GLO: General LiDAR-Only Odometry With High Efficiency and Low Drift

Yun Su, Shiliang Shao, *Member, IEEE*, Ziyong Zhang, Pengfei Xu, Yong Cao, and Hui Cheng, *Member, IEEE*

***Abstract*—This study proposes GLO, a general LiDAR-only odometry method with high efficiency and low drift. First, we propose a map data structure using multilevel voxels to improve map update efficiency. Each voxel node actively maintains plane features, minimizing redundant fitting and enhancing matching efficiency. By calculating the occupancy probability of each voxel node, dynamic and unstable points in the map can be efficiently removed. Second, we introduce a weighted elastic matching algorithm that adjusts the weights of each matching constraint across multiple dimensions, such as measurement depth, fitting error, occupancy probability, and matching error, making the matching constraints elastic and enhancing accuracy. This letter also proposes a progressive optimization framework combining prediction, coarse matching, and fine matching. Coarse matching ensures matching convergence and corrects point-cloud motion distortion, while fine matching further refines accuracy. Extensive experiments on public datasets and real LiDAR data demonstrate the efficiency and accuracy of the proposed GLO method compared to state-of-the-art methods.**

***Index Terms*—Localization, mapping, SLAM.**

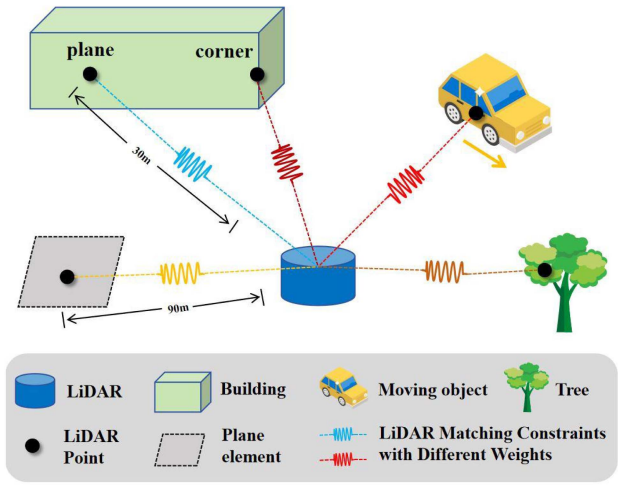

Fig. 1. Illustration of proposed weighted elastic matching (WEM) algorithm. The matching constraints are weighted from multiple dimensions to make the matching constraints elastic.

I. Introduction

In RECENT years, advancements in sensor technology have led to continuous improvements in LiDAR performance alongside reductions in its cost. LiDAR (and LiDAR-inertial) odometry [1], [2], [3], [4], [5] is increasingly being applied to robots and handheld devices. Many robots rely on low-computing chips to simultaneously operate multiple modules, and the SLAM module can only occupy a small portion of the CPU resources (usually less than 30%). Therefore, the SLAM algorithm is required to have high efficiency while ensuring accuracy. The computational cost of LiDAR odometry [6], [7], [8], [9], [10], [11], [12] is primarily distributed across three key tasks: correspondence search and construction, pose solving, and map updating. Nonlinear optimization methods are employed for pose solving, which have relatively fixed processing time. correspondence search and map updates is closely linked to the data structure of map.

Therefore, based on ivox [3], this letter proposes a multilevel voxel-based map data structure, iPvox, to improve point cloud matching and map update efficiency.

The existing matching algorithms for LiDAR Odometry mainly adopt the idea of ICP algorithm, which searches for neighboring map points based on initial pose values, and then constructs matching constraints using geometric models of point-to-point, point-to-line, and point-to-plane for pose solving. However, the quality of the matching constraints constructed by LiDAR points is not the same. If each matching constraint is treated equally, the resulting pose estimation can be skewed by constraints with larger errors. Therefore, this letter analyzes the error sources in these matching constraints and proposes a weighted elastic matching (WEM) algorithm, as shown in Fig. 1. The algorithm assigns weights to matching constraints across multiple dimensions, including measurement depth, fitting error, occupancy probability, and matching error. This weighted approach makes the constraints elastic, reducing the impact of high-error constraints on pose estimation and enhancing matching accuracy.

The error of initial pose and the search range of correspondence are crucial for LiDAR Odometry. We have tested the existing LiDAR Odometry and found that when the initial pose error is large or the motion is severe, it is necessary to increase the search range of correspondence to ensure the convergence of

Received 30 October 2024; accepted 4 February 2025. Date of publication 14 February 2025; date of current version 3 March 2025. This letter was recommended for publication by Associate Editor B. Englot and Editor J. Civera upon evaluation of the reviewers' comments. *(Corresponding authors: Yun Su; Shiliang Shao.)*

Yun Su, Ziyong Zhang, Pengfei Xu, and Yong Cao are with the Guangzhou Shiyuan Electronic Technology Company Limited, Guangzhou 110016, China (e-mail: robosu12@gmail.com; zhangziyong@cvte.com; xupengfei@cvte.com; caoyong@cvte.com).

Shiliang Shao is with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China (e-mail: shaoshiliang@sia.cn).

Hui Cheng is with the Sun Yat-Sen University, Guangzhou 510275, China (e-mail: chengh9@mail.sysu.edu.cn).

The source code of GLO has been released at https://github.com/robosu12/GLO.

Digital Object Identifier 10.1109/LRA.2025.3542704

matching. However, a larger search range will increase the proportion of outliers. Therefore, this letter proposes a progressive optimization framework (PCF), which uses different resolutions and correspondence ranges in different matching stages to ensure matching convergence and improve matching accuracy.

The main contributions of this work are as follows:

- A multilevel voxel-based map data structure (iPvox) that improves the efficiency of map updates and point cloud matching. With an occupancy probability attribute, iPvox efficiently removes dynamic and unstable points from the map.
- A weighted elastic matching(WEM) algorithm that can weight matching constraints from multiple dimensions, making the matching constraints elastic and improving matching accuracy.
- A progressive optimization framework that combines prediction, coarse matching, and fine matching(PCF). The coarse matching stage ensures the convergence of the matching and corrects the motion distortion of the point cloud, while the fine matching stage further improves accuracy.
- An open-source C++ implementation of the proposed GLO method, providing configuration files for multiple datasets to facilitate readers in reproducing the experimental results of this letter.

II. Related Works

The ICP algorithm [13] is an ancient but useful algorithm that can solve the point cloud registration problem. In order to improve performance, many variants of the ICP algorithm were later proposed [14], [15]. The existing LiDAR Odometry [6], [7], [8], [9], [10], [11], [12], [16], [17], [18], [19], [20] still adopts the ICP algorithm, which searches for correspondence, and then constructs matching constraints for pose solving. Finally, the map is updated based on the matching results.

A. Map Data Structure of LiDAR Odometry

The map data structure is closely related to the correspondence search and map update of LiDAR Odometry, and currently there are two types of data structures: tree and voxel. K-dimensional tree (k-d tree) is well-known type of data structures to split the space for kNN search, adopted by LOAM [6], LeGo-LOAM [7], and F-LOAM [8]. However, when we insert new points to and delete old points from a k-d tree, fully re-build is needed to update local map, which results in considerable computation. Therefore, **FAST-LIO2** [2], [21] proposed an incremental k-d tree data structure, **ikd-Tree**, that enables incremental updates and dynamic re-balancing, and improved the efficiency of map updating. **VoxelMap** [17] proposed an adaptive voxel mapping method. Although octree can be used for higher resolution plane fitting, small plane is susceptible to noise and has lower efficiency in map maintenance. **VoxelMap++** [18] designed a plane merging module which can merge the planes in different voxels. This method can improve matching accuracy in structured scenes, but it is not suitable for outdoor and cluttered environments. **Faster-LIO** [3] proposed a sparse incremental voxels (**iVox**) to organize the point cloud map and show that iVox can achieve higher incremental update and k-NN search speed than ik-d tree. However, iVox requires distance calculation when we insert map points, which results in additional computation time. Moreover, the voxel nodes of iVox only retain map points, requiring repeated plane feature fitting. Other algorithms such as KISS-ICP [11], CT-ICP [12], Traj-LO [20] also employ voxel structures for local map maintenance. Therefore, this letter proposes a multi-level voxel structure based on iVox [3] to further improve the efficiency of point cloud matching and map updating.

B. Point Cloud Registration of LiDAR Odometry

The existing LiDAR Odometry first preprocesses the point cloud, and then constructs matching residuals for pose solving based on different geometric models. According to different preprocessing methods for point clouds, we divide LiDAR Odometry into feature-based method and direct method.

LOAM [6] is a milestone in feature-based LiDAR Odometry, which has first proposed to use curvature to extract corner and plane features of point cloud, and then construct point-to-line and point-to-plane constraints for pose solving. **LeGo-LOAM** [7] and **F-LOAM** [8] have optimized the algorithm, but the overall approach and performance are basically equivalent. **LOAM-Livox** [22] has designed corner and plane features extraction method for solid-state LiDAR. **MULLS** [16] further extracts more types of features (ground, facade, pillar, beam, etc.), and a multi metric linear least square iterative closed point algorithm has been put forward for pose solving. **BALM** [23] and **BALM2** [24] present bundle adjustment methods for corner and plane features, which can improve the accuracy, but have poor efficiency.

Feature based methods require feature extraction and have poor adaptability to different LiDAR and environments. Therefore, in recent years, many LiDAR Odometry [10], [11], [12], [17], [18], [19], [20] have adopted the direct method, which directly constructs point-to-point or point-to-plane residuals for pose solving. **DLO** [10] adopts the GICP algorithm to match the downsampled point cloud with the map to calculate the odometer. **DLIO** [4] further integrates IMU to enhance the dynamic performance of the algorithm. **FAST-LIO** [1], [2] and **Faster-LIO** [3] utilize point-to-plane models to construct geometric constraints for point cloud, enabling the algorithm to adapt to different types of LiDAR. **KISS-ICP** [11] proposed a simple yet effective LiDAR odometry which solely relies on point-to-point metric. It also proposed an adaptive thresholding method for correspondence matching. **CT-ICP** [12], **Traj-LO** [20], and **ATI-CTLO** [25] regard the measurements of LiDAR as streaming points continuously captured at high frequency and introduced a novel continuous time approach, which incorporates the motion un-distortion into the registration. **I2EKF-LO** [26] proposed a dual Iterative Extended Kalman Filter framework that leverages state updates to iteratively mitigate motion distortion in LiDAR point clouds. These methods have better accuracy during intense motion of LiDAR, but their efficiency is lower due to the need for multiple iterations. In order to adapt to different types of LiDAR and environments, the proposed GLO also employs direct method. This letter proposes a progressive optimization framework that can adopt intermediate results to perform motion distortion correction on point cloud, ensuring efficiency while improving matching accuracy.

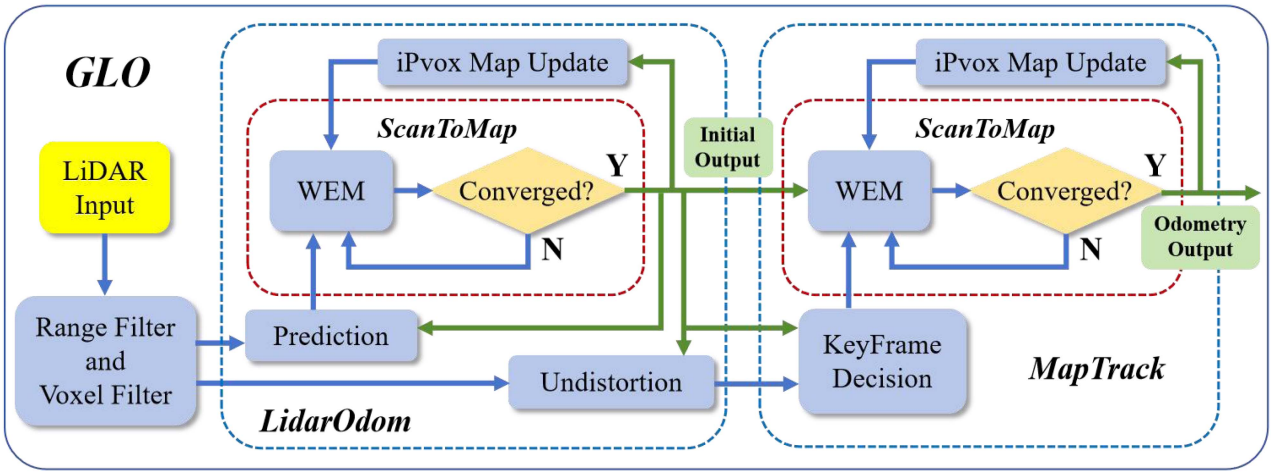

Fig. 2. The overview of proposed ***GLO*** system. System consists of three parts: Point cloud preprocessing, LidarOdom, and MapTrack. LidarOdom first performs fast coarse matching based on the prediction to obtain the initial odometry, and then uses the matching results to correct the motion distortion of the point cloud. MapTrack further performs fine matching and produces higher precision odometry. LidarOdom and MapTrack run in parallel on two threads.

III. The Proposed GLO

A. System Overview

The framework of proposed GLO is shown in Fig. 2. The system consists of three parts: point cloud preprocessing, LidarOdom, and MapTrack. During preprocessing, the point cloud is initially filtered and then downsampled using a spatial voxel structure. GLO uses a constant-velocity model for state prediction. The design principle of GLO is to adopt appropriate matching strategies for different situations. Considering that the intense movement of LiDAR can lead to prediction errors, LidarOdom adopts a larger correspondence search range to improve the convergence of matching and a lower resolution to enhance the efficiency. So, the role of LidarOdom is to output fast and robust odometry, ensuring the stability of the system. Then the odometry output by LidarOdom is used to correct motion distortion of point cloud. Based on the output of LidarOdom, MapTrack uses a smaller correspondence search range and higher resolution to further improve matching accuracy. The proposed GLO adopts progressive optimization framework from coarse to fine matching, which can improve matching accuracy while ensuring robustness. Moreover, LidarOdom and MapTrack are independent of each other and can run in parallel in two threads. Therefore, compared to CT-ICP [12], Traj-LO [20], and I2EKF-LO [26], which require multiple iterations to correct the distortion of point cloud motion, the proposed GLO has higher efficiency.

B. Map Data Structure of Ipvox

LiDAR odometry maintains a local map centered around LiDAR and performing incremental updates. The spatial voxel structure can store point cloud in a sparse manner, and each voxel is independent of each other, allowing incremental insertion and deletion. Therefore, based on iVox [3], this letter proposes a more efficient multilevel voxel structure **iPVox**, as shown in Fig. 3. The **P** in iPvox embodies three key concepts: i) Each voxel node is subdivided into smaller subvoxels, with each subvoxel retaining only one **P**oint. ii) Each voxel node actively maintains one **P**lane feature. iii) Each voxel node incorporates an occupancy **P**robability.

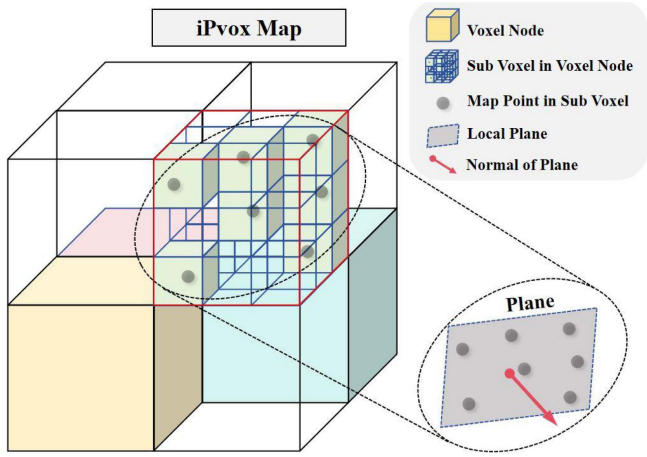

Fig. 3. Illustration of proposed iPvox for local map. Voxel nodes filled with color contain map points inside. Each voxel node is further divided into sub voxels, with each sub voxel retaining only one map point. Each voxel node contains a plane feature.

1) Incremental Updating: Map updates include two operations: insertion and deletion. This letter proposes dividing each voxel node into higher resolution subvoxels, with each subvoxel retaining only one point. Each voxel node manages its subvoxels using a hash table. If the subvoxel is empty, a map point is directly inserted; otherwise, no action is taken. The multilevel voxel structure of iPvox leverages the discrete characteristics of voxels to achieve an even distribution of map points. Compared to iVox [3], iPVox avoids point-to-point distance calculations, significantly improving the efficiency of map point insertion.

For map-point deletion, iPvox uses a least recently used (LRU) cache strategy to track recently accessed voxels and remove older ones, similar to the approach used in iVox [3].

2) Occupancy Probability Updating: The ray model is commonly used for grid probability updates; however, it is computationally intensive. Due to the sparsity of point cloud map, this letter proposes a simple and efficient hit-and-rinse (HR) probability update method based on spatial observation consistency. The steps are as follows:

- Each voxel node has an initial probability.
- When a voxel node is hit by a laser point, its probability is incremented by a specified value (Hit).
- After inserting one complete frame of point cloud, the probabilities for all voxel nodes decrease by a specified value (Rinse).

Due to the resolution of the point cloud, voxel nodes may not be consistently hit with each observation. To address this, the probability growth gradient is set to three times the descent gradient, ensuring stable probability values. The occupancy probability of voxel nodes in static environment will increase due to repeated observations by LiDAR, while for dynamic targets or noise, it decreases due to fewer observations. Since map points progressively move beyond the visible range of LiDAR, if the update number of each voxel node exceeds the set maximum value, its probability will be fixed.

The proposed HR method utilizes linear accumulation for occupancy probability update and only operates on sparse voxel nodes, which has high efficiency. Based on the probability of voxel nodes, dynamic and unstable points in the map can be efficiently removed. When map matching is performed, the point cloud should be as close as possible to the static map points with high probability, so the occupancy probability of the voxel node can provide weight for matching.

3) Active Maintenance of Plane Features: During the movement of LiDAR, voxel nodes on the map will be observed multiple times. To prevent repetitive feature fitting and improve matching efficiency, this letter introduces active maintenance of plane feature within each voxel node. Additionally, a two-level plane-fitting strategy is proposed to improve the matching recall rate:

- If a voxel node contains a sufficient number of map points, they are used to fit the fine-plane feature.
- If there are fewer map points within the voxel node, the map points of neighboring voxel nodes will be searched to fit the rough-plane feature.

This two-level plane fitting strategy can improve the recall of matching. When the number of map points within a voxel node reach its upper limit, the plane feature is fixed, and the points within the voxel node are cleared. Compared to VoxelMap [17] and iVox [3], the proposed iPvox offers both higher efficiency and reduced memory usage. Both LidarOdom and MapTrack have their own independent local maps, which are maintained using the proposed iPvox.

C. Weighted Elastic Matching (WEM)

For existing LiDAR odometry, the matching between LiDAR points and map have the same constraint force on pose, that is, it is rigid. Although methods such as GICP [14] and VoxelMap [17] utilize the covariance of local points to account for local structural consistency and weigh each matching constraint, this approach remains limited. This letter categorizes errors in matching constraints as follows:

- Due to the internal rotation and external vibration of LiDAR, measurement errors tend to increase for laser points at greater distances.
- In the point-to-plane model, lower coplanarity of map points results in poorer constraint quality.
- Map points on dynamic targets.
- The greater the distance from a point to a plane, the higher the possibility of a mismatch.

This letter introduces a WEM algorithm that assigns weights to matching constraints across the four identified dimensions, reducing the impact of low-quality matches on pose estimation and enhancing matching accuracy. As shown in Fig. 1, each match receives a weight tailored to specific conditions, rendering the matching constraints elastic. The algorithm first applies depth-based weighting for matching constraints. The depth weight, ω^d, is calculated as follows:

$$\omega_i^d = \frac{G_r}{1 + 2.5 r_i / R_{\max}} \tag{1}$$

where G_r is the depth weight gain, r_i is the depth of the laser point, and $R_{\max}$ is the maximum measurement distance of the LiDAR. As the depth of the laser point increases, the weight decreases. When the depth is 0.4 times the maximum measurement distance of the LiDAR, the depth weight reduces to half of the initial value.

Faster-LIO [3], F-LOAM [8], and similar methods directly discard matches with large plane errors. For map features with poor coplanarity, even imperfect matches can provide useful surface constraints and should be retained in optimization with lower weights rather than discarded. Thus, this letter uses plane fitting error to calculate structural weight:

$$\omega_i^s = \frac{G_s}{1 + 0.5 d_i^{fit} / R_{map}} \tag{2}$$

where G_s is the structural weight gain, R_{map} is the map resolution, and d_i^{fit} is the plane fitting error. The structural weighting strategy ensures that the point cloud is as close as possible to the planar area of the map, reducing the impact of cluttered objects on the matching results. When running in unstructured scenarios, the overall weight of the structure will decrease, but it will not reduce the number of matching constraints, which can ensure the stability of the matching and make the algorithm more adaptable and flexible.

Due to the expanded field of view, LiDAR generates new observation points at the edge of the map, which are susceptible to mismatches. In addition, map points on dynamic targets are also prone to mismatches. Since edge and dynamic map points have low occupancy probabilities, this letter employs the occupancy probability of voxel nodes to calculate probability weight as follows:

$$\omega_i^p = 0.75 p_i^{patch} / P_{init}. \tag{3}$$

where p_i^{patch} is the occupancy probability of the voxel node currently matched, and P_{init} is the initial probability. As can be seen, for the newly added voxel nodes, their probability weight is 0.75. Probability weight will make the matching results to approach static and stable regions with high occupancy probability, thereby improving matching accuracy.

When prediction errors are large, the overall matching residual may increase, and removing outliers with a fixed threshold can result in an insufficient number of matching constraints. This letter introduces a proportional weighting method to reduce the influence of outliers and maintain robust matching under large prediction errors. The matches are first sorted by residual, and the residual weight for the matching constraints are then calculated as follows:

$$\omega_i^r = 1.0 - 0.5 * i / N, i < 0.96 * N. \tag{4}$$

where N is the number of matches and i is the sorted index. It can be seen that the lower the ranking of matching constraint is, the lower the weight is. The last 4% of matching constraints will be discarded. This strategy can ensure sufficient constraints and dynamically reduce the influence of outliers based on the matching situation.

Finally, the matching result is estimated by minimizing the following weighted residuals:

$$\hat{\mathbf{X}}_k^w = \arg\min_{\mathbf{X}_k^w} \sum_i^N \omega_i^d \omega_i^s \omega_i^p \omega_i^r f(p_i, \mathbf{X}_k^w). \tag{5}$$

where $f(p_i,\ \mathbf{X}_k^w)$ is the distance from the LiDAR point to the plane feature. The matching optimization process will undergo multiple iterations. If the pose increment after optimization is less than the set threshold or the number of iterations is greater than the set threshold, it is considered to have converged and the matching result returns.

D. Odometry Estimation

Existing LiDAR-only odometry faces two primary issues. First, while a large correspondence search range enhances matching convergence, it also increases the number of outliers. Second, algorithms such as KISS-ICP [11] apply a constant velocity model for correcting point-cloud motion distortion, which can result in errors when LiDAR experiences rapid movement. Continuous-time methods, such as those used in CT-ICP [12] and Traj-LO [20], correct for distortion during matching iterations but are computationally intensive.

To address these issues, this letter proposes a progressive optimization framework that combines prediction, coarse matching, and fine matching, as illustrated in Fig. 2. Firstly, a constant-velocity model is applied for preliminary distortion correction of the point cloud and pose prediction. LidarOdom then utilizes a larger correspondence search range to improve matching recall and robustness, and a coarser resolution to enhance efficiency. Finally, motion distortion correction is performed again based on the initial odometry.

LidarOdom ensures motion continuity, so only point cloud that undergoes significant motion can be judged as keyframe by MapTrack for matching, thereby avoiding redundant matching and improving efficiency. Based on the initial odometry, MapTrack uses a smaller correspondence range to reduce outliers and a higher resolution to improve accuracy. Then the WEM algorithm is used again to match the refined point cloud with local map, yielding more accurate odometry results. LidarOdom and MapTrack operate independently and can run in parallel on separate threads to improve efficiency.

IV. Experiments

To validate the accuracy, efficiency, and robustness of the proposed GLO method across various scenes, platforms, and LiDAR types, we conducted experiments using multiple datasets, including autonomous vehicles, robots, unmanned aerial vehicles, and handheld device. And configuration files for each dataset are included in the open-source code, allowing readers to easily reproduce the experimental results.

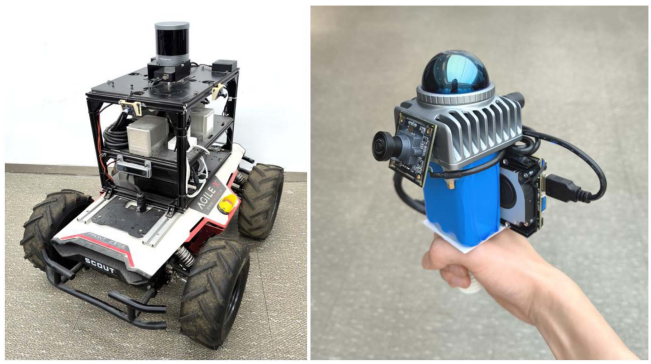

Fig. 4. Hardware platforms used for the GR-dataset.

A. Datasets

KITTI odometry dataset [27] is an autonomous driving dataset, where the LiDAR scans are collected from a Velodyne HDL-64E mounted on a car in urban environments and the in-frame motion has been compensated in advance. As in [29], a vertical angle of 0.22 degree is employed to correct the calibration errors in raw point clouds.

NTU VIRAL [28] is a visual-inertial-ranging-LiDAR dataset for unmanned aerial vehicles. It has two 16-channel Ouster LiDARs and we use the one installed horizontally.

GR-dataset is a multi-modal dataset collected from robots and handheld device, incorporating spinning and solid-state LiDAR, cameras, IMU, wheel odometry, and GNSS. Fig. 4 presents some of hardware used in GR-dataset. To evaluate the algorithm's performance with solid-state LiDAR, we selected two sequences—**Park** and **Street**—from the GR dataset. The Park sequence is collected in an industrial park based on a wheeled robot platform, covers a trajectory length of 586m. The Street sequence, collected on a street with handheld devices, spans a trajectory length of 893 m.

B. Accuracy Evaluation

Firstly, we evaluated the accuracy on the KITTI dataset and compared it with F-LOAM [8], KISS-ICP [11], CT-ICP [12], Traj-LO [20], MULLS [16], and MAD-ICP [29]. Table I shows that F-LOAM and MAD-ICP exhibit the highest odometry errors. MULLS achieves better accuracy by extracting a wider variety of features for matching. CT-ICP, KISS-ICP, and Traj-LO demonstrate comparable odometry accuracies. The proposed GLO achieves the highest average accuracy, obtaining the best performance on six sequences and ranking highly on others. This improvement is attributed to the WEM algorithm, which reduces the influence of low-quality matching constraints, thereby enhancing matching accuracy.

We evaluated the accuracy of the algorithm on the NTU VIRAL dataset, which includes more intense motion. Trajectory errors are shown in Table II, and mapping results are shown in Fig. 5. Both F-LOAM and KISS-ICP exhibit substantial errors. Although KISS-ICP applies a constant velocity model to correct the motion distortion of the point cloud, the rapid speed changes of the drone lead to considerable errors. CT-ICP and Traj-LO, which employ continuous-time methods to compensate for the motion of point cloud, demonstrate smaller trajectory errors. However, CT-ICP proved unstable, with significant errors and

TABLE I
RELATIVE TRANSLATION ERROR (RTE) [%] ON KITTI ODOMETRY DATASET

Approach	00	01	02	03	04	05	06	07	08	09	10	Average
F-LOAM[8] (2021)	0.71	0.71	0.73	0.98	0.57	0.62	0.33	0.47	1.04	0.88	1.02	**0.73**
MULLS[16] (2021)	0.54	0.62	0.69	0.61	**0.35**	0.29	0.30	0.38	0.83	0.51	0.61	**0.52**
CT-ICP[12] (2022)	**0.49**	0.76	**0.52**	0.72	0.39	0.26	0.27	0.31	**0.81**	0.49	**0.48**	**0.50**
KISS-ICP[11] (2023)	0.52	0.72	0.53	0.65	**0.35**	0.30	**0.26**	0.33	**0.81**	0.49	0.54	**0.50**
Traj-LO[20] (2024)	0.51	0.81	**0.52**	0.67	0.40	**0.25**	0.27	**0.30**	**0.81**	0.45	0.55	**0.50**
MAD-ICP[29] (2024)	0.73	0.85	0.76	0.84	0.69	0.48	0.48	0.45	1.21	1.02	0.96	**0.77**
GLO(ours)	0.51	**0.53**	**0.52**	**0.63**	0.46	**0.25**	0.28	**0.30**	0.82	**0.44**	0.59	**0.48**

TABLE II
ABSOLUTE TRANSLATION ERROR (ATE) [M] ON NTU VIRAL DATASET

Approach	eee-03	nya-03	sbs-03	rtp-03	spms-03
F-LOAM	1.13	1.50	1.08	2.22	failed
KISS-ICP	1.01	1.27	1.03	2.38	5.45
CT-ICP	11.17	0.07	1.55	0.09	failed
Traj-LO	**0.04**	**0.05**	**0.04**	**0.06**	**0.10**
GLO(ours)	0.09	0.08	0.05	0.17	0.14

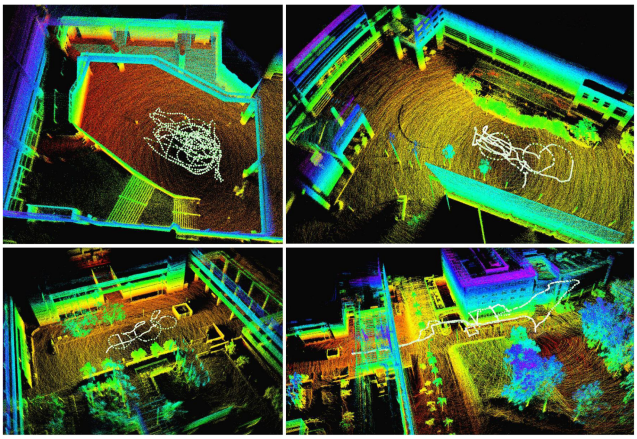

Fig. 5. The mapping results of GLO on the NTU dataset. The white line represents the trajectory of the LiDAR.

TABLE III
ABSOLUTE TRANSLATION ERROR (ATE) [M] ON GR-DATASET

GR dataset	F-LOAM	KISS-ICP	Traj-LO	**GLO(ours)**
Park (586m)	-	2.34	1.58	**1.22**
Street (893m)	-	4.93	4.89	**4.45**

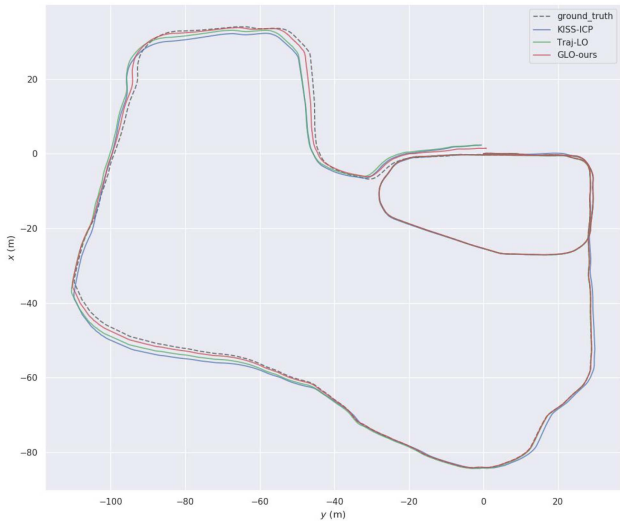

Fig. 6. The trajectory results on the ***Park*** sequence of GR-dataset.

even failures on eee-03 and spms-03 sequences. Traj-LO divides the point cloud into smaller segments, resulting in the highest accuracy. The accuracy of GLO is comparable to Traj-LO, with an average error of less than 0.1 m. This performance is attributed to the proposed progressive optimization framework, which not only uses initial odometry to correct point-cloud motion distortion but also provides accurate predictions for fine matching, enhancing both robustness and accuracy. In addition, it is noteworthy that the efficiency of GLO is significantly higher than that of Traj-LO, as discussed in detail in the efficiency evaluation section.

To evaluate the adaptability of the algorithm to different LiDAR and environments, we conducted experiments on the GR-dataset. Trajectory errors are shown in Table III, and the trajectory of the Park sequence is shown in Fig. 6. F-LOAM is not suitable for solid-state LiDAR. In the Park sequence, which features buildings and trees, KISS-ICP's reliance on a point-to-point model leads to significant errors. The Street sequence contains a large number of trees and fewer planar elements, so the errors of KISS-ICP and Traj-LO are close. The proposed GLO demonstrated the smallest error across both sequences. The WEM algorithm proposed in this letter effectively reduces the weight of non-planar features while retaining sufficient matching constraints, thereby ensuring matching accuracy and stability even in environments with limited planar elements. The GR-dataset verifies the adaptability of GLO to different LiDAR and environments.

C. Efficiency Evaluation

Then we evaluated the efficiency of the algorithm. All the experiments are conducted on an Intel i7-10875H CPU (2.30 GHz x 8 cores) with 16GB RAM. To ensure fairness, we limited all algorithms to two CPU cores, despite GLO's ability to accelerate with multithreading. Figs. 7 and 8 show the results on the KITTI and NTU datasets, respectively. The average processing time for Traj-LO was approximately 120 ms, while F-LOAM and KISS-ICP averaged approximately 60 ms. Although Traj-LO achieves high accuracy, its use of complex continuous-time

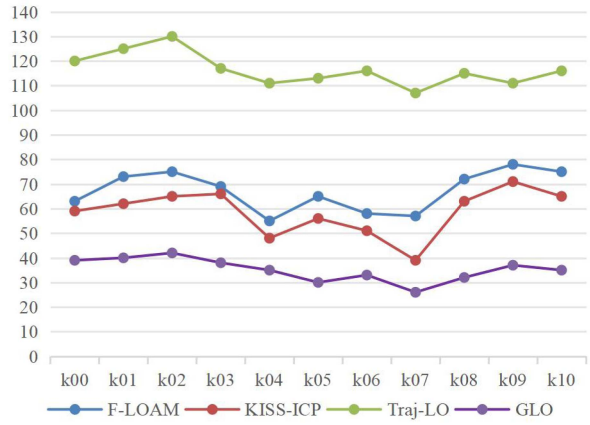

Fig. 7. Running time (ms) on the KITTI dataset.

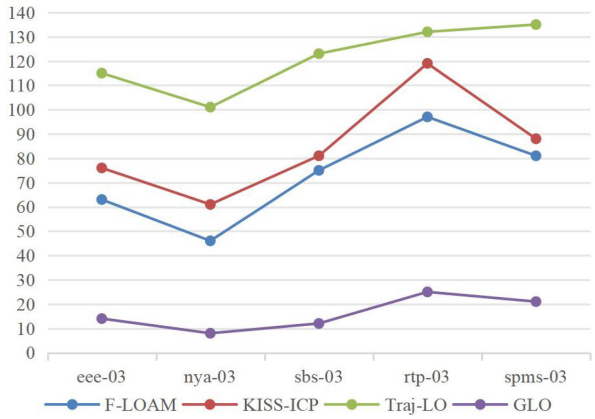

Fig. 8. Running time (ms) on the NTU dataset.

TABLE IV
RUNNING TIME (MS) OF DIFFERENT STEPS ON THE NTU DATASET

steps	GLO(ours)	KISS-ICP	F-LOAM	Faster-lio
Correspondence	**2.8**	23.3	26.5	5.6
Optimization	4.5	**3.9**	4.2	18.5
Map updating	**1.8**	4.6	11.3	3.8

methods reduces efficiency. The proposed GLO exhibited an average processing time of 35 ms on the KITTI dataset and only 17 ms on the NTU dataset, with a notable 8 ms processing time on the nya-03 sequence. This efficiency is attributed to iPvox's multilevel voxel structure, which improves map point insertion. Additionally, iPvox actively maintains plane features, and once the feature fitting converges, the feature parameters are fixed, minimizing map matching time and achieving high efficiency.

Figs. 7 and 8 also reveal that the processing time of GLO on the NTU dataset is significantly lower than on KITTI. This difference arises because the motion range of the LiDAR in the NTU dataset is limited (as shown by the white trajectory in Fig. 5), leading to longer feature lifespans and fixed feature parameters across most of the map. Therefore, GLO achieves higher efficiency on the NTU dataset than on KITTI.

We have further analyzed the running time for each step, with results displayed in Table IV. The pose optimization times are generally comparable (except for faster-lio). However, GLO excels with notably lower time for both correspondence and map updating compared to other algorithms. This indicates that the proposed iPvox has higher efficiency for point cloud map operations.

TABLE V
ABLATION EXPERIMENT ON KITTI DATASET

Approach	00	01	02	04	06	08	10	Ave
GLO-full	**0.51**	**0.53**	**0.52**	0.46	0.28	**0.82**	0.59	**0.53**
GLO-nd	0.52	0.55	0.53	0.46	0.28	0.84	**0.58**	**0.54**
GLO-ns	0.55	0.55	0.57	0.48	**0.26**	0.86	0.73	**0.57**
GLO-np	0.52	0.55	0.53	0.46	0.28	0.86	0.61	**0.54**
GLO-nr	0.54	**0.53**	0.54	0.47	0.28	0.83	0.65	**0.55**
GLO-na	0.55	0.54	0.58	0.48	**0.26**	0.89	0.71	**0.58**
Traj-LO[20]	0.51	0.81	0.52	0.40	0.27	0.81	0.55	**0.55**

All errors are represented as relative translation error (rte) [%]. Denotations: -full: with all weights. -nd: without depth weight. -ns: without structural weight. -np: without probability weight. -nr: without residual weight. -na: without all weights.

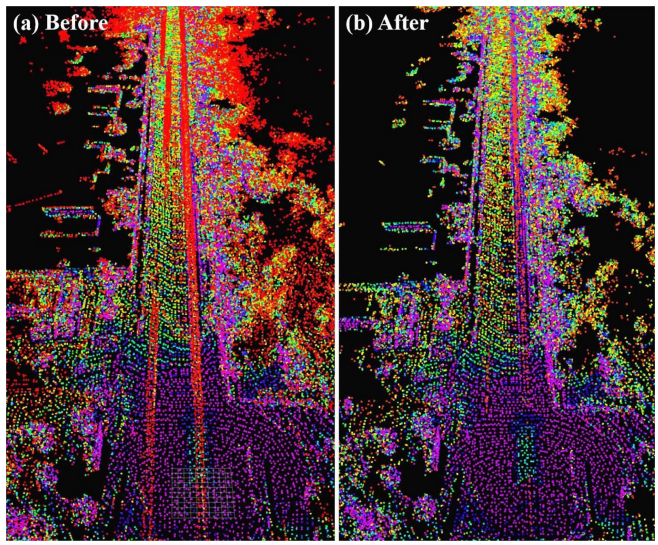

Fig. 9. Dynamic objects removal results of GLO on the KITTI-06 sequence. The colors of map points represent different occupancy probabilities, with red being the lowest and purple being the highest.

D. Dynamic and Unstable Map Points Removal

The proposed iPvox leverages occupancy probability of voxel nodes to remove dynamic targets and noise in the map. This capability was evaluated using the KITTI-06 sequence, with results shown in Fig. 9, where map point colors represent different occupancy probabilities (red for lowest, purple for highest). As observed, static features such as ground surfaces and buildings appear in purple, indicating high occupancy probability. The red strip-shaped point cloud generated by moving vehicles in the middle of the road were effectively removed. The unstable point cloud and noise generated by trees on both sides of the road also were removed. It is worth noting that compared with algorithms such as [30], [31], [32], [33], [34], the iPvox proposed in this letter can more easily and effectively remove dynamic and unstable map points.

E. Ablation Experiment of WEM Algorithm

Furthermore, we evaluated the performance of various weighting strategies. As shown in Table V with our ablation experiments on the KITTI dataset, it's clear that structural and residual weights have a more substantial impact on results compared to depth and probability weights. Structural weight help reduce errors from non-coplanar features, while residual weight address errors caused by outliers—both critical sources of matching errors. When all weights are removed, the average error is maximized. These results indicate that weights from different dimensions can have positive effect on the matching, thereby proving the effectiveness of the proposed method.

V. Conclusion

This study proposes a general, efficient, and accurate LiDAR odometry GLO, which improves accuracy and robustness through weighted elastic matching algorithm and progressive optimization strategy. Additionally, a multilevel voxel structure, iPvox, was employed to improve efficiency and eliminate dynamic map points. Experimental results across multiple datasets demonstrated that the proposed GLO achieved high accuracy, with efficiency gains of two to four times over existing algorithms. Moreover, GLO proved adaptable to various LiDAR and scenarios. Open-source code and configuration files are provided to enable readers to reproduce the experimental results. In future work, we plan to integrate LiDAR, IMU, wheel odometry, and GNSS to further enhance algorithm performance.

References

[1] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021.

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

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

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

[5] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

[6] J. Zhang and S. Singh, "Loam: LiDAR odometry and mapping in realtime," *Robot., Sci. Syst.*, vol. 2, pp. 1–9, 2014.

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

[8] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-LOAM: Fast LiDAR odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4390–4396.

[9] S. Yi, Y. Lyu, L. Hua, Q. Pan, and C. Zhao, "Light-LOAM: A lightweight LiDAR odometry and mapping based on graph-matching," *IEEE Robot. Automat. Lett.*, vol. 9, no. 4, pp. 3219–3226, Apr. 2024.

[10] K. Chen, B. T. Lopez, A.-A. Agha-Mohammadi, and A. Mehta, "Direct LiDAR odometry: Fast localization with dense point clouds," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2000–2007, Apr. 2022.

[11] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In defense of point-to-point ICP – Simple, accurate, and robust registration if done the right way," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 1029–1036, Feb. 2023.

[12] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP: Real-time- elastic LiDAR odometry with loop closure," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 5580–5586.

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

[14] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Robot., Sci. Syst.*, vol. 2, no. 4, p. 435, 2009.

[15] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11054–11059.

[16] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11633–11640.

[17] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online LiDAR odometry," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8518–8525, Jul. 2022.

[18] C. Wu et al., "VoxelMap++: Mergeable voxel mapping method for online LiDAR (-Inertial) odometry," *IEEE Robot. Automat. Lett.*, vol. 9, no. 1, pp. 427–434, Jan. 2024.

[19] Z. Chen, Y. Xu, S. Yuan, and L. Xie, "iG-LIO: An incremental GICP-based tightly-coupled LiDAR-inertial odometry," *IEEE Robot. Automat. Lett.*, vol. 9, no. 2, pp. 1883–1890, Feb. 2024.

[20] X. Zheng and J. Zhu, "Traj-LO: In defense of LiDAR -only odometry using an effective continuous-time trajectory," *IEEE Robot. Automat. Lett.*, vol. 9, no. 2, pp. 1961–1968, Feb. 2024.

[21] Y. Cai, W. Xu, and F. Zhang, "ikd-tree: An incremental kd tree for robotic applications," 2021, *arXiv:2102.10808*.

[22] J. Lin and F. Zhang, "Loam-livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3126–3131.

[23] Z. Liu and F. Zhang, "Balm: Bundle adjustment for lidar mapping," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3184–3191, Apr. 2021.

[24] Z. Liu, X. Liu, and F. Zhang, "Efficient and consistent bundle adjustment on lidar point clouds," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4366–4386, Dec. 2023.

[25] B. Zhou, J. Wu, Y. Pan, and C. Lu, "ATI-CTLO: Adaptive temporal interval-based continuous-time LiDAR-only odometry," *IEEE Robot. Autom. Letters*, vol. 9, no. 12, pp. 11162–11169, Dec. 2024.

[26] W. Yu et al., "I^{2}EKF-LO: A dual-iteration extended kalman filter based LiDAR odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 10453–10460.

[27] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The kITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[28] 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," *Int. J. Robot. Res.*, vol. 41, no. 3, pp. 270–280, 2022.

[29] S. Ferrari, L. D. Giammarino, L. Brizi, and G. Grisetti, "MAD-ICP: It is all about matching data–robust and informed LiDAR odometry," *IEEE Robot. Automat. Lett.*, vol. 9, no. 11, pp. 9175–9182, Nov. 2024.

[30] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10758–10765.

[31] J. Schauer and A. Nuchter, "The peopleremover—Removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1679–1686, Jul. 2018.

[32] H. Lim, S. Hwang, and H. Myung, "ERASOR: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3D point cloud map building," *IEEE Robot. Autom. Letters*, vol. 6, no. 2, pp. 2272–2279, Apr. 2021.

[33] C. Qian, "RF-LIO: Removal-first tightly-coupled lidar inertial odometry in high dynamic environments," in *Proc. 2021 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Prague, Czech Republic, 2021, pp. 4421–4428.

[34] T. Fan, B. Shen, H. Chen, W. Zhang, and J. Pan, "Dynamicfilter: An online dynamic objects removal framework for highly dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 7988–7994.