# CCMD-SLAM: Communication-Efficient Centralized Multirobot Dense SLAM With Real-Time Point Cloud Maintenance

Chenle Zuo, Zhao Feng, and Xiaohui Xiao, *Member, IEEE*

***Abstract*— This article presents a communication-efficient centralized multirobot dense simultaneous localization and mapping (CCMD-SLAM) system with real-time point cloud maintenance, to address the limitations of data transmission and real-time creation and updating of dense maps in multirobot SLAM. This method solves the problem of high bandwidth consumption for information transmission in multirobot SLAM by preprocessing and compressing the transmitted data and filtering the red green blue-depth (RGB-D) information using the co-viewing degree of keyframes. The proposed method uses loop closure detection to integrate information from multiple robots and establish a global point cloud. In addition, a keyframe and point cloud storage mechanism is designed to facilitate real-time maintenance of global point cloud data. Through comprehensive evaluations using standard datasets and real-world experiments, CCMD-SLAM significantly alleviates data transmission pressures, enables flexible global point cloud management across multiple robots, and effectively achieves dense mapping for multirobot systems.**

***Index Terms*— Dense mapping, multirobot, red green blue-depth (RGB-D), simultaneous localization and mapping (SLAM).**

## I. Introduction

DENSE mapping is increasingly popular in various fields, such as environmental mapping, search and rescue, and augmented/virtual reality [1], [2]. Multirobot systems can efficiently accomplish the large-scale dense mapping of scenes [3], [4]. However, multirobot dense mapping is a challenging task. Size and weight limitations make it challenging for conventional perception robots, such as unmanned aerial vehicles (UAVs), to use powerful processors and storage devices for distributed real-time dense reconstruction and storage [5], [6]. In contrast, a centralized method can address these limitations by centralizing data processing and storage on a central server. However, the large volume of red green blue-depth (RGB-D) data introduces significant transmission burdens, and the scalability and performance of the system are constrained by network bandwidth. On the other hand, while establishing a global consistent dense map through the data from multiple robots, real-time updating and maintenance of the established global dense map are essential for achieving more accurate reconstruction [7]. This maintenance involves incorporating update information from different levels of the map to ensure its continuous accuracy and reliability.

In recent years, numerous researchers have focused on studying vision-based algorithms for simultaneous localization and mapping (SLAM) of multiple robots, which have demonstrated promising results in both map building and localization. These methods can be broadly classified into filter-based and keyframe-based approaches. Some of the classic works based on filtering include C2TAM [8] and Flying Smartphones [9]. In contrast to filtering-based approaches, keyframe-based methods exhibit advantages in terms of data transmission and fusion, such as classical work CoSLAM [10], MOARSLAM [11], CVISLAM [12], and COVINS [13]. On the other hand, dense mapping techniques have also been extensively researched and applied. Examples include KinectFusion [14] as a model-based approach, and Contour-SLAM [15] and ElasticFusion [16] which use visual feature bundle adjustment (BA) and pose graph optimization (PGO). Scholars have continuously studied and improved the tradeoff between speed and accuracy in dense map reconstruction. However, research on multirobot dense mapping remains limited [17].

Multirobot dense mapping algorithms are crucial for large-scale scene reconstruction. Achieving multirobot dense mapping presents several challenges such as limited bandwidth and data transfer, as well as the joint maintenance of a global dense map by multiple robots. Current researches focus on task assignment and collaborative path planning [4], dense mapping for distributed robot clusters [18], or reducing transmission bandwidth by increasing the computational pressure on the front-end which requires powerful capabilities for each robot [17]. To achieve centralized multirobot dense mapping while minimizing the communication overhead and enabling fast updates and maintenance of the dense map, there is a need for an algorithm that addresses these requirements.

To solve the above problems, we propose a communication-efficient centralized multirobot real-time dense mapping system. The system uses a generalized visual odometry (VO) front-end. A transmission module

Manuscript received 21 August 2023; revised 15 March 2024; accepted 5 April 2024. Date of publication 14 May 2024; date of current version 29 May 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB2100903. The Associate Editor coordinating the review process was Dr. Jochen Lang. *(Corresponding authors: Zhao Feng; Xiaohui Xiao.)*

The authors are with the School of Power and Mechanical Engineering, Wuhan University, Wuhan 430072, China (e-mail: zcl0008@whu.edu.cn; fengzhao@whu.edu.cn; xhxiao@whu.edu.cn).

Digital Object Identifier 10.1109/TIM.2024.3398100

is designed to mitigate transmission stress using data preprocessing, selective transmission, and compressed transmission techniques. Loop closure detection in the back-end consolidates information from multiple robots to establish a comprehensive global dense map. Moreover, a storage mechanism for keyframes and point clouds is devised to facilitate efficient information retrieval and global point cloud updates. Through these methodologies, the system successfully achieves centralized multirobot dense reconstruction. The contributions are summarized as follows.

1) We propose a novel framework for dense mapping in multirobot systems. Using a back-end dense mapping approach, we alleviate the computational burden on the front-end proxies.
2) A transmission module is designed to mitigate transmission stress using data preprocessing, selective transmission, and compressed transmission techniques.
3) We establish a globally consistent dense map and design a storage mechanism for keyframes and point clouds to facilitate efficient information retrieval and global point cloud updates.

The remainder of this article is organized as follows: Section II discusses related work, Section III gives the system overview of our proposal, Section IV shows the mapping and transmission algorithm implemented on the front-end robots, and Section V describes the dense mapping algorithm executed on the back-end server. Section VI shows the experimental results, and Section VII gives the conclusions and future work.

## II. Related Work

In the task of centralized multirobot dense mapping, the core research topic is in multirobot SLAM and dense mapping, and in this section we briefly review the related work in these two parts.

### A. Multirobot SLAM

With the development of SLAM algorithms and multirobot systems, the study of multirobot SLAM has received more and more attention. CoSLAM [10] proposes a centralized multimonocular camera SLAM algorithm in dynamic environments, which achieves better results in indoor scenes. C2TAM [8] is a distributed multirobot SLAM algorithm that uses a cloud server for map optimization and storage, reducing the memory and computational pressure on individual robots. However, it has a high demand for network connectivity. MOARSLAM [11] investigates the scalability of multiple robots and makes it possible for heterogeneous devices to easily extend the map in SLAM through a client-server approach. CCM-SLAM [19] presents a multirobot centralized collaborative SLAM framework. Each robot runs an independent visual odometer and the server performs loop closure detection and global optimization to achieve collaborative multirobot mapping. However, due to the pressure of communication traffic, the bandwidth limits the number of robots. CVI-SLAM [12], on the other hand, proposes a keyframe-based multirobot visual inertial collaborative SLAM system, which is also similarly limited by the communication bandwidth. To reduce the communication pressure and increase the number of robots, PairCon-SLAM [20] leverages socket transmission mechanism to distribute computational and storage resources across multiple PCs for enhanced scalability in large-scale dense mapping. COVINS [13] proposes a framework that increases the number of robots to up to 12 robots for simultaneous mapping by reducing redundant information and coordination overhead and enables flexible client-side robot scalability.

In the aforementioned multirobot SLAM algorithms, researchers have made significant efforts in achieving global consistency mapping and multirobot communication, successfully constructing sparse maps in multirobot scenarios. However, in the context of dense mapping with multiple robots, the transmission of a vast amount of dense information remains a challenge. The existing communication methods need further research and exploration to enable efficient transmission of such data. This article presents a framework for optimizing data transmission in multirobot systems, incorporating techniques such as filtering out irrelevant data, optimizing data formats, and using data compression to further enhance communication bandwidth utilization. This framework provides a reliable foundation for dense mapping in the back-end of multirobot systems.

### B. Dense Mapping SLAM

SLAM algorithms for dense mapping are generally constructed using LiDAR [21], RGB-D cameras [22], or stereo vision [23]. KinectFusion [14] aligns frame to model using iterative closest point (ICP) algorithm, which is the earliest algorithm based on RGB-D cameras to achieve dense scene building and requires high stability of motion. ElasticFusion [16] uses the method of surfels based on KinnectFusion to perform loop closure detection by means of model-to-model to ensure the global consistency of the closed loop. BundleFusion [24], on the other hand, improves on the reconstruction of large scenes. In recent years, the proposed ORBSLAM [25] makes feature-based dense mapping algorithms with better results [26], [27], and feature-based algorithms are also more convenient for direct data transfer from multiple robots. In multirobot dense mapping, a part of the research [4], [28] mainly focuses on task assignment and path planning of multirobot in maps. Kimera-multi [18], on the other hand, implements a distributed multirobot semantic SLAM system using an efficient communication method. The other algorithm [17], [29] focuses on front-end mapping and improves the representation of point cloud data to alleviate communication overhead. However, this approach increases computational burden on the front-end and lacks efficiency in timely optimization and updating of dense maps.

Compared with the aforementioned papers, this article proposes a novel framework for multirobot dense mapping. By adopting a back-end dense mapping approach, the computational burden on the front-end robots is reduced. In addition, the framework introduces a well-designed mechanism for storing keyframes and point clouds, allowing for flexible management and real-time maintenance of the back-end point cloud data.

## III. System Overview

In this article, we propose communication-efficient centralized multirobot dense SLAM (CCMD-SLAM)-based on COVINS (visual–inertial SLAM for centralized collaboration) [13]. The key differences between CCMD-SLAM and COVINS lie in the data interaction module, point cloud management module, and dense VO module. The proposed algorithm is specifically designed and aims to achieve efficient multirobot dense mapping.

The system structure of CCMD-SLAM is shown in Fig. 1. In this system, each robot uses an RGB-D VO front-end to perform local mapping and maintenance. On the robot side, we propose a method to optimize data transmission by filtering and compressing the data, which reduces the transmission pressure. Robots transmit data through networks, offloading computationally intensive processes to back-end servers. In terms of transmission, we design an efficient data transfer framework by optimizing, filtering, and compressing the data, reducing transmission pressure and delivering it to the server for decompression and conversion. After back-end mapping and optimization, we propose a point cloud management module. The system establishes local dense point cloud maps for each robot individually based on their RGB-D information. Through the map management module, overlapping maps are detected and fused to create a global dense point cloud map. To enhance the accuracy and storage efficiency of the global point cloud map, we introduce keyframe information labels for the point clouds and use hierarchical storage of the point cloud data. This enables us to dynamically update the pose of each frame in the overall point cloud in real-time, based on the results of both local and global optimization. In addition, redundant point cloud information is removed during this process, optimizing the overall point cloud map for precision and storage capacity.

## IV. Vision Front-End and Data Transmission Processing

### A. VO Front-End

To achieve dense mapping with multiple robots, our system uses RGB-D cameras to perform front-end tasks while optimizing and filtering the data based on camera parameters. The commonly used depth cameras have a depth measurement range of 0.1–25 m and an accuracy of $\pm 1\%$ (typical) or $\pm 2\%$ (maximum) in depth measurement. The depth uncertainty ranges from 1 to 10 mm. In this article, we use the Intel RealSense D435 camera for testing.

In the framework proposed in this article, any VO system based on keyframes and processing RGB-D information can be used to generate a multirobot globally consistent map, while allowing the use of inertial information to improve operational accuracy. The RGB-D front-end module from ORBSLAM3 [30] is used in the testing, and the size of the local map it maintained is restricted to alleviate computational burden.

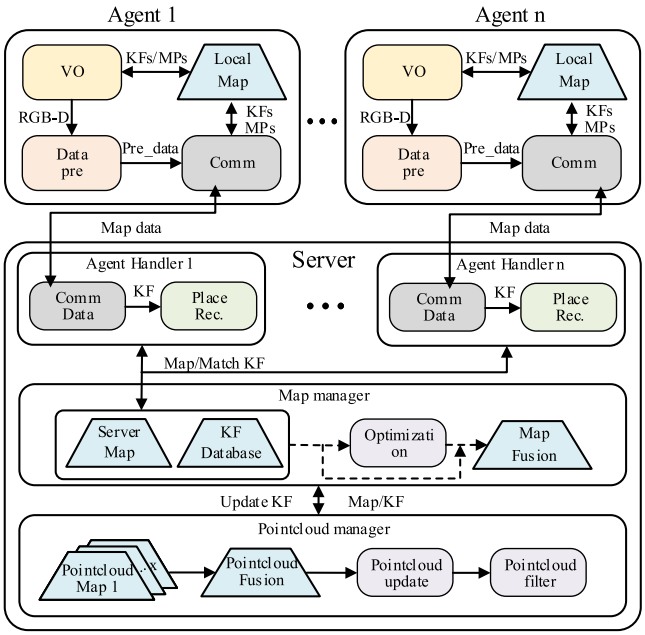



Fig. 1. Overview of the CCMD-SLAM system architecture.

### B. Keyframe-Based Transfer Format Adjustment and Image Sampling

During the CCMD-SLAM data transmission process, the client transmits several types of information to the server, including keyframe pose, feature point position, ORB features, and RGB-D information. The transmission information is stored in 8-bit and 16-bit formats according to the data characteristics instead of the 32-bit format in the original transmission information, which can significantly save. On the other hand, encoding and compressing the transmitted data can avoid the redundant space generated by the serialization process of structured data, which can further reduce the transmission pressure.

RGB-D information typically consists of color and depth images and has the highest bandwidth usage. The color image has a resolution of $(1920, 1080, 3)$ with each pixel having a value between 1 and 255. After format conversion, each pixel is represented by 8 bits and it is resampled to a resolution of $(640, 480, 3)$ to match the depth image. For depth images, the measurement error tends to escalate with increasing distance. Considering the camera error parameters and depth range outlined in Section I, it can be determined that under the existing conditions of a depth camera, the distance error is given by

$$\delta = P_c \times d \tag{1}$$

where 1 mm represents the minimum measurement error.

To minimize data while maintaining precision, any data below 1 mm is deemed unreliable within the transmission framework. In addition, due to the camera depth range being 100–25 000 mm, which is less than the representation range of 16-bit depth data, it is possible to convert all the 32-bit data into 16-bit for transmission, effectively reducing data transfer pressure.

To further reduce data transmission, image downsampling can be performed without compromising the accuracy of dense mapping. The geometric precision of the dense point cloud relies heavily on the camera parameters used for point cloud reconstruction. The main influencing factors can be attributed to the camera's image resolution and the depth accuracy of the depth camera. Specifically, the resolution determines the level of detail captured in the camera image, while accuracy determines the range of measurement errors in depth values. Both the factors collectively affect the precision of the coordinate calculations for each point in the point cloud. Specifically, given a camera resolution of $w \times h$, the range corresponding to each pixel in physical space can be represented as

$$\begin{cases} \Delta x_r = \dfrac{W}{w} = \dfrac{2Z\tan(\theta_x/2)}{w} \\ \Delta y_r = \dfrac{H}{h} = \dfrac{2Z\tan(\theta_y/2)}{h} \end{cases} \tag{2}$$

where $W$ and $H$ represent physical spatial dimensions, while $Z$ denotes the measurement depth. In addition, $\theta_x$ and $\theta_y$ denote the camera's horizontal and vertical field of view, respectively.

On the other hand, the depth accuracy of a depth camera, usually expressed as $\delta z/z$, is typically represented as a percentage, such as 1%. According to the principle of similar triangles, the magnitude of the physical error corresponding to that pixel can be calculated as follows:

$$\begin{cases} \Delta x_d = 2Z\tan\dfrac{\theta_x}{2} \times \dfrac{\delta Z}{Z} \\ \Delta y_d = 2Z\tan\dfrac{\theta_y}{2} \times \dfrac{\delta Z}{Z}. \end{cases} \tag{3}$$

Combining these two factors, the coordinate error of each point in the point cloud can be approximated as

$$\begin{cases} \Delta x = \max(x_d, x_r) \\ \Delta y = \max(y_d, y_r). \end{cases} \tag{4}$$

It is evident that higher resolution and accuracy contribute to the reduction of point cloud errors. Nevertheless, it should be noted that excessively high resolution will eventually reach a saturation point, at which accuracy becomes the determining factor.

In this article, the image downsampling rate is expressed as a proportionality factor $r$ which represents the proportionality between the side dimensions of the new image and the side dimensions of the original image. According to the tested parameters of the RealSense D435 camera, the results are shown in Fig. 2. In consideration of maintaining the geometric precision of the point cloud and for facilitating pixel computations, this article adopts downsampling rates of 0.5 and 0.25 for image processing. This strategy aims to further reduce data transmission. The subsequent experiments will delve into the specific effects of downsampling.

### C. RGB-D Information Filtering Based on Co-View

The framework in this article diverges from the traditional method of model-to-model point cloud fusion. Using front-end VO, it achieves relatively accurate keyframe poses. These poses are further refined through back-end global optimization, enhancing their precision. Consequently, transmitting highly overlapping keyframe point clouds becomes redundant.

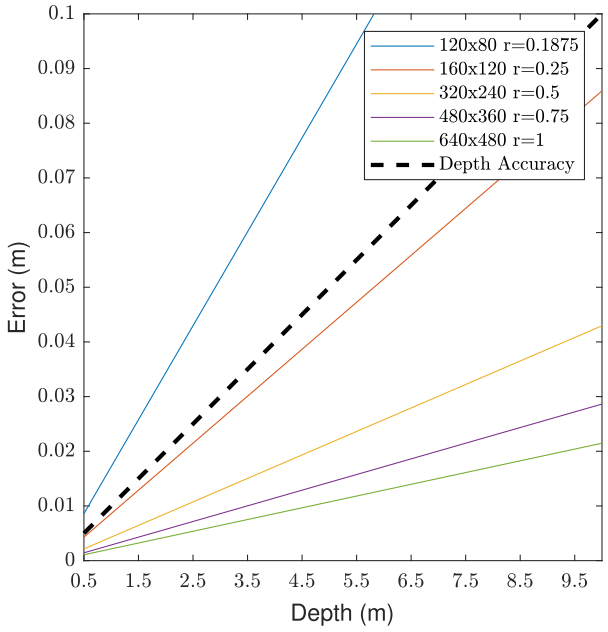



Fig. 2. Relationship between point cloud accuracy, depth accuracy, and resolution.

In ORBSLAM [30], a covisibility-based approach is used to determine keyframes, where a keyframe is inserted when the number of tracked keypoints falls below a certain threshold. The system compares the ORB descriptors of the recognized feature points in the current frame with the ORB descriptors of map points, considering points below a threshold as tracked key points, and establishes a covisibility graph. The similarity between keyframes can be visually demonstrated by counting the number of tracked points between them. Consequently, we propose a filtering method for RGB-D information based on the covisibility graph established by the front-end. This approach allows for consecutive keyframes to be without RGB-D information. Our method aims to reduce redundant RGB-D data transmission by evaluating the shared observation between the current keyframe $\mathrm{KF}_c$ and connected keyframes.

As illustrated in Algorithm 1, the connected keyframes KFs that have co-observation with $\mathrm{KF}_c$ are first extracted from the covisibility graph and sorted by their connection weights weight with $\mathrm{KF}_c$. The weights indicate the number of shared feature points observed by the keyframe pair. After removing KFs that have not been transmitted, each remaining KF is checked whether its weight exceeds the minimum threshold $S_{\min}$. This indicates significant observation overlap with $\mathrm{KF}_c$. Therefore, the RGB-D data of $\mathrm{KF}_c$ could be skipped for transmission. The initial $S_{\min}$ is decrementally updated for each qualified KF to adaptively adjust the filtering threshold. If no connected KF satisfies the $S_{\min}$ criteria, the RGB-D data of $\mathrm{KF}_c$ will be transmitted.

The keyframe filtering thus selectively share RGB-D data based on quantified co-observation. By incrementally updating $S_{\min}$, it avoids sending redundant data while adapting to different levels of visual overlap between keyframes. The approach

provides an efficient RGB-D transmission mechanism for visual SLAM systems.

**Algorithm 1** RGB-D Keyframe Selection

**Require:** Current keyframe $KF_c$, connected keyframes $KFs$, initial threshold $S_{\min}$
**Ensure:** Transmission decision for $KF_c$

```
Extract KFs connected to KF_c from covisibility graph
Sort KFs by shared feature weights weight with KF_c
Remove KFs that have not been transmitted
for each remaining KF in KFs do
    if weight > S_min then
        Decrement S_min adaptively
    end if
end for
if No KF satisfies threshold S_min then
    Return Transmit KF_c
else
    Return Skip transmitting KF_c
end if
```

### D. Transfer of Mapping Data Compression

Adding the converted RGB-D information to the keyframe transfer requires an additional transfer traffic per frame. To further reduce transmission pressure, data compression algorithms are used to reduce image transmission traffic, and image compression is performed on the basis that the time of compression does not affect real-time image construction.

Color image compression algorithms have matured over the years, often relying on image transformation coding techniques to compress and remove redundant data. To better integrate image compression algorithms with SLAM algorithms, general compression algorithms were tested during execution of the VO front-end algorithm selected in Section IV-A to evaluate compression ratios and runtimes. Based on the compression ratios and times [31], JPEG compression is chosen for processing.

For depth images, common image transform coding methods such as discrete cosine transform (DCT) and discrete wavelet transform (DWT) cannot achieve satisfactory results due to the high discreteness of pixel values. In addition, since the numeric values directly affect the accuracy of dense mapping, a near-lossless compression method is required. The system uses run-length encoding to compress depth images, ensuring data precision is preserved during the compression process. Run-length encoding can be represented as

$$\mathrm{RLE}(s) = \sum_{i=1}^{n} c_i r_i \tag{5}$$

where $\mathrm{RLE}(s)$ denotes the tour code of the string $s$, $n$ is the number of tours, $c_i$ is the character of the $i$th tour, and $r_i$ is the number of repetitions of the $i$th tour.

Run-length encoding achieves lossless compression by storing repeated fields in the data efficiently, and it enables fast decompression for data restoration. To further enhance the efficiency of run-length encoding, we have designed a depth image quantizer tailored to the characteristics of the RealSense D435 camera

$$\begin{cases} d = 0, & (d \leqslant 200) \\ d = d - 200, & (200 < d \leqslant 3000) \\ d = 2800 + \dfrac{d - 3000}{2}, & (3000 < d \leqslant 6000) \\ d = 4300 + \dfrac{d - 1500}{4}, & (6000 < d \leqslant 12000) \\ d = 5800 + \dfrac{d - 12000}{8}, & (12000 < d \leqslant 24000) \\ d = 0, & (24000 < d) \end{cases} \tag{6}$$

where $d$ is the depth image pixel depth in millimeters.

The quantizer divides the depth range into six parts based on the camera properties and sets values outside the actual range to 0. The camera's ideal range, from 0.3 to 3 m, remains unchanged. As indicated by Formula (1), considering that depth image errors are distance-dependent, the range outside the ideal bounds is proportionally divided into three parts, each compressed separately. Quantizing the depth image allows for maximum compression without compromising data accuracy. Finally, the compressed data are binary encoded using the LZ77 algorithm [32], completing the compression of the depth image.

## V. Multimap and Point Cloud Management

### A. Multimap Management

After receiving the information of each robot, the map manager in the back-end server will initialize a new map, decompress, and store the keyframe data. The map manager will designate the first mapped map as the primary map for dense mapping and allow other maps to be fused. The system keeps running loop closure detection, matching and identifying each keyframe using the bag-of-words method [33]. Once a loop closure is detected, a 3-D–2-D RANSAC is used to calculate the relative pose $T$ between matched keyframes, and PGO is used to optimize the poses. When a loop closure occurs between two robots, their maps are fused using the relative transform $T$, and a new map is built to replace the two original maps. Only after the primary map finishes map fusion, the fused map keyframes are allowed to generate point clouds for further dense reconstruction.

### B. Point Cloud Mapping

The point cloud generation module runs attached to the primary map, fusing allowed keyframes based on the optimized poses to build a dense map. For mainstream RGB-D cameras, the depth image pixel value represents the distance in the $z$-axis direction in the camera coordinate system rather than the actual distance to the camera's optical center. Let $P = (X, Y, Z)^{\mathrm{T}}$ be a point in space under the camera coordinate system and $p$ be the coordinate of the point under the pixel coordinate system. The relationship between the two coordinate systems is

$$Zp = Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = KP \tag{7}$$

where $u$ and $v$ are the coordinates under the pixel coordinate system, $K$ is the camera internal reference matrix, and $f_x$, $f_y$, $c_x$, and $c_y$ are the camera internal references.

Then the coordinates in the actual space can be obtained as

$$\begin{aligned} X &= Z(u - c_x)/f_x \\ Y &= Z\big(v - c_y\big)/f_y \\ Z &= d. \end{aligned} \tag{8}$$

The RGB-D image can be used to convert each pixel into the corresponding 3-D point coordinates by applying (8).

For each keyframe, we define the rotation and translation of the spatial point $P_{\text{kf}}$ relative to the camera coordinate system as $R_{\text{kf}}^c$ and $T_{\text{kf}}^c$, respectively. In addition, we define the rotation and translation from the camera coordinate system to the world coordinate system as $R_c^w$ and $T_c^w$, respectively. Consequently, the coordinates $P_w$ of the point in the world coordinate system can be calculated as follows:

$$P_w = R_c^w * \left(R_{\text{kf}}^c * P_{\text{kf}} + T_{\text{kf}}^c\right) + T_c^w. \tag{9}$$

The keyframe point clouds from robots are fused into a global point cloud using (9). Since there are many duplicate parts across the keyframe point clouds, the fusion will generate a large number of redundant points. Therefore, voxel filtering is used to filter the point cloud. The global dense reconstruction is finished by voxel filtering the point cloud using the $(x, y, z)$ coordinates and keeping the maximum labeled data in each voxel.

### C. Point Cloud Management

The dense map manager runs on the map manager and maintains point cloud data for all the robots. The framework of the dense map manager is shown in Fig. 3. Each time a new map is fused into the primary map, a new point cloud collection is created, enabling individual modification and reading of point cloud data for each robot. Each point cloud set can undergo a global pose transformation during the map fusion process to ensure global consistency. The transformed point cloud sets can then be merged into a global point cloud through point cloud registration.

In point cloud operations, the most computationally intensive steps are point cloud filtering and searching for a specific frame of point cloud. These two steps have a computational time proportional to the number of point clouds being processed. To reduce the computational resources consumed by updating and stitching point clouds, a tree index is used to divide the global point cloud into point clouds maintained separately for each robot. And each robot point cloud is divided into local maintenance point clouds by different map optimization and update methods. The filtering procedure uses a sliding window based on keyframe index. At each incremental update or addition of a keyframe, the point cloud within the window undergoes filtering to mitigate the computational burden introduced by the filtering process. The point clouds are stored in the format $P = (X, Y, Z, R, G, B, N)^{\text{T}}$, where the label $N$ is combined with the index of each level in the tree diagram, distinguishing the point cloud of each keyframe for efficient retrieval and updating.

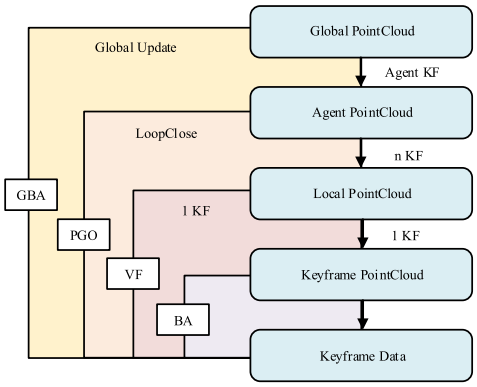



Fig. 3. Tree structure dense map manager. BA is bundle adjustment, PGO is pose graph optimization, and VF is voxel filtering.

In multirobot dense mapping tasks, the map accumulates errors from multiple robots as the mapping area expands. To maintain the accuracy of the dense map, real-time optimization and updates are required. The keyframe-based construction method makes it easy and efficient to optimize the point cloud using information from keyframes. Point cloud updates can be categorized into three types based on the optimization method used and the content being updated.

*1) Front-End VO Local BA Optimization:* Front-end VO local BA optimization updates single-frame point cloud. BA optimization updates are delivered separately via data transfer and are performed in real-time to update the keyframe point cloud. For updating a single-frame point cloud, the spatial point $P_{\text{kf}}$ that updates the point cloud is retrieved from the local point cloud $P$ using its $N$ label, as explained in Section V-C. After deleting the frame point cloud from the local point cloud, the updated point cloud is added to the local point cloud and merged with adjacent point clouds. Finally, voxel filtering is used to remove redundant spatial points.

*2) Back-End PGO:* When two keyframes are detected as loop closures, PGO optimization is performed on the visible edges of the loop closure interval and the keyframe poses and local point clouds are updated. Let $P_{\text{ref}}$ be the keyframes detected as loops. $P_{\text{ref}}$ is first deleted from the global point cloud. The updated multiframe point cloud and adjacent partial clouds are then merged to rebuild the global cloud, which is finally filtered.

*3) Back-End GBA Optimization:* Unlike COVINS, an additional global bundle adjustment (GBA) strategy is implemented for RGB-D front-end VO in the absence of inertial information. The global BA minimizes the reprojection error of all the KFs and MPs using G2O's Levenberg–Marquardt algorithm to improve the accuracy of the overall map and global point cloud. For the global optimization of the overall point cloud update, we determine the keyframes that have been changed after the global optimization according to the update result and reconstruct each robot point cloud $P_i$ that has changed. The updated point cloud of all the robots is integrated into a new global point cloud $P_{\text{new}}$.

In summary, our algorithm integrates BA optimization for front-end robots with multirobot PGO optimization and multirobot global BA optimization for updating maps and global point clouds. This aims to effectively reduce cumulative

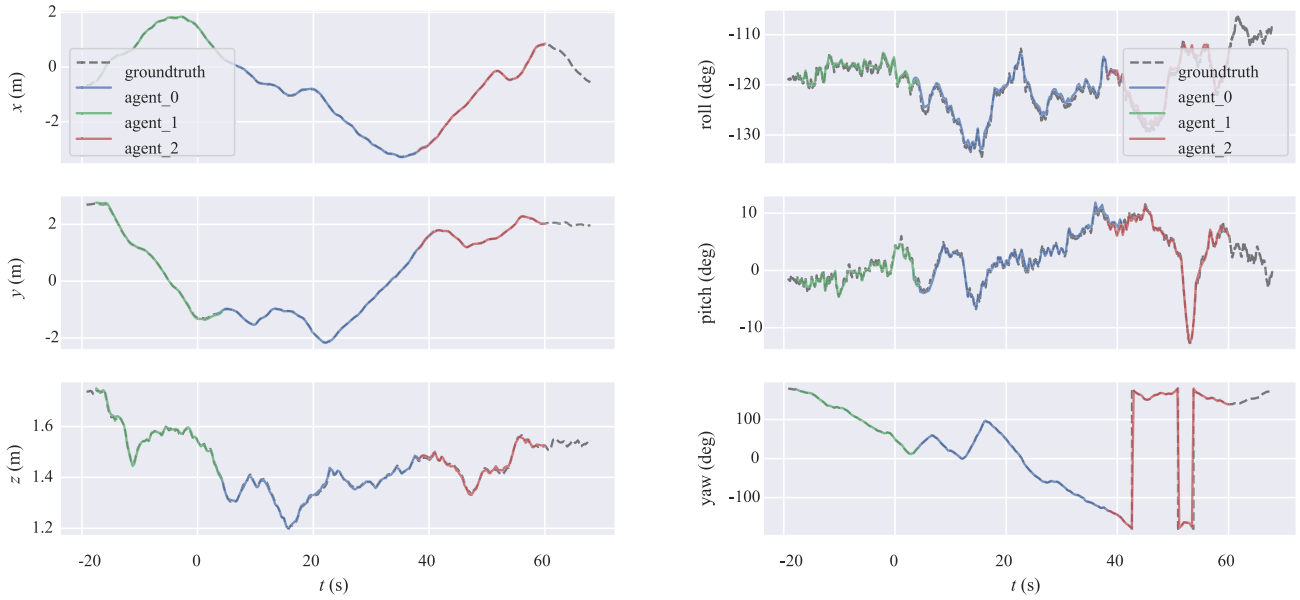



Fig. 4. Comparison results of three client trajectory fusion with real data. $x$-, $y$-, $z$-axis translation and roll, pitch, and yaw angle estimated results.

errors during the mapping process and improve map accuracy. In addition, the modular point cloud update strategy is designed to efficiently conserve computational resources, based on optimization types and scopes.

## VI. Evaluation

The experimental platform for this chapter is AMD Ryzen 7 4800H (eight cores at 2.90 GHz) with 16 GB of RAM. In this section, to validate the performance of CCMD-SLAM, we use the RGB-D benchmark dataset [34] (TUM dataset) from the Technical University of Munich for evaluation by extensive experiments. We use the partitioned dataset testing method [35], splitting the TUM datasets into multiple segments with 10% overlap. Multiple clients are simultaneously run to simulate multirobot front-end mapping, with data transmitted over the network to another server for collaborative global map building and real-time dense mapping and maintenance. The effects of multirobot mapping, dense mapping, and data transfer efficiency are tested and compared through simulations with single and multiple clients, respectively.

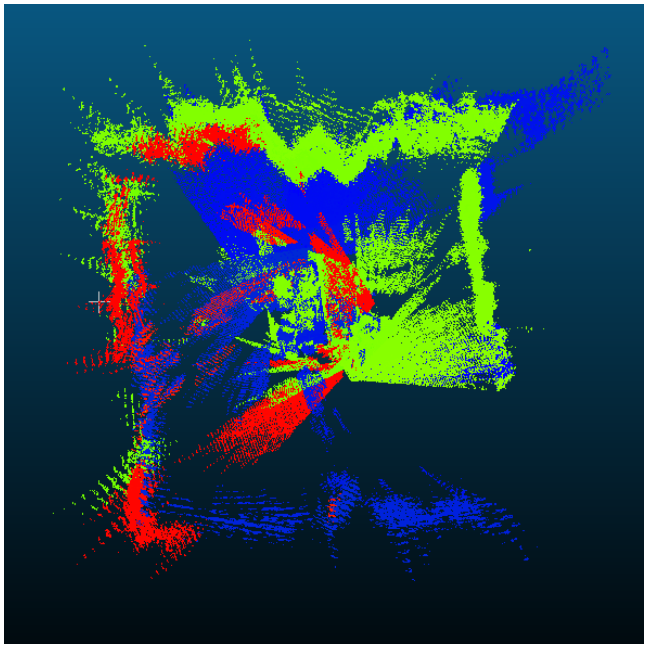


Fig. 5. Merged reconstruction of local point clouds maintained by multiple clients on CCMD-SLAM.

### A. Multirobot Mapping Evaluation

We first conduct performance tests on the f2/desk dataset using three clients to evaluate the performance of multirobot mapping and trajectory fusion. The errors between the fused trajectories of the three clients and the real acquisition trajectories are shown in Fig. 4. The plot reveals that the fused trajectories of multirobot mapping exhibit minimal deviation from the ground-truth trajectories. Furthermore, the fused sections display a smooth connection with no significant introduction of errors. The dense mapping result on the TUM f2/desk dataset is shown in Fig. 5 which reflects the local point clouds maintained by multiple clients. And the reconstruction results on the TUM dataset and the ICL-NUIM dataset are shown in Fig. 6. Next, ten measurements were evaluated using the evo tool [36], and the absolute trajectory error root mean square error (RMSE) is used to assess the performance of our system. The results of the tests are presented in Table I.

The experimental results show that our system can effectively correct the trajectory drift, and the RMSEs on the f1/desk, f2/$xyz$, and f3/office datasets reach 0.020, 0.004, and 0.015 m. Regarding the mapping results, our RMSEs on all the three datasets exceed those of the existing dense reconstruction RGB-D SLAM algorithms. This improvement in mapping accuracy is attributed to the establishment of numerous precise constraints among robots through multirobot PGO and GBA optimization processes. Also during the

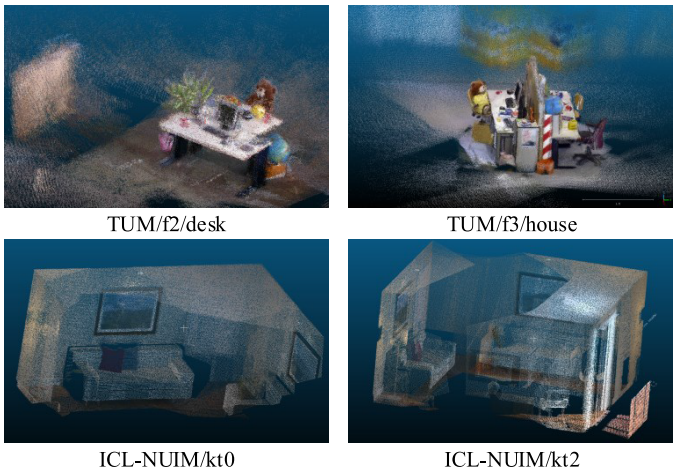



Fig. 6. Reconstruction results on the TUM dataset and the ICL-NUIM dataset.

experiment, our system is able to run on datasets where the existing RGB-D SLAM algorithms do not work properly, such as f2/large_with_loop and f2/360_hemisphere.

### B. Evaluation of the Effect of Dense Reconstruction

To compare the quality of scene reconstruction, our system is used to reconstruct scenes in several datasets from TUM. Real ground data are also used to construct actual reconstructed point clouds, and the two point cloud models were compared and analyzed. To qualitatively analyze and compare the quality of point cloud, reconstruction errors were calculated using the Hausdorff distance and average distance, with the aid of CloudCompare software. The Hausdorff distance can be expressed as

$$H(A, B) = \max(h(A, B), h(B, A)). \tag{10}$$

After aligning the centers of the reconstructed point clouds with the ICP algorithm, we calculated the Hausdorff distance and average distance between the reconstructed point cloud produced by our method and the actual data reconstructed point cloud. The results of the calculations are presented in Table II.

During the scene reconstruction process, a point cloud reconstruction effect with centimeter-level accuracy is achieved. The overall Hausdorff distance ranged between 0.01 and 0.1 m, while the average error is within 0.001–0.01 m. Compared with the keyframe-based reconstruction algorithm [35], the error of 0.44 m on f1/desk has been significantly improved.

For datasets with relatively small scenes, such as f1/desk and f2/$xyz$, transmitting images with a sampling rate of 2 can yield more detailed reconstructed scenes and improve the reconstruction effect. Therefore, the appropriate sampling rate can be chosen based on the size of the scene.

We compared the reconstruction accuracy on the ICL-NUIM dataset by measuring the average distance between the reconstructed point clouds and real models, along with several other RGB-D methods, as shown in Table III. Two agents were deployed to autonomously process segmented datasets for scene reconstruction. Due to the limited number of keyframes in the kt1 dataset, comparative experiments were conducted on the other three datasets. In terms of average performance, our algorithm achieved a reconstruction average error of 0.93 cm, surpassing other algorithms in average accuracy. In the reconstruction of some small scenes, it exhibited slight errors compared with ElasticFusion [16]. A distinguishing feature of our algorithm is its exclusive reliance on CPU computation throughout, conserving computational resources and rendering it particularly suitable for multirobot systems.

TABLE I
COMPARISON OF STATISTICAL RESULTS OF ABSOLUTE TRAJECTORY ERROR RMSE (UNIT: m)

| | f1/desk | f2/xyz | f3/office |
|---|---|---|---|
| CCMD-SLAM(our) | 0.020 | 0.004 | 0.015 |
| DVO-SLAM | 0.022 | 0.014 | 0.035 |
| Kintinuous | 0.142 | 0.032 | 0.064 |
| ElasticFusion | 0.022 | 0.009 | 0.017 |
| DI-Fusion | 0.044 | 0.023 | 0.156 |
| BAD-SLAM | 0.017 | 0.011 | 0.017 |
| NICE-SLAM | 0.027 | 0.018 | 0.030 |

TABLE II
COMPARISON RESULTS OF RECONSTRUCTED ACCURACY ON TUM DATASETS WITH REAL DATA

| Dataset | Agent Num | Hausdorff Dist. (m) | Avg Dist. (m) |
|---|---|---|---|
| f1/desk | 2 | 0.0297403 | 0.0060615 |
| f2/xyz | 3 | 0.0411498 | 0.0006829 |
| f2/360 | 2 | 0.0889183 | 0.0307399 |
| f2/desk | 3 | 0.0694975 | 0.0068381 |
| f3/house | 3 | 0.0691603 | 0.0016194 |

TABLE III
COMPARISON RESULTS OF RECONSTRUCTION ACCURACY IN THE ICL-NUIM DATASET (UNIT: cm)

| | kt0 | kt2 | kt3 | avg. |
|---|---|---|---|---|
| ours | 0.9 | 0.7 | 1.2 | 0.93 |
| DVO SLAM | 3.2 | 11.9 | 5.3 | 6.80 |
| Kintinuous | 1.1 | 0.9 | 15.0 | 5.77 |
| ElasticFusion | 0.7 | 0.8 | 2.8 | 1.43 |

### C. Data Transfer Effect Evaluation

To evaluate the data transmission effect of our system, we build and collect transmission data on the above dataset using single and multiple robots, respectively, as shown in Table IV. As can be seen from the table, the complexity and scale of the scene increase, and the volume of transmitted data and the number of keyframes also increase. Depending on the complexity and spatial size of the captured scenes in the datasets, the algorithm's data transmission ranges from 15 to 350 kB/s. Furthermore, the transfer rate of keyframes ranges from 0.3 to 5.5 Hz. It is worth noting that common WiFi facilities are capable of easily achieving the desired transmission goals. In addition, the number of proxies and the complexity of the scene do not affect the data volume of individual proxies. Therefore, the total network traffic caused by proxy-to-server data transmission is directly proportional to the number of participating proxies. After data processing and

TABLE IV

COMPARISON OF AVERAGE TRAFFIC PER SECOND (ATPS), AVERAGE KEYFRAME RATE (AKFR), AND AVERAGE KEYFRAME SIZE (AKFS) OF MULTIPLE ROBOTS FOR MAP BUILDING

| Dataset | ATPS (KB) | AKFS (KB) | AKFR (Hz) |
|---|---|---|---|
| f1/desk | 263.479627 | 59.71606061 | 4.412207106 |
| f2/xyz | 20.70009428 | 57.44616438 | 0.360339015 |
| f2/360 | 345.2537847 | 62.373706 | 5.535245648 |
| f2/desk | 99.33134974 | 58.75648 | 1.690559913 |
| f3/house | 127.993085 | 55.58647975 | 2.302593825 |

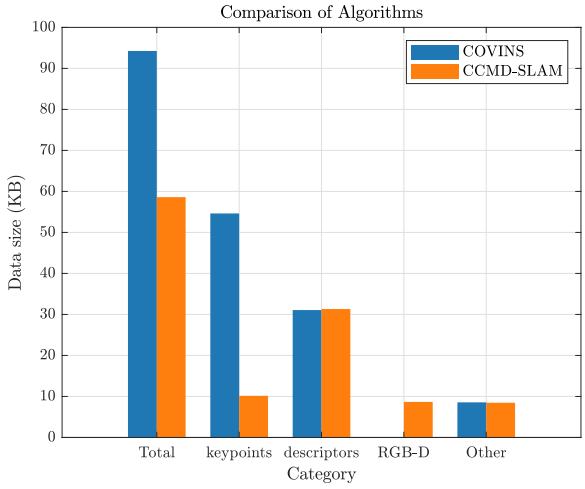



Fig. 7. Comparison of data transmission between CCMD-SLAM and COVINS.

compression, the average size of keyframes reached 58.78 kB. Next, we conducted tests on the transmission of individual keyframes.

The keyframe-based approach to centralized multirobot mapping requires passing keypoint data, feature descriptors, and other parameters to the back-end server. At the same time, to implement dense mapping, additional RGB-D information needs to be passed to the back-end. We compare the transmission of each part of the CCMD-SLAM and COVINS algorithms as shown in Fig. 7. As illustrated in the figure, CCMD-SLAM achieves a reduction of key point transmission data volume to 18.6% after data processing and compression. In addition, the RGB-D information, which is initially 2.05 MB/frame, is reduced to an average of 8.67 kB/frame, corresponding to a volume of only 0.4% of the unprocessed information. Overall, our algorithm achieves a 38.2% reduction in the overall data transmission volume compared to COVINS, while additionally implementing dense graph construction.

### *D. Module Ablation Experiments*

Subsequently, we conducted ablation experiments on the designed module algorithm using the TUM dataset and evaluated the effects of our parameter choices and algorithm. First, we tested the effects of downsampling and keyframe selection, setting the global point cloud voxel size to 5 cm. The results of mapping and transmission under different parameters are as shown in Table V

It can be observed from Table V that using images with different sampling rates for transmission significantly impacts

TABLE V

EFFECTS OF DOWNSAMPLING AND KEYFRAME SELECTION ON MAPPING AND TRANSMISSION

| KF Select | Sample Rate | RGB-D Size(kb) | Point Num |
|---|---|---|---|
| 0 | 1 | 58.0348 | 197611 |
| 1 | 1 | 33.6546 | 148917 |
| 0 | 0.5 | 21.1737 | 164191 |
| 1 | 0.5 | 12.4878 | 106530 |
| 0 | 0.25 | 7.9046 | 111125 |
| 1 | 0.25 | 4.3261 | 76584 |

TABLE VI

DEPTH COMPRESS TEST RESULT. (DS REFERS TO IMAGE DOWNSAMPLING, QT REFERS TO IMAGE QUANTIZATION, AND RLE REFERS TO RUN-LENGTH ENCODING)

| Method | Size(Bite) | Time(ms) |
|---|---|---|
| DS | 153600 | 0.546 |
| DS+RLE | 46084 | 0.784 |
| DS+RLE+LZ77 | 31630 | 1.048 |
| DS+QT+RLE | 7252 | 0.95 |
| DS+QT+RLE+LZ77 | 5359 | 1.286 |

the size of transmitted RGB-D information. Reducing the size of the image edge length by half results in a quarter reduction in pixel count. However, due to the overlapping perspectives of keyframes and partial perspectives from multiple robots, the average decrease in the constructed point cloud count is 28.3%. Therefore, within permissible limits, decreasing the sampling rate enhances transmission efficiency. The highest transmission efficiency is achieved when the sampling rate $r = 0.25$. In addition, keyframe selection strategies can further reduce the transmission. After applying different sampling rates and filtering strategies, the transmitted RGB-D information can be reduced to 7.45% of the original. On the other hand, with the reduction of transmission pressure, the point cloud becomes sparse. In extreme cases, the point cloud quantity is 38.75% of the original image, indicating an increasing efficiency in transmission. In practical operation, appropriate downsampling parameters can be selected based on the desired point cloud density.

The test results of depth compression are shown in Table VI, where we conducted tests using various combinations of algorithms in our module. It can be observed that using a quantizer for run-length encoding of depth images yields better results, and the quantizer incurs an additional time consumption of only 0.166 ms. Furthermore, binary encoding can further compress data on the basis of run-length encoding, achieving maximum compression effects. The overall time consumption of 1.286 ms is acceptable compared with the fastest keyframe transmission rate of 5.5 Hz.

Finally, we compared our strategy for updating dense point clouds with the original method of updating through reconstructing point clouds, as shown in Fig. 8. It can be observed that using the strategy of reconstructing point clouds for updates requires a significant amount of computational time, which continuously increases with the growth of point cloud quantity. Our algorithm, using point cloud indexing and grouped storage, can swiftly locate and modify keyframe point clouds needing updates during the update process, consistently within a computation time of 0.4 ms or less.

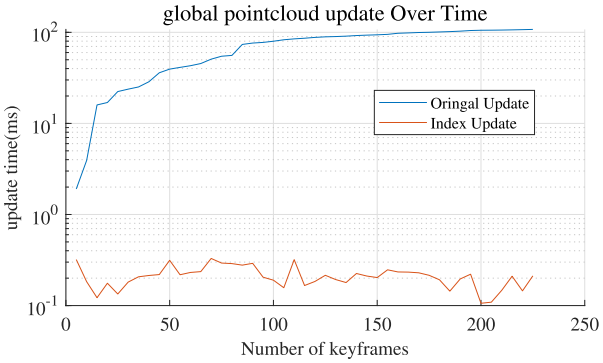



Fig. 8. Point cloud update strategy consumption comparison.

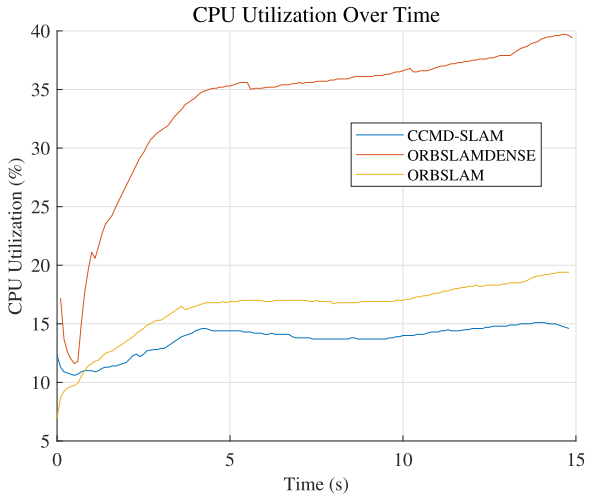



Fig. 9. CPU usage of three different algorithms: CCMD-SLAM, the general ORB-SLAM3, and the ORB-SLAM algorithm with direct dense mapping.

### E. Runtime Performance Analysis

To validate that our algorithm alleviates the computational burden on the robot's front end, we conducted tests on the CPU usage of the front-end robot. Furthermore, we compared the CPU usage of our algorithm with that of other algorithms executed on the same computer using the method outlined in the article [17]. Our system was compared with the standard ORB-SLAM3 and ORB-SLAM algorithms using direct dense mapping on the f2/desk dataset. The experimental results are depicted in Fig. 9. From the graph, it can be observed that using the approach of direct mapping in the robot front-end leads to a continuous increase in CPU usage as the mapped area expands. In contrast, our system adopts a centralized mapping approach where the front-end only needs to maintain a limited map, resulting in more stable CPU usage during runtime. Furthermore, as shown in the graph, even when running the data transmission thread, CCMD-SLAM exhibits lower CPU usage compared with the ORB-SLAM algorithm. Moreover, by offloading the dense mapping task to the back-end server, our algorithm reduces the front-end's CPU usage for dense mapping by 61.28%.

During testing, we conducted performance analysis of the front-end and back-end modules to evaluate the algorithm's real-time capability. Specifically, the average time consumption for data preprocessing and transmission at the front-end was 7.0 and 6.1 ms, respectively. The back-end algorithm required 3.3 ms for data processing and 5.7 ms for dense reconstruction. As illustrated in Table IV, the minimum time interval for keyframe processing was 180 ms. Within each keyframe interval, real-time reconstruction and updating of the back-end dense map are fully achievable. Simultaneously, during system optimization updates to keyframe poses, only transmitting keyframe indices and pose transformation matrices between the front- and back-ends is sufficient to ensure real-time updates of the map. In addition, since the modules operate in parallel, the inclusion of data processing stages does not impact keyframe creation.

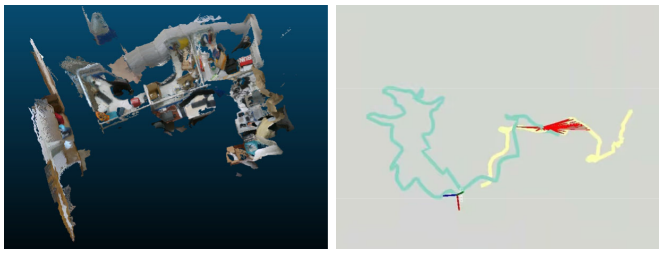


Fig. 10. CCMD-SLAM real-world multirobot dense reconstruction result (left image shows the dense reconstruction of the office, and the right image depicts the reconstructed trajectory after loop closure fusion).

### F. Real-World Experimental Testing

We conducted tests of our algorithm in a real-world scenario. We used a desktop computer as the back-end server and two laptops as front-end agents. Data transmission occurred through a local area network (LAN). Each laptop was connected to a RealSense D435 camera for image acquisition. We performed multirobot dense reconstruction tests in an office environment, and the reconstruction results are illustrated in Fig. 10.

In the reconstructed scene, loop closures of the two agents were rapidly identified, establishing a globally consistent map. Simultaneously, various objects within the office were well-reconstructed without experiencing shape distortions. In the testing phase, due to the high complexity of the scene, the front-end transmitted an average of 2.94 frame/s, with an average transfer rate of 381 kB/s. The maximum transfer rate reached 1.27 MB/s, and the server accumulated 51.1 MB of data. In terms of computational load, the back-end server experienced a peak CPU usage of 16.37%, while each front-end agent only used a maximum of 7.74%. According to the test results, it can be concluded that ordinary WiFi load and the computational resources of the robots are sufficient to meet the algorithm's requirements. However, real-world testing also revealed some issues, including a narrow camera field of view and suboptimal tracking performance under severe shaking, which will be addressed in subsequent research.

## VII. Conclusion and Future Work

In this article, we propose CCMD-SLAM, a multirobot SLAM that allows efficient data transfer and dense mapping and real-time point cloud maintenance in the back-end.

We establish a complete dense map based on keyframe information by constructing local maps using multiple RGB-D front-ends and transmitting them to the back-end. To reduce data transmission, preprocessing and compression of keyframe data and RGB-D images are performed, and RGB-D image transmission is filtered by a co-viewing graph, resulting in a 38.2% reduction in data transmission flow. Furthermore, we propose a dense map management strategy to establish and real-time maintain a global dense point cloud, using RGB-D information collected by multiple robots. During the mapping process, different strategies can be used to quickly update the point cloud map according to various map optimization methods, leading to a more precise dense mapping result. Finally, through experiments on standard datasets and real-world testing, we have verified the effectiveness of the algorithm in reducing communication burden. In addition, we have successfully achieved multirobot dense mapping while ensuring rapid updates of the dense map.

In future work, we plan to further investigate the structural characteristics of the point cloud, optimize the point cloud reconstruction effect and establish an even more accurate dense map. These efforts will enhance and refine our proposed method.

## REFERENCES


[1] H. Xie et al., "Semi-direct multimap SLAM system for real-time sparse 3-D map reconstruction," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[2] J. Yuan, S. Zhu, K. Tang, and Q. Sun, "ORB-TEDM: An RGB-D SLAM approach fusing ORB triangulation estimates and depth measurements," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–15, 2022.

[3] C. Fan, J. Hou, and L. Yu, "Large-scale dense mapping system based on visual-inertial odometry and densely connected U-Net," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–16, 2023.

[4] S. Dong et al., "Multi-robot collaborative dense scene reconstruction," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–16, Aug. 2019.

[5] G. He, Q. Zhang, and Y. Zhuang, "Online semantic-assisted topological map building with LiDAR in large-scale outdoor environments: Toward robust place recognition," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022.

[6] J. Yin, D. Luo, F. Yan, and Y. Zhuang, "A novel LiDAR-assisted monocular visual SLAM framework for mobile robots in outdoor environments," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–11, 2022.

[7] Y. Pan, X. Xu, X. Ding, S. Huang, Y. Wang, and R. Xiong, "GEM: Online globally consistent dense elevation mapping for unstructured terrain," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.

[8] L. Riazuelo, J. Civera, and J. M. M. Montiel, "C2TAM: A cloud framework for cooperative tracking and mapping," *Robot. Auto. Syst.*, vol. 62, no. 4, pp. 401–413, Apr. 2014.

[9] G. Loianno et al., "A swarm of flying smartphones," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 1681–1688.

[10] D. Zou and P. Tan, "CoSLAM: Collaborative visual SLAM in dynamic environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 354–366, Feb. 2012.

[11] J. G. Morrison, D. Gálvez-López, and G. Sibley, "MOARSLAM: Multiple operator augmented RSLAM," in *Proc. 12th Int. Symp. Distrib. Auton. Robot. Syst.* Japan: Springer, 2016, pp. 119–132.

[12] M. Karrer, P. Schmuck, and M. Chli, "CVI-SLAM—Collaborative visual-inertial SLAM," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2762–2769, Oct. 2018.

[13] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli, "COVINS: Visual-inertial SLAM for centralized collaboration," in *Proc. IEEE Int. Symp. Mixed Augmented Reality Adjunct (ISMAR-Adjunct)*, Oct. 2021, pp. 171–176.

[14] R. A. Newcombe et al., "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 127–136.

[15] S. Lin, J. Wang, M. Xu, H. Zhao, and Z. Chen, "Contour-SLAM: A robust object-level SLAM based on contour alignment," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–12, 2023.

[16] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Proc. 11th Robot., Sci. Syst.*, Rome, Italy, Jul. 2015, pp. 1–9.

[17] X. Liu, W. Ye, C. Tian, Z. Cui, H. Bao, and G. Zhang, "CoxGraph: Multi-robot collaborative, globally consistent, online dense reconstruction system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 8722–8728.

[18] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2022–2038, Aug. 2022.

[19] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *J. Field Robot.*, vol. 36, no. 4, pp. 763–781, 2019.

[20] D. Zhu, G. Xu, X. Wang, X. Liu, and D. Tian, "PairCon-SLAM: Distributed, online, and real-time RGBD-SLAM in large scenarios," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–14, 2021.

[21] N. Prieto-Fernández, S. Fernández-Blanco, Á. Fernández-Blanco, J. A. Benítez-Andrades, F. Carro-De-Lorenzo, and C. Benavides, "Weighted conformal LiDAR-mapping for structured SLAM," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–10, 2023.

[22] Z. Pan, J. Hou, and L. Yu, "Optimization RGB-D 3-D reconstruction algorithm based on dynamic SLAM," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[23] C. Brand, M. J. Schuster, H. Hirschmüller, and M. Suppa, "Submap matching for stereo-vision based indoor/outdoor SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 5670–5677.

[24] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration," *ACM Trans. Graph.*, vol. 36, no. 4, p. 1, Aug. 2017.

[25] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[26] J. Niu, Q. Hu, Y. Niu, T. Zhang, and S. Kumar Jha, "Real-time dense reconstruction of indoor scene," *Comput., Mater. Continua*, vol. 68, no. 3, pp. 3713–3724, 2021.

[27] S. Wen et al., "Dynamic SLAM: A visual SLAM in outdoor dynamic scenes," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, 2023.

[28] X. Wang, C. Olston, A. D. Sarma, and R. Burns, "CoScan: Cooperative scan sharing in the cloud," in *Proc. 2nd ACM Symp. Cloud Comput.*, Oct. 2011, pp. 1–12.

[29] L. Bartolomei, M. Karrer, and M. Chli, "Multi-robot coordination with agent-server architecture for autonomous navigation in partially unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 1516–1522.

[30] 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 Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[31] B. A. Lungisani, C. K. Lebekwe, A. M. Zungeru, and A. Yahya, "Image compression techniques in wireless sensor networks: A survey and comparison," *IEEE Access*, vol. 10, pp. 82511–82530, 2022.

[32] Y. Collet, "Zstandard—Real-time data compression algorithm," Facebook, 2023. [Online]. Available: https://github.com/facebook/zstd

[33] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

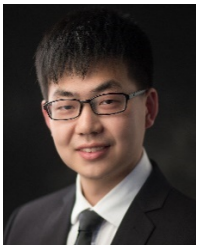
[34] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

[35] T. Ma, T. Zhang, and S. Li, "Multi-robot collaborative slam and scene reconstruction based on RGB-D camera," in *Proc. Chin. Autom. Congr. (CAC)*, May 2020, pp. 139–144.

[36] Michael Grupp. (2017). *EVO: Python Package for the Evaluation of Odometry and SLAM*. [Online]. Available: https://github.com/MichaelGrupp/evo

**Chenle Zuo** received the bachelor's and master's degrees in mechanical engineering from the School of Power and Mechanical Engineering, Wuhan University, Wuhan, China, in 2020 and 2022, where he is currently pursuing the Ph.D. degree in mechanical engineering.

His latest research interests include dense mapping and robot perception.



**Zhao Feng** received the B.S. and Ph.D. degrees in mechanical engineering from the School of Power and Mechanical Engineering, Wuhan University, Wuhan, China, in 2014 and 2020, respectively.

From 2019 to 2020, he was a Joint Ph.D. Student at the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. From October 2020 to October 2022, he was a Research Fellow sponsored by the UM Macao Postdoctoral Associateship (UMPA) at the Department of Electrical and Computer Engineering, Faculty of Science and Technology, University of Macau, Macao, China. Since November 2022, he has been an Associate Professor with the School of Power and Mechanical Engineering, Wuhan University. His current research interests include mechanical design and precision control of piezoelectric nanopositioning systems, robot learning, and control.

**Xiaohui Xiao** (Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Wuhan University, Wuhan, China, in 1991 and 1998, respectively, and the Ph.D. degree in mechanical engineering from Huazhong University of Science and Technology, Wuhan, in 2005.

She joined Wuhan University, in 1998, where she is currently a Full Professor with the Mechanical Engineering Department, School of Power and Mechanical Engineering. She has published more than 30 articles in the areas of mobile robots, dynamics and control, sensors, and signal procession. Her current research interests include mobile robotics, robot learning, high-precision positioning control, and signal processing.