An Efficient Odometry and Mapping Framework Based on Multi-LiDAR Systems

Mingyue Cui[1], Chiawen Liao[1], Yuxuan Chen[1], Junda Lai[1], Zifeng Deng, and Kai Huang[1,*]

***Abstract*— Due to the accuracy and reliability of range measurements, LiDAR-based odometry and mapping have been the focus of research in autonomous driving. However, a single LiDAR is usually limited by the field of view (FoV) occlusion and data sparsity, which bring challenges for accurate pose estimation. To address this problem, we propose an efficient odometry and mapping framework based on a multi-LiDAR system. We first propose an online extrinsic calibration method that combines hand-eye calibration and optimization based on local map features. Then we adopt the sliding window with bundle adjustment to match secondary extracted structured features from fused point clouds for front-end odometry. In addition, we design a global manifold optimization strategy based on a factor graph with loop closure constraints and a weight noise model to optimize the states marginalized after the bundle adjustment. The experimental results indicate that our method achieves higher quality performance compared to other single LiDAR-based methods.**

I. INTRODUCTION

In recent years, LiDAR has gradually become the core sensor for its accurate and intuitive 3D point cloud information, which is widely used in simultaneous localization and mapping (SLAM) systems. SLAM is a technology for robots to locate their own position and model environment in unknown environments [1], and the main process is to estimate their states by using constructed incremental environment map. An efficient odometry and mapping method is essential for the detection, segmentation, planning, and other fields of intelligent vehicles.

To make accurate pose estimation, researchers have conducted extensive study [2], [3]. The iterative closest point algorithm (ICP) [4] is the traditional point-wise matching algorithm, which seeks the corresponding relationships point-wise and aligns the associated point sets periodically, iterating until convergence. Li He *et al.* further propose GICP [5], which combines point-to-point, point-to-plane, and plane-to-plane residuals as optimization strategies, introducing covariance matrices with probabilistic information to generate a unified model for ICP. Different from the above point-wise based matching methods, the famous LOAM [2] extracts edge points and planar points for point-to-line and point-to-plane feature matching. However, the sparsity and Fov occlusion of a single LiDAR limit the performance and operation range of SLAM algorithms.

*Corresponding author.

[1]The authors are with School of Computer Science and Engineering, Sun Yat-sen University, 510006, Guangzhou, P.R. China, {cuimy, liaocw3, chenyx558, laijd7, dengzf}@mail2.sysu.edu.cn, huangk36@mail.sysu.edu.cn.

To address this problem, this paper proposes an advanced odometry and mapping framework that leverages a multi-LiDAR system for enhanced performance. Our method mainly consists of the following four aspects: two-step extrinsic calibration, the secondary feature extraction for front-end odometry, the factor graph with loop closure constraints, and a weighted noise model for back-end optimization. Specifically, we adopt hand-eye calibration for initial extrinsic parameters and then utilize local map matching to acquire accurate and stable extrinsic parameters. After that, we perform feature matching using keyframes and sliding windows to obtain the initial motion pose estimation. At the same time, the bundle adjustment is applied on the front-end odometer pose to reduce the cumulative drift in the sliding windows. Finally, we design a global manifold optimization model based on the factor graph using loop closure constraints and a weight noise model to optimize the keyframes marginalized after bundle adjustment. The contributions of this paper are summarized as follows:

- We propose an efficient odometry and mapping framework with online extrinsic calibration based on a multi-LiDAR system, which can improve the accuracy of motion pose effectively.
- By introducing a factor graph with loop closure constraints and a weight noise model, our method achieves excellent back-end optimization performance.
- We conduct experiments on both public dataset and real-world deployment. The experimental results show that compared to other methods, our method obtains better localization and mapping performance.

II. RELATED WORK

According to different back-end optimization methods, LiDAR SLAM is divided into two main categories: filter-based SLAM and optimization-based SLAM. Filter-based SLAM is derived from Bayesian filtering, which solves for the current optimal state estimation based only on the previous step states. The classical EKF-SLAM [6], which takes in observed landmarks from the environment and compares them with known landmarks to find associations and new landmarks, applies an extended Kalman filter to online SLAM using maximum likelihood data association. The optimization-based SLAM usually optimizes the batch states uniformly. LeGO-LOAM [3] is a widespread concern LiDAR SLAM framework based on LOAM. LeGO-LOAM further adds back-end graph optimization and ground constraint modules to improve the odometry and mapping performance of robots moving in the plane. HDL [7] fuses the information

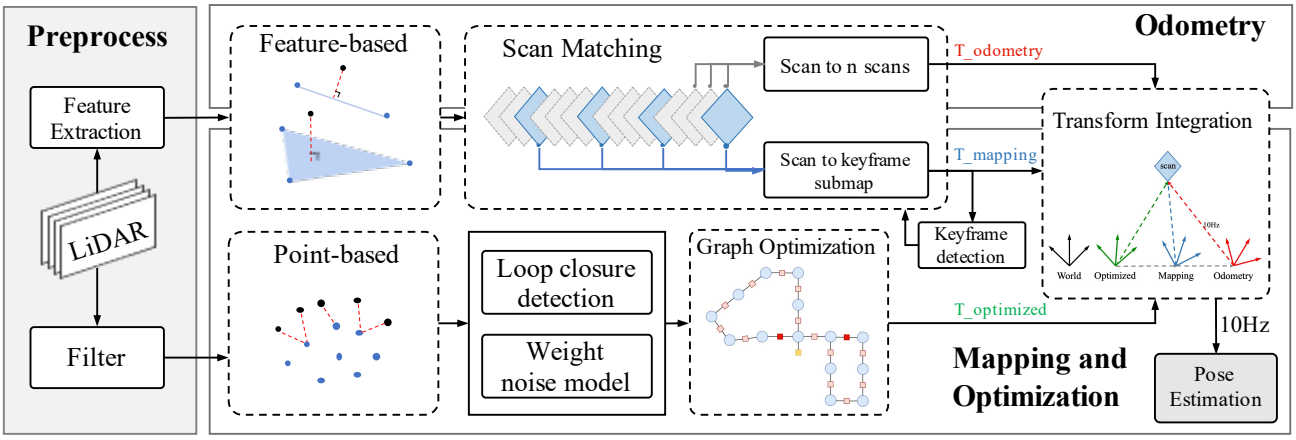


Fig. 1: The framework of our method. The input point clouds are from multiple LiDAR sensors and further refine it using a preprocessing module. We design the two-step calibration with initial hand-eye calibration and further calibration optimization based on local map matching. We adopt sliding windows with bundle adjustment to match secondary extracted structured features for the front-end odometry. The back-end mapping and optimization further combine the loop closure detection and weight noise model with factor graphs to obtain refined poses and map information.

of IMU, GPS, and ground detection as additional constraints of the graph for pose graph update and loop detection, which aims to obtain a better graph optimization effect. For bundle adjustment, BALM [8] proposes a search method based on adaptive voxel laser point cloud characteristics using sliding windows. By deriving the gradient and Hessian matrix of the cost function, efficient bundle adjustment optimization is achieved.

Up till today, most studies focus on single-LiDAR-based SLAM and there have not been many thorough prior studies on applying multiple LiDARs to SLAM, especially evaluated with real-life experiments on the vehicle, which prompted this article.

III. METHOD

As shown in Figure 1, we present the proposed framework using multi-LiDAR systems to make odometry and mapping efficiently. Firstly, we filter the noise and outliers of point clouds and extract strong planar features such as the ground and surfaces with low curvature, which ensures more robust and reliable constraints. On this basis, we perform point cloud fusion for different LiDARs using hand-eye calibration based on motion correlation analysis. Secondly, to ensure accurate estimation through bundle adjustment while maintaining a significant number of historical feature points, we adopt a meticulous keyframe selection strategy that prioritizes data with high significance and informativeness. The front-end odometry performs structured feature matching using keyframes and sliding windows to obtain initial motion pose estimation. Finally, in the back-end optimization, we design a global manifold optimization model based on a factor graph with loop closure constraints and a weight noise model to optimize the states marginalized after bundle adjustment. In this process, we utilize local map matching for optimization calibration to obtain accurate and stable extrinsic parameters.

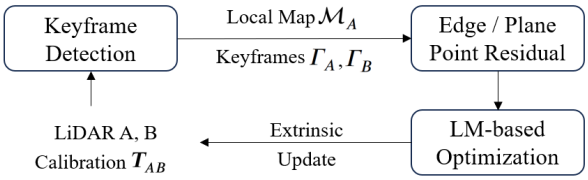


Fig. 2: Calibration optimization

A. Preprocess and Calibration

For the raw point cloud, by calculating the average distance of all neighboring points, the preprocessing module firstly filters the noise and outliers data caused by occlusion. We extract point clouds from 0-80 meters above the ground except the ground to ensure adequate features with small errors. Considering the differences in the placement and timing of different LiDARs, it is necessary to calibrate and fuse them. For initial extrinsic calibration, we use hand-eye calibration based on motion correlation analysis to fuse point cloud data from different LiDARs. For feature extraction, unlike LOAM [2], we introduce the secondary planar feature extraction to further extract robust planar features, such as the ground and surfaces with low curvature, which further enhances the accuracy of feature matching.

To further improve the calibration accuracy, we perform calibration optimization with local map feature matching based on hand-eye calibration. As shown in Figure 2, we describe the two-step calibration optimization process. By combining feature matching and utilizing dense structured features in the local map, we construct minimum residual constraints. With initial calibration accomplished, keyframe sets Γ_A, Γ_B are further detected from multiple LiDARs and we construct a local map $\mathcal{M}_A$ with keyframes of LiDAR A for subsequent matching. The initial extrinsic parameters are utilized to calculate the pose transformations to $\mathcal{M}_A$ at each keyframe from LiDAR B. With the transformation converting

features of keyframes to the local map, we can derive the nearest straight line in the local map corresponding to the edge features. Similarly, for plane features, a plane in $\mathcal{M}_A$ corresponding to the plane features is derived for subsequent residual calculation. The residual calculation function is as follows:

$$\mathbf{T}^*_{\mathbf{AB}} = \arg\min_{\mathbf{T_{AB}}} \sum \sum \left(\mathbf{r_{edge}} + \mathbf{r_{plane}}\right), \tag{1}$$

$\mathbf{T_{AB}}$ denotes the initial extrinsic parameters of LiDAR A to B, $\mathbf{r_{edge}}$ represents the residual of edge features, and $\mathbf{r_{plane}}$ is the residual of plane features. By minimizing the objective function 1 using Levenberg-Marquardt Algorithm [9], the optimized extrinsic parameter $\mathbf{T}^*_{\mathbf{AB}}$ can be obtained. By incorporating structured residual constraints based on frame-to-map matching, both the rotation and translation extrinsic parameters are optimized.

B. Front-end Odometry

The front-end odometry consists of two main components: keyframe selection and feature matching within the sliding windows. Selected keyframes serve as reference points for subsequent frame-to-frame feature matching and enable the correction of accumulated drift. Keyframe selection involves evaluating the computed transformation between the current and the previous frames. If the translation and rotation exceed a predefined threshold, the current frame is then classified as a keyframe. All keyframes are stored in the keyframe set using shared memory. This selection criterion ensures that keyframes capture significant changes in robot pose, enabling accurate tracking and mapping.

In our model, we estimate the current pose with feature matching between frames and the sliding windows, which is periodically optimized by bundle adjustment and contains a set of keyframes. Assuming that the current sliding window is $\mathcal{W}_i$ and the keyframes within are $\mathbb{F}_{i-m:i}$:

$$\mathcal{W}_i = \mathbb{F}_i \oplus \mathbb{F}_{i-1} \oplus \cdots \oplus \mathbb{F}_{i-m}, \tag{2}$$

$$\mathcal{W}_i = \{\mathcal{W}_{\mathcal{E}_i}, \mathcal{W}_{\mathcal{G}_i}\}. \tag{3}$$

Each keyframe $\mathbb{F}_i$ contains edge and plane features $\mathcal{E}_i$, $\mathcal{G}_i$ respectively, and they are all located in the local coordinate system, while the sliding window $\mathcal{W}_i$ located in the world coordinate system. $\oplus$ is LiDAR frame concatenation operation, which projects the point cloud to the world coordinate system. Algorithm 1 describes the frame-to-frame feature-matching process based on a keyframe sliding window. When a new frame arrives, we first predict current pose $\mathbf{T_{pred}} = \mathbf{T_i} \cdot (\mathbf{T_{i-1}})^{-1} \cdot \mathbf{T_i}$ with past two frames $\mathbf{T_{i-1}}$ and $\mathbf{T_i}$. Subsequently, the algorithm proceeds by iterating through the edge features and plane features of the current frame, respectively selecting five nearest neighbor points from the sliding window for each type of feature. For the edge feature, with centers and eigenvectors of the five neighboring points, we can derive two points on estimated straight line, denoting as $\mathbf{p}_j$, $\mathbf{p}_k$. Similarly, for plane features, the algorithm examines whether these neighbor points are able to approximate a plane. By minimizing the cost function calculated using the point-to-line distance and point-to-plane distance, the pose of front-end odometry $\mathbf{T}^*_{i+1}$ can be approximately estimated.

Algorithm 1 Frame-to-frame feature extraction

Input : Current features $\mathcal{E}_{i+1}$, $\mathcal{G}_{i+1}$
Features $\mathcal{W}_{\mathcal{E}i}$, $\mathcal{W}_{\mathcal{G}i}$ within sliding windows
Pose prediction of current frame $\mathbf{T}_{\text{pred}}$
Output: Optimal estimated pose $\mathbf{T}^*_{i+1}$

$\mathbf{T}_{i+1} = \mathbf{T}_{\text{pred}}$
$\dot{\mathcal{E}}_{i+1} = \boldsymbol{T}_{\text{i+1}}\mathcal{E}_{i+1}, \quad \dot{\mathcal{G}}_{i+1} = \boldsymbol{T}_{\text{i+1}}\mathcal{G}_{i+1}$;
for feature point $\dot{\boldsymbol{p}}_i$ in $\dot{\mathcal{E}}_{i+1}$,$\dot{\mathcal{G}}_{i+1}$ **do**
  if Edge features **then**
    approximate a line with 5 nearest points in $\mathcal{W}_{\mathcal{E}_i}$
    describe the line with points $\mathbf{p}_j$, $\mathbf{p}_k$ on it
    $r_{\text{edge}} = |(\boldsymbol{p}_i - \boldsymbol{p}_j) \times (\boldsymbol{p}_i - \boldsymbol{p}_k)| / |\boldsymbol{p}_j - \boldsymbol{p}_k|$;
  else
    approximate a plane with 5 nearest points in $\mathcal{W}_{\mathcal{G}_i}$
    with normal vector of the plane $\mathbf{n}$ and scalar d
    $r_{\text{plane}} = \boldsymbol{n}^\top \boldsymbol{p}_i + d$
  $\mathbf{T}^*_{i+1} = \arg\min_{\mathbf{T}_{i+1}} \{\sum r_{\text{edge}} + \sum r_{\text{plane}}\}$
return $\mathbf{T}^*_{i+1}$

C. Back-end Optimization

To ensure global consistency and enhance overall accuracy of the SLAM system, we begin by conducting bundle adjustment, which achieves local optimization by simultaneously constraining common features between different frames within the sliding windows. We approximate the feature points of the same feature as a plane. Subsequently, we measure the distance between these feature points and the plane as residuals. It is essential to map these feature points from different frames to world coordinates based on their respective poses before conducting the adjustment. The objective function can be expressed as follows:

$$(\mathbf{T}^*, \mathbf{n}^*, \mathbf{q}^*) = \arg\min_{\mathbf{T},\mathbf{n},\mathbf{q}} \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{n}^\top (\mathbf{p}_i - \mathbf{q})\right)^2, \tag{4}$$

where $\mathbf{T}^*$ represents the pose to be optimized in the sliding window, $\mathbf{q}$ represents a point on the plane, $\mathbf{n}$ represents the normal vector of the plane, $\mathbf{i} \in \{\mathbf{1}, ..., \mathbf{N}\}$ represents that feature points corresponding to the same feature in the point set and $\mathbf{p}_i$ represents the position of a feature point in world coordinate system. By simple calculation, we minimize the objective function, and iteratively update and propagate the pose within the sliding window, realizing local optimization of front-end odometry pose.

As shown in Figure 3, we further use factor graph for global graph optimization. Instead of directly removing old frames from the sliding window when new keyframes are added, we incorporate these marginalized frames into the factor graph. Specifically, our global graph optimization incorporates loop closure detection and employs weight noise models.

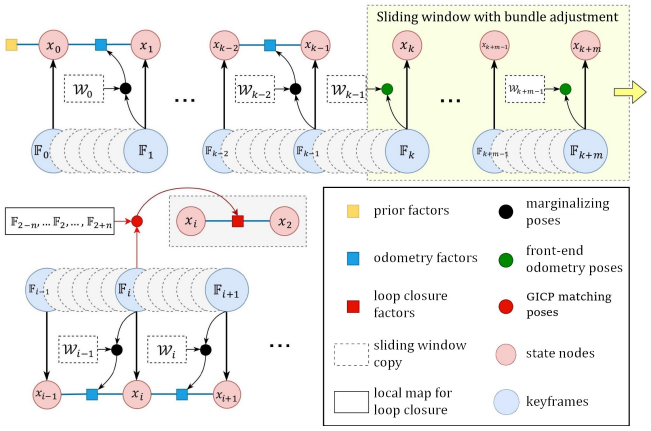


Fig. 3: Construction of the factor graph. Odometry factors record the estimated pose between adjacent keyframes and are incorporated into the factor graph after marginalizing the state.

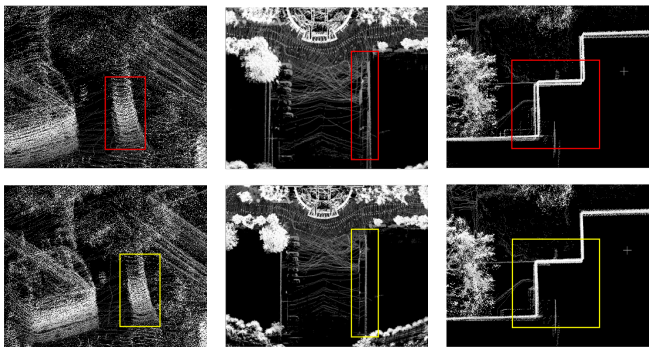


Fig. 4: Enhanced precision and accuracy: a comparison of results before (above) and after (below) applying the noise model.

a) Loop Closure Detection: We implement a simple and effective loop closure detector based on Euclidean distance, which is able to correct the scale drift appearing in SLAM. When a new odometry factor (edge in the factor graph) and a new state node $\mathbf{x}_i$ are added to the factor graph, the first step is to search within the factor graph for the nearest state $\mathbf{x}_l$ that is within a search threshold distance from the state $\mathbf{x}_i$. If a loop is detected, we compute a similarity transformation that informs about the drift accumulated in the loop. To construct a local map $\mathcal{M}_l$ around $\mathbb{F}_l$, we consider 2n neighboring keyframes. These keyframes are transformed into the coordinate system of $\mathbb{F}_l$'s frame, resulting in the transformed local map denoted as $\mathcal{M}_l'$:

$$\mathcal{M}_l = \mathbb{F}_{l-n} \oplus \ldots \oplus \mathbb{F}_l \oplus \ldots \oplus \mathbb{F}_{l+n}, \tag{5}$$

$$\mathcal{M}_l' = \boldsymbol{T}_l^{-1} \cdot \mathcal{M}_l, \tag{6}$$

With the initial transformation described as $\Delta \boldsymbol{T}_{l,i} = \boldsymbol{T}_l^{-1} \boldsymbol{T}_i$, we use the GICP to perform point cloud registration on the original point cloud $\mathcal{P}$ and calculate the loop closure factor $\Delta \boldsymbol{T}_{l,i}^*$, which will be put into the factor graph to trigger a global optimization process:

$$\Delta \boldsymbol{T}_{l,i}^* = \mathrm{GICP}\left(\mathcal{P}_{\mathcal{M}_l'},\ \mathcal{P}_l,\ \Delta \boldsymbol{T}_{l,i}\right). \tag{7}$$

b) Weight Noise Model: To better fit the noise data, we dynamically allocate noise weights to factors based on the certainty between point clouds which can reduce the overall drift error of the system.

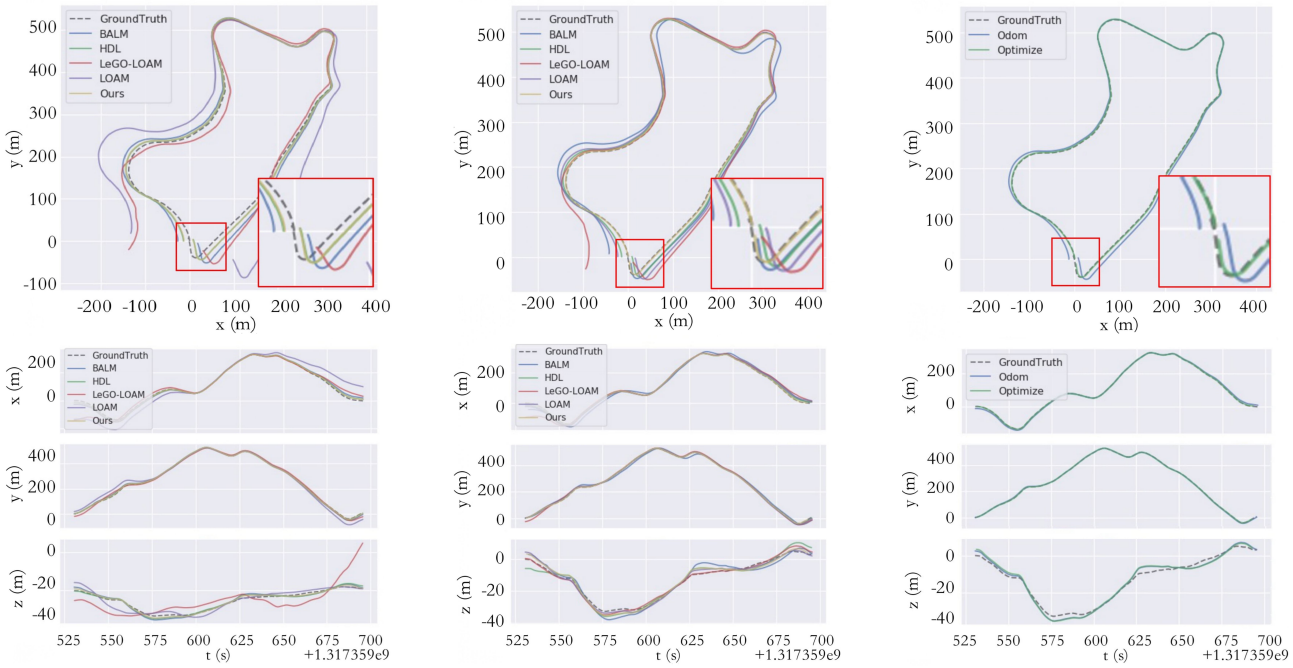
In common noise models, the confidence of measurements is typically assumed to be determined by the constant measurement noise of the sensor. This leads to fixed weights for global factor graph optimization during the optimization of odometry and loop closure. For improvement, we adopt a dynamic method by calculating the information matrix between the point clouds based on their corresponding point sets, which allows us to dynamically calculate the error and update the global factor map based on the weight assigned to each factor during optimization. We further compute the information matrix [10] to capture the correspondence between points in point clouds:

$$\mathrm{Info}\left(\mathcal{P}_i, \mathcal{P}_j, \mathbf{T}_{i,j}\right) = \sum_{(\mathbf{p}_i, \mathbf{q}_j) \in \mathcal{C}_{ij}} \mathbf{G}_{\mathbf{p}}^{\top} \mathbf{G}_{\mathbf{p}}, \tag{8}$$

where $\mathcal{P}_i$ and $\mathcal{P}_j$ represent two point clouds, $\mathbf{T}_{i,j}$ represents the transformation between the point clouds, $\mathbf{p}_i \in \mathcal{P}_i$ and $\mathbf{q}_j \in \mathcal{P}_j$, $\mathcal{C}_{ij}$ represents the set of corresponding points between $\mathcal{P}_i$ and $\mathbf{T}_{i,j} \cdot \mathcal{P}_j$, where the distance between corresponding points is less than the specified threshold. $\mathbf{G}_{\mathbf{p}} = [-[\mathbf{p}]_\times \mid \mathbf{I}]$ and $[\mathbf{p}]_\times$ is the skew-symmetric matrix associated with $\mathbf{p}$. This information matrix is used to model Gaussian noise for odometry and loop closure factors. Higher coefficients in the information matrix indicate more accurate measurements. Different noise models have varying levels of confidence, and during global optimization based on the factor graph, nodes with higher confidence are estimated more accurately. Additionally, a prior noise model, which describes the impact of noise on the system output when a given input is provided, is also taken into account.

Figure 4 showcases partial fragments from our collected datasets. The upper portion of each sub-image displays the point cloud using a preset noise model. The red-boxed areas indicate point cloud ghosting. Conversely, the lower portion of each sub-image represents the point cloud utilizing the information matrix as a noise model. The corresponding yellow-boxed regions demonstrate a relative reduction in point cloud ghosting, thereby enhancing point cloud consistency. The normal noise model exhibits a more substantial impact on features with lower sensitivity (such as turning). Our framework, where the information matrix serves as a weight noise model, dynamically calculates noise for each factor to mitigate the influence of erroneous matches.

To alleviate the computational burden of back-end optimization, we incorporate a marginalization threshold τ, which specifies the accumulation of a certain number of τ keyframes before conducting the optimization. Rather than updating the optimization for each new keyframe, these accumulated keyframes are marginalized together as a batch. By incorporating odometry factors, loop closure factors, and a weight noise model, we construct a straightforward yet



(a) Front-end odometry **(b)** Back-end optimization **(c)** Our method

Fig. 5: Quality performance of pose estimation between different methods. Note that the vertical axis scales in the lower half of the graph are not uniform.

TABLE I: Pose error evaluations.

Metrics	BALM	HDL	LeGO-LOAM	LOAM	Ours
Max	41.49	15.29	97.88	23.33	**3.90**
Mean	20.28	5.71	15.91	8.52	**2.20**
Median	20.18	5.08	6.73	6.05	**2.23**
Min	2.39	0.63	2.01	0.47	0.57
RMSE	21.74	7.11	25.47	11.08	**2.33**
Std	7.84	4.23	19.89	7.09	**0.76**

effective factor graph. The optimized pose derived from this process can be utilized for enhanced trajectories and a refined global map.

IV. EXPERIMENT AND ANALYSIS

A. Setup

We evaluate our method on both public KITTI [11] dataset and real-world deployment. In the experiment, we choose the ninth sequence KITTI09 dataset, which contains 1594 frames collected by HDL-64E LiDAR with a total trajectory length of 1705m, and provides the trajectory ground truth. For real-world deployment, we build two mobile robot platforms mounted single and dual RS-16 LiDAR of Robosense respectively. We utilize absolute pose error (APE) as the error metric of the system, as APE evaluates the global consistency of the system. We further compare the proposed method with four frontier methods, LOAM [2], LeGO-LOAM [3], HDL [7], and BALM [8].

B. Quality Performance

Figure 5 presents the trajectory results of different methods following front-end odometry and back-end optimization. The upper sub-figure displays the 2D trajectory of the x-y plane and the lower sub-figure illustrates the localization error under different directions. It is evident that our method outperforms other methods in both front-end odometry and back-end optimization and achieves similar performance with the ground truth. That benefits from the fusion of multiple LiDAR sensors and the effective optimization techniques, which is also confirmed in Figure 5c. Another observation is that trajectories of all methods become smoother after the back-end optimization. The reason is that compared with the front-end odometry, back-end optimization using filtering or nonlinear optimization obtains optimal pose estimation and globally consistent map. Overall, the results show that our global manifold optimization strategy based on a factor graph with loop closure constraints and a weight noise model improves quality performance effectively.

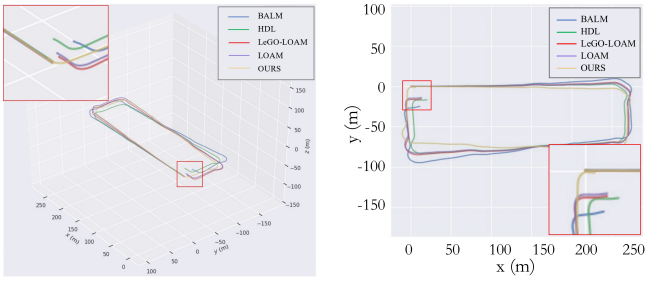
Table I further compares the evaluation results of different methods under different error metrics. From the table, it can be seen that our method consistently outperforms other methods on most evaluation metrics. There is not much difference in our method under those error metrics, which means that the method can obtain more accurate and robust results than others. These results demonstrate the effectiveness of our framework based on multi-LiDAR systems.

C. Real-world deployment

To evaluate the effectiveness of our model in the real world, we conduct a road test. The length of the path for

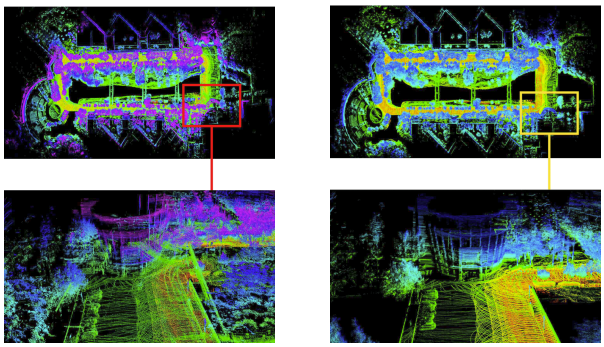
TABLE II: Calibration error evaluations.

Method	Rotation (deg)			Translation (m)			E_R (deg, ↓)	E_t (m, ↓)
	x	y	z	x	y	z		
Hand-eye	-0.195	0.866	-179.653	-0.825	-0.016	0.000	0.629	0.046
Optimization	-0.157	0.509	-179.671	-0.828	-0.017	-0.042	0.492	0.028
GT	0.000	0.000	-179.380	-0.806	-0.035	-0.038	-	-



(a) 3D trajectory **(b)** 2D trajectory of x-y plane

Fig. 6: Trajectory of the car between different methods.



(a) LOAM **(b)** Ours

Fig. 7: The mapping results of different methods.

the road test is 747 m, including turns, driving straight, and lane changing. The experimental results are shown in Figure 6 and Figure 7. From Figure 6, it can be observed that most SLAM methods exhibit cumulative drift. In contrast, our method (indicated by the yellow-colored trajectory) effectively corrects the drift through global optimization. Although LeGO-LOAM and HDL also have loop closure detection modules, the limitations of FoV occlusion and sparsity of single LiDAR result in poor performance. Considering that there is no ground truth in real-world deployment, we also report the 2D trajectory of the x-y plane, as shown in Figure 6b. It can be seen that our method exhibits a relatively consistent trajectory when moving along the x-axis from the starting point, while the other methods display a noticeable angle deviation after covering a certain distance. In Figure 7, we further visualize the quality performance of the mapping. From the figure, we can observe our method achieves smoother and more accurate mapping results.

As shown in Table II, we also evaluate the accuracy of the calibration. As expected, the two-step calibration effectively reduces rotation (0.629° to 0.492°) and translation errors (from $0.046m$ to $0.028m$), which is benefited by local map feature matching effectively reduces the point-to-line and point-to-plane residual constraints. Note that the ground truth is obtained through manual point selection and further refined via multi-LiDAR calibrations.

V. CONCLUSION

This paper presents an efficient odometry and mapping framework with online extrinsic calibration based on multi-LiDAR systems. By combining factor graphs with loop closure constraints and the weight noise model for back-end optimization, our method can achieve accurate pose estimation. The experimental results verify the effectiveness of the proposed method on both the public dataset and real-world deployment. We hope our method can provide a new perspective for LiDAR-based odometry and mapping.

REFERENCES

[1] Chongjian Yuan, Jiarong Lin, Zheng Liu, Hairuo Wei, Xiaoping Hong, and Fu Zhang. Btc: A binary and triangle combined descriptor for 3d place recognition. *IEEE Transactions on Robotics*, 2024.

[2] Ji Zhang and Sanjiv Singh. Loam: LiDAR odometry and mapping in real-time. In *Proceedings of Robotics: Science and Systems*, volume 2, 2014.

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

[4] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[5] Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno. Voxelized gicp for fast and accurate 3d point cloud registration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11054–11059. IEEE, 2021.

[6] HAGC Premachandra, Ran Liu, Chau Yuen, and U-Xuan Tan. Uwb radar slam: an anchorless approach in vision denied indoor environments. *IEEE Robotics and Automation Letters*, 2023.

[7] Kenji Koide, Jun Miura, and Emanuele Menegatti. A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement. *International Journal of Advanced Robotic Systems*, 16, 02 2019.

[8] Zheng Liu and Fu Zhang. Balm: Bundle adjustment for lidar mapping. *IEEE Robotics and Automation Letters*, 6(2):3184–3191, 2021.

[9] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*, pages 105–116. Springer, 2006.

[10] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12786–12796, June 2022.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.