2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

CDC-SLAM: Switchable Centralized and Distributed Collaborative LiDAR SLAM Framework for Robotic Swarms

Xiangnan Liu[1], Xiang Huo[1], Haifei Zhu[1], Xuefeng Zhou[2], Yisheng Guan[1], Hong Zhang[3], Weinan Chen[1*]

***Abstract*— In GPS-denied environments, robotic swarms must concurrently accomplish collaborative tasks and autonomous localization, a process that remains highly challenging. Existing mainstream collaborative frameworks are generally categorized into centralized and distributed (or decentralized) approaches: centralized methods often exhibit limited robustness, while distributed or decentralized methods typically encounter accuracy constraints. The intrinsic limitations of these two framework types render a single architecture inadequate for effectively addressing complex and diverse real-world scenarios. To address this challenge, we introduce CDC-SLAM, an adaptive centralized–distributed collaborative SLAM framework that supports flexible dual-architecture switching, thereby enhancing scenario adaptability and enabling efficient LiDAR-inertial collaborative state estimation. The system adaptively selects optimization strategies according to real-time inter-robot distances: when the swarm is relatively close, a central node performs global centralized optimization and disseminates the results; when dispersed, the system switches to distributed optimization, exchanging only essential data to alleviate the computational burden on individual robots.Meanwhile, back-end data sharing and outlier rejection mechanisms preserve the consistency of the global map during switching. Extensive evaluations on public datasets demonstrate that the proposed CDC-SLAM system delivers improved localization accuracy and mapping performance.**

IndexTerms—Distributed, Centralized, Multi-robot SLAM.

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) serves as a core technology for real-time perception of unknown environments, pose estimation, and map construction. Single-robot SLAM has been extensively studied for decades, with representative systems such as VINS-Mono (monocular visual-inertial) [1], LIO-SAM (LiDAR-inertial) [2], ORB-SLAM3 (multimodal) [3], and Fast-LIO2 [4], all enabling efficient exploration. However, its low efficiency and limited coverage in large-scale environments remain significant challenges. Consequently, multi-robot SLAM, which achieves collaborative mapping through swarm-level information sharing, has emerged as a critical research direction.

This work was supported in part by the National Natural Science Foundation of China under Grant 62103179, 52475008, in part by the Guangzhou Basic and Applied Basic Research Foundation under Grant 2024A04J4070, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515140044 and Grant 2025A1515010194, and in part by the Guangdong Association for Science and Technology Youth Talent Cultivation Program under Grant SKXRC2025078.

*corresponding author (chenwn@gdut.edu.cn).

[1]Biomimetic and Intelligent Robotics Lab, School of Electromechanical Engineering, Guangdong University of Technology, Guangzhou, China.

[2]Guangdong-Hong Kong-Macao Greater Bay Area National Technology Innovation Center.

[3]Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China.

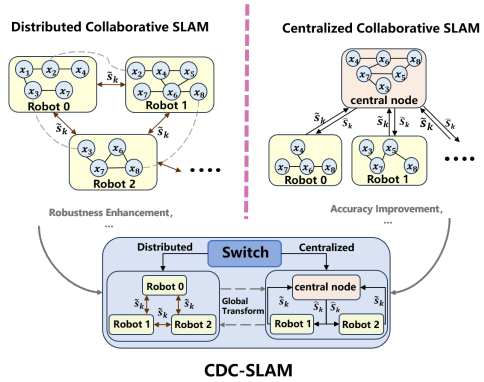

Fig. 1: Overview of our CDC-SLAM framework, integrating the distributed architecture (left) and the centralized architecture (right) to achieve reliable and high-precision mapping and localization.

Currently, multi-robot SLAM has emerged as a key approach for large-scale environmental exploration and mapping. Mainstream collaborative architectures are generally categorized into three types: centralized [5], distributed [6][7][8], and decentralized [9]. Although these frameworks provide fundamental support for collaborative operations, each exhibits clear limitations. Centralized architectures depend on a single central node to process all data, imposing stringent requirements on network stability and risking system failure if the node malfunctions, thereby lacking robustness. In contrast, distributed and decentralized architectures reduce network dependence but suffer from insufficient inter-node information exchange, difficulty in ensuring global consistency, and complex algorithmic implementation—making it challenging to balance mapping accuracy with real-time performance.

To address the shortcomings of existing architectures, this paper introduces an adaptive switching centralized–distributed fusion framework for multi-robot collaborative SLAM, which mitigates the aforementioned issues through switchable optimization modes. When the distance between a robot and the designated central node becomes large (i.e., beyond communication range), the system transitions to distributed optimization, exchanging only keyframe information among nodes within communication reach to effectively reduce both computational and communication overhead. When the distance falls below the threshold, centralized optimization is triggered: the central node aggregates data from all robots, exploits global information to enable

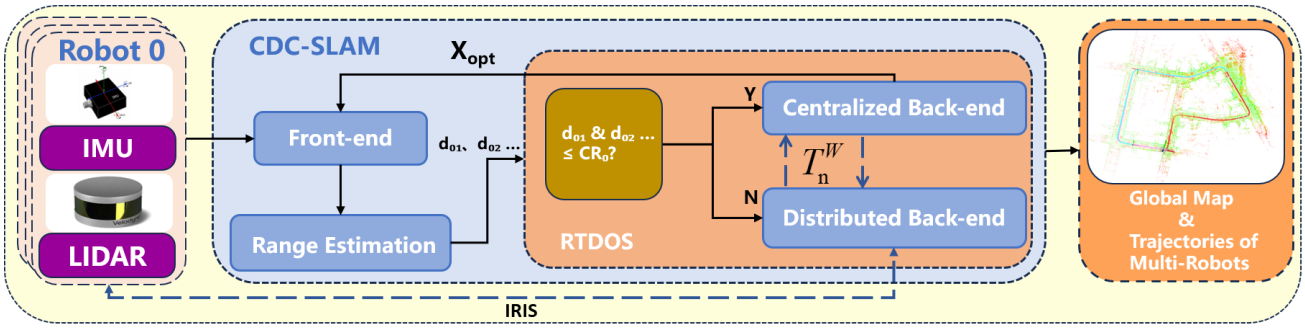

Fig. 2: Architecture of the CDC-SLAM system. The back-end selection is performed via a distance-triggered mechanism, considering d_{01} (distance from Robot 1 to the central node) and CR_0 (communication range of the central node). The centralized back-end uses X_{opt} (optimized keyframe pose) to correct the front-end.

high-precision loop closure detection and pose estimation, and ultimately constructs a globally consistent map.

To further improve the system's practicality and reliability, this paper presents a pose fusion approach based on dynamically scaled correction factors, which combines optimized poses with front-end poses to produce refined poses. These fused poses not only remain close to the ground truth but also prevent pose discontinuities. Relative factors computed from consecutive fused poses are then fed back to the front-end to enable real-time pose correction.

The proposed framework has been validated on the processed public KITTI dataset and the multi-robot collaborative dataset S3E [10]. The main contributions are as follows:

- A collaborative LiDAR SLAM framework that adaptively switches between distributed and centralized collaboration.
- A pose fusion method based on dynamically scaled correction factors, which generates refined poses and feeds relative factors back to the front-end for correction.
- Extensive evaluations on public datasets, covering both accuracy and scalability, to demonstrate the system's effectiveness. The source code and processed datasets are released for public access.

II. RELATED WORKS

A. Architectures of Multi-Robot SLAM

Currently, the development of multi-robot SLAM focuses on balancing global consistency, resource overhead, and scenario adaptability, giving rise to four mainstream collaborative architectures: centralized, distributed, decentralized, and hybrid. The centralized architecture relies on a central node for global control, where slave robots transmit complete data, such as LiDAR point clouds, to the central node for pose estimation and map maintenance. Techniques such as BA [11][12] and graph optimization [13][14] achieve high accuracy in small-to-medium-scale scenarios [15]. For example, CCM-log-LAM [16] demonstrates a robust centralized collaborative SLAM system. However, this architecture faces scalability challenges: data transmission bandwidth increases linearly with the swarm size, and a failure of the central node can lead to complete system collapse [17][18]. In distributed architectures, each robot maintains local computational capabilities and exchanges key constraints over limited communication channels. DDF-SAM [19] first introduced this concept using constrained factor graphs, while subsequent approaches, such as the two-stage distributed Gauss-Seidel method [20], further optimized aspects including privacy. Door-SLAM and DiSCo-SLAM leverage DPGO or DFGO to mitigate the single-point failure inherent in centralized systems, yet ensuring global consistency remains challenging [21]. Tian et al.'s distributed certifiably correct pose-graph optimization [22] achieved notable breakthroughs, although it remains highly sensitive to initial values, posing a risk of converging to local optima under poor initialization.

The hybrid architecture, a key research focus, seeks to combine the strengths of both centralized and distributed paradigms to enhance scenario adaptability. For instance, D2SLAM [23] modulates collaboration based on the scenario, yet it lacks deep integration mechanisms such as SE(d)-synchronization-based global calibration [24], while still ensuring consistency when robots operate in isolation. Our framework dynamically switches between modes according to the distance between robots and the central node—employing centralized operation for near-field precision and distributed operation for far-field overhead reduction—thereby addressing the rigidity limitations of existing architectures in adapting to diverse scenarios.

Nevertheless, despite enhanced adaptability, all three architectures share a common bottleneck: front-end pose estimates remain susceptible to disturbances, and accumulated errors can compromise map consistency. Consequently, pose correction strategies aimed at mitigating accumulated errors have been extensively investigated.

B. Pose Correction

Although backend optimization in SLAM can theoretically provide high-precision pose estimates, the absence of effective feedback to the frontend limits the exploitation of these results and exacerbates the poor adaptability of existing systems in complex or rigid scenarios. For instance, VINS-Mono employs sliding-window BA for backend optimiza-

tion to correct frontend drift, while LOAM balances high-frequency frontend odometry with low-frequency backend map optimization [25]; yet even these representative approaches fail to bridge the frontend-backend feedback gap. In fact, fully harnessing high-precision optimization results has become critical for accurate hybrid collaboration, rendering pose correction a central focus of recent research.

Mainstream backend optimization schemes exhibit inherent limitations: global BA requires offline backtracking to replace frontend poses, often causing trajectory jumps that contradict smooth pose transitions on the special Euclidean manifold [26]; CCM-log-SLAM replaces poses in fixed-interval batches, resulting in trajectory discontinuities during aggressive robot motion; ASAPP [27] supports dynamic updates but merely superimposes optimization increments without a fusion mechanism, leading to trajectory fluctuations and long-trajectory accuracy degradation in the absence of loop closure (unresolved by current manifold optimization-based backends). Within distributed architectures, correction depends on inter-node constraint propagation: Door-SLAM's adjustments rely on neighbor graph connectivity, causing local deviations to propagate to the frontend under limited communication, which distributed Gauss-Seidel methods partially mitigate but cannot fully resolve in terms of privacy and efficiency; DiSCo-SLAM does not filter constraint outliers, where erroneous constraints trigger multi-node frontend pose offsets, and even certifiably correct distributed pose-graph optimization cannot compensate for this; Isolated robot subgroups cause intra-subgroup correction deviations, which DDF-SAM, a constrained factor graph based fully distributed SLAM system, does not address, and resolving these deviations requires substantial computational resources for reconnection alignment.

Hybrid architectures, such as D2SLAM, aim to combine correction benefits but lack centralized global calibration, still depending on distributed constraint propagation—thus, consistency cannot be ensured when robots operate in isolation, even under certifiably correct distributed optimization. Current correction methods exhibit three main limitations: absence of a motion consistency model (violating the smoothness of manifold optimization), limited scenario robustness, and reliance on a single architectural paradigm.

To address these challenges, this paper introduces a pose-adaptive fusion correction mechanism. By using the difference between the backend-optimized pose and the frontend real-time pose as the primary metric, the method dynamically adjusts the frontend's conversion weight toward the high-precision reference: small differences increase the weight for rapid alignment, while large differences decrease it to prevent overcorrection-induced jitter or trajectory rupture. This difference-driven adaptive adjustment adheres to the smoothness principles of manifold optimization and enables coordinated enhancement of frontend accuracy, trajectory continuity, and overall system robustness.

III. METHODOLOGY

The system structure is shown in Fig. 2: each robot is assigned a unique ID. The frontend processes LiDAR/IMU data, extracts keyframes and feature points, performs loop closure detection, and converts information into lightweight, rotation-invariant descriptors for backend transmission to save bandwidth and ensure real-time performance. The backend uses the proposed RTDOS (Range-Triggered Dual-Optimization Switching) to select centralized or distributed optimization: centralized mode feeds back optimized global poses to correct the frontend; distributed mode only exchanges in-range data, filters noise, and selects optimal loop closures.

A. Optimization methods

1) Centralized and Distributed Optimization Architectures: The following narrative unfolds with a scenario involving three robots. In a 3-robot scenario: when Robots 1 and 2 are within Robot 0's communication range, RTDOS issues a Centralized Trigger Command (Cent Trigger Cmd) to activate the centralized framework (Fig. 3(a)). Strong short-range signals prevent node overload. The frontend sends keyframes to the backend, which fuses optimized and raw keyframes to correct frontend poses. For error suppression: the first 5 frames use a global transformation; subsequent poses rely on adjacent relative poses (Robot 1 example:)

$$\mathbf{x}_{k1}^{W} = T_1^{W} \cdot \mathbf{x}_{k1}^{L}, \tag{1}$$

$$\mathbf{x}_{k1}^{W} = \mathbf{x}_{k1}^{W} \cdot \left(T_1^{W}\right)^{-1} \cdot T_1^{L}\left(k-1 \to k\right), \tag{2}$$

Where $\mathbf{x}_{k1}^{W}$ is the global pose of Robot 1 at the k-th frame, T_1^{W} is its global initial transformation, and $T_1^{L}(k-1 \to k)$ is the relative transformation between adjacent frames in the local coordinate system. The centralized architecture adopts factor graph constraints, with the prior factor as follows:

$$f_{\text{prior}}\left(\mathbf{x}_{0i}\right) = \left\| \mathbf{x}_{0i} \ominus \mathbf{x}_{0i}^{\text{prior}} \right\|_{\Sigma_{\text{prior}}}^{2}, \tag{3}$$

f denotes a factor. $\mathbf{x}_{0i}^{\text{prior}}$ is robot i's prior initial pose; $\ominus$ represents the subtraction operation for poses (SE(3) group differences express error transformation); $\|\cdot\|_{\Sigma_{\text{prior}}}^{2}$ is squared Mahalanobis distance. In the centralized framework, only Robot 0 is the prior origin, others get prior initial poses via loop-closure detection. Optimized poses approximate the prior values, thus preventing uncontrolled drift of the global trajectory.

$$\begin{aligned} f_{\text{odom}}\left(\mathbf{x}_{(k-1)i}, \mathbf{x}_{ki}\right) = \Big\| \mathbf{x}_{ki} \ominus \big(\mathbf{x}_{(k-1)i} \\ \oplus \Delta\mathbf{x}_{i}^{\text{odom}}\left(k-1 \to k\right)\big) \Big\|_{\Sigma_{\text{odom}}}^{2}, \end{aligned} \tag{4}$$

$\Delta\mathbf{x}_{i}^{\text{odom}}(k-1 \to k)$ is radar odometer-measured relative pose; $\oplus$ represents the pose addition operation to constrain local motion continuity.

$$f_{\text{loop}}\left(\mathbf{x}_{ki}, \mathbf{x}_{lj}\right) = \left\| \mathbf{x}_{lj} \ominus \left(\mathbf{x}_{ki} \oplus \Delta\mathbf{x}_{(ki)\to(lj)}^{\text{loop}}\right) \right\|_{\Sigma_{\text{loop}}}^{2}, \tag{5}$$

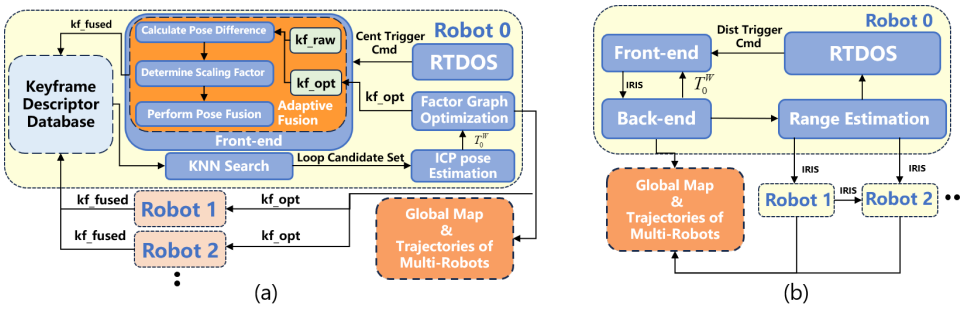

Fig. 3: (a) Centralized multi-robot collaborative SLAM architecture: Robots use front-end adaptive fusion for accurate and stable pose estimation, combined with KNN search and the keyframe descriptor database, and complete mapping collaboratively through factor graph optimization. (b) Distributed multi-robot collaborative SLAM architecture: Each robot builds the global map while coordinating with others via relevant modules to ensure consistency. Both architectures share the distance-triggered dual-optimization switching mechanism, which governs the transition between centralized and distributed modes to maintain adaptability and performance.

In the formula: $\mathbf{x}_{ki}$ (robot i, k-th frame), $\mathbf{x}_{lj}$ (robot j, l-th frame), and $\Delta\mathbf{x}^{\text{loop}}_{(ki)\to(lj)}$ (their relative pose). This factor ensures multi-robot global consistency, corrects long-term errors and aligns poses via loop closure. From the constraints of prior, odometry and loop factors, centralized optimization builds the following nonlinear least squares problem:

$$\begin{aligned}\hat{\mathbf{x}}_{ki} = \arg\min_{\mathbf{x}_{ki}} \Bigg(& \sum_{\text{prior}} \left\| f_{\text{prior}}\left(\mathbf{x}_{0i}\right)\right\|^2_{\Sigma_{\text{prior}}} \\ & + \sum_{\text{odom}} \left\| f_{\text{odom}}\left(\mathbf{x}_{(k-1)i}, \mathbf{x}_{ki}\right)\right\|^2_{\Sigma_{\text{odom}}} \\ & + \sum_{\text{loop}} \left\| f_{\text{loop}}\left(\mathbf{x}_{ki}, \mathbf{x}_{lj}\right)\right\|^2_{\Sigma_{\text{loop}}} \Bigg),\end{aligned} \tag{6}$$

This problem is efficiently solved using incremental factor graph optimization, a Bayesian tree data structure, and an iterative update scheme, ultimately yielding optimized poses.

Our distributed architecture is consistent with the centralized framework, using the same global descriptor for keyframe transmission. To coordinate with centralized optimization data, it employs factor graph optimization—allowing centralized data to provide good initial values for distributed optimization, and vice versa. The distributed framework adopts factor graph optimization, with its nonlinear least squares problem summarized as:

$$\begin{aligned}\hat{\mathbf{x}}_{ki} = \arg\min_{\mathbf{x}_{ki}} \Bigg(& \sum_{\text{prior}} \left\| f_{\text{prior}}\left(\mathbf{x}_{0i}\right)\right\|^2_{\Sigma_{\text{prior}}} \\ & + \sum_{\text{odom}} \left\| f_{\text{odom}}\left(\mathbf{x}_{(k-1)i}, \mathbf{x}_{ki}\right)\right\|^2_{\Sigma_{\text{odom}}} \\ & + \sum_{\text{loop}_{\text{intra}}} \left\| f_{\text{loop}_{\text{intra}}}\left(\mathbf{x}_{ki}, \mathbf{x}_{ji}\right)\right\|^2_{\Sigma_{\text{loop}_{\text{intra}}}} \\ & + \sum_{\text{loop}_{\text{inter}}} \left\| f_{\text{loop}_{\text{inter}}}\left(\mathbf{x}_{ki}, \mathbf{x}_{lj}\right)\right\|^2_{\Sigma_{\text{loop}_{\text{inter}}}} \Bigg),\end{aligned} \tag{7}$$

Among them, $f_{\text{loop}_{\text{intra}}}$ is the intra-robot loop-closure factor and $f_{\text{loop}_{\text{inter}}}$ the inter-robot one. For communication-constrained scenarios, a range limit is imposed: keyframe descriptors are transmitted only when robots and their interaction targets are within range. As illustrated in the Range Estimation module (Fig. 4(b)), two distance checks are performed: one for mode switching (the system switches to distributed mode if distance exceeds the limit, sending mode signals to the frontend and keyframes to centralized optimization); the other for communication constraints (keyframe descriptors are exchanged only when within communication range).

2) Range-Triggered Dual-Optimization Switching Mechanism: In summary, the centralized architecture's global optimization and the distributed architecture's communication adaptability jointly form the foundation for handling complex scenarios. As shown in Fig. 3, the shared Range-Triggered Dual-Optimization Switching Mechanism is central to collaborative scheduling. Combined with Fig. 4, the logic is as follows: optimization mode is determined by inter-robot distance—centralized optimization is used within communication range with continuous updates to the global initial transformation; if a robot is out of range, the system switches to distributed mode, initializing the backend with the centralized global transformation.To avoid stability degradation from frequent switching, set a post-switch delay threshold to mask new triggers. When returning to centralized mode, the updated global transformation is sent, local poses are unified, and then loop closure detection and point cloud registration are performed, enabling mutual optimization across modes rather than a simple switch.

As shown in Fig. 4, centralized mode employs a separate factor graph to maintain variables and produce optimized poses and global transformations for each robot. The distributed mode not only uses this transformation to improve efficiency but also feeds back its own to the centralized mode, which integrates it into the global coordinate system for subsequent processing.

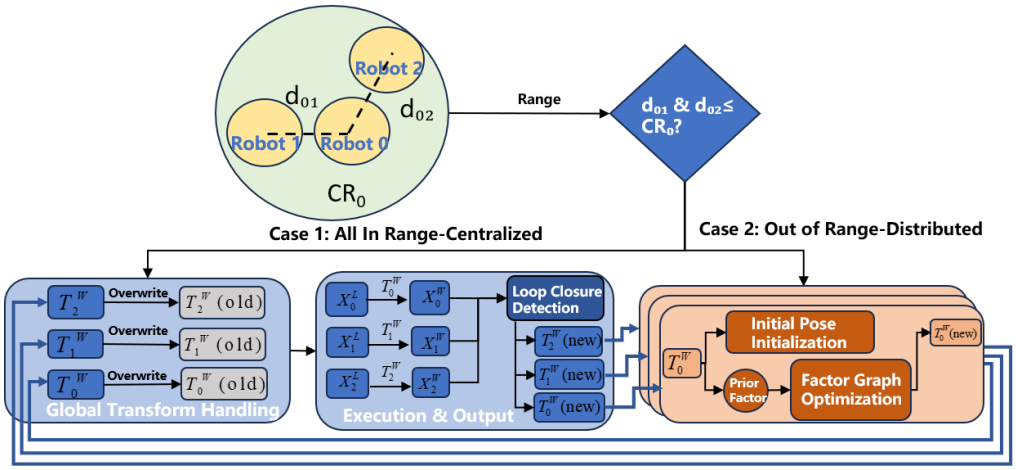

Fig. 4: Range-Triggered Dual-Optimization Switching: Centralized optimization is used when all robots are within the central node's range; otherwise, distributed optimization is applied. The two modes exchange global transformations — centralized optimization supplies prior factors to distributed optimization for faster loop closure and convergence, while distributed optimization provides initial poses to the centralized back-end to improve overall performance.

B. Robot's local pose correction

In SLAM systems, feedback from the backend to the frontend is crucial for accuracy, and multi-robot SLAM relies on it to reduce errors. However, direct strong constraints in some systems can lead to over-constraint, so existing studies [28][29] introduce Dynamic Adjustment Factors to mitigate this issue.

Motivated by single-robot backend correction logic, which enhances single-robot accuracy, this paper proposes a pose fusion method using dynamic adjustment factors to correct frontend poses via relative factors. The first-step formula of the method is given as follows:

$$\text{diff}_{\text{pose}}(k) = \left\| \left(\tilde{R}_k^L\right)^T \left(\hat{x_k^L} - \tilde{x_k^L}\right) \right\|_2, \tag{8}$$

$$\text{diff}_{\text{rot}}(k) = \left\| \text{rpy}\left(\left(\tilde{R_k^L}\right)^T \cdot \hat{R_k^L}\right) \right\|_2, \tag{9}$$

Here, we use R to represent the rotation matrix of the pose, and $\boldsymbol{x} = [x\ y\ z]^T$ to denote the position vector. $\tilde{R}$ and $\tilde{\boldsymbol{x}}$ represent the pose rotation matrix and position vector that have not been optimized by the back-end, while $\hat{R}$ and $\hat{\boldsymbol{x}}$ represent the rotation matrix and position vector optimized by the centralized back-end.$\tilde{x_k^L}$ and $\tilde{R_k^L}$ represent the local position vector and rotation matrix of the k-th frame. By feeding back the pose optimized by the centralized method to the front-end, we calculate the pose difference between two frames with the corresponding pose of the front-end.

$$\text{posescale}_k = \begin{cases} \dfrac{pos_{max}}{\text{diff}_{\text{pos}}(k)}, & \text{if } \text{diff}_{\text{pos}}(k) > pos_{max} \\ 1, & \text{otherwise} \end{cases}, \tag{10}$$

$$\text{rotscale}_k = \begin{cases} \dfrac{rot_{max}}{\text{diff}_{\text{rot}}(k)}, & \text{if } \text{diff}_{\text{rot}}(k) > rot_{max} \\ 1, & \text{otherwise} \end{cases}, \tag{11}$$

In the above equations, pos_{max} and rot_{max} are predefined thresholds, adjusted according to the strength of environmental changes, where stronger variations lead to smaller thresholds to suppress excessive corrections. By comparing the calculated differences with these thresholds, posescale_k and rotscale_k represent the magnitudes of the translation scaling factor and rotation scaling factor for the k-th frame. λ_k is the finally determined scaling factor, which selects the minimum of the two factors as the scaling ratio. This helps control the smoothness of pose fusion and prevents pose jumps.

$$\theta = \arccos\left(\text{Re}\left(\tilde{q_k^L} \circ \hat{\tilde{q}_k^L}\right)\right), \tag{12}$$

$$\bar{x_k^L} = \tilde{x_k^L} + \lambda_k \cdot \left(\hat{x_k^L} - \tilde{x_k^L}\right), \tag{13}$$

$$\bar{q_k^L} = \left(\tilde{q_k^L} \cdot \frac{\sin\left[1 - \lambda_k \cdot \theta\right]}{\sin\theta}\right) + \hat{q_k^L} \cdot \frac{\sin\left[\lambda_k \cdot \theta\right]}{\sin\theta}, \tag{14}$$

$\bar{q_k^L}$ and $\bar{x_k^L}$ represent the local position and quaternion of the k-th frame after fusion. Among them, $\tilde{q_k^L}$ and $\hat{q_k^L}$ are the unoptimized and optimized pose quaternions converted from rotation matrices. θ is the spherical angle between the unoptimized quaternion and the optimized quaternion. $\hat{\tilde{q}_k^L}$ denotes the conjugate quaternion of $\tilde{q_k^L}$, where $Re(\cdot)$ represents the operation of taking the real part of a quaternion, and $\circ$ stands for the quaternion multiplication. The final fused position is obtained by adding the unoptimized position to the product of the scaling ratio and the position difference. For the pose, a new fused pose is generated through spherical linear interpolation (slerp) of quaternions. Compared with the local pose corresponding to the front-end, this fused pose is closer to the optimized pose, thus solving issues such as IMU reset caused by excessively large pose jumps between consecutive frames. As shown in Fig. 5, the frontend fuses multiple measurement data within the factor graph optimization framework, including the LiDAR odometry factor,

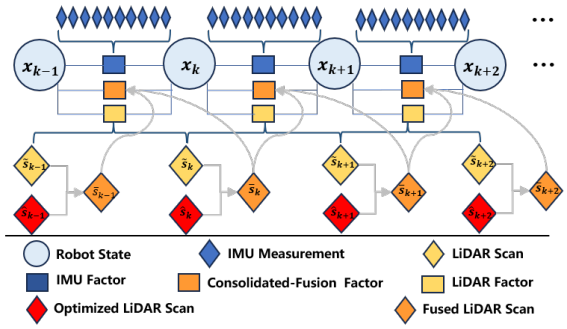

Fig. 5: The Consolidated-Fusion Factor is formed by fusing the optimized LiDAR scan with the corresponding front-end frame and then integrating it with IMU and LiDAR factors. It optimizes subsequent robot states and provides initial poses for distributed and centralized SLAM.

the IMU factor, and the Consolidated-Fusion factor. This is formulated as a nonlinear least squares (NLS) problem:

$$\mathbf{x} = \underset{\mathbf{x}}{\operatorname{argmin}} \Big(\sum_{\text{lidar}} \| f_{\text{lidar}} \|^2 + \sum_{\text{imu}} \| f_{\text{imu}} \|^2 + \sum_{\text{cf}} \| f_{\text{cf}} \|^2 \Big). \tag{15}$$

Here, "lidar" and "imu" denote all LiDAR and IMU odometry, respectively. The relative transformation is computed from the fused poses of the previous and next frames and inserted as a new relative factor into the corresponding frontend graph node. It is jointly optimized with the original frontend LiDAR odometry, IMU, and loop closure factors to determine the next frame pose, thus correcting the frontend using the backend-optimized pose and enhancing system accuracy. In the experiments, factor graph optimization is employed to realize multi-source factor fusion.

IV. EXPERIMENTS

This chapter evaluates the performance and accuracy of the proposed CDC-SLAM system across multiple scenarios, with comparisons to other state-of-the-art SLAM methods. All experiments are performed on a workstation with an Intel® Xeon E5-2620 v4 CPU, an NVIDIA TITAN Xp GPU, and 125.8 GiB of RAM. The experimental datasets include processed KITTI sequences and the public S3E dataset.

A. Implementation

The proposed CDC-SLAM integrates dual frameworks and multiple libraries, with ROS1 as middleware, and is implemented in C++ on Ubuntu 20.04 LTS. Its frontend adopts LIO-SAM, a mature LiDAR-based open-source front-end, and leverages the ICP [30] algorithm along with k-nearest neighbor search provided by the libnabo library. DiSCo-SLAM is used as the baseline for the distributed system; it supports multi-node scheduling, offers good compatibility and scalability, and ensures configuration consistency between distributed and centralized optimization. Within DiSCo-SLAM, IRIS [31] replaces Scan Context [32] for efficient loop detection, and PCM is employed to filter out erroneous loop closures, improving overall mapping accuracy.

B. Experimental Design

We selected KITTI sequences 05, 06, 07, and 09, cropping each into three trajectories with overlapping start and end regions, and time-shifting them for simultaneous playback. These sequences were chosen to generate diverse loop trajectories with ground truth for full system evaluation: Seq. 5 contains overlapping straight segments, Seq. 9 overlapping curves, and Seqs. 6–7 full loops. Additionally, various trajectories from the S3E dataset, including concentric circles, intersecting curves, and rays, were used to validate the system's adaptability and robustness across motion patterns. Dataset details are provided in Table I.

TABLE I: Details of S3E and KITTI Datasets

	Datasets	Sensor	Trajectory Length [m]			Size [GB]
			α	β	γ	
S3E	*Library_1*	LVI	507.6	517.2	498.9	17.5
	Dormitory_1	LVI	727.0	719.3	721.9	25.3
	Playground_1	LVI	407.7	425.6	445.4	9.4
	Playground_2	LVI	265.6	315.7	456.4	6.7
	Square_1	LVI	546.6	496.5	529.2	10.1
	Square_2	LVI	304.6	250.5	246.4	10.1
	Tunnel_1	LVI	522.0	502.4	501.1	8.2
KITTI	*Seq. 5*	LI	487.7	493.0	1070.8	8.8
	Seq. 6	LI	385.9	472.0	766.1	4.5
	Seq. 7	LI	300.2	328.0	426.3	4.3
	Seq. 9	LI	547.7	645.5	973.4	5.9

LVI indicates LiDAR-Visual-Inertial and LI indicates LiDAR-Inertial.

For evaluation, we employ Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) as metrics. The performance of our proposed system is compared against state-of-the-art SLAM methods, including LIO-SAM, DiSCo-SLAM, and two variants of our system: those with and without local pose correction.

C. Evaluation on S3E Dataset

In this section, we evaluate our system on the S3E outdoor dataset using two key metrics: Absolute Trajectory Error (ATE) and Relative Pose Error (RPE), where RPE measures local consistency. As shown in Table II, our system achieves notable improvements across multiple scenarios (Dormitory_1, Playground_1, Tunnel_1), maintaining high trajectory accuracy even when other methods fail in critical initialization or real-time tracking.

Table II reports the ATE between the estimated trajectories and ground truth. Our system (Ours (corrected)) achieves lower errors than other methods across various sequences, highlighting its advantage in global trajectory accuracy. Combined with the RPE results in Table III, it also exhibits superior local rotation and pose consistency, maintaining stable tracking in both regular and complex sequences. This is attributed to multi-robot loop closure optimization, where

TABLE II: ATE [m] on S3E outdoor datasets. **Failed** indicates that the system cannot be initialized or track frames.

Methods	Library_1			Dormitory_1			Playground_1			Playground_2			Square_1			Square_2			Tunnel_1		
	α	β	γ	α	β	γ	α	β	γ	α	β	γ	α	β	γ	α	β	γ	α	β	γ
LIO-SAM	1.20	2.12	1.39	0.82	1.71	**0.70**	0.64	0.85	0.77	-	2.80	**0.70**	1.31	6.41	**0.70**	-	0.71	1.19	4.03	4.00	3.65
DiSCo-SLAM	1.44	1.73	1.45	0.77	**0.96**	0.92	0.48	**0.59**	0.67	-	**Failed**	0.71	1.35	**Failed**	0.82	-	0.75	0.49	4.09	4.40	3.71
Ours (uncorrected)	1.63	1.75	1.41	0.63	1.64	0.93	0.50	0.70	0.69	-	**Failed**	0.83	1.34	**Failed**	0.79	-	**0.70**	0.88	4.04	4.01	3.65
Ours (corrected)	**1.14**	**1.64**	**1.34**	**0.58**	1.50	0.89	**0.46**	0.66	**0.65**	-	**0.95**	0.78	**1.28**	**1.36**	0.79	-	0.73	**0.45**	**4.02**	**3.60**	**3.55**

TABLE III: RPE [m] on and S3E outdoor datasets. **Failed** indicates that the system cannot be initialized or track frames.

Methods	Library_1			Dormitory_1			Playground_1			Playground_2			Square_1			Square_2			Tunnel_1		
	α	β	γ	α	β	γ	α	β	γ	α	β	γ	α	β	γ	α	β	γ	α	β	γ
LIO-SAM	2.69	3.82	3.28	1.23	0.83	**6.90**	2.27	2.26	2.34	-	**2.43**	3.21	4.53	**1.56**	10.17	-	1.58	1.60	**0.12**	0.13	0.14
DiSCo-SLAM	**2.55**	3.93	3.26	**1.10**	0.74	6.94	2.15	**2.25**	2.48	-	**Failed**	3.20	4.52	**Failed**	10.14	-	1.59	1.58	0.13	**0.11**	0.15
Ours (uncorrected)	2.67	3.92	**3.24**	1.23	0.72	6.94	2.16	2.26	2.35	-	**Failed**	3.19	**4.48**	**Failed**	**10.12**	-	**1.57**	1.67	0.13	0.14	0.14
Ours (corrected)	2.63	**3.79**	3.31	1.20	**0.69**	6.93	**2.10**	2.27	**2.29**	-	2.44	**3.16**	4.52	1.61	10.16	-	1.58	**1.52**	0.14	0.14	**0.12**

TABLE IV: ATE and RPE of the KITTI Odometry Sequences

Seq.	Methods	ATE [m]			RPE [m]		
		α	β	γ	α	β	γ
Seq. 5	LIO-SAM	1.34	-	**0.86**	**0.05**	-	0.13
	DiSCo-SLAM	2.64	-	0.13	0.14	-	0.12
	Ours (uncorrected)	1.38	-	0.90	0.07	-	**0.11**
	Ours (corrected)	**0.86**	-	0.94	0.06	-	0.12
Seq. 6	LIO-SAM	**15.84**	2.23	21.80	**1.61**	1.73	**2.94**
	DiSCo-SLAM	26.90	5.79	**Failed**	1.74	1.76	**Failed**
	Ours (uncorrected)	16.12	1.96	20.92	1.67	1.73	3.18
	Ours (corrected)	17.32	**1.93**	**19.14**	1.72	**1.71**	4.23
Seq. 7	LIO-SAM	0.41	0.43	**0.53**	0.06	**0.04**	**0.05**
	DiSCo-SLAM	**0.40**	0.78	0.59	0.06	0.09	1.10
	Ours (uncorrected)	0.41	0.79	0.55	0.05	0.07	0.05
	Ours (corrected)	0.42	**0.42**	0.54	**0.04**	0.05	0.06
Seq. 9	LIO-SAM	1.76	2.80	9.54	0.04	0.16	0.42
	DiSCo-SLAM	2.25	1.77	2.63	0.09	0.06	0.07
	Ours (uncorrected)	1.73	1.74	5.69	0.03	0.10	0.28
	Ours (corrected)	**1.72**	**1.32**	**2.44**	**0.03**	**0.03**	**0.04**

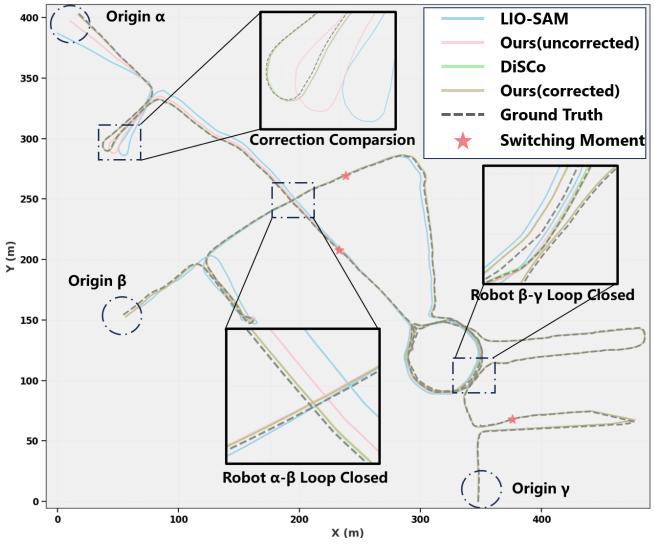

Fig. 6: Trajectory comparison of each system against the ground truth on the Dormitory sequence of the S3E dataset, where curves of different colors or styles denote the respective systems; the star markers indicate the points at which the optimization mode switches.

optimized poses provide real-time corrections to frontend positions, ensuring accurate local tracking and better alignment with ground truth.

To further illustrate these advantages, Fig. 6 presents the results on the "Dormitory sequence." The compared systems include CDC-SLAM (Ours (corrected), Ours (uncorrected)), LIO-SAM, and DiSCo-SLAM. Ours (corrected) aligns most closely with ground truth, maintaining stable trajectories without noticeable drift. By contrast, LIO-SAM and DiSCo-SLAM show larger deviations, especially at abrupt scene transitions and sharp turns, highlighting the robustness of our correction mechanism.

Notably, LIO-SAM shows greater drift in the first half of the sequence. The star markers in Fig. 6 indicate moments when our system switches between centralized and distributed modes. At loop closure positions, Ours (corrected) aligns best with the true loop-closing poses. The "Correction Comparison" section further shows that our correction improves both trajectory accuracy and global consistency, demonstrating the effectiveness of our adaptive mechanism in preserving local and global trajectory fidelity.

In summary, using S3E's ATE/RPE metrics (Tables II, III) and Fig. 6's visualization, our CDC-SLAM shows better accuracy, consistency and reliability. It stays stable in tricky scenarios (e.g., transitions, sharp turns) where LIO-SAM/DisCo-SLAM fail, proving robustness for outdoor multi-robot SLAM.

D. Evaluation on KITTI Dataset

We further validated the advantages of our system on the KITTI dataset, which comprises diverse urban driving scenarios. This dataset contains loop closure situations of varying complexity, and as presented in Table IV, our system maintains consistent accuracy across multiple sequences (e.g., Seq. 6, Seq. 9), demonstrating reliable long-term trajectory estimation. These results illustrate the system's robust adaptability and resilience in urban environments, where effective loop closure detection and precise correction are critical for ensuring trajectory stability over extended distances. Furthermore, the proposed system can handle challenging maneuvers, sharp turns, and complex intersections

without significant drift.

In summary, KITTI dataset experiments further confirm our CDC-SLAM's strengths: it maintains consistent long-term trajectory accuracy across diverse urban sequences, handling complex loop closures, turns, and intersections without major drift. Its effective loop detection and real-time correction ensure reliability for extended urban multi-robot SLAM, showing strong adaptability to dynamic cities.

V. CONCLUSIONS

This paper presents a range-triggered centralized-distributed collaborative LiDAR SLAM framework, named CDC-SLAM, designed for multi-robot teams to adaptively switch optimization modes and perform collaborative mapping in unknown environments. The system integrates two core mechanisms: Range-Triggered Dual-Optimization Switching and an adaptive fusion mechanism. Experimental results on both urban and campus environment datasets demonstrate that the framework achieves high accuracy and strong robustness. For future work, we plan to enhance the system's scalability—for example, enabling the selection of different front-ends and lightweight descriptors according to environmental conditions. In addition, we aim to develop more advanced methods to improve the system's resilience against incorrect loop closures and its ability to provide accurate initial poses.

REFERENCES

[1] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[2] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5135–5142, 2020.

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[4] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[5] S. Zhong, H. Chen, Y. Qi, D. Feng, Z. Chen, J. Wu, W. Wen, and M. Liu, "Colrio: Lidar-ranging-inertial centralized state estimation for robotic swarms," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3920–3926, 2024.

[6] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "Door-slam: Distributed, online, and outlier resilient slam for robotic teams," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1656–1663, 2020.

[7] 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 Robotics and Automation Letters*, vol. 7, no. 2, pp. 1150–1157, 2022.

[8] 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 Journal*, vol. 24, no. 4, pp. 4786–4797, 2024.

[9] P.-Y. Lajoie and G. Beltrame, "Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 475–482, 2024.

[10] D. Feng, Y. Qi, S. Zhong, Z. Chen, Q. Chen, H. Chen, J. Wu, and J. Ma, "S3e: A multi-robot multimodal dataset for collaborative slam," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11401–11408, 2024.

[11] B. Triggs, P. F. Mclauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *International Workshop on Vision Algorithms*, 2000.

[12] M. I. A. Lourakis and A. A. Argyros, "Sba: A software package for generic sparse bundle adjustment," *ACM Trans. Math. Softw.*, vol. 36, pp. 2:1–2:30, 2009.

[13] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, 2011.

[14] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," *Georgia Institute of Technology*, 2012.

[15] N. Boumal, *An Introduction to Optimization on Smooth Manifolds*. 2023.

[16] P. Schmuck and M. Chli, "Ccmlam: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *Journal of Field Robotics*, vol. 36, no. 4, pp. 763–781, 2019.

[17] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli, "Covins: Visual-inertial slam for centralized collaboration," in *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 171–176, 2021.

[18] J. Xie, X. He, J. Mao, L. Zhang, and X. Hu, "C2vir-slam: Centralized collaborative visual-inertial-range simultaneous localization and mapping," *Drones*, vol. 6, no. 11, 2022.

[19] A. Cunningham, M. Paluri, and F. Dellaert, "Ddf-sam: Fully distributed slam using constrained factor graphs," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3025–3030, 2010.

[20] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed trajectory estimation with privacy and communication constraints: A two-stage distributed gauss-seidel approach," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5261–5268, 2016.

[21] A. Cunningham, V. Indelman, and F. Dellaert, "Ddf-sam 2.0: Consistent distributed smoothing and mapping," in *2013 IEEE International Conference on Robotics and Automation*, pp. 5220–5227, 2013.

[22] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, "Distributed certifiably correct pose-graph optimization," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2137–2156, 2021.

[23] H. Xu, P. Liu, X. Chen, and S. Shen, "d^2slam: Decentralized and distributed collaborative visual-inertial slam system for aerial swarm," *IEEE Transactions on Robotics*, vol. 40, pp. 3445–3464, 2024.

[24] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, "Se-sync: a certifiably correct algorithm for synchronization over the special euclidean group," *Sage Publications*, 2019.

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

[26] H. Song, C. Liu, and H. Dai, "Bundledslam: An accurate visual slam system using multiple cameras," in *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 7, pp. 106–111, 2024.

[27] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, "Asynchronous and parallel distributed pose graph optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5819–5826, 2020.

[28] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "Pl-slam: A stereo slam system through the combination of points and line segments," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 734–746, 2019.

[29] J. Engel, T. Schps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*, 2014.

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

[31] Y. Wang, Z. Sun, C.-Z. Xu, S. E. Sarma, J. Yang, and H. Kong, "Lidar iris for loop-closure detection," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5769–5775, 2020.

[32] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4802–4809, 2018.