Measurement Science
and Technology

PAPER

CMDS-SLAM: real-time efficient centralized multi-robot dense surfel SLAM

To cite this article: Chenle Zuo *et al* 2024 *Meas. Sci. Technol.* **35** 116303

IOP Publishing Measurement Science and Technology
Meas. Sci. Technol. **35** (2024) 116303 (11pp) https://doi.org/10.1088/1361-6501/ad64fa

CMDS-SLAM: real-time efficient centralized multi-robot dense surfel SLAM

Chenle Zuo, Zhao Feng* and Xiaohui Xiao*

School of Power and Mechanical Engineering, Wuhan University, Wuhan 430072, People's Republic of China

E-mail: fengzhao@whu.edu.cn, xhxiao@whu.edu.cn and zcl0008@whu.edu.cn

Received 17 April 2024, revised 13 July 2024
Accepted for publication 18 July 2024
Published 6 August 2024

Abstract

Real-time dense mapping technology for multi-robot systems is crucial in scenarios like search and rescue. This paper presents CMDS-SLAM, a centralized multi-robot dense surfel SLAM system aimed at overcoming limitations in hardware constraints, data transmission, and real-time creation and updating of dense maps in multi-robot SLAM. CMDS-SLAM reduces the transmission of dense information by employing a dense information filtering mechanism based on co-visual keyframes, in conjunction with the extraction and compression of superpixels. Additionally, the method employs a three-stage superpixel segmentation approach to optimize transmission and enhance the efficiency of surfel map generation. Finally, a surfel co-visibility graph is established, and multi-robot surfel map maintenance and updates are achieved through co-visibility graph and map optimization. A comprehensive evaluation of CMDS-SLAM indicates that the method enables multi-robot surfel mapping and significantly alleviates data transmission pressures while achieving real-time updates and maintenance of the surfel map.

Keywords: SLAM, RGB-D, multi-robot, dense mapping, surfel

1. Introduction

With the advancement of Simultaneous Localization and Mapping (SLAM) technology and the growing use of robotic swarm applications, multi-robot real-time dense reconstruction is becoming more popular in different fields such as environmental mapping, search and rescue, and augmented and virtual reality [1]. These algorithms utilize a centralized architecture where several robots perform their individual tasks and send gathered data to a central server located in the backend. This allows for the quick reconstruction of the scene and the creation of a globally consistent map. However, the growing number of robots creates greater requirements for the system [2], including data redundancy, bandwidth usage, and data storage. In the realm of large-scale dense mapping, traditional dense point cloud maps require substantial storage capacity. Simultaneously, due to their vast data volume, these maps pose challenges for real-time maintenance and updates.

In recent years, visual SLAM algorithms for multi-robot systems have been studied by many scholars and have successfully built globally consistent maps. The field of dense mapping has also been extensively researched and applied. Various approaches exist, such as DVO-SLAM for constructing global dense point cloud maps [3], KinectFusion [4], and Kintinuous [5] utilizing truncated signed distance function (TSDF) to represent the model, ElasticFusion [6] for constructing surfel models. Scholars continuously study and enhance methods of dense map representation and construction accuracy [7].

In order to achieve efficient real-time dense mapping with multiple robots, it is essential to reduce the amount of data transmitted and stored by the system, while ensuring that the maps created are globally consistent. Simultaneously, it is necessary to decrease the computational demands of the algorithms while taking into account the hardware capabilities

* Authors to whom any correspondence should be addressed.

of the front-end robots. Dense mapping algorithms currently available often necessitate the use of high-performance GPUs for computation, rendering them unsuitable for implementation in multi-robot systems. Presently, ongoing studies in multi-robot dense mapping primarily concentrate on three areas: task assignment and collaborative path planning [8], dense mapping for distributed robot clusters [9], and dense mapping performed by front-end robots with data compression during transmission [10]. Each of these approaches requires powerful computational and storage capabilities for each robot.

To address the challenges of computational resources, transmission bandwidth consumption in multi-robot dense mapping, as well as to facilitate the rapid creation and upkeep of dense maps, we propose a surfel-based multi-robot dense mapping system. The system employs a generic visual odometry for sparse mapping on the robot. It then transfers the map optimization and dense mapping tasks to the back-end server. The transmission module employs techniques such as co-visible keyframe selection, superpixel extraction, and compressed transmission to alleviate the transmission load of dense information. The back-end of the system incorporates loop closure detection to integrate information from multiple robots and construct a global sparse map. Moreover, by utilizing the worldwide sparse map, we create a surfel co-visibility graph and incorporate superpixel data to build a comprehensive global surfel map. Subsequently, real-time updates of the surfel map are achieved through co-visibility and map optimization techniques. Through these methods, the system effectively achieves centralized multi-robot dense reconstruction. The contributions are summarized as follows:

(i) We present an approach for multi-robot surfel mapping. In our method, the front-end provides superpixels and compressed data to the back-end for surfel mapping. Importantly, our method relies exclusively on CPU computation throughout the process.
(ii) We present a three-stage superpixel segmentation algorithm that effectively integrates superpixel generation with surfel mapping, resulting in enhanced reconstruction quality.
(iii) A multi-robot surfel co-visibility graph is presented to achieve real-time maintenance and updates of surfel map.

The remainder of this article is organized as follows: section 2 discusses related work, section 3 gives the system overview of our proposal, section 4 shows the mapping, superpixel extraction, and data transmission of the robot front-end in the system, and section 5 describes the establishment of global consistent maps and the construction of surfel maps on the back-end server. Section 6 shows the experimental results, and section 7 gives the conclusions and future work.

2. Related work

2.1. Multi-robot SLAM

With the development of SLAM algorithms and multi-robot systems, the study of multi-robot SLAM has received more and more attention. C2TAM [11] adopts a distributed architecture, leveraging cloud servers for map optimization and storage. This approach reduces the storage and computational burden on individual robots but comes with a high demand for network connectivity. MOARSLAM [12], on the other hand, extends the scalability of the system through a client-server approach and allows the incorporation of heterogeneous devices. CCM-SLAM [13] proposes a centralized framework where robots run a visual odometry front-end and a back-end server performs loop closure detection and global optimization, however, transmission bandwidth limits the number of robots. COVINS [14] proposes a framework that increases the number of robots to up to 12 robots for simultaneous mapping by reducing redundant information and coordination overhead. On the other hand, SwarmMap [15] designed three modules to optimize communication data, coordinate task priorities, and eliminate redundant map data, thereby enhancing multi-robot system performance. CCMD-SLAM [16] facilitates multi-robot system operation by reducing communication load through data preprocessing and compression. Scholars have dedicated significant efforts in the field of multi-robot SLAM, focusing on the development of globally consistent maps and communication systems. Nevertheless, the transmission of substantial quantities of dense data continues to pose a significant challenge for multi-robot dense mapping. Enhancing the format of dense data representation has been identified as a viable approach to mitigate communication stress, as suggested by previous studies [10]. In the interim, it is imperative to conduct additional research and investigation into current communication methodologies in order to facilitate the effective transmission of dense data.

2.2. Dense mapping SLAM

SLAM algorithms for dense mapping are generally constructed using LiDAR [17, 18], RGB-D cameras [19] or stereo vision [20, 21]. Certain academics employ a frame-to-frame algorithm to process RGB-D data, thereby constructing a comprehensive global dense point cloud map [3]. Nevertheless, the utilization of dense point clouds presents certain challenges in terms of data updating and storage due to their substantial size. Additionally, they are not well-suited for navigation purposes. Truncated signed distance function (TSDF) representation [22] and voxel representation [23] can discretize space into volumetric elements, providing more compact spatial descriptions compared to raw point cloud representation. KinectFusion [4] adopts a TSDF surface model and aligns it with the iterative closest point (ICP) algorithm on GPU to

achieve dense reconstruction. InfiniTAM [24] achieves efficient memory utilization by modifying, clipping and compressing the data structure of the TSDF model. DSVIO [21] integrates depth information from keyframes into a TSDF model, and enhances reconstruction accuracy by combining stereo optimization with direct visual-inertial odometry. However, the scalability and accuracy of these models are constrained by the spatial discretization. ElasticFusion [6] builds upon the KinectFusion approach by integrating a surfel-based representation and utilizes a model-to-model loop closure detection mechanism to maintain global consistency in loop closures. The surfel maps in BundleFusion [25] are constructed through a process of aligning sparse map to a dense data. SurfelNeRF [26] utilizes the surfel map representation to optimize neural radiance fields for 3D reconstruction, significantly improving reconstruction speed compared to voxel-based representations. Since these algorithms require GPU computation, a CPU-based method [27] for surfel generation based on superpixels is proposed.

3. System overview

In this paper, we propose CMDS-SLAM (Centralized Multi-Robot Dense Surfel SLAM) based on DSM (Real-time Scalable Dense Surfel Mapping) [27]. We have improved the superpixel segmentation algorithm of DSM and established a surfel co-visibility graph to make it applicable in multi-robot systems. Furthermore, we have conducted further research on data transmission and map updates in multi-robot scenarios, thereby establishing a complete multi-robot surfel mapping system.

The system structure of CMDS-SLAM is shown in figure 1. In this system, each robot runs an RGB-D visual odometry front-end and maintains a finite map locally. We divide the process of superpixel extraction and surfel generation between the front-end robots and the back-end server to minimize data transmission and robot computational resources. The system employs a co-visibility criterion to screen keyframes within the local map and selects the RGB-D data keyframes that require dense information transmission for superpixel extraction. Then, we propose a three-stage superpixel generation method based on surfel characteristics and trim the redundant information of transferred superpixels. Subsequently, the superpixel information and keyframe information are transmitted wirelessly to the back-end server for data storage and global map construction. Within the server, the map manager employs loop closure detection to identify overlapping regions of maps from multiple robots and fuses these regions to create a merged map. Pose graph optimization and global optimization techniques are then utilized to enhance the overall accuracy of the global map. The surfel manager, on the other hand, establishes a surfel co-visibility graph using the poses of dense information keyframes in the global map and their co-visibility relationships. This co-visibility graph continuously initializes new surfels and updates the surfel map, enabling the creation and maintenance of the global surfel map. [1]

4. Agent-side mapping and data processing

4.1. Visual odometry front-end

To achieve dense mapping with multiple robots, our system employs RGB-D cameras to perform front-end tasks. In the framework proposed in this paper, any visual odometry (VO) system based on keyframes and processing RGB-D information can be employed to generate a multi-robot globally consistent map, while allowing the use of inertial information to improve operational accuracy. The back-end of the server performs map fusion and global optimization by using keyframe information to build a global dense map. The RGB-D front-end module from ORBSLAM3 [28] is employed in the testing, and the size of the local map it maintained is restricted to alleviate computational burden.

4.2. Superpixel extraction

Image superpixel segmentation refers to the algorithmic process of grouping pixels into homogeneous regions based on color, texture, or spatial information. This technique enables rapid computation of segmentation results while requiring minimal computational resources, making it suitable for real-time mapping with low-performance hardware in multi-robot scenarios. To achieve dense reconstruction while reducing hardware requirements on the robot, we adopt the approach proposed in DSM [27] by generating surfels through superpixel-based methods to construct a dense map.

Our system extracts RGB-D information into superpixel data at the robot end and performs cropping and compression for transmission, which significantly reduces the data transmission overhead. In the DSM approach, the use of the SLIC algorithm (Simple Linear Iterative Clustering) [29] for generating superpixels can lead to errors in surfel map construction due to the significant differences in shape between superpixels and surfels. And the generated superpixels fail to accurately represent the original region colors. To address this issue and achieve more efficient surfel map construction, we have designed a three-stage superpixel extraction method.

In the first stage, we employ the k-means clustering method from SLIC for initial superpixel segmentation, utilizing depth distance as an additional criterion for clustering evaluation. Initially, $\frac{w\times h}{n^2}$ cluster centers Φ are uniformly distributed across the RGB image for initialization, where w and h represent the width and height of the image, respectively, and n denotes the sampling ratio, which is set to 8 or 16 in this work. A cluster center $\Phi = [\phi_x, \phi_y, \phi_i, \phi_d, \phi_r, \phi_{is}, \phi_a]^T$, where $[\phi_x, \phi_y]$ refers to the cluster center coordinates, $[\phi_i, \phi_d]$ refers

[1] https://github.com/suifengshaonian/cmds_ws

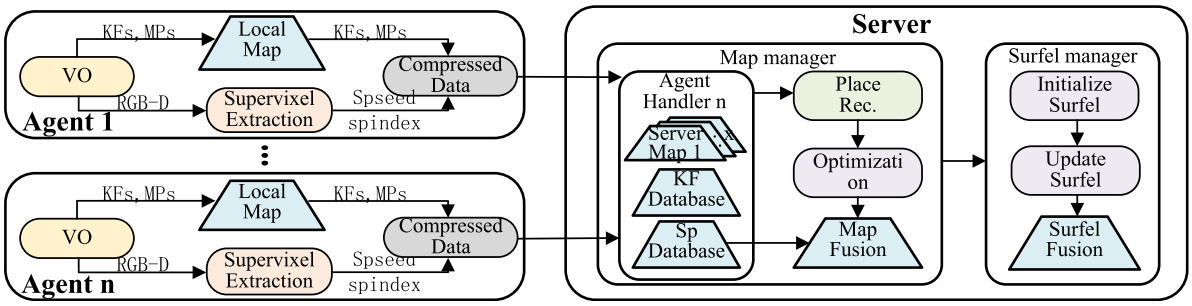

Figure 1. Overview of the CMDS-SLAM system architecture.

to the average intensity and average depth of the cluster, ϕ_r refers to the maximum pixel distance from the cluster center, ϕ_{is} refers to the cluster intensity distance sum, and ϕ_a refers to the number of clustered pixel points. By calculating the distance between pixels and cluster centers, we iteratively update the cluster membership of pixels and the properties of cluster centers Φ, and record ϕ_r, ϕ_{is}, ϕ_a during the iteration. At the same time, record the updated average color of clustered pixels as the superpixel color. The distance is defined as:

$$\begin{cases} d_{\text{dist}} = \frac{(\phi_x - u_x)^2 + (\phi_y - u_y)^2}{\lambda_1}, \\ d_{\text{intensity}} = \frac{(\phi_i - u_i)^2}{\lambda_2}, \\ d_{\text{depth}} = \frac{(1/\phi_d - 1/u_d)^2}{\lambda_3}, \\ d = d_{\text{dist}} + d_{\text{intensity}} + d_{\text{depth}} \end{cases} \tag{1}$$

where, d_{dist}, $d_{\text{intensity}}$,d_{depth} are the pixel distance, intensity distance and depth distance respectively. $[u_x, u_y]$, u_i, u_d are the pixel coordinates, pixel intensities and depth values respectively. λ_1, λ_2, λ_3 are used to normalize the spatial, depth and intensity distances.

When superpixels are segmented, due to the inconsistency between the discreteness of the the initial cluster centers and the pixel intensity values, there may be color-mismatched pixel blocks in some superpixels. In the second stage, we use a color-based clustering method to refine the initial superpixels. This approach takes into account the local color characteristics and geometric shapes of the scene to adjust the boundaries of superpixels, ensuring better alignment with the shape of surface pixels. In the previous stage, we record the intensity variance ϕ_{is} between each pixel within a superpixel and its cluster center. We perform an assessment on ϕ_{is} and decide to split superpixel blocks that exceed a certain threshold. Subsequently, we initialize new cluster centers Φ_{new} by setting their coordinates to the position within the original superpixel with the maximum intensity-distance, and then, we iteratively recalculate the superpixels using (1).

Finally, in the third stage, a post-processing step is performed to split the superpixels based on their shape. This step further refines the boundaries of the superpixels, ensuring a more accurate correspondence with the surface shape. Additionally, it allows for a more precise pixel segmentation using a reduced number of superpixels. Because surfels replace superpixels to represent the original spatial shape with circles, the earlier division of elongated superpixels will affect

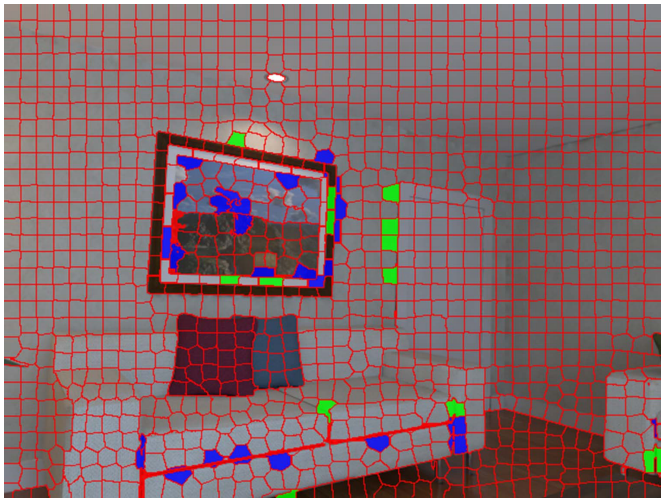

Figure 2. Superpixel segmentation.The green color indicates regions with unevenly assigned colors, the blue color denotes areas requiring further division, and the red color signifies regions earmarked for deletion.

the generation of surfel models. We define a shape metric and judgment mechanism for superpixels based on isochronous quotient [30]:

$$\rho = \frac{\phi_s}{\phi_a} + \frac{\phi_a}{\phi_r} \tag{2}$$

$$\begin{cases} \rho < \theta_1 \quad (\text{invaild}) \\ \theta_1 < \rho < \theta_2 \quad (\text{split}) \\ \rho > \theta_2 \quad (\text{vaild}) \end{cases} \tag{3}$$

where ϕ_s is the perimeter of the superpixel boundary, and two thresholds θ_1 and θ_2 are set to determine shape standards.

When $\rho < \theta_1$, as shown in the red part of figure 2, due to the overly elongated shape of the superpixels, would result in significant errors when mapped to surfels. Simultaneously, since the adjacent elements can fully cover these areas when forming surfels, the superpixel indices in such regions are reset to 0, and no further calculations are performed on them. When $\rho > \theta_2$, the superpixel is 'full' enough not to be modified. When $\theta_1 < \rho < \theta_2$, the superpixels are split into multiple segments along the axis direction. Initialize the new clustering centers by setting the center coordinates evenly along the pixel shapes, and redistribute the superpixels that need to be

modified using d_{dist} in (1) as the iterative calculation of the distances.

4.3. RGB-D information filtering based on co-view

Our system employs a dense mapping algorithm based on keyframes, enabling the extraction of initially accurate point cloud poses within the front-end visual odometry, and enhancing pose precision through back-end optimization. Due to the high overlap among keyframes, selectively filtering the point cloud data during keyframe transmission can significantly reduce data transfer volume without affecting the construction of dense maps. We propose a method for selecting keyframes which aims to reduce redundant RGB-D data transmission by evaluating the shared observation between the current keyframe and connected keyframes.

Firstly, we extract multiple keyframes adjacent to the current keyframe from the co-visibility graph and sort them based on their co-visibility weight w_i with respect to the current keyframe. The weight represents the number of shared feature points observed by keyframe pairs. Check whether the co-visibility weight w_i of the previously transmitted keyframe in the set of adjacent keyframes exceeds the minimum threshold ζ_{min}. When $w_i > \zeta_{\text{min}}$, it indicates a significant observation overlap with the current keyframe. Consequently, the RGB-D data transmission for this keyframe is omitted. For each keyframe that is not transmitted, the initial ζ_{min} is decremented, adapting the filtering threshold dynamically. If no connected keyframe meets the $w_i > \zeta_{\text{min}}$ criterion, then directly transmit the RGB-D data of the current keyframe. Therefore, keyframe selection will selectively transmit RGB-D data based on quantified co-visibility results.

4.4. Organization and transmission of data

To enable multi-robot dense mapping on the back-end, the front-end transmits the established local map information and RGB-D data to the back-end server for map fusion. Regarding the local map information, the transmission module packs the ORB feature descriptors, feature point information, keyframe poses, and other data into 8-bit and 16-bit arrays based on their effective bit counts. This approach minimizes redundant data traffic. The data is bound to the keyframe sequence numbers. When pose transformations occur during front-end or back-end map optimization, only the relative transformation matrices corresponding to the relevant keyframe sequence numbers are transmitted.

For the transmission of RGB-D information, the robot front-end converts the RGB-D data into superpixel indices matrix $\mathbf{P}_{\text{index}}$ and superpixel seeds $\widetilde{\mathbf{P}} \subseteq \mathbb{R}^{12}$, which are then transmitted to the back-end for surfel mapping. The matrix $\mathbf{P}_{\text{index}}$ is a (640×480) matrix, where each element represents the superpixel index corresponding to the pixel position in the original RGB-D image. $\mathbf{P} = [\mathbf{P}_p, \mathbf{P}_n, \mathbf{P}_c, P_s, P_v, P_d] \in \widetilde{\mathbf{P}}$, where $\mathbf{P}_p$ describes the 3D coordinate of the superpixel, $\mathbf{P}_n$ is the normal vector, $\mathbf{P}_c$ represents the color information, P_s is the maximal pixel distance, P_v is the viewing angle, and P_d is the average depth. After excluding the invalid depth points from the RGB image, $\mathbf{P}_p$, $\mathbf{P}_c$, and P_d are obtained based on the cluster center Φ. Furthermore, the average normal vector $\mathbf{P}_n$ of the superpixel plane is computed using (4), while P_v, the angle between the camera viewing direction and the plane normal vector, is computed using (5).

$$\begin{cases} \vec{\mathrm{d}x} = p(u+1,v) - p(u,v) \\ \vec{\mathrm{d}y} = p(u,v+1) - p(u,v) \\ \mathbf{P}_n(u,v) = \mathrm{d}x \times \mathrm{d}y \end{cases} \tag{4}$$

where $\vec{\mathrm{d}x}$ and $\vec{\mathrm{d}y}$ represent differential vectors in the pixel coordinate system of the depth image, while $p(u,v)$ denotes a pixel depth in the depth image.

$$P_v = -\frac{\vec{\mathbf{P}_p} \cdot \vec{\mathbf{P}_n}}{|\vec{\mathbf{P}_p}||\vec{\mathbf{P}_n}|} \tag{5}$$

where the vector $\vec{\mathbf{P}_p}$ is the vector from the origin to $\mathbf{P}_p$.

Superpixel seeds $\widetilde{\mathbf{P}}$ are trimmed by setting thresholds on P_v and P_d to remove invalid superpixels for mapping. Since the superpixel index $\mathbf{P}_{\text{index}}$ is generated by clustering the original image and contains many duplicate elements, run-length encoding [31] can be used to compress it. $\mathbf{P}_{\text{index}}$ is read using differential encoding. The repeated element fields are merged and stored in binary format:

$$aaaabbbbbbcddddd \rightarrow 04a06b01c05d \tag{6}$$

where *abcd* represents the distinct index values in the $\mathbf{P}_{\text{index}}$.

Finally, the compressed $\mathbf{P}_{\text{index}}$ data, serialized superpixel seed $\widetilde{\mathbf{P}}$ information, and the packed keyframe data arrays undergo further compression using the LZ77 algorithm [32] and dictionary compression algorithms, completing the data compression process.

5. Back-end server mapping

5.1. Mutil-map management

The transmission module of the robot end sends various data to the server, including keyframe poses, feature point locations, ORB features, superpixel information, and optional IMU information. After the back-end server receives information from each robot, the map manager initializes multiple map representations based on the number of robots. It then unpacks and stores keyframe data and superpixel data. The map manager will designate the first 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 3D-2D RANSAC is used to calculate the relative pose T between matched keyframes, and Pose Graph Optimization (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. The keyframes are permitted to generate surfels and integrate into the global surfel map only after the map mappings have been incorporated into the main map.

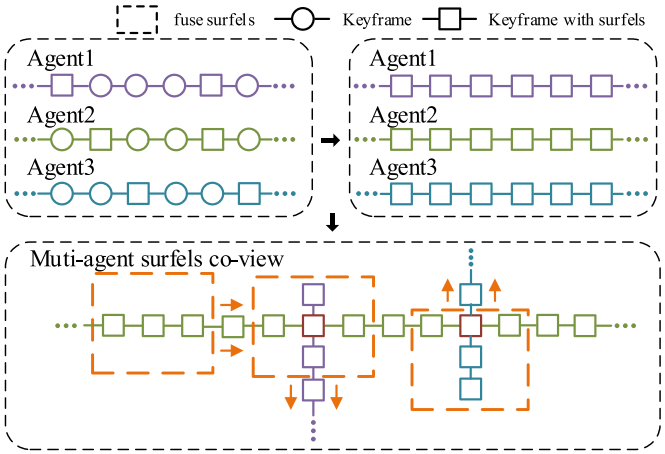

Figure 3. The surfel co-visibility framework.

5.2. Surfel map initialization and update

In the surfel map representation, a scene is accurately depicted as a composition of a series of surfels. Each surfel can be described as $\mathbf{S} = [\mathbf{S}_p, \mathbf{S}_n, \mathbf{S}_c, S_s, S_v, S_i]$, where $\mathbf{S}_p$ represents the 3D position of the surfel, $\mathbf{S}_n$ is the normal vector, $\mathbf{S}_c$ denotes the color information in RGB color model, S_s is the radius, S_v is the viewing angle and S_i is the index of the surfel. During the addition of a new keyframe, the data for each surfel is calculated using the surfel initialization equation (7), utilizing the superpixel seeds $\widetilde{\mathbf{P}}$ transmitted from the robot side

$$\begin{cases} \mathbf{S}_p = \mathbf{R}\mathbf{P}_p + \mathbf{T} \\ \mathbf{S}_n = \mathbf{R}\mathbf{P}_n \\ S_s = P_s \times \left| \frac{P_d}{(F \times P_v)} \right| \end{cases} \tag{7}$$

where, $\mathbf{R}$ is the relative rotation matrix of the keyframe in the global map, $\mathbf{T}$ is the relative translation matrix, F is the camera focal length.

To establish and update the global surfel map for multiple robots, it is necessary to construct the surfel co-visibility graph for the multi-robot system based on the connectivity relationships among the keyframes. The surfel co-visibility graph operates in conjunction with the primary map. It extracts and merges the keyframes carrying superpixel information from each robot according to the relationships within their local maps. When multiple robot maps are combined on the server, a surfel co-visibility graph is created by merging keyframes using loop closure keyframe indices and relative poses, as depicted in figure 3.

To enhance computational efficiency, the server employs a sliding window approach on the co-visibility graph to establish and update the global surfel map. Whenever a new robot map is introduced, a sliding window thread is added, and the window continuously tracks the most recently updated keyframe from that robot. The system performs surfel fusion on the frames within the sliding window. This fusion process encompasses not only the overlapping surfels from individual robot keyframes but also the integration of surfels at the interconnection keyframes between multiple robots within the window. And the surfel frames outside the sliding window are only modified during the optimization process. During the surfel fusion process for keyframes within the window, the newly added frame is projected onto the keyframes that have already generated surfels. The surfel information from the overlapping superpixel regions is determined, and surfel fusion is performed accordingly. For the non-overlapping regions, new surfels are created and added following (7). When fusing the surfel information $\mathbf{S}_{\text{new}}$ from the new keyframe with the surfel information $\mathbf{S}_{\text{old}}$ from the fusion frame, $\mathbf{S}_p$ and $\mathbf{S}_n$ are computed using (8). Simultaneously, $\mathbf{S}_c$, S_v and S_s are determined based on the maximum value of S_v

$$\mathbf{S}_{\text{fuse}} = \frac{\mathbf{S}_{\text{new}}\mathbf{S}_{\text{old}} + \mathbf{S}_{\text{new}}\mathbf{S}_{\text{old}}}{\mathbf{S}_{\text{new}} + \mathbf{S}_{\text{old}}}. \tag{8}$$

5.3. Surfel map optimization

In multi-robot dense mapping tasks, the map accumulates errors from multiple robots as the mapping area expands. In order to maintain the accuracy of the dense map, real-time optimization and updates are required. Surfel map optimization can be easily and quickly corrected by optimizing keyframes, which can be divided into three types of optimization depending on the optimization method and the content of the update:

(i) Local Bundle Adjustment (BA) optimization. Local BA is performed at the front-end, with the updated information transmitted separately through the transmission module. The server updates the pose of individual keyframe surfels based on keyframe index information obtained from the global map.

(ii) Pose Graph Optimization (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 are updated. To update the local surfel in loop closure detection, the keyframe surfel $\mathbf{S}_{\text{ref}}$ that has been identified as a loop is eliminated from $\mathbf{S}_{\text{global}}$. Subsequently, the revised keyframes and adjacent keyframes are reassembled within the surfel generation thread. On the other hand, when a loop closure is detected, updating the poses of the affected surfels is sufficient to complete the correction.

(iii) Global Bundle Adjustment (GBA) optimization. The GBA minimizes the reprojection error of all KFs and MPs using G2O's Levenberg-Marquardt algorithm to improve the accuracy of the overall map and global surfel map. Keyframes that have pose changes after optimization are identified based on the optimization results. The surfels $\mathbf{S}_{\text{i}}$ of each robot with changes are then reconstructed.

In summary, by BA, PGO, and GBA optimization, the accumulation of errors during the mapping process can be mitigated, ensuring map updates. Concurrently, the corresponding surfel update strategy enables efficient and rapid updates to the surfel map while conserving computational resources.

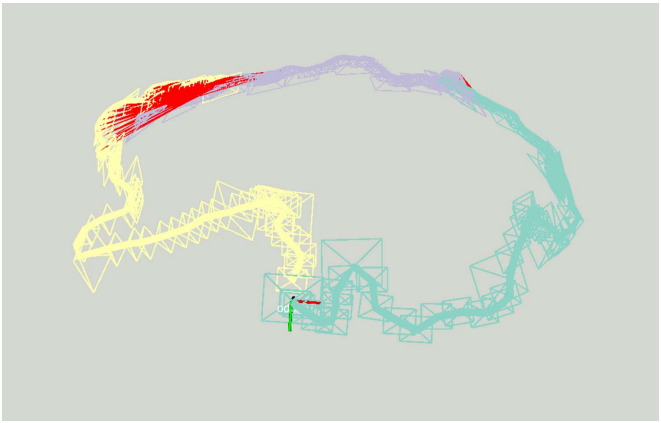

Figure 4. Fusion effect of the three client trajectories.

6. Evaluation

In this section, we perform comprehensive experiments to assess various aspects of the performance of CMDS-SLAM using the RGB-D standard datasets TUM [34] and ICL-NUIM [35], along with the image segmentation dataset NYU Depth Dataset V2 [36]. Specific tests and comparisons are conducted to assess the effectiveness of multi-robot dense mapping, superpixel segmentation, and data transmission efficiency. The experimental platform for this chapter consists of a laptop equipped with a Ryzen 7 4800 H CPU (8 cores @ 2.90GHz), 16GB of memory, and a desktop computer featuring an Intel(R) Core(TM) i5-10 400 CPU (6 cores @ 2.90GHz).

6.1. Multi-robot reconstruction

To assess the effectiveness of multi-robot mapping, we employed a dataset segmentation method as described in [37]. The dataset was segmented into multiple segments with a 10% overlap. Simultaneously, multiple agent front-ends were executed, and their outputs were transmitted over the network to another server. Subsequently, real-time dense mapping and maintenance tasks were performed on the transmitted data.

We first conducted performance evaluations on the *f2_desk* dataset using three clients to assess the mapping performance and trajectory fusion capabilities of the multi-robot system. The fused trajectory is illustrated in figure 4, where trajectories from three distinct agents are represented by different colors. The shared visual border between the trajectories is denoted by the red line. The trajectory fusion exhibits smooth integration in its connected segments, resulting in accurate fusion of trajectories across multiple robots.

Next, we validate the performance of multi-robot mapping on additional datasets from TUM, comparing it with other dense SLAM algorithms. And the absolute trajectory error RMSE (Root Mean Square Error) is utilized to assess the performance of our system. The results of the tests are presented in table 1. The experimental findings provide evidence that our system is capable of accurately rectifying trajectory drift. Specifically, the f1/desk, f2/xyz, and f3/office datasets exhibit root mean square errors (RMSEs) of 0.020 m, 0.004 m, and 0.013 m, respectively. Benefiting from keyframe-based map construction and multi-robot keyframe fusion, our reconstruction RMSEs on all three datasets surpass these dense reconstruction RGB-D SLAM algorithms.

Table 1. Comparison of statistical results of absolute trajectory error RMSE (unit: m).

Methods	f1/desk	f2/xyz	f3/office
Ours	0.020	0.004	0.013
DVO SLAM	0.022	0.014	0.035
Kintinuous	0.142	0.032	0.064
ElasticFusion	0.022	0.009	0.017
BundleFusion	0.016	0.014	0.028
PSM SLAM	0.016	0.088	0.015

Our system employs the CPU to generate a surfel map associated with keyframes through superpixel segmentation. The generated surfel map can be represented using point clouds from the centers of the facets or rendered as a map composed of facets, as shown in figure 5. It is evident that higher superpixel sampling rates can be employed for reconstructing large architectural scenes such as walls, floors, or outdoor streets. Conversely, lower sampling rates are suitable for detailed reconstruction of indoor small objects. Furthermore, the system ensures effective reconstruction outcomes.

To compare the effectiveness of scene reconstruction, multi-robot scene reconstruction was performed on the ICL-NUIM dataset, and a comparative analysis was conducted based on the ground truth model. Due to the insufficient number of keyframes in the kt1 dataset, testing was conducted on the other three datasets. The ICP (Iterative Closest Point) algorithm was utilized to align the centers of point clouds with the assistance of the CloudCompare software. For qualitative analysis, the reconstruction error was quantified by calculating the average distance between the reconstructed point cloud and the nearest surface of the actual 3D model.

We compared our system with several other RGB-D methods, and the results are presented in table 2. It can be seen that our system outperforms other algorithms on the kt0 and kt1 datasets. Overall, our algorithm surpasses all algorithms except for BundleFusion in terms of average performance. On the other hand, in comparison to BundleFusion, which utilizes GPU computation, our algorithm runs on CPU, conserving computational resources. Moreover, the reconstruction performance is superior to the DSM algorithm, which is also based on CPU computation.

6.2. Superpixel segmentation effect test

Visual comparison between our algorithm and the SLIC algorithm in DSM (referred to as DSM-SLIC in the following) is illustrated in figure 6. It is evident that the image after superpixel segmentation preserves a significant amount of original texture details. However, in terms of color assignment, our algorithm exhibits better performance. This is due to dynamic blurring of image edges caused by motion during

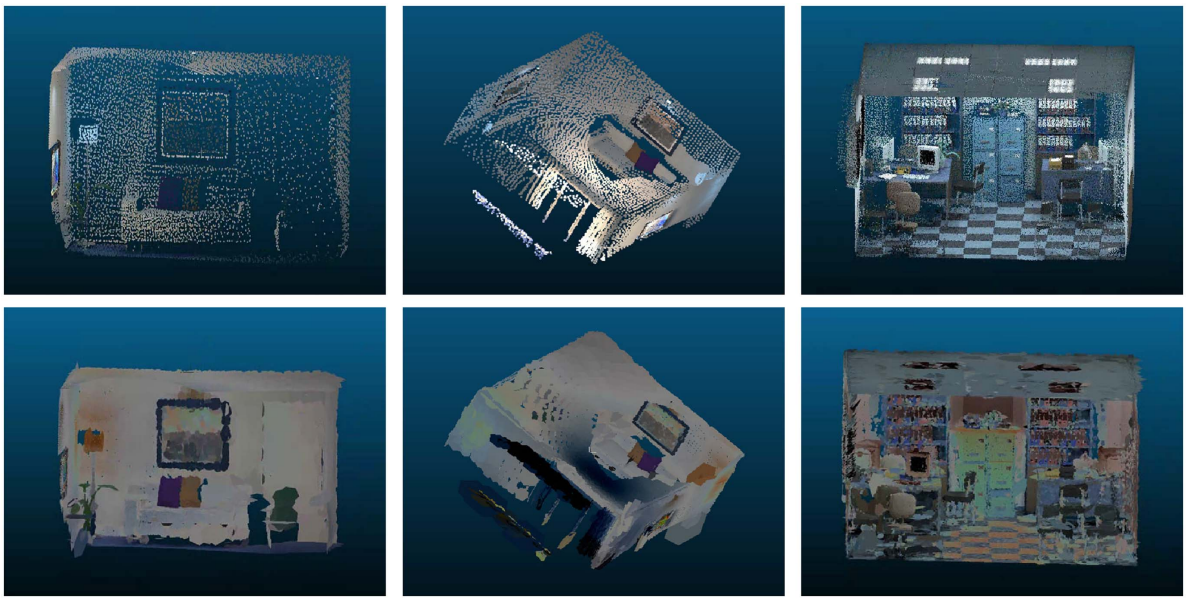

Figure 5. The multi-robot reconstruction results of the ICL-NUIM and WHU-RSVI datasets. The upper images depict point cloud representations, while the lower images show surface representations. Superpixel sampling rates of 16, 8, and 4 are selected from left to right, respectively.

Table 2. The reconstruction accuracy comparison results for the ICL-NUIM dataset (unit: cm).

Methods	kt0	kt2	kt3	avg.
OURS	0.3	1.3	0.7	0.77
DVO SLAM [3]	3.2	11.9	5.3	6.80
Kintinuous [5]	1.1	0.9	15.0	5.77
ElasticFusion [6]	0.7	0.8	2.8	1.43
BundleFusion [25]	0.5	0.7	0.8	0.67
DSM [27]	0.7	1.1	0.8	0.87
PSM SLAM [38]	0.6	2.4	0.9	1.30

image acquisition with a moving camera. Applying superpixel algorithms on such images leads to color deviations at object boundaries. Using the color at superpixel centers to represent superpixels then introduces color errors and discontinuities in the image.

To quantitatively validate the effectiveness of superpixel segmentation in our system, we compared it with the DSM-SLIC algorithm [27] and the VCCS algorithm [39], both applicable to RGB-D information. The comparison was conducted based on an image superpixel count of 1000. We conducted tests on commonly used metrics using the NYU Depth Dataset V2 [36], including: **Compactness (CO)** (Measures the compactness of superpixel shapes [30]), **Boundary Recall (BR)** (Measures the real boundary score overlapping with the segmented boundary), **Under-Segmentation Error (UE)** (Measures the error of superpixels overlapping with multiple objects; a lower value indicates better performance) **Achievable Segmentation Accuracy (ASA)** (Measures the achievable highest segmentation).

From table 3, it is evident that our algorithm exhibits a 6% increase in computation time compared to the DSM-SLIC algorithm. However, both algorithms are ten times faster than the VCCS algorithm. This improvement is attributed to the utilization of multi-threading during computation, significantly enhancing the processing speed. In terms of other metrics, our algorithm has also shown improvement. Notable is the substantial improvement that our algorithm has exhibited, specifically in the compactness measurement utilized critical for surfel generation.

To evaluate the performance of the improved superpixel algorithm in practical mapping, we conducted an ablation study on the TUM dataset. Specifically, during algorithm execution, we tested each surfel keyframe by computing superpixel segmentation using the original algorithm (DSM) and our proposed three-stage improved algorithm (Stages 2 and 3). We then calculated the pixel difference between the superpixel segmentation and the original image, using intensity distance as the metric, and computed the average intensity distance across all keyframes. For all superpixels in an image, the average intensity distance is defined as:

$$\overline{DI} = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{N_j}\sum_{i=1}^{N_j}|I(p_i) - \mu_j| \tag{9}$$

where M is the total number of superpixels in the image, N_j is the number of pixels in the jth superpixel, and μ_j is the average grayscale value of the jth superpixel.

The results in table 4 demonstrate that our improved algorithm significantly reduces the average intensity distance across all test sequences, thereby enhancing the quality of superpixel segmentation. Taking the f1/desk sequence as an example, the original DSM algorithm has an average color distance of 209.588, which is reduced to 129.407 and 123.829 after the second and third stages, respectively, achieving

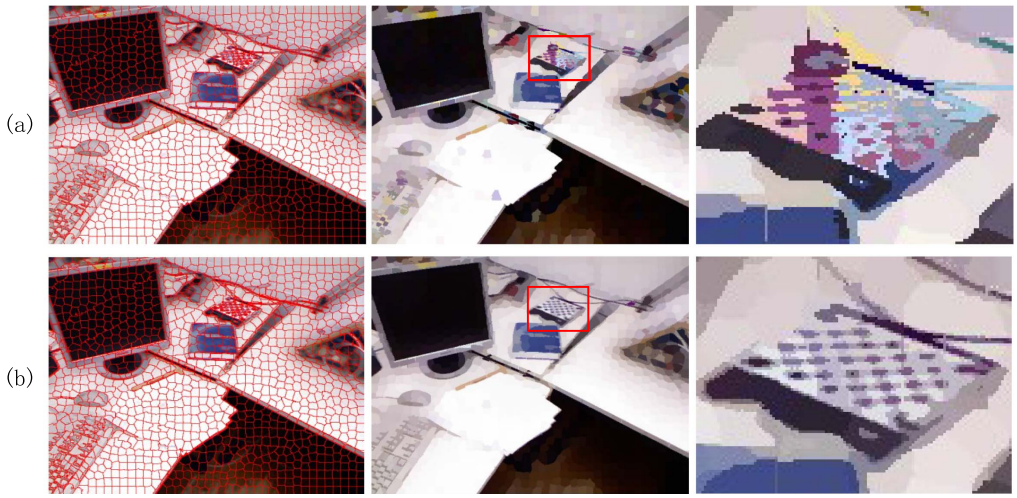

Figure 6. Comparison of superpixel segmentation effect. (a) DSM-SLIC, (b) Our method. Three images respectively display the results of superpixel segmentation, visualization, and local zoom-in effects.

Table 3. Comparison results of superpixel segmentation algorithms.

Methods	Time (s)	BR	1-UE	Co	ASA
OURS	0.06 156	0.8231	0.1095	0.4927	0.9482
DSM-SLIC	0.05 796	0.8182	0.1113	0.4656	0.9464
VCCS	0.53 852	0.7509	0.1315	0.3789	0.9348

Table 4. Average Intensity Distance of Superpixel Algorithms on the TUM Dataset.

Methods	f1/desk	f2/desk	f3/office
DSM	209.588	203.624	191.808
Second Stage (OURS)	129.407	125.289	116.912
Third Stage (OURS)	123.829	120.249	112.728

improvements of 38.3% and 40.9%. Similar enhancements are observed in the f2/desk and f3/office sequences. These results validate that our algorithm better adheres to image color information while maintaining superpixel compactness, leading to more accurate superpixel segmentation.

6.3. Data transfer effect evaluation

To evaluate the data transmission effect, we conducted multi-robot mapping on the TUM dataset while gathering transmission data. The results of the data transmission efficiency are presented in table 5. From the table, it can be observed that the average size of a keyframe is 50.60 KB, and its size remains unaffected by the dataset. In comparison to COVINS [14], where the average size of key frames is 95.93 KB, we managed to reduce data transmission consumption by 52.74%. On the other hand, the transmission of RGB-D information achieved an average of 39.47 KB per frame. The variation in transmission costs across different datasets is attributed to the varying degrees of inter-frame co-visibility in different datasets, influencing the selection of RGB-D information. In conventional algorithms [40], the

Table 5. Data transfer results (unit: KB).

Methods	f1/desk	f2/desk	f2_xyz	f3/office	avg.
keyframes	50.59	50.10	50.34	51.38	50.60
RGB-D	40.48	44.71	40.76	31.96	39.47
COVINS(keyframes)	94.70	96.03	95.77	97.22	95.93

RGB-D information for each keyframe is constituted by a color image of dimensions (640×480) pixels along with depth image. The direct transmission incurs a cost of 1.46 MB per frame. Our system achieves a reduction in the transmission cost of RGB-D information to 2.43%, enabling surfel-based mapping. The number of agents and complexity of the scene do not affect the bandwidth usage of each individual agent, and common WiFi infrastructure can easily facilitate transmission.

6.4. Runtime performance analysis

We employed the method proposed in [10] to evaluate the CPU utilization of the front-end system. On the f2/desk dataset, our system was evaluated in comparison to the conventional ORB-SLAM3 and ORB-SLAM algorithms that utilize direct dense mapping. The experimental results are depicted in the figure 7. From the graph, it is evident that the CPU utilization in robotic mapping increases continuously with the expansion of the maintained map. In contrast, our system employs a front-end that manages a finite map, resulting in a more stable CPU usage during runtime. With both the superpixel extraction thread and the data transmission thread running, when the mapping time exceeds a certain duration, the CPU utilization of CMDS-SLAM is lower than that of ORB-SLAM. Furthermore, the system offloads dense mapping tasks to the back-end server, leading to a 48.93% reduction in CPU utilization on the front-end compared to dense mapping tasks performed solely on the front-end.

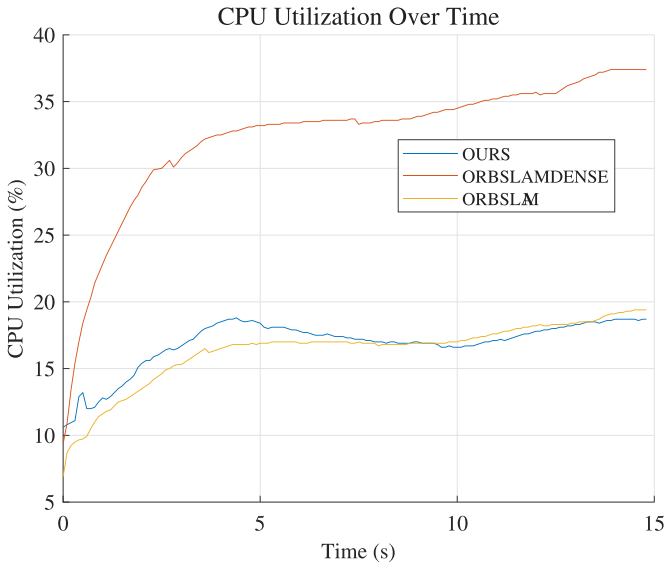

Figure 7. The CPU usage of three different algorithms: CMDS-SLAM, the general ORB-SLAM3, and the ORB-SLAM algorithm with direct dense mapping.

In terms of data transmission time, we evaluated the time consumption at different stages. The time interval for generating new keyframes in scenarios of varying sizes ranged from 292 ms to 877 ms. During the data transmission process, the average time for front-end data processing was 99.6 ms, while the average data transmission time was 3.6 ms. In the back-end processing stage, the data processing time was 2.4 ms, and the computation time for dense mapping was 1.33 ms. These time overheads demonstrate the real-time capability of the proposed SLAM system, enabling efficient data processing and timely updates to the 3D reconstruction of the environment.

7. Conclusion and future work

This article has proposed a dense surfel SLAM system named CMDS-SLAM for centralized multiple robots. The system incorporates a transmission framework that, through a co-visibility frame filtering mechanism, superpixel segmentation, and data compression, effectively alleviates the data transmission burden. And based on the principles of superpixel generation for surfel mapping, we devised a three-stage superpixel segmentation method. This method exhibits enhanced compactness, thereby improving the effectiveness of dense mapping. Finally, a surfel frame co-visibility graph was established, and rapid and accurate updates of multi-robot surfel maps were achieved through map optimization and co-visibility relationships. We conducted comprehensive tests on various aspects of the system using the TUM, ICL-NUIM, and NYU Depth Dataset V2 datasets. The results indicate that our system can effectively reduce transmission pressure, achieving high-precision establishment and real-time updating of multi-robot surfel maps.

The CMDS-SLAM system presents a new outlook on addressing the challenges of multi-robot dense mapping in real-time scenarios. By successfully balancing the trade-offs between data transmission, processing efficiency, and map quality, our work opens up new possibilities for applications in complex environments such as search and rescue operations. The innovative approach of combining co-visual keyframe filtering, superpixel segmentation, and surfel co-visibility graphs provides a framework that can be adapted and extended to various multi-robot collaborative tasks requiring dense environmental mapping.The success of CMDS-SLAM in reducing data transmission while maintaining map quality suggests potential applications in scenarios with limited bandwidth or where real-time decision-making based on dense environmental information is critical.

In future work, the optimization of the transmission framework using superpixel segmentation will be further studied, and a more precise surfel map will be built to further improve this method. Additionally, investigating the scalability of the system to larger robot teams and exploring its performance in more diverse and challenging environments could provide valuable insights for the broader field of multi-robot SLAM.

Data availability statement

The data cannot be made publicly available upon publication because the cost of preparing, depositing and hosting the data would be prohibitive within the terms of this research project. The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

This work was supported in part by National Key R&D Program of China under Grant 2018YFB2100903.

ORCID iDs

Chenle Zuo https://orcid.org/0000-0002-0150-6412
Xiaohui Xiao https://orcid.org/0000-0002-8212-2452

References

[1] Liu H, Wang X, Liu R, Xie Y and Li T 2024 Air-ground multi-agent system cooperative navigation based on factor graph optimization SLAM *Meas. Sci. Technol.* **35** 066303
[2] Zhang Z, Zhang F and Ji C 2021 Multi-robot cardinality-balanced multi-bernoulli filter simultaneous localization and mapping method *Meas. Sci. Technol.* **33** 035101
[3] Kerl C, Sturm J and Cremers D 2013 Dense visual SLAM for RGB-D cameras *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (IEEE) pp 2100–6
[4] Newcombe R A, Izadi S, Hilliges O, Molyneaux D, Kim D, Davison A J, Kohi P, Shotton J, Hodges S and Fitzgibbon A 2011 Kinectfusion: real-time dense surface mapping and tracking *2011 10th IEEE Int. Symp. on Mixed and Augmented Reality* (IEEE) pp 127–36
[5] Whelan T, Kaess M, Fallon M, Johannsson H, Leonard J and McDonald J 2012 Kintinuous: spatially extended kinectfusion 3rd RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras

[6] Whelan T, Leutenegger S, Salas-Moreno R, Glocker B and Davison A 2015 Elasticfusion: real-time dense slam and light source estimation *Int. J. Robot. Res.* **35** 1697–716

[7] Fu X, Li G and Yu L 2020 Interior dense 3D reconstruction system with RGB-D camera for complex large scenes *Meas. Sci. Technol.* **32** 015403

[8] Dong S, Xu K, Zhou Q, Tagliasacchi A, Xin S, Nießner M and Chen B 2019 Multi-robot collaborative dense scene reconstruction *ACM Trans. Graph.* **38** 1–16

[9] Tian Y, Chang Y, Herrera Arias F, Nieto-Granda C, How J P and Carlone L 2022 Kimera-multi: robust, distributed, dense metric-semantic SLAM for multi-robot systems *IEEE Trans. Robot.* **38** 2022–38

[10] Liu X, Ye W, Tian C, Cui Z, Bao H and Zhang G 2021 Coxgraph: Multi-robot collaborative, globally consistent, online dense reconstruction system *2021 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (IEEE) pp 8722–8

[11] Riazuelo L, Civera J and Martınez Montiel J M 2014 C^2TAM: a cloud framework for cooperative tracking and mapping *Robot. Auton. Syst.* **62** 401–13

[12] Morrison J G, Gálvez-López D and Sibley G 2016 MOARSLAM: multiple operator augmented rslam *Distributed Autonomous Robotic Systems: the 12th Int. Symp.* (Springer) pp 119–32

[13] Schmuck P and Chli M 2019 CCM-SLAM: robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams *J. Field Robot.* **36** 763–81

[14] Schmuck P, Ziegler T, Karrer M, Perraudin J and Chli M 2021 COVINS: visual-inertial SLAM for centralized collaboration *2021 IEEE Int. Symp. on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)* (IEEE) pp 171–6

[15] Xu J, Cao H, Yang Z, Shangguan L, Zhang J, He X and Liu Y 2022 *SwarmMap*: scaling up real-time collaborative visual *SLAM* at the edge *19th USENIX Symp. on Networked Systems Design and Implementation (NSDI 22)* pp 977–93

[16] Zuo C, Feng Z and Xiao X 2024 CCMD-SLAM: communication-efficient centralized multi-robot dense SLAM with real-time point cloud maintenance *IEEE Trans. Instrum. Meas.* **73** 1–12

[17] Dai Z, Zhou J, Li T, Yao H, Sun S and Zhu X 2023 An intensity-enhanced LiDAR SLAM for unstructured environments *Meas. Sci. Technol.* **34** 125120

[18] Liu Y, Wang C, Wu H, Wei Y, Ren M and Zhao C 2022 Improved LiDAR localization method for mobile robots based on multi-sensing *Remote Sens.* **14** 6133

[19] Kuang B, Yuan J and Liu Q 2022 A robust RGB-D SLAM based on multiple geometric features and semantic segmentation in dynamic environments *Meas. Sci. Technol.* **34** 015402

[20] Li X, Shen Y, Lu J, Jiang Q, Xie O, Yang Y and Zhu Q 2022 DyStSLAM: an efficient stereo vision SLAM system in dynamic environment *Meas. Sci. Technol.* **34** 025105

[21] Liu Y, Wu H, Wang C, Wei Y, Ren M and Feng T 2023 Real-time dense construction with deep multi-view stereo using camera and imu sensors *IEEE Sens. J.* **23** 19648–659

[22] Reijgwart V, Millane A, Oleynikova H, Siegwart R, Cadena C and Nieto J 2019 Voxgraph: globally consistent, volumetric mapping using signed distance function submaps *IEEE Robot. Autom. Lett.* **5** 227–34

[23] Whelan T, Kaess M, Johannsson H, Fallon M, Leonard J J and McDonald J 2015 Real-time large-scale dense RGB-D SLAM with volumetric fusion *Int. J. Robot. Res.* **34** 598–626

[24] Adrian Prisacariu V, Kähler O, Golodetz S, Sapienza M, Cavallari T, Torr P H S, and Murray D W 2017 InfiniTAM v3: a framework for large-scale 3D reconstruction with loop closure (arXiv:1708.00783)

[25] Dai A, Nießner M, Zollhöfer M, Izadi S and Theobalt C 2017 Bundlefusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration *ACM Trans. Graph.* **36** 1

[26] Gao Y, Cao Y-P and Shan Y 2023 Surfelnerf: neural surfel radiance fields for online photorealistic reconstruction of indoor scenes *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition* pp 108–18

[27] Wang K, Gao F and Shen S 2019 Real-time scalable dense surfel mapping *2019 Int. Conf. on Robotics and Automation (ICRA)* (IEEE) pp 6919–25

[28] Campos C, Elvira R, Gómez Rodríguez J J, Montiel J M M and Tardós J D 2021 ORB-SLAM3: an accurate open-source library for visual, visual–inertial and multimap SLAM *IEEE Trans. Robot.* **37** 1874–90

[29] Achanta R, Shaji A, Smith K, Lucchi A, Fua P and Süsstrunk S 2012 Slic superpixels compared to state-of-the-art superpixel methods *IEEE Trans. Pattern Anal. Mach. Intell.* **34** 2274–82

[30] Schick A, Fischer M and Stiefelhagen R 2012 Measuring and evaluating the compactness of superpixels *Proc. 21st Int. Conf. on Pattern Recognition (ICPR2012)* (IEEE) pp 930–4

[31] Harry Robinson A and Cherry C 1967 Results of a prototype television bandwidth compression scheme *Proc. IEEE* **55** 356–64

[32] Collet Y Zstandard - Real-time data compression algorithm 2023 (Facebook) (available at: https://github.com/facebook/zstd)

[33] Gálvez-López D and Tardos J D 2012 Bags of binary words for fast place recognition in image sequences *IEEE Trans. Robot.* **28** 1188–97

[34] Sturm J, Engelhard N, Endres F, Burgard W and Cremers D 2012 A benchmark for the evaluation of RGB-D SLAM systems *2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (IEEE) pp 573–80

[35] Handa A, Whelan T, McDonald J and Davison A J 2014 A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM *2014 IEEE Int. Conf. on Robotics and Automation (ICRA)* (IEEE) pp 1524–31

[36] Silberman N, Hoiem D, Kohli P and Fergus R 2012 Indoor segmentation and support inference from RGBD images *Computer Vision–ECCV 2012: 12th European Conf. on Computer Vision (Florence, Italy, October 7–13, 2012 Proc. Part V* vol 12) (Springer) pp 746–60

[37] Ma T, Zhang T and Li S 2020 Multi-robot collaborative SLAM and scene reconstruction based on RGB-D camera *2020 Chinese Automation Congress (CAC)* (IEEE) pp 139–44

[38] Yan Z, Ye M and Ren L 2017 Dense visual SLAM with probabilistic surfel map *IEEE Trans. Vis. Comput. Graph.* **23** 2389–98

[39] Papon J, Abramov A, Schoeler M and Worgotter F 2013 Voxel cloud connectivity segmentation-supervoxels for point clouds *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 2027–34

[40] Zhu D, Xu G, Wang X, Liu X and Tian D 2021 Paircon-SLAM: distributed, online and real-time RGBD-SLAM in large scenarios *IEEE Trans. Instrum. Meas.* **70** 1–14