2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

MERSYS: A Collaborative Estimation and Dense Mapping System for Multi-Agent Generic SLAM

Qianhua Lai*, Enhao Zhao*, Shicai Fan and Jianxiao Zou

***Abstract*— Multi-agent collaborative Simultaneous Localization and Mapping (SLAM) is an effective way for large-scale mapping. However, this approach, which relies on Visual-Inertial Odometry(VIO) as input, suffers from limitations such as susceptibility to environmental influences and the difficulty in accurately constructing dense 3D maps. To address these challenges, this paper presents Multi-Estimation Robust SLAM System (MERSYS), a novel framework for three-dimensional dense mapping based on the fusion of Lidar-Inertial Odometry(LIO) and VIO. Benefiting from lower communication's costs and dense information acquisition capability, the proposed framework aims to achieve compatibility in processing both LIO and VIO inputs, establish joint loop closure detection to enable multi-map fusion, and then create a comprehensive global 3D dense point cloud map. Furthermore, an efficient communication strategy has been proposed to enable bidirectional transmission of dense and voluminous data. Experimental evaluations conducted on the publicly available HILTI SLAM 2021 dataset[10] as well as a real world dataset. Experimental results show that MERSYS achieves better results than state-of-the-art methods. The source code is available on the GitHub[1].**

I. INTRODUCTION

With the rapid advancement and widespread adoption of technologies such as autonomous driving and virtual reality, SLAM techniques have also experienced significant growth[11]. Presently, SLAM techniques have matured significantly for single unmanned vehicles and drones[5]. However, as SLAM technology is now being applied to larger and more complex tasks such as map generation, virtual reality, and wilderness search and rescue[8], there is a growing need for SLAM systems to support long-term mapping and the participation of multiple agents[2]. These requirements call for the design of a SLAM framework that encompasses communication between front-end and back-end components, compatibility with various sensor data, and the capability to fuse maps from multiple agents into a global map.

In map generation[20], individual unmanned drones or vehicles face challenges such as battery life, mapping efficiency, communication range limitations and sensor limitation. While recent works have mostly focused on achieving excellent results in specific scenarios, more practical scenarios demand a versatile SLAM framework. This framework should address the aforementioned challenges while maintaining the ability to rapidly and accurately mapping[19].

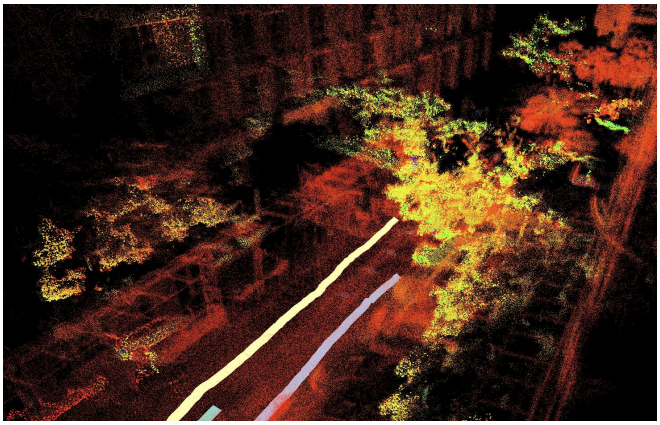

Fig. 1: We introduce MERSYS, a multi-agent SLAM framework compatible with both LIO and VIO.

MERSYS provides a novel, versatile, open-source platform that supports collaborative mapping with multiple agents and various input types. It integrates LIO and VIO input to simultaneously process and create dense 3D point cloud maps with RGB color information. Abundant environmental data supports more complex tasks in the future, such as object detection and dynamic object recognition. Furthermore, through the utilization of temporal and structural semantic features of dense data to optimize communication strategies, MERSYS attains real-time transmission and processing capabilities. Most notably, it leverages maps from heterogeneous agents, employing either LIO or VIO, to achieve fusion and optimization, thereby generating a global map. The refined data is subsequently fed back to the agents to rectify their respective local localization.

The contributions of this work can be summarized as follows:

- Introduction of an open-source, versatile, multi-robot localization and mapping framework compatible with LIO and VIO, capable of processing and fusing various types of input on the server.
- Implementation of loop detection between different devices and sensors to correct trajectory drift and improve localization and mapping accuracy.
- Creation of a 3D dense RGB point cloud map, providing scenes with realistic geometric shapes, rich textures, and colors, thereby supporting more

The authors are with the University of Electronic Science and Technology of China {qhlai, ehzhao, shicaifan, jxzou}@std.uestc.edu.cn
* Authors contributed equally to this work.
[1]https://github.com/UESTC-ARS/MERSYS

complex tasks in the future.

- An optimized transmission strategy has been devised, capitalizing on the temporal and structural semantics of dense data. This strategy facilitates real-time transmission and processing of substantial data volumes.

II. RELATED WORK

In 1986, Hugh Durrant-Whyte and John J. Leonard [16]introduced a probabilistic approach to robot localization and mapping and this approach is now what we call SLAM. Recent efforts have primarily focused on single-robot mapping,which can categorize into three types based on the sensor types used: LIDAR-Inertial Odometry-based, Visual-Inertial Odometry-based, and LIDAR-Visual-Inertial Odometry-based. LOAM[31], FASTLIO[29], [28], and LEGO-LOAM[26] use LIDAR and IMU to access environmental information, while camera and IMU-based works often use the ORBSLAM series[3] and VINS[21]. In recent years, some new methods are based on LIDAR-Visual-Inertial Odometry such as LVI-SAM[27] which integrate more sensor information to achieve better result. However, these single-robot systems, when applied in practical scenarios like large-scale mapping, lack competitiveness in terms of efficiency and performance compared to multi-robot systems like MERSYS.

Frameworks like COVINS[25], maplab2.0[6] and Swarm-Slam[15] are based on VIO for multi-robot applications. For instance, COVINS can receive inputs from various VIO front-ends and build a global map in the server back-end. Maplab2.0 uses semantic information to optimize relative pose errors. However, multi-robot frameworks are primarily designed for outdoor applications, where image information from cameras can be easily affected by environmental factors such as light condition. In addition, these methods all struggle to create dense map. In contrast, laser scanners are less affected, making LIO-based methods more robust and suitable for practical applications.

RaLI-Multi[12] is a distance-assisted laser-inertial multi-vehicle mapping system that designs a multi-metric weighted radar-inertial odometry. It receives local maps from each vehicle and measurements of distances between vehicles to optimize the global map. This fusion method is limited to situations where multiple front-ends are within each other's laser scanner range. It is a coarse map fusion method that lacks vector information and cannot perform inter-robot pose optimization.

Our approach, on the other hand, identifies similar scenes between different robot maps through loop closure detection. It stitches two local maps together by calculating a transformation matrix. This approach removes temporal and spatial consistency constraints, enhancing system flexibility. It also provides a more accurate transformation matrix, improving map fusion accuracy. Additionally, MERSYS builds 3D dense point cloud maps with RGB colors and implements loop closure detection between 3D point clouds and RGB images. This allows the image information from one robot to be applied to the colored point cloud of another robot connected through loop closure, enriching the map with textures and color information. This robust foundation supports more complex tasks. Moreover, considering the challenges of transmitting such vast amounts of information in practical applications, we have improved the communication module to establish stable communication between the front-end and back-end.

TABLE I: Compare with state-of-the-art framworks.(LC: Loop Closure; EXT:External source; OPT:Optimization)

	MULTI-Agent	Inter-LC	Intra-LC	LIDAR Input	Camera Input	EXT Localization Free	Heterigeneous OPT	Large Dense Map	RGB Features	Open Source
ORBSLAM3			√		√	√			√	√
VINS			√		√	√			√	√
FASTLIO 2				√		√				√
LVI-SAM			√	√	√	√	√		√	√
RALI-Multi	√			√						
COVINS	√	√	√		√	√	√			√
Maplab 2.0	√	√	√	√	√				√	√
MERSYS	√	√	√	√	√	√	√	√	√	√

In summary, MERSYS is an universal SLAM framework designed to address the challenges of multi-robot collaborative mapping in practical applications. It is compatible with VIO and LIO front-ends, supporting a wider range of sensors. It enhances localization, mapping capabilities, and robustness compared to previous frameworks.

III. The MERSYS Framswork

A. System Overview

As illustrated in Fig. 2, on the left side, the front-end comprises multiple LIO-based and VIO-based agents, each capable of running independently and building its own local map. Between these two ends, MERSYS utilizes peer-to-peer data frame transmission, including point clouds and RGB images with odometer information.

Next comes the critical aspect of multi-map fusion: place recognition. This module is mainly divided into two parts: loop closure detection between LIDAR frames and between LIDAR and RGB camera frames.For LiDAR point clouds, the feature matrix of the point cloud is extracted and combined with kd-tree for two rounds of comparison. The similarity of the comparison determines whether it is a loop relationship, greatly improving the efficiency of loop detection.The second part involves loop closure detection between LIDAR and RGB images.In

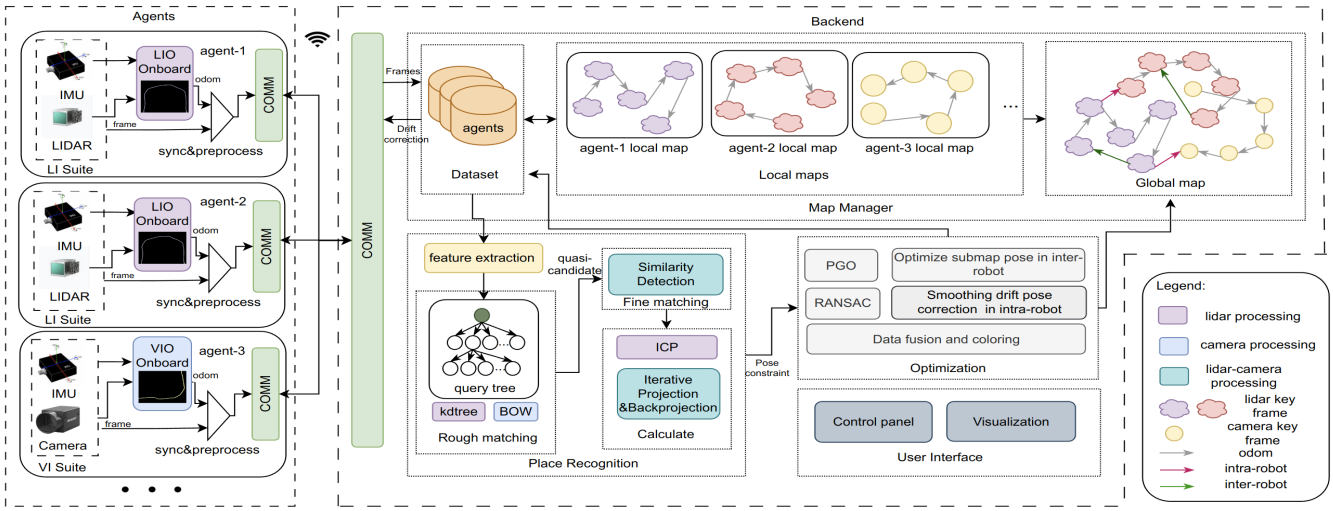

Fig. 2: Overview of the MERSYS framework and its main components

our method, we project the 3D point cloud onto the image plane and backproject it to calculate residuals, iteratively minimizing the mapping residuals using the Levenberg Marquardt algorithm[18] and Ceres Solver[1].

Lastly, trajectory optimization is conducted, and primarily, the optimization of inter-frame and inter-map constraint relations is achieved through Pose-Graph Optimization (PGO) [7], [13] . At the same time, the removal of erroneous or redundant constraint relations is performed using the Random Sample Consensus (RANSAC)[4] technique. After these processes, local maps (map1, map2, map3) shown in the Fig. 2 are merged into a global map.

B. Front-end Input

This pipeline utilizes a fundamental front-end component to furnish it with essential keyframe generation and local odometry. These inputs are then generalized and transmitted to the backend.

To evaluate the performance of MERSYS, in the experiments presented in this paper, we employ FASTLIO2[28] and VINS[21] as the LIO and VIO front-ends on the robot side.

C. Bidirectional Communication

In the context of multi-agent large-scale mapping, a set of asymmetric communication schemes has been developed to mitigate the impact of delay and bandwidth. In the front-end to back-end communication of MERSYS, redundant data transmission is avoided by considering the similarity between adjacent keyframes. Additionally, data downsampling is performed based on network conditions to ensure efficient communication.

Furthermore, by exploiting the structured scene blocks present in the point cloud, region growth, primarily focused on planes, is accomplished through the integration of spatial and point cloud intensity information with reconstructed normals. The resultant transmitted information comprises a serialized hybrid point cloud that encompasses both the positional and edge description details of planes, as well as the remaining scattered points. This approach significantly enhances transmission efficiency.

Regarding the back-end to front-end transmission, MERSYS incorporates drift correction to guide the optimization of local maps for each agent based on the fused global map. This correction process helps refine subsequent local process information.

D. Place Recognition

The solid-state LIDAR utilized in this study different from conventional mechanical LIDAR by its narrower field of view (FOV), which imposes limitations on the applicability of conventional LCD methods. However, it offers a significantly lower price.

To detect similar positions within multi-agent data and establish constraint relationships among the local maps of them, our proposed framework incorporates two loop detection's forms: LIO-to-LIO detection and LIO-to-VIO detection. Notably, this framework does not rely on prior pose information such as GNSS.

1) Loop Detection Between LIDAR Frames:

Due to the significant number of keyframes generated by multiple robots, a two-step search approach has been implemented in this work to enhance performance. Initially, a lightweight coarse search is conducted to identify candidate keyframes, followed by a high-precision search to find the most similar one. Subsequently, this keyframe is extracted and passed to the Iterative Closest Point (ICP)[23] algorithm to compute accurate transformation relationships.

The two-step search approach in this study draws inspiration from [14]. It operates in the robot coordinate system and encodes the observed point cloud data into a 2D matrix with reduced search cost, while preserving its detection specificity and rotational invariance. The construction of the 2D matrix involves dividing the polar

Algorithm 1: Place recognition

Input: client id: i_q, query frame: f_q, query frame's local position: T_q
Output: match map id: i_m, relative pose transfer matrix: T

Save f_q to the database with id i_q
if f_q *is a solid-state lidar frame* **then**
  1. f_q -> 2D matrix m_1, m_{fov} and save;
  2. The rough matrix queries a k-d tree -> quasi-candidate frames;
  3. Fine matching with the higher precision matrices ->candidate frames & similarity.;
else if f_q *is a RGB image frame* **then**
  1. f_q edge information -> 2D matrix and save.;
  2. Two step matching -> candidate frames & similarity.;
if *similarity* > *threshold* **then**
  1. Calculate T (ICP or iterative projection and backprojection) ;
  2. Get the best i_m and T ;
  return i_m and T;
else
  return;

coordinates into grids with equal intervals in terms of angle and distance, centered around the robot. The Z-axis height of the highest point observed within each corresponding grid is assigned to that grid. Two 2D matrices are generated based on different grid sizes. The larger grid matrix is employed for coarse matching, utilizing a kd-tree[9] to efficiently filter out groups of candidate keyframes that approximately satisfy the overall contour of the query point cloud. Subsequently, the corresponding smaller grid matrix groups are extracted for fine matching.

In the fine matching phase, to assess the similarity between the query matrix M^q and the candidate matrix M^c, the presence of column offsets caused by different viewpoints within the same scene makes it challenging to accurately calculate the similarity between the two matrices. Therefore, a preprocessing step is performed to mitigate this issue. Firstly, the column means of the matrices are computed, resulting in two one-dimensional matrices. These one-dimensional matrices are then compared through a sliding window approach to identify the column offset that minimizes the difference between the two matrices. Subsequently, this determined column offset is incorporated into the following formula to calculate the similarity:

$$d(M^q, M^c) = \frac{1}{N_r} \sum_{j=1}^{N_r} \left(1 - \frac{c_j^q \cdot c_j^c}{\|c_j^q\| \|c_j^c\|} \right). \tag{1}$$

Here, N_r represents the number of rows in the matrices, c_j^q and c_j^c denote the column vectors of the two matrices after applying the column offset. The similarity score between two column vectors is computed by taking the dot product of the column vectors and dividing it by the product of their magnitudes. By calculating the similarity score for all column vectors, the overall similarity between the two matrices can be obtained. Finally, a selection is made from the set of candidate keyframe groups, choosing several frames with the highest similarity scores to form the candidate keyframe group.

After obtaining a set of reliable candidate keyframes, the original point cloud data is extracted from these frames. Subsequently, the ICP[23] is applied to align the extracted point cloud with the query point cloud. The ICP aims to find the transformation (rotation and translation) that minimizes the distance between corresponding points from the two point clouds. The mathematical formulation of the ICP can be expressed as follows:

$$E(R,t) = \frac{1}{n} \sum_{i=1}^{n} \|q_i - (Rp_i + t)\|^2 \tag{2}$$

where n represents the number of nearest neighbor point pairs, p_i denotes a point from the target point cloud P, and q_i represents the closest point in the source point cloud Q corresponding to p_i. The rotation matrix is denoted by R, and t represents the translation vector.

If the obtained transformation matrix exhibits a sufficiently high level of confidence in its feasibility, it is saved and passed on to the Map Manager.

2) Loop Detection Between LIDAR Frame and Image Frame:

Point clouds and images are two different modalities that cannot be directly compared to compute the transformation matrix between them. Additionally, to conserve computational resources, we first utilize the bag-of-words approach from the previous section to reduce the number of candidates for comparison. Subsequently, we refer to [17] and [30] for the next stage of comparison. According to solid-state LiDAR, the FOV of a single-frame LiDAR point cloud is similar to that of camera images. Therefore, scale variation issues during the projection process are not significant. The 3D LiDAR points are projected onto the plane of the image using [24], and the LiDAR's pose is optimized using LiDAR Bundle Adjustment (BA) to reduce pose drift. In this context, calibration residuals between visual points and the LiDAR plane are introduced to assess their relationship. The following formula is used for this purpose:

$$^{L_i}\mathbf{p}_j = {}^{L_i}_{C}\mathbf{R}\,{}^{C}\mathbf{p}_j + {}^{L_i}_{C}\mathbf{t} = {}^{L_i}\mathbf{p}_j^{gt} + \boldsymbol{\delta}_{L_i\mathbf{p}_j} \tag{3}$$

$$where\, {}^{L_i}\mathbf{p}_j^{gt} = {}^{L_i}_{C}\mathbf{R}\,{}^{C}\mathbf{p}_j^{gt} + {}^{L_i}_{C}\mathbf{t}, \delta_{L_i\mathbf{p}_j} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \Sigma_{L_i\mathbf{p}_j}) \tag{4}$$

$$\boldsymbol{\Sigma}_{L_i\mathbf{p}_j} = {}^{L_i}_{C}\mathbf{R}\boldsymbol{\Sigma}^{L_i}_{C_{\mathbf{p}_j C}}\mathbf{R}^T \tag{5}$$

$$\mathbf{n}_{ij}^T({}^{L_i}\mathbf{p}_j^{gt} - \bar{\mathbf{q}}_{ij}) = 0. \tag{6}$$

For each feature point ${}^{C}\mathbf{p}_j$, its projected point ${}^{L_i}\mathbf{p}_j$ on the LiDAR plane is determined by the transformation matrix ${}^{L_i}\mathbf{T}_C$, which is obtained from the SE(3) transformation from the first camera frame to the i-th LiDAR frame, given as ${}^{L_i}\mathbf{T}C = (C^{L_i}\mathbf{R}, C^{L_i}\mathbf{t})$. Considering that

${}^{C}\mathbf{p}j$ is subject to Gaussian noise, it can be expressed as ${}^{C}\mathbf{p}j = {}^{C}\mathbf{p}j^{gt} + \boldsymbol{\delta}C\mathbf{p}j$, where ${}^{C}\mathbf{p}j^{gt}$ represents the ground-truth position and $\boldsymbol{\delta}_{\boldsymbol{p}_j} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{C_{\mathbf{p}_j}})$ represents the uncertainty.

By combining the equations and employing maximum likelihood estimation (MLE), the residuals between points and planes can be derived using the following equation:

$$E^P = \frac{1}{2} \sum_{(i,j)\in\mathcal{V}^P} \frac{(\mathbf{n}_{ij}^T({}^{L_i}\mathbf{p}_j - \bar{\mathbf{q}}_{ij}))^2}{\mathbf{n}_{ij}^T \, {}_{L_i\mathbf{p}_j}\mathbf{n}_{ij}} \tag{7}$$

If the residual value exceeds a predefined threshold, the feature point is considered to be associated with the corresponding LiDAR plane. Subsequently, the feature points are reprojected back onto the image, and the reprojection residuals are computed. The process is similar to point-to-plane projection, and the specific formula for the final calculation is as follows:

$$E^V = \frac{1}{2} \sum_{(i,j)\in\mathcal{V}^V} (\mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij})^T \boldsymbol{\Sigma}_{\mathbf{x}_{ij}}^{-1} (\mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij}) \tag{8}$$

Finally, by combining the two residuals and utilizing the Levenberg-Marquardt algorithm with the Ceres Solver, the transformation matrix between camera frames and LiDAR frames is obtained, and it is subsequently passed to the map fusion module.

E. Map Alignment and Optimization

1) Map Fusion Process:

In the map fusion module, after obtaining the pose correspondences between different agents, a RANSAC-based approach [4] is initially employed to remove outlier data. Subsequently, for agents equipped with LIO, a factor graph optimization is applied to refine their poses in the global map. This optimization process is further used to continuously propagate and correct drift errors from previous frames within the factor graph. For agents with VIO, upon establishing associations with other agents through loop closures, a similar optimization process is conducted. In addition, local point cloud frames associated with RGB keyframes are identified, and color information is propagated by mapping images reproject to point clouds, achieving RGB map.

2) PGO:

In order to make full use of the overconstrained or indirect constraint relationships between these sub maps, the map alignment module integrates PGO [7], [13] to estimate the attitude relationships between maps as accurately as possible. The formula used is as follows:

$$\Theta^* = \arg\min_{\Theta} \frac{1}{2} \sum \|h_i(\Theta_i) - z_i\|_{\sum_i}^2 \tag{9}$$

Θ represents the set of map positions. $|h_i(\Theta_i) - z_i|$ represents the residual between the measurements obtained from Sec. III-D and the prior pose. Meanwhile, $\sum_i$ is used to denote the information matrix for the relative pose constraints.

IV. EXPERIMENTS AND DISCUSSIONS

We evaluate our framework by localizing and mapping multiple ground vehicles as shown in Fig. 4 in reference maps for both the Hilti SLAM Dataset 2021[10] and a customized dataset named YX. Our pipeline is implemented in C++, compatible with ROS [22] and runs in real-time on an Intel i5 CPU with 64 GB of RAM on the backend and AMD R7 NUC on the agents. The front end is equipped with Livox Avia, Hikvision Industrial Camera.To capture ground truth data, we use RTK sensors for outdoor environments and the NOKOV motion capture system for indoor environments.

The Hilti SLAM dataset 2021 is mainly used to demonstrate the ability of pipelines to locate vehicles in challenging environments, their robustness in scene recognition from different viewpoints, and accuracy compared to other methods. This customized multi-agent dataset is recorded in a community located in Shenzhen, China, as well as on the ground floor of a office building at our university, and conducted real-world experiments. And the main experiment objectives for this dataset are to showcase the pipeline' s ability to handle and align reference maps created from more unmanned devices, compatibility with different scenarios, and its efficiency in mapping large scenes.

The community presents a highly complex and extensive environment, characterized by uneven terrain, various buildings, trees, extensive shrubbery, and dynamic elements such as pedestrians and vehicles, which continuously challenge the solution's self-localization capabilities and its ability to collect environmental data for mapping in the presence of strong interference. Additionally, in indoor environment where GNSS signals cannot be received, and where scene details are richer and more diverse, the challenge of constructing dense maps becomes even greater. Furthermore, we captured outdoor environmental datasets during nighttime to evaluate the mapping performance of LIO and VIO in environments with significant changes in lighting conditions.

A. Single-Agent Mapping

The Table.II demonstrates the superior positioning accuracy of MERSYS. By evaluating its performance on the Hilti dataset, MERSYS exhibits significantly improved positioning accuracy compared to existing state-of-the-art algorithms. Furthermore, when applied to collaborative mapping scenarios involving multiple ground platforms, MERSYS outperforms other methods in terms of map fusion, yielding more favorable outcomes.

B. Outdoor Multi-Agent Mapping

During outdoor environment recording, we employed three unmanned vehicles that initiated motion from random locations within the community. Each vehicle operated independently for a certain duration, with some overlap in trajectories. The total length of these three trajectories was 2350 meters. As shown in the left side

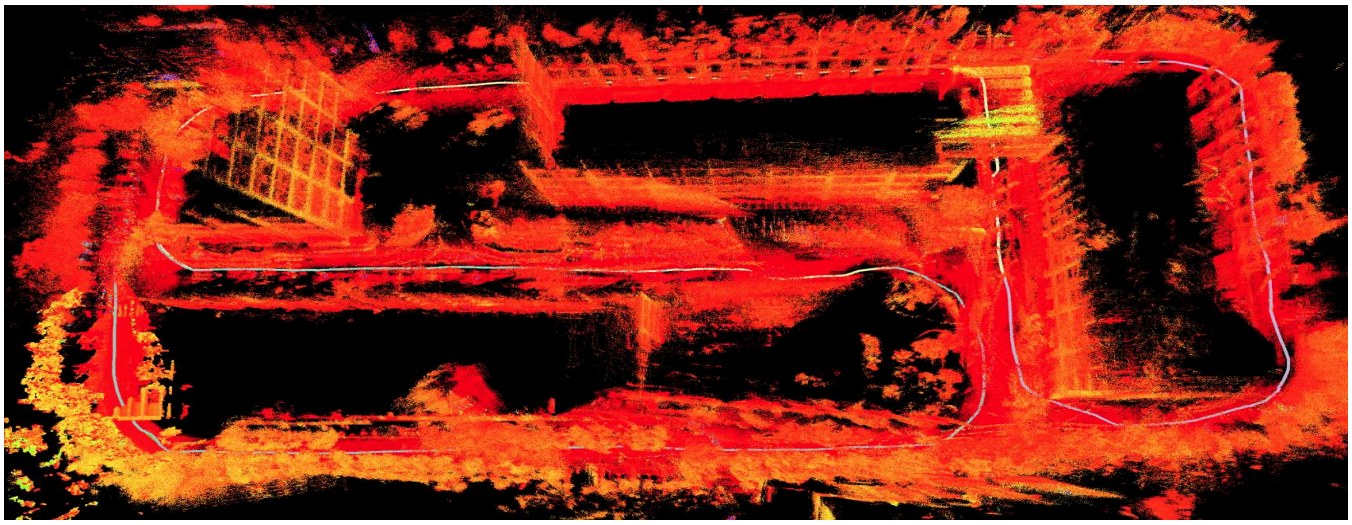

Fig. 3: In the context of real-world community scenarios, a global top-down map was constructed by amalgamating the local trajectories of three robots. The blue trajectory line represents client1, the yellow one represents client2, and the purple one represents client3.

TABLE II: The Root Mean Squared Error (RMSE) of the Absolute Position Error (APE) is used to compare the multi-machine and single-machine algorithms. The MERSYS employs FASTLIO 2[28] as its front-end. X: unable to process multi-agent mapping. S: single agent M_{num}: multi-agent with a quantity of num. I: indoor.O: outdoor.

	Single Client				MULTI Client		
	Hilti Basement 4	Hilti Construction Site 2	Hilti Campus 2	YX SO	YX M_2I	YX M_3O	YX M_3O-night
MERSYS	0.04	0.07	0.08	0.19	0.06	0.24	0.20
COVINS-G	0.37	0.34	0.28	1.26	0.38	0.33	2.77
Maplab2.0	0.12	0.19	0.18	0.89	0.22	0.17	1.65
FASTLIO 2	0.04	0.07	0.08	0.06	X	X	X
ORB-SLAM3	1.71	2.77	2.24	1.43	X	X	X

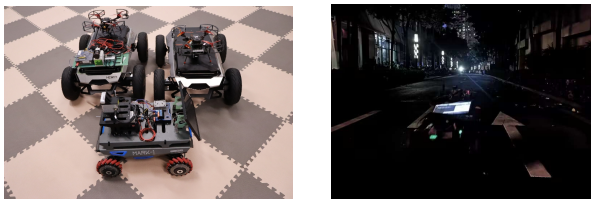

Fig. 4: Unmanned ground vehicles with experimental devices and the real-world test in night

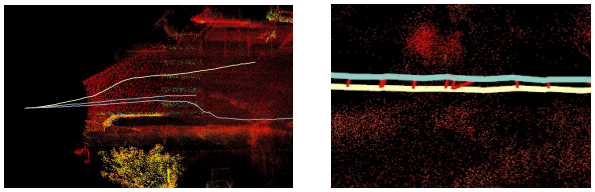

Fig. 5: This is the global map for outdoor mapping. On the left, the image shows the initial map when the robots have just started moving. Since the robots haven't acquired global poses yet, all three sub maps are overlaid. On the right, the image depicts the scenario when loop closure occurs between client1 and client2. The red line indicates the loop closure frames detected by both clients

of Fig. 5, initially, each robot establishes a local coordinate system with its starting point as the origin. Consequently, the back-end receives data from the three front-ends, each mapping their surroundings with their respective starting points as origins. Thus, at this stage, the global map is characterized by disorder and inaccuracies. However, when a inter-agent loop closure was detected between client1 and client2, indicated by the red line in the right side of Fig.5, the algorithm described in Sec.III-D was utilized to compute the transformation matrix T_{ff} between two frames and then obtain the transformation matrix T_{12} between the two maps. With this key information, the server established constraints between this two clients and optimized its own trajectory. After optimization, the server fused these two sub maps of the global map based on the new transformation and corrected trajectories.

Subsequently, when a new inter-agent loop closure occurred between client2 and client3, yielding the transformation matrix T between them, and using the factor graph optimization algorithm based on the connection between client1 and client2, constraints between client1 and client3 were determined. This increased the number of constraints and improved trajectory correction. According to the data in Table.II and III, the average APE of MERSYS stands at only 0.19 meters, with an average bandwidth of 772.62 KB/s. These metrics exemplify its remarkable robustness in complex environments, precise localization capabilities, and its ability to consistently construct dense point cloud maps in such large-scale environments. Moreover, it ensures stable and efficient data transmission between the frontend and backend components.

In addition, we evaluated the recorded dataset on COVINS-G and maplab 2.0, and the test results are also presented in the Table.II. In cases with relatively stable lighting conditions, both COVINS-G and maplab exhibit good localization accuracy, with APE values of 1.26 meters and 0.89 meters, respectively. However, when compared to MERSYS, there is still a gap in performance. This is attributed to the complex environment with numerous occlusions, making it challenging to accurately estimate the true spatial poses of distant objects and leading to deviations in self-pose estimation. MERSYS, benefiting from the incorporation of LIO input, is capable of promptly correcting such errors. Furthermore, we conducted tests using nighttime datasets. Due to poor lighting conditions at night, characterized by only a few street lamps and billboards, the robot's camera received varying and unstable light intensities during motion. In areas with excessive brightness or darkness, it was unable to extract meaningful feature points. Thus, the comparative analysis demonstrates that MERSYS has enhanced robustness compared to existing multi-robot cooperative methods.

C. Indoor Multi-Agent Mapping

In indoor testing, the primary focus was on evaluating map fusion accuracy and localization precision in complex indoor geometric environments. As illustrated in Fig.6, this environment consists of a main exhibition hall, a secondary exhibition hall, multiple walls with intricate color information, and an irregular geometric display wall. Due to the relatively small size of this environment, it was suitable for collaborative mapping with two robots. The selection of this scenario was crucial because the environment not only exhibited various colors and textures but also featured complex and fragmented geometric surfaces. This complexity, coupled with the limited vertical information acquisition capabilities of LiDAR sensors, added to the challenge of obtaining accurate and useful information.

Furthermore, the environment included many glass walls, which could lead to reflection phenomena and result in erroneous LiDAR data. However, with the integration of loop closure detection, not only could corrections be made between LiDAR scans but errors could also be further reduced by incorporating loop closures with images. Therefore, MERSYS was able to construct a 3D point cloud map that closely represents reality, with high texture and detail fidelity, as evident in the colored image. This high-quality color information provided by VIO can be directly used for complex tasks such as object recognition in the future.

The data presented in the Table.II reveals that in indoor environments, MERSYS exhibits localization accuracy comparable to that of COVINS-G and maplab2.0. This is because the robustness of VIO in indoor environments is significantly improved compared to outdoor settings, and the mapping scene has fewer occlusions and distinct geometric edges, which facilitates feature point extraction. In summary, in indoor environments favoring VIO frontend performance over LIO input, the work presented in this paper has achieved results on par with existing methods. Additionally, it has established a 3D dense RGB point cloud map, making it advantageous for practical applications.

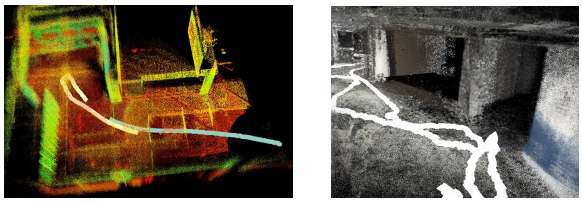

Fig. 6: Dense indoor multi machine mapping and point cloud coloring

D. Network Transmision

In our experiments, a pre-recorded dataset was utilized, and both the front-end and back-end components engaged in actual network transmissions over a wireless network. The results of our tests on different datasets, as shown in Table.III, demonstrate that by employing the communication methods described in III-C, we significantly reduced the transmission bandwidth usage.

Fig. 7: Optimization of semantic transmission of point clouds

TABLE III: The network transmission rate and latency of the back-end which are measured by taking the average of three consecutive runs and the average of multiple agents.

Recordings	*Receive*	*Transmit*	*Avg Latency*
Basement 4	422.22kB/s	1.14kB/s	832.3ms
Campus 2	386.67kB/s	1.32kB/s	932.3ms
Construction Site 2	325.41kB/s	1.15kB/s	899.3ms
YX SO	772.62kB/s	2.12kB/s	1154.8ms
YX M_3O	709.29kB/s	2.04kB/s	1221.5ms

V. CONCLUSION

We propose a pipeline for a multi robot collaborative mapping platform that is compatible with multiple inputs in a large-scale environment. Experiments with the Hilti SLAM dataset 2021 and our custom multi-robot dataset have substantiated the accuracy of our pipeline, showcasing its proficiency in effectively fusing maps derived from multiple data sources across multiple robots.

REFERENCES

[1] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 3 2022.

[2] Luca Bartolomei, Marco Karrer, and Margarita Chli. Multi-robot coordination with agent-server architecture for autonomous navigation in partially unknown environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1516–1522, 2020.

[3] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[4] H Cantzler. Random sample consensus (RANSAC). 1981.

[5] Shiyuan Chai, Zhen Yang, Jichuan Huang, Xiaoyang Li, Yiyang Zhao, and Deyun Zhou. Study on cooperative air-to-ground surveillance planning and controlling for unmanned aerial vehicles. In *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, pages 905–910, 2022.

[6] Andrei Cramariuc, Lukas Bernreiter, Florian Tschopp, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, and Cesar Cadena. maplab 2.0 –a modular and multi-modal mapping framework. *IEEE Robotics and Automation Letters*, 8(2):520–527, 2023.

[7] Frank Dellaert and GTSAM Contributors. borglab/gtsam, May 2022.

[8] Jeffrey Delmerico, Stefano Mintchev, Alessandro Giusti, Boris Gromov, Kamilo Melo, Tomislav Horvat, Cesar Cadena, Marco Hutter, Auke Ijspeert, Dario Floreano, Luca M. Gambardella, Roland Siegwart, and Davide Scaramuzza. The current state and future outlook of rescue robotics. *Journal of Field Robotics*, 36(7):1171–1191, 2019.

[9] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, sep 1977.

[10] Michael Helmberger, Kristian Morin, Beda Berner, Nitish Kumar, Giovanni Cioffi, and Davide Scaramuzza. The hilti SLAM challenge dataset. *IEEE Robotics and Automation Letters*, 7(3):7518–7525, jul 2022.

[11] Yujiao Jia, Xinying Yan, and Yihan Xu. A survey of simultaneous localization and mapping for robot. In *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 1, pages 857–861, Dec 2019.

[12] Zhe Jin and Chaoyang Jiang. Range-aided lidar-inertial multi-vehicle mapping in degenerate environment. 2023.

[13] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *2011 IEEE International Conference on Robotics and Automation*, pages 3281–3288, May 2011.

[14] Giseop Kim and Ayoung Kim. Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Oct. 2018.

[15] Pierre-Yves Lajoie and Giovanni Beltrame. Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. *IEEE Robotics and Automation Letters*, 9(1):475–482, 2024.

[16] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pages 1442–1447 vol.3, 1991.

[17] Liang Li, Haotian Li, Xiyuan Liu, Dongjiao He, Ziliang Miao, Fanze Kong, Rundong Li, Zheng Liu, and Fu Zhang. Joint intrinsic and extrinsic LiDAR-camera calibration in targetless environments using plane-constrained bundle adjustment. Number arXiv:2308.12629. arXiv.

[18] Jorge J. Moré. The levenberg-marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, pages 105–116, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

[19] Manthan Patel, Aditya Bandopadhyay, and Aamir Ahmad. Collaborative mapping of archaeological sites using multiple uavs. In Marcelo H. Ang Jr, Hajime Asama, Wei Lin, and Shaohui Foong, editors, *Intelligent Autonomous Systems 16*, pages 54–70, Cham, 2022. Springer International Publishing.

[20] Lukas Platinsky, Michal Szabados, Filip Hlasek, Ross Hemsley, Luca Del Pero, Andrej Pancik, Bryan Baum, Hugo Grimmett, and Peter Ondruska. Collaborative augmented reality on smartphones via life-long city-scale maps. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 533–541, 2020.

[21] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[22] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, may 2009.

[23] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[24] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[25] Patrik Schmuck, Thomas Ziegler, Marco Karrer, Jonathan Perraudin, and Margarita Chli. Covins: Visual-inertial slam for centralized collaboration. pages 171–176, 2021.

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

[27] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5692–5698, 2021.

[28] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[29] Wei Xu and Fu Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021.

[30] Chongjian Yuan, Xiyuan Liu, Xiaoping Hong, and Fu Zhang. Pixel-level extrinsic self calibration of high resolution LiDAR and camera in targetless environments. (arXiv:2103.01627).

[31] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. 07 2014.