CVI-SLAM – Collaborative Visual-Inertial SLAM

Marco Karrer*, Patrik Schmuck* and Margarita Chli
Vision for Robotics Lab, ETH Zurich, Switzerland

***Abstract*—With robotic perception constituting the biggest impediment before robots are ready for employment in real missions, the promise of more efficient and robust robotic perception in multi-agent, collaborative missions can have a great impact many robotic applications. Employing a ubiquitous and well-established visual-inertial setup onboard each agent, in this paper we propose CVI-SLAM, a novel visual-inertial framework for centralized collaborative SLAM. Sharing all information with a central server, each agent outsources computationally expensive tasks, such as global map optimization to relieve onboard resources and passes on measurements to other participating agents, while running visual-inertial odometry onboard to ensure autonomy throughout the mission. Thoroughly analyzing CVI-SLAM, we attest to its accuracy and the improvements arising from collaboration, and evaluate its scalability in the number of participating agents and applicability in terms of network requirements.**

***Index Terms*—Multi-Robot Systems, SLAM, Visual-Based Navigation**

I. INTRODUCTION

WITH state-of-the-art Simultaneous Localization And Mapping (SLAM) systems having reached substantial robustness and accuracy in the centimeter range [1] for single-robot applications, multi-robot systems have been gaining growing popularity in numerous scenarios, ranging from search-and-rescue applications to digitization of archeological sites. Increasing the robustness of the system by sharing information amongst the participants, boosting the efficiency of a mission by dividing up a task or enabling tasks otherwise impossible for a single robot are only some of the advantages a team of robots has to offer. At the same time, multi-robot scenarios pose significant challenges, such as dealing with network characteristics (e.g. time delays and bandwidth) and ensuring transparent and consistent information access among all agents. In this spirit, this paper proposes CVI-SLAM, a centralized ***C***ollaborative SLAM system for multiple robotic agents, each equipped with a ***V***isual-***I***nertial sensor suite, and a central ground station, the "server". Fig. 1 shows a snapshot of our proposed system. Taking the powerful monocular setup of [2] a step further towards real deployment, CVI-SLAM agents employ a visual-inertial sensor suite, enabling metric scale estimation and gravity alignment of the estimated trajectories and maps, which is necessary for autonomous exploration of an environment, guaranteeing higher accuracy and robustness compared to monocular SLAM [3], [4], due

*Marco Karrer and Patrik Schmuck contributed equally to this work

Manuscript received: February 24, 2018; Accepted April 23, 2018.

This paper was recommended for publication by Editor Nak Young Chong upon evaluation of the Associate Editor and Reviewers' comments.

This research was supported by the Swiss National Science Foundation (SNSF, Agreement no. PP00P2157585), EC's Horizon 2020 Programme under grant agreement n. 644128 (AEROWORKS), and NCCR Robotics.

Digital Object Identifier (DOI): see top of this page.

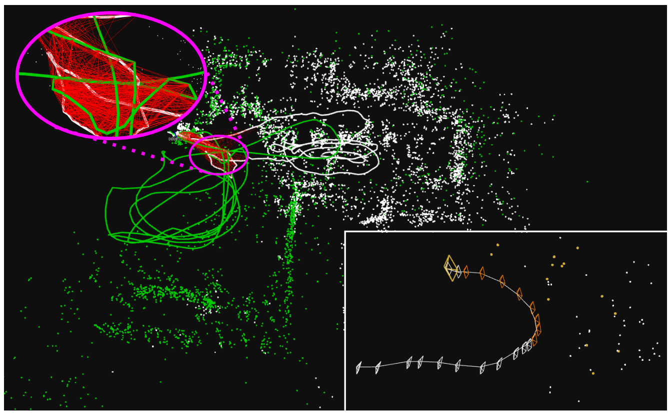
Fig. 1: A snapshot of CVI-SLAM with two agents collaboratively building the map. Trajectories and map points are colored in white and green for agent 1 and 2, respectively. Covisibility constraints across different agents are indicated in red. The inlet on the bottom right depicts the limited local map of agent 1 with the newest frame and its observed map points colored in yellow, and keyframes which received pose updates from the server in orange.

to the complimentary nature of the sensors. Similarly to [2], here information is consequently shared amongst all agents. However, incorporating an Inertial Measurement Unit (IMU) increases the complexity of the system, since more data has to be processed, and sharing and removing of data becomes more difficult, since IMU constraints depend on the relative timings between measurements. Furthermore, camera images and IMU measurements need to be synchronized, varying IMU bias terms need to be constantly estimated, and for accurate results, the manifold structure of the rotation group $SO(3)$ needs to be addressed. Therefore, we use a new visual-inertial odometry system designed for CVI-SLAM, implementing on-manifold pre-integration of IMU measurements [5], having shown its beneficial characteristics for IMU handling already in other SLAM systems [4], [6]. We thoroughly evaluate CVI-SLAM on a public dataset in scenarios with one to four agents on aerial platforms, as some of the most challenging platform for robotic perception (due to their high agility and constrained resources). Our analysis discusses the bandwidth requirements for CVI-SLAM and its scalability to the number of agents. Attesting to the accuracy of CVI-SLAM and improved performance from sharing data, we compare the collaborative trajectory estimates against ground truth, exhibiting equivalent performance to other state-of-the-art SLAM systems in single agent scenarios, while outperforming these system in the multi-agent applications.

II. RELATED WORK

While several works in the literature deal with either collaborative localization [7], [8], [9] or collaborative mapping [10], [11], [12], only few existing works are able to perform collaborative SLAM with multiple agents. Eliminating the need of a pre-computed map (as in collaborative localization) or known robot poses (required in collaborative mapping), collaborative SLAM promises to exploit the full spectrum of possibilities for robot collaboration in far more generic and realistic setups.

A centralized architecture for robotic collaboration is usually employed in the literature when it comes to systems applied to practical scenarios. However, some works tackle collaborative SLAM in a decentralized manner, such as [13], proposing a fully distributed SLAM system evaluated in simulation, emulating a sensor setup with visual, inertial and GPS sensors. An efficient place recognizer distributed amongst all agents with real data, but in simulation was shown in [14]. Most recently, Choudhary et al. [15] showed a decentralized SLAM system where co-localization of the participating robots is based on commonly observed pre-trained objects.

Guaranteeing data consistency and avoiding double-counting are the biggest challenges in a decentralized setup, whereas a centralized system has a clearer allocation of information. Furthermore, a centralized client-server-architecture allows agents to outsource non time-critical, but computationally expensive algorithms, such as global map optimization, to the server, which is potentially much more powerful. This allows an agent to allocate its potentially limited onboard resources to the most critical tasks, such as visual odometry.

Probably the most powerful vision-only collaborative SLAM system is CoSLAM [16], which has the ability to handle dynamic environments, albeit at high computational cost, requiring GPUs, often prohibitive in resource-constraint robots. Receiving image data from multiple monocular cameras as input, CoSLAM groups cameras with scene overlap, relying on the assumptions that all cameras are synchronized and observe the same scene at initialization. Together with the requirement for a GPU, these assumptions render CoSLAM impractical to run online onboard multiple robots.

In an earlier attempt to multi-robot collaboration, [17] extended a structure from motion pipeline to collaborative SLAM for UAVs, with each agent running a keyframe-based visual odometry sending all keyframes to a central server. The server would search for overlap across maps and merge them if necessary. While impressive, this system fell short of sending any feedback from the server back to the agents and therefore, cannot profit from optimization results and data from other agents.

In a system more suitable for multi-robot collaboration, C^{2}TAM [18] proposed to perform position tracking onboard each agent, while all mapping tasks are run on the server. The server then sends the complete map to each agent for further tracking steps, enabling operation with agents with very limited computational resources. However, C^{2}TAM's assumption that an agent is always in communication with the server heavily restricts the agent's autonomy, while the assumption of being able to repeatedly send the whole map to an agent further restricts C^{2}TAM's practicality and generality. Targeting multi-device mapping applications with hand-held devices, MOARSLAM [19] proposed to employ a server to act as central memory for storing and distributing data amongst agents, with each agent executing all parts of a full SLAM system (i.e. visual odometry, place recognition and global map optimization) onboard. While employing a visual-inertial setup, MOARSLAM only uses the IMU as one of two alternatives for scale disambiguation during pose estimation from visual odometry (with stereo images being the second option). The server back-end presented by Deutsch et al. [20] can be run on a server to combine different SLAM systems in a collaborative framework. Shifting all intelligence and computation to the agents, [19] and [20] do not exploit the centralized architecture to its full potential, restricting its employment with powerful agents. Instead, CVI-SLAM outsources tasks that are computationally too expensive for the resource-limited agents to make full use of the potential of a centralized collaborative architecture, while ensuring that all tasks critical to the autonomy of each agent are still run onboard, as opposed to [18]. In contrast to systems sending no [17] or only partial [20], [21] feedback to the agents, CVI-SLAM consequently promotes full transparency of information.

Proposing collaborative monocular SLAM, [2] showed a proof of concept of a powerful centralized architecture suitable for resource-constraint platforms. Inspired by the extent of collaboration that this architecture can enable, in this work we propose a system to fuse visual and inertial information from each agent to a globally consistent map that can be reused in full or in parts by each agent. Adding to the challenge of consistent sharing of data and efficient handling of multiple agents, CVI-SLAM enables collaborative SLAM estimation with metric scale and high accuracy, and boosts the robustness and scalability of the system employing redundancy detection and removal at global scope.

III. PRELIMINARIES

A. Notation

For the denotion of vectors we use bold small letters (e.g. $\boldsymbol{a}$), while matrices are denoted by bold capital letters (e.g. $\boldsymbol{A}$). To distinguish different coordinate frames we use capital letters (e.g. A). As a result, a vector expressed in A is denoted by ${}_A\boldsymbol{x}$. For the coordinate frames, S denotes the IMU body frame, C the camera coordinate system and W for the origin (i.e. the inertial frame). The summarization of sets of variables is denoted by calligraphic letters (e.g. $\mathcal{A}$). To denote a rigid body transformation from coordinate frame B into A we use the notation $\boldsymbol{T}_{AB}$, where the rotational and the translational part of $\boldsymbol{T}$ are denoted by $\boldsymbol{R}$ and $\boldsymbol{t}$, respectively.

B. IMU Model and System States

In this paper, we use the standard IMU measurement model, assuming that measurements from both the accelerometer ${}_S\boldsymbol{a}(t)$ and the gyroscope ${}_S\boldsymbol{\omega}_{WS}(t)$ are corrupted by additive white noise $\boldsymbol{\eta}$ and have an unknown, time varying sensor bias $\boldsymbol{b}$:

$$ {}_S\boldsymbol{a}(t) = \boldsymbol{R}^{\intercal}_{WS}(t)\left({}_W\hat{\boldsymbol{a}}(t) - {}_W\boldsymbol{g}\right) + \boldsymbol{b}_a(t) + \boldsymbol{\eta}_a(t) \ , \quad (1) $$

$$ {}_S\boldsymbol{\omega}_{WS}(t) = {}_S\hat{\boldsymbol{\omega}}_{WS}(t) + \boldsymbol{b}_g(t) + \boldsymbol{\eta}_g(t) \ . \quad (2) $$

The true values of the respective variables are indicated by $\hat{\cdot}$, while ${}_W\boldsymbol{g}$ denotes the gravity vector. In order to account for these characteristics of the IMU measurements, the system state, denoted by $\boldsymbol{\Theta}$, besides the keyframe (KF) poses $\{\boldsymbol{R}_{WS}, \boldsymbol{t}_{WS}\}$ and map point (MP) positions ${}_W\boldsymbol{l}$ also includes the linear velocities ${}_W\boldsymbol{v}$ and bias terms $\boldsymbol{b}$:

$$\boldsymbol{\Theta} := \{\underbrace{\boldsymbol{R}^k_{WS}, \boldsymbol{t}^k_{WS}, {}_W\boldsymbol{v}^k, \boldsymbol{b}^k}_{\mathrm{KF}_k}, {}_{S_r}\boldsymbol{l}^i\} \quad \forall k \in \mathcal{V}, \forall i \in \mathcal{L} \ , \tag{3}$$

where the set of all KFs and all MPs are denoted by $\mathcal{V}$ and $\mathcal{L}$, respectively. In this paper, we denote individual state variables as θ_j when appropriate. As proposed in [22], instead of expressing the MPs in a global reference frame, we express them in coordinates of a reference KF S_r. While in our system we employ the inverse depth parameterization for the MPs, for the sake of readability in the following we treat the MPs as if they were expressed in Euclidean coordinates.

IV. METHODOLOGY

A. System Architecture

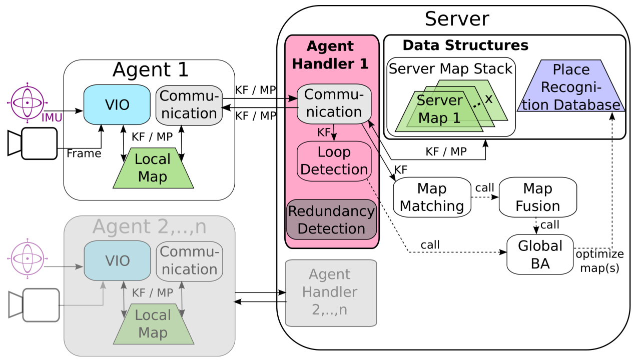

Fig. 2: Overview of the CVI-SLAM's system architecture. An agent runs real-time *visual-inertial odometry* (VIO) maintaining a *local map*. A *communication interface* is used to exchange *keyframes* (KF) and *map points* (MP) between the agent and the server. The server is a powerful computer that runs computationally expensive and non time-critical tasks: *redundancy detection*, *loop detection*, *map fusion*, and *bundle adjustment* (BA).

CVI-SLAM uses the basic system architecture shown in Fig. 2, which was first introduced in [2]. Here, however, each agent runs the *visual-inertial odometry* (VIO) system discussed in Sec. IV-C onboard, and the messages and processes are adapted to handling visual and inertial data. VIO estimates the local trajectory of each robot and maintains a *local map* limited to a fixed number of N KFs of the immediate surroundings of the robot. Furthermore, a *communication interface* on each agent module serves as the interface from and to the server. The server can communicate with all agents, coordinating the exchange of information amongst them. The server maintains maps of unbounded size (*server maps*) on a *server map stack*, holding all data that was collected by all agents throughout the mission. Starting with one server map for each agent, the server merges maps throughout the mission to associate the data from the individual agents. The *agent handlers* on the server side, one for each individual agent, distribute the information from their corresponding agent to the correct modules on the server. The server runs two place recognition modules, a *loop detection* module to detect loop closure inside one map, and a *map matching* module to detect overlap between distinct maps, allowing to merge these individual maps. A *keyframe database* implements an efficient look-up procedure allowing for a new KF to query other KFs, using an inverted file index. A successful query of the place recognition modules is then followed by a global optimization step (aka *Bundle Adjustment* (BA)). All optimization schemes are implemented using the Ceres[1] solver. An in-depth discussion of the system architecture can be found in [2]. In addition to the handling of IMU data, here we use an extended version of this architecture employing the *redundancy detection* module discussed in Sec. IV-E.

B. Error Residuals Formulation

The problem of KF-based VI-SLAM can be expressed as a nonlinear weighted least-squares optimization, where the state variables θ_j are optimized w.r.t. to some observation measurements $\boldsymbol{z}_i$. With the definition of a residual error as:

$$\boldsymbol{e}_i := \boldsymbol{z}_i - h_i(\mathcal{A}_i) \ , \tag{4}$$

where $\mathcal{A}_i$ denotes all variables θ_j involved in measurement $\boldsymbol{z}_i$ and $h_i(\cdot)$ is a function that predicts the measurement according to the current state. Using this notation, the objective that is minimized can be written as:

$$\begin{aligned}\boldsymbol{\Theta}^* &= \underset{\boldsymbol{\Theta}}{\arg\min} \left\{ \sum_i \|\boldsymbol{z}_i - h_i(\mathcal{A}_i)\|^2_{\boldsymbol{W}_i} \right\} \\ &= \underset{\boldsymbol{\Theta}}{\arg\min} \left\{ \sum_i \boldsymbol{e}_i^\intercal \boldsymbol{W}_i \boldsymbol{e}_i \right\} \ ,\end{aligned} \tag{5}$$

where $\|\boldsymbol{x}\|^2_{\boldsymbol{W}} = \boldsymbol{x}^\intercal \boldsymbol{W} \boldsymbol{x}$ denotes the squared Mahalanobis distance in information form. Within our VI-SLAM system, we use essentially three different types of residuals:

1) Reprojection Residual: With a given MP position ${}_{S_r}\boldsymbol{l}^j$ expressed in the reference KF's IMU frame S_r, the KF poses for both KF$_k$ and KF$_r$ and the keypoint observation $\boldsymbol{z}^{k,j}$ in the image of KF$_k$, we define the reprojection residual as:

$$\boldsymbol{e}^{k,j}_r := \boldsymbol{z}^{k,j} - g\left(\boldsymbol{K}^k \boldsymbol{T}_{CS} \boldsymbol{T}^k_{SW} \boldsymbol{T}^r_{WS} {}_{S_r}\boldsymbol{l}^j\right) \ , \tag{6}$$

where $g(\cdot)$ is a function to convert homogeneous into image coordinates and $\boldsymbol{K}^k$ represents the camera intrinsics matrix. Note that in this paper we use undistorted keypoints, hence, Eq. (6) does not contain a distortion model.

2) IMU pre-integration Residual: With a given sequence of IMU readings between two consecutive KFs, both from the accelerometer and the gyroscope, integration of the measurements can be performed in order to obtain a relative constraint between the KFs. In order to avoid re-integration of the measurements upon changes in the bias terms, [5] presented a method allowing to perform the integration only once and apply linearized corrections considering changes of the biases after the integration. With a given pre-integration, the resulting residuals can be written as

[1]http://ceres-solver.org

$$\begin{aligned}\boldsymbol{e}_{\Delta \boldsymbol{R}}^{k-1,k} &= \log\left(\left(\Delta\tilde{\boldsymbol{R}}_{k-1,k}(\bar{\boldsymbol{b}}_g^{k-1})\exp\left(\frac{\partial\Delta\bar{\boldsymbol{R}}_{k-1,k}}{\partial\boldsymbol{b}_g}\delta\boldsymbol{b}_g\right)\right)^{\intercal}\right.\\ &\qquad \left.\boldsymbol{R}_{WS}^{k-1^{\intercal}}\boldsymbol{R}_{k-1,k}\right)\\ \boldsymbol{e}_{\Delta\boldsymbol{v}}^{k-1,k} &= \Delta\boldsymbol{R}_{WS}^{k-1^{\intercal}}\left({}_W\boldsymbol{v}^k - {}_W\boldsymbol{v}^{k-1} - {}_W\boldsymbol{g}\Delta t_{k-1,k}\right) \qquad (7)\\ &\quad - \left(\Delta\tilde{\boldsymbol{v}}_{k-1,k}(\bar{\boldsymbol{b}}) + \frac{\partial\Delta\bar{\boldsymbol{v}}_{k-1,k}}{\partial\boldsymbol{b}_a}\delta\boldsymbol{b}_a + \frac{\partial\Delta\bar{\boldsymbol{v}}_{k-1,k}}{\partial\boldsymbol{b}_g}\right)\\ \boldsymbol{e}_{\Delta\boldsymbol{t}}^{k-1,k} &= \boldsymbol{R}_{WS}^{k-1,k^{\intercal}}\left(\Delta\boldsymbol{t}_{k-1,k} - {}_W\boldsymbol{v}^{k-1}\Delta t_{k-1,k} - \frac{1}{2}\boldsymbol{g}\Delta t_{k-1,k}^2\right)\\ &\quad - \left(\Delta\tilde{\boldsymbol{t}}_{k-1,k}(\bar{\boldsymbol{b}}^{k-1}) + \frac{\partial\Delta\bar{\boldsymbol{t}}_{k-1,k}}{\partial\delta\boldsymbol{b}_a}\delta\boldsymbol{b}_a + \frac{\partial\Delta\bar{\boldsymbol{t}}_{k-1,k}}{\partial\delta\boldsymbol{b}_g}\delta\boldsymbol{b}_g\right) ,\end{aligned}$$

where values denoted by $\bar{\cdot}$ are obtained with the bias estimate $\bar{\boldsymbol{b}}$ at the time of the pre-integration and $\tilde{\cdot}$ denotes values using the current estimate of the state variables. The scalar value $\Delta t_{k-1,k}$ represents the time interval between the two KFs. For a detailed explanation of the used pre-integration method, we kindly refer the reader to [5]. Using the individual parts in Eq. (7), we define the residual vector for the IMU pre-integration as

$$\boldsymbol{e}_a^{k-1,k} = \left[\boldsymbol{e}_{\Delta\boldsymbol{R}}^{k-1,k^{\intercal}}, \boldsymbol{e}_{\Delta\boldsymbol{v}}^{k-1,k^{\intercal}}, \boldsymbol{e}_{\Delta\boldsymbol{t}}^{k-1,k^{\intercal}}\right]^{\intercal} . \qquad (8)$$

3) Prior Residual: In order to use prior knowledge $\bar{\theta}^k$ about a variable θ at time instance t_k, we define the prior residual as

$$\boldsymbol{e}_\theta^k := \bar{\theta}^k - \theta^k . \qquad (9)$$

Here, for non-Euclidean variables, such as rotations or bearing vectors, the minus operation needs to be adapted to the widely used box-minus operator.

C. Visual-Inertial Odometry

1) Frame Tracking: For every incoming Frame F, we extract a set of ORB features [23] and perform integration of the IMU readings queued since the last frame. Using the integrated IMU measurements, we perform a prediction of the pose for the current frame to carry out a guided correspondence search by projecting the agent's MPs in the current frame and matching the associated descriptors. Using the established correspondences, we perform an alignment step by minimizing the reprojection error against the observed MPs followed by a second correspondence search. In order to obtain a smooth trajectory and an accurate velocity estimate for the current frame, we execute a motion-only BA in which we optimize the KF poses and the IMU states of the most recent n KFs together with the current frame as shown in Fig. 3. After the optimization, we check whether to insert a new KF in the agent's map. A new KF is inserted when one of the three following conditions is met:

(a) more than 20 frames have passed since the last KF was created,
(b) the area where 2D keypoints are found covers less than 40% of the image, or
(c) less than 15 MPs are observed by the current frame.

Criterion (a) enforces temporal constraints between successive KFs, in order to avoid weak IMU constraints due to the accumulated uncertainty from the integration. Criteria (b) and (c) ensure sufficient overlap between the KFs, while adaptively inserting more KFs during challenging motions such as fast rotations.

2) Local Map: The *local map* holds the KFs and MPs in the surroundings of the current position. It is bounded to the N closest KFs in the vicinity of the agent, with N primarily depending on the available computational onboard resources of the agent. The KFs in the local map are both connected by the IMU constraints between consecutive KFs as well as covisibility constraints derived from common observations of MPs. In CVI-SLAM, two KFs are considered covisible if they share at least 15 MPs. As indicated in Fig. 3, we keep a constant number M, smaller than N, of consecutive KFs in a *local window* inside the *local map*, in order to ensure well defined IMU constraints. Note that while the temporal KFs must come from the same agent as the local map, the covisible KFs are purely defined by their covisibility, regardless of which agent created the KF. This allows the local odometry to benefit from experiences of other agents and improve the accuracy of the estimation in collaboration.

3) Local Mapping: The *local mapping* of our VIO runs in a separate thread and is responsible for maintaining and optimizing the *local map* and is triggered every time the *tracking* inserts a new KF. For map maintenance, we employ a scheme inspired by [1], in which MPs with insufficient observations in the tracking are culled. After culling we triangulate new MPs between the local KFs and merge them with existing MPs considering their vicinity and their associated ORB descriptors. In order to improve the accuracy and consistency of the *local map*, we perform a *Local Bundle Adjustment* (LBA) step. The scope of the LBA is defined by the most recent M KFs in the *local window*, and all MPs observed in those KFs. Additionally we insert all KFs that have common observations with the KFs in the *local window*, as illustrated in Fig. 3. These KFs outside the *local window* are inserted with their pose fixed and serve as anchors to stabilize the optimization. Therefore, we can formulate the objective of the LBA as follows:

$$\begin{aligned}J(\boldsymbol{\Theta}) &:= \sum_{k\in\mathcal{V}}\sum_{j\in\mathcal{L}(k)}\delta\left(\|\boldsymbol{e}_r^{k,j}\|^2_{\boldsymbol{W}_r^{k,j}}\right) \qquad (10)\\ &+ \sum_{k-1,k\in\mathcal{V}\setminus\mathcal{V}_s}\left(\|\boldsymbol{e}_a^{k-1,k}\|^2_{\boldsymbol{W}_a^{k-1,k}} + \|\boldsymbol{e}_b^{k-1,k}\|^2_{\boldsymbol{W}_b^{k-1,k}}\right) ,\end{aligned}$$

where $\delta(\cdot)$ is a robust cost function, here the Cauchy loss, aiming to reduce the influence of outliers, $\mathcal{L}(k)$ are the MPs observed by KF k, and $\boldsymbol{e}_b^{k-1,k}$ penalizes changes in the IMU bias for successive KFs. The set $\mathcal{V}_s$, denoting all KFs inserted with a fixed pose, can include KFs created by other agents.

4) IMU Initialization: Since the IMU biases, as well as the gravity direction are unknown, a dedicated initialization is necessary in order to perform the tracking as described above. In this work, we employ the strategy proposed in [6], which performs the initialization in three steps. In the initial phase, the vision-only tracking as in [2] is employed, while performing the IMU pre-integration with all bias variables set to zero. To avoid large integration windows between the KFs, we restrict the number of frames between KFs to 3. After

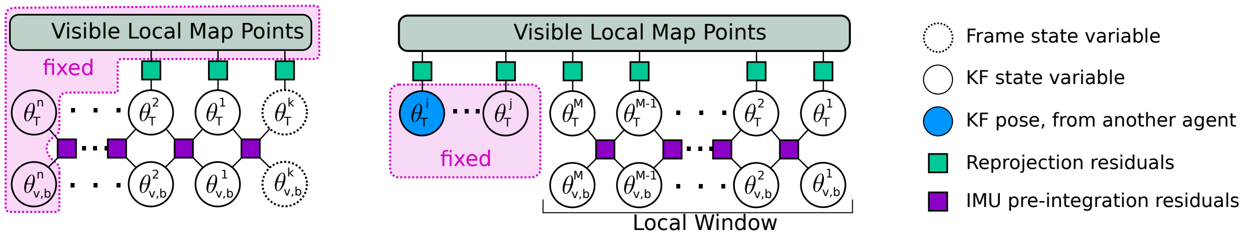

Fig. 3: Schematics depicting the variables involved in the local optimization during frame tracking (left) and the *Local Bundle Adjustment* (LBA) with a *local window* of M KFs running onboard the agent (right). While the tracking considers only the most recent n KFs stemming from the same agent, the LBA can also include KFs from other agents.

reaching 15 KFs, the visual structure is bundle adjusted and the gyroscope bias is initialized in a linear least squares fashion. Afterwards, the scale of the visual structure, the velocities in S and gravity direction are linearly estimated followed by a refinement of the gravity direction. Upon successful computation of all steps, the visual structure is scaled and aligned with the gravity direction, while the body velocities are rotated in the world frame W. If the initialization fails up to a window of 20 KFs, we re-initialize the visual-only tracking and start again.

D. Communication

The *communication modules* on the agents' and the server's sides act as the interface between both sides that serializes and de-serializes data that is shared between them. Using this interface, each agent informs the server about changes in its *local map*, i.e. any added or changed KFs and MPs. Constantly sending the whole data structure for KFs and MPs (including e.g. 2D feature keypoints extracted from the image) would result in high network traffic, with messages of around 300 Bytes for MPs and 29 KB for KFs (for our setup with 500 ORB features per KF). Therefore, after sending a KF or a MP once, for all following changes an update message is sent, having only a size of 184 Bytes for KFs and 52 Bytes for MPs. Messages from the server to the agent contain the K KFs with the strongest covisibility to the current position of the agent, to augment the agent's *local map* with past data and KFs and MPs from other agents. Using these messages, we also transmit IMU related information (IMU measurements and current bias and velocity estimates) from the agents to the server. From the KF messages received from the agent, we reconstruct the initial pre-integration factors created by VIO for incorporation in global map optimiation (Sec. IV-F). From the agent to the server, we do not include the pre-integration factors, since IMU data is not considered for KFs outside the *local window*. However, we include the most current velocity and bias estimates for KFs to update this data on the agent with refined results from BA. In case of a loss of connection to the server, an agent will not be able to exchange data with the server any more, and fall back to a VIO system with limited *local map* size N.

E. Redundancy Detection and Removal

When one or more agents visit the same location multiple times during a mission, the map contains several KFs from the same distinct place. Some of these KFs encode almost the same information, if the measurement was taken from a similar viewpoint. Since the size of the map affects the timings of most processes on the server, such as place recognition, map queries or BA, it is desirable to remove these redundancies to boost the scalability of the system. Our rejection scheme randomly selects a KF from the map, and compares it to its neighbors in the covisibility graph. If the number of commonly observed landmarks with all neighbors is higher than a pre-defined threshold, the KF is considered redundant. Furthermore, for consecutive KFs connected by IMU constraints, we allow a maximum time of 2s between two KFs, to bound inaccuracies from IMU integration for optimization.

F. Loop & Map Fusion

To detect repeatedly visited locations inside one *server map* (*loop closure*) and separated maps (*map matching*) on our *server map stack*, we employ a multi-stage place recognition system. Firstly, for a query Keyframe KF$_q$, we select a subset of possibly matching candidates $\mathcal{C}$ from all map data using a bag-of-words approach [24], followed by a brute force descriptor matching of KF$_q$ against all KFs in $\mathcal{C}$. Upon successful matching of KF$_c$, we employ a projective RANSAC scheme to compute the initial transformation $\boldsymbol{T}_{cq}$, which is then refined by minimizing the reprojection error. Using the optimized $\boldsymbol{T}_{cq}$, we search for additional matches in the vicinity of KF$_q$ and KF$_c$ by projecting associated MPs from KF$_q$ into KF$_c$ and vice versa. In the case of a *loop closure*, we first perform a pose-graph optimization using the scheme of [2], followed by transforming the MPs using the optimized poses. For the *map matching*, we transform the map of the query KF into the map of the candidate KF using $\boldsymbol{T}_{cq}$. Finally we perform global BA (GBA) using the following objective function:

$$J(\boldsymbol{\Theta}) := \|e_p^c\|^2_{\boldsymbol{W}_p^c} + \sum_{k\in\mathcal{V}} \sum_{j\in\mathcal{L}(k)} \delta\left(\|\boldsymbol{e}_r^{k,j}\|^2_{\boldsymbol{W}_r^{k,j}}\right) + \sum_{k-1,k\in\mathcal{V}} \left(\|\boldsymbol{e}_a^{k-1,k}\|^2_{\boldsymbol{W}_a^{k-1,k}} + \|\boldsymbol{e}_b^{k-1,k}\|^2_{\boldsymbol{W}_b^{k-1,k}}\right), \tag{11}$$

where the first term is a prior on the pose of KF$_c$ in order to remove the gauge degree of freedom. The optimization is performed in two steps; at first, we only optimize the MP positions and the IMU states, while fixing the KF poses, followed by a full optimization over all states involved in Eq. (11). Note that the IMU constraints in Eq. (11) are only inserted between consecutive KFs created by the same agent.

V. EXPERIMENTAL RESULTS

We evaluate CVI-SLAM in a variety of different scenarios to test its performance, applicability, scalability and accuracy. We analyze the network traffic between agents and the server to reveal the bandwidth requirements of CVI-SLAM, as well as the timings of the modules onboard an agent, in single-agent and multi-agent scenarios. Furthermore, we show the importance of the redundancy detection on the server side, and evaluate the accuracy of the system on the five Machine Hall (MH) sequences (MH1 – MH5) of the EuRoC dataset [25], where a small UAV flies several trajectories through an industrial environment, and compare the results to other state-of-the-art SLAM and VIO systems, namely the visual-inertial version of ORB-SLAM [4] and VINS-mono [6], which both also use an IMU pre-integration scheme. For all experiments, we use the following setup:

- Server: Lenovo Legion Y920 notebook (Core i7-7820HK @ 2.90GHz $\times$ 8, 32 GB RAM)
- Agents 1 – 4: Intel NUC 5i7RYH (3.1 GHz $\times$ 4, 8 GB RAM), used e.g. on the AscTec Neo UAV

Throughout our experiments, we use a *local window* size of M = 10 KFs, *local map* size of N = 20 KFs, and K = 10 KFs for messages from the server to an agent. For the analysis on the pre-recorded datasets, the agents and the server are connected via a wireless network, so that real communication between the server and the agents takes place. Then the datasets are processed onboard the agents. This makes our evaluation across different runs more comparable and provides us with ground truth, while still using real network communication as we would during a robotic mission. All values in this section are averaged over 3 runs for each experiment if not stated otherwise.

A. Accuracy

In order to obtain a baseline for the accuracy of CVI-SLAM, we first evaluate the absolute error as well as the scale error for the KF trajectory obtained from the server, while running a single agent and compare against state-of-the-art methods. In Table I, we compare the performance to both VI-ORB-SLAM and VINS-mono. With the exception of the sequence MH4, we perform comparably or slightly better than the state-of-the-art methods, indicating the effectiveness of CVI-SLAM's server-agent architecture. We attain the higher error on the MH4 sequence to the fact that CVI-SLAM is able to close the loop approximately half-way through the trajectory, but does not do so at the end and therefore, does not propagate the correction over the whole trajectory. For the evaluation of the map merging capability of our system, we run experiments both in a two-agent as well as in a four-agent setting, while for each agent we use a different sequence of the dataset. As VI-ORB-SLAM is closed source, we are unable to compare against in this setup. The comparison with the open-sourced VINS-mono is performed using its multi-session capability by sequentially running the different sequences. The values in Table II report the error of the joint trajectory of all agents at the end of all runs. In all sequences, we consistently perform better than VINS-mono. This can be explained by the fact that we can correct both the KF states as well as the map structure during loop-closures rendering the optimization much more powerful than the pose-graph optimization employed by VINS-mono. Furthermore, as we maintain a covisibility graph and are able to re-use parts of the map, the constraints upon loop-closures are generally stronger. Compared to the single-agent scenario, we get more consistent errors in a similar range, indicating the power of sharing and re-using information between agents in the collaborative setting.

TABLE I: Single-agent Root Mean Squared Error (RMSE) and scale error over the global KF trajectory. The lowest error is indicated in bold.

Dataset	VI-ORB [4]		VINS [6]		CVI-SLAM	
	RMSE	Scale	RMSE	Scale	RMSE	Scale
MH1	**0.075 m**	0.5%	0.177 m	0.359%	0.085 m	1.81%
MH2	0.084 m	0.8%	0.13 m	1.117%	**0.063 m**	1.055%
MH3	0.087 m	1.5%	0.1 m	0.318%	**0.065 m**	0.336%
MH4	0.217 m	3.4%	**0.155 m**	0.947%	0.293 m	3.178%
MH5	0.082 m	0.5%	0.136 m	0.314%	**0.081 m**	0.299%

TABLE II: Multi-agent joint KF-trajectory evaluation with 2 and 4 agents. The lowest error is indicated in bold.

Datasets	VINS [6]		CVI-SLAM	
	RMSE	Scale	RMSE	Scale
MH1 & MH2	0.158 m	0.150%	**0.050 m**	0.673%
MH2 & MH3	0.197 m	0.408%	**0.073 m**	0.538%
MH4 & MH5	0.198 m	0.575%	**0.115 m**	0.756%
MH1;2;3;5	0.244 m	1.33%	**0.156 m**	0.137%

The last experiment aims at the investigation of the collaborative setting on the tracking that runs onboard an agent. For this purpose, we run the experiments both in a single-agent as well as in a two-agent setting. The error values reported in Table III are computed by aligning the global trajectory obtained by the tracking with ground truth. For each experiment we align and evaluate only the results of the second sequence in the multi-agent case, while for the single-agent case we only run the second sequence. As it can be seen, the tracking accuracy is consistently improved in the multi-agent setting, even for the sequences MH2 & MH3, which have only little overlap. This highlights the clear benefit of sharing information obtained from multiple agents amongst them in order to improve the accuracy in real-time during collaboration.

TABLE III: Tracking error on the agent.

Datasets	CVI-SLAM Single		CVI-SLAM Multi	
	RMSE	Scale	RMSE	Scale
MH1 & MH2	0.224 m	3.693%	**0.139 m**	1.002%
MH2 & MH3	0.295 m	1.444 %	**0.256 m**	0.856%
MH4 & MH5	0.412 m	3.341%	**0.34 m**	1.208%

B. Scalability

The efficient architecture of CVI-SLAM boosts the scalability of the system in terms of agents. Fig. 4 shows the timings of the onboard processes on the agent (*frame tracking*, *local mapping* of the VIO, *communication*) with the number of participating agents. The number of participants in CVI-SLAM does not influence the timings onboard each agent. With an average tracking time of around 36 ms per frame, the onboard modules can run in real-time with the 20 Hz camera image stream provided by the sensor used for the EuRoC dataset.

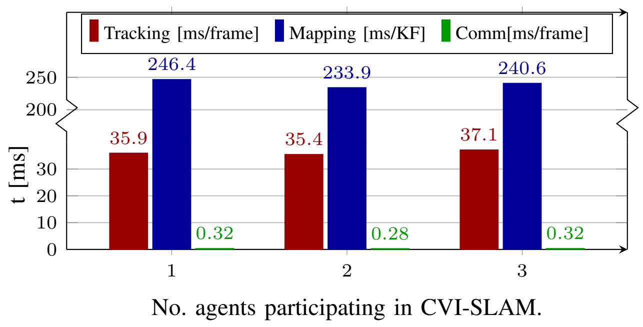

Fig. 4: Timings of the agent's onboard modules with a growing number of participating agents, measured on Agent 1. All values are averaged over 5 runs using the EuRoC dataset sequences MH1, MH2 and MH3.

Also for the network traffic shown in Fig. 5, no increase from more participating agents can be observed, implying a linear scalability of the network traffic with the number of agents. The reduced network traffic from server to agent originates from the situation that in a multi-agent scenario, more loop closures occur compared to the single-agent case, resulting in a increased number of optimization steps and therefore, more time periods where a server map is locked and no data is be transmitted to the agents.

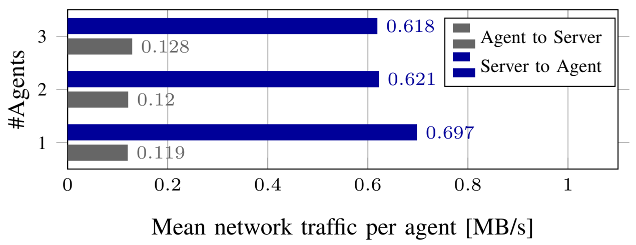

Fig. 5: Network traffic between agent and server with a growing number of participating agents. All values are averaged over 5 runs for each experiment using the EuRoC dataset sequences MH1, MH2 and MH3.

On the server's side, more participating agents result in more data to be managed. While place recognition using an inverted file index scales linearly with the number of KFs, the complexity of BA is cubic in the number of KFs and MPs incorporated in the optimization step. Therefore, the detection and removal of redundant information is essential for the scalability of a collaborative SLAM system with the number of agents. Fig. 6 shows the resulting number of KFs in the *server map stack* with and without redundancy detection throughout the experiment, evaluated in a 4-agent scenario. The tendency clearly shows a smaller increase in the number of KFs in the system over time, boosting the scalability of CVI-SLAM. The redundancy detection could reduce the number of KFs by 20% from 1460 to 1170, achieving similar accuracy of the resulting estimate.

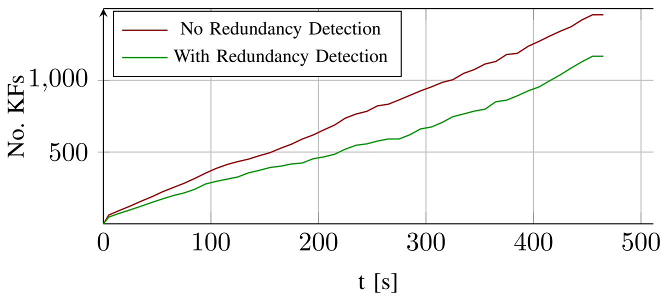

Fig. 6: Number of KFs in the *server map stack* over runtime in a 4 agent scenario, with and without redundancy detection and removal, using the EuRoC sequences MH1, MH2, MH3 and MH5.

C. Network Traffic

Three sequences of the EuRoC dataset are used to analyze the network traffic between the agents and the server. While an agent sends new KFs and MPs to the server when they enter the map, the server sends with every message the $K = 10$ closest KFs to the current position of the agent. The agent can send up to 5 messages per second to the server, while for the server this number is limited to 2, since these messages are usually larger than the ones from the agent. Table IV shows the average traffic between server and agent. On the server, the average traffic is around 0.7 MB/s, which is composed of 2×10 KF messages per second of approximately 30 KB, plus a varying number of observed MPs. By reducing the publishing frequency or the number K of KFs per message, the network traffic can be reduced, though also reducing the information shared with the agent. For the agent, the main influencing factor on the traffic, besides the message limit per second, is the number of KFs inserted in the map. Since the datasets MH3 and MH5 are more challenging than MH1, more KFs are inserted here, resulting in a slightly higher network traffic from the agents to the server for these datasets.

TABLE IV: Mean network traffic between an agent and the server on different trajectories of the EuRoC dataset

Dataset	**Agent $\rightarrow$ Server [MB/s]**	**Server $\rightarrow$ Agent [MB/s]**
EuRoC MH1	0.119	0.697
EuRoC MH3	0.135	0.707
EuRoC MH5	0.151	0.720

Fig. 7 shows the network traffic between the server and the agent over runtime. The communication from agent to server starts with successful IMU initialization, therefore the first message contains the initial map with the size of this depending on the size of the IMU initialization window (15 KFs in our case), causing the spike in the network traffic at the beginning of each experiment.

From server to agent (S$\rightarrow$A), the traffic first fluctuates around the mean traffic of approximately 0.7 MB/s, but shows several drops later in the experiment. When the agent returns to previously visited locations, the global BA triggered by the loop closures temporarily suspends the transmission of data to the agents, causing these drops.

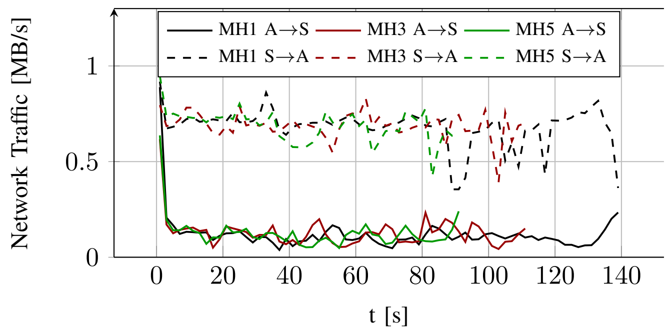

Fig. 7: Network traffic from agent to server (A→S) and vice-versa (S→A) on several sequences of the EuRoC dataset. Sending the initial map after IMU initialization causes a higher traffic on the agent side at the beginning. Global optimization locking the map for data exchange, resulting from loop closure, causes the drops towards the end of the trajectories on the server side.

VI. CONCLUSION

In this paper, we present CVI-SLAM, an accurate and powerful system for keyframe-based collaborative SLAM. Participating agents are equipped with a visual-inertial sensor suite and constraint onboard calculation power, sharing all information throughout the mission with a more powerful central server. The server merges information from the participating agents and distributes it throughout the system, such that agents can profit from measurements contributed by collaborating agents. Our experiments demonstrate that CVI-SLAM's accuracy is comparable to state-of-the-art visual-inertial SLAM systems, outperforming them in collaborative scenarios. Furthermore, our evaluation confirms that sharing information amongst participating agents during collaborative SLAM estimation improves the accuracy of pose estimation onboard each agent in real-time compared to single-agent scenarios. A comparison of single- and multi-agent experiments and analysis of the network traffic attests to the scalability and applicability of CVI-SLAM. Compared to existing collaborative SLAM systems, CVI-SLAM combines efficient collaboration in mapping *and* localization, sharing all information amongst all participating agents, with accurate collaborative scene estimation and practical applicability of the system. To the best of our knowledge, CVI-SLAM is the first full visual-inertial collaborative SLAM system implementing two-way communication between agent and server, tested on real data. Future work will focus on further boosting the accuracy improvements achieved from collaborative scene estimation, and increasing the number of participating agents in the system.

REFERENCES

[1] R. Mur-Artal, J. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] P. Schmuck and M. Chli, "Multi-UAV Collaborative Monocular SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[3] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *International Journal of Robotics Research (IJRR)*, vol. 34, no. 3, pp. 314–334, 2015.

[4] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters (RAL)*, vol. 2, no. 2, pp. 796–803, 2017.

[5] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics (T-RO)*, vol. 33, no. 1, pp. 1–21, 2017.

[6] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *arXiv preprint arXiv:1708.03852*, 2017.

[7] M. W. Achtelik, S. Weiss, M. Chli, F. Dellaert, and R. Siegwart, "Collaborative stereo," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[8] N. Piasco, J. Marzat, and M. Sanfourche, "Collaborative localization and formation flying using distributed stereo-vision," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[9] J. Dong, E. Nelson, V. Indelman, N. Michael, and F. Dellaert, "Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

[10] T. A. Vidal-Calleja, C. Berger, J. Solà, and S. Lacroix, "Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain," *Robotics and Autonomous Systems (RAS)*, vol. 59, no. 9, pp. 654–674, 2011.

[11] C. X. Guo, K. Sartipi, R. C. DuToit, G. Georgiou, R. Li, J. OLeary, E. D. Nerurkar, J. A. Hesch, and S. I. Roumeliotis, "Large-scale cooperative 3D visual-inertial mapping in a Manhattan world," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[12] A. Caccavale and M. Schwager, "A distributed algorithm for mapping the graphical structure of complex environments with a swarm of robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[13] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[14] T. Cieslewski and D. Scaramuzza, "Efficient decentralized visual place recognition using a distributed inverted index," *IEEE Robotics and Automation Letters (RAL)*, vol. 2, no. 2, pp. 640–647, 2017.

[15] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *International Journal of Robotics Research (IJRR)*, vol. 36, no. 12, pp. 1286–1311, 2017.

[16] D. Zou and P. Tan, "CoSLAM: Collaborative visual SLAM in dynamic environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 2, pp. 354–366, 2013.

[17] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular SLAM with multiple micro aerial vehicles," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[18] L. Riazuelo, J. Civera, and J. Montiel, "C2TAM: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems (RAS)*, vol. 62, no. 4, pp. 401–413, 2014.

[19] J. G. Morrison, D. Gálvez-López, and G. Sibley, "MOARSLAM: Multiple Operator Augmented RSLAM," in *Distributed Autonomous Robotic Systems*, 2016, vol. 112, pp. 119–132.

[20] I. Deutsch, M. Liu, and R. Siegwart, "A Framework for Multi-Robot Pose Graph SLAM," in *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2016.

[21] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, "Cloud-based collaborative 3D mapping in real-time with low-cost robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 423–431, 2015.

[22] J.-L. Blanco, J. González-Jiménez, and J.-A. Fernández-Madrigal, "Sparser relative bundle adjustment (SRBA): constant-time maintenance and local optimization of arbitrarily large maps," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[24] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics (T-RO)*, vol. 28, no. 5, pp. 1188–1197, 2012.

[25] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *International Journal of Robotics Research (IJRR)*, vol. 35, no. 10, pp. 1157–1163, 2016.