sensors

MDPI

Article

LiDAR Inertial Odometry Based on Indexed Point and Delayed Removal Strategy in Highly Dynamic Environments

Weizhuang Wu * and Wanliang Wang

College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China
* Correspondence: 2112012070@zjut.edu.cn

Abstract: Simultaneous localization and mapping (SLAM) is considered a challenge in environments with many moving objects. This paper proposes a novel LiDAR inertial odometry framework, LiDAR inertial odometry-based on indexed point and delayed removal strategy (ID-LIO) for dynamic scenes, which builds on LiDAR inertial odometry via smoothing and mapping (LIO-SAM). To detect the point clouds on the moving objects, a dynamic point detection method is integrated, which is based on pseudo occupancy along a spatial dimension. Then, we present a dynamic point propagation and removal algorithm based on indexed points to remove more dynamic points on the local map along the temporal dimension and update the status of the point features in keyframes. In the LiDAR odometry module, a delay removal strategy is proposed for historical keyframes, and the sliding window-based optimization includes the LiDAR measurement with dynamic weights to reduce error from dynamic points in keyframes. We perform the experiments both on the public low-dynamic and high-dynamic datasets. The results show that the proposed method greatly increases localization accuracy in high-dynamic environments. Additionally, the absolute trajectory error (ATE) and average RMSE root mean square error (RMSE) of our ID-LIO can be improved by 67% and 85% in the UrbanLoco-CAMarketStreet dataset and UrbanNav-HK-Medium-Urban-1 dataset, respectively, when compared with LIO-SAM.

Keywords: LiDAR-inertial SLAM; dynamic objects; sliding window; dynamic scenarios

Citation: Wu, W.; Wang, W. LiDAR Inertial Odometry Based on Indexed Point and Delayed Removal Strategy in Highly Dynamic Environments. *Sensors* **2023**, *23*, 5188. https://doi.org/10.3390/s23115188

Academic Editors: Chee Kiat Seow, Henrik Hesse, Yanliang Zhang, Torr Polakow and Kai Wen

Received: 17 April 2023
Revised: 10 May 2023
Accepted: 11 May 2023
Published: 30 May 2023

1. Introduction

Mobile robots and autonomous driving both require accurate and reliable state estimation. It is challenging for traditional GNSS/INS to achieve high accuracy localization requirements in indoor and urban environments.

In recent years, some developments have been made in both visual and LiDAR SLAM. Due to the wide field of view and detailed texture information that cameras provide, vision-base slam can estimate a six-degree of freedom (6-DOF) state. However, it is easily affected by lighting shifts and low-texture and repetitive-texture environments [1,2]. Compared to cameras, LiDAR has more direct, accurate and reliable depth information for SLAM systems, allowing for robust state estimations even in challenging environments [3].

1.1. Localization and Mapping Based on LiDAR and Inertial

The most widely used SLAM technique in urban areas is a localization and mapping algorithm based on LiDAR [4], because LiDAR can accurately estimate the depth of the point cloud. We can categorize these investigations using representations of LiDAR measurements as a basis. The most widely used and basic method, the iterative closest point approach (ICP) [5], uses LiDAR point clouds directly. Recent studies have proposed a variety of representations, such as grid cells, features of LiDAR point cloud and surface elements. The primary distinction between these algorithms is matching search. In particular, to increase computational efficiency, variations in ICP approaches match associated points using downsampled points and various data structures, such as kd-trees, ikd-trees,

incremental voxels, and projection depth maps [6–8]. The use of projection geometry for a nearest-neighbor search was proposed in [7,9]. SuMa, a powerful 3D LiDAR-based SLAM system that employs a shader language for parallel processing, was proposed by Behley et al. [9]. Using the input point clouds, they executed a spherical image projection and correlated it, treating these points as the closest points. In addition, this paper indicated that motion drift can be greatly decreased through surface element-based map creation and matching. Deep learning, in recent years, has made significant advancements. Many scholars have developed approaches based on deep learning [10,11] for LiDAR motion estimation using projected LiDAR features.

Zhang et al. [12] proposed LiDAR odometry and mapping (LOAM), which is a LiDAR slam solution based on LiDAR features. Instead of using complete point clouds in ICP, planar and edge point features are matched in LOAM to provide real-time and low-drift state estimation. This consists of the odometry system and the mapping system. The odometry module performs point-to-edge/plane matching at a high frequency, and the mapping module runs at a lower frequency to perform scan-to-map matching, which can obtain more precise pose estimations. During the nearest-neighbor search in the former module, kd-trees are built for each feature set from each LiDAR scan. The mapping module uses these LiDAR features to construct a global map and optimize poses further. In the KITTI odometry benchmark site, LOAM has achieved the best performance [13]. In order to further enhance performance, numerous LiDAR slam frameworks based on LOAM have been proposed. F-LOAM [14] uses a non-iterative, two-stage distortion compensation approach to boost the system's real-time performance and computational efficiency. Nevertheless, there is still no sub-module for global optimization, such as a loop detection strategy. This will lead to significant cumulative errors over long periods of time in large-scale environments. HDL SLAM [15] is a multi-sensor fusion framework that can combine LiDAR, IMU and GNSS sensors; however, the reliability of the frame-to-frame registration is low because it is based on the NDT algorithm. LeGo-LOAM [16] extracts two types of points in the feature extraction module: ground and non-ground points. This significantly improves feature registration efficiency. Then, the LiDAR mapping module receives its initial value via two-step registration. Moreover, a keyframe selection strategy and loopback detection method are added to the backend for better global localization consistency and real-time performance. However, because of the loose coupling of IMU and LiDAR, running in a wide range of scenarios still leads to large cumulative errors. Liu et al. [17] proposed a feature extraction method and computed descriptors based on deep learning. This method uses two-stage state estimation and experiments in a long-range environment. Ye et al. [18] proposed a tightly coupled LiDAR SLAM algorithm, LIO-Mapping, built on VINS and LOAM. For feature extraction and state estimation, a LiDAR front-end component in LOAM takes the role of the visual front-end component in VINS-Mono. Unfortunately, due to the size of the optimization issue, real-time execution on an unmanned vehicle is challenging. LiLi-OM [19] presents an adaptive keyframe selection strategy that can be used for solid-state LiDAR and conventional LiDAR. In addition, it introduces a method with metric weights for sensor fusion. To further increase processing speed and trajectory accuracy, LIO-SAM [20] represents a tightly coupled LiDAR-inertial odometry system based on the framework of incremental smoothing and mapping (iSAM2) [21]. The factor graph optimization can also incorporate GPS and loop closure factors. An algorithm for loosely coupled adaptive covariance estimation was proposed by Zhang et al. [22]. Better experimental results are achieved compared to the LIO system with a tightly coupled framework. Liu et al. [3] proposed a feature extraction algorithm to extract rod-shaped and planar features, which reduces computational consumption and improves the accuracy of the LIO.

In addition to optimization-based LIO systems, filter-based LIO systems have emerged in recent years. The error state Kalman filter is used in [23] to propose a filter-based method for LiDAR-inertial odometry that estimates the robot's pose state. FAST-LIO [24] represents a new method to calculate the Kalman gain to accelerate the system. FAST-LIO2 [25] directly uses the original LiDAR points to perform scan-to-map registration and proposes the ikd-Tree structure to reduce the time consumption of map updates compared with the static k-d tree structure. Faster-LIO [26] has an iVox structure to organize voxels through a hash table and uses a LRU cache to realize map point addition and deletion, whose parallel accelerated version can achieve a performance that completely surpasses the performance of ikd-Tree. However, these existing SLAM systems are assumed to perform in a static environment. In fact, when an autonomous system navigates in realistic environments, the spatial structure will become more complex as more moving objects, such as moving people and cars, enter the environment. Therefore, for pose estimation and mapping, the online removal of dynamic objects is essential.

1.2. Dynamic Point Removal Approaches in SLAM

In the real world, moving objects such as cars and pedestrians are common. However, the excellent SLAM systems mentioned above are designed assuming a static environment and cannot robustly perform in dynamic scenes. Moving objects must be recognized and eliminated from the LiDAR point clouds in order to provide accurate positions and navigation. The following is a summary of the relevant methods:

- Model-based approaches: These approaches are based on the simple prior model. For example, the ground is required to be removed first in [27,28]. Ref. [29] is based on the concept that most moving objects in urban scenes will inevitably come into contact with the ground.
- Voxel map-based approaches: These approaches construct a voxel map and track the emitted ray from LiDAR. When the end point of a LiDAR ray hits a voxel, it is considered to be occupied. Moreover, the LiDAR beam is regarded as traveling across free voxels. The voxel probability in the voxel map can be computed in this way. However, these methods are computationally expensive. Even with engineering acceleration in the latest method [30], processing a large number of 3D points online is still difficult [31]. In addition, these methods need highly accurate localization information, which is a challenge for SLAM. In [32], an offline approach for labeling dynamic points in LiDAR scans based on occupancy maps is introduced, and the labeled results are used as training datasets for deep learning-based LiDAR SLAM methods.
- Visibility-based approaches: In contrast to building a voxel map, the visibility-based approaches just need to compare the visibility difference rather than maintaining a large voxel map [33–35]. Specifically, the observed point should be considered dynamic if the view from the previously observed point blocks out the view from the current point. RF-LIO [36] proposed an adaptive dynamic object rejection method based on removert, which can perform SLAM in real time.
- Learning-based method: The performance of semantic segmentation and detection methods based on deep learning has significantly improved. Ruchti et al. [37] integrated a neural network and an octree map to estimate the occupancy probability. Point clouds in a grid with a low occupancy probability are considered dynamic points. Chen et al. [38] proposed a fast-moving object segmentation network to divide the LiDAR scan into dynamic and still objects. The network is able to operate even faster than the LiDAR frequency. Wang et al. [39] proposed a 3D neural network, SANet, and added it to LOAM for semantic segmentation of dynamic objects. Jeong et al. [40] proposed 2D LiDAR odometry and mapping based on CNN, which used the fault detection of scan matching in dynamic environments.

1.3. New Contribution

This paper mainly aims to improve the accuracy of LIO in dynamic urban environments. The following are the main contributions of this paper:

1. An online and effective dynamic point detection method at the spatial dimension is optimized and integrated. This approach fully utilizes the height information of the ground in the point clouds to detect dynamic points.
2. An indexed point-based dynamic point propagation and removal algorithm is proposed to remove more dynamic points in a local map along the spatial and temporal dimensions and detect dynamic points in historical keyframes.
3. In the LiDAR odometry module, we propose a delayed removal strategy for keyframes. Additionally, a lite slide window method is utilized to optimize the poses from scan-to-map module. We assign dynamic weights to the well-matched LiDAR feature points in the historical keyframes in the sliding window.

2. Materials and Methods

An overview of our ID-LIO is shown in Figure 1. Our system includes three main modules. The first module is IMU pre-integration. This is employed to estimate system motion and obtain an initial pose result for point cloud decomposition and LiDAR odometry. Second, the feature extraction module includes de-skewed point clouds and feature extraction. In the de-skewed point clouds part, point clouds from a raw LiDAR scan are de-skewed using the pose from the IMU integration. In the feature extraction part, the roughness of the points is computed in order to extract the planar and edge features.

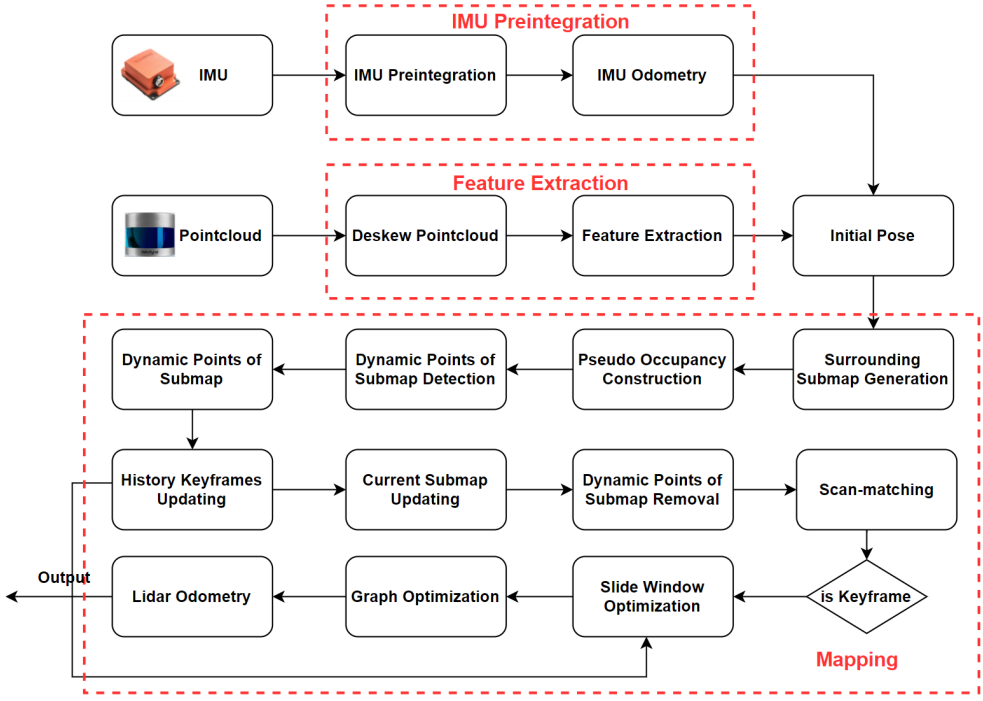


Figure 1. An overall framework of ID-LIO system.

Last, the key component of our LIO system is the mapping module. To achieve dynamic points in local map removal online before the scan-to-map module, several critical steps are applied, as follows: (i) IMU odometry is used to estimate the initial pose. (ii) The raw LiDAR scan and the matching local map are constructed, respectively, as the pseudo occupancies. (iii) The main moving objects of the local map are detected by comparing the height difference in the pseudo occupancy in the scan and the local map. (iv) Through a comparison of the dynamic observation numbers of map points with the threshold, we can finally remove the primary moving points from the local map. After the historical keyframes

are updated, the dynamic points in the previous keyframes can also be labeled. (v) If the current frame is determined to be a keyframe, only the LiDAR scan-to-map residuals with dynamic weight are optimized via the lite slide window optimization method.

2.1. IMU Pre-Integration

The raw IMU measurements include acceleration $\hat{a}_B(t)$ and angular velocity $\hat{\omega}_B(t)$. The measured values are all under the IMU coordinate system **B**, which is the same as the frame of motion body. The IMU measurement model is expressed as:

$$\hat{\omega}_B(t) = \omega_B(t) + b^g(t) + \eta^g(t) \tag{1}$$

$$\hat{a}_B(t) = a_B(t) - R_W^B(t)g^W + b^a(t) + \eta^a(t) \tag{2}$$

where b is the bias and η is white noise. The raw, measured values of IMU are affected by them. The gravity vector is $g^W = [0, 0, g]^T$ in the world frame **W**. It only affects the measurement of the accelerometer.

In our work, we assume that $i+1$ is the current frame and i is the previous frame. When $\hat{\omega}_B^{i+1}$ and $\hat{a}_B^{i+1}$ come, the initial pose state between frame i and frame $i+1$ can be estimated using the IMU pre-integration approach:

$$R_{WB}^{i+1} = R_{WB}^i \Delta R_{i,i+1} Exp(J_{\Delta R}^{\omega} \delta b_{\omega}^i) \tag{3}$$

$$v_{WB}^{i+1} = v_{WB}^i + g^W \Delta t_{i,i+1} + R_{WB}^i (\Delta v_{ij} + J_{\Delta v_{i,i+1}}^{\omega} \delta b_{\omega}^i + J_{\Delta v_{i,i+1}}^{a} \delta b_a^i) \tag{4}$$

$$p_{WB}^{i+1} = p_{WB}^i + v_{WB}^i \Delta t_{i,i+1} + \frac{1}{2} g^W \Delta t_{i,i+1}^2 + R_{WB}^i (\Delta p_{ij} + J_{\Delta p_{i,i+1}}^{\omega} \delta b_{\omega}^i + J_{\Delta p_{i,i+1}}^{a} \delta b_a^i) \tag{5}$$

where R_{WB}^{i+1} is the attitude rotation, v_{WB}^{i+1} is velocity and p_{WB}^{i+1} is position.

2.2. Indexed Point Initialization

In our work, we concentrate on feature point clouds in keyframes. We denote the extracted edge and planar features from a LiDAR scan at time step $t+1$ as F_{t+1}^e and F_{t+1}^p, respectively. All the features extracted at time $t+1$ compose a LiDAR keyframe F_{t+1}, where $F_{t+1} = \left\{F_{t+1}^e, F_{t+1}^p\right\}$. A number of keyframes of edge points close to the current $t+1$ moment together constitute the local edge point cloud map M_t^e corresponding to the current frame, where $M_t^e = \left\{F_{t-i}^e, F_{t-i+1}^e, \cdots, F_{t-1}^e, F_t^e\right\}$. $i+1$ indicates the number of frames that make up the current local corner point map. Similarly, the local plane point cloud map at $t+1$ moment is represented as M_t^p, where $M_t^p = \left\{F_{t-i}^p, F_{t-i+1}^p, \cdots, F_{t-1}^p, F_t^p\right\}$. M_t^e and M_t^p compose a local map M_t corresponding to the current keyframe F_{t+1}, where $M_t = \left\{M_t^e, M_t^p\right\}$.

In order to better find the point and then update the status of the point, a new type of point is defined. This operation of storing extra information in addition to the 3D position of a point is motivated by directed geometry point [41]. In our work, a point **P** is a vector with size 7:

$$\mathbf{P} = [\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z, \mathbf{i}_p, \mathbf{i}_{kf}, \mathbf{c}_p, \mathbf{n}_{do}]^T \tag{6}$$

where the first 3×1 sub-vector $\mathbf{p} = [\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z]^T$ denotes the point 3D position, and the second 2×1 vector $\mathbf{i} = [\mathbf{i}_p, \mathbf{i}_{kf}]$ stores the index information of point $\mathbf{p}$. $\mathbf{i}_p$ records the point index in $\mathbf{i}_{kf}$, and $\mathbf{i}_{kf}$ indicates which keyframe point $\mathbf{i}_p$ belongs to the keyframe set. In addition, $\mathbf{c}_p$ is the category of the point. $\mathbf{n}_{do}$ is dynamic observation number (DON) that records the number of times the point was observed as a dynamic point.

We do not initialize every LiDAR scan. When a current LiDAR scan is selected as a keyframe, we initialize the points of this keyframe. This is because the local map is composed of multiple keyframes. The state of feature points in a keyframe is what we are

concerned with. The difference between a common point and an indexed point is shown in Figure 2. In Figure 2a, the local map consists of multiple keyframes composed of common points. In Figure 2b, because of the indexed point, we can know from which keyframe the points on the local map come. This part is used in Section 2.4.

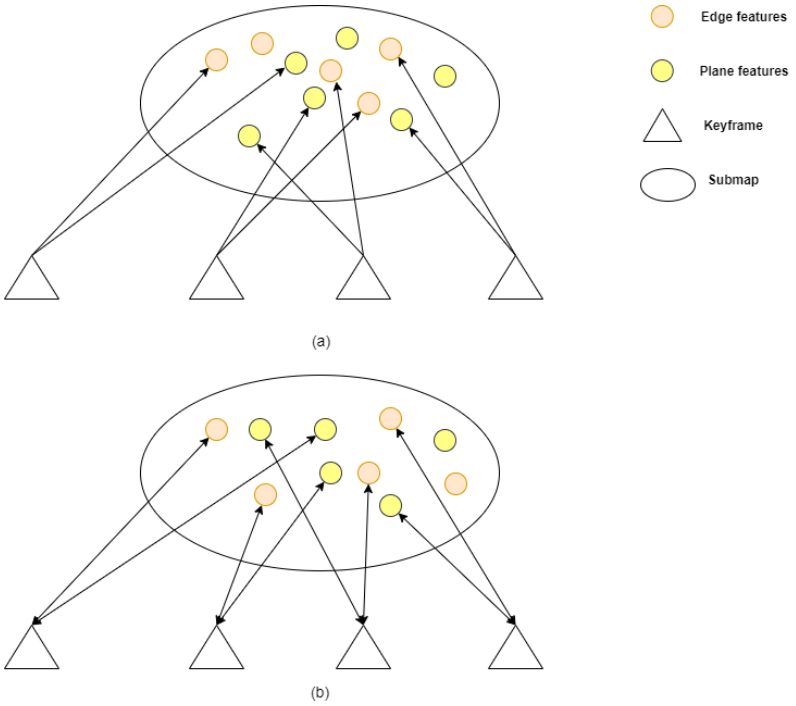


Figure 2. The difference between common points and index points. In (**a**), the local map consists of multiple keyframes composed of common points. In (**b**), the local map consists of multiple keyframes composed of indexed points. In this way, we can know from which keyframe the points on the local map come.

2.3. Dynamic Point Detection

For dynamic point detection in high-dynamic urban environments and in the process of SLAM, the offline post-processed ERASOR [29] method is optimized and integrated into the SLAM online system. As shown in Figure 3, we can see the overview of the online dynamic point detection method.

2.3.1. Problem Definiton

We first denote the current LiDAR frame and corresponding voxel submap at time step $t+1$ as F^B_{t+1} and M^W_t, respectively. The former is a full query LiDAR scan and the latter is a feature map that is composed of some keyframes around F_{t+1}, found by the sliding window method. Note that M^W_t is in the world frame **W**. However, F^B_{t+1} is in frame **B**. In the ERAOSR algorithm, the point clouds on the local map need to be transformed to frame **B**, and the offline local map needs to be constructed in advance. This takes substantial computational time because of the large number of point clouds on the local map. In order to improve the real-time performance of the system, we use the same local map as the scan-to-map module, and transform frame F^B_{t+1} to the world coordinate system **W** before the moving point detection, to obtain F^W_{t+1} and record the current position of the motion robot $\mathbf{p}^W_{t+1}$ in frame **W** first., This can be formulated as follows:

$$F^W_{t+1} = {}^W_B T_{t+1} * F^B_{t+1} \tag{7}$$

$$\mathbf{p}^W_{t+1} = [x^W_{t+1}, y^W_{t+1}, z^W_{t+1}]^T \tag{8}$$

where ${}^{B}_{W}T_{t+1}$ is the relative transformation between F^{B}_{t+1} and M^{W}_{t} from IMU odometry; Equation (7) represents the conversion of all point cloud coordinates in the **B** to the **W**.

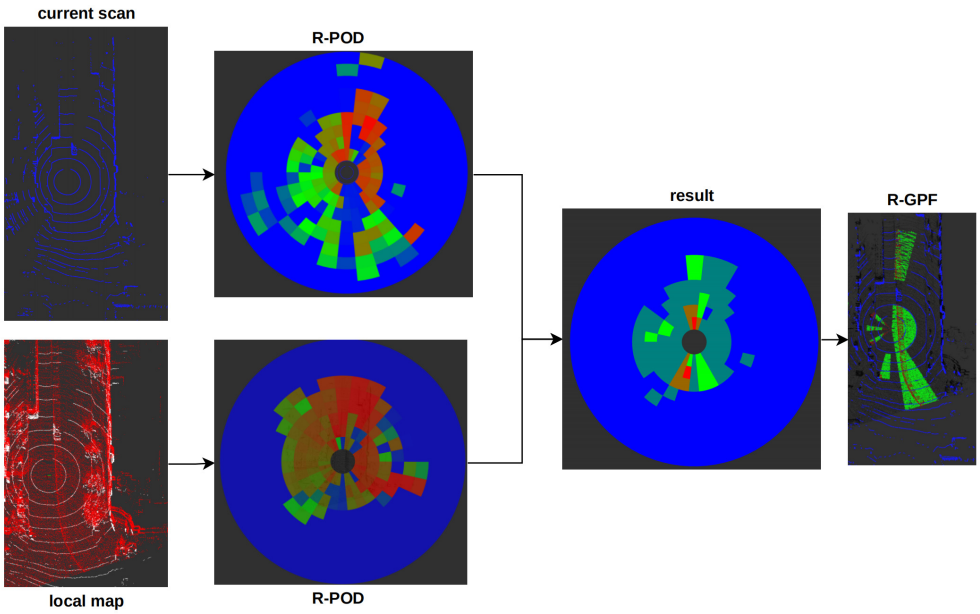


Figure 3. Overview of the online dynamic point detection method. First, we construct R-POD for the current frame and local map, respectively. Then, we compare the height difference in the pseudo occupancy in the scan R-POD and local map R-POD and obtain the dynamic pseudo occupancy of the local map. Last, we use the R-GPF method to obtain the ground plane, and the points above the ground are dynamic points.

2.3.2. Pseudo Occupancy-Based Removal Method

First, we divide the LiDAR frame F^{W}_{t+1} and the submap M^{W}_{t} to region-wise pseudo occupancy descriptors (R-POD) O^{M}_{t} and O^{F}_{t+1}, respectively, in a process similar to scan context:

$$O^{M}_{t} = \bigcup_{i\in[r_N], j\in[\theta_N]} O^{M}_{(i,j)} \qquad O^{F}_{t+1} = \bigcup_{i\in[r_N], j\in[\theta_N]} O^{F}_{(i,j)} \tag{9}$$

where r_N is the ring number and θ_N is the sector number. $O^{F\ or\ M}_{(i,j)}$ is the (i,j)-th bin of R-POD. Let $\theta = arctan2(y - y^{W}_{t+1}, x - x^{W}_{t+1})$. Each $O^{F\ or\ M}_{(i,j)}$ includes the point clouds that meet the following formulas:

$$O^{M}_{(i,j)} = \left\{ \mathbf{p}_m | \mathbf{p}_m \in M^{W}_{t}, \frac{(i-1)\cdot D_{max}}{r_N} \le \rho_m < \frac{i\cdot D_{max}}{r_N}, \frac{(j-1)\cdot 2\pi}{\theta_N} - \pi \le \theta_m < \frac{j\cdot 2\pi}{\theta_N} - \pi \right\} \tag{10}$$

$$O^{F}_{(i,j)} = \left\{ \mathbf{p}_f | \mathbf{p}_f \in F^{W}_{t+1}, \frac{(i-1)\cdot D_{max}}{r_N} \le \rho_f < \frac{i\cdot D_{max}}{r_N}, \frac{(j-1)\cdot 2\pi}{\theta_N} - \pi \le \theta_f < \frac{j\cdot 2\pi}{\theta_N} - \pi \right\} \tag{11}$$

where $\rho_m = \sqrt{(x_m - x^{W}_{t+1})^2 + (y_m - y^{W}_{t+1})^2}$; $\rho_f = \sqrt{(x_f - x^{W}_{t+1})^2 + (y_f - y^{W}_{t+1})^2}$; $D_{max} = \frac{R_l}{2}$ and R_l is the maximum detection distance of LiDAR. Then, we calculate the maximum height difference $\Delta h^{M}_{(i,j)}$ and $\Delta h^{F}_{(i,j)}$ in each bin. Thereafter, we can obtain the potentially dynamic bins in O^{M}_{t} if:

$$\Delta h^{M}_{(i,j)} \quad / \quad \Delta h^{F}_{(i,j)} > 0.2 \tag{12}$$

Finally, we use region-wise ground plane fitting (R-GPF) [29] to obtain ground points in dynamic bins and the ground plane. The dynamic points are points in dynamic bin, expecting ground points, and all dynamic points in dynamic bins consist of the final dynamic points set M_D^m. Therefore, we can obtain the dynamic set M_D, where M_D is

$$M_D = M_D^m \tag{13}$$

2.4. Dynamic Point Propagation and Removal

ERAOSR [29] is an offline method that uses the long sequence local map. However, we need to eliminate dynamic points in the online process of SLAM. We can only obtain the short sequences in the SLAM, which will decrease the performance of dynamic point removal. Therefore, we propose a dynamic point removal method based on spatial and temporal dimensions to improve the removal performance.

In our work, we do not remove dynamic points in a submap when obtaining the dynamic point set M_D after the dynamic point detection module, like RF-LIO. We judge whether the feature point is dynamic not only according to the difference between the current LiDAR scan and submap but also via the history difference. We propagate these points to historical keyframes based on the indexed points in Section 2.2. Then, we perform the dynamic point removal. If a feature point is a dynamic feature point, the frequency with which the feature point is judged to be a dynamic feature point will be high. Thus, the dynamic observation numbers (DON) of a point is used to determine whether the point is dynamic in the temporal dimension. The dynamic point propagation and removal process is summarized in Algorithm 1.

Algorithm 1: Dynamic Point Propagation and Removal Algorithm

Input : Dynamic points set of edge points submap ${}^{D}M_t^e$ and plane submap ${}^{D}M_t^p$; corresponding edge points submap M_t^e and plane points submap M_t^p; nearest edge keyframes buffer array$\{KF_e\}$ and plane keyframes buffer array $\{KF_p\}$

Output: Nearest keyframes buffer array $\{KF_e\}$ and plane keyframes buffer array $\{KF_p\}$;
static edge point map ${}^{S}M_t^e$. and static plane point map ${}^{S}M_t^p$.

for p_i^e *in* ${}^{D}M_t^e$ **do**
  Get the keyframe index i and the edge point index j;
  $KF_e[i][j].don+ = 1$;
  Get the point index k in M_t^e;
  $M_t^e[k].don+ = 1$;
for P_i^p *in* ${}^{D}M_t^p$ **do**
  Get the keyframe index i and the plane point index j;
  $KF_p[i][j].don+ = 1$;
  Get the point index k in M_t^p;
  $M_t^p[k].don+ = 1$;
for *edge map point* mp_i^e *in* M_t^e **do**
  if $mp_i^e.don <= \tau_D$ **then**
    ${}^{S}M_t^e$.pushback(mp_i^e)
for *plane map point* mp_i^p *in* M_t^p **do**
  if $mp_i^p.don <= \tau_D$ **then**
    ${}^{S}M_t^p$.pushback(mp_i^p)

In this algorithm, τ_D is a threshold used to distinguish dynamic feature points and static feature points. If τ_D is small, there will be many static points misclassified as dynamic points. If τ_D is large, we cannot remove dynamic feature points better. In our system , τ_D is set to 3 and 3 is an empirical value; it ensures a balance between the removal of dynamic points and the retention of static points.

2.5. LiDAR Odometry

This module includes scan-to-map matching and front-end optimization. To describe this section clearly, we start with the scan-to-map matching module, which is followed by the front-end optimization module.

2.5.1. Feature-Based Scan-to-Map Matching

The sensor motion between two consecutive LiDAR scans is estimated using the front end of SLAM. There are a variety of scan-matching methods, and the fundamental method of this module is the same as LOAM. We compute the roughness of each point in each LiDAR frame. Then, we extract edge and planar features for this new scan. The formula for calculating roughness is as follows:

$$roughness = \frac{1}{n.||p_i||}|| \sum_{j=1, j\neq i}^{n} (p_j - p_i)|| \tag{14}$$

where p_i represents the target point and p_j is in the same ring as the LiDAR scan. The edge and planar features from the LiDAR frame $t+1$ are denoted as F^e_{t+1} and F^p_{t+1}, respectively. Then, the feature points in F^e_{t+1} and F^p_{t+1} are transformed to W and obtain $\left\{ {}^W\mathbf{F}^e_{t+1}, {}^W\mathbf{F}^p_{t+1} \right\}$. IMU pre-integration provides the initial transformation. By using the nearest-neighbor search method, we are able to locate its correlating feature points in $\left\{ M^e_t, M^p_t \right\}$. Thus, the following formula can be used to calculate the distance between a target point and its related edge:

$$d^e_t = \frac{(p^{ei}_{t+1} - m^{e1}_t) \times (p^{ei}_{t+1} - m^{e2}_t)}{|m^{e1}_t - m^{e2}_t|} \tag{15}$$

where $p^{ei}_{t+1} \in {}^W\mathbf{F}^e_{t+1}$ is an edge feature point and $m^{e1}_t, m^{e2}_t \in M^e_t$ are two different points on the corresponding edge line. Similarly, the distance from a target point to its associated plane can be calculated as follows:

$$d^p_t = \frac{|(p^{pi}_{t+1} - m^{p1}_t)^T((m^{p1}_t - m^{p2}_t) \times (m^{p1}_t - m^{p3}_t))|}{|(m^{p1}_t - m^{p2}_t) \times (m^{p1}_t - m^{p3}_t)|} \tag{16}$$

where $p^{pi}_{t+1} \in {}^W\mathbf{F}^p_{t+1}$ is an planar feature point and $m^{p1}_t, m^{p2}_t, m^{p3}_t \in M^p_t$ are three different points on the corresponding plane.

At last, we can estimate the pose between the current frame and local map by solving the optimization problem:

$$\min_{\Delta T^t_{t+1}} r_L = \min_{\Delta T^t_{t+1}} \left\{ \sum_{p^{ei}_{t+1} \in {}^W\mathbf{F}^e_{t+1}} d^e_t + \sum_{p^{pi}_{t+1} \in {}^W\mathbf{F}^p_{t+1}} d^p_t \right\} \tag{17}$$

2.5.2. Front-End Optimization

Within the scanning-to-mapping part, the local map is used with the dynamic points removed. However, we use a LiDAR scan, in which dynamic points exist. Because, in a single LiDAR frame, there are not enough sparse LiDAR points to use the dynamic point removal method as described above [42], we propose a delayed removal strategy for the keyframe and build a cost function including only LiDAR measurements with a dynamic observation-related weight jointly motivated by VINS-Mono. To obtain more accurate odometry for each keyframe of LiDAR, we optimize the pose states in the lightweight sliding window iteratively.

In our LIO system, the states in the sliding window that need to be optimized are defined as $\mathbf{X}^w = [\mathbf{x}^w_{t-n}, \mathbf{x}^w_{t-n+1}, \mathbf{x}^w_{t-1}, ..., \mathbf{x}^w_t]$, where we optimize the poses of the n keyframes before the current moment **t**, and $x^w_{t-i} = [p^W_{b_{t-i}}, q^W_{b_{t-i}}]$. For a sliding window of size n, these states are obtained by minimizing the final cost function, which is described as:

$$\min_{\mathbf{X}^w} \left\{ \sum_{k=0}^{n} r^e_L(\mathbf{x}^w_k) + \sum_{k=0}^{n} r^p_L(\mathbf{x}^w_k) \right\} \tag{18}$$

In this cost function, $r^e_L(\mathbf{x}^w_k)$ denotes edge-line geometric terms of LiDAR and $r^p_L(\mathbf{x}^w_k)$ means the planar LiDAR error terms. Based on the previous LiDAR term calculation of each scan, the LiDAR term incorporates geometric constraints with dynamic weight from the dynamic point propagation and removal module (see Section 2.4) into the cost function scheme. The term is denoted as:

$$r_L(\mathbf{x}^w_k) = \sum_{i}^{m} w_i d^e_i + \sum_{j}^{n} w_j d^p_j \tag{19}$$

and the dynamic observations-related weight is defined as:

$$w = \begin{cases} 1 & n_{do} <= 1 \\ \frac{1}{n_{do}} & n_{do} >= 2 \end{cases} \tag{20}$$

where d^e_i is the distance from the i-th target edge point to its corresponding edge and d^p_j is the distance from target planar point to its corresponding plane in Section 2.5.1; w_i is a dynamic observations-related weight; and n^i_{do} is the dynamic observations of the i-th feature point. In this paper, we set the value of dynamic weight according to the DON. This indicates that the higher the DON, the less important the corresponding residuals d are in the cost function r.

3. Results

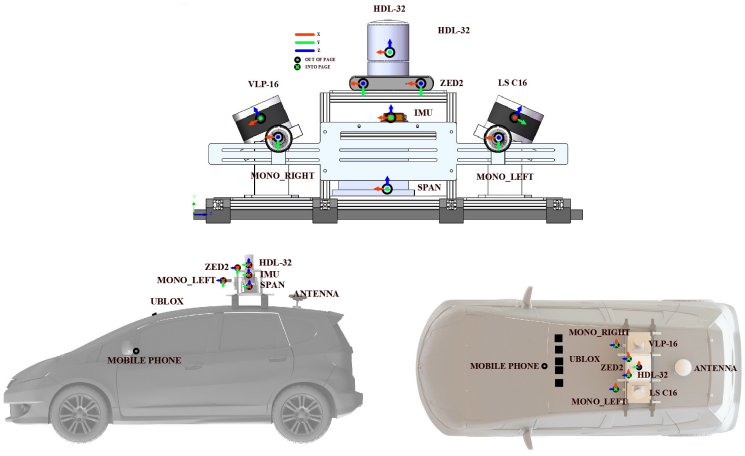
3.1. Experimental Setup

We validate our system ID-LIO by conducting a set of experiments on public datasets, and ID-LIO is contrasted with FAST-LIO2, Faster-LIO and LIO-SAM. Compared with the LIO-SAM, the ID-LIO employs the same feature extraction module, loopback detection strategy and optimization method for the back-end using GTSAM. We use evo [43] to evaluate the accuracy. The ablation experiment demonstrates the effect of our proposed methods. The proposed methods are executed on a computer with an Intel i5 CPU and 32G RAM. Ubuntu 18.04 with the robot operating system (ROS) melodic [44] is our computer system. The datasets used for validation include UrbanNav datasets in Hong Kong (UNHK) [45] and UrbanLoco datasets in California (ULCA) [46]. The UrbanNav-HK-Data20190428 is a dataset containing some movable objects that is defined as a low-dynamic dataset. We define the others in UNHK and ULCA as high-dynamic datasets because they contain masses of movable objects. As shown in Table 1, the data information is listed in detail. The sensor suite of the UBHK dataset is shown in Figure 4.

Table 1. UrbanNav and UrbanLoco datasets' details.

Dataset	Trajectory Length (m)	Dynamic Level	Scale Level
UNHK-Data20190428 [45]	1800	Low	Medium
UNHK-TST [45]	3640	High	Medium
UNHK-Mongkok [45]	4860	High	Medium
UNHK-Whampoa [45]	4510	High	Medium
ULCA-MarketStreet [46]	5690	High	Large
ULCA-RussianHill [46]	3570	High	Medium

(**a**)



(**b**)

Figure 4. The sensor suite of the UNHK [45] dataset. (**a**) is a view of the sensor kit in the car. (**b**) shows the distributions location of each sensors.

3.2. Dynamic Observations Number Analysis

The impact of the appropriate DON value on the removal results of dynamic objects is very critical. Consequently, the purpose of this experiment is to demonstrate that the approach mentioned in Section 2.4 is accurate. Setting the correct DON threshold is important. If the DON value is too low, the static points may be incorrectly regarded as dynamic points. If it is too large, many moving points may be wrongly identified as static points. In the implementation of our algorithm, we set the DON value to 3. This can ensure that static points and dynamic points are correctly distinguished. To further prove the

rationality of this threshold value, we chose three dynamic urban datasets. We tested the LIO-SAM using the different thresholds between 1 and 6. DON1 to 6 means the value of DON is from 1 to 6. The root mean square error (RMSE) of the absolute trajectory error (ATE) of them is shown in Table 2, which demonstrates that the fixed threshold 3 can provide more reliable and accurate pose estimation.

Table 2. RMSE of ATE of different DON thresholds in three datasets.

Sequence/ATE	DON1	DON2	DON3	DON4	DON5	DON6
UNHK-Data20190428 [45]	6.55	6.36	5.96	6.55	6.34	6.43
UNHK-TST [45]	7.51	4.86	4.27	4.70	5.20	7.39
ULCA-MarktStreet [46]	86.69	87.500	49.26	49.98	49.75	86.05

ULCA-MarktStreet was chosen for our analysis. Figure 5 shows the trajectories of different DONs with our system in ULCA-MarktStreet in an X–Y plot when the thresholds are 1, 3 and 6. This figure shows the total error for the three distinct thresholds. It can be seen that the trajectory corresponding to DON3 is closer to the ground truth because the DON3 system achieves a balance between dynamic point removal and static point retention.

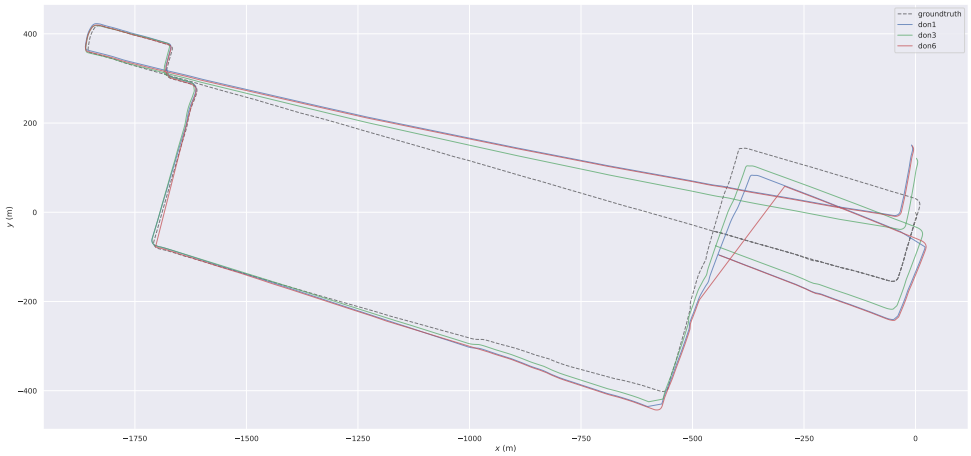


Figure 5. The trajectories of the three DON thresholds on ULCA-MarktStreet. The trajectory of DON3 is the most accurate; it is closer to the ground truth.

The box diagram of the deviation is shown in Figure 6, which allows us to have a better view of the magnitude of the deviation. The DON3 system has the smallest error.

As shown in Figure 7, there are two kinds of red point sets in the figures. Red cloud points on the street indicate dynamic objects by moving cars, and the more red points, the better. However, red cloud points at other places indicate static objects that have been incorrectly removed, and the fewer red points, the better. If the threshold is 1, more dynamic points are removed. At the same time, many static points are incorrectly removed. If the threshold is 6, more static points are preserved. However, many dynamic points are also preserved in comparison with DON1. Neither of these two extreme cases applies to dynamic point removal. However, DON3 can achieve better rejection results. It retains most of the static points of the local map while removing more dynamic points.

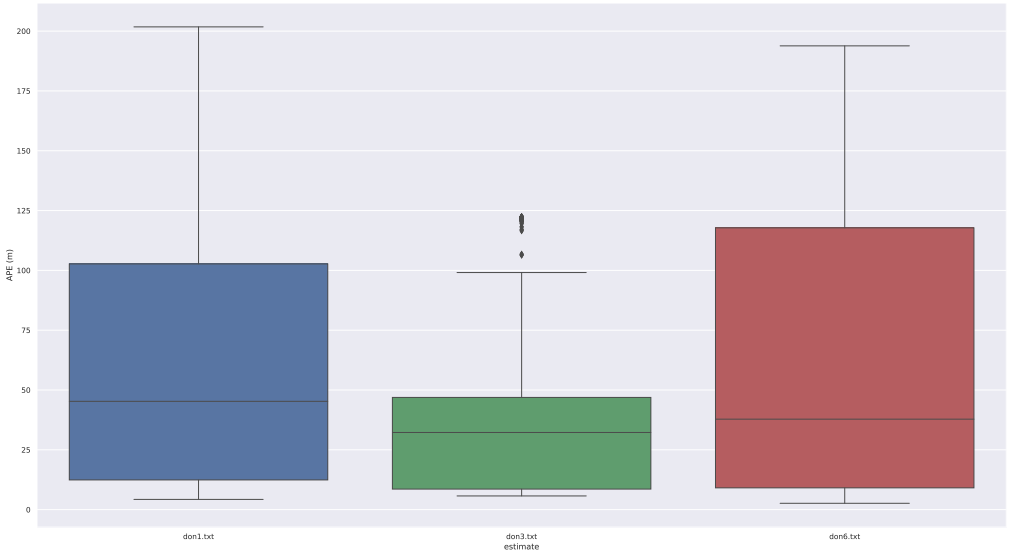


Figure 6. Box diagram of the deviation.

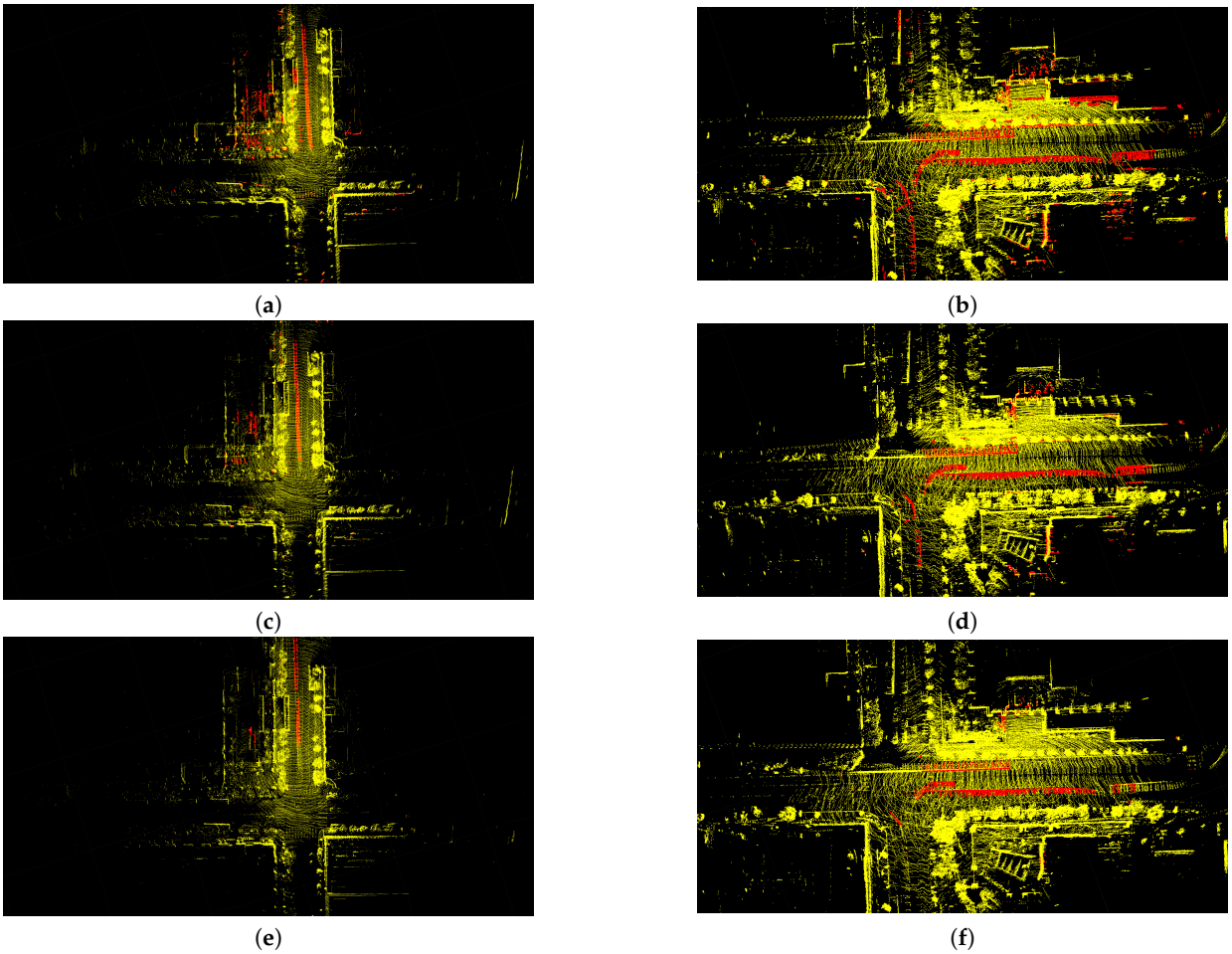


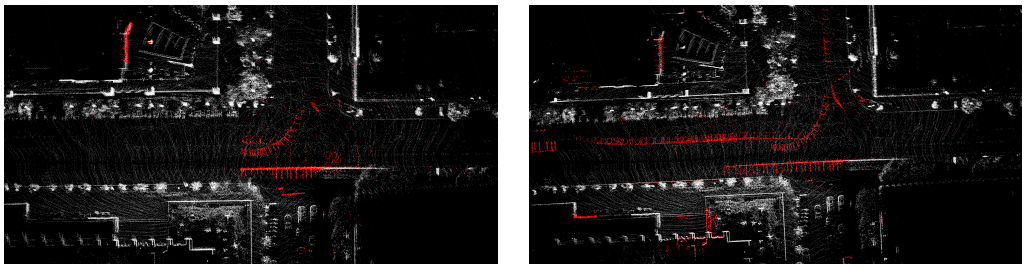
Figure 7. The removal results of local maps corresponding to different DON thresholds. (**a**,**c**,**e**) are the local maps corresponding to different DON thresholds in keyframe 110. (**b**,**d**,**f**) are the local maps corresponding to different DON thresholds in keyframe 310.

3.3. Comparison of Dynamic Removal Strategies for Local Map

To further demonstrate the efficacy of the proposed removal strategy based on the pseudo occupancy and indexed point in the spatial and temporal dimensions (denoted as LIO-SAM + Spatial + Temporal), we make comparisons with the removal strategy based on the pseudo occupancy in the spatial dimension only (denoted as LIO-SAM + Spatial). The ATEs of the mapping thread are shown in Table 3. Figure 8a shows the dynamic point removal results obtained by these two methods. The red points on the street are dynamic traces by moving cars. Figure 8b identifies part of the dynamic feature points. In contrast, our method based on spatial and temporal dimensions successfully removes most of the dynamic traces. After the trajectories of both systems are aligned with the ground truth, we acquire the error maps for these two approaches, which are illustrated in Figure 9.

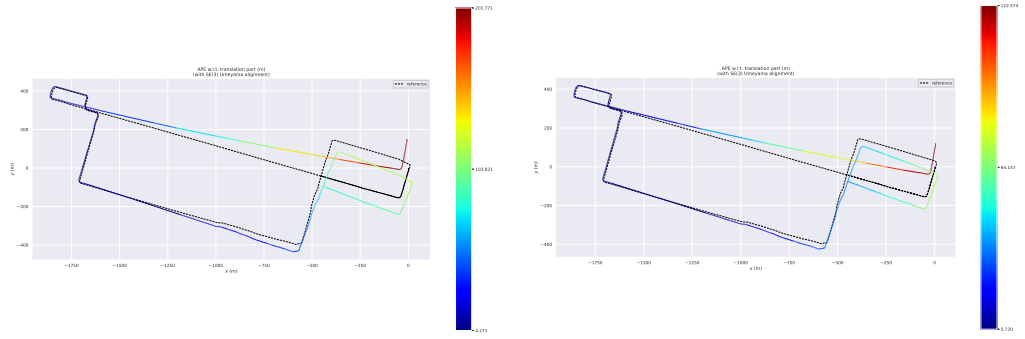
Table 3. RMSE of ATE for LIO-SAM with dynamic removal method based on spatial dimension (denoted as LIO-SAM + Spatial) and LIO-SAM with dynamic removal method based on spatial and temporal dimensions (denoted as LIO-SAM + Spatial + Temporal).

Sequence/ATE	LIO-SAM + Spatial	LIO-SAM + Spatial + Temporal
UNHK-Data20190428 [45]	6.55	5.96
UNHK-TST [45]	7.51	4.27
ULCA-MarktStreet [46]	86.69	49.26



(**a**) (**b**)

Figure 8. Comparison of dynamic point removal results produced by different removal methods. Dynamic objects are represented by red dots, and it is better if there are more red points. (**a**) Dynamic point removal results by removal method based on spatial dimension. (**b**) Dynamic point removal results by removal method based on spatial and temporal dimension.



(**a**) (**b**)

Figure 9. Error map of two dynamic removal methods. (**a**) Error map of removal method based on spatial dimension. (**b**) Error map of removal method based on spatial and temporal dimension.

3.4. Comparison of Delayed Removal Strategy for Keyframe

During the scan-to-map matching, both the dynamic points in the map and the LiDAR scan can result in significant deviation. According to the moving points on the local map, we can obtain the dynamic points in keyframes. When we perform the slide window optimization, the dynamic LiDAR terms will obtain the dynamic weights. Although we cannot remove dynamic points in the current keyframe, they will be removed in the next slide window optimization. Therefore, we conducted this experiment to prove the efficiency of the delayed removal strategy. The ATEs of the mapping thread are shown in Table 4. Figure 10 shows the trajectories of our system with delayed removal strategy and without the strategy. Figure 11 shows the error maps of our system without different removal strategy. Figure 12 shows the removal result. Figure 12a,b shows the real environment with moving cars and people corresponding to the current frame. Figure 12d shows a keyframe with red dynamic points after the delayed removal strategy. The dynamic cars and people are marked with red points. Figure 12c shows a static keyframe after removing the dynamic points of this keyframe.

The goal of this experiment is to demonstrate the effectiveness of the delayed removal method for dynamic points in historical keyframes. We use a dynamic point propagation method to obtain the dynamic points in historical keyframes based on the dynamic local map feature points. Figure 12f shows the dynamic traces on the local map. Figure 12e shows the dynamic points in this keyframe obtained by the dynamic point propagation method; for example, red points marked by yellow rectangles in Figure 12. There are five people walking on the road. These red points are part of the red points in the yellow rectangle in Figure 12f. Similarly, the car in the yellow oval in Figure 12e is part of the red points in the yellow oval in Figure 12f. The error maps of the ID-LIO with delayed removal strategy and without this strategy are shown in Figure 11. We can see that the trajectory in our system with this method is closer to the ground truth.

Table 4. RMSE of ATE for our system without delayed removal strategy (denoted as Ours + No Delayed Removal) compared to ours with the strategy (denoted as Ours + Delayed Removal).

Sequence/ATE	Ours + No Delayed Removal	Ours + Delayed Removal
UNHK-Data20190428 [45]	5.96	5.99
UNHK-TST [45]	4.86	1.06
ULCA-MarktStreet [46]	49.26	28.02

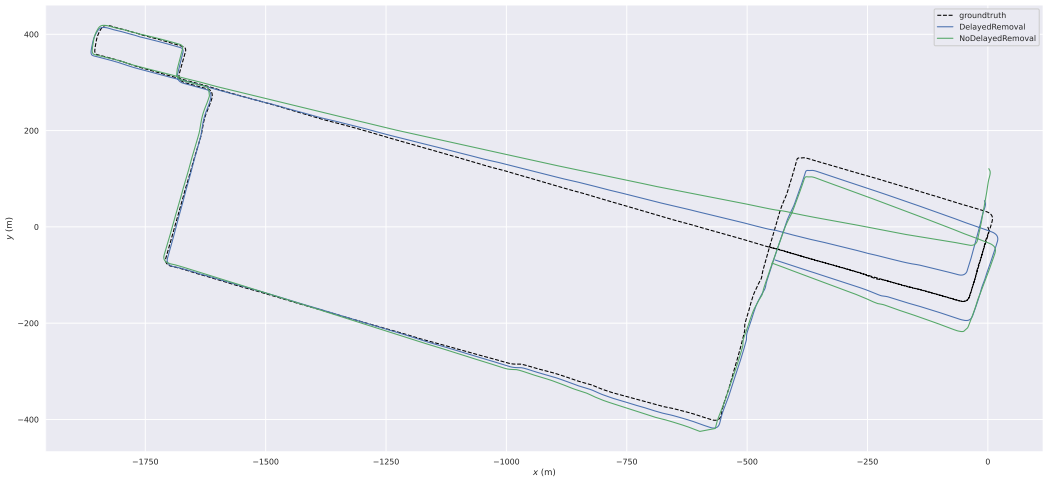
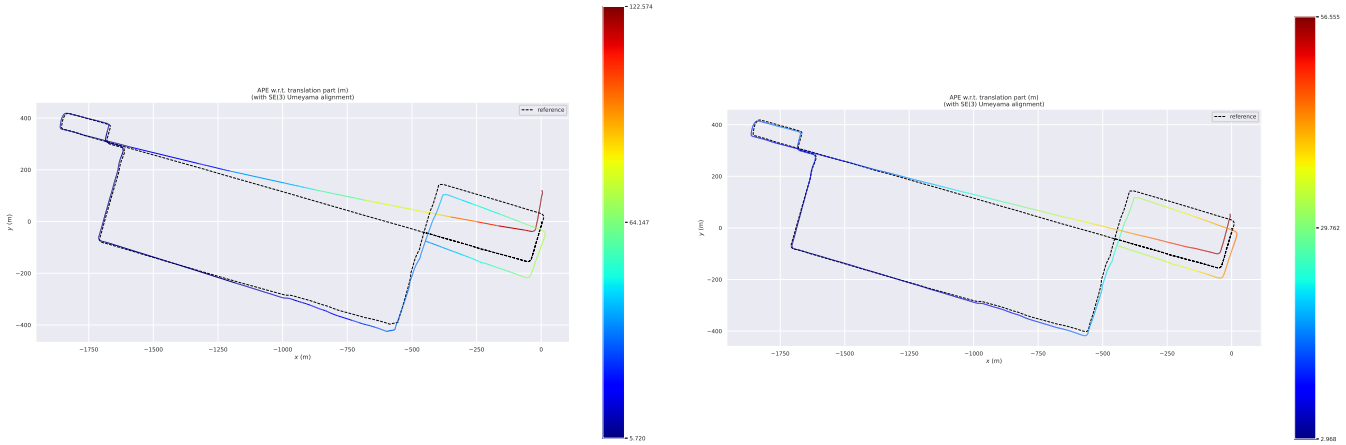
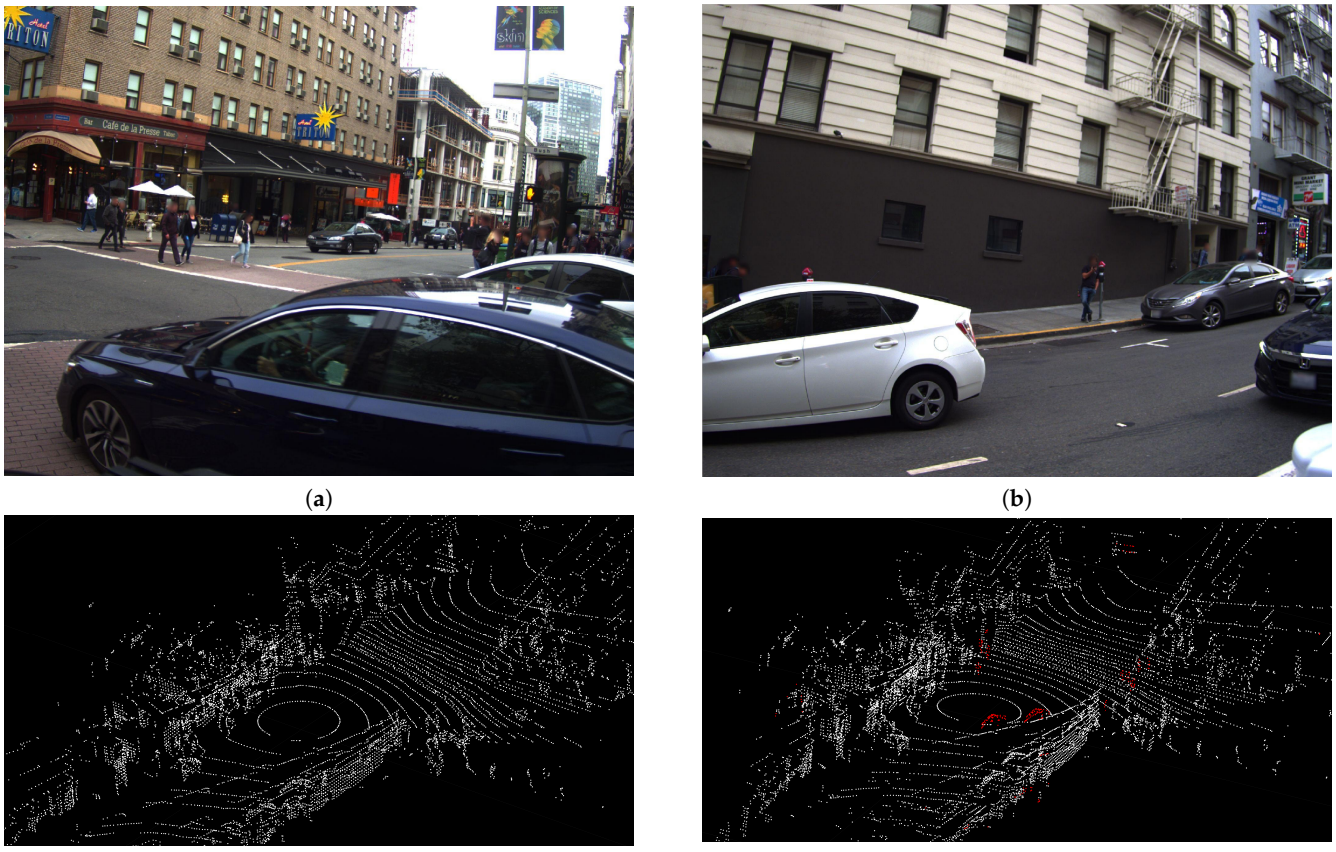


Figure 10. The trajectories of our system with delayed removal strategy and without the strategy.



(**a**) (**b**)

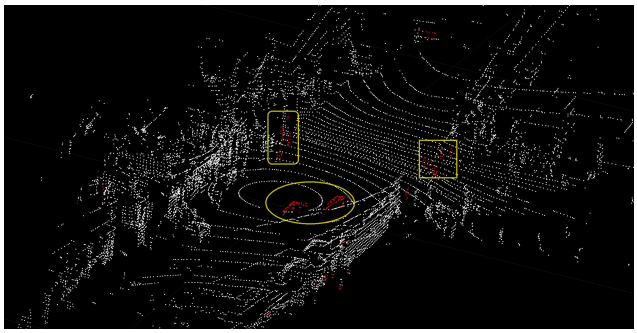
Figure 11. Error map of our system without different removal strategy. (**a**) Error map of our system with delayed removal strategy. (**b**) Error map of our system with the strategy.



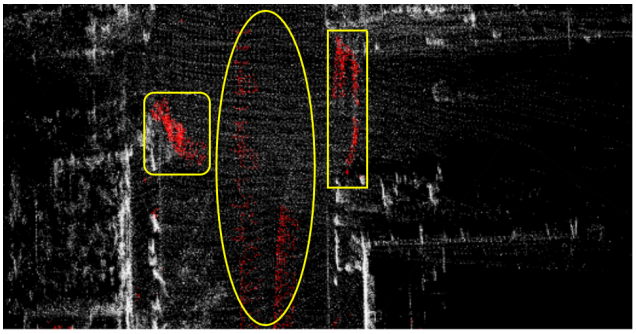
(**a**) (**b**)

(**c**) (**d**)

Figure 12. *Cont*.



(**e**)



(**f**)

Figure 12. Keyframe with red dynamic points after the delayed removal strategy. (**a**,**b**) show the real environment with moving cars and people corresponding to the current frame. (**c**) shows a static keyframe after removing the dynamic points of this keyframe. (**d**) shows a keyframe with red dynamic points after the delayed removal strategy. (**e**) shows the dynamic points in this keyframe obtained by the dynamic point propagation. (**f**) shows the dynamic traces on the local map.

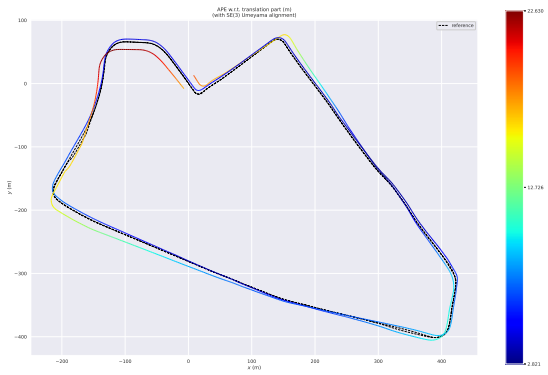
3.5. Results on Low- and High-Dynamic Datasets

We evaluate our ID-LIO system on six datasets with the ground truths from the UrbanNav and UrbanLoco datasets, and we compare the results with LIO-SAM, Fast-LIO, Faster-LIO and the ground truths. For quantitative evaluation, we computed the ATE for these three methods on different datasets using the evaluation tool EVO, and the results are shown in Table 5.

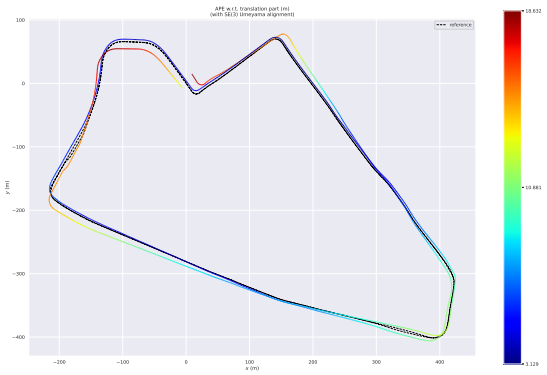
Table 5. RMSE of ATE on UNHK and ULCA datasets (m).

Sequence/ATE	Faster-LIO	FAST-LIO	LIO-SAM	Ours
UNHK-Data20190428 [45]	7.53	7.46	6.55	5.96
UNHK-TST [45]	9.81	9.34	7.51	1.06
UNHK-Mongkok [45]	10.45	10.65	8.89	3.45
UNHK-Whampoa [45]	5.13	5.38	3.32	0.85
ULCA-MarktStreet [46]	-	-	86.69	28.02
ULCA-RussianHill [46]	100.56	110.37	60.35	15.34

After the trajectories of these four systems are aligned with the ground truth, we acquire the error maps for these approaches. These are illustrated in Figure 13, from which we can see that the error of each position calculated by our system is less than 1.10 m, while the error of Faster-LIO, FAST-LIO and LIO-SAM can be up to 9.81 m, 9.34 m and 7.51 m, respectively.

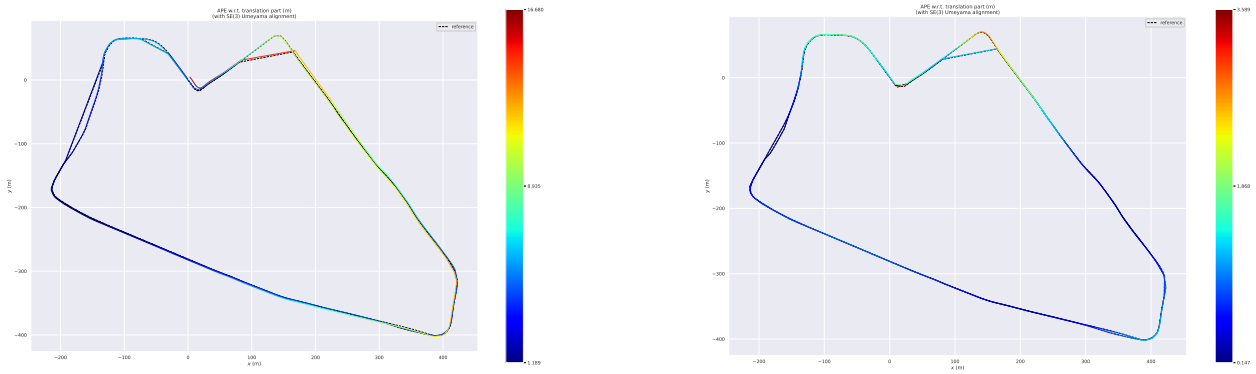


(**a**)



(**b**)

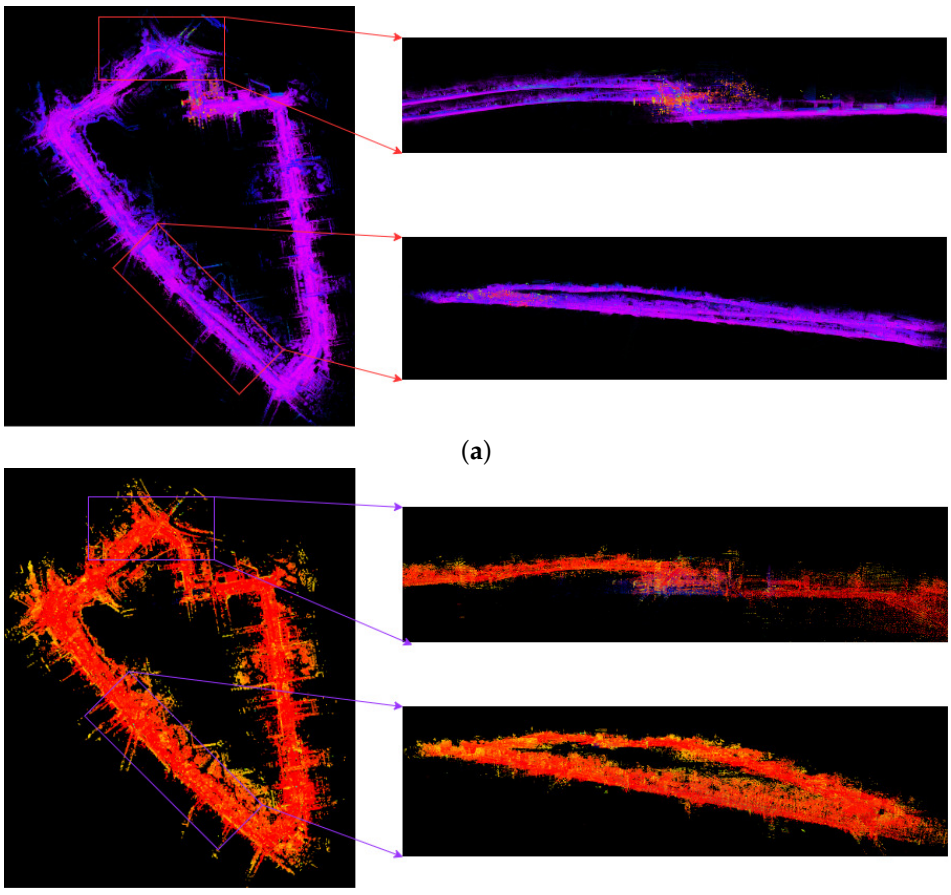
Figure 13. *Cont.*



(**c**) (**d**)

Figure 13. Error maps of four systems on UNHK-TST dataset. (**a**) Error map of Faster-LIO. (**b**) Error map of Fast-LIO. (**c**) Error map of LIO-SAM. (**d**) Error map of ours.

Finally, we compare the differences between the point cloud maps of FAST-LIO and ID-LIO in Figure 14. We can see that Fast-LIO is the least effective because it does not have back-end loopback detection, and LIO-SAM is unable to detect the closed-loop accurately because there are many dynamic objects in the urban environment, resulting in inaccurate poses obtained by scan2map. Our ID-LIO adopts a dynamic point detection and rejection algorithm and achieves better loopback results compared to LIO-SAM.



(**a**)

(**b**)

Figure 14. FAST-LIO and ID-LIO mapping results on the UNHK-TST. (**a**) Mapping result of FAST-LIO. (**b**) Mapping result of ID-LIO.

3.6. Runtime Performance Analysis

Real-time performance is also a critical criterion to consider when evaluating SLAM systems in practice. We record the time consumption of ERASOR, LIO-SAM and ID-LIO on these six datasets. We can see that the runtime of ERASOR is more than 100 ms because of the amount of time spent on map construction and point cloud coordinate conversion. LIO-SAM is a real-time SLAM system that processes each frame in less than 100 ms. ID-LIO is a system with a dynamic point removal module that performs well in real time in five datasets. ULCA-MarktStreet is a highly dynamic and large scale dataset. Therefore, the runtime of ID-LIO is more than 100 ms. The runtime results are shown in Table 6.

Table 6. Average runtime of three systems for processing a scan (ms).

Sequence	ERAOSR	LIO-SAM	ID-LIO
UNHK-Data20190428 [45]	130.5	50.4	89.5
UNHK-TST [45]	135.8	54.8	95.5
UNHK-Mongkok [45]	138.4	68.6	98.9
UNHK-Whampoa [45]	135.1	55.3	97.8
ULCA-MarktStreet [46]	140.6	80.5	120.6
ULCA-RussianHill [46]	134.5	65.3	99.5

4. Discussion

In this paper, we proposed an improved and robust LIO system in dynamic environments. A dynamic point removal method based on pseudo occupancy and indexed points is used for a local map before scan-to-map registration. It reduces the influence of dynamic points on feature matching and improves pose accuracy. In addition, we proposed a delayed removal strategy for keyframes, and the optimization based on sliding window incorporates the LiDAR measurement with dynamic weights to reduce error from dynamic points in keyframes, which further improves the pose accuracy. ID-LIO enables the real-time, accuracy and robustness requirements of SLAM. Compared with the offline ERASOR algorithm, our algorithm is 40% faster. In addition, the results of our dynamic object removal algorithm are better than ERASOR. We can remove more dynamic points and preserve more static points, as shown in Figure 9. The ATE average RMSE of our ID-LIO can be improved by 67% and 85% in the UrbanLoco-CAMarketStreet dataset and UrbanNav-HK-Medium-Urban-1 dataset, respectively, when compared with LIO-SAM. The dynamic point removal module enhances the robustness and accuracy of ID-LIO in highly dynamic environments. Compared with FAST-LIO and Faster-LIO, the ATE average RMSE of our ID-LIO can be improved by 88% and 89%, respectively, in the UNHK-TST dataset.

For future work, we note that it is important to improve the real-time performance of the LIO system in highly dynamic and large-scale environments. We can see that in the ULCA-MarktStreet dataset, the runtime of our system is more than 120 ms. This is because we need to construct local maps for scan-to-map registration and pseudo occupancy maps for dynamic point removal repetitively in our system, which is time-consuming. In the future, we will focus on spatial data structure for local map addition and deletion to improve the system's efficiency. It is also important to note that the LIO system is susceptible to Z-direction drift in vast scenarios. In the future, the ground constraint will be added to reduce drift on the Z-axis.

Author Contributions: Funding acquisition, W.W. (Wanliang Wang); investigation, W.W. (Weizhuang Wu); methodology, W.W. (Weizhuang Wu); supervision, W.W. (Wanliang Wang); writing—original draft, W.W. (Weizhuang Wu); writing—review editing, W.W. (Weizhuang Wu). All authors have read and agreed to the published version of the manuscript.

Funding: Not applicable.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. He, G.; Yuan, X.; Zhuang, Y.; Hu, H. An integrated GNSS/LiDAR-SLAM pose estimation framework for large-scale map building in partially GNSS-denied environments. *IEEE Trans. Instrum. Meas.* **2020**, *70*, 1–9. [CrossRef]
2. Ban, X.; Wang, H.; Chen, T.; Wang, Y.; Xiao, Y. Monocular visual odometry based on depth and optical flow using deep learning. *IEEE Trans. Instrum. Meas.* **2020**, *70*, 1–19. [CrossRef]
3. Lu, Q.; Pan, Y.; Hu, L.; He, J. A Method for Reconstructing Background from RGB-D SLAM in Indoor Dynamic Environments. *Sensors* **2023**, *23*, 3529. [CrossRef] [PubMed]
4. Park, J.; Cho, Y.; Shin, Y.S. Nonparametric Background Model-Based LiDAR SLAM in Highly Dynamic Urban Environments. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 24190–24205. [CrossRef]
5. Besl, P.J.; McKay, N.D. Method for registration of 3-D shapes. In Proceedings of the Sensor Fusion IV: Control Paradigms and Data Dtructures, Boston, MA, USA, 12–15 November 1992; SPIE: Bellingham, WA, USA, 1992; Volume 1611, pp. 586–606.
6. Pomerleau, F.; Colas, F.; Siegwart, R.; Magnenat, S. Comparing ICP variants on real-world data sets: Open-source library and experimental protocol. *Auton. Robots* **2013**, *34*, 133–148. [CrossRef]
7. Serafin, J.; Grisetti, G. NICP: Dense normal based point cloud registration. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September– 2 October 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 742–749.
8. Ren, Z.; Wang, L.; Bi, L. Robust GICP-based 3D LiDAR SLAM for underground mining environment. *Sensors* **2019**, *19*, 2915. [CrossRef]
9. Behley, J.; Stachniss, C. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In Proceedings of the Robotics: Science and Systems, Pittsburgh, PA, USA, 26–30 June 2018; Volume 2018, p. 59.
10. Li, Q.; Chen, S.; Wang, C.; Li, X.; Wen, C.; Cheng, M.; Li, J. Lo-net: Deep real-time lidar odometry. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8473–8482.
11. Cho, Y.; Kim, G.; Kim, A. Unsupervised geometry-aware deep lidar odometry. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 2145–2152.
12. Zhang, J.; Singh, S. LOAM: Lidar odometry and mapping in real-time. In Proceedings of the Robotics: Science and Systems, Berkeley, CA, USA, 12–16 July 2014; Volume 2, pp. 1–9.
13. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 3354–3361.
14. Wang, H.; Wang, C.; Chen, C.L.; Xie, L. F-loam: Fast lidar odometry and mapping. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 4390–4396.
15. Koide, K.; Miura, J.; Menegatti, E. A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 1729881419841532. [CrossRef]
16. Shan, T.; Englot, B. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 4758–4765.
17. Liu, T.; Wang, Y.; Niu, X.; Chang, L.; Zhang, T.; Liu, J. LiDAR Odometry by Deep Learning-Based Feature Points with Two-Step Pose Estimation. *Remote Sens.* **2022**, *14*, 2764. [CrossRef]
18. Ye, H.; Chen, Y.; Liu, M. Tightly coupled 3d lidar inertial odometry and mapping. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 3144–3150.
19. Li, K.; Li, M.; Hanebeck, U.D. Towards high-performance solid-state-lidar-inertial odometry and mapping. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5167–5174. [CrossRef]
20. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 5135–5142.
21. Kaess, M.; Ranganathan, A.; Dellaert, F. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robot.* **2008**, *24*, 1365–1378. [CrossRef]
22. Zhang, J.; Wen, W.; Huang, F.; Chen, X.; Hsu, L.T. Coarse-to-Fine Loosely-Coupled LiDAR-Inertial Odometry for Urban Positioning and Mapping. *Remote Sens.* **2021**, *13*, 2371. [CrossRef]

23. Qin, C.; Ye, H.; Pranata, C.E.; Han, J.; Zhang, S.; Liu, M. Lins: A lidar-inertial state estimator for robust and efficient navigation. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 8899–8906.
24. Xu, W.; Zhang, F. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3317–3324. [CrossRef]
25. Xu, W.; Cai, Y.; He, D.; Lin, J.; Zhang, F. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Trans. Robot.* **2022**, *38*, 2053–2073. [CrossRef]
26. Bai, C.; Xiao, T.; Chen, Y.; Wang, H.; Zhang, F.; Gao, X. Faster-LIO: Lightweight tightly coupled LiDAR-inertial odometry using parallel sparse incremental voxels. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4861–4868. [CrossRef]
27. Dewan, A.; Caselitz, T.; Tipaldi, G.D.; Burgard, W. Motion-based detection and tracking in 3d lidar scans. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 4508–4513.
28. Dewan, A.; Caselitz, T.; Tipaldi, G.D.; Burgard, W. Rigid scene flow for 3d lidar scans. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1765–1770.
29. Lim, H.; Hwang, S.; Myung, H. ERASOR: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3D point cloud map building. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2272–2279. [CrossRef]
30. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robots* **2013**, *34*, 189–206. [CrossRef]
31. Schauer, J.; Nüchter, A. The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1679–1686. [CrossRef]
32. Pfreundschuh, P.; Hendrikx, H.F.; Reijgwart, V.; Dubé, R.; Siegwart, R.; Cramariuc, A. Dynamic object aware lidar slam based on automatic generation of training data. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 11641–11647.
33. Pomerleau, F.; Krüsi, P.; Colas, F.; Furgale, P.; Siegwart, R. Long-term 3D map maintenance in dynamic environments. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 3712–3719.
34. Yoon, D.; Tang, T.; Barfoot, T. Mapless online detection of dynamic objects in 3d lidar. In Proceedings of the 2019 16th Conference on Computer and Robot Vision (CRV), Kingston, QC, Canada, 29–31 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 113–120.
35. Kim, G.; Kim, A. Remove, then revert: Static point cloud map construction using multiresolution range images. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 10758–10765.
36. Qian, C.; Xiang, Z.; Wu, Z.; Sun, H. RF-LIO: Removal-First Tightly-coupled Lidar Inertial Odometry in High Dynamic Environments. *arXiv* **2022**, arXiv:2206.09463.
37. Bescos, B.; Fácil, J.M.; Civera, J.; Neira, J. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4076–4083. [CrossRef]
38. Yu, C.; Liu, Z.; Liu, X.J.; Xie, F.; Yang, Y.; Wei, Q.; Fei, Q. DS-SLAM: A semantic visual SLAM towards dynamic environments. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1168–1174.
39. Wang, W.; You, X.; Zhang, X.; Chen, L.; Zhang, L.; Liu, X. LiDAR-Based SLAM under Semantic Constraints in Dynamic Environments. *Remote Sens.* **2021**, *13*, 3651. [CrossRef]
40. Jeong, H.; Lee, H. CNN-Based Fault Detection of Scan Matching for Accurate SLAM in Dynamic Environments. *Sensors* **2023**, *23*, 2940. [CrossRef] [PubMed]
41. Liang, S.; Cao, Z.; Wang, C.; Yu, J. A novel 3D LIDAR SLAM based on directed geometry point and sparse frame. *IEEE Robot. Autom. Lett.* **2020**, *6*, 374–381. [CrossRef]
42. Fan, T.; Shen, B.; Chen, H.; Zhang, W.; Pan, J. DynamicFilter: An Online Dynamic Objects Removal Framework for Highly Dynamic Environments. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 7988–7994.
43. Sanfourche, M.; Vittori, V.; Le Besnerais, G. eVO: A realtime embedded stereo odometry for MAV applications. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 2107–2114.
44. Będkowski, J.; Pełka, M.; Majek, K.; Fitri, T.; Naruniec, J. Open source robotic 3D mapping framework with ROS—robot operating system, PCL—point cloud library and cloud compare. In Proceedings of the 2015 International Conference on Electrical Engineering and Informatics (ICEEI), Denpasar, Indonesia, 10–11 August 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 644–649.
45. Hsu, L.T.; Kubo, N.; Wen, W.; Chen, W.; Liu, Z.; Suzuki, T.; Meguro, J. UrbanNav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas. In Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), St. Louis, MO, USA, 20–24 September 2021; pp. 226–256.

46. Wen, W.; Zhou, Y.; Zhang, G.; Fahandezh-Saadi, S.; Bai, X.; Zhan, W.; Tomizuka, M.; Hsu, L.T. UrbanLoco: A full sensor suite dataset for mapping and localization in urban scenes. In Proceedings of the 2020 IEEE international conference on robotics and automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 2310–2316.