electronics

MDPI

Article

CORB2I-SLAM: An Adaptive Collaborative Visual-Inertial SLAM for Multiple Robots

Arindam Saha [1], Bibhas Chandra Dhara [1], Saiyed Umer [2,*], Ahmad Ali AlZubi [3], Jazem Mutared Alanazi [4] and Kulakov Yurii [5]

[1] Department of Information Technology, Jadavpur University, Kolkata 700098, India
[2] Department of Computer Science and Engineering, Aliah University, Kolkata 700156, India
[3] Computer Science Department, King Saud University, Riyadh 11437, Saudi Arabia
[4] Computer Science Department, Community College, King Saud University, Riyadh 11437, Saudi Arabia
[5] Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv 03056, Ukraine
* Correspondence: saiyed.umer@aliah.ac.in

Abstract: The generation of robust global maps of an unknown cluttered environment through a collaborative robotic framework is challenging. We present a collaborative SLAM framework, CORB2I-SLAM, in which each participating robot carries a camera (monocular/stereo/RGB-D) and an inertial sensor to run odometry. A centralized server stores all the maps and executes processor-intensive tasks, e.g., loop closing, map merging, and global optimization. The proposed framework uses well-established Visual-Inertial Odometry (VIO), and can be adapted to use Visual Odometry (VO) when the measurements from inertial sensors are noisy. The proposed system solves certain disadvantages of odometry-based systems such as erroneous pose estimation due to incorrect feature selection or losing track due to abrupt camera motion and provides a more accurate result. We perform feasibility tests on real robot autonomy and extensively validate the accuracy of CORB2I-SLAM on benchmark data sequences. We also evaluate its scalability and applicability in terms of the number of participating robots and network requirements, respectively.

Keywords: Visual-Inertial Odometry; visual-inertial SLAM; collaborative SLAM; multi-map SLAM; client–server architecture; heterogeneous camera configuration

Citation: Saha, A.; Dhara, B.C.; Umer, S.; AlZubi, A.A.; Alanazi, J.M.; Yurii, K. CORB2I-SLAM: An Adaptive Collaborative Visual-Inertial SLAM for Multiple Robots. *Electronics* **2022**, *11*, 2814. https://doi.org/10.3390/electronics11182814

Academic Editor: Dah-Jye Lee

Received: 13 July 2022
Accepted: 16 August 2022
Published: 6 September 2022

1. Introduction

The autonomous navigation of robots requires robust estimation of robot poses (position and orientation) as well as 3D scene structure. Visual Simultaneous Localization and Mapping (VSLAM) [1–5] is the most accepted framework for estimating accurate poses and scene structure. Hence, VSLAM has become a popular application for unmanned aerial vehicles (UAVs) as well as unmanned ground vehicles (UGVs). VSLAM has some inherent problems such as cumulative drift, losing camera track, unknown scale [1], etc., due to the limitations in VO [1,6,7] estimation, and Visual Inertial SLAM (VISLAM) alleviates some of such problems because VIO [8–11] uses the measurements from the Inertial Measurement Unit (IMU) to estimate the initial motion. Moreover, visual and IMU sensors are complementary in nature; therefore, IMU can provide estimations when VO fails. In general, VIO is expected to produce better pose estimation than VO, but this depends on the quality of the IMU sensor in terms of accurate measurement data [2]. In practice, noisy IMU measurements can reduce the quality of combined measurements. Therefore, the system requires intelligence to validate the IMU measurements before fusing them with any other sensors.

The use of multiple heterogeneous mobile robots on a big mission is advantageous because a complicated task can be broken down and allocated to multiple robots based on their capabilities to reduce the final completion time. A collaborative SLAM framework

helps multiple robots to navigate in an unknown environment in cooperation, generates a global environmental map, and shares information among the robots. Information sharing leads to the rapid exploration of the region and results in a more reliable system because the system can still function even if one or more robot becomes inoperable. The major challenges of any collaborative framework are communication among participating robots, information availability among all robots, fusing partial maps generated by individual robots, avoiding losing track, the reuse of map information, handling network delays, etc.

In this spirit, we propose a collaborative VISLAM framework, CORB2I-SLAM, where each participating robot is treated as a client and accompanied by at least one visual sensor (e.g., monocular/stereo/RGB-D camera) and may carry an IMU. These robots have limited processing capabilities and navigate using either VIO or VO on a local map depending on the availability of the IMU. The framework contains a centralized server with higher processing and larger storage capability. The server receives information from all participating robots and executes all computationally complex tasks for VSLAM, e.g., loop closing, map management and merging, and optimization. The central server shares optimized information with the respective robots whenever required. The rest of the paper uses robots, clients, or agents synonymously to indicate the participating robots. Our proposed system uses Robot Operating System [12] (ROS)-based message passing with approximate time synchronization. The main contribution of this work can be summarized as follows:

- A collaborative visual-inertial SLAM framework that supports agents with multiple types of cameras, such as monocular, stereo, and RGB-D. It can prevent erroneous IMU pose integration.
- An efficient criterion to estimate the reliability of camera pose and decide the loss of tracking once the reliability is below a threshold.
- An efficient collaborative SLAM framework design with multi-map operation either within a single client or among multiple clients. A novel algorithm for efficient map fusion on the server.

The rest of the paper is organized as follows: Section 2 describes the literature survey of collaborative VSLAMs and VISLAMs. Section 3 describes our proposed collaborative framework and our contribution. Section 4 presents the experimental results. Finally, Section 5 concludes the paper.

2. Related Work

We describe the related work on collaborative SLAMs that use either visual or visual-inertial sensors. VSLAMs are broadly classified into two categories, feature-based and direct methods, and we refer to [13,14] for a detailed description of all types of VSLAMs. Researchers mainly choose feature-based VSLAMs for collaborative frameworks, as explained in [15]. Therefore, we restrict our discussion to feature-based VSLAMs.

Multiple collaborative frameworks are present in [16–18], which use the global fused map to guide the trajectory estimation of UAVs but never share the global map with the UAVs. Therefore, these systems are limited to 3D scene reconstruction. Choudhary et al. [19] propose an object-based distributed SLAM system where all agents exchange information directly among each other and perform all information fusion on-board, without having a central instance. The relative localization is based on commonly observed pre-trained objects. Some recent distributed SLAM systems are proposed in [20–23], which focus on different aspects (e.g., decentralized place recognition, map overlap identification, efficient distributed loop closure, data exchange, robustness, etc.) of decentralized collaborative SLAM. However, the challenges with these distributed systems are that participating robots must be equipped with high computation processing, assurance of data consistency, and the avoidance of double-counting of information.

Zou and Tan [24] propose CoSLAM, a centralized collaborative monocular SLAM capable of handling dynamic environments. The major drawback of this system is that all cameras are synchronized by observing the same scene at initialization. Forster et al.

propose a collaborative SLAM framework [25] based on the structure from motion pipeline in a client–server model, but the server never shares optimized information with the agents. Riazuelo et al. propose C^2TAM [26], based on a client–server model where each client uses PTAM [27] and the server periodically sends back the optimized complete map to every agent. The system is communication-heavy for a large map in widely distributed areas. Deutsch et al. present a collaborative framework [28] that allows agents to use different monocular SLAMs. Each agent is informed only of updates to its local pose graph by the server and is unaware of sub-maps from other agents. Schmuck and Chli propose CCM-SLAM [15], a client–server-based collaborative SLAM framework where each client runs monocular VO on a local map with a fixed number of key frames (KFs) and does not have any relocalization mechanism after losing the camera track, which is obvious in a VSLAM scenario. Recently, Richard et al. have presented ORB-Atlas [29], a multi-map system for a single agent, in which it creates a new sub-map after tracking fails and merges multiple sub-maps afterwards. The system is not designed for a collaborative framework. Recently, Ouyang et al. have presented a collaborative framework [30], which uses a similar design to CCM-SLAM for only UGVs to carry monocular or RGB-D cameras. The proposed system claims to fuse maps between a monocular camera (without metric scale) and an RGB-D camera (with scale), but the literature does not provide the mechanism of such fusion. The proposed system considers each camera as an independent client in a scenario where a single agent carries multiple cameras, which makes the system more computationally intensive.

Stefan et al. present an open KF-based visual inertial SLAM (OKVIS) [8] for a single agent that utilizes non-linear optimization with visual reprojection errors and IMU motion errors on a sliding window. Tong et al. present a robust corner-feature-based visual inertial SLAM (VINS-Mono) [31] for a single agent that applies a loosely coupled sensor fusion initialization but uses pre-integrated IMU measurements before using them for pose optimization on a sliding window. Recently, Marco et al. proposed a VIO version of CCM-SLAM, CVI-SLAM [32], for only monocular cameras and used pre-integrated IMU measurements for the optimization of KF poses. Recently, Campos et al. presented ORB-SLAM3 [33], a visual-inertial version of ORB-SLAM2 [3] with multi-map support for a single agent. Jialing et al. present a collaborative visual-inertial SLAM [34] using monocular cameras for an augmented-reality application in which multiple users interact with their smartphones. The system is designed as a client–server architecture that models the maps as deformable maps and all the sub-maps in the fused maps are optimized independently to solve the problem of common map distortion. Patrik et al. recently present a visual inertial SLAM for centralized collaboration, COVINS [35], which is also designed as a client–server architecture and is capable of incorporating twelve agents jointly. All these proposed visual inertial SLAMs show state-of-the-art (SoA) accuracy on open data sets that are equipped with good IMU sensors. However, no system is adaptable enough to deal with noisy sensor measurements.

3. Proposed Methodology

3.1. Framework Description

Figure 1 presents the architecture of our proposed framework. Every client robot is equipped with at least one monocular/stereo/RGB-D camera, an IMU (optional), a processing unit, a small memory unit, and a wireless module. The central server contains high-performance computation capabilities along with large storage and wireless communication capabilities. The system configuration does not assume any specific configuration of client robots, so any type of UGV or UAV can participate as a client. If IMU is available, any agent must try to initialize with VIO and continue with VIO if the biases of the accelerometer and gyroscope, gravity direction, and velocities are estimated correctly; otherwise, VO only estimation follows. In this context, we assume that the environment is feature-rich and has sufficient illumination.

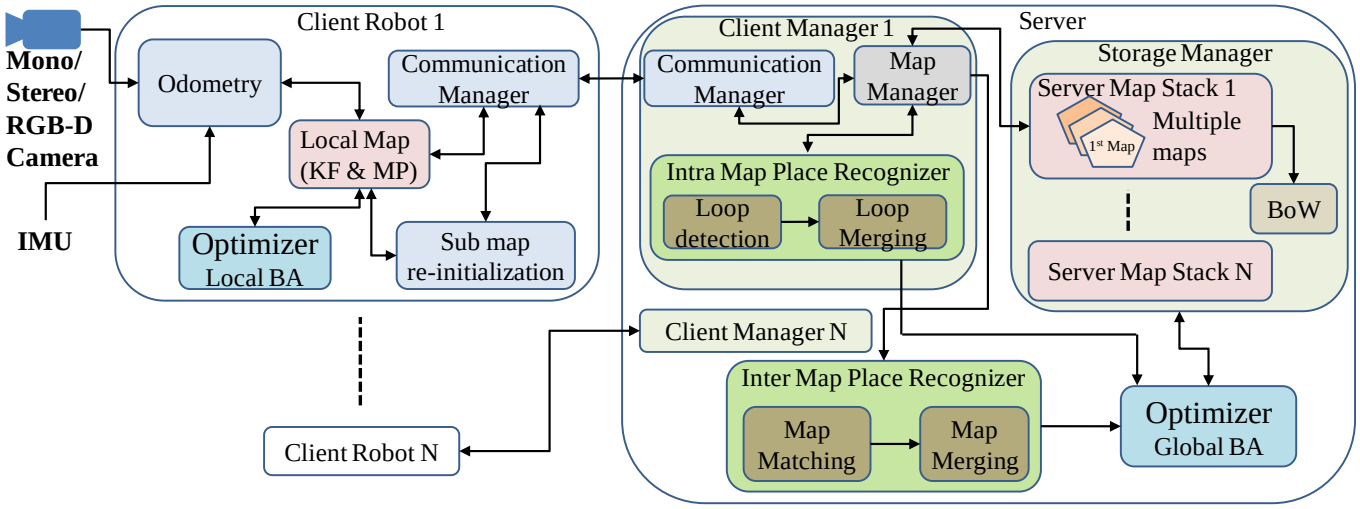

Figure 1. CORB2I-SLAM framework architecture.

Each client runs ORB VO/VIO on the local map, which follows a similar design to ORB-SLAM2. In the present system, the local map structure is a sparse map, and we have not considered a dense map formulation on the client in order to avoid high computations. Therefore, the map structure, the KFs, and map points (MPs) follow similar conventions to ORB-SLAM2 except for their globally unique numbers. At the beginning, every client receives a unique client identification number (client ID, e.g., client_1, client_2, etc.) from the server and initializes a local map. The local map with a fixed number (N) of KFs represents the immediate vicinity of the client; therefore, clients offload the map structure, including all KFs and MPs, to the server and delete all old KFs and MPs that are not part of the local map. To reduce communication overhead, the communication messages are designed for only visual data types. Therefore, the server map contains only visual data. The server allocates a separate client manager for every client, which handles the data management for associated clients. Data management includes receiving new KFs and MPs from the corresponding client and storing them in the appropriate map stack. The server sends back past KFs and maps to a client only when the server finds that the client reaches a place near to a past location and the client's local map does not contain the past location. If the client finds any matches with these past KFs, the server executes a loop closer. We use similar map structures for transmission between a client and the server as in CCM-SLAM. We incorporate a novel reinitialization procedure with a globally unique map into the framework when any client experiences track-loss. The server runs all computationally expensive algorithms, namely intra- and inter-map place recognition, map fusion, and global bundle adjustment (BA) [36]. The server can include more modules based on its computing power, e.g., dense map creation [37] and scene graph creation [38,39], to realize semantic information, but these are not considered in the present work. The present design makes the collaborative functionalities of the framework heavily dependent on the server's performance, and the performance decreases with an increase in the number of active clients.

3.2. Notation

We use C to denote the camera coordinate, W to denote the world coordinate, and B to denote the body coordinate of the IMU. We use small bold letters to denote vectors and bold capital letters for matrices. The position and rotation of the world frame W relative to the i-th camera frame can be described by the rigid body transformation $\boldsymbol{T}_{C_i}^{w} \in SE(3)$, where $\boldsymbol{R}_{C_i}^{w}$ is the corresponding rotation matrix and $\boldsymbol{p}_{C_i}^{w}$ is the corresponding translation vector. A landmark is represented as a 3D point $\varkappa \in \mathbb{R}^3$ with its image projection as $\boldsymbol{u} = \pi(\varkappa)$ where $\boldsymbol{u} \in \mathbb{R}^2$.

3.3. IMU Measurements

The IMU frequency is much higher than that of the visual camera; therefore, we require performing a pre-integration of the IMU measurements between two consecutive KFs in order to calculate a relative relationship between the consecutive KFs. We use a manifold-based pre-integration of IMU measurements to transform IMU measurement data into visual KF constraints, as proposed in [40]. The residual errors from KF k to KF $k+1$ with this IMU pre-integration model are given in Equation (1).

$$\begin{aligned}
\Delta \boldsymbol{p}_{B_{k+1}}^{B_k} &= \boldsymbol{R}_w^{B_k}(\boldsymbol{p}_{B_k}^w + \boldsymbol{v}_{B_k}^w \Delta t_k - \frac{1}{2}\boldsymbol{g}^w \Delta t_k^2) \\
&\quad - [\check{\boldsymbol{p}}_{B_k}^w(\hat{\boldsymbol{b}}^g, \hat{\boldsymbol{b}}^a) + \frac{\partial \hat{\boldsymbol{p}}_{B_k}^w}{\partial \boldsymbol{b}^g}\delta \boldsymbol{b}^g + \frac{\partial \hat{\boldsymbol{p}}_{B_k}^w}{\partial \boldsymbol{b}^a}\delta \boldsymbol{b}^a] \\
\Delta \boldsymbol{v}_{B_{k+1}}^{B_k} &= \boldsymbol{R}_w^{B_k}(\boldsymbol{v}_{B_k}^w - \boldsymbol{g}^w \Delta t_k) \\
&\quad - [\check{\boldsymbol{v}}_{B_k}^w(\hat{\boldsymbol{b}}^g, \hat{\boldsymbol{b}}^a) + \frac{\partial \hat{\boldsymbol{v}}_{B_k}^w}{\partial \boldsymbol{b}^g}\delta \boldsymbol{b}^g + \frac{\partial \hat{\boldsymbol{v}}_{B_k}^w}{\partial \boldsymbol{b}^a}\delta \boldsymbol{b}^a] \\
\Delta \boldsymbol{R}_{B_{k+1}}^{B_k} &= \log((\check{\boldsymbol{R}}_{B_k}^w(\hat{\boldsymbol{b}}_{B_k}^g)\exp(\frac{\partial \hat{\boldsymbol{R}}_{B_k}^w}{\partial \boldsymbol{b}^g}\delta \boldsymbol{b}^g))^T \boldsymbol{R}_w^{B_k} \boldsymbol{R}_{B_{k+1}}^w)
\end{aligned} \tag{1}$$

where $\Delta \boldsymbol{p}_{B_k}^w$, $\Delta \boldsymbol{v}_{B_k}^w$ and $\Delta \boldsymbol{R}_{B_k}^w$ denote position, velocity, and rotation, respectively, in IMU body coordinate at the time of KF k. $\boldsymbol{g}^w$, $\boldsymbol{b}^a$, and $\boldsymbol{b}^g$ denote the gravity vector in the world frame, the biases of the accelerometer, and the biases of the gyroscope, respectively. Symbols '$\check{\boldsymbol{x}}$' and '$\hat{\boldsymbol{x}}$' denote the bias estimation at the time of pre-integration of variable '$\boldsymbol{x}$' and the current estimates of the variable '$\boldsymbol{x}$', respectively. We refer to [40] for a detailed explanation of the pre-integration method.

3.4. Tracking (Visual-Inertial and Visual)

We follow the strategy as proposed in CVI-SLAM for the initialization of VIO. In the present system, we first initialize the VO tracking, where we extract ORB features from two frames, a reference frame, and the current frame, and estimate their poses either by homography or by fundamental matrix, a similar approach to ORB-SLAM2. After a successful initialization, we create a map structure with KFs and MPs, where each of them is identified with a unique identification number formed with a numerical value and the client id, for example, a KF id (KF_x, client_y), where x and y are numerical values that indicate KF's number and client number, respectively, and a MP id (MP_z, client_w), where z and w are numerical values that indicate MP's number and client number, respectively. We keep the distance between two consecutive KFs at 5 to avoid large IMU pre-integration. The visual structure is optimized using BA with 20 KFs and the gyroscope bias is initialized in a linear least squares fashion. The unknown scaling parameter for the metric scale, velocity, and gravity directions are estimated linearly and refined. Once we achieve reliable estimations in all the steps, we assume that the IMU estimation is correct, and we scale the visual structure, followed by an alignment with the gravity direction. If the initialization is unsuccessful, we iterate the process and declare the IMU unstable temporarily if the initialization fails three times consecutively. Subsequently, we continue with pure VO estimation as described in ORB-SLAM2. The client system continuously checks for a nonlinear motion of the client using the estimated poses of the KFs from VO. Once the nonlinear motion is observed, IMU measurements between the selected KFs are collected, and IMU estimations are reiterated as described in Section 3.3. We continue with VIO initialization as described above if we achieve stable IMU estimations; otherwise, we declare the IMU as permanently unstable.

We extract ORB features on every incoming frame and integrate all the IMU measurements accumulated since the last frame to estimate the motion model. This motion model helps in predicting the pose of the incoming frame, and a guided search is performed to find 2D correspondences by projecting the 3D map points into the current frame. We

perform a two-step frame alignment, where the initial alignment is based on the matched correspondences and further matching correspondences are searched on the frame by projecting other map points and optimizing the final alignment using motion-only BA on a local window. Here, we check whether to consider the current frame as a new KF and store it in the local map. We select the current frame as a new KF if one of the following conditions is satisfied.

(a) The current frame is 20 frames apart from the last KF.
(b) The current frame observes less than 15 old MPs.
(c) 2D key points cover less than 40% of the image area.

In the case of VO, we follow the process as in underline ORB-SLAM2, and we select the current frame as a new KF when one of the following conditions is satisfied.

(a) The current frame is 20 frames apart from the last KF.
(b) The current frame observes less than 70 close MPs.
(c) The current frame observes less than 80% MPs than the last KF.
(d) 2D key points cover less than 40% of the image area.

3.5. VO Pose Consistency

The estimated poses using VIO are computed using measurements from IMU, as explained in Sections 3.3 and 3.4, where the motion estimation of the agent is guided through the IMU measurements. Whereas the estimated poses using VO are purely based on visual features that convey a geometric interpretation. Therefore, poses become erroneous when visual features are not geometrically rich. Therefore, we evaluate the poses of every frame with a heuristically proposed verification step to declare that the camera tracking is correct in the case of VO estimation.

Scene Geometry Consistency: Once the detected points are not geometrically rich enough to carry a geometric interpretation of the structure, then the estimated pose must be erroneous. We measure the structural continuity of the reconstructed visual structure to measure the geometric consistency as first proposed in [4]. We extract edges from the RGB image and reconstruct the longest edge in 3D using sampled points on the selected edge and the estimated pose of the frame. Afterwards, we check the depth continuity of the reconstructed 3D points using their positional coordinates. In this case, we try to extract the line by fitting a line equation, as explained in [41–43], and consider the 3D line segment as continuous if all the 3D points satisfy the line equation. The basic assumption is that a line segment would be continuous in 3D if it is continuous in 2D. Once the depth continuity is broken, we consider that the pose is not accurate enough for tracking.

Pose Observability: Camera pose observability is first proposed in the ORB-Atlas [29]. We use a modified form of pose observability. Camera pose estimation becomes poor when the tracked features have a very high depth. The 2D motions of such features on consecutive images are negligible and thus fail to encode the true motion of the camera. We therefore use the uncertainty of watching a 3D point $\varkappa_i$ into a camera C_k as $\Psi_{\varkappa_i,C_k}$. This uncertainty is proportional to the observational depth of $\varkappa_i$, which encodes a higher depth point with a higher uncertainty of observability.

The estimated six-degrees-of-freedom (DoF) camera pose of the kth frame is $\hat{\boldsymbol{T}}^{w}_{C_k}$. We represent the uncertainty of this estimated pose with an unbiased Gaussian vector of six parameters, $\boldsymbol{\rho}_{C_k}$, which defines the Lie algebra approximation of $\boldsymbol{T}^{w}_{C_k}$ around $\hat{\boldsymbol{T}}^{w}_{C_k}$, as given in Equation (2).

$$\begin{aligned} \boldsymbol{T}^{w}_{C_k} &= \exp\left(\boldsymbol{\rho}_{C_k}\right) \oplus \hat{\boldsymbol{T}}^{w}_{C_k} \\ \boldsymbol{\rho}_{C_k} &= (x, y, z, \omega_x, \omega_y, \omega_z) \sim \mathcal{N}(0, \boldsymbol{Cov}_{C_k}) \\ \boldsymbol{Cov}_{C_k} &\approx \Big(\sum_{\varkappa_i}^{\varkappa_n} \boldsymbol{J}^{T}_{\varkappa_i,C_k} \Psi_{\varkappa_i,C_k} \boldsymbol{J}_{\varkappa_i,C_k}\Big)^{-1} \end{aligned} \tag{2}$$

where n map points are visible on camera C_k. $\boldsymbol{Cov}_{C_k}$ is the covariance matrix that represents the observability accuracy of camera C_k, and $\boldsymbol{J}_{\varkappa_i,C_k}$ is the Jacobian of observability measurement of camera C_k for point $\varkappa_i$. Translation is the most weakly estimated parameter in most of the cases, as described in the ORB-Atlas, and therefore, we try to obtain an error estimation of translation only with the diagonal values of $\boldsymbol{Cov}_{C_k}$. We consider the camera tracking to be lost either when we find the number of tracked points below the threshold or when we find the pose consistency to be very low, as described in Equation (2). Figure 2 shows an example of the utilization of pose consistency evaluation, where Figure 2a shows the GPS ground truth path on the Google map from the Malaga 07 [44] sequence and Figure 2b shows the erroneous path estimation using ORB-SLAM2. In Figure 2c, P3 indicates the position where the SLAM was initialized, moved towards the position P4 (white path), and took a u-turn, and the pose consistency failed at the position P1 and went into track-loss mode. The proposed pose consistency prevents the system from continuing with erroneous pose estimations that can decrease overall accuracy. Section 3.6 further describes the re-initialization process after losing the camera track and shows the overall improvement in accuracy for the example in Figure 2.

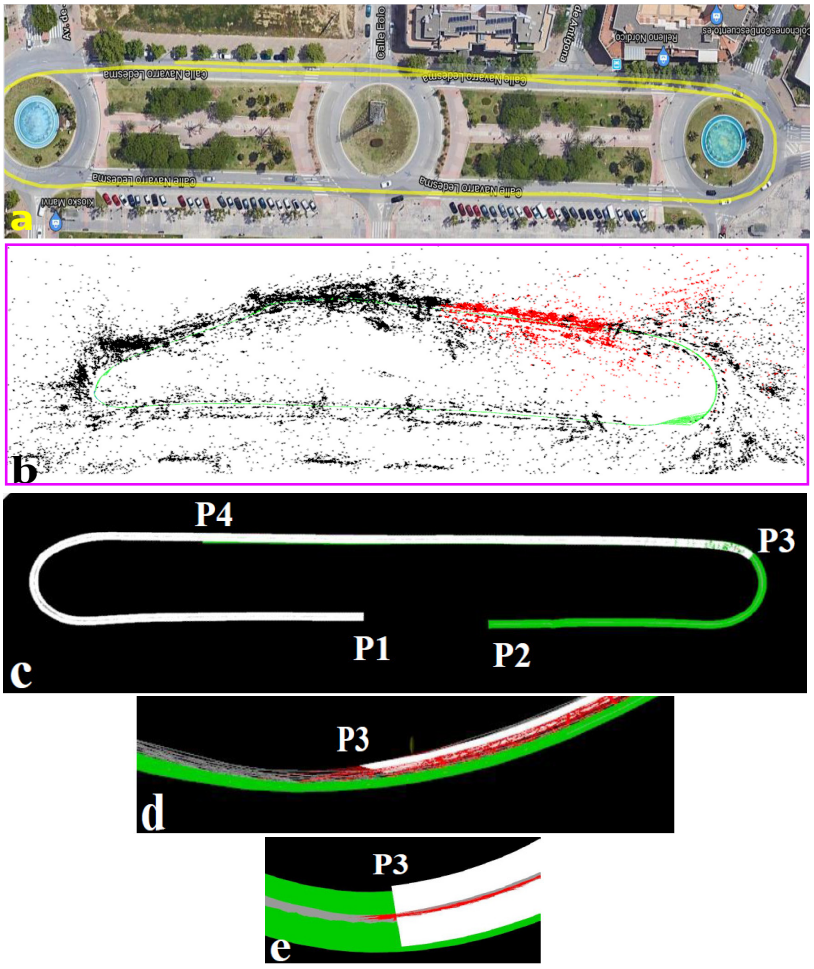

Figure 2. Accuracy improvement using pose-consistency evaluation. (**a**) The GPS ground truth path of Malaga 07 data set [44] on Google map. (**b**) The map generated using ORB-SLAM2 [3]. (**c**) The merged camera track with our pose-consistency evaluation. White and green show the two individual camera tracks. (**d**,**e**) Magnified views show map fusion accuracy.

3.6. Re-Initialization

A client may lose track of an incoming frame either due to erroneous pose estimation in VO, as described in Section 3.5, or due to the reduction in the number of 3D-2D matched features. The client tries to relocalize with its own local map for a small user-defined

duration, as in many cases we found that a sudden jerk creates track-loss and regains the track immediately after the jerk. The client goes into a track-loss mode if the client is unable to regain the track within the specified time. In this scenario, the client is left with two options: either it can continue trying to relocalize or it can reinitialize from the beginning and continue. If the client chooses the first option, autonomy is greatly affected because the client can only start the navigation after the relocalization, and it can impact the completion time of the entire mission. If the client chooses the second option, autonomy is restored, but the trajectory would not be continuous and there would be extra overhead to fuse multiple sub-maps. We opt for the second option, in which our autonomy is unaffected. In the present system, the client immediately sends a track-lost notification message to the server with the last KF id of the local map. The server creates a new map structure for that client in the corresponding map stack upon receiving the track-lost message and waits till the last KF is received. The server sends an acknowledgment message with the next available client id to the client immediately after the last KF is received. The client again reinitializes from the beginning with the new client id and a new local map and continues with VIO or VO based on its previous configuration. To avoid ambiguity, each local map uses a globally unique identifier, which is the associated client id. The message flow diagram is shown in Figure 3.

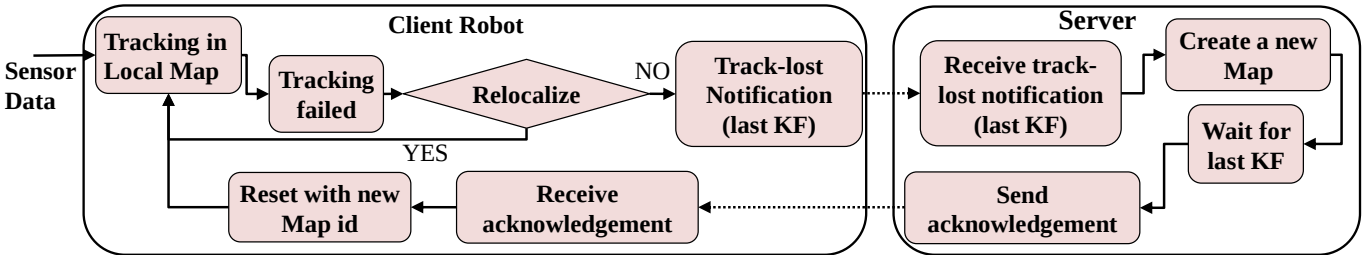

Figure 3. Block diagram of information flow after camera tracking fails.

Let us revisit the example in Figure 2, where the agent goes to the track-loss mode at position P1, as shown in Figure 2c. The agent is reinitialized at position P2 and continues tracking. Figure 2c shows the camera track (green path) after the re-initialization. The agent moves towards position P3, and the map-matching module finds matches between these two maps when the agent arrives at position P3. The map fusion module (Section 3.8) merges these two maps, and Figure 2c shows the overlapped regions from position P3 to position P4. The overall accuracy of our estimated camera track is greatly improved from the ORB-SLAM2 estimation, but the camera track becomes discontinuous between positions P1 and P2. The camera track enhancement and discontinuity are realized pictorially in Figure 2a–c.

3.7. Map Structure and Optimization

The local map consists of the nearest N number of KFs from the current location of the camera and is periodically updated by either inserting a new KF or updating the pose of an old KF, received from the server. Now the local map structure has KFs with two different types of connected constraints in the case of VO and VIO. We follow a similar structure as proposed in CCM-SLAM [15] for VO estimation. In the case of VIO, the KFs have connected constraints with IMU observation between consecutive KFs and covisibility constraints with common MPs visibilities. The KFs of map structures in the server have only covisibility constraints. Therefore, when an old KF is inserted into the local map in the client, it only contains covisibility constraints, and the local BA excludes such a KF for further optimization. The local BA runs on a local window of a fixed number of KFs, which is smaller than the local map size to ensure valid IMU constraints within a small boundary. The global BA on the server is a vision-only, BA where we make the scale fixed when any contributed client runs VIO.

3.8. Map Fusion

The place recognizer module uses the ORB-SLAM2 place-recognition technique and generates a pair of matched KFs along with the associated matched MPs from two different maps, either from the same client or from multiple clients. The fusion module is well aware of each client's sensors for estimating VIO or VO; therefore, it always fuses maps from a non-metric scale to a metric scale.

Let us assume F^s and F^d are the matched KFs from the maps M^s and M^d. The matched MPs generate one set of 3D–3D point correspondences from M^s to M^d and two sets of 3D–2D point correspondences from M^s to F^d and from M^d to F^s. There can be three situations: (1) both the maps are in the metric scale, (2) only map M^d is in the metric scale, and (3) no maps are in the metric scale.

Case 1: The 3D–3D correspondence set generates a $SE(3)$ transformation [45] $\boldsymbol{T1}_{M^s}^{M^d}$ and the 3D–2D point correspondence sets generate two more $SE(3)$ transformations $\boldsymbol{T2}_{M^s}^{M^d}$ and $\boldsymbol{T3}_{M^d}^{M^s}$, where $\boldsymbol{T3}_{M^s}^{M^d} = (\boldsymbol{T3}_{M^d}^{M^s})^{-1}$. Finally, an average $SE(3)$ transformation is formulated by rotation averaging [46] and translation averaging [47,48].

Case 2: M^d is only aligned in metric scale, meaning 3D–3D correspondences are not in the same scale. Therefore, we calculated the scale difference from M^s to M^d using Equation (3).

$$s = \frac{1}{n} \sum_{i,j \in \psi}^{n} \frac{E_{M_i^d, M_j^d}}{E_{M_i^s, M_j^s}} \tag{3}$$

where ψ is the 3D–3D point correspondence set, and $E_{M_i^x, M_j^x}$ denotes the Euclidean distance between 3D points i and j in map M^x. The scaled map $\hat{M}^s = sM^s$ and M^d generate a $SE(3)$ transformation $\boldsymbol{T1}_{\hat{M}^s}^{M^d}$ similar to the previous case. The 3D–2D point correspondence sets generate two more $Sim(3)$ [45] transformations, $\boldsymbol{S2}_{M^s}^{M^d}$ and $\boldsymbol{S3}_{M^d}^{M^s}$, and we calculate an average $Sim(3)$ transformation by rotation averaging and translation averaging.

Case 3: We follow a similar process as proposed in CCM-SLAM.

These calculated transformations in all three cases allow us to create a new map structure M^f, where map M^s is inserted after using the transformation and M^d enters directly. Both clients obtain access to the fused map M^f. Global BA [36] runs after map fusion. Figure 2c–e show an accurate map fusion using the proposed map fusion module.

3.9. Communication Bandwidth Requirement

Any new KF and MP that is created by a client is shared with the server with the whole data structure, including 2D features, their feature descriptors, and associated 3D map points. The average size is 56 KB for a new KF, considering 1000 feature points, and 200 bytes for a new MP. Any retransmission of an old KF or MP between client and server requires 148 bytes and 52 bytes, respectively, as any retransmission does not send old 2D feature points. The rest of the communication messages between the server and the client are of negligible bandwidth because the message passing executes on the occurrence of any event.

4. Experimental Results

We evaluate CORB2I-SLAM extensively on open sequences (EuRoC [49], Freiburg2 [50]) as well as in real autonomy with a total of five experiments. Section 4.1 presents Experiment 1 to show the accuracy of a single agent, and Section 4.2 presents Experiment 2 to show the map merging accuracy among multiple agents with homogeneous cameras. Section 4.3 presents Experiment 3 and 4 to show the map-merging accuracy among multiple agents with heterogeneous cameras, and finally, Section 4.4 presents Experiment 5, which shows the map fusion in real flight. Experiments 1–3 are performed on EuRoC [49] sequences and Experiment 4 is performed on Freiburg2 [50] sequences. Every client uses a standard laptop with an Intel Core i5-5200, four cores @2.2GHz and 4 GB of RAM while executing open

sequences. The server uses another laptop with an Intel Core i7-8750H, 12 cores @ 2.20 GHz, and 16 GB of RAM, and we found this server performs without any delay with six active clients, but it lags once the number of active clients increases further. The real autonomy is tested on a Tarot drone with an NVidia Jetson TX2 board with an Intel RealSense D435i RGB-D camera. The server remains the same laptop in real autonomy. The communication is over a dedicated 4G wireless network. We assign values to multiple parameters experimentally, where we keep N = 30 KFs in the local map for on-board VIO/VO calculation and 15 KFs for local BA in all our experiments. The errors are estimated as the Root Mean Square (RMS) of Absolute Translation Error (ATE) and presented as the average value of 10 executions.

4.1. Experiment 1: Single-Agent Accuracy

We evaluate the basic accuracy of CORB2I-SLAM on a single agent, where CORB2I-SLAM is evaluated with IMU and without IMU. We calculate the RMS ATE and compare it with the SoA VIO- or VO-based methods. Table 1 presents the details of the comparisons, where the values of CVI-SLAM [32] are taken from the author's publication because of a closed source. The estimated trajectories are aligned with an $SE(3)$ transformation before the ATE calculation. The symbol 'X' indicates that the open source implementation either does not support such a configuration or is not present in the author's publication. Vision-only CORB2I-SLAM experienced three tracking failures while executing the V203 sequence, resulting in the creation of new maps. Therefore, the value in Table 1 shows up after all four maps are merged. The accuracy of CORB2I-SLAM is comparable with that of SoA. ORB-SLAM3 [33] shows marginally better accuracy in some cases because of better loop corrections. The accuracy of CORB2I-SLAM (without IMU) indicates that using an IMU is not always beneficial, as IMU measurements are noisy in many cases, and the CORB2I-SLAM framework is adaptive to detect and reject very noisy IMU measurements.

Table 1. The RMS ATE (meter) of single agent for experiments on EuRoC sequences [49]. '⇐', '⇔', and '⇳' indicate monocular, stereo, and IMU usage, respectively. '*' indicates the trajectories are scaled off-line because the metric scale is not present.

Data Set	VINS ⇐⇳	CCM * ⇐	CVI ⇐⇳	ORB-SLAM3 ⇔⇳	CORB2I ⇔⇳	CORB2I ⇔
MH01	0.120	0.113	0.085	0.036	0.035	**0.034**
MH02	0.120	0.089	0.063	**0.033**	0.034	0.037
MH03	0.102	0.078	0.065	**0.035**	0.036	0.036
MH04	0.155	0.138	0.293	0.051	**0.045**	0.133
MH05	0.136	0.129	0.081	0.082	**0.059**	0.078
V103	0.190	X	X	0.024	**0.023**	0.048
V203	0.220	X	X	0.024	**0.022**	0.129

4.2. Experiment 2: Multiple Agents Map Fusion Accuracy

We evaluate the map fusion accuracy on EuRoC sequences [49], where multiple clients run in parallel. Table 2 presents a comparative analysis with SoA, where the comparison shows a good improvement in accuracy with the merged map. This is due to our novel multi-constrained map fusion approach and a strongly constrained loop-closer correction. Information sharing among clients helps in more accurate localization. The comparison clearly shows that accuracy increases in collaboration as compared with the values in Table 1.

Table 2. The RMS ATE (meter) of multiple agents for experiments on EuRoC sequences. † indicates that the sequences are run sequentially because the system is designed for a single agent and the rest of the symbols have the same meanings as in Table 1.

Data Set	VINS † ⇐⇳	CCM * ⇐	CVI ⇐⇳	ORB-SLAM3 † ⇔⇳	CORB2I ⇔⇳	CORB2I ⇔
MH01,02	0.159	0.097	0.050	0.035	**0.028**	0.036
MH01,02, 03	0.192	0.092	X	0.037	**0.029**	0.035
MH01,02, 03,04	0.239	0.079	X	0.051	**0.034**	0.069
MH01,02, 03,04,05	0.278	X	X	0.086	**0.035**	0.052

Figure 4 shows the outcome of two experiments pictorially. Figure 4a shows a snapshot of the first experiment on EuRoC MH04 and MH05 sequences running VIO on two different clients (trajectories are in white and green), where white covisibility edges represent KFs from one client, whereas red covisibility edges show KFs from multiple clients and MPs. The inlet shows the magnified view of the merged map. Figure 4b shows a snapshot of the second experiment on EuRoC MH01 and MH02 sequences running on two different clients. The client executing the MH01 sequence has limited computing power and fails to track twice due to frame skipping, but reinitializes immediately. The camera trajectories of sub-maps are shown in white, blue, and purple. CORB2I-SLAM merges all sub-maps.

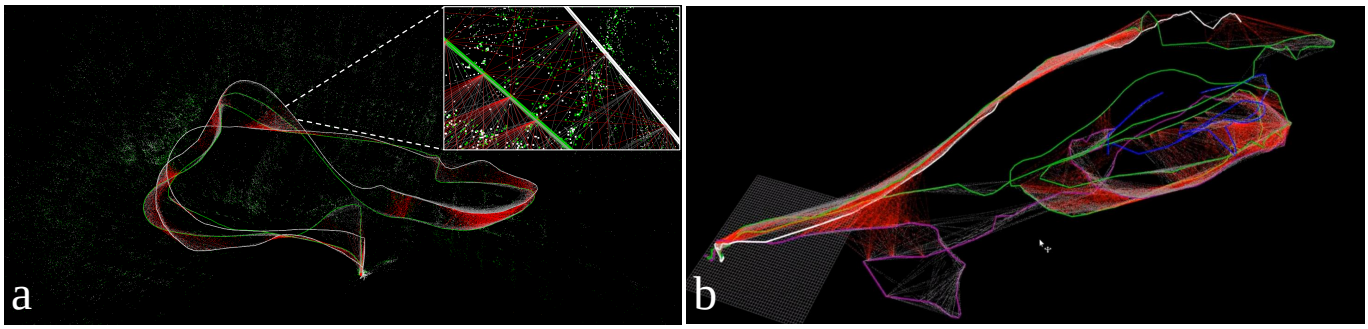

Figure 4. Snapshots of the CORB2I-SLAM map fusion experiment on EuRoC sequences [49]. (**a**) Map fusion among multiple agents. (**b**) Map fusion in a single agent as well as multiple agents.

4.3. Experiment 3 and 4: Map Fusion with Heterogeneous Sensor

We use two sequences from the Freiburg2 data set [50] in these experiments where we run CORB2I-SLAM without IMU integration. Experiment 3 and experiment 4 use the same data sequences in the clients, but client_1 runs VO on monocular images in experiment 3 and RGB-D images in experiment 4. Table 3 presents the quantitative details of these two experiments. The RMS ATE values in our collaborative framework show great improvement in the case of experiment 3 compared with experiment 4. The reason for the higher accuracy is early map fusion. The values show that localization estimation improves a great deal in the case of faster map fusion or information sharing.

Table 3. Quantitative details for experiment on Freiburg2 sequences.

	Experiment 3		**Experiment 4**	
	Client_1	**Client_2**	**Client_1**	**Client_2**
Sequence	*fr2/xyz*	*fr2/rpy*	*fr2/xyz*	*fr2/rpy*
Sensor Type	Monocular	RGB-D	RGB-D	RGB-D
Merging time offset (sec)	31.84		46.27	
Single agent RMS ATE (m)	0.002438	0.033917	0.0046	0.033917
CORB2I-SLAM RMS ATE (m)	**0.008357**		**0.034113**	

4.4. Experiment 5: Map Fusion on Real Autonomy

We use one Tarot drone in the outdoor and navigate using preset GPS way points. We found the GPS point-based navigation to be quite erroneous; therefore, ground-truth verification was excluded. Two clients run sequentially on a single drone. The start coordinates of the two sequences were kept 3 meters apart, and each trajectory was about 55 meters long. In this experiment, we evaluate the proposed adaptive VIO- or VO-selection mechanism. We use a noisy mem-based IMU to test the framework, where the IMU based initialization is unsuccessful every time and continues with monocular VO. Here, we choose monocular instead of RGB-D because of the long-range depth, which is beyond the depth-sensing range of RealSense D435i. The camera tracking fails multiple times for both the clients due to fast rotation, but the clients are reinitialized. The framework is able to fuse the sub-maps of client_2 with the biggest sub-map of client_1 after 11.3 seconds of client_2 execution. Figure 5 shows the intermediate trajectories and the fused trajectories with MPs.

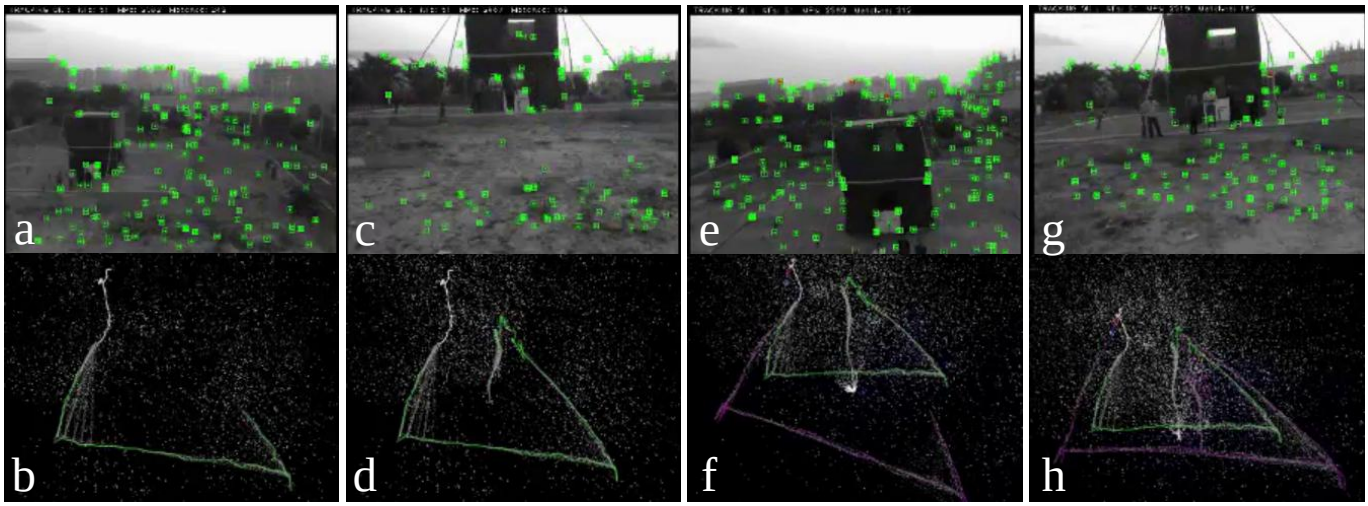

Figure 5. Test on real autonomy: Each column shows the camera view and corresponding map. (**a**,**c**) show camera views of client_1. (**e**,**g**) show camera views of client_2. (**b**,**d**) show the camera trajectories of client_1. (**f**) shows the camera trajectories of client_1 and client_2 before fusion. (**h**) shows the entire fused map of client_1 and client_2.

4.5. Execution Time

We evaluate CORB2I-SLAM execution on EuRoC MH01 to MH04 sequences. The on-board execution of any client is independent of the total number of participating clients; therefore, the execution time is similar for single and multiple clients. The tracking thread takes on average 36 ms, the mapping thread takes 240 ms to 245 ms, and communication takes about 0.28 ms. Server performance changes with the number of KFs to be processed for

an operation. Figure 6 shows the average execution time for computing the transformation for a loop closing and map merging.

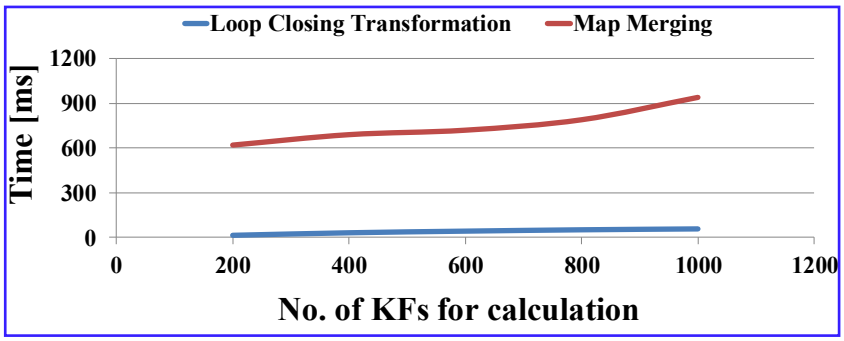

Figure 6. The server execution time against a given number of KFs to be processed. The values are tested on EuRoC MH01 to MH04 sequences with an average of 5 executions.

5. Conclusions

We present a novel architecture of a centralized collaborative SLAM framework for heterogeneous vision sensors and inertial sensors. In this centralized architecture, client robots are considered with limited computation capabilities and a central server with higher computation capabilities. The framework allows the clients to run either VO or VIO independently without any server dependency, and it is designed to adapt to detect noisy inertial sensors and exclude them in pose estimation. We proposed a new criterion to estimate the accuracy of poses and reinitialize with a sub-map in the case of tracking being inaccurate. The sub-maps can be fused into a single map once a match is found among multiple maps. We also propose a novel map-merging procedure between a non-metric scale map and a metric scale map that produces better accuracy compared to SoA techniques. We evaluate our proposed framework extensively on open data sets as well as in real flight and show better accuracy. The CORB2I-SLAM may not have better accuracy on sudden illumination changes or low-light conditions. We keep such enhancements in the scope of future work.

Author Contributions: Original Draft Preparation, Conceptualization, Methodology, Formal Analysis, A.S.; Supervision, Investigation, Methodology B.C.D.; Conceptualization, Formal Analysis S.U.; Validation K.Y.; Acquisition J.M.A.; Funding A.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Researchers Supporting Project (No. RSP-2021/395), King Saud University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Approval by an ethics committee or institutional review board is not required for this manuscript. This research respects all the sentiments, dignity, and intrinsic values of animals or humans.

Informed Consent Statement: Not applicable.

Data Availability Statement: In this manuscript, the employed data sets were taken with license agreements from the corresponding institutions with proper channels.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Scaramuzza, D.; Fraundorfer, F. Visual Odometry [Tutorial]. *IEEE Robot. Autom. Mag.* **2011**, *18*, 80–92. [CrossRef]
2. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
3. Mur-Artal, R.; Tardos, J. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
4. Maity, S.; Saha, A.; Bhowmick, B. Edge slam: Edge points based monocular visual slam. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops, Venice, Italy, 22–29 October 2017.

5. Yang, S.; Scherer, S. Direct Monocular Odometry Using Points and Lines. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3871–3877.
6. Fraundorfer, F.; Scaramuzza, D. Visual Odometry: Part II - Matching, Robustness, and Applications. *IEEE Robot. Autom. Mag.-IEEE Robot. Autom.* **2012**, *19*, 78–90. [CrossRef]
7. Alkendi, Y.; Seneviratne, L.; Zweiri, Y. State of the Art in Vision-Based Localization Techniques for Autonomous Navigation Systems. *IEEE Access* **2021**, *9*, 76847–76874. [CrossRef]
8. Leutenegger, S.; Furgale, P.; Rabaud, V.; Chli, M.; Konolige, K.; Siegwart, R. Keyframe-based visual-inertial SLAM using nonlinear optimization. *Int. J. Robot. Res.* **2015**, *34*, 314–334. [CrossRef]
9. Yang, Y.; Geneva, P.; Eckenhoff, K.; Huang, G. Visual-Inertial Odometry with Point and Line Features. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 2447–2454. [CrossRef]
10. Scaramuzza, D.; Zhang, Z. Visual-Inertial Odometry of. In *Encyclopedia of Robotics*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 1–9. [CrossRef]
11. Li, H.; Stueckler, J. Visual-Inertial Odometry With Online Calibration of Velocity-Control Based Kinematic Motion Models. *IEEE Robot. Autom. Lett.* **2022**, *7*, 6415–6422. [CrossRef]
12. Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A. Ros: An open-source robot operating system. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009.
13. Chen, W.; Shang, G.; Ji, A.; Zhou, C.; Wang, X.; Xu, C.; Li, Z.; Hu, K. An Overview on Visual SLAM: From Tradition to Semantic. *Remote. Sens.* **2022**, *14*, 3010. [CrossRef]
14. Macario Barros, A.; Michel, M.; Moline, Y.; Corre, G.; Carrel, F. A Comprehensive Survey of Visual SLAM Algorithms. *Robotics* **2022**, *11*, 24. [CrossRef]
15. Schmuck, P.; Chli, M. CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *J. Field Robot.* **2018**, *36*, 763–781. [CrossRef]
16. Loianno, G.; Mulgaonkar, Y.; Brunner, C.; Ahuja, D.; Ramanandan, A.; Chari, M.; Diaz, S.; Kumar, V. A swarm of flying smartphones. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 1681–1688.
17. Schwager, M.; Dames, P.; Rus, D.; Kumar, V. A multi-robot control policy for information gathering in the presence of unknown hazards. In *Robotics Research*; Springer: Cham, Switzerland, 2017; pp. 455–472.
18. Kushleyev, A.; Mellinger, D.; Powers, C.; Kumar, V. Towards a swarm of agile micro quadrotors. *Auton. Robot.* **2013**, *35*, 287–300. [CrossRef]
19. Choudhary, S.; Carlone, L.; Nieto, C.; Rogers, J.; Christensen, H.; Dellaert, F. Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. *Int. J. Robot. Res.* **2017**, *36*, 1286–1311. [CrossRef]
20. Egodagamage, R.; Tuceryan, M. Distributed monocular slam for indoor map building. *J. Sens.* **2017**. [CrossRef]
21. Giamou, M.; Khosoussi, K.; How, J. Talk resource-efficiently to me: Optimal communication planning for distributed loop closure detection. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 1–9.
22. Cieslewski, T.; Choudhary, S.; Scaramuzza, D. Data-efficient decentralized visual slam. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2466–2473.
23. Zhang, H.; Chen, X.; Lu, H.; Xiao, J. Distributed and collaborative monocular simultaneous localization and mapping for multi-robot systems in large-scale environments. *Int. J. Adv. Robot. Sys.* **2018**, *15*. [CrossRef]
24. Zou, D.; Tan, P. Coslam: Collaborative visual slam in dynamic environments. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 354–366. [CrossRef] [PubMed]
25. Forster, C.; Lynen, S.; Kneip, L.; Scaramuzza, D. Collaborative monocular slam with multiple micro aerial vehicles. In Proceedings of the International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 3962–3970.
26. Riazuelo, L.; Civera, J.; Montiel, J. Coslam: Collaborative visual slam in dynamic environments. *Robot. Auton. Syst.* **2014**, *62*, 401–413. [CrossRef]
27. Klein, G.; Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. In Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan, 13–16 November 2007; pp. 225–234.
28. Deutsch, I.; Liu, M.; Siegwart, R. A framework for multi-robot pose graph slam. In Proceedings of the IEEE International Conference on Real-time Computing and Robotics (RCAR), Angkor Wat, Cambodia, 6–10 June 2016; pp. 567–572.
29. Elvira, R.; Tardos, J.; Montiel, J. Orbslam-atlas: A robust and accurate multi-map system. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 6253–6259.
30. Ouyang, M.; Shi, X.; Wang, Y.; Tian, Y.; Shen, Y.; Wang, D.; Wang, P. A Collaborative Visual SLAM Framework for Service Robots. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 8679–8685.
31. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]

32. Karrer, M.; Schmuck, P.; Chli, M. Cvi-slam-collaborative visual-inertial slam. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2762–2769. [CrossRef]
33. Campos, C.; Elvira, R.; Rodriguez, J.; Montiel, J.; Tardos, J. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
34. Liu, J.; Liu, R.; Chen, K.; Zhang, J.; Guo, D. Collaborative Visual Inertial SLAM for Multiple Smart Phones. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 11553–11559. [CrossRef]
35. Schmuck, P.; Ziegler, T.; Karrer, M.; Perraudin, J.; Chli, M. COVINS: Visual-Inertial SLAM for Centralized Collaboration. In Proceedings of the 2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Bari, Italy, 4–8 October 2021; IEEE Computer Society: Los Alamitos, CA, USA, 2021; pp. 171–176. [CrossRef]
36. Triggs, B.; McLauchlan, P.; Hartley, R.; Fitzgibbon, A. Bundle Adjustment—A Modern Synthesis. In *International Workshop on Vision Algorithms: Theory and Practice*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 298–372.
37. Matsuki, H.; Scona, R.; Czarnowski, J.; Davison, A.J. CodeMapping: Real-Time Dense Mapping for Sparse SLAM using Compact Scene Representations. *IEEE Robot. Autom. Lett.* **2021**, *6*, 7105–7112. [CrossRef]
38. Zhang, S.; Li, S.; Hao, A.; Qin, H. Knowledge-inspired 3D Scene Graph Prediction in Point Cloud. In *Proceedings of the Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: New York, NY, USA, 2021; Volume 34, pp. 18620–18632.
39. Wu, F.; Yan, F.; Shi, W.; Zhou, Z. 3D scene graph prediction from point clouds. *Virtual Real. Intell. Hardw.* **2022**, *4*, 76–88. [CrossRef]
40. Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. On-Manifold Preintegration for Real-Time Visual–Inertial Odometry. *IEEE Trans. Robot.* **2017**, *33*, 1–21. [CrossRef]
41. Chen, T.; Wang, Q. 3D Line Segment Detection for Unorganized Point Clouds from Multi-view Stereo. In *Asian Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 400–411.
42. Lin, Y.; Wang, C.; Cheng, J.; Chen, B.; Jia, F.; Chen, Z.; Li, J. Line segment extraction for large scale unorganized point clouds. *ISPRS J. Photogramm. Remote Sens.* **2015**, *102*, 172–183. [CrossRef]
43. Tian, P.; Hua, X.; Tao, W.; Zhang, M. Robust Extraction of 3D Line Segment Features from Unorganized Building Point Clouds. *Remote Sens.* **2022**, *14*, 3279. [CrossRef]
44. Blanco, J.; Moreno, F.; Gonzalez-Jimenez, J. The malaga urban dataset: High-rate stereo and lidars in a realistic urban scenario. *Int. J. Robot. Res.* **2014**, *33*, 207–214. [CrossRef]
45. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2004.
46. Hartley, R.; Trumpf, J.; Dai, Y.; Li, H. Rotation averaging. *Int. J. Comput. Vis.* **2013**, *103*, 267–305. [CrossRef]
47. Cui, Z.; Tan, P. Global structure-from-motion by similarity averaging. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 864–872.
48. Zhuang, B.; Cheong, L.; Lee, G. Baseline desensitizing in translation averaging. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4539–4547.
49. Burri, M.; Nikolic, J.; Gohl, P.; Schneider, T.; Rehder, J.; Omari, S.; Achtelik, M.; Siegwart, R. The EuRoC MAV Datasets. *Int. J. Robot. Res.* **2016**, *35*, 1157–1163. [CrossRef]
50. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A Benchmark for the Evaluation of RGB-D SLAM Systems. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 573–580.