151.66 ms	**36.02 ms**

• The best result is highlighted in **bold**.

V. CONCLUSION

In this paper, a lightweight and accurate LiDAR odometry method based on the plane map is proposed. Instead of using the point clouds to construct the map, a plane-based map is constructed for pose estimation. Thanks to the fact that the number of planes in the map is much smaller than that of points, the computational cost is significantly reduced. Meanwhile, the featured characters of a plane are saved to the plane map. Therefore the time for character updating and plane searching can be reduced. The proposed method can save both time and memory, making it suitable for some resource-constrained robotic applications. Furthermore, experiments in indoor and outdoor environments are conducted to verify the efficiency and accuracy of the proposed method with comparisons to state-of-the-art LiDAR SLAM algorithms and frameworks, including F-LOAM, LIO-SAM, LOAM, and BALM.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] C. Debeunne and D. Vivet, "A review of visual-lidar fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, 2020.

[3] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[4] A. V. Segal, D. Hähnel, and S. Thrun, "Generalized-icp," in *Robotics: Science and Systems*, 2009.

[5] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 145–152.

[6] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.

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

[8] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3126–3131.

[9] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-loam : Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4390–4396.

[10] D. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[11] 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, p. 381–395, jun 1981.

[12] J. M. Biosca and J. L. Lerma, "Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 63, no. 1, pp. 84–98, 2008, theme Issue: Terrestrial Laser Scanning.

[13] A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto, "Octree-based region growing for point cloud segmentation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 104, pp. 88–100, 2015.

[14] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "Cpa-slam: Consistent plane-model alignment for direct rgb-d slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1285–1291.

[15] X. Zhang, W. Wang, X. Qi, Z. Liao, and R. Wei, "Point-plane slam using supposed planes for indoor environments," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[16] R. Yunus, Y. Li, and F. Tombari, "Manhattanslam: Robust planar tracking and mapping leveraging mixture of manhattan frames," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6687–6693.

[17] D. Chen, S. Wang, W. Xie, S. Zhai, N. Wang, H. Bao, and G. Zhang, "Vip-slam: An efficient tightly-coupled rgb-d visual inertial planar slam," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5615–5621.

[18] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8518–8525, 2022.

[19] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[20] 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)*, 2020, pp. 5135–5142.

[21] Z. Liu and F. Zhang, "Balm: Bundle adjustment for lidar mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.

[22] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, "Ntu viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 270–280, 2022.

[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[24] T.-M. Nguyen, S. Yuan, M. Cao, L. Yang, T. H. Nguyen, and L. Xie, "Miliom: Tightly coupled multi-input lidar-inertia odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5573–5580, 2021.