Robust Collaborative Visual-Inertial SLAM for Mobile Augmented Reality

Xiaokun Pan Gan Huang Ziyang Zhang Jinyu Li Hujun Bao Guofeng Zhang

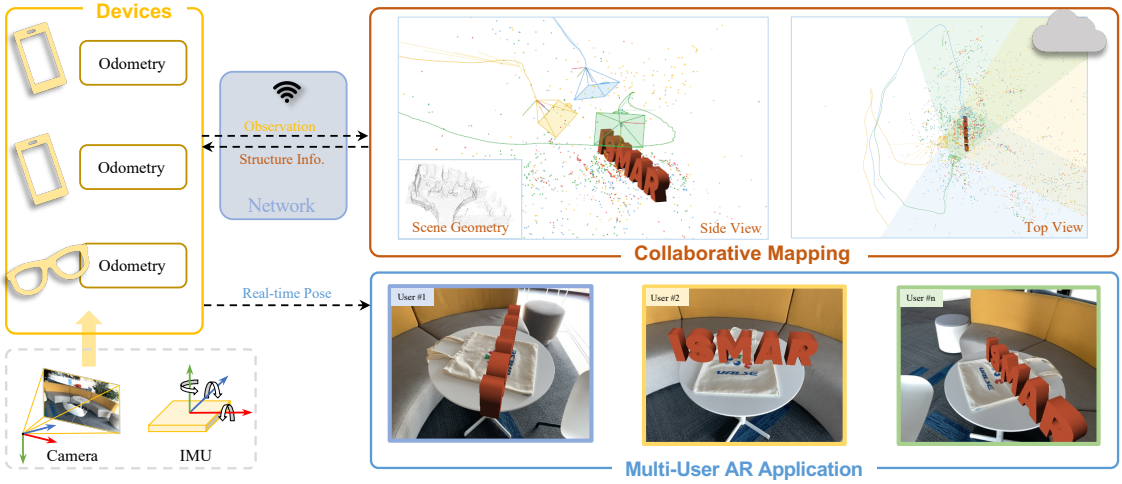

Fig. 1: We propose a robust centralized collaborative SLAM system with multiple agents. Each agent communicates with the server backend process via a wireless network connection. The system is capable of real-time camera pose tracking and global map reconstruction, with each camera and its corresponding scene view annotated in the same color. Leveraging multi-agent collaboration, we demonstrate interactive augmented reality technology involving multiple users.

Abstract—Achieving precise real-time localization and ensuring robustness are critical challenges in multi-user mobile AR applications. Leveraging collaborative information to augment tracking accuracy on lightweight devices and fortify overall system robustness emerges as a crucial necessity. In this paper, we propose a robust centralized collaborative multi-agent VI-SLAM system for mobile AR interaction and server-side efficient consistent mapping. The system deploys a lightweight VIO frontend on mobile devices for real-time tracking, and a backend running on a remote server to update multiple submaps. When overlapping areas between submaps across agents are detected, the system performs submap fusion to establish a globally consistent map. Additionally, we propose a map registration and fusion strategy based on covisibility areas for online registration and fusion in multi-agent scenarios. To improve the tracking accuracy of the frontend on agent, we introduce a strategy for updating the global map to the local map at a moderate frequency between the camera-rate pose estimation of the frontend VIO and the low-frequency global map optimization, using a tightly coupled strategy to achieve consistency of the multi-agent frontend poses estimation in the global map. The effectiveness of the proposed method is further confirmed by executing backend mapping on the server and deploying VIO frontends on multiple mobile devices for AR demostration. Additionally, we discuss the scalability of the proposed system by analyzing network traffic, synchronization frequency, and other factors at both the agent and server ends.

Index Terms—SLAM, VIO, Collaborative, Tightly coupled, Map fusion

1 Introduction

In recent years, state-of-the-art VIO/VI-SLAM systems have witnessed significant advancements in terms of accuracy and robustness within single-agent frameworks. This progress has prompted researchers to explore the extension of these systems to multi-agent applications. Such exploration involves harnessing sensor inputs from multiple agents, encompassing cameras, inertial measurement unit (IMU), or GPS data, to collaboratively amalgamate information. This collaborative effort aims to achieve a comprehensive perception of the environment and to enhance the localization capabilities of individual agents. Multi-agent collaborative perception systems hold vast potential in efficiently exploring uncharted environments, updating and maintaining dynamic digital maps, and facilitating immersive multi-user augmented reality (AR) experiences. With the prosperous interest in AR applications, facilitating interactive AR interactions among multiple users has emerged as a critical requirement. Nevertheless, the deployment of multi-agent systems is fraught with challenges, including map registration across agents, constraints related to network bandwidth and latency, and the issue of pose drift in AR scenarios within single-agent contexts. These challenges pose significant obstacles to the effective implementation and broader adoption of multi-agent systems in the realm of AR and beyond.

VIO has emerged as a crucial area of SLAM research, owing to its proficiency in delivering real-time estimations of 6-degree of freedom (DoF) poses while effectively resolving scale ambiguities. The integration of multi-sensor fusion techniques further augments the robustness of VIO-based methodologies, rendering them superior to their purely visual SLAM counterparts. In the realm of single-agent applications,

- *Xiaokun Pan, Gan Huang, Ziyang Zhang, Hujun Bao, Guofeng Zhang are with State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China. E-mails: {xkpan, huanggan, zhangzion, baohujun, zhangguofeng}@zju.edu.cn*
- *Jinyu Li is with Dreame Techonlogy. E-mail: mail@jinyu.li*
- *Corresponding Author: Guofeng Zhang.*

particularly within augmented reality (AR) and virtual reality (VR), VIO's capability for real-time tracking and local map reconstruction in uncharted environments has been instrumental. Notable implementations in this sector include augmented reality platforms such as ARKit[1] and ARCore[2]. These platforms leverage visual-inertial fusion for enhanced tracking accuracy, while also maintaining local backend maps to assure spatial consistency.

In contrast to VIO's emphasis on local tracking, the essence of a collaborative SLAM system is the establishment and maintenance of a globally coherent map. This is achieved by sharing mapping data among multiple agents and refining uncertain state estimations. As a result, the system not only ensures global map coherence but also significantly improves localization precision with efficiency and robustness. Current collaborative systems built upon VIO frameworks, whether they adopt centralized methodologies [19,30,33,34] or distributed [6,38,39], typically focus on integrating local maps from individual agents into a unified global map at the backend. Such methods often neglect the potential of the globally consistent maps to correct the frontend drift accumulation, or they only employ loosely coupled strategies, which can result in discontinuities in the VIO frontend poses. Additionally, the intricate process of cross-agent map alignment poses challenges for efficient deployment in multi-user AR scenarios.

In this work, our focus is the development of a robust collaborative SLAM system for multi-user AR interactions, concentrating on collaborative localization and consistent mapping between multiple agents. Unlike previous research that mainly concentrated on backend collaborative processing of agent data for accurate mapping [6, 19, 30], our proposed framework intends to deploy lightweight algorithms on consumer-grade mobile devices, such as smartphones and AR glasses. This deployment provides frontend states with consistent map states in the backend for states fusion, while also enhancing the efficiency and accuracy of collaborative systems.

As shown in Fig. 1, the proposed collaborative framework with multi-agent efficiently accomplishes cross-agent map registration and exhibits high-precision real-time pose estimation in AR scenarios. The main contributions of this paper are as follows:

- We present a robust multi-agent collaborative SLAM framework that achieves real-time and precise localization for resource-constrained frontends across multiple agents, coupled with collaborative and consistent mapping on the backend. Our frontend can be deployed on the mobile devices with limited computing capabilities, while the backend is designed for cloud or local server to facilitate collaboration.
- A lightweight and efficient online cross-agent map fusion strategy based on covisibility is introduced, which mitigates the latency and inefficiency in cross-agent map registration and fusion stemming from bag-of-words loop detection. This approach enhances the efficiency of cross-agent map registration and fusion in AR scenes.
- We propose a tightly coupled approach that synchronize the backend structure with the frontend's real-time pose estimation. By integrating the backend's map points and keyframes, which are constrained by global consistency, with the VIO frontend's sliding window state, our system dynamically corrects for drift in real-time, resulting in enhanced pose accuracy.
- The entire codebase of our system and the dataset will be made publicly available, including the frontend VIO for agents, the backend server code, and an iOS project tailored for mobile AR, thereby contributing to the open-source community and facilitating further advancements in this field.

Through extensive comparative evaluations conducted on the EuRoC dataset [4] and our newly collected VICON dataset, our approach is shown to outperform existing state-of-the-art multi-agent collaborative SLAM methodologies in both accuracy and robustness. We not only validated the effectiveness of the proposed map fusion strategy based on online covisibility area but also demonstrated that the precision of the frontend agents benefits from the tightly-coupled architecture we have introduced. Remarkably, our framework is well-suited for deployment on mobile devices with limited computational resources. We have developed an AR demonstration using consumer-grade mobile phones to showcase the practical applicability and effectiveness of our proposed collaborative SLAM framework in real-world scenarios.

[1] https://developer.apple.com/documentation/arkit

[2] https://developers.google.com/ar

2 Related Works

In the realm of real-time tracking and mapping tasks for augmented reality based on multi-agent collaboration, this study is aimed at addressing several key challenges. These challenges include effectively utilizing an individual agent to construct a local map and achieve positioning within an independent coordinate system, as well as leveraging multiple agents to establish a structurally consistent global map. Additionally, it is crucial to explore mechanisms for delivering reliable localization outcomes and ensuring map consistency from the global map to the frontend. Ongoing research in this field is investigating innovative strategies, including SLAM systems that integrate advanced deep learning solutions [26, 36, 37, 40, 43] and the incorporation of additional prior information, such as diverse sensors [7, 10, 28, 45] or pre-built maps [2, 17, 18, 25], to optimize key modules within the system, thereby enhancing tracking accuracy and overall performance efficiency.

2.1 VIO/SLAM

VIO/VI-SLAM have emerged as a significant area of interest within the realm of computer vision research. Visual-only odometry is detailed in [41]. Due to the scale ambiguity of monocular VO and VSLAM, scale is typically recovered by fusing inertial measurements from an IMU with visual measurements. VIO systems can be categorized into two types based on the fusion technique: filter-based [3, 13, 16, 20] and optimization-based [5, 11, 20–22, 31].

Filter-based methodologies have been among the first to be validated due to their computational advantages. A significant advancement in this area is MSCKF [27], an early VIO system grounded in Kalman filtering. It coalesces geometric constraints and IMU measurements within a multi-state constrained Extended Kalman Filter (EKF). ROVIO [3] is another innovative system that champions the use of photometric error in visual measurements, thereby obviating the need for pre-extracted feature points as necessitated by MSCKF. Moreover, OpenVINS [13], a widely used open-source platform, modularizes a positioning and navigation system based on the MSCKF.

With the advancement and increasing sophistication of nonlinear optimization techniques, optimization-based VIO/SLAM systems have emerged and demonstrated higher accuracy in localization and mapping. Recent VI-SLAM systems, including VINS-Mono [31] and VINS-Fusion, adopting keyframe-based bundle adjustment and sliding window techniques in the frontend, as well as pose graph optimization (PGO) based loop closure in the backend to mitigate drift accumulation, thereby achieving better accuracy. In the latest development, ORB-SLAM3 [5] has demonstrated a hightly tightly coupled system that can produce precise results, solving the complete SLAM problem. However, due to the need for early poses to be optimized through subsequent observations, this is impractical for real-time applications. On the other hand, RD-VIO [21] proposes a moving point removal strategy in dynamic scenes and a subframes mechanism for camera motion degradation, aiming to achieve lightweight, high-precision real-time tracking on mobile devices.

In VIO-based AR systems, fusing observations from multiple sensors can significantly enhance accuracy. However, despite these improvements, the accumulation of drift in VIO systems remains an inevitable issue as tracking distances increase. This phenomenon is primarily attributed to factors such as measurement noise and the inherent non-convexity of the SLAM least squares problem, among others.

Table 1: Comparison of different Collaborative SLAM methods. In the architecture, C represents centralized, and D represents distributed. Compared to previous work, our proposed method adopts a tightly coupled optimization for frontend camera-rate pose estimation and is capable of running on mobile phones.

Method	Arch.	Camera	IMU	Coupled	Mobile
CoSLAM [44]	C	Mono	No	No	No
CVI-SLAM [19]	C	Mono	Yes	No	No
CCM-SLAM [33]	C	Mono	No	No	No
COVINS(-G) [30, 34]	C	Mono	Yes	No	No
Kimera-Multi [6]	D	Stereo/Depth	Yes	-	No
D^2SLAM [38]	D	Mono	Yes	-	No
Ours	C	Mono	Yes	Yes	Yes

2.2 Collaborative SLAM

The collaborative SLAM systems for multi-agent setups can be categorized into centralized and distributed architectures based on their architectures design.

In a distributed architecture [6, 38, 39, 42], each agent independently processes its own sensor inputs while sharing data with other agents. This approach enables faster localization and map reconstruction by processing local information, and enhances system robustness by eliminating reliance on a central server. However, it requires frequent communication and data synchronization, potentially increasing network latency and communication overhead.

In contrast, centralized solutions use a client-server model where a central server handles data storage and dissemination. This allows the server to manage computationally intensive tasks, reducing the load on local agents and enabling them to focus on real-time processing. Most research supports the centralized approach, as seen in studies like [19, 30, 32–34].

C2TAM [32] proposed an early framework attempting collaborative tracking and mapping. It deployed all mapping tasks on the server, while the frontend agents relied on limited computational resources for visual tracking. However, this work did not optimize robustness and system efficiency. [29] leverages edge servers to maintain map databases, perform global optimization, and design an efficient communication channel for real-time information sharing among robots. However, the frontend needs to maintain local maps and real-time data sharing, resulting in significant communication overhead. CVI-SLAM [19] adopts a centralized collaborative approach and introduces bidirectional communication to provide feedback from the backend map optimization to the frontend VIO system. [23] focuses on collaborative dense reconstruction with multiple agents, employing a compact 3D representation for map transmission. It also introduces a robust map fusion algorithm based on joint optimization of trajectories and submaps, enabling real-time generation of globally consistent dense maps with reduced transmission load. [44] proposed a system where multiple monocular visual agents collectively uphold a global map containing static background points and dynamic objects. COVINS [34] developed a novel multi-robot system framework to achieve efficient and accurate frontend data processing at the server while reducing redundant information in data communication, showing significant advantages in cooperative overhead. Furthermore, COVINS-G [30] discussed a frontend-agnostic collaborative backend framework, aiming to deploy various types of VIO frontends. The Kimera-Multi [6] proposed a robust collaborative framework under limited communication bandwidth conditions, employing a fully distributed communication scheme to implement the global consistency metric-semantic 3D grid model for environment reconstruction. In Tab. 1, we present a comparison between our method and state-of-the-art collaborative SLAM approaches. The *coupled* means the tightly coupled between frontend and backend in centralized architecture, and the *mobile* means it can be deployed on a mobile device with limited resource such as a mobile phone.

3 Overview

3.1 Framework

Fig. 2 illustrates our system framework, which employs a centralized server-client architecture. A lightweight VIO frontend is deployed on the mobile device (*e.g.*, mobile phone or AR glasses), while a backend process operates on the server side. The mobile frontend and the server backend establish a connection via wireless communication to enable smooth interaction and data synchronization between the two components. By presetting a listening port on the server side, we can establish communication with an indeterminate number of agents, engaging them in collaborative localization and mapping.

Frontend: The VIO frontend consists of a feature extractor and a tracking module based on a sliding window, similar to the approach in [21, 31]. Therefore, during the tracking thread, it is only necessary to maintain a local map determined by the sliding window for localizing the latest image. The feature tracking module utilizes the KLT [24] optical flow algorithm to establish feature correspondences by maintaining feature points across consecutive frames. We maintain a fixed-size N_W sliding window composed of the latest keyframes, and through frequent bundle adjustment (BA) executions, we achieve joint optimization of pose and local map points. Simultaneously, we employ motion-only visual-inertial optimization to output state estimates at the camera rate.

Backend: The server side consists of three modules: map maintenance module, loop detector and closure module, and optimization module.

- *Map Maintenance*: It is responsible for receiving keyframes from the frontend VIO localization, constructing a local map, and, when the system detects that multiple agents have covisibility area, completing map registration and fusion. This aspect will be detailed in Sec. 4.2.1 .
- *Loop Detector and Closure*: We adopt a strategy similar to [11, 31] for loop detection, using a bag-of-words model to describe images and maintain a keyframe database of the current map for detecting similar areas in the scene. We will introduce this in the Sec. 4.3.
- *Optimizer*: This module includes local bundle adjustment (LBA) for the local map, pose graph optimization (PGO) during loop closure, and the final global bundle adjustment (GBA). When registering a new keyframes in the backend, we execute LBA to achieve joint optimization of the current frame and the local map. Upon detecting a loop, we first execute PGO to correct the drift of keyframes. Once this is completed and the system state allows, GBA is executed in a separate thread.

3.2 Notation

To ensure clarity and conciseness, we first define the symbols and coordinate systems used in this work. Due to the timeliness of IMU pre-integration, the state of frames in the frontend VIO differs slightly from that in the backend. In the frontend VIO, the state vector of the k-th frame is parameterized as

$$s_k = \left\{ p_k, q_k, v_k, b_k^g, b_k^a \right\}, \tag{1}$$

where these variables represent position p, orientation q, velocity v, gyroscope bias b^g, and accelerometer bias b^a, respectively. We use the Hamilton quaternion form q to denote rotation, and R denotes the corresponding matrix form, means camera to world. In frontend, we employ the inverse depth parameterization [8] to represent landmark. Therefore, the state of landmark x_j includes only a one-dimensional inverse depth parameter d_j, and its position in the k-th frame is denoted as $z_{k,j}$, which is associated with the reference keypoint position $z_{k,r}$ on the reference frame r. For simplicity, we assume the intrinsic camera parameter matrix K is constant, and the extrinsic parameters between the camera and IMU are ideal, *i.e.*, their relative transformation is represented by the identity matrix. In the backend, as the IMU pre-integration information is removed, the state of the k-th keyframe can be simply parameterized as $s_k^b = \left\{ p_k^b, q_k^b \right\}$, and the global map points are parameterized based on Cartesian coordinates, representing 3DoF

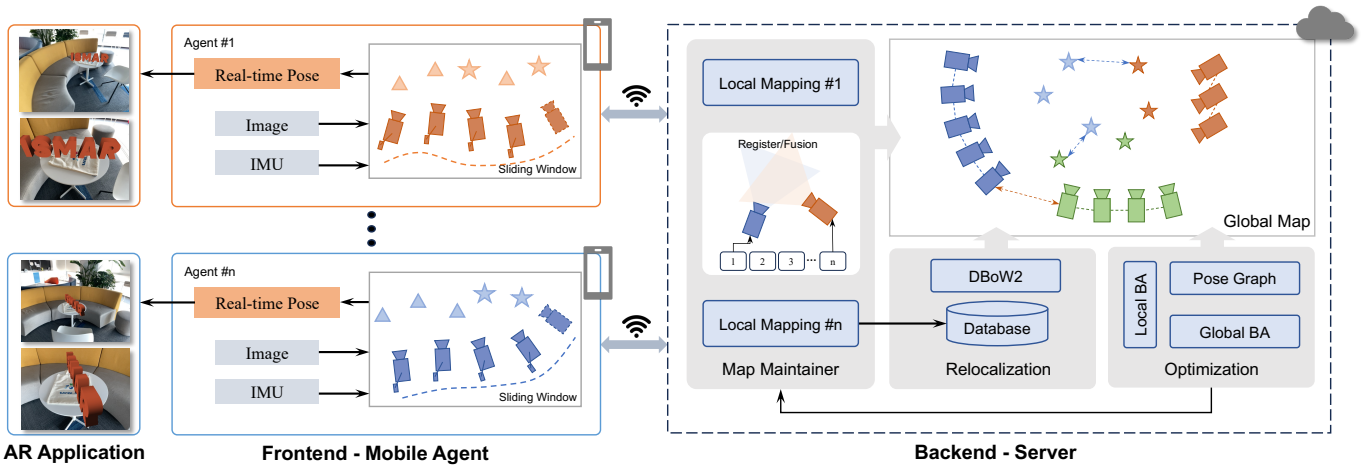

Fig. 2: System architecture of RCO-SLAM: The system is designed for deployment across a variable number of mobile devices in conjunction with a central server. The system's frontend utilizes image and IMU data to facilitate camera-rate 6DoF pose estimation. Data from the frontend is transmitted to the server side via a wireless network, where the backend performs consistent mapping. This is achieved through various modules, including multi-map management, relocalization, and optimization processes.

as $x_j^b = \{x, y, z\}$. We use $\Pi(\cdot)$ to denote normailized projection and represent a vector's homogeneous form as $\bar{z} = \begin{pmatrix} z \\ 1 \end{pmatrix}$. Therefore, 3D representation of the frontend map point is given by

$$x_j = \frac{1}{d_j} R_r K_r^{-1} \bar{z}_{k,r} / \left\| K^{-1} \bar{z}_{k,r} \right\| + p_r. \tag{2}$$

Subsequently, we will no longer explicitly distinguish between the specific forms of frontend map points and backend map points.

4 APPROACH

4.1 Frontend VIO

Mainstream approaches for VIO frontend can be broadly categorized into two types. The first type, exemplified by works such as [21, 31], utilizes a sliding window mechanism to construct a local map. This method employs the KLT [1] optical flow based feature tracking algorithm to establish inter-frame data associations. Subsequently, it achieves frame localization by constraining visual tracking within the sliding window and IMU pre-integration. The second type, represented by works like [5, 11], adopts feature matching based on descriptor for local mappings. This approach involves extracting redundant feature points and descriptors from images, ensuring an adequate number of correspondences, typically around 1000 [11], and performing matching verification for each point to ensure stable matching. In contrast to the feature matching based approach, the sliding window based method offers performance advantages. It requires extracting only 200~300 points per frame and does not necessitate feature matching. Therefore, our frontend adopts the sliding window based strategy. However, for maintaining the backend map, we still rely on feature-based methods, which will be elaborated on in Sec. 4.2.1.

The sliding-window-based VIO frontend is designed to solve a bundle adjustment problem by incorporating visual-inertial constraints across a sequence of keyframes (KF) and the corresponding observations of landmark. Its objective is to refine the extrinsic parameters of KFs and estimate the states of landmark. The optimization process of the sliding-window frontend encompasses both visual constraints and IMU pre-integration constraints. Specifically, reprojection residuals are formulated as follows:

$$e_r^{k,j} = z_{k,j} - \Pi(K_k R_k^{-1}(x_j - p_k)). \tag{3}$$

The residual for IMU pre-integration has been derived in reference [22, 31], and we ultimately adopt the following form for the keyframe k with its last one:

$$e_i^k = [e_q^k, e_p^k, e_v^k, e_g^k, e_a^k]^\top, \tag{4}$$

Here, $e_q^k = \log(\hat{q}_k^{-1} q_k)$, $e_p^k = p_k - \hat{p}_k$, $e_v^k = v_k - \hat{v}_k$, $e_g^k = b_{g_{k-1}} - b_{g_k}$, $e_a^k = b_{a_{k-1}} - b_{a_k}$. The variables with $\hat{(\cdot)}$ in the state s_k represent the measured motion state obtained by pre-integrating of IMU frame k. Additionally, $\log(\cdot)$ denotes the distance between two quaternions on the Lie-algebra manifold.

In addition, we also consider the marginalization error e_m [35], which is used to simplify the discarded states when keyframes slide out of the window. Then the total residual for sliding-window-based VIO can be described as

$$\arg\min_{\mathscr{F},\mathscr{M}} \sum_{k\in\mathscr{F}} \sum_{j\in\mathscr{M}} \rho\left(\left\| e_r^{k,j} \right\|_{W_r^{k,j}}^2 \right) + \sum_{k\in\mathscr{F}} \left(\left\| e_i^k \right\|_{W_i^k}^2 \right) + \|e_m\|^2. \tag{5}$$

Here, $\mathscr{F} = \{s_i\}$ and $\mathscr{M} = \{x_i\}$ represent the sets of keyframes and landmarks, respectively. $\|\cdot\|_W$ denote the squared Mahalanobis distance with their corresponding covariances.. We set the size of our sliding window as $N_W = 8$ and use a Cauchy kernel function $\rho(\cdot)$ for visual residuals to reduce the impact of outliers, with c controls the width of the kernel, it is defined as

$$\rho(r) = \frac{c^2}{2} \log\left(1 + \left(\frac{r}{c}\right)^2\right). \tag{6}$$

4.2 Backend Map Maintenance

4.2.1 Local Mapping

As described in Sec. 4.1, our approach adopts a lightweight sliding-window-based frontend VIO method, which necessitates the maintenance of map descriptors in the backend to construct a globally consistent map and establish data associations across agents. Consequently, we build a keyframe-based local map on the server side using these descriptors. This strategy aligns with the methods employed in [11, 31], where keyframes from each agent's sliding window are integrated into the local map, and additional 3D landmarks are triangulated to enhance the map's accuracy. For each keyframe received from the frontend, we transmit ORB features along with their corresponding descriptors and other relevant information to the server. Then, we carry out the registration of the nearest keyframe, which is based on 2D matches between image descriptors for initialization and 2D-3D matches between keyframe and point clouds for tracking. Moreover, the descriptors of the 3D points within the map are updated accordingly. Given that the keyframes obtained from the frontend VIO already possess relatively reliable poses, projecting the 3D point clouds onto these keyframes enables us to achieve accurate image matching relationships, thereby enhancing the overall performance of our proposed system in constructing a globally consistent map.

To avoid redundancy in the keyframe database, we have applied certain selection criteria for keyframe selection:

- The current frame can track a proportion p of map points from the previous keyframe.
- The maximum frame interval between the current frame and the previous keyframe is N_G. In our subsequent experiments, we set this to $N_G = 10$.

To maintain the consistency and stability of the local map, we perform local bundle adjustment (LBA), which involves the current frame and its associated covisiability keyframes as well as the map points observable by these keyframes. Covisiability keyframes are defined as frames where the number of covisiability map points exceeds a certain threshold th_{covis}. The optimization objective of LBA is:

$$\underset{\mathcal{K}_L, \mathcal{M}_L^b}{\arg\min} \sum_{k \in \mathcal{K}_L} \sum_{j \in \mathcal{M}_L^b} \rho \left(\left\| e_r^{k,j} \right\|^2_{W_r^{k,j}} \right), \tag{7}$$

where $\mathcal{K}_L = \left\{ {}^m s_i^b \right\}$ represents the set of keyframes in the local map, and $\mathcal{M}_L^b = \left\{ {}^m x_i^b \right\}$ denotes the set of backend local map points. Here, ${}^m(\cdot)$ indicates from the m-th agent, and $(\cdot)_L$ denotes the local map. Unlike in single-agent backend mapping, after completing the fusion of submaps from agents m and n, $\mathcal{K}_L$ and $\mathcal{M}_L^b$ may be shared by the two agents.

4.2.2 Online Map Fusion

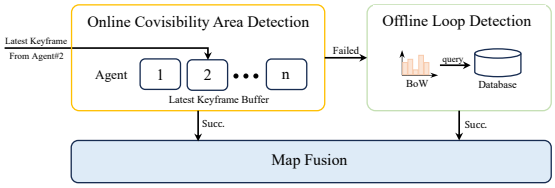

Fig. 3: Online map fusion and offline loop closure in collaborative mapping

Cross-agent map registration and fusion can be achieved using loop detection modules (Sec. 4.3), which is a common practice in previous work [19, 30]. However, we argue that cross-agent map registration based on loop detection often encounters issues such as registration failures or delayed registration. This is because the use of image retrieval based on the bag-of-words model [12] only provides results that are similar in the image-words space, and the presence of considerable noise leads to inefficiencies in the map registration. Therefore, we propose an online map registration and fusion approach based on covisibility area. The term *online* here indicates that the keyframe database at this time consists of "the latest keyframes from other agents," as opposed to historical keyframes based on the bag-of-words model.

As shown in Fig. 3, we maintain a buffer of the latest keyframes from each agent, through which we can also obtain the locally observed map points (MPs). In this module, for each newly arrived keyframe and other keyframes in the buffer, we perform descriptor-based 2D-2D image matching. Based on this cross-agent matching situation, after consistency verification using the RANSAC algorithm and surpassing a predefined matching threshold, we consider it highly likely that agents a and b are observing the same area. We then proceed to a 2D-3D verification stage. For the keyframe pair (KF_a, KF_b) from the two related agents, we attempt cross-agent frame registration using the PnP [15] algorithm to obtain T_a^b, the pose of keyframe KF_a in the local map point constraints of KF_b. Using this transformation for projection matching, we can obtain more geometric constraints between KF_a and KF_b. At this point, it can be assumed that the two agents can register or fuse their maps.

4.3 Loop Detection and Closure

The online map fusion method based on covisibility area proposed in Sec. 4.2.2 is applicable to cross-agent scenarios. For offline map fusion,

Algorithm 1 Online Covisibility-Area-based Map Fusion

```
Buffer = [KF_0, KF_1, ..., KF_n]
function COVISIBILITY MAP REGISTRATION(Buffer)
    [KF_a, KF_b, M_2D2D] ← Search 2D-2D Matching in Buffer
    if len(M_2D2D) > th_2D2D then
        T_KF_a ← PnP(KF_a, MPs_b, M_2D2D)
        KF_a^covis ← CovisiableGraph(KF_a)
        M_2D3D ← ProjectionMatching(KF_a^covis, T_KF_a, MPs_b)
        for MP ∈ MPs_a do
            Fuse(MP, MPs_b)
        end
    end
end function
```

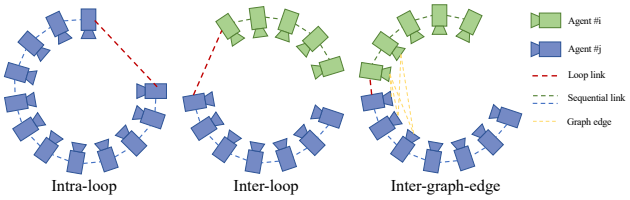

Fig. 4: Loop closure in collaborative mapping

we utilize the bag-of-words model [12] for keyframe database queries, used for loop detection, similar to approaches such as [11, 31]. We identify a candidate set $\mathscr{C}$ by searching all map data for KF_q, and then perform descriptor-based brute-force matching with each keyframe in $\mathscr{C}$. By setting a specific matching quantity threshold, we successfully match KF_m, determining whether KF_q and KF_c are in the same submap. We then execute different map maintenance strategies, as illustrated in the Fig. 4.

Intra-loop: When the matched frame KF_m and the query frame KF_q belong to the same submap, comprehensive optimization of the entire loop is necessary. Initially, we apply a RANSAC-based PnP solution [15] on the map points in KF_q and KF_m, followed by outlier rejection, resulting in an initial correction transformation T_{mq}. Subsequently, we utilize T_{mq} for additional map point projection matching for KF_q and its co-observing frames. Following this, similar to [31], we conduct pose graph optimization, as shown in Fig. 5. Upon completion of the optimization, the associated map points are transformed. Finally, global bundle adjustment is executed.

Inter-loop: In the scenario where the loop frame KF_m and the query frame KF_q originate from different submaps, after completing the fusion of local maps based on the consensus graph, if it is the first cross-agent loop closure, a map fusion strategy is executed. The fused map is then distributed to the local mapping threads of different agents, enabling map sharing between these agents.

The optimization objective of PGO involves the poses of adjacent keyframes, KF i and KF j in the sequence edge set $\mathscr{S}$, while maintaining the relative pose of keyframes in the loop edge set $\mathscr{L}$ fixed, as defined in [31]. Thus, residual constraints can be introduced:

$$e_l^{ij} = \begin{bmatrix} R_i^{-1}(p_j - p_i) - \hat{p}_{i,j} \cdot \\ R_i^{-1} R_j \hat{R}_{ij}^{-1} \end{bmatrix}. \tag{8}$$

$\hat{R}_{ij}$ and $\hat{p}_{ij}$ is obtained from the loop closure. Therefore, the entire pose graph optimization problem can be formulated as the minimization of:

$$\underset{\mathscr{S}, \mathscr{L}}{\arg\min} \sum_{(i,j) \in \mathscr{S}} \left\| e_l^{ij} \right\|^2 + \sum_{(i,j) \in \mathscr{L}} \rho(\left\| e_l^{ij} \right\|^2). \tag{9}$$

For global bundle adjustment, we execute the following objective function:

$$\underset{\mathcal{K}_G, \mathcal{M}_G^b}{\arg\min} \sum_{k \in \mathcal{K}_L} \sum_{j \in \mathcal{M}_L^b} \rho \left(\left\| e_r^{k,j} \right\|^2_{W_r^{k,j}} \right), \tag{10}$$

where $\mathcal{K}_G = \left\{ s_i^b \right\}$ represents the set of keyframes in the global map,

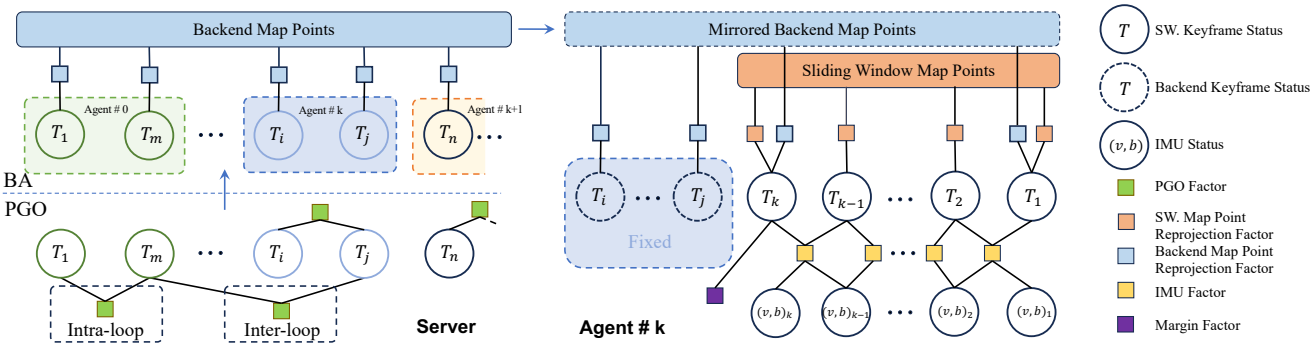

Fig. 5: A schematic diagram illustrating the fusion process of the map state in backend with the frontend sliding window state. On the server side, we first utilize pose graph optimization to constrain the poses between keyframes involved in intra-loop and inter-loop closures. Subsequently, bundle adjustment is performed to achieve joint optimization of keyframes and map points across agents. The state of the local map in backend is then synchronized with the agent's sliding window, completing the correction of drift in the frontend's local tracking. It is important to note that we have simplified the representation of the marginalization factor in the diagram.

and $\mathscr{M}_G^b = \{x_i^b\}$ denotes the set of global map points. We utilize scale-aware frontend information to initialize the backend map. During each global bundle adjustment, we maintain the scale of the backend map by fixing the scale of the initialized window.

As shown in Fig. 3, the two map fusion strategies are not mutually exclusive. Through online covisibility area detection, we can continuously detect potential geometric constraints between the current agents in real time. If none are found, we continue with offline loop detection to update the keyframe database and perform detection.

4.4 Visual-Inertial State Fusion

In related work [30, 34], these collaborative system directly compares the states of the backend's keyframes from global map with the estimated states of the keyframes in the local tracking, thereby estimating the drift of the frontend VIO. Although the keyframes from the backend map undergo drift correction through global optimization, this loosely-coupled behavior for drift correction disrupts the smoothness of trajectory in frontend VIO, which is significantly impacting the AR experience based on the camera-rate output of frontend. Drawing inspiration from works such as [2, 7] we adopt a tightly-coupled approach based on the collaborative consistency map in backend, fusing the states of the global map from backend with those in the sliding window in the frontend. Fig. 6 illustrates a schematic diagram of our approach.

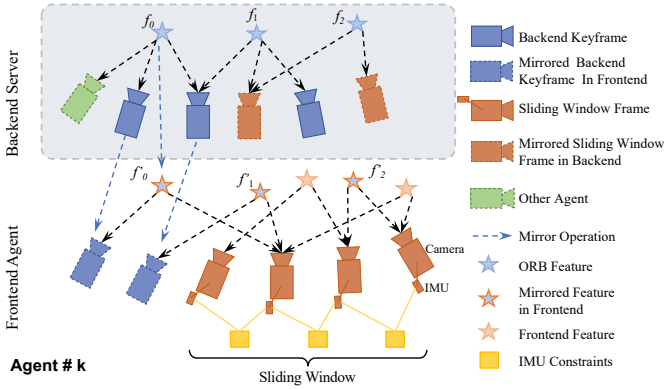

Fig. 6: Illustration the tightly couple strategy of sliding window in frontend with backend information. On the server side, we achieve a more robust map state estimation with multiple agents' observations and integrate it into the frontend sliding window by wirelessly transmitting. This integration aims to constrain the accumulated drift caused by the local tracking in frontend.

In the map registration module, we do not predefine the pivot map. Instead, we merge maps solely based on covisibility area or loop detection schemes. Consequently, for a given agent, its corresponding backend map undergoes global coordinate transformations once it merge with another agent. For convenience, we maintain only one global transformation $(R_G^{C_i}, t_G^{C_i})$ from the backend map to the corresponding frontend i, which is updated upon map fusion. As illustrated in Fig. 6, when features f_i from the backend map are sent to the frontend, they are transformed into $f_i^{'}$. Additionally, the frames observing these features are divided into two categories: those found within the sliding window, denoted as $\mathscr{K}^l$, and those that have slid out of the window, denoted as $\mathscr{K}^g$. The map points are represented as $\mathscr{L}$. At this point, the introduction of the backend map state results in the new residual terms.

$$ {}^b e_r^{k,j} = z_{k,j} - \Pi(K_k R_k^{-1}(R_G^{C_i} x_j^b + p_G^{C_i} - p_k)). \tag{11} $$

Fig. 5 show the factor graph of our sliding widnow optimization with the backend map states, the optimization objective now is updated to:

$$ \begin{aligned} \arg\min_{\mathscr{F},\mathscr{M}} & \sum_{k\in\mathscr{F}}\sum_{j\in\mathscr{M}} \rho\left(\left\|e_r^{k,j}\right\|^2_{W_r^{k,j}}\right) + \sum_{k\in\mathscr{K}^l\cup\mathscr{K}^g}\sum_{j\in\mathscr{L}} \rho\left(\left\|{}^b e_r^{k,j}\right\|^2_{W_r^{k,j}}\right) \\ & + \sum_{k\in\mathscr{F}}\left(\left\|e_i^k\right\|^2_{W_i^k}\right) + \|e_m\|^2 . \end{aligned} \tag{12} $$

Relative to other residual factors, the influence factors of ${}^b e_r^{k,j}$ include the current network traffic status and the mapping quality of the backend, among others. In the case of low mapping quality in the backend, on the one hand, strict observation thresholds are set for the backend map points, while on the other hand, robust kernels are utilized to constrain deviations of the backend map state from the sliding window. Regarding the network traffic status, considering the delay in synchronizing the backend map to the frontend, if the backend state is outdated in the frontend window, we can consider the frontend to have degraded to a completely independent VIO, *i.e.*, as the Eq. 5.

4.5 Communication

We establish TCP full-duplex communication between the agents and the server, similar to [30, 34], for serializing and deserializing shared data and transmitting it over the network in the form of byte streams. On the server, we listen on a predefined port, enabling dynamic connections with an unspecified number of agents.

As shown in Fig. 7, The data flow primarily consists of two types: 1) The frontend VIO transmits keypoint information, feature descriptors, initial poses of the frontend keyframes, and corresponding sliding window information. 2) The backend transmits optimized map states, encompassing mirrored keyframes, mirrored map points, as well as the poses and keypoint positions of corresponding non-sliding window keyframes. A comprehensive analysis of network traffic will be

Table 2: Comparison of Keyframe Accuracy in Different Collaborative Scenarios of EuRoC. The best results are highlighted in bold.

Sequence	VINS	CVI-SLAM	COVINS	Ours
MH01 & MH02	0.148	0.050	**0.065**	0.069
MH02 & MH03	0.204	0.073	0.081	**0.079**
MH04 & MH05	0.178	0.115	0.270	**0.109**
MH01,02,03	0.133	-	0.063	**0.060**
MH01,02,03,05	0.284	0.156	0.125	**0.113**

presented in Sec. 5.4.

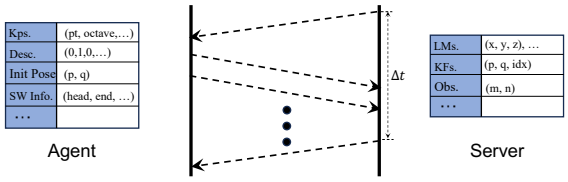

Fig. 7: Illustration of communication module. We establish TCP connections and facilitate data transmission between the agent and the server through the serialization and deserialization of data flow.

5 Experiments

In this section, we conducted a series of experiments to evaluate the robustness of our collaborative SLAM system and the effectiveness of the proposed methods. We assessed the collaborative localization accuracy of the multi-agent system, specifically focusing on the frontend localization accuracy of multiple agents and the precision of backend keyframes localization under consistent mapping conditions. Additionally, to evaluate the proposed tightly coupled strategy with sliding window , we compared the quality of the pose estimation in frontend in a single-machine scenario. We also qualitatively analyzed the proposed online map fusion strategy based on covisibility areas. Furthermore, we systematically analyzed the performance of the multi-agent framework, including network traffic, communication overhead, *etc.*, to discuss the system's scalability. Finally, we deployed the frontend of the system on mobile phone for real-time demonstration, while the backend consistency mapping was performed on a server for real-world test. Our approach is built upon RD-VIO [9, 21], and due to the absence of dynamic scenes in our experimental setup, we removed the dynamic point removal strategy, considering this VIO as our frontend baseline. The proposed method is referred to as RCO-SLAM.

To evaluate on datasets, we deployed the frontend on a personal computer in the same subnet as the server, and data was transmitted wirelessly to the server. Our testing environment is as follows:

- Server: AMD Ryzen 9 7950x 5GHz x 16, 125GB memory, Ubuntu 20.04
- Agent: AMD Ryzen 7 3800x 2.8GHz x 8, 32GB memory, Ubuntu 20.04

In subsequent performance comparisons, unless otherwise stated, we will use the above experimental setup. To eliminate the interference of random factors, we run each set of experiments 5 times and take the average value.

5.1 EuRoC Datasets

We first conducted a comparison between our method and state-of-the-art approaches on the public available EuRoC [4] dataset. The EuRoC dataset serves as a benchmark for VIO and SLAM algorithms. It comprises 11 sequences, including five Machine Hall sequences (`MH_01~MH_05`) and 6 Vicon Room sequences(`V1,V2`). Each sequence provides stereo grayscale images at 20Fps and ADIS16448 IMU data at 200Hz. We use evo [14] to evaluate the accuracy of the estimated trajectory and report the RMSE of absolute pose error (APE) for each method.

Table 3: The evaluation of real-time camera estimated pose on the EuRoC dataset. In this table, we focus on pose estimation at camera rate. we denoted the top 3 result in each sequcne, and underlined to indicate a higher accuracy than the baseline.

Algorithm	V1-01	V1-02	V1-03	V2-01	V2-02	V2-03
ORB-SLAM3 (*offline*)	0.043	0.019	0.031	0.049	0.017	0.027
MSCKF	0.520	0.567	-	0.236	-	-
OKVIS	0.139	0.232	0.262	0.163	0.211	0.291
RD-VIO	0.060	**0.091**	0.168	**0.058**	0.100	0.147
VINS-Fusion (*online*)	0.066	0.287	0.169	0.131	0.226	0.172
ORB-SLAM3 (*online*)	1.624	0.399	1.506	0.343	1.225	1.753
Baseline-VIO	**0.056**	0.101	0.134	0.066	0.089	**0.122**
Ours(*online*)	0.057	0.093	**0.111**	0.059	**0.083**	0.123

Collaborative Mapping: We initially compared the overall accuracy of the entire SLAM system. Adhering to the experimental methodology of CVI-SLAM [19], we evaluated the localization accuracy of keyframes in collaborative mapping within multi-agent scenarios. The compared methods include VINS-Fusion [31], which is not specifically designed for collaborative SLAM, its capability of multi-session make it possible to perform registration within an existing map. Consequently, we executed the sequences sequentially. Since CVI-SLAM is not open-source, we utilized the data from their paper for reference. Additionally, we compared COVINS [34], which alos use the centerized architecture and extends the backend to support collaborative mapping with multiple agents. We designed five collaborative scenarios, each involving 2, 3, or 4 agents. During the final accuracy calculation, we considered the error of the trajectories of multiple agents in the global map. As illustrated in Tab. 2, our results demonstrate close accuracy comparability to COVINS, as exemplified by the `MH_01&MH_02` and `MH_02&MH_03` scenarios, since we employed similar backends for multi-agent collaboration. However, our approach significantly outperforms COVINS in some scenarios, such as `MH_04&MH_05`. These two scenarios in the `MH` sequence are more challenging than previous ones, which we attribute to potential advantages in loop detection and initialization from precise estimates in the frontend, ultimately leading to better convergence during backend optimization.

Tightly-Coupling: We focus on the effectiveness of the proposed tight-coupling scheme involving the state fusion between backend and frontend. Therefore, we evaluated the real-time pose estimation accuracy of the single-agent frontend on the `V1` and `V2` sequences of EuRoC. The compared approaches include MSCKF [27] which are filter-based solutions, while OKVIS [20] and RD-VIO [21] are optimization-based VIO solutions. VINS-Fusion and ORB-SLAM3 are full SLAM solutions with backend maps that provide robust map maintenance capabilities. However, we only utilize their real-time output poses for the final evaluation, denoted with *online*. As shown in Tab. 3, compared to the baseline, we synchronize the backend map state to the frontend sliding window through the tight-coupling scheme, merging it with the frontend sliding window state. The trajectory accuracy of this approach is slightly higher than that of the baseline. We attribute the impact of our tight-coupling scheme on the results to consistency in mapping. When loop closures are not detected, the backend map cannot guarantee global consistency, even if the backend state is synchronized to the frontend. Therefore, it cannot correct drift, leading to insignificant improvements in performance , as observed in `V1_01`, `V2_03`. Instead, due to the noise in the map structure information at the backend, the trajectory accuracy slightly decreases after coupling with the sliding window. However, in collaborative scenarios, frequently loop closures can significantly enhance global map consistency, as demonstrated in the results of Tab. 2. It is important to note that the online trajectory of ORB-SLAM3 exhibits jitter and a significant APE. Our analysis attributes this to ORB-SLAM3's atlas management strategy, where a new map is initialized upon tracking failure, leading to discontinuities in the online trajectory. Additionally, the online pose estimation may suffer from poor convergence due to continuous backend optimization, and global optimization by the backend may cause sudden shifts in the trajectory. Furthermore, we report the accuracy of ORB-SLAM3's of-

Table 4: The accuracy of frontend tracking on the VICON dataset. The best results are highlighted in bold.

Sequence	VINS	COVINS	Ours(Baseline)	Ours(w/o. CA)	Ours
scene 1	0.150	**0.090**	0.114	0.091	0.096
scene 2	0.195	0.380	0.264	0.095	**0.094**
scene 3	0.295	0.183	0.201	0.192	**0.171**
scene 4	0.115	0.071	0.110	0.103	**0.069**
scene 5	0.128	0.093	0.094	0.093	**0.085**

fline trajectory, which is denoted *offline*, acknowledging the high APE RMSE at this stage. But it is essential to recognize that post-optimized poses hold little significance for real-time AR applications.

5.2 Multi-User AR Datasets

The EuRoC dataset captures indoor drone movements, ensuring different sequences within the same environment. However, it lacks collaborative observations similar to covisibility areas in AR interactive scenes. Therefore, we collected sequences under 5 settings, each consisting of 3 segments, simulating scenarios with 3 agents. Additionally, we utilized VICON motion capture to obtain the motion trajectories of the recording devices, enabling post-processing to acquire ground truth body/camera trajectories. We setup a collaborative system based on baseline VIO, denoted as Ours(Baseline). To validate the covisibility-area-based map fusion strategy, we conducted a control experiment by removing the online covisibility-area-based map fusion strategy, denoted as (w/o. CA). As the result shown in Tab. 4, in the simplest scenario, scene 1, multiple cameras rotate around a central scene, and a sufficient number of co-observable constraints lead to both our approach and COVINS [34] achieving similar levels of result accuracy. We observed that COVINS performed poorly on scene 2, experiencing erroneous optimizations during pose graph optimization, causing trajectories from different agents to deviate increasingly from the ground truth. Our final approach slightly outperformed the version without covisibility area detection, indicating that in this scenario, the covisibility-area-based approach could add more constraint edges to the fused map which across agents, thereby enhancing global map consistency. However, this indirect enhancement led to limited improvements in frontend accuracy.

5.3 Qualitative Comparison

5.3.1 Collaborative Mapping

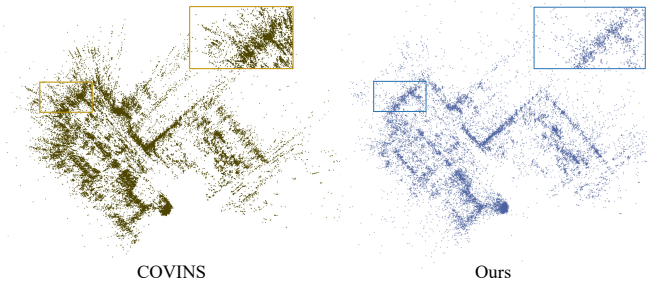

Fig. 8: Illustration of collaborative mapping in Machine Hall sequences for 3 agents. The top view of the hall.

In the EuRoC `MH` sequences, we utilized `MH_01`, `MH_02` and `MH_03` to perform collaborative mapping. All agents initiated from a stationary position on a wooden board, completed the local area motion in the hall, and then returned to the starting point, collaborating during this process to reconstruct the cross-agent map. Similar to COVINS, we were able to achieve cross-agent map registration and fusion during the brief motion on the wooden board. However, our point cloud structure was more prominent than that of COVINS, particularly in areas such as the wall, as shown in Fig. 8. This area, due to being observed from a very narrow baseline range by only `MH_01` and `MH_02`, exhibited significant depth uncertainty, resulting in a cluttered point cloud structure. Nevertheless, through a more flexible map fusion strategy, we were able to maximize the utilization of cross-agent co-observations to constrain map points, thereby obtaining a more consistent map structure.

5.3.2 Online Covisiability-Area-based Loop Detection

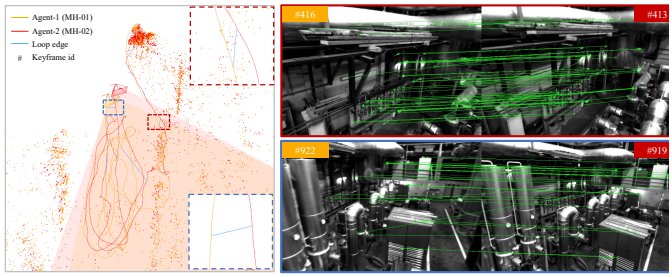

Fig. 9: Illustration of the online covisibility-area detection and the feature matching of image pairs from the latest keyframe buffer.

To validate the effectiveness of the online covisibility-area-based map fusion strategy proposed in Sec 4.2.2, we conducted a qualitative test in a collaborative scenario involving two agents, `MH_01` and `MH_02`. The two agents had long-term real-time covisibility fields and consistent motion areas, which provided an ideal test environment for the proposed strategy. To mitigate the impact of bag-of-words model-based loop detection, we disabled the offline loop closure module. This allowed us to focus solely on the performance of the covisibility-area-based approach. The results of the test showed that the online approach was able to sensitively achieve cross-agent place recognition, thereby timely completing map fusion. Fig. 9 illustrates the results of the test. The left map shows the covisibility areas of the two agents, while the right image pair depicts the latest keyframe in buffer when a similar area is recognized. We also visualized the 2D2D ORB-based feature matching, which revealed a few outliers. However, these outliers can be robustly filtered out during the later RANSAC stage.

5.4 System Performance

In addition to pursuing higher localization accuracy and more consistent map structures, collaborative SLAM systems also pay close attention to the overall system performance.

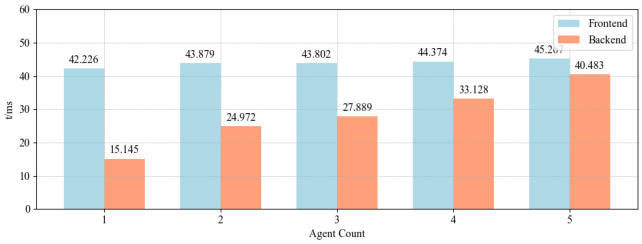

Fig. 10: The relationship between the time cost of frame tracking in frontend and keyframe registration in backend collaborative mapping with the number of participated agents.

We conducted a comparative analysis of the temporal expenditure associated with frame tracking in the frontend and keyframe registration in the backend collaborative mapping, with varying quantities of agents within the system. We test on `MH` sequences and the result is depicted in Fig. 10.As the number of agents increasing, the average time required for tracking in the frontend maintains a stable range, approximately around 45ms. Conversely, the time expenditure for backend keyframe registration at the server exhibits a linear increase. This trend is attributed to the server's necessity to initiate additional threads to accommodate requests from an increasing number of agents, consequently leading to a linear decline in overall system performance.

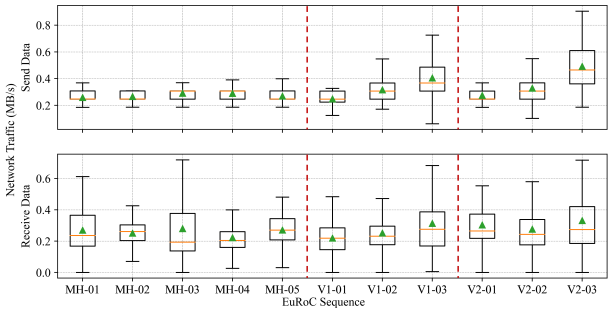

Fig. 11: The network traffic at the agent side in the 3 collaborative scenarios, where the green arrows represent the mean values.

Table 5: The average network traffic between the server side and the agent side in single and multiple scenarios. The unit is kB/s.

Sequence	COVINS [34]		Ours	
	Agent → Server	Server → Agent	Agent → Server	Server → Agent
Avg. (8 seq.)	493.36	2.31	276.8	259.56
MH1	422.83	2.29	258.52	269.97
MH5	540.37	2.32	269.56	270.19
V103	609.22	2.31	405.26	313.33

We analyzed the network traffic at the agent side in three collaborative scenarios, as shown in Fig. 11, where the `MH` sequences involves 5 agents, and `V1`, `V2` involve 3 agents each. In each scenario, as the index of the agents increases, their motion patterns become more challenging (*e.g.*, vigorous movements, rapid rotations). We observed that as the agents' movements become more vigorous, they require sending more data (in `V1` and `V2`), whereas the agents' movements in `MH_01~MH_05` are relatively gentle, resulting in similar amounts of network traffic. However, the received data does not exhibit a specific correlation with the agents' motion patterns, generally ranging between 200~400KB. This can be attributed to the fact that the frontend VIO must promptly synchronize complex scene variations with the server during tracking, resulting in increased network traffic. Furthermore, we compared the average traffic and individual traffic of specific sequences with COVINS [34], as demonstrated in the Tab. 5. Our network traffic for communication from agents to the server is lower compared to COVINS. This is due to the fact that while COVINS necessitates synchronizing the total frontend map points with the backend, our approach only requires synchronizing the initial state of frontend keyframes with the backend.

The tightly coupled approach we propose requires timely states synchronization between the backend map and the frontend sliding window. If the backend map fails to synchronize with the frontend in time, the capability of drift correction of backend map observation constraint term in the optimization objective Eq. 12 is lost. At this point, the constraints on the sliding window state degrade to Eq. 5. We conducted experiments to investigate the impact of such synchronization delays on the final accuracy of the frontend trajectory. As illustrated in the Fig. 7, we set the time interval Δt for server-side synchronization and tested the single-agent trajectory accuracy under different Δt conditions. As shown in Tab. 6, when Δt is sufficiently large (200ms), the frontend sliding window state tends to degrade into local odometry tracking, thereby affecting accuracy due to synchronization delay. However, setting a lower delay allows timely state synchronization, thereby benefiting the tracking accuracy of the frontend through the coupled strategy.

Table 6: Frontend tracking performance of a single agent under different synchronization delay settings. The best results are highlighted in bold.

Delay(ms)	MH01	MH03	MH05	V102	V202
Δ t=30	**0.104**	0.129	**0.237**	**0.090**	0.089
Δ t=50	**0.104**	**0.125**	0.241	0.096	**0.088**
Δ t=100	0.106	0.133	0.239	0.093	0.090
Δ t=200	0.109	0.131	0.244	0.094	0.093

5.5 AR Demo

Fig. 12: Collaborative AR in a multi-user setting. We employed three iPhone 12 and placed a virtual *Ferrair* on the table. By capturing different perspectives simultaneously, we could confirm that the multi-agent system had successfully completed map fusion.

To better illustrate the practical application value of our approach, we implemented an AR demo based on the iOS. We deploy the frontend VIO on three iPhone 12, with a personal computer (the Agent mentioned in experimental configuration) serving as the server running the backend collaborative mapping service. We demonstrated a multi-user AR scene involving three users, as shown in Fig. 12. As there are no other open-source works available for collaborative AR on mobile devices, we solely presented our own results. Due to the lack of ground truth, we relied on the relative position of AR objects with respect to the static environmental background to intuitively assess the reliability of pose estimation. During the demonstration, leveraging a map fusion strategy based on online covisibility areas enabled direct mutual registration of multi-agent submaps. At the same time, we demonstrated the registration of virtual *Ferrair* from the perspectives of different users (User #1, #2, #3) in the global map, as well as the registration from the viewpoints (View A, View B) of User #3, showcasing the consistency of the global map across agents and the accuracy of local pose estimation. We capture AR demonstration through screen recording and confirm the smooth operation of the application on the mobile device from a third-person perspective. All demonstrations are presented in real time. Please refer to our supplementary materials for further details.

6 Conclusion

We have proposed a centralized multi-agent collaborative SLAM system tailored for mobile AR, capable of achieving precise real-time 6-DoF pose estimation at the frontend and efficient, consistent collaborative mapping at the backend. Compared to state-of-the-art methods, our approach achieves better accuracy and robustness. Furthermore, we demonstrate the effectiveness of the system by deploying the proposed system on both mobile devices and servers. We also discuss the system's scalability, which exhibits a certain level of robustness when integrating more agents into the system. We believe that this work contributes to interactive AR experiences based on collaborative SLAM and will also explore different system architectures to enhance the robustness of collaborative systems in the future.

However, due to the constraints in network bandwidth and server concurrency performance in our experiments, our system has a restricted capacity of connecting agents, which limits simultaneous access for a larger user base. Therefore, enhancing network bandwidth capacity and server concurrency performance is crucial for further improving the system's scalability. Additionally, our method works under the assumption of static world for tracking and mapping. Achieving the collaborative SLAM in dynamic scene will be our future work.

7 Acknowledgments

This work was partially supported by NSF of China (No. 61932003). The authors would like to thank Tianxing Fan for his kind help in demo recording and advice on paper formatting.

REFERENCES

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56:221–255, 2004. 4

[2] H. Bao, W. Xie, Q. Qian, D. Chen, S. Zhai, N. Wang, and G. Zhang. Robust tightly-coupled visual-inertial odometry with pre-built maps in high latency situations. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2212–2222, 2022. 2, 6

[3] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072, 2017. 2

[4] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016. 2, 7

[5] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 2, 4

[6] Y. Chang, Y. Tian, J. P. How, and L. Carlone. Kimera-multi: a system for distributed multi-robot metric-semantic simultaneous localization and mapping. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11210–11218. IEEE, 2021. 2, 3

[7] G. Cioffi and D. Scaramuzza. Tightly-coupled fusion of global positional measurements in optimization-based visual-inertial odometry. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5089–5095. IEEE, 2020. 2, 6

[8] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 10 2008. 3

[9] X. Contributors. Openxrlab visual-inertial SLAM toolbox and benchmark. https://github.com/openxrlab/xrslam, 2022. 7

[10] B. Dong and K. Zhang. A tightly coupled visual-inertial gnss state estimator based on point-line feature. *Sensors*, 22(9):3391, 2022. 2

[11] R. Elvira, J. D. Tardós, and J. M. Montiel. ORBSLAM-Atlas: a robust and accurate multi-map system. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6253–6259. IEEE, 2019. 2, 3, 4, 5

[12] D. Galvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012. doi: 10.1109/TRO.2012.2197158 5

[13] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang. OpenVINS: A research platform for visual-inertial estimation. In *IEEE International Conference on Robotics and Automation*, pp. 4666–4672. IEEE, 2020. 2

[14] M. Grupp. evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017. 7

[15] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pp. 383–390, 2011. doi: 10.1109/ICCV.2011.6126266 5

[16] Z. Huai and G. Huang. Robocentric visual-inertial odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6319–6326. Madrid, Spain, 2018. 2

[17] H. Huang, H. Ye, J. Jiao, Y. Sun, and M. Liu. Geometric structure aided visual inertial localization. *arXiv preprint arXiv:2011.04173*, 2020. 2

[18] H. Huang, H. Ye, Y. Sun, and M. Liu. Gmmloc: Structure consistent visual localization with gaussian mixture models. *IEEE Robotics and Automation Letters*, 5(4):5043–5050, 2020. 2

[19] M. Karrer, P. Schmuck, and M. Chli. CVI-SLAM—collaborative visual-inertial SLAM. *IEEE Robotics and Automation Letters*, 3(4):2762–2769, 2018. 2, 3, 5, 7

[20] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. 2, 7

[21] J. Li, X. Pan, G. Huang, Z. Zhang, N. Wang, H. Bao, and G. Zhang. RD-VIO: Robust visual-inertial odometry for mobile augmented reality in dynamic environments. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2, 3, 4, 7

[22] P. Li, T. Qin, B. Hu, F. Zhu, and S. Shen. Monocular visual-inertial state estimation for mobile augmented reality. In *IEEE international symposium on mixed and augmented reality*, pp. 11–21. IEEE, 2017. 2, 4

[23] X. Liu, W. Ye, C. Tian, Z. Cui, H. Bao, and G. Zhang. Coxgraph: multi-robot collaborative, globally consistent, online dense reconstruction system. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8722–8728. IEEE, 2021. 3

[24] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, pp. 674–679, 1981. 3

[25] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems*, vol. 1, p. 1, 2015. 2

[26] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison. Gaussian splatting SLAM. *arXiv preprint arXiv:2312.06741*, 2023. 2

[27] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 3565–3572, 4 2007. 2, 7

[28] T.-M. Nguyen, S. Yuan, M. Cao, T. H. Nguyen, and L. Xie. Viral SLAM: Tightly coupled camera-imu-uwb-lidar SLAM. *arXiv preprint arXiv:2105.03296*, 2021. 2

[29] M. Ouyang, X. Shi, Y. Wang, Y. Tian, Y. Shen, D. Wang, P. Wang, and Z. Cao. A collaborative visual SLAM framework for service robots. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8679–8685. IEEE, 2021. 3

[30] M. Patel, M. Karrer, P. Bänninger, and M. Chli. Covins-g: A generic back-end for collaborative visual-inertial SLAM. *arXiv preprint arXiv:2301.07147*, 2023. 2, 3, 5, 6

[31] T. Qin, P. Li, and S. Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. 2, 3, 4, 5, 7

[32] L. Riazuelo, J. Civera, and J. M. Montiel. C2tam: A cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems*, 62(4):401–413, 2014. 3

[33] P. Schmuck and M. Chli. CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics*, 36(4):763–781, 2019. 2, 3

[34] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli. Covins: Visual-inertial SLAM for centralized collaboration. In *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 171–176. IEEE, 2021. 2, 3, 6, 7, 8, 9

[35] G. Sibley, L. Matthies, and G. Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010. 4

[36] Y. Tang, J. Zhang, Z. Yu, H. Wang, and K. Xu. Mips-fusion: Multi-implicit-submaps for scalable and robust online neural rgb-d reconstruction. *ACM Transactions on Graphics (TOG)*, 42(6):1–16, 2023. 2

[37] H. Wang, J. Wang, and L. Agapito. Co-SLAM: Joint coordinate and sparse parametric encodings for neural real-time SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13293–13302, 2023. 2

[38] H. Xu, P. Liu, X. Chen, and S. Shen. D^2-SLAM: Decentralized and distributed collaborative visual-inertial slam system for aerial swarm. *arXiv preprint arXiv:2211.01538*, 2022. 2, 3

[39] H. Xu, Y. Zhang, B. Zhou, L. Wang, X. Yao, G. Meng, and S. Shen. Omni-swarm: A decentralized omnidirectional visual–inertial–uwb state estimation system for aerial swarms. *IEEE Transactions on Robotics*, 38(6):3374–3394, 2022. 2, 3

[40] X. Yang, H. Li, H. Zhai, Y. Ming, Y. Liu, and G. Zhang. Vox-Fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *IEEE International Symposium on Mixed and Augmented Reality*, pp. 499–507. IEEE, 2022. 2

[41] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad. An overview to visual odometry and visual SLAM: Applications to mobile robotics. *Intelligent Industrial Systems*, 1(4):289–311, 2015. 2

[42] P. Zhu, Y. Yang, W. Ren, and G. Huang. Cooperative visual-inertial odometry. In *2021 ieee international conference on robotics and automation (icra)*, pp. 13135–13141. IEEE, 2021. 3

[43] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. NICE-SLAM: Neural implicit scalable encoding for SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12786–12796, 2022. 2

[44] D. Zou and P. Tan. CoSLAM: Collaborative visual slam in dynamic environments. *IEEE transactions on pattern analysis and machine intelligence*, 35(2):354–366, 2012. 3

[45] X. Zuo, P. Geneva, Y. Yang, W. Ye, Y. Liu, and G. Huang. Visual-inertial localization with prior lidar map constraints. *IEEE Robotics and Automation Letters*, 4(4):3394–3401, 2019. 2