ICJK
Chinese Journal of Information Fusion
http://dx.doi.org/10.62762/CJIF.2026.632794

RESEARCH ARTICLE

Design and Verification of a Centralized Multi-Agent Visual SLAM System Based on Cloud Native Architecture

Lian Yu[1], **Dongsheng Li**[1], **Guoyan Wang**[1,*] **and Fei Zhao**[1]

[1] College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China

Abstract

To alleviate the computational and memory pressure faced by resource-constrained edge platforms in large-scale homogeneous multi-agent visual SLAM tasks, this paper designs and validates a centralized homogeneous multi-agent visual SLAM system based on a cloud-native architecture. The system adopts a cloud-edge functional decoupling strategy: real-time perception modules, such as visual odometry, keyframe extraction, and local tracking, are retained on the edge side, while computationally intensive tasks, including loop-closure detection, pose graph optimization, global map management, and multi-source map fusion, are centrally executed in the cloud. To accommodate bandwidth fluctuations, transmission latency, and short-term packet loss, the system establishes a ROS-based bidirectional asynchronous communication mechanism and combines keyframe-level compression, serialization, and ACK confirmation feedback to support reliable keyframe uploading. Meanwhile, an edge-side resource management workflow based on "upload–acknowledge–release" is designed to limit the growth of unacknowledged keyframe caches and improve long-term operational stability. For map fusion, the system adopts probabilistic occupancy-grid updating based on log-odds representation and, under homogeneous sensor conditions and constraints from globally optimized poses, projects local maps from multiple agents into a shared coordinate system for normalized equal-weight fusion. Experiments on public datasets, Gazebo collaborative simulations, baseline comparisons, communication-constrained tests, and long-horizon stress tests show that, under controlled experimental conditions, the proposed system reduces the edge-side load while maintaining localization accuracy and improves the stability of cloud-side map updating and long-term operation. Communication-disturbance experiments further analyze the influence of bandwidth limitation, transmission latency, random jitter, packet loss, and short-term outage on system performance, confirming the operational adaptability of the system under weak-network conditions.

Keywords: cloud-native architecture, multi-agent SLAM, map fusion, edge computing.

Academic Editor:
Yulong Huang

Submitted: 16 January 2026
Revised: 25 June 2026
Accepted: 10 July 2026
Published: 28 September 2026

Vol. 3, **No.** 3, 2026.
10.62762/CJIF.2026.632794

***Corresponding author:**
✉ Guoyan Wang
wangguoyan@nudt.edu.cn

Citation
Yu, L., Li, D., Wang, G., & Zhao, F. (2026). Design and Verification of a Centralized Multi-Agent Visual SLAM System Based on Cloud Native Architecture. *Chinese Journal of Information Fusion*, 3(3), 209–225.

1 Introduction

Simultaneous Localization and Mapping (SLAM) is one of the core technologies that enables mobile agents to achieve autonomous navigation and environmental perception [1–5]. With advances in computer vision, visual SLAM has been widely applied in unmanned aerial vehicle inspection, warehousing logistics, post-disaster rescue, and other fields because of its low sensor cost and its ability to provide rich environmental information, such as texture and color [6–9]. However, because a single agent is limited in both sensing range and computational capability, its operational efficiency in complex environments often cannot meet task requirements. Therefore, multi-agent systems (MASs) exploit parallel exploration, information sharing, and task redundancy to achieve higher search efficiency and robustness than single-agent systems in large-scale, highly dynamic, or time-sensitive scenarios [10].

In practical deployment, however, high-precision multi-agent visual SLAM faces severe computational resource constraints [11]. Visual SLAM is intrinsically computation-intensive; as the exploration area expands, the computational complexity of backend nonlinear optimization and loop-closure detection increases rapidly [2–4, 12]. This creates a dilemma for edge devices such as UAVs and small service robots: due to size, weight, and power (SWaP) constraints, such devices cannot accommodate high-performance computing units and therefore struggle to maintain high-precision real-time mapping during long-duration tasks [13].

Existing solutions to these challenges mainly follow two directions: hardware enhancement and algorithmic lightweighting. Both approaches involve inherent trade-offs. On the one hand, integrating high-performance computing modules, such as the NVIDIA Jetson series, can substantially improve data processing capability, but it also inevitably increases power consumption and physical payload, seriously limiting the endurance of miniature mobile platforms. On the other hand, reducing computational load through sparse feature extraction or network pruning often sacrifices the ability to perceive environmental details, which significantly degrades localization robustness in texture-poor or complex dynamic environments [14].

Because single-machine hardware has inherent physical limitations and algorithmic lightweighting has an upper bound in accuracy, reconstructing the computing paradigm at the system-architecture level has become an important path for addressing this problem. In recent years, the rise of cloud robotics has provided a new approach for overcoming edge-computing bottlenecks [15–19]. By offloading computationally intensive tasks, such as global map optimization and multi-agent map fusion, to a cloud platform with stronger computing resources, the edge side can focus on latency-sensitive lightweight perception. This strategy achieves a better balance between computational capability and energy consumption. In addition, for multi-agent systems, a centralized cloud architecture naturally provides a global perspective, which helps resolve inter-agent data association and global consistency more efficiently and avoids the high bandwidth cost and data conflicts often encountered in distributed communication.

To address performance degradation and runtime interruptions caused by limited computing and storage resources on edge platforms during long-duration and large-scale multi-agent visual SLAM tasks, this paper designs and implements a cloud-edge collaborative centralized multi-agent mapping framework. By migrating computationally intensive components such as backend optimization and loop-closure detection to the cloud for unified processing, the proposed architecture improves scalability and engineering feasibility at the system level. The main contributions of this paper are as follows:

(i) A cloud-native hierarchical cloud-edge multi-agent SLAM framework is designed and implemented. Following the principle of "lightweight perception at the edge and centralized computation in the cloud," the framework offloads key computation-intensive modules, including backend optimization and loop-closure detection, to the cloud and decouples system components in a modular manner to improve deployment flexibility and scalability.

(ii) A data transmission and device-side resource release mechanism for weak-network environments is constructed. Based on ROS bidirectional asynchronous communication, the system combines keyframe-level compressed transmission and ACK feedback to enhance adaptability to bandwidth fluctuations, transmission latency, and short-term packet loss. The corresponding cache is released after cloud-side acknowledgment, thereby improving continuous operation stability.

(iii) **A probabilistic occupancy grid fusion process for cloud-side unified mapping is constructed.** This process is coupled with cloud-side global pose optimization, projects local observations from multiple agents into a unified coordinate frame, and adopts normalized equal-weight fusion under homogeneous sensor conditions, thereby providing a stable and interpretable map representation for system-level collaborative mapping.

(iv) **The proposed architecture is evaluated on public benchmark datasets, Gazebo collaborative simulations, baseline comparisons, communication-constrained tests, and long-horizon stress tests.** Runtime efficiency, localization accuracy, and operational stability in small-scale multi-agent scenarios are systematically examined, demonstrating the engineering feasibility of the proposed framework in resource-constrained scenarios.

The rest of this paper is organized as follows. Section 2 reviews research related to this work, focusing on embedded lightweight methods, cloud-edge distributed architectures, and resource scheduling and adaptive strategies. Section 3 introduces the proposed cloud-native cloud-edge collaborative framework, describes the layered design of the edge perception layer, data transmission layer, and cloud processing layer, and further presents the probabilistic occupancy grid fusion mechanism and its global coordinate alignment and merging process. Section 4 validates the proposed system in terms of runtime efficiency, localization accuracy, collaborative mapping performance, adaptability under communication constraints, and long-term operational stability. The experiments include evaluations on the TUM and EuRoC public datasets, Gazebo multi-agent collaborative mapping experiments and comparisons with CCM-SLAM and Swarm-SLAM, communication-constrained tests, loop-closure ablation experiments, and long-horizon stress tests under limited memory. Finally, Section 5 summarizes the conclusions and discusses current limitations and future work.

2 Related Work

Given the limited computational resources of terminal devices, existing studies mainly focus on three aspects: lightweight algorithms for edge platforms, distributed computing architectures, and resource scheduling with adaptive strategies.

2.1 Lightweight Research on Embedded Edge Platforms

Constrained by the physical payload and power budget of platforms such as micro-UAVs and service robots, namely SWaP constraints, early studies mainly attempted to reduce the runtime overhead of SLAM algorithms on the edge side. Yanik and Ilgin [20] benchmarked the computational cost of mainstream visual SLAM methods, showing that maintaining both high-frame-rate tracking and high-precision mapping on embedded platforms has a performance boundary. To address this issue, Em-SLAM proposed by Wu et al. [21] achieves faster monocular localization on embedded systems by improving initial pose estimation and iterative optimization strategies. In addition to algorithmic logic optimization, data-level dimensionality reduction is also a feasible approach. Picard et al. [22] explored image quantization technology and showed that reducing the bit width of input data can effectively decrease data throughput and alleviate edge-side storage pressure. Khalufa et al. [23] introduced a runtime adaptive mechanism that adjusts parameters according to motion status to balance accuracy and energy consumption. These works reduce edge-side computational pressure to some extent, but because they are limited by local hardware capacity, they still cannot independently support large-scale or multi-input global map maintenance tasks.

2.2 Distributed Architectures for Edge and Cloud

To address the limitation of single-device resources, migrating computationally intensive tasks to servers has become an important technical trend. Eger et al. [24] evaluated a task-allocation strategy in which frontend tracking is retained locally and backend mapping is offloaded to the cloud, verifying the feasibility of distributed processing. To further optimize offloading efficiency, Sossalla et al. [25] performed parameter-sensitivity analysis on an Edge SLAM system; by parallelizing the map update process and optimizing the number of extracted feature points and the keyframe packaging strategy, they significantly reduced offloading latency and improved robustness under network fluctuations. Xu et al. [26] designed the EdgeSLAM framework, which uses edge servers to assist semantic segmentation and mapping. To address interference in dynamic environments, FAES-SLAM proposed by Gao et al. [27] uses edge computing resources for target detection and removal. Li et al. [18] analyzed data desynchronization that may be caused by frontend and backend

decoupling in edgeSLAM2 and discussed schemes for mitigating latency using on-chip intelligence. Existing distributed-architecture studies mainly focus on verifying the effectiveness of computational offloading under single-device or point-to-point connections; more concrete system-level implementations are still needed to support centralized processing and fusion after multi-channel data uploads.

2.3 Resource Scheduling and Adaptive Strategies

In bandwidth-constrained network environments, efficient allocation of computing and communication resources is a current research focus. AdaptSLAM proposed by Chen et al. [28] reduces network load by selecting keyframe subsets for transmission based on uncertainty minimization. Zhang et al. [29] proposed a joint offloading and scheduling orchestration algorithm that uses predicted regional feature importance to cope with input randomness. The main contributions of this type of work lie in scheduling-algorithm optimization and theoretical modeling. In practical engineering applications, however, building a system that can operate stably while completing multi-channel data uploads and cloud-side map fusion remains an engineering problem that must be solved for multi-agent deployment.

In summary, compared with existing collaborative SLAM methods, this paper differs in system architecture, resource management mechanism, and map representation and fusion strategy. First, methods such as DOOR-SLAM and Swarm-SLAM mainly adopt distributed or decentralized collaborative architectures, focusing on inter-robot loop-closure detection, information exchange, outlier-constraint rejection, and collaborative optimization. In contrast, this paper adopts a cloud-edge collaborative architecture with centralized cloud processing. The edge side serves as a lightweight perception and keyframe-upload node, while loop-closure detection, pose graph optimization, global map management, and multi-source map fusion are centrally deployed in the cloud. This centralized design enables cloud computing resources to undertake global optimization and map maintenance, reduces sustained computation and caching pressure on the edge side, and avoids frequent complex distributed optimization, data synchronization, and consistency maintenance among resource-constrained nodes.

Second, centralized collaborative SLAM methods such as CCM-SLAM also manage multi-robot keyframes, MapPoints, and pose graphs through a central server and perform map matching, sparse map fusion, and global optimization. Compared with these methods, this paper further targets homogeneous multi-agent scenarios by dividing system functions into edge-side perception/uploading and cloud-side centralized processing. It combines keyframe-level compressed transmission, ACK feedback, and the "upload–acknowledge–release" edge-side resource management workflow to control the growth of unacknowledged keyframe caches and improve long-term operational stability.

In addition, this paper differs in map representation and fusion. Existing collaborative SLAM methods mostly use sparse pose graphs, keyframes, and feature points as the main objects for optimization and fusion, focusing on maintaining multi-robot trajectory consistency and optimizing sparse maps. This paper instead focuses on the cloud-side global map construction process. After multi-agent pose estimation and global optimization, the local maps generated by different agents are projected into a unified coordinate system, and a global map is generated through probabilistic occupancy-grid updating and normalized equal-weight fusion. This representation unifies local spatial observations from different agents into an accumulative and updateable map representation, facilitates cloud-side multi-source map fusion, and visually presents the coverage relationships and fusion results of different local maps in the global coordinate system.

3 Methodology

To address the difficulty of performing large-scale mapping on edge devices under SWaP constraints, this paper proposes a cloud-native cloud-edge collaborative layered architecture. The architecture follows the design principle of "lightweight perception at the edge and centralized computation in the cloud" and uses containerization technology to decouple the functional modules of the SLAM system into independent microservices. The overall system architecture is shown in Figure 1 and consists mainly of an edge perception layer, a data transmission layer, and a cloud processing layer.

3.1 Cloud-Edge Framework

Edge perception layer: The edge side is designed to ensure real-time performance and low power consumption. This layer runs on the onboard computing unit of each agent and is mainly

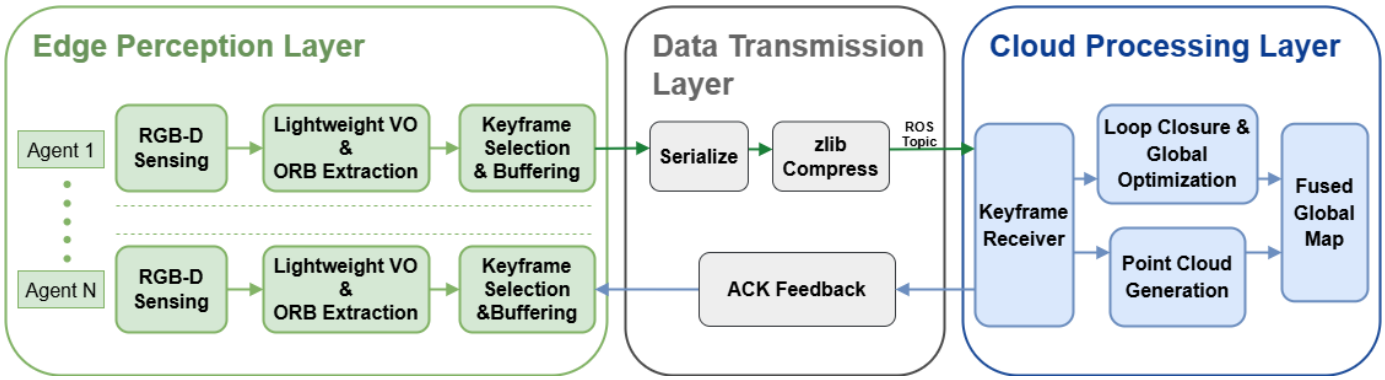

Figure 1. Flowchart of the multi-agent SLAM system based on a cloud-native architecture.

responsible for high-frequency sensor-data acquisition and preprocessing. Through a lightweight visual odometry algorithm, the edge side estimates the relative pose between adjacent frames in real time and extracts keyframes and their corresponding ORB feature descriptors. To reduce transmission bandwidth, the edge side maintains only a compact sliding-window local map; after the cloud completes reception and returns an ACK confirmation, the edge side releases the corresponding keyframe transmission cache.

Data transmission layer: The system adopts keyframe-level data uploading instead of continuous raw-image streaming. After frontend tracking and keyframe selection, the edge side serializes observations such as keyframe RGB-D data and camera poses into a unified data packet and sends it to the cloud through an upload node. To reduce network transmission pressure, the upload node supports keyframe-packet compression and assigns a sequence number to each packet for acknowledgment management. After receiving a packet, the cloud backend decodes, decompresses, and reconstructs the keyframe, and then returns an acknowledgment message after successful reception. The edge side removes the corresponding unacknowledged packet according to the acknowledgment message, forming a "keyframe upload–cloud acknowledgment–edge-side release" resource management workflow. Through this mechanism, the system can reliably transmit the key data required for backend optimization and map construction to the cloud while keeping the edge side lightweight.

Assume that the sliding window maintained at the edge contains W keyframes, and let Q_t denote the number of keyframes in the transmission queue at time t that have not yet been acknowledged by the cloud. The total edge-side memory footprint can be approximated as

$$M_{\text{edge}}(t) = M_0 + W \cdot m_{\text{win}} + Q_t \cdot m_{\text{kf}} \tag{1}$$

where M_0 denotes the system base memory footprint and m_{win} denotes the average memory cost of a single keyframe and its associated local map within the sliding window.

To make the memory model more interpretable, m_{win} is highly correlated with the density of features extracted by the frontend and can be approximated as

$$m_{\text{win}} \approx m_{\text{pose}} + N_{\text{feat}} \cdot m_{\text{point}} \tag{2}$$

where N_{feat} is the number of ORB features extracted per frame, and m_{point} includes the storage required for the coordinates and binary descriptor of a single 3D MapPoint. In the architecture proposed in this paper, dense probabilistic occupancy grid map fusion and updating are performed in the cloud, so m_{win} is not directly determined by global map resolution. Map resolution affects this term only indirectly when the edge side additionally caches local grid submaps or extra image buffers. The term m_{kf} denotes the actual memory footprint of a single keyframe message after serialization and zlib compression when it enters the transmission queue.

The dynamic update model of the keyframe buffer queue can be expressed as

$$Q_{t+1} = \max(0, Q_t + a_t - u_t) \tag{3}$$

In this model, the physical meanings and triggering mechanisms of the variables are defined as follows. First, $a_t \in \{0, 1\}$ is an indicator function denoting whether a new keyframe is generated and pushed into the queue at time step t. To ensure mapping quality and avoid injecting redundant data into the network, a_t is jointly determined by the magnitude of motion change and the current tracking quality. A

new keyframe is triggered when any of the following conditions is met: the relative translation of the agent exceeds Δd_{th}, the relative rotation exceeds $\Delta\theta_{\text{th}}$, or the inlier ratio of feature matching between the current frame and the local map falls below the threshold τ_{track}.

Second, $u_t \in \mathbb{N}$ denotes the number of keyframes that have been successfully uploaded and acknowledged during time step t. To ensure robustness under weak-network conditions, the system introduces an acknowledgment-driven queue management mechanism: a keyframe is counted in u_t only when the edge side receives the keyframe reception and insertion acknowledgment returned by the cloud within the time window T_{ack}; the corresponding item is then removed from the transmission buffer queue, and its cached message copy is released. If a timeout or packet loss occurs, the data remain in Q_t and await retransmission.

Based on this strict-triggering and acknowledgment-driven transmission-release strategy, the system can safely cache critical data during network fluctuations and release acknowledged data promptly after communication recovers, gradually clearing the backlog. At the system level, this mechanism can suppress unbounded growth of the pending keyframe queue Q_t and keep it bounded under finite retransmission and acknowledgment conditions, thereby substantially improving the operational stability of resource-constrained devices in long-duration and large-scale mapping tasks.

Cloud processing layer: This layer centrally executes computation-intensive backend tasks, including loop-closure detection, map-point fusion, pose graph optimization, global bundle adjustment (BA), point cloud reconstruction, and multi-agent map fusion. Cloud nodes receive keyframe data from edge devices and insert them into a global keyframe database and map management module, providing the data foundation for subsequent global optimization and map fusion.

For each newly inserted keyframe, the cloud adopts a keyframe-level loop-closure detection method based on ORB features. The system first retrieves potential loop candidates through a bag-of-words model and then uses feature matching and geometric-consistency verification to select valid loop-closure constraints. After a loop-closure constraint is established, the cloud performs map-point fusion, pose graph optimization, and global BA to correct keyframe poses and suppress accumulated drift. Let $\mathbf{X} = \{\mathbf{T}_i\}$ be the set of global poses to be estimated, and let $\mathbf{Z}_{ij}$ be the relative-pose constraint between nodes i and j obtained from odometry or loop-closure detection. The cloud solves the following weighted least-squares problem:

$$\mathbf{X}^\star = \arg\min_{\mathbf{X}} \sum_{(i,j)\in\mathcal{E}} \left\| \text{Log}\left(\mathbf{Z}_{ij}^{-1}\mathbf{T}_i^{-1}\mathbf{T}_j\right)\right\|_{\mathbf{\Omega}_{ij}}^2 \tag{4}$$

where $\text{Log}(\cdot)$ maps the pose error from the Lie group to the vector space, and $\mathbf{\Omega}_{ij}$ denotes the information matrix, i.e., the weight matrix, of constraint (i, j). The optimized pose $\mathbf{X}^\star$ provides a globally consistent alignment basis for subsequent multi-agent map fusion.

To reduce the computational overhead of map updating, this paper adopts a loop-closure-triggered global map reconstruction strategy. During local backend optimization, the cloud mainly maintains keyframes, map points, and their observation constraints. After loop-closure detection and global optimization are completed, the system reprojects observations based on the optimized keyframe poses and updates the global point cloud and occupancy grid map. Through this strategy, the results of backend pose optimization are consistently propagated to map reconstruction and fusion, avoiding the additional computational burden caused by frequent global reconstruction.

3.2 Map Fusion Framework

The core challenge of multi-agent collaborative mapping is how to unify environmental information from multiple local coordinate systems into a global coordinate system while handling observation noise and dynamic obstacles. The fusion design in this paper is driven by two requirements: maintaining numerical stability during long-term recursive updates and achieving global consistency across agents through cloud-provided alignment results. Therefore, the system adopts a probabilistic occupancy grid map fusion scheme.

3.2.1 Probabilistic Occupancy Grid Update

To avoid precision underflow in floating-point calculation and simplify the Bayesian update process, the system introduces log-likelihood ratios to represent grid states. Let m_i be the i-th grid cell in the map, and let $p(m_i)$ be its occupancy probability. The log-likelihood ratio $L(m_i)$ of this grid cell is defined as

$$L(m_i) = \log \frac{p(m_i)}{1 - p(m_i)} \tag{5}$$

Based on the binary Bayes filter, the recursive update formula for the grid state is

$$L(m_i \mid z_{1:t}) = L(m_i \mid z_{1:t-1}) + L_{\text{inv}}(m_i \mid z_t, x_t) - L_0(m_i) \tag{6}$$

Here, $L(m_i \mid z_{1:t})$ denotes the posterior log-likelihood ratio of grid cell m_i given the observation sequence $z_{1:t}$ at time t; $L_0(m_i)$ is the prior log-likelihood ratio, usually with $p(m_i) = 0.5$, i.e., $L_0 = 0$. $L_{\text{inv}}(m_i \mid z_t, x_t)$ is the inverse sensor model, which describes the contribution of a single observation z_t to the grid state under the current pose x_t. Specifically, according to the observation distance r from the sensor to the measurement endpoint, the inverse sensor model divides the measurement space into three regions: occupied, free, and unknown:

$$L_{\text{inv}}(m_i \mid z_t, x_t) = \begin{cases} l_{\text{occ}}, & \text{if } |r - d_i| < \delta, \\ l_{\text{free}}, & \text{if } d_i < r - \delta, \\ l_{\text{prior}}, & \text{otherwise} \end{cases} \tag{7}$$

In this formula, d_i is the Euclidean distance from the grid-cell center to the sensor, δ is the measurement uncertainty tolerance, and l_{occ} and l_{free} are the preset occupied and free log-likelihood-ratio increments, respectively.

3.2.2 *Global Coordinate Transformation and Map Merging*

To realize multi-source map fusion, the transformation from each agent's local coordinate system $\{L\}$ to the unified global coordinate system $\{G\}$ must first be solved. For the local map generated by each agent, the system maps it to a shared global reference frame through the coordinate transformation T_{GL}. For a 2D grid map, this projection relationship can be expressed as

$$\begin{bmatrix} x_G \\ y_G \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ 1 \end{bmatrix} \tag{8}$$

After spatial alignment, the system merges overlapping regions in the probability domain. Let $p_k(i)$ denote the occupancy probability of the k-th map at grid cell i, and let V_i denote the index set of agents that provide valid observations at grid cell i. The fused probability is computed as

$$p_{\text{fused}}(i) = \sum_{k \in V_i} \omega_{k,i} p_k(i) \tag{9}$$

where $\omega_{k,i}$ is the normalized fusion weight of the k-th agent at grid cell i and satisfies

$$\sum_{k \in V_i} \omega_{k,i} = 1, \quad \omega_{k,i} \geq 0 \tag{10}$$

In the experiments, all agents are equipped with homogeneous sensors and use the same processing pipeline. This paper adopts equal-weight fusion for valid observations, namely

$$\omega_{k,i} = \frac{1}{|V_i|} \tag{11}$$

where $|V_i|$ denotes the number of agents that provide valid observations at grid cell i. The corresponding implementation process is summarized in Algorithm 1.

Algorithm 1: Multi-Agent Map Merging Algorithm

Data: A set of occupancy grid maps $\{\text{grid}_k\}_{k=1}^{K}$, unknown-state probability p_{unk}
Result: grid, the final merged map

```
grid.header ← grid_1.header;
grid.info ← grid_1.info;
for i ← 0 to grid.data.size do
    V_i ← ∅;
    for k ← 1 to K do
        if grid_k.data[i] ≠ p_unk then
            V_i ← V_i ∪ {k};
        end
    end
    if V_i = ∅ then
        grid.data[i] ← p_unk;
    else
        for k ∈ V_i do
            ω_{k,i} ← 1/|V_i|;
        end
        grid.data[i] ← Σ_{k∈V_i} ω_{k,i} · grid_k.data[i];
    end
end
```

4 Experiments

To verify the effectiveness and adaptability of the proposed cloud-edge collaborative multi-agent visual SLAM architecture, this paper conducts experiments in terms of runtime efficiency and trajectory accuracy, public dataset comparison, Gazebo collaborative mapping, adaptability under communication constraints, loop-closure ablation, and long-horizon stability. First, the TUM public dataset is used to quantitatively evaluate runtime efficiency and trajectory accuracy under edge-side and cloud-side execution. Second, the EuRoC MAV stereo dataset is used for comparison with representative collaborative SLAM methods to further evaluate localization performance. Third,

Table 1. Runtime comparison on TUM sequences (edge versus cloud).

Dataset	Execution Mode	Total Runtime (s)	Median Tracking Time (s)	Mean Tracking Time (s)
freiburg1_desk	Cloud	**23.9218**	**0.0163**	**0.0170**
	Local	43.8943	0.0236	0.0254
freiburg1_desk2	Cloud	**25.7752**	**0.0138**	**0.0144**
	Local	42.3488	0.0226	0.0234
freiburg1_room	Cloud	**55.3761**	**0.0139**	**0.0154**
	Local	151.327	0.0197	0.0208
freiburg2_rpy	Cloud	**132.515**	**0.0235**	**0.0232**
	Local	281.557	0.0301	0.0295
freiburg2_xyz	Cloud	**145.695**	**0.0148**	**0.0154**
	Local	230.095	0.0217	0.0231
freiburg2_pioneer_slam3	Cloud	**135.001**	**0.0130**	**0.0162**
	Local	192.120	0.0260	0.0238

multi-agent collaborative mapping scenarios are constructed in Gazebo to evaluate the stability of cloud-side centralized processing and map fusion under complex environments and multi-agent conditions. Finally, long-horizon stress tests are conducted to verify the effectiveness of the proposed "upload–acknowledge–release" mechanism in alleviating edge-side memory constraints.

4.1 Experimental Platform and Parameter Settings

To verify the feasibility of the proposed cloud-edge decoupled architecture in controlled experimental scenarios, this paper builds a heterogeneous experimental platform composed of edge-side simulation nodes, a communication simulation module, and a cloud computing node. To ensure consistency with the homogeneous multi-agent visual SLAM setting of this paper, each edge-side agent uses the TurtleBot model in Gazebo as the mobile platform and is configured with the same camera model to collect color and depth images. The software environment is Ubuntu 18.04 LTS and ROS 1 Melodic; the edge-side nodes mainly perform frontend tracking, keyframe extraction, and data uploading. The cloud side is deployed on a workstation equipped with an Intel Core i7-12700K CPU, 32 GB RAM, and an NVIDIA RTX 3080 GPU, and is responsible for loop-closure detection, backend optimization, point cloud reconstruction, and map fusion.

The communication layer implements bidirectional asynchronous transmission using the Topic/TCPROS mechanism of ROS 1 Melodic. The uplink adopts keyframe-triggered transmission, serializing keyframes into binary packets and sending them to the cloud; the downlink returns lightweight ACK confirmation messages. Each upload packet contains the agent ID, sequence number, timestamp, compression flag, camera parameters, keyframe pose, RGB-D images, ORB features, and descriptors, and zlib compression is enabled by default. Under baseline conditions, the keyframe transmission frequency is approximately 1.9 Hz, the average raw keyframe packet size is approximately 2.62 MB, and the compressed size is approximately 0.50 MB. Without additional communication disturbance, the average uplink transmission delay on the test platform is approximately 0.25 ms, and the average ACK-link delay is approximately 0.06 ms.

4.2 Runtime Efficiency and Trajectory Accuracy on the TUM Dataset

To quantitatively evaluate the influence of cloud offloading on system runtime efficiency, six typical sequences from the TUM RGB-D dataset are selected: `freiburg1_desk`, `freiburg1_desk2`, `freiburg1_room`, `freiburg2_rpy`, `freiburg2_xyz`, and `freiburg2_pioneer_slam3`. These sequences cover small-scale desktop scenes, indoor room scenes, camera rotational and translational motion, and relatively large-scale indoor mobile scenes, enabling evaluation of the proposed system in terms of scene structure, motion pattern, and trajectory length. The experiments compare total runtime, median single-frame tracking time, and mean single-frame tracking time under local execution and cloud-side execution.

The local execution mode does not simply disconnect the cloud; instead, computation-intensive modules such as loop-closure detection, backend optimization, and map management are deployed locally so that the system completes the full SLAM pipeline on a single machine. In the cloud collaborative mode, only lightweight tasks such as frontend tracking and keyframe extraction are retained at the edge, while the above computation-intensive modules are executed by the cloud. This setting enables a comparative evaluation of the influence of cloud offloading on system runtime efficiency and edge-side computational load. The results are shown in Table 1.

Table 1 reports the runtime statistics of the six sequences under the two execution modes. The total runtime of the cloud mode is lower than that of the local mode on all sequences, with the average total runtime reduced by approximately 44.6%. At the same time, both the median and mean single-frame tracking times are lower, indicating that offloading backend computation and map-maintenance tasks to the cloud can effectively alleviate the computational burden on the edge side.

In addition to runtime efficiency, the root mean square error (RMSE) of the absolute trajectory error (APE) and relative pose error (RPE) is used as a quantitative metric to evaluate localization performance on the TUM dataset. The results are shown in Tables 2 and 3.

Table 2. APE RMSE comparison on TUM sequences.

TUM Sequence	Cloud RMSE (m)	Local RMSE (m)
freiburg1_desk	0.064	0.205
freiburg1_desk2	2.572	4.127
freiburg1_room	0.107	0.205
freiburg2_rpy	0.081	*Fail*
freiburg2_xyz	0.147	0.406
freiburg2_pioneer_slam3	1.396	*Fail*

Table 3. RPE RMSE comparison on TUM sequences.

TUM Sequence	Cloud RMSE (m)	Local RMSE (m)
freiburg1_desk	0.027368	0.028778
freiburg1_desk2	0.082179	0.226273
freiburg1_room	0.031086	0.043682
freiburg2_rpy	0.054521	*Fail*
freiburg2_xyz	0.080208	0.090762
freiburg2_pioneer_slam3	0.154111	*Fail*

As shown in Table 2, for the sequences in which both modes successfully complete trajectory estimation, the cloud mode achieves lower absolute trajectory error. For the `freiburg2_rpy` and `freiburg2_pioneer_slam3` sequences, the local mode fails to maintain tracking, while the cloud mode still generates valid trajectories. This indicates that, under the current experimental settings, the cloud mode provides better absolute localization accuracy and operational stability.

As shown in Table 3, for all sequences that yield valid trajectory estimates, the cloud mode achieves lower relative pose error overall. This suggests that cloud-side centralized backend optimization can suppress local error propagation to some extent and thereby improve the local stability of trajectory estimation.

Overall, Tables 1–3 show that the proposed method achieves a good balance between runtime efficiency and trajectory accuracy under the current experimental settings. The results indicate that offloading computation-intensive tasks such as backend optimization to the cloud helps reduce edge-side computational burden and improves localization accuracy and stability under long sequences and complex motion conditions.

4.3 Comparative Evaluation on the EuRoC Dataset

To further evaluate the relative performance of the proposed method on public datasets, comparative experiments are conducted on the Machine Hall sequences of the EuRoC dataset. Three representative sequences, namely MH_01_easy, MH_02_easy, and MH_03_medium, are selected. APE RMSE is used as the evaluation metric. CCM-SLAM [12] and Swarm-SLAM [11] are selected as comparison baselines, where CCM-SLAM represents a centralized collaborative SLAM framework and Swarm-SLAM represents a distributed collaborative SLAM framework. In this experiment, each method processes the selected sequences independently in single-agent mode to evaluate localization accuracy, as the EuRoC dataset does not provide synchronized multi-agent trajectories. The results are shown in Table 4.

Table 4. APE RMSE comparison on EuRoC Machine Hall sequences.

Method	MH01 (m)	MH02 (m)	MH03 (m)	Overall RMSE (m)
CCM-SLAM	0.061	0.081	0.048	0.065
Swarm-SLAM	0.145	0.079	0.170	0.136
Proposed	0.037	0.045	0.039	0.041

The results in Table 4 show that the proposed method achieves lower RMSE than all compared baselines in all evaluated test settings. Based on the overall APE RMSE, the proposed method reduces the error by approximately 36.9% relative to CCM-SLAM and 69.9% relative to Swarm-SLAM. These results indicate

favorable trajectory-estimation performance under the current test conditions and selected baseline settings.

4.4 Gazebo-Based Collaborative SLAM Experiments

To verify the operational performance of the proposed system in multi-agent collaborative mapping tasks, collaborative experiments are further conducted in the Gazebo simulation environment. According to the experimental objectives, this section is divided into two levels. First, a two-robot collaborative experiment is performed in a basic indoor scene to qualitatively validate the complete collaborative mapping pipeline, with emphasis on whether keyframe upload, cloud-side global processing, unified coordinate alignment, and map fusion can operate stably. Second, in an extended indoor scene, the number of robots is increased and the environmental complexity is enhanced to quantitatively evaluate trajectory error and runtime stability under different collaboration scales. The TurtleBot simulation model is used, and a 15 m $\times$ 15 m structured indoor basic scene containing corridors, display boards, and multiple types of geometric obstacles is built in Gazebo, as shown in Figure 2.

Figure 2. 3D map of the simulation environment.

In the initial stage, two agents start simultaneously from different initial positions and perform collaborative exploration. During system operation, the edge side is responsible for image acquisition, visual odometry, and ORB feature extraction, while the cloud receives keyframes and feature information uploaded by each agent and performs global pose optimization and map fusion.

As shown in Figure 3, the real-time GUI displays the operating status of the edge-side visual frontends of two agents in the simulation environment. The upper windows respectively show the image frames acquired by the two agents while moving in Gazebo and the corresponding ORB feature extraction results, where green markers indicate the ORB feature points

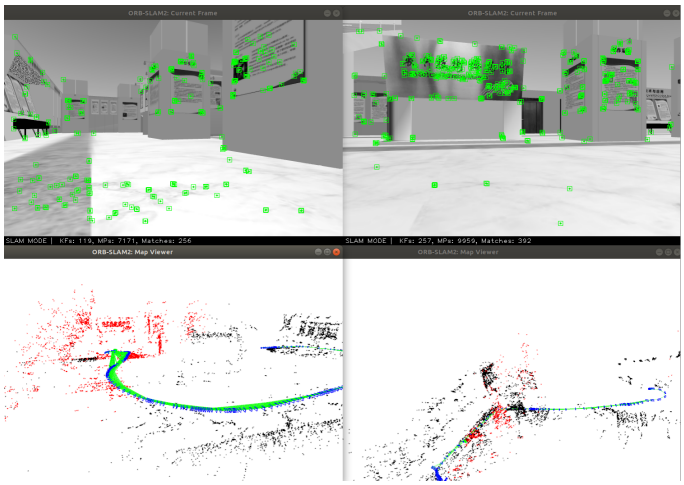

Figure 3. Edge-side operation diagram in the basic scenario.

detected in the current frame. The lower windows show the sparse point cloud map and motion trajectory maintained in real time by the local SLAM thread, where the blue curves denote the estimated trajectories of the agents, the green line segments denote covisibility connections between keyframes, the black points denote MapPoints generated and retained by historical frames, and the red points denote MapPoints currently observable by the agents.

This visualization indicates that, during parallel operation of the two agents, the edge-side visual frontend can complete image input, ORB feature extraction, local map maintenance, and trajectory updating at the shown time instant.

Figure 4. Top view of the basic scenario and cloud-side mapping result.

As shown in Figure 4, white denotes safe areas, green denotes unknown areas, and black denotes occupied or non-traversable areas. The cloud aggregates local map information from different agents and performs map fusion. The visualization shows that the map regions contributed by different agents can be gradually stitched into the same global coordinate system. The fused map maintains a continuous overall

structure, without obvious structural misalignment, duplicated boundaries, or local discontinuities. Non-traversable areas and major obstacle structures are consistently reflected in the fused occupancy grid map. This result indicates that the proposed cloud-edge collaborative centralized architecture can effectively support consistent representation and stable fusion of multi-source map information.

Based on the above experiments, the evaluation is further extended in terms of both environment scale and agent number to systematically examine the robustness and scalability of the proposed architecture. Two types of environments are considered: the original 15 m $\times$ 15 m basic indoor scene and a 25 m $\times$ 52 m extended indoor scene. For each environment, two-agent and three-agent collaborative mapping configurations are tested. Four groups of experiments are conducted: two agents in the basic indoor scene, three agents in the basic indoor scene, two agents in the extended indoor scene, and three agents in the extended indoor scene. Compared with the basic environment, the mapping area of the extended scene increases from 225 m^2 to 1300 m^2, approximately 5.78 times the original, and more structural and obstacle models are introduced to increase environmental complexity. The extended scene is shown in Figure 5, and the operation of the three-agent system is shown in Figure 6.

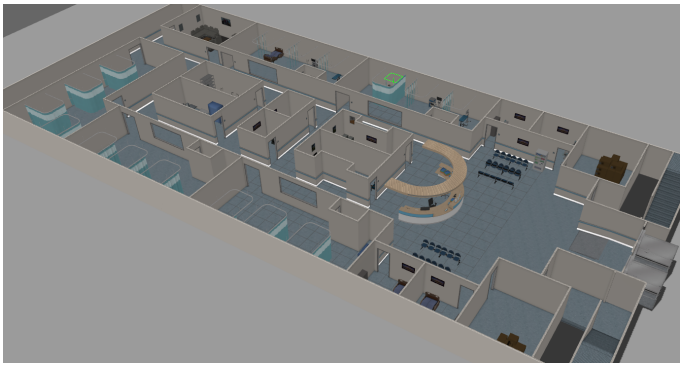

Figure 5. 3D map of the extended indoor simulation environment.

Compared with the two-agent experiment in the basic indoor scene, the extended scene and the three-agent setting impose higher requirements on the system. On the one hand, a larger scene scale implies longer trajectories and more map information to maintain. On the other hand, increasing the number of agents leads to more frequent multi-channel data uploads and greater aggregation pressure on the cloud side. Nevertheless, as shown in Figures 6 and 7, the system still maintains a relatively continuous frontend

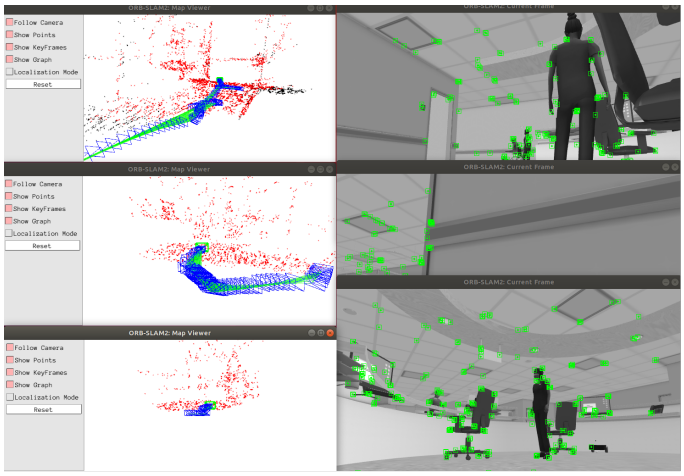

Figure 6. Edge-side operation diagram in the extended scenario.

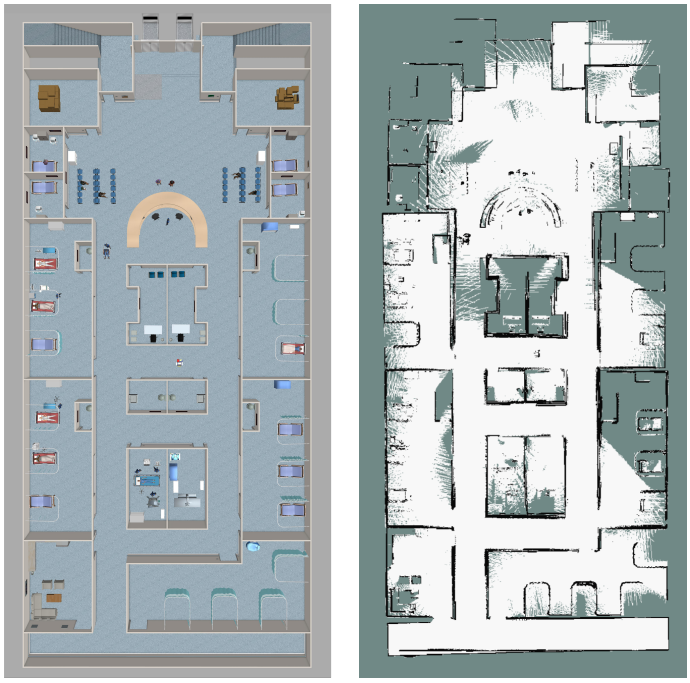

Figure 7. Cloud-side mapping result in the extended indoor scene.

tracking process and a relatively stable cloud-side map update process, indicating that, under the tested two-agent and three-agent configurations and extended indoor scene, the proposed architecture can maintain continuous operation and stable cloud-side map updates.

To quantitatively analyze system performance under different scene scales and numbers of agents, multi-agent collaborative mapping experiments are conducted in Gazebo, and the APE RMSE of each agent trajectory is calculated. To ensure consistency in method comparison, the experiments first record rosbag data corresponding to fixed motion

Table 5. APE RMSE results of Gazebo collaborative mapping under different scene scales and numbers of agents.

Method	Scene	Number of agents	Agent 1 (m)	Agent 2 (m)	Agent 3 (m)	Overall RMSE (m)
CCM-SLAM	Basic indoor scene	2	0.308	0.282	—	0.299
		3	0.057	0.119	0.107	0.092
	Extended indoor scene	2	*Fail*	0.318	—	0.318
		3	*Fail*	*Fail*	*Fail*	*Fail*
Swarm-SLAM	Basic indoor scene	2	0.116	0.165	—	0.138
		3	0.044	0.229	0.161	0.134
	Extended indoor scene	2	0.129	0.156	—	0.142
		3	0.120	0.127	0.174	0.140
Proposed	Basic indoor scene	2	0.019	0.032	—	0.025
		3	0.026	0.040	0.017	0.028
	Extended indoor scene	2	0.041	0.047	—	0.044
		3	0.049	0.042	0.059	0.049

Fail indicates that trajectory interruption occurred in all three repeated experiments. Overall RMSE is computed as the root mean square of the per-agent RMSE values over all agents that completed the task successfully.

trajectories in each scene, including RGB images, depth images, ground-truth poses, and robot motion states. Then, the proposed method, CCM-SLAM, and Swarm-SLAM are tested by replaying the same rosbag data, so that different methods run under the same trajectories, visual inputs, and scene conditions. CCM-SLAM is used as a representative centralized collaborative SLAM method, while Swarm-SLAM is used as a representative distributed collaborative SLAM method. During trajectory-error evaluation, to avoid time-association bias caused by communication latency, callback queuing, or node scheduling, the image acquisition timestamp is used to associate the estimated trajectory with the ground-truth trajectory rather than the node reception or processing timestamp. The results are shown in Table 5.

Table 5 presents the APE RMSE results of Gazebo collaborative mapping under different scene scales and numbers of agents. In the basic indoor scene, the multi-agent collaborative APE RMSE values of CCM-SLAM, Swarm-SLAM, and the proposed method under the two-agent configuration are 0.299 m, 0.138 m, and 0.025 m, respectively. Under the three-agent configuration, the multi-agent collaborative APE RMSE values of the three methods are 0.092 m, 0.134 m, and 0.028 m, respectively. Compared with CCM-SLAM, the errors of the proposed method under the two configurations are reduced by approximately 91.6% and 69.6%, respectively. Compared with Swarm-SLAM, the errors are reduced by approximately 81.9% and 79.1%, respectively. These results show that, under the same trajectory input and evaluation settings in the basic indoor scene, the proposed method achieves lower trajectory-estimation errors.

In the extended indoor scene, the scene scale and trajectory length further increase, and the differences in completion status and trajectory error among different methods become more evident. CCM-SLAM exhibits partial trajectory interruption under the two-agent configuration and cannot complete valid trajectory estimation under the three-agent configuration. Swarm-SLAM can complete both two-agent and three-agent collaborative operation, with multi-agent collaborative APE RMSE values of 0.142 m and 0.140 m, respectively. The proposed method also completes collaborative operation under both configurations, with multi-agent collaborative APE RMSE values of 0.044 m and 0.049 m, respectively. Compared with Swarm-SLAM, the errors of the proposed method under the two-agent and three-agent configurations are reduced by approximately 69.0% and 65.0%, respectively. In the extended two-agent scenario where the comprehensive error of CCM-SLAM can be calculated, the error of the proposed method is reduced by approximately 86.2%.

Across the four Gazebo test groups, the proposed method completes collaborative operation in all cases, and the multi-agent collaborative APE RMSE remains between 0.025 m and 0.049 m. Combining the comparison results in the basic and extended scenes, the proposed method achieves

lower trajectory-estimation error than the selected centralized and distributed collaborative SLAM baselines under the current Gazebo collaborative simulation scenarios, fixed motion trajectories, and identical rosbag inputs, and maintains continuous operation under extended scenes and multi-agent configurations.

4.5 Communication Tests

To evaluate the influence of communication constraints on continuous system operation and the completeness of cloud-side updates, communication-constrained experiments are conducted. All experiments replay the same Gazebo rosbag data to ensure consistent visual input, depth input, and ground-truth trajectory across different experimental groups. Communication disturbances are applied to the keyframe-upload link from the edge side to the cloud side. To simulate bandwidth limitation, transmission latency, random packet loss, and short-term outage, a ROS communication simulation node is inserted between the edge-side upload node and the cloud-side receiving node. By controlling the message forwarding rate, forwarding latency, and random dropping probability, controllable weak communication conditions are constructed. Based on this mechanism, different transmission delays, bandwidth limits, and random packet-loss rates are configured to analyze the effect of communication degradation on edge-side localization continuity, cloud-side keyframe access, and backend trajectory optimization results.

Specifically, fixed transmission delays are set to $d = \{100, 300, 600, 1000\}$ ms, and four random jitter conditions of 300 ± 100 ms, 300 ± 200 ms, 300 ± 400 ms, and 300 ± 600 ms are further set. The uplink bandwidth is set to $B = \{100, 50, 30, 20, 15, 10, 5, 2, 1, 0.5, 0.25\}$ Mbps; random packet-loss rates are set to $p = \{0\%, 2\%, 5\%, 10\%, 20\%\}$; and communication outage durations are set to $t = \{0, 5, 10, 15, 30, 60\}$ s. The experimental results are shown in Figures 8, 9, 10, and 11.

As shown in Figure 8, in the bandwidth-limitation experiment, when the uplink bandwidth decreases from 100 Mbps to 20 Mbps, the cloud-side APE RMSE remains approximately within 0.021–0.026 m, and the Cloud/Input KFs ratio remains at a relatively high level. This indicates that keyframe-level compressed transmission can still support cloud-side backend processing and map updates within this bandwidth range. When the bandwidth further decreases below

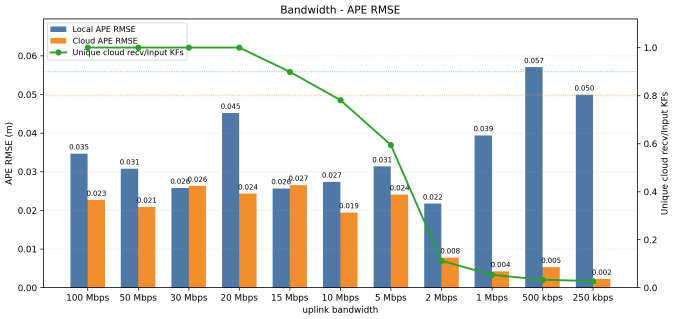

Figure 8. Bandwidth limitation experimental results.

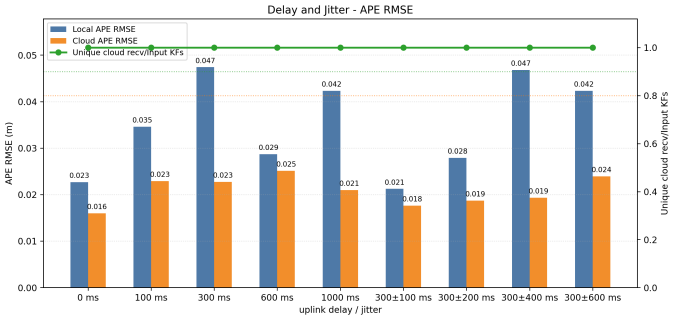

Figure 9. Transmission delay and jitter experimental results.

15 Mbps, the ratio of keyframes received by the cloud begins to drop significantly, indicating that insufficient bandwidth becomes the main factor limiting keyframe access and the completeness of cloud-side map updates. When the bandwidth is lower than 15 Mbps, the absence of a significant increase in cloud-side APE RMSE does not mean that the quality of the complete trajectory is unaffected by communication degradation. Because cloud-side error statistics are calculated only from trajectory segments that are successfully received and associated by timestamp, the cloud-side trajectory may have missing segments when a large number of subsequent keyframes cannot be inserted into the cloud in time. Therefore, this metric mainly reflects the estimation error of received trajectory segments and should not be directly compared with the complete local trajectory error.

By contrast, Figures 9, 10, and 11 show that the system exhibits certain adaptability to communication disturbances, including transmission latency, jitter, random packet loss, and communication outage. Under fixed delays of 0–1000 ms, random jitter, communication outage, and random packet loss of up to 20%, the cloud-side APE RMSE remains at a similar level, and no loss is observed in the final cloud-side keyframe access. This indicates that the asynchronous keyframe transmission mechanism can decouple edge-side real-time tracking from cloud-side backend processing, so that communication latency

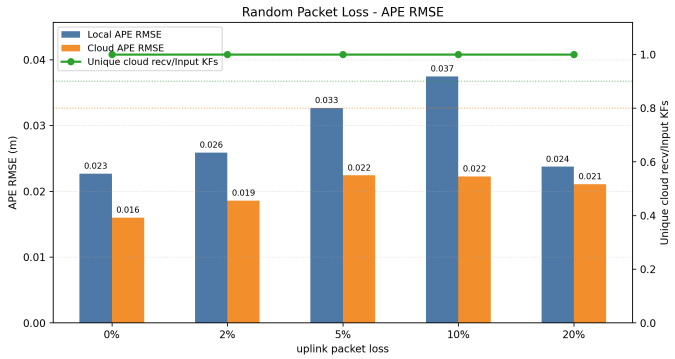

Figure 10. Random packet-loss experimental results.

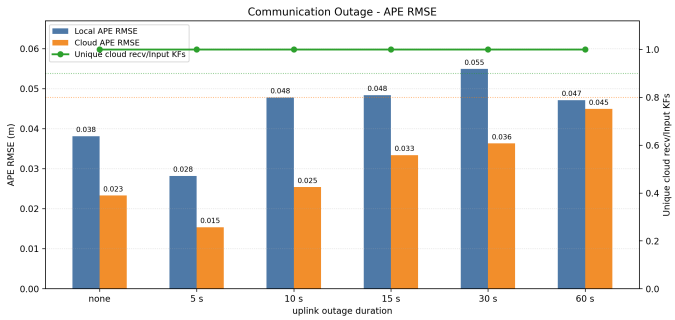

Figure 11. Communication outage experimental results.

and short-term packet loss mainly affect the timeliness of cloud-side updates rather than directly affecting or interrupting the system operation process.

The experimental results show that keyframe-level compressed uploading, asynchronous transmission, and the ACK confirmation–retransmission mechanism prevent edge-side tracking interruption within the tested ranges of latency, jitter, and short-term packet loss. When the uplink bandwidth falls below 15 Mbps, the cloud-side keyframe access ratio decreases, indicating that this mechanism is still constrained by severe bandwidth limitations.

4.6 Influence of Loop-Closure Detection on Map Fusion Consistency

To further analyze the influence of loop-closure detection on multi-agent map fusion consistency, an ablation experiment is conducted in the Gazebo collaborative mapping scenario to compare two cloud-side processing modes: disabling loop-closure detection and enabling loop-closure detection. When loop-closure detection is disabled, the system mainly relies on frontend odometry constraints and local keyframe connections for pose estimation and map fusion. When loop-closure detection is enabled, the cloud retrieves candidate loop closures using ORB features and a bag-of-words model, introduces valid loop-closure constraints after feature matching and geometric-consistency verification, performs pose graph optimization, and updates the global fused map based on the optimized keyframe poses.

(a) Fusion map without loop-closure detection.

(b) Fusion map with loop-closure detection.

Figure 12. Effect of loop-closure detection on fusion map consistency.

Figure 12 compares the fused maps obtained with loop-closure detection disabled and enabled. When loop-closure detection is disabled, accumulated pose errors cannot be globally corrected and are propagated to the map-stitching process, causing structural misalignment, boundary ghosting, and rough contours in some overlapping regions, which affects the continuity of wall and corridor structures. After loop-closure detection is enabled, the cloud can optimize keyframe poses using valid loop-closure constraints and use the optimization results for map reconstruction and fusion. As a result, the overall fused map structure becomes more continuous, local misalignment and ghosting are reduced, and wall boundaries and obstacle contours become clearer. This result indicates that loop-closure detection provides a more consistent global alignment basis for multi-source map fusion and thereby improves global map consistency.

4.7 Long-Horizon Stress Test

To evaluate the continuous operation capability of the edge side under a limited memory budget, long-horizon stress tests are further conducted in Gazebo to compare local single-device execution and cloud-edge collaborative execution during long-duration operation. The results are shown in Table 6.

The results show that, in the local single-device mode, the system crashes after processing approximately 1,067 keyframes, and the error log indicates that the cause is memory overflow. This shows that, as global map data and optimization graph structures

Table 6. Long-horizon stress-test results under a limited memory budget (local single-device versus cloud-edge collaboration).

Mode	Maximum Number of Keyframes	Status	Failure Cause
Local single-device	1,067	Crashed	Out of memory
Cloud-edge collaboration	2,511	Completed	–

continue to grow, a purely edge-side solution remains constrained by persistent memory pressure. By contrast, under the cloud-edge collaborative architecture, the system successfully completes the entire task, accumulates and optimizes 2,511 keyframes, and does not terminate abnormally. These results indicate that, by transferring computation and map-maintenance load to the cloud and applying the upload–acknowledge–release resource management mechanism on the edge side, the system can significantly improve long-term operational stability and avoid resource exhaustion encountered by purely edge-side schemes in large-scale tasks.

5 Conclusion and Future Work

This paper targets resource-constrained homogeneous multi-agent visual SLAM scenarios and designs and validates a cloud-edge collaborative centralized system framework. The framework retains real-time perception tasks such as visual odometry, keyframe extraction, and local tracking on the edge side, deploys computationally intensive tasks such as loop-closure detection, pose graph optimization, global map management, and multi-source map fusion in the cloud, and controls the growth of unacknowledged edge-side keyframe caches through keyframe-level compressed transmission, asynchronous uploading, and ACK confirmation.

Experimental results show that, under public datasets, Gazebo collaborative simulations, communication-constrained experiments, and long-horizon stress tests, the proposed framework can reduce edge-side computational and storage pressure while maintaining continuous frontend tracking and cloud-side map updates. Keyframe-level compressed transmission, asynchronous upload, and the ACK confirmation mechanism allow the edge side to maintain local pose estimation under transmission latency, random jitter, and short-term packet loss, whereas cloud-side centralized optimization and loop-closure constraints provide a more consistent global alignment basis for multi-source map fusion. These results indicate that cloud-edge functional decoupling can serve as a feasible system implementation for resource-constrained homogeneous multi-agent visual SLAM, rather than simply stacking single-machine SLAM modules.

Future work will further conduct system deployment and validation on real homogeneous multi-robot platforms and real wireless communication environments, with emphasis on analyzing the influence of practical motion-control errors, sensor noise, wireless-link blockage, multipath effects, larger-scale agent concurrency, and long-term map fusion error accumulation on system performance. Future work will also combine observation-uncertainty modeling, fusion-weight design, and cloud resource scheduling mechanisms to further analyze the applicable scope and performance boundaries of the proposed framework in more complex collaborative tasks.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that DeepSeek-R1 was used for translation of the first draft of the manuscript from Chinese to English. The authors have carefully reviewed, revised, and verified the AI-assisted output and take full responsibility for the content of the manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., ... & Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics, 32*(6), 1309-1332. [CrossRef]

[2] Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., & Tardós, J. D. (2021). Orb-slam3: An accurate open-source library for visual, visual–inertial, and

multimap slam. *IEEE transactions on robotics, 37*(6), 1874-1890. [CrossRef]

[3] Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE transactions on robotics, 31*(5), 1147-1163. [CrossRef]

[4] Mur-Artal, R., & Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics, 33*(5), 1255-1262. [CrossRef]

[5] Zou, D., Tan, P., & Yu, W. (2019). Collaborative visual SLAM for multiple agents: A brief survey. *Virtual Reality & Intelligent Hardware, 1*(5), 461-482. [CrossRef]

[6] Zhuang, L., Zhong, X., Xu, L., Tian, C., & Yu, W. (2024). Visual SLAM for unmanned aerial vehicles: Localization and perception. *Sensors, 24*(10), 2980. [CrossRef]

[7] Huang, P., Zeng, L., Chen, X., Luo, K., Zhou, Z., & Yu, S. (2022). Edge robotics: Edge-computing-accelerated multirobot simultaneous localization and mapping. *IEEE Internet of Things Journal, 9*(15), 14087-14102. [CrossRef]

[8] Zhang, T., Zhang, L., Chen, Y., & Zhou, Y. (2022). CVIDS: A collaborative localization and dense mapping framework for multi-agent based visual-inertial SLAM. *IEEE transactions on image processing, 31*, 6562-6576. [CrossRef]

[9] Macario Barros, A., Michel, M., Moline, Y., Corre, G., & Carrel, F. (2022). A Comprehensive Survey of Visual SLAM Algorithms. *Robotics, 11*(1), 24. [CrossRef]

[10] Lajoie, P. Y., Ramtoula, B., Chang, Y., Carlone, L., & Beltrame, G. (2020). DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams. *IEEE Robotics and Automation Letters, 5*(2), 1656-1663. [CrossRef]

[11] Lajoie, P. Y., & Beltrame, G. (2023). Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. *IEEE robotics and automation letters, 9*(1), 475-482. [CrossRef]

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

[13] Delmerico, J., & Scaramuzza, D. (2018, May). A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 2502-2509). IEEE. [CrossRef]

[14] Tang, J., Ericson, L., Folkesson, J., & Jensfelt, P. (2019). GCNv2: Efficient correspondence prediction for real-time SLAM. *IEEE Robotics and Automation Letters, 4*(4), 3505-3512. [CrossRef]

[15] Chen, W., Chen, S., Leng, J., Wang, J., Guan, Y., Meng, M. Q. H., & Zhang, H. (2024). A review of cloud-edge SLAM: Toward asynchronous collaboration and implicit representation transmission. *IEEE Transactions on Intelligent Transportation Systems, 25*(11), 15437-15453. [CrossRef]

[16] Chen, W., Yaguchi, Y., Naruse, K., Watanobe, Y., Nakamura, K., & Ogawa, J. (2018). A study of robotic cooperation in cloud robotics: Architecture and challenges. *IEEE Access, 6*, 36662-36682. [CrossRef]

[17] Chen, W., Lin, Z., Zhu, L., Chen, S., Zhu, H., Guan, Y., & Zhang, H. (2024). Cloud-edge collaborative submap-based VSLAM using implicit representation transmission. *IEEE Transactions on Vehicular Technology, 73*(10), 14537-14546. [CrossRef]

[18] Li, D., Zhao, Y., Xu, J., Zhang, S., Shangguan, L., & Yang, Z. (2024, May). EdgeSLAM2: Rethinking edge-assisted visual SLAM with on-chip intelligence. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications* (pp. 1481-1490). IEEE. [CrossRef]

[19] Sossalla, P., Rischke, J., Hofer, J., & Fitzek, F. H. (2021, September). Evaluating the advantages of remote SLAM on an edge cloud. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (pp. 01-04). IEEE. [CrossRef]

[20] Yanık, Ö. F., & Ilgın, H. A. (2023). A comprehensive computational cost analysis for state-of-the-art visual slam methods for autonomous mapping. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, 65*(1), 1-15. [CrossRef]

[21] Wu, Y., Li, Z., Palaiahnakote, S., & Lu, T. (2018, August). Em-SLAM: a fast and robust monocular SLAM method for embedded systems. In *2018 24th International Conference on Pattern Recognition (ICPR)* (pp. 1882-1887). IEEE. [CrossRef]

[22] Picard, Q., Chevobbe, S., Darouich, M., & Didier, J. Y. (2022, October). Image quantization towards data reduction: robustness analysis for SLAM methods on embedded platforms. In *2022 IEEE International Conference on Image Processing (ICIP)* (pp. 4158-4162). IEEE. [CrossRef]

[23] Khalufa, A., Riley, G., & Lujan, M. (2019, September). Poster: Runtime adaptations for energy-efficient vslam. In *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)* (pp. 475-476). IEEE. [CrossRef]

[24] Eger, S., Pries, R., & Steinbach, E. (2020, September). Evaluation of different task distributions for edge cloud-based collaborative visual SLAM. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)* (pp. 1-6). IEEE. [CrossRef]

[25] Sossalla, P., Hofer, J., Rischke, J., Busch, J., Nguyen, G. T., Reisslein, M., & Fitzek, F. H. (2022, December). Optimizing Edge SLAM: Judicious parameter settings and parallelized map updates. In *GLOBECOM*

2022-2022 IEEE Global Communications Conference (pp. 1954-1959). IEEE. [CrossRef]

[26] Xu, J., Cao, H., Li, D., Huang, K., Qian, C., Shangguan, L., & Yang, Z. (2020, July). Edge assisted mobile semantic visual SLAM. In *IEEE INFOCOM 2020-IEEE Conference on computer communications* (pp. 1828-1837). IEEE. [CrossRef]

[27] Gao, S., Zhu, J., Wang, Z., Ren, S., Zhang, K., & Chen, P. (2023, December). FAES-SLAM: Fast and Accurate Edge-Assisted Semantic SLAM Towards Dynamic Environments. In *GLOBECOM 2023-2023 IEEE Global Communications Conference* (pp. 3591-3596). IEEE. [CrossRef]

[28] Chen, Y., Inaltekin, H., & Gorlatova, M. (2023, May). AdaptSLAM: Edge-assisted adaptive SLAM with resource constraints via uncertainty minimization. In *IEEE INFOCOM 2023-IEEE Conference on computer communications* (pp. 1-10). IEEE. [CrossRef]

[29] Zhang, Y., Mao, Y., Wang, H., Yu, Z., Guo, S., Zhang, J., ... & Guo, B. (2025). Orchestrating joint offloading and scheduling for low-latency edge slam. *IEEE transactions on mobile computing*, *24*(8), 6901-6917. [CrossRef]

Lian Yu is studying for the Master's degree at the National University of Defense Technology, Changsha, China. Her research interests include multi-agent visual SLAM and 3DGS SLAM. (Email: yulian24@nudt.edu.cn)

Dongsheng Li is a Ph.D. candidate at the National University of Defense Technology, Changsha, China, and a joint-training student at the Belarusian State University of Informatics and Radioelectronics, Minsk, Belarus. His research interests include multi-camera visual SLAM, multi-object tracking, and information fusion. (Email: 1657268543@qq.com)

Guoyan Wang is a Lecturer with the College of Electronic Science and Technology, National University of Defense Technology, Changsha, China. She received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China, in 2012. Supported by the China Scholarship Council, she went to Bauman Moscow State Technical University, Moscow, Russia, to pursue the M.S. and Ph.D. degrees. From 2018 to 2020, she served as a Teaching Assistant there. Her current research interests include laser-based and multi-camera visual SLAM, as well as multi-source information fusion. (Email: wangguoyan@nudt.edu.cn)

Fei Zhao is an Associate Research Professor at the National University of Defense Technology, Changsha, China. His research mainly focuses on optical image target recognition and real-time information processing system design. (Email: f_z2020@126.com)