Large-Scale Multi-Session Point-Cloud Map Merging

Hairuo Wei, *Graduate Student Member, IEEE*, Rundong Li, Yixi Cai, *Member, IEEE*, Chongjian Yuan, Yunfan Ren, *Graduate Student Member, IEEE*, Zuhao Zou, Huajie Wu, *Member, IEEE*, Chunran Zheng, *Graduate Student Member, IEEE*, Shunbo Zhou, Kaiwen Xue, and Fu Zhang, *Member, IEEE*

***Abstract*—This paper introduces LAMM, an open-source framework for large-scale multi-session 3D LiDAR point cloud map merging. LAMM can automatically integrate sub-maps from multiple agents carrying LiDARs with different scanning patterns, facilitating place feature extraction, data association, and global optimization in various environments. Our framework incorporates two key novelties that enable robust, accurate, large-scale map merging. The first novelty is a temporal bidirectional filtering mechanism that removes dynamic objects from 3D LiDAR point cloud data. This eliminates the effect of dynamic objects on the 3D map model, providing higher-quality map merging results. The second novelty is a robust and efficient outlier removal algorithm for detected loop closures. This algorithm ensures a high recall rate and a low false alarm rate in position retrieval, significantly reducing outliers in repetitive environments during large-scale merging. We evaluate our framework using various datasets, including KITTI, HeLiPR, WildPlaces, and a self-collected colored point cloud dataset. The results demonstrate that our proposed framework can accurately merge maps captured by different types of LiDARs and data acquisition devices across diverse scenarios.**

***Index Terms*—Mapping, multi-robot SLAM, SLAM.**

I. Introduction

LARGE-SCALE 3D maps are vital in various applications for mobile robots, such as autonomous navigation [1], urban planning [2], and disaster management [3]. These maps provide an in-depth understanding of the environment, which allows autonomous systems to navigate through complex surroundings using efficient paths and informed decisions. State-of-the-art LiDAR mapping techniques, such as Simultaneous Localization and Mapping (SLAM), facilitate the generation of dense 3D point cloud maps [4], [5]. However, the majority of SLAM research focuses on map reconstruction using only a single agent, which presents significant challenges in creating large-scale 3D maps due to limitations in onboard resources and sensing capabilities. The onboard resources, such as limited battery life, memory storage capacity, and computational power, hinder the agent's ability to cover vast areas and handle the large amounts of data required to create detailed maps. Additionally, the finite sensing range of a single agent can result in incomplete or sparse map reconstruction, making it difficult to cover large-scale scenes effectively.

To efficiently reconstruct large-scale scenes, it is necessary to utilize multiple agents simultaneously for data collection. This approach enhances the coverage and efficiency of the mapping process. However, using multiple agents for reconstruction requires precise map merging to integrate the collected data accurately. Map merging allows these agents to work collaboratively, generating large-scale maps that cover areas beyond the reach of a single agent. It also facilitates the integration of information from various perspectives and locations, thereby mitigating errors and inconsistencies that may arise in individual maps.

Current map merging techniques often rely on modeling-based [6], [7], [8], [9], [10] or learning-based [11], [12], [13], [14] approaches to extract features or environmental semantic information directly from LiDAR point clouds, followed by data association and global optimization. Nonetheless, merging a large number of distinct map segments, particularly in complex environments, continues to be a challenging issue [15]. One primary challenge in map merging lies in place feature extraction. As sensors may perceive an environment from different perspectives, the data captured from sensors of the same place can be very different. Additionally, map segments may contain appearance changes under different illumination conditions or structural changes (i.e. dynamic objects in the environment, such as vehicles and pedestrians, alter the appearance of scenes), which will introduce further data association failures. Another challenge in map merging is the robustness of data association, as map merging techniques are sensitive to false positive matches. Even very few incorrect data associations may turn global map optimization into an ill-posed problem [16].

In this study, we propose LAMM, a point cloud map merging framework, which enables robust and precise large-scale map merging from different agents in diverse environments. Specifically, we developed a temporal bidirectional filtering module based on [17] to eliminate dynamic objects from the LiDAR point cloud. Furthermore, we designed an efficient mechanism to filter out false positive matches in detected loop closures to obtain accurate geographical matches between multiple submaps and optimize the global map accordingly. The contributions of our proposed work can be summarized as follows:

- We designed a temporal bidirectional filtering method to remove dynamic objects from 3D LiDAR point cloud data, and employ our recent work Binary Triangle Combined descriptors (BTC) [10] to provide accurate place description thus improving the precision of the final merged results.

Received 25 August 2024; accepted 3 November 2024. Date of publication 21 November 2024; date of current version 28 November 2024. This article was recommended for publication by Associate Editor J. Zhang and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported by the Smart Traffic Fund under Grant PSRI/69/2306/RA. *(Hairuo Wei and Rundong Li are contributed equally to this work.) (Corresponding author: Fu Zhang.)*

Hairuo Wei, Rundong Li, Yixi Cai, Chongjian Yuan, Yunfan Ren, Zuhao Zou, Huajie Wu, Chunran Zheng, and Fu Zhang are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong SAR , China (e-mail: hairuo@connect.hku.hk; yixicai@connect.hku.hk; fuzhang@hku.hk).

Shunbo Zhou and Kaiwen Xue are with the Huawei Cloud Computing Technical Innovation Department, Huawei Cloud Computing Technologies Company Ltd., Gui'an 550025, China.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3504317, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3504317

- We designed an efficient outlier removal mechanism for detected loop closures to reject false positives, significantly reducing outliers and ensuring robust merging in complex, large-scale environments.
- We provide a framework that enables the merging of large-scale 3D LiDAR point cloud maps in various environments. It successfully merges submaps captured at different times and by different devices, automatically establishing loop closures, and performing alignment and global optimization. We have made our code open source and available on https://github.com/hku-mars/LAMM to encourage further research in this area.
- We conducted comprehensive qualitative and quantitative evaluations on various large-scale datasets, including public datasets such as KITTI, HeLiPR, and WildPlaces, as well as our self-collected colored point cloud dataset in Shenzhen. The proposed framework accurately merged maps, demonstrating its high versatility across different LiDAR types, heterogeneous data collection platforms, and various scenarios. The corresponding video is available at https://youtu.be/X2WSILJe-Ew.

II. Related Works

A. Map Merging

Map merging involves reorganizing unordered sub-maps into a single, global, and consistent map. LiDAR sensors have been widely applied in large-scale map merging tasks due to their high accuracy and long detection range. Early map merging algorithms utilized 2D LiDAR [18] to build occupancy grid maps and perform map merging based on spectral information. However, simple 2D occupancy maps do not meet the requirements for complex 3D navigation tasks, leading to increasing attention to 3D map merging [15]. SegMap [19] can extract semantic information from 3D point clouds and provide street block-like global map merging, but its data association heavily relies on the segmentation of distinguishable semantic objects, which is difficult to achieve in city-scale or campus-scale map merging tasks. AutoMerge [20] offers a framework capable of merging 3D segments in city-scale environments without requiring initial coordinate estimations, but it struggles in environments with numerous dynamic objects.

The success of large-scale 3D LiDAR map merging heavily depends on robust feature extraction from point clouds and accurate data association between different segments, which are challenging to ensure, especially for large-scale map merging. In this section, we focus on the LiDAR map merging task, reviewing key techniques for 3D feature extraction and large-scale data association.

Creating robust place descriptors for 3D LiDAR point clouds is a significant challenge in large-scale environments. LiDAR SLAM systems typically extract place descriptors based on the structural features of point clouds. Methods using model-based 3D point cloud descriptors, such as ScanContext [8], [9], leverage structural features to evaluate point cloud similarities for data association but are sensitive to large translation differences [21]. There are also methods employing learning-based descriptors. AutoMerge [20] generates viewpoint-invariant descriptors through neural networks. SegMap [19] construct place descriptors that rely on semantic object segmentation. As these methods heavily rely on training data, their performance may not generalize well to new or unseen data.

In all the above methods, dynamic objects are rarely addressed, yet their elimination is crucial. Dynamic objects can alter the structure and appearance of scenes, affecting place feature extraction, leading to errors in data association, and negatively impacting subsequent tasks such as navigation.

Another challenge in large-scale mapping is the presence of similar scenes, which can result in false positive loop constraints during the establishment of correspondences between different submaps. Common methods like RANSAC [20], [22] and PCM [23], [24] typically use outlier filtering by setting thresholds based on the maximum consistency of transformations derived from all loop closures.

In our work, we introduce a temporal bidirectional filtering module based on M-Detector [17], which effectively removes dynamic points from the 3D point cloud, resulting in a clean, static map for subsequent tasks. We then utilize BTC [25] as a descriptor for place features. BTC efficiently extracts both global and local features from 3D point clouds, achieving full pose invariance and high performance across diverse environments. Its efficiency in both extraction and search operations makes BTC particularly suitable for data association in large-scale environments. We also propose a RANSAC-based method that transforms loop closures between sub-maps back to the starting point, offering a more accurate and robust mechanism for rejecting false positives.

B. Dynamic Object Removal

Nowadays, dynamic object removal algorithms for LiDAR point cloud maps mainly utilize dynamic object detection to provide dynamic/static information for each point, and then delete the dynamic points. In dynamic points detection, traditional algorithms offer greater flexibility and robustness compared to learning-based methods, which often require labeled datasets and network training.

Traditional algorithms are typically classified into ray-casting, visibility-based, and visibility-free methods. OctoMap [26] uses ray-casting to update occupancy grids from sensor hits and misses but can be computationally demanding. Kim et al. [27] introduced a static point cloud map using multi-resolution range images based on visibility. However, it faces challenges when detecting distant ground points for sparse LiDAR queries. Lim et al. [28] tackled these issues by utilizing height differences between the raw map and the query for detection. However, this method requires careful tuning of maximum and minimum height ranges, limiting its effectiveness in certain scenarios. Newer approaches, like the BeautyMap algorithm [29], enhance traditional methods by utilizing binary-encoded matrices for dynamic point detection, reducing latency and improving adaptability across different environments. M-Detector [17], a recent technique, leverages point-by-point event detection using LiDAR occlusion principles, offering microsecond-level latency and generalizing well across different datasets.

III. Methodology

A. Problem Formulation

The task of map merging is to combine multiple trajectory sequences generated by different robots or mapping sessions

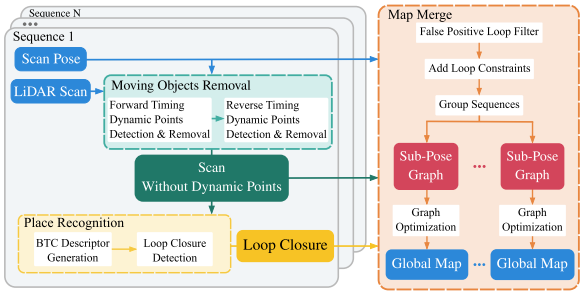

Fig. 1. The input to LAMM consists of LiDAR scans and their initial poses from sequences awaiting merging. These initial scan poses are in the respective frames of each sequence, typically obtained from a front-end SLAM system such as FAST-LIO2 [30], and may include drift. Once all data is loaded, LAMM constructs the sub-pose graph for the associated sequences, merges them into a consistent frame and optimizes all poses to build a global map. Finally, the framework outputs the global maps.

into a consistent global map representing the environment. Each trajectory sequence consists of multiple frames of LiDAR point clouds combined with odometry data, estimated using front-end SLAM algorithms such as FAST-LIO2 [30] for each frame. We define the list of trajectory sequences awaiting merging as $S_N = \{s_1, s_2, \ldots, s_n\}$, where each sequence s_i starts from a different position and has no prior knowledge of the relative pose to others. Using the map merging algorithm, the relative poses between any two overlapping sequences are determined, and together with optimized odometry, a globally consistent and accurate map is generated.

B. System Overview

As illustrated in Fig. 1, the LAMM framework offers an automated map merging solution for large-scale mapping tasks involving single or multiple agents. This framework takes trajectory sequences from several agents carrying LiDARs as input and merges them into several globally consistent maps. The LAMM framework is organized into three key modules: in-sequence moving object removal, inter- and inner-sequence place recognition, and global map merging.

The input to the framework is S_N. For each sequence $s_i \in S_N$, the moving objects removal module loads registered scans (LiDAR point clouds) obtained from front-end SLAM algorithms (e.g., FAST-LIO2 [30]) and employs a bidirectional filtering mechanism to eliminate moving objects, resulting in scans containing points from the static environment. The place recognition module receives these cleaned scans and applies the BTC algorithm to extract place descriptors for this sequence. It then conducts loop closure detection among all sequences and provides loop closure information for S_N to the map merge module. False positive loop closures are filtered out during the map merging stage, and valid loop constraints are retained. Note that the sequences may not be fully connected, meaning they form several independent places. To address this, we check the connectivity of sequences in S_N and divide them into M sub-pose graphs based on filtered loop closure information. Each sub-pose graph is optimized separately to obtain M consistent global maps.

C. Moving Objects Removal

We developed a moving object removal method based on M-Detector [17]. M-detector operates on a point-by-point basis, making decisions immediately after each LiDAR point is captured. It transforms each frame of the point cloud into a depth map and uses the accumulated depth maps to make occlusion tests. M-Detector uses three types of occlusion tests to determine whether a point is moving: (1) detecting if a point occludes previously observed points, (2) detecting recursive occlusion where a moving object occludes itself, and (3) detecting movement along the LiDAR beam. M-detector identifies points passing these tests as dynamic in real-time without needing to accumulate entire frames, providing a low-latency solution. However, since the accumulation of depth images is time-dependent, it can lead to missed detections at the start of the detection process and in certain specific cases.

We considered an intrinsic characteristic: if an object is dynamic, it should remain dynamic in both forward and reverse temporal sequences, as its motion is inherent to the time series. For example, a moving car's position shifts across multiple timestamps in the forward sequence. When the sequence is reversed, the same positional changes occur in reverse order, confirming its dynamic nature. This means that both forward and reverse temporal sequences can be used to detect dynamic points. Since our framework is designed for offline processing, we propose a bidirectional detection method that detects dynamic objects in both forward and reverse temporal sequences. This enhances the detection of dynamic objects, thereby improving the extraction of place features. It is noted that since the number of static points is significantly greater than that of dynamic points, mistakenly identifying some static points as dynamic does not affect our place feature extraction.

To illustrate the effectiveness of the bidirectional filtering mechanism, we firstly explain the working logic of M-Detector. M-Detector employs three types of tests. The first test (case 1) detects dynamic points of objects moving perpendicular to the LiDAR laser rays. In this case, the object must occlude background objects observed previously. The second test (case 2) detects points of objects moving away from the LiDAR in parallel to the sensor laser rays, where the objects must be repeatedly occluded by themselves. The third test (case 3) detects points of objects moving towards the LiDAR in parallel to the sensor laser rays, where the objects must repeatedly occlude themselves.

In both cases 2 and 3, repeated occlusion requires time accumulation, so the initial scans containing moving objects cannot effectively detect moving points. To fully leverage all available point cloud data, we design a bidirectional filtering mechanism that utilizes both forward and reverse timestamps. This method assesses occlusion relationships from both temporal directions and performs dual filtering to remove dynamic points effectively. When reversing the timestamp, the initial scans become the last scans, allowing the detection of moving points in these scans. Besides, as shown in Fig. 2, dynamic points that originally only triggered case 3 can now trigger case 1, as they occlude previously observed background objects. This approach ensures a more thorough detection and removal of moving objects in the input point clouds, enhancing the accuracy of our dynamic object detection.

D. Place Recognition

In our study, BTC [10] is employed as a descriptor of place feature and conduct loop closure detection based on it. The comprehensive workflow of BTC in our framework is depicted in Fig. 1.

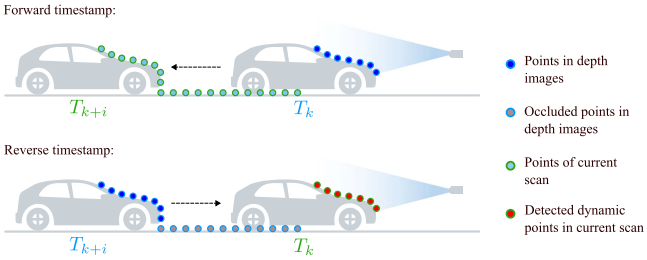

Fig. 2. How bidirectional filtering mechanism help to detect more moving points. In the detection of forward timestamp, points of the car cannot be detected as they did not pass any occlusion tests. However, in the detection of reverse timestamp, points of the car occlude the points of the background and pass the occlusion test of case 1, thus they can be treated as dynamic points.

1) Binary Triangle Combined Descriptor: BTC demonstrates greater adaptability and significant improvements in precision compared to its counterparts, especially in challenging scenarios with large viewpoint variations (e.g., reverse direction, large translation, and/or rotation) [10], which is crucial for large-scale map merging tasks. It is a novel descriptor that combines global and local features.

2) Loop Closure Detection: To ensure comprehensive loop closure detection for the input scan across all sequences, each sequence loaded into the system maintains its own descriptor database, numbered according to the sequence's load order (i.e., the database for the first loaded sequence is numbered 1, the second is numbered 2, and so on). Loop detection is sequentially processed across all existing databases, including its own, in descending order to obtain both inner-sequence and inter-sequence loop closure results. The loop detection process consists of three main steps: rough loop detection, fine loop detection, and geometric verification. During rough detection, potential loop closure candidates are quickly identified by matching BTC descriptors in the database using a hash table. Fine detection verifies these candidates by computing transformation matrices between triangle pairs and clustering them to determine the most supported transformations. The final geometric verification step calculates the point-to-plane distances (using fewer than 50 key points from the source submap) for these candidates, selecting the candidate with the minimal distance as the valid loop closure.

E. Map Merging

To perform map merging in large-scale environments, we employ pose graph optimization. However, the presence of multiple similar scenes can compromise loop detection and lead to inaccurate data associations such as potential mismatches among non-overlapping sequences. To address these challenges, we introduce a false loop filtering mechanism alongside a strategy to classify trajectories based on their geographical overlap, allowing for the correct optimization of each distinct group.

1) False Positive Loop Filter: For inner-sequence loop closures, we calculate the geographical difference between the matching frames based on the initial pose and filter out loop closures with significant geographical discrepancies to reject false positives. For inter-sequence loop closures, we gather all detected loops between every two sequences and apply a Random Sample Consensus (RANSAC) based method to perform outlier rejection.

Consider two trajectories s_i and s_j, each with its own frame. The start position of s_j in the frame of s_i is defined as $\mathbf{j}^{\mathcal{I}} \in \mathbb{R}^3$.

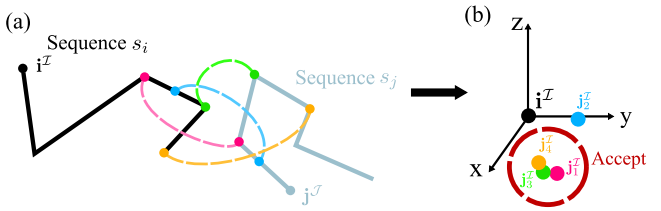

Fig. 3. A visual example of the proposed outlier rejection method between two sequences s_i and s_j. Suppose four loop closures are detected between s_i and s_j. Among these, only the second(colored blue) loop closure is a false positive, while the others(colored red, yellow and green respectively) are true positives. We transform the start position of s_j into the frame $\mathcal{I}$ of s_i according to the four different loop closures, obtaining the points $j_1^{\mathcal{I}}$, $j_2^{\mathcal{I}}$, $j_3^{\mathcal{I}}$, and $j_4^{\mathcal{I}}$. It is evident that $j_2^{\mathcal{I}}$ is significantly distant from the other points in the frame of sequence s_i, making it an easy candidate for rejection.

Using the detected loop closures, we can project the start position of s_j into the frame of s_i. Consequently, we obtain multiple start positions of sequence s_j in the frame of sequence s_i, each calculated from an individual loop closure. These positions are represented by three-dimensional points $\mathbf{j}_k^{\mathcal{I}} \in \mathbb{R}^3$, where k denotes the point calculated from the k-th loop closure transformation $\mathbf{T}_k \in SE3$ provided by the place recognition module. The scan pose of match scans in their own frame for k-th loop closure can also be written as $\mathbf{T}_k^i \in SE3$ and $\mathbf{T}_k^j \in SE3$. Thus we can calculate $\mathbf{j}_k^{\mathcal{I}}$ as follows:

$$\mathbf{T}_k^i = (\mathbf{R}_k^i, \mathbf{t}_k^i)$$

$$\mathbf{T}_k^j = (\mathbf{R}_k^j, \mathbf{t}_k^j)$$

$$\mathbf{T}_k = (\mathbf{R}_k, \mathbf{t}_k)$$

$$\mathbf{j}_k^{\mathcal{I}} = -\mathbf{R}_k^j * \mathbf{R}_k{}^T * \mathbf{R}_k^i{}^T * \mathbf{t}_k^i - \mathbf{R}_k^j * \mathbf{R}_k{}^T * \mathbf{t}_k + \mathbf{t}_k^i \quad (1)$$

For true positive loop closures, the transformed points $\mathbf{j}_k^{\mathcal{I}}$ should be close to each other in Euclidean distance, while outliers are typically far away due to incorrect transformations. We then apply RANSAC clustering to these points to eliminate outliers corresponding to erroneous loop closures, retaining only the loop closure results associated with inliers.

2) Connectivity Check: The sequences input into the system do not necessarily have spatial overlap. Directly optimizing all sequences in a single pose graph without distinction only adds unnecessary computational complexity. A more efficient approach is to classify the sequences into multiple groups with internal overlap but no overlap between different groups, and then optimize the pose graph for each group separately. Fig. 4 illustrates the pose graphs for each group, which we refer to as sub-pose graphs.

3) Pose Graph Optimization: The pose graph optimization aims to align multiple sequences in the same sub-pose graph into a global consistent map. The optimization problem is formulated as:

$$\min_{\mathbf{X}} \left(\|\mathbf{x}_0 \ominus \mathbf{x}_0^*\|_{\Sigma_0}^2 + \sum_{(i,j)\in\mathcal{E}} \left\|\mathbf{z}_{ij} \ominus \left(\mathbf{x}_i^{-1} \oplus \mathbf{x}_j\right)\right\|_{\Sigma_{ij}}^2 \right) \quad (2)$$

where $\mathbf{X}$ denotes the set of all poses from multiple sequences in the same sub-pose graph, with $\mathbf{x}_i$ representing the i-th pose in this sub-pose graph. The anchor node $\mathbf{x}_0$ corresponds to the first node of the sub-pose graph, and it is constrained by the

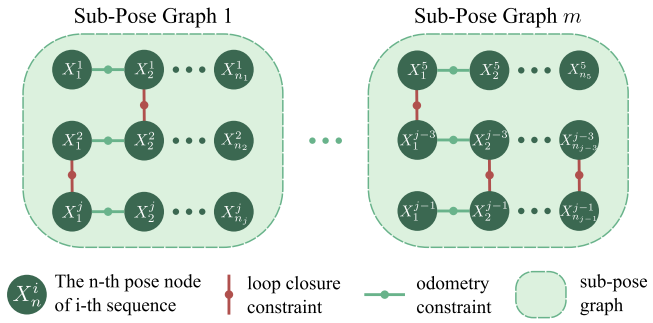

Fig. 4. The structure of sub-pose graphs: Based on the filtered loop closure constraints among the j sequences, we divide the sequences into different groups. In each group, loop closures are detected between sequences, whereas no loop closures exist between sequences in different groups. For each group, we construct a sub-pose graph, resulting in a total of m sub-pose graphs for these j sequences.

TABLE I
THE OVERVIEW OF THE DATASETS UTILIZED IN EVALUATION

Dataset	Environment	Trajectory Length (km)	LiDAR
HeLiPR	Residential area	50	Avia & Ouster
KITTI	Urban	15	Velodyne
WildPlaces	Wild	33	Velodyne
Shenzhen	Urban	2	Hesai

term $\|\mathbf{x}_0 \ominus \mathbf{x}_0^*\|^2_{\Sigma_0}$, where $\mathbf{x}_0^*$ is a known reference pose, often set to the origin, to prevent the entire graph from drifting. The set of relative pose constraints $\mathcal{E}$ includes both intra-trajectory and inter-trajectory constraints. For any connected nodes i and j, $\mathbf{z}_{ij}$ is the measured relative pose between them, with Σ_{ij} representing the associated covariance matrix that captures the measurement uncertainty. The operator $\ominus$ computes the $SE3$ difference between two poses, while $\oplus$ performs pose composition in $SE3$. The optimization seeks to adjust the set of all pose variables $\mathbf{X}$ to minimize the total residuals, ensuring both local consistency within each sequence and global alignment across different sequences. We apply standard back-end pose graph optimization using GTSAM [31] for each sub-graph to adjusts all poses to achieve a global consistent map.

IV. EXPERIMENTS

In this section, we evaluate the accuracy and robustness of our map merging method using public datasets (KITTI [32], HeLiPR [33], and WildPlaces [34]) and a self-collected dataset from Shenzhen, China. We compare our method with the multi-map merging method described in BTC [10] and conduct ablation studies to verify the effectiveness of each module in our algorithm. All algorithms are implemented in C++, and all experiments are conducted on a desktop computer with an Intel i9-13900K @ 3.0 GHz processor and 128 GB of memory.

A. Benchmark Comparison

1) Map Merging: To evaluate the efficacy of our framework, we selected six sequences from the well-known KITTI [32] dataset, as shown in Table II. These sequences represent urban scenarios with several overlapping regions, making them ideal for generating multi-session data. Similar to previous studies [24], we divided each sequence into multiple sessions, ensuring that all sessions overlap with at least one other session. We use FAST-LIO2 [30] as the initial odometry for our method. Since there is no open-source method specifically designed for 3D LiDAR point cloud map merging, we compare our framework with multi-SLAM system Disco-SLAM [35] and DCL-SLAM [36]. We also compare our framework with the map merging method based solely on loop detection proposed in BTC [10]. This approach simply leverages BTC descriptors to align multiple sequences collected at different times. Similar to the place recognition module in our method, it uses BTC descriptors to identify overlapping scans between sequences, providing both intra- and inter-sequence constraints for constructing a global pose graph. After incorporating all detected loop closure constraints, the pose graph is optimized using GTSAM [31], resulting in a globally consistent map.

TABLE II
RMSE OF THE ATE(m) ON KITTI DATASET

Sequence	Origin	FAST -LIO2	Disco -SLAM	DCL -SLAM	BTC	LAMM full model	LAMM wo M-detector	LAMM wo loop filter	LAMM -SOLiD	LAMM -PCM
00	1.407 1.654 2.676	3.330	2.097	4.024	Fail	**2.040**	2.392	Fail	2.061	2.707
02	1.717 0.879 4.060	8.749	Fail	Fail	Fail	**4.746**	6.723	Fail	5.931	5.748
05	0.662 0.476 0.911	1.765	1.653	2.783	1.875	1.639	1.788	1.639	1.658	**1.590**
06	0.503 0.895	**0.938**	0.973	0.984	1.051	0.941	0.948	0.941	0.939	0.983
07	0.416 0.386	0.956	14.877	1.466	0.574	**0.561**	0.577	0.561	12.913	1.752
08	2.260 1.154 1.950	4.255	4.255	5.735	4.468	**4.152**	4.247	4.644	4.299	5.588

The experiment results are shown in Table II and Fig. 5. Our framework successfully merged all sequences with higher accuracy (i.e., lower ATE) compared to other methods. Compared to FAST-LIO2's results after a single run, our method demonstrated similar or even greater accuracy, highlighting its robustness and precision. Disco-SLAM and DCL-SLAM failed to merge sequence 02, likely due to the small overlap of this sequence, which demands more precise loop closure detection to achieve the merging task. Additionally, as BTC lacks a false positive rejection module, it failed to merge sequence 00 and 02, because incorrect loop closure constraints were added to the pose graph, causing the program to crash unexpectedly.

2) Place Recognition: In this section, we verify the superiority of BTC descriptors in place recognition in comparison with other methods. We selected Scan Context [8] used by Disco-SLAM, LiDAR IRIS [37] used by DCL-SLAM, as well as two newly designed descriptors, SOLiD [38] and RING++ [39], for comparison. We use KITTI sequences 00, 02, 05, 06, 07, and 08 to evaluate their performance in place recognition. In the experiment, we determine the loop closure as true positive if the ground-truth pose distance between the query and the predicted loop submap is below a certain threshold (e.g., 5 meters in our experiment).

The precision-recall curves of all 5 methods on the 6 sequences are plotted in Fig. 6. We also include Recall@1, F1 max score and AUC [39] as evaluation metrics in Table III. As shown, BTC achieves superior performance in most sequences for Recall@1, with the exception of KITTI07 and KITTI08, where SOLiD performs slightly better. For both the F1 max score and AUC, BTC consistently outperforms the other four methods across all sequences.

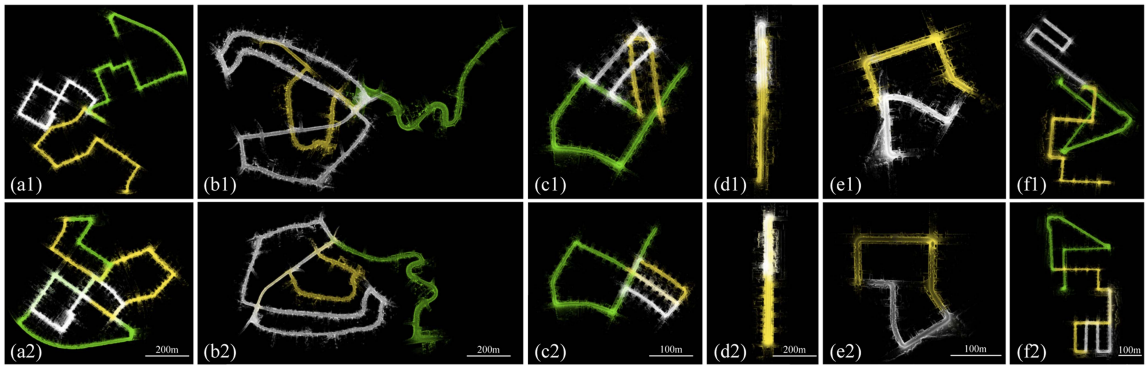

Fig. 5. **Merging results of LAMM on KITTI dataset.** (a1)-(f1) show the original submaps of KITTI00, 02, 05, 06, 07, and 08, while (a2)-(f2) present their merged global maps, with each sequence represented in different colors.

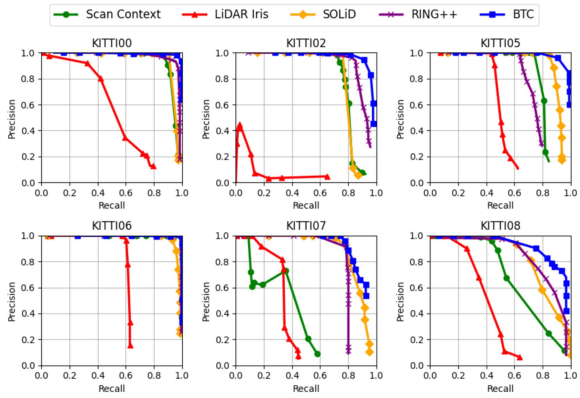

Fig. 6. The precision-recall curves of different methods on KITTI dataset.

3) Dynamics Removal: To show the effectiveness of our bidirectional M-Detector method, we compare it with Removert [27], ERASOR [28], BeautyMap [29] and M-Detector [17] in the Town 01 sequence of HeLiPR dataset, which is populated with many dynamic elements, including pedestrians and vehicles. We followed the benchmark method in [40] for a quantitative comparison. It is noted that similar to ERASOR [28], we manually selected 141 consecutive frames from the Town 01 sequence that feature a significant number of dynamic objects to quantitatively evaluate the algorithms.

The process of dynamic point removal requires maintaining high recall in classifying both dynamic and static points, evaluated by Dynamic Accuracy (DA%) and Static Accuracy (SA%), respectively. Additionally, we use Associated Accuracy (AA%) and Harmonic Accuracy (HA%) in [40] and [29] to provide a comprehensive evaluation of both static and dynamic accuracy in one metric, offering a holistic assessment of the algorithm's performance. The results are shown in Table IV. Although Bi-M-Detector does not outperform in static or dynamic accuracy, it achieves the most balanced classification of static and dynamic points.

B. Ablation Study

To further verify the effectiveness of our framework, we conducted ablation studies on two key components: the false positive loop filter and moving object removal. The ATE results of our framework without the loop filter demonstrate that its absence deteriorates odometry accuracy and can even cause task failure (sequences 00 and 02), as false positive loop closure constraints make the pose graph ill-posed. This confirms that the false positive loop filter module enhances both the accuracy and robustness of our framework. Similarly, the ATE results of our framework without the M-detector, shown in Table II, indicate that the absence of this module results in lower accuracy. Thus, moving object removal increases the accuracy of our framework.

We also conducted additional experiments by replacing both the descriptor and the loop closure verification methods. Based on the place recognition comparison results in Table III, we selected SOLiD [38] descriptor to replace BTC. Additionally, we

TABLE III
EVALUATION OF DESCRIPTORS ON KITTI DATASET

Sequence	Approach	Recall@1	F1 max score	AUC
KITTI00	BTC	**0.9963**	**0.9743**	**0.9935**
	Scan context	0.9764	0.9201	0.9461
	LiDAR IRIS	0.7935	0.5511	0.5682
	SOLiD	0.9714	0.9480	0.9528
	RING++	0.9851	0.9446	0.9748
KITTI02	BTC	**0.9773**	**0.9292**	**0.9730**
	Scan context	0.9223	0.8308	0.8070
	LiDAR IRIS	0.6505	0.1435	0.0756
	SOLiD	0.8706	0.8640	0.8119
	RING++	0.9579	0.8934	0.9208
KITTI05	BTC	**0.9901**	**0.9336**	**0.9852**
	Scan context	0.8449	0.8506	0.8322
	LiDAR IRIS	0.6243	0.6095	0.5546
	SOLiD	0.9384	0.9104	0.9248
	RING++	0.7952	0.7772	0.8014
KITTI06	BTC	**1.0000**	**0.9945**	**0.9980**
	Scan context	0.9963	0.9907	0.9963
	LiDAR IRIS	0.6296	0.7426	0.6773
	SOLiD	0.9889	0.9509	0.9805
	RING++	0.9963	0.9926	0.9966
KITTI07	BTC	0.9256	**0.8584**	**0.9296**
	Scan context	0.5868	0.4778	0.3828
	LiDAR IRIS	0.4463	0.4706	0.3656
	SOLiD	**0.9504**	0.8468	0.8811
	RING++	0.8017	0.8444	0.8078
KITTI08	BTC	0.9683	**0.8291**	**0.9227**
	Scan context	0.9714	0.6227	0.6856
	LiDAR IRIS	0.6381	0.4589	0.4241
	SOLiD	**1.0000**	0.7601	0.8360
	RING++	0.9683	0.7567	0.8478

TABLE IV
EVALUATION OF THE MOVING OBJECTS REMOVAL METHODS

Methods	SA[%]	DA[%]	AA[%]	HA[%]
ERASOR	59.3693	83.842	70.5524	69.5147
Removert	**99.2293**	10.4161	32.1493	18.8531
BeautyMap	44.3985	**88.884**	62.8197	59.2173
M-Detector	96.0476	28.4789	52.3004	43.9317
Bi-M-Detector	92.8589	70.0526	**80.6536**	**79.8594**

TABLE V
RMSE OF THE ATE(m) ON HELIPR DATASET

Environment	Sequences	FAST-LIO2	Merged	Merged Multi-LiDAR
Town	Ouster 1	2.884	2.818	4.008
	Ouster 2	2.563		
	Ouster 3	2.896		
	Avia 1	7.477	4.273	
	Avia 2	14.413		
	Avia 3	21.199		
Roundabout	Ouster 1	1.573	2.125	2.101
	Ouster 2	1.627		
	Ouster 3	2.237		
	Avia 1	9.336	2.247	
	Avia 2	10.134		
	Avia 3	8.328		

replaced our RANSAC-based loop closure verification method with the widely used PCM [23]. The ATE verification results on the KITTI dataset are presented in Table II. While LAMM-SOLiD slightly outperformed LAMM on KITTI06, the difference was only 2 mm, which is negligible. LAMM-SOLiD generally performs at the same or lower level compared to LAMM. LAMM-PCM showed minor improvements over LAMM on KITTI05 but performed significantly worse on other sequences. Therefore, we conclude that LAMM performs better than both LAMM-SOLiD and LAMM-PCM in general.

C. Further Evaluation

To cover a variety of scenarios, we further test our framework on datasets of different environments. Detailed characteristics of each dataset are provided in Table I.

1) HeLiPR: HeLiPR is over 50 km, which is a large-scale dataset that covers diverse environments, from urban cityscapes to high-dynamic freeways, over a month. HeLiPR is the first heterogeneous LiDAR dataset, providing different LiDARs' (Spinning and Solid State) data of the same sequences for each environment.

In our experiment, we conduct our merging framework on sequences of Livox Avia and OS2-128 in two large-scale environment, Roundabout and Town, respectively. The Roundabout sequences feature three roundabouts with a prominently large roundabout paired with an external hexagon. The Town sequences showcase a juxtaposition of tight alleyways and expansive boulevards with multiple dynamic elements, including pedestrians and vehicles. We use FAST-LIO2 to obtain the odometry of each sequence and perform the merging of sequences of the same LiDAR type as well as the merging of sequences of different LiDAR types.

The merging result of HeLiPR in Table V, demonstrates the robustness of our method. In both the Roundabout and Town scenes, the results include the merging of three sequences collected by Ouster LiDAR, the merging of three sequences collected by Avia LiDAR, the merging of all the six sequences. For each combination, we achieve a globally consistent large map with odometry accuracy comparable to the sequences before merging. Due to page limit, the point cloud merging result of HeLiPR is put in the Supplementary Material [41].

2) WildPlaces & Shenzhen Dataset: We also test our framework on the WildPlaces dataset and the self-collected Shenzhen dataset. WildPlaces is a large-scale dataset over 33 km in unstructured, natural environments, which is especially challenging for place recognition and map merging tasks. Shenzhen is self-collected in an urban area near Shenzhen North Railway Station in Shenzhen, China, which is gathered using a backpack device equipped with a Hesai 128-line LiDAR and four Hikvision cameras. After employing the R^{3}LIVE [42] algorithm for front-end odometry, the 6DoF pose data and colored point cloud can be obtained as inputs for our algorithm. Due to page limit, we put the implementation details and result analysis of these experiments in the Supplementary Material [41].

V. CONCLUSION

In this paper, we propose a large-scale map merging method for multi-agent mapping tasks. Our method is capable of merging maps from multiple agents equipped with different types of LiDARs into a global consistent map in various scenarios. We introduce a moving objects detection and removal module to filter out dynamic points in the input point cloud, enhancing the accuracy of place recognition. We also propose a false positive loop filter module to reject false positive loop closures, ensuring the accuracy of loop detection and optimization. We conduct extensive experiments on public datasets and a self-collected dataset to validate the effectiveness of our method. The results show that our method outperforms the state-of-the-art map merging method in terms of accuracy and robustness. In the future, we plan to further improve the efficiency of our method and extend it to more complex scenarios.

REFERENCES

[1] M. Whitty, S. Cossell, K. S. Dang, J. Guivant, and J. Katupitiya, "Autonomous navigation using a real-time 3D point cloud," in *Proc. 2010 Australas. Conf. Robot. Automat.*, 2010, pp. 1–3.
[2] F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin, "Applications of 3D city models: State of the art review," *ISPRS Int. J. Geo- Inf.*, vol. 4, no. 4, pp. 2842–2889, 2015.
[3] S. Nikoohemat, A. A. Diakité, S. Zlatanova, and G. Vosselman, "Indoor 3D reconstruction from point clouds for optimal routing in complex buildings to support disaster management," *Autom. Construction*, vol. 113, 2020, Art. no. 103109.
[4] J. Zhang et al., "LOAM: LiDAR odometry and mapping in real-time," *Robot.: Sci. Syst.*, vol. 2, no. 9, pp. 1–9, 2014.
[5] 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.
[6] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *Proc. 2011 IEEE Int. Conf. Robot. Biomimetics*, 2011, pp. 2987–2992.
[7] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proc. 11th Eur. Conf. Comput. Vis.*, Heraklion, Crete, Greece, 2010, pp. 356–369.
[8] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Proc. 2018 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 4802–4809.
[9] G. Kim, S. Choi, and A. Kim, "Scan context: Structural place recognition robust to rotation and lateral variations in urban environments," *IEEE Trans. Robot.*, vol. 38, no. 3, pp. 1856–1874, Jun. 2022.

[10] C. Yuan, J. Lin, Z. Liu, H. Wei, X. Hong, and F. Zhang, "BTC: A binary and triangle combined descriptor for 3D place recognition," *IEEE Trans. Robot.*, vol. 40, pp. 1580–1599, 2024.
[11] M. A. Uy and G. H. Lee, "PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4470–4479.
[12] Z. Liu et al., "LPD-net: 3D point cloud learning for large-scale place recognition and environment analysis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 2831–2840.
[13] J. Komorowski, "MinkLoc3D: Point cloud based large-scale place recognition," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2021, pp. 1790–1799.
[14] Z. Fan, Z. Song, H. Liu, Z. Lu, J. He, and X. Du, "SVT-net: Super light-weight sparse voxel transformer for large scale place recognition," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 551–560.
[15] S. Yu, C. Fu, A. K. Gostar, and M. Hu, "A review on map-merging methods for typical map types in multiple-ground-robot SLAM solutions," *Sensors*, vol. 20, no. 23, p. 6988, 2020.
[16] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, Winter 2010.
[17] H. Wu, Y. Li, W. Xu, F. Kong, and F. Zhang, "Moving event detection from LiDAR point streams," *Nat. Commun.*, vol. 15, no. 1, 2024, Art. no. 345.
[18] S. Carpin, "Fast and accurate map merging for multi-robot systems," *Auton. Robots*, vol. 25, pp. 305–316, 2008.
[19] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3D segment mapping using data-driven descriptors," in *Proc. Robotics: Sci. Syst. XIV*, 2018.
[20] P. Yin et al., "AutoMerge: A framework for map assembling and smoothing in city-scale environments," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3686–3704, Oct. 2023.
[21] P. Yin et al., "Stabilize an unsupervised feature learning for LiDAR-based place recognition," in *Proc. 2018 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1162–1167.
[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[23] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *Proc. 2018 IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2916–2923.
[24] Z. Yu, Z. Qiao, L. Qiu, H. Yin, and S. Shen, "Multi-session, localization-oriented and lightweight LiDAR mapping using semantic lines and planes," in *Proc. 2023 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 7210–7217.
[25] C. Yuan, J. Lin, Z. Liu, H. Wei, X. Hong, and F. Zhang, "BTC: A binary and triangle combined descriptor for 3-D place recognition," *IEEE Trans. Robot.*, vol. 40, pp. 1580–1599, 2024.
[26] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.
[27] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *Proc. 2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10758–10765.
[28] 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. Automat. Lett.*, vol. 6, no. 2, pp. 2272–2279, Apr. 2021.
[29] M. Jia, Q. Zhang, B. Yang, J. Wu, M. Liu, and P. Jensfelt, "BeautyMap: Binary-encoded adaptable ground matrix for dynamic points removal in global maps," *IEEE Robot. Automat. Lett.*, vol. 9, no. 7, pp. 6256–6263, Jul. 2024.
[30] 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.
[31] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Inst. Technol., Atlanta, GA, Tech. Rep. 2, 2012.
[32] 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.
[33] M. Jung, W. Yang, D. Lee, H. Gil, G. Kim, and A. Kim, "HeLiPR: Heterogeneous LiDAR dataset for inter-LiDAR place recognition under spatiotemporal variations," *Int. J. Robot. Res.*, vol. 43, pp. 1867–1883, 2023.
[34] J. Knights, K. Vidanapathirana, M. Ramezani, S. Sridharan, C. Fookes, and P. Moghadam, "Wild-places: A large-scale dataset for LiDAR place recognition in unstructured natural environments," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.* 2023, pp. 11322–11328.
[35] Y. Huang, T. Shan, F. Chen, and B. Englot, "DiSCo-SLAM: Distributed scan context-enabled multi-robot LiDAR SLAM with two-stage global-local graph optimization," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1150–1157, Apr. 2022.
[36] S. Zhong, Y. Qi, Z. Chen, J. Wu, H. Chen, and M. Liu, "DCL-SLAM: A distributed collaborative LiDAR SLAM framework for a robotic swarm," *IEEE Sensors J.*, vol. 24, no. 4, pp. 4786–4797, Feb. 2024.
[37] Y. Wang, Z. Sun, C.-Z. Xu, S. E. Sarma, J. Yang, and H. Kong, "LiDAR IRIS for loop-closure detection," in *Proc. 2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.* 2020, pp. 5769–5775.
[38] H. Kim, J. Choi, T. Sim, G. Kim, and Y. Cho, "Narrowing your FOV with solid: Spatially organized and lightweight global descriptor for FOV-constrained LiDAR place recognition," *IEEE Robot. Automat. Lett.*, vol. 9, no. 11, pp. 9645–9652, Nov. 2024.
[39] X. Xu et al., "RING: Roto-translation-invariant gram for global localization on a sparse scan map," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4616–4635, Dec. 2023.
[40] Q. Zhang, D. Duberg, R. Geng, M. Jia, L. Wang, and P. Jensfelt, "A dynamic points removal benchmark in point cloud maps," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst.*, 2023, pp. 608–614.
[41] H. Wei et al., "LAMM: Supplementary materials," (n.d). [Online]. Available: https://github.com/hku-mars/LAMM/blob/main/LAMM_supplementary.pdf
[42] J. Lin and F. Zhang, "R3 live: A robust, real-time, RGB-colored, LiDAR-inertial-visual tightly-coupled state estimation and mapping package," in *Proc. 2022 IEEE Int. Conf. Robot. Automat.*, 2022, pp. 10672–10678.