Loop-Box: Multiagent Direct SLAM Triggered by Single Loop Closure for Large-Scale Mapping

M. Usman Maqbool Bhutta, *Member, IEEE*, Manohar Kuse, *Member, IEEE*, Rui Fan, *Member, IEEE*, Yanan Liu, *Graduate Student Member, IEEE*, and Ming Liu, *Senior Member, IEEE*

***Abstract*—In this article, we present a multiagent framework for real-time large-scale 3-D reconstruction applications. In SLAM, researchers usually build and update a 3-D map after applying nonlinear pose graph optimization techniques. Moreover, many multiagent systems are prevalently using odometry information from additional sensors. These methods generally involve extensive computer vision algorithms and are tightly coupled with various sensors. We develop a generic method for the key challenging scenarios in multiagent 3-D mapping based on different camera systems. The proposed framework performs actively in terms of localizing each agent after the first loop closure between them. It is shown that the proposed system only uses monocular cameras to yield real-time multiagent large-scale localization and 3-D global mapping. Based on the initial matching, our system can calculate the optimal scale difference between multiple 3-D maps and then estimate an accurate relative pose transformation for large-scale global mapping.**

***Index Terms*—Direct simultaneous localization and mapping (SLAM), large-scale 3-D mapping, loop closure, multiagent SLAM.**

I. Introduction

IN MODERN times, many researchers have turned their focus toward developing multiagent simultaneous localization and mapping (SLAM) systems that can merge multiple maps built by each agent connected with the centralized system [1], [2]. However, the usage of these systems is usually computationally intensive, and it is hard to achieve real-time performance in large-scale applications. Maplab [3] has provided a multisession solution for such SLAM problems. This is useful when the agents lose their relative pose information due to an abrupt change in light or when someone passes by the camera system. Maplab can easily build a connection between the previous odometry and the new odometry.

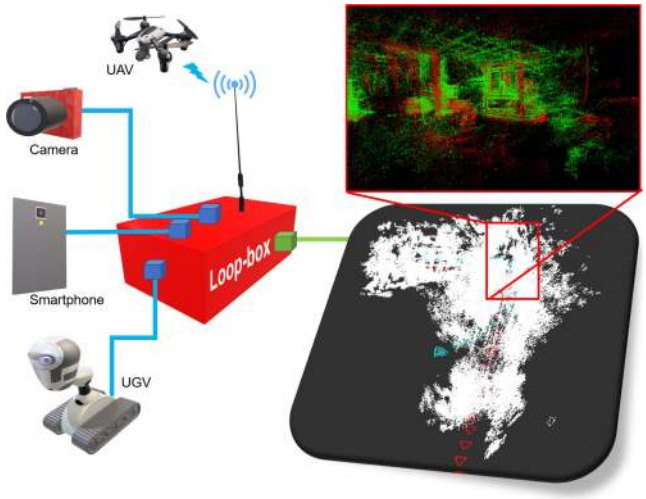

Fig. 1. Loop-box assists different cameras integrated into a system for multiagent SLAM. The poses of the source and target agents are shown in red and cyan, respectively. Their 3-D maps fused using Loop-box are shown in red and green, respectively.

The lack of a state-of-the-art multiagent SLAM system that can easily connect various robotic platforms also emphasizes the importance of developing a ready-to-use framework for multiple agents [4]. Visual-inertial-sensor-based SLAM systems [5]–[8] perform excellently for a single agent, but for large-scale 3-D collaborative mapping, a complicated calibration setup along with intensive computing power is essential. In addition to this, the most ambitious problem is to estimate an accurate relative pose transformation between different agents after observing the first loop closure. The system should be sufficiently robust to fuse multiple 3-D maps, regardless of the scale variation between them. An example of a multiagent system is shown in Fig. 1. It presents the central system that processes the 3-D maps generated by different agents. Furthermore, the agents can face several key challenging states while moving around. These states are shown in Fig. 2.

In these studies, agents are constrained to the loop closures where they are following the same path, as shown in Fig. 2(a). Fig. 2(b)–(d) shows real-world scenarios that agents might face to achieve loop closure during a short interval, for example, when agents are following the same direction and

Manuscript received April 6, 2019; revised June 4, 2020 and August 18, 2020; accepted September 20, 2020. Date of publication November 6, 2020; date of current version June 16, 2022. This work was supported in part by the National Natural Science Foundation of China, under Grant U1713211, and in part by Collaborative Research Fund by Research Grants Council Hong Kong, under Project C4063-18G. The work of Ming Liu was supported in part by HKUST-SJTU Joint Research Collaboration Fund, under Project SJTU20EG03. This article was recommended by Associate Editor S. X. X. Yang. *(Corresponding author: M. Usman Maqbool Bhutta.)*

M. Usman Maqbool Bhutta and Ming Liu are with the Robotics and Multi-Perception Laboratory in Robotics Institute, Hong Kong University of Science and Technology, Hong Kong (e-mail: usmanmaqbool@outlook.com).

Manohar Kuse is with Robotics Institute, Hong Kong University of Science and Technology, Hong Kong.

Rui Fan is with the Jacobs School of Engineering and the Jacobs School of Medicine, University of California at San Diego, San Diego, CA 92093 USA, and also with the MVI-Lab, ATG Robotics, Hangzhou 310000, China.

Yanan Liu is with the Bristol Robotics Laboratory, University of Bristol, Bristol BS16 1QY, U.K.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2020.3027307.

Digital Object Identifier 10.1109/TCYB.2020.3027307

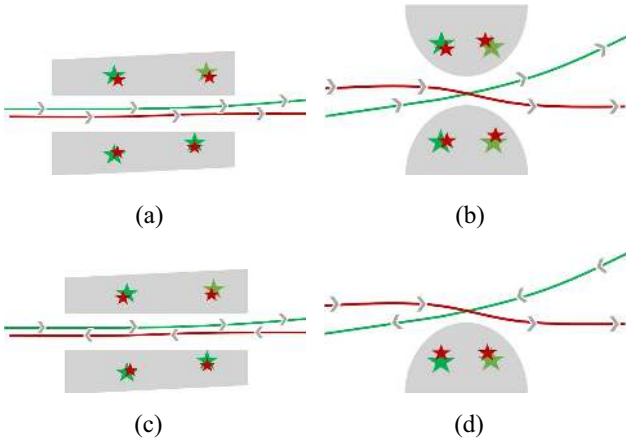

Fig. 2. Key challenging states a multiagent SLAM system may encounter. Red and green lines represent the paths of each of two agents. The landmarks are illustrated with red and green stars. As seen from different agents, a scale difference may occur. (a) Two agents move in the same direction, and they have several loop closures. (b) Two agents move in the same direction, and they have only one loop closure (c) Two agents move in opposite directions, and they have several loop closures. (d) Two agents move in opposite directions, and they have one-sided loop closure.

detect a loop closure for a short time. Another intriguing case is where agents are coming from opposite directions, and the system detects a short-duration loop closure. Hence, a robust multiagent framework is needed to determine the optimum relative pose difference of such agents.

A. Contributions

We develop an intelligent system that can help establish a relative connection between the agents at first interaction. Our proposed framework is capable of working at the first loop closure under the key challenging scenarios mentioned earlier, in Fig. 2. To show the robustness of our system, we use the large-scale direct (LSD) SLAM [9] system on each agent. In addition, this SLAM system does not have any scale information or feature point tracking on each end. Fig. 1 presents the final 3-D point cloud, which is shown in an enlarged boxed area with a contrasting color scheme. Red shows the transformed source point cloud map, and green is the target point cloud map. The transformed trajectories of both agents inside the fused semidense point cloud maps using the proposed method are also shown in Fig. 1. The results demonstrate the efficiency of our presented system after the first loop closure; hence, its name is *Loop-box*. The Loop-box system can work collaboratively to make a large-scale 3-D-map based on semidense or sparse SLAM approaches. Our contributions include the following:

1) a multiagent direct SLAM system that can merge 3-D maps with different scales to a global representation;
2) a robust system that can accurately determine the relative change between multiple robots during a very short interval [Fig. 2(b) and (d)]; in particular, no previous study, to our knowledge, has considered short intervals;
3) a multiagent system that collaborates pairwise directly; therefore, each agent can estimate the relative pose transformation and registering of the 3-D maps after the first loop closure;
4) a signal-loop-closure-based method that welcomes features to direct SLAM methods for large-scale 3-D collaborative mapping. This allows the proposed framework to work effectively in any of the states shown in Fig. 2.

We believe that our multiagent SLAM is one of the first direct SLAM-based multiagent schemes that unites a wide variety of use cases within a single system. We firmly believe that computer vision researchers will use it for intelligent mapping and localization of agents for robotics applications.

B. Paper Organization

The remainder of this article is organized as follows. Section II reviews modern multiagent systems. In Section III, brief introductory material related to this work is included. In Section IV, we provide the details of the proposed framework. In Section V, the experimental results are illustrated, and the performance of the system is evaluated in discussions in Section VI. Finally, Section VII concludes this article and provides recommendations for future work.

II. Related Work

In the world of robotics applications, the 3-D map has great importance, but a single agent is not sufficient to yield a large 3-D map. Therefore, many robotics scientists have turned their concentration toward developing multiagent SLAM systems capable of processing multiple maps. The first such reported work is found in [4], where an efficient large-scale mapping and navigation framework is used to incorporate numerous maps. For large-scale mapping, Deutsch *et al.* [10] proposed a standard solution of using a multirobot pose graph SLAM. In 2017, Schneider *et al.* [3] introduced an open-source visual-inertial mapping framework called Maplab. This framework provides a collection of tools, including visual-inertial batch optimization, loop closure detection, and map merging [3]. Maplab has proved to be one of the most useful open-source SLAM frameworks in recent years.

Schmuck and Chli [1], [2] presented multiagent visual-inertial-based SLAM systems that can provide high localization accuracy for a single trajectory, and these approaches can be used in session-based collaborative work. However, due to extensive computation by the centralized system, these systems are limited to a maximum of three agents while mapping the entire environment. Furthermore, Chen *et al.* [11] proposed a distributed multiagent SLAM system that can fulfill a variety of large-scale outdoor tasks without relying on the maps to determine the relative poses.

Most of the visual-based [10], [12] and visual-inertial-based [3], [8] frameworks can perform well only in the case shown in Fig. 2(a), where the two agents move in the same direction and they can observe several landmarks across the path. However, when the agents do not move in the same direction and their paths intersect once [see Fig. 2(c) and (d)], the existing multiagent SLAM systems often fail to merge the 3-D maps. Moreover, the visual-inertial-based frameworks usually require additional sensor calibrations to ensure their performance can give excellent results for the multiagent SLAM system.

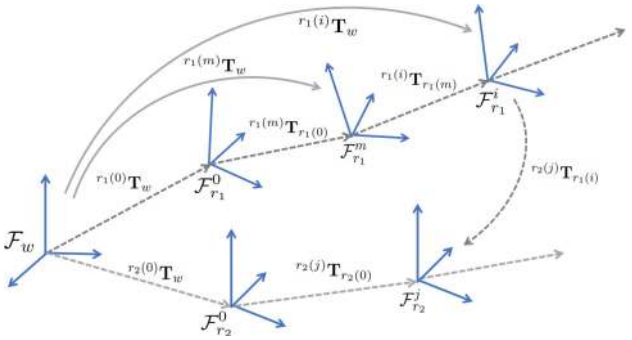

Fig. 3. Camera frame transformations relative to the world frame and other agents. The gray continuous-line arrows present the transformation from the world frame. The gray dashed-line arrows represent the relative edges' transformation.

III. Preliminaries

A. Notations

We denote the matrices as a bold face capital letter $\mathbf{R}$ and vectors by a bold face lowercase letter $\boldsymbol{t}$. At the ith instant, the robot r-associated keyframe and point clouds are expressed as $\mathbf{K}_r^i$ and $P_{r(i)}^{\mathcal{F}_r^i}$, respectively.

B. Transformations

We define camera frames by $\mathcal{F}_{r_{\text{ID}}} \in \text{SE}(3)$ with $\text{ID} \in \mathbb{Z}^+$. The operator ${}^b\mathbf{T}_a(.)$ shows the 3-D rigid body transformation from $\mathcal{F}_a$ to $\mathcal{F}_b$, where ${}^b\mathbf{T}_a \in \text{SE}(3)$. Moreover, ${}^a\mathbf{T}_b$ is the inverse transformation of ${}^b\mathbf{T}_a$. This matrix is further divided into rotation matrix $\mathbf{R}_{ab} \in \text{SO}(3)$ and translation vector $\mathbf{t}_{ab} \in \mathbb{R}^3$. ${}^{r(i)}\mathbf{T}_w$ shows the homogeneous transformation matrix representing the pose of robot r at the ith instant with respect to the world frame $\mathcal{F}_w$. The transformation notations for a multiagent system are explained in Fig. 3.

Furthermore

$$\mathcal{F}_b = \left({}^a\mathbf{T}_b\right)^{-1}.\mathcal{F}_a = {}^b\mathbf{T}_a.\mathcal{F}_a$$

where ${}^b\mathbf{T}_a$ transforms the pose of the robot from camera frame $\mathcal{F}_a$ to frame $\mathcal{F}_b$. Similarly, transformation ${}^y\mathbf{T}_w = {}^y\mathbf{T}_x^x\mathbf{T}_w$ is used to rotate and translate the transformation from ${}^x\mathbf{T}_w$ to ${}^y\mathbf{T}_w$. A 3-D relative transformation $\mathbf{T} \in \text{SE}(3)$ is further divided into scaling (σ), rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$, which is defined by

$$\mathbf{T} = \begin{pmatrix} \sigma\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}, \text{ where } \mathbf{R} \in \text{SO}(3), \mathbf{t} \in \mathbb{R}^3 \text{ and } \sigma \in \mathbb{R}^+. \tag{1}$$

C. Information Matrix for Pose Graph SLAM

For the estimation of uncertainty among edges in pose graph SLAM, the information matrix plays a key role. We calculate the information matrix by taking the inverse transform of a 6×6 covariance matrix in the 6-degrees-of-freedom (DOFs) system. We can rewrite the objective function J of the iterative closest point (ICP) as

$$J = \min \sum_{i=1}^{n} F^2 \tag{2}$$

where $F = \|G\|$, and $G = \mathbf{R}P_i + \mathbf{T} - Q_i$. For a Jacobian-based objective function, Censi [13] proposed an approximation of the covariance matrix as follows:

$$\text{cov}(\mathbf{x}) \approx \left(\frac{\partial^2 J}{\partial \mathbf{x}^2}\right)^{-1} \left(\frac{\partial^2 J}{\partial \mathbf{z} \partial \mathbf{x}}\right) \text{cov}(\mathbf{z}) \left(\frac{\partial^2 J}{\partial \mathbf{z} \partial \mathbf{x}}\right)^T \left(\frac{\partial^2 J}{\partial \mathbf{x}^2}\right)^{-1} \tag{3}$$

where $\mathbf{x} = [x \quad y \quad z \quad a \quad b \quad c]$ and $\mathbf{z}$ are the n sets of correspondances $\{P_i, Q_i\}$. In our previous work [14], we presented an efficient approach, point cloud registration (PCR)-Pro, for aligning and registering multiple 3-D point clouds with different scales. It uses a 3-D-closed-form solution method [15] and OpenGV [16] for the relative transformation and information matrix computation.

IV. Loop-Box Framework

The Loop-box framework, shown in Fig. 4, can work for both real-time or offline scenarios. It consists of two major components.

1) For *distributed multiagent SLAM*, if agents are capable of exchanging keyframes and detecting the loop closure, Loop-box will help in their relative scale difference computation and the estimation of the relative pose transformation.
2) For *centralized multiagent SLAM*, Loop-box processes a few connected keyframes along with their poses and point clouds to merge them.

Loop-box is built on the robot operating system (ROS) [17], and packages are created using the catkin workspace. An ROS-based central system is developed to process the SLAM information of multiple agents. This can be used as an ROS package as well as an extension library to be integrated with any multiagent SLAM-related applications. The C++11 standard is used along with third-party libraries, such as Eigen [18], OpenCV [19], OpenGV [16], and Ceres [20] for programming the entire system. For the detailed visualization and error estimation, we use rviz, ParaView, and EVO [21].

A. Agents Overview

We name agents as *slaves* and the server as the *master*. All the slave robots are equipped with a monocular camera, and a SLAM operation is executed on them individually. In our case, we use the LSD-SLAM [9] system on each agent. LSD-SLAM assists in providing keyframe, point cloud, and pose information. ROS is used as a core part of the Loop-box method to connect the master with the slaves.

B. Loop Closure Detection

The majority of prior research has applied vocabulary building for place recognition [22], [23]. Sometimes the use of CNNs [24] has also given promising results. We use FAB-MAP [25] for place recognition. For instance, our system detects loop closure LC_{ij} among agents at an instant when the source slave is at position i, and the target slave is at j. Moreover, $P_{s(i)}^{\mathcal{F}_s^i}$ and $P_{t(j)}^{\mathcal{F}_t^j}$ will be observed along with the matched keyframes $\mathbf{K}_s^i$ and $\mathbf{K}_t^j$, respectively. After detection of the loop closure, the next and previous keyframes are matched

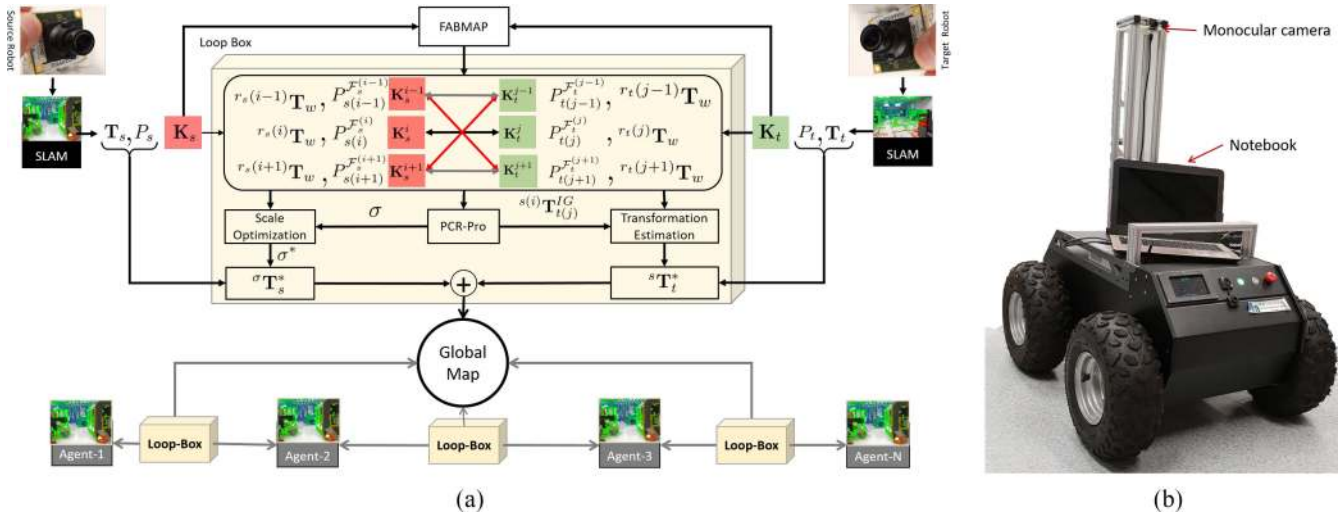

Fig. 4. (a) System overview of the Loop-box framework. Each agent uses a monocular camera and runs LSD-SLAM over it. The keyframes ($\mathbf{K}_s$ and $\mathbf{K}_t$) are processed for the loop closure detection using FABMAP. The camera frames ($\mathcal{F}_s$, $\mathcal{F}_t$) and point clouds (P_s, P_t) are transformed using the final scale and relative transformation for the global mapping. (b) UGV used for the collection of the datasets.

to determine the direction of both agents. If both the agents are moving in the same direction, then the previous and next matches are taken, as shown in gray in Fig. 4(a). If the opposite direction is detected, then the alternative matches are taken for further computation, as shown in red in Fig. 4(a).

C. Scale Information

After the loop closure detection, the first step is to find out the scale difference. Monocular-camera-based SLAM and direct SLAM approaches lack the provision of scale information. For the best relative transformation between both agents, scale difference σ is an essential factor. Usually, it is detected by estimating the volumes of both point clouds. Our previously reported PCR-Pro [14] provides a robust method to deal with such problems by calculating the scale difference and relative transformation between them efficiently.

1) Relative Transformation of Matches at "LC_{ij}": PCR-Pro estimates the scale difference σ and then aligns both point clouds. Moreover, it helps in estimating the good feature points among keyframes. The matched SIFT features are utilized to measure the relative transformation between each matching keyframe. In PCR-Pro, we use OpenGV [16] with the eight-point algorithm method to find out the relative camera pose ${}^{s(i)}\mathbf{T}^{\mathrm{RC}}_{t(j)}$ and initial guess transformation ${}^{s(i)}\mathbf{T}^{\mathrm{IG}}_{t(j)}$.

2) Scale Computation Using Kalman Filter: After evaluating the relative transformation using the eight-point algorithm, the matched points of the source agent keyframe are transformed to a 3-D world frame of reference of target agent $\mathcal{F}_t$. We use the Kalman filter to map the 3-D key points to the target agent key points. Based on the alignment, the scale difference σ is calculated.

3) Optimal Scale Difference σ^*: In our single loop closure method, more than three consecutive keyframes, including $\mathbf{K}^i_s$ and $\mathbf{K}^j_t$ data, are used, as shown in Fig. 4(a). For each match, the scale difference is estimated.

Let us assume γ_{ij} is the number of matched keypoints among two keyframes, i and j. These matches yield a distinct scale difference σ_{ij} depending on the number of matched keypoints γ_{ij}. The optimal scale difference σ^* will be

$$\sigma^* = \underset{\gamma}{\operatorname{argmax}} \frac{1}{2}\left|\gamma\left(\sigma_{ij}\right), \gamma\left(\sigma_{i'j'}\right)\right| \tag{4}$$

where σ_{ij} and $\sigma_{i'j'}$ are the two nearest points such that

$$\left|\sigma_{ij} - \sigma_{i'j'}\right| \leq \Delta^* \quad \forall \quad i, j, i' \text{and} j' \in \mathbb{Z}^+, \quad \Delta^* \in \mathbb{R}. \tag{5}$$

The estimation is further explained in Algorithm 1.

D. Relative Pose Transformation

Point cloud information assists further in finding the relative pose transformation of slave robots. After obtaining the optimal scale difference σ^*, the scale factor is multiplied by the point cloud to make both point clouds of equal scale. By multiplying σ^* with the rotation matrix $\mathbf{R}$ in (1), we obtain the scaling transformation ${}^{\sigma}\mathbf{T}^*_s$. For instance, if $P^{\mathcal{F}^i_s}_{s(i)}$ is smaller than $P^{\mathcal{F}^j_t}_{t(j)}$, then $P^{\mathcal{F}^i_s}_{s(i)}$ will be scaled by σ^* as follows:

$${}^{*}P^{\mathcal{F}^i_s}_{s(i)} = {}^{\sigma}\mathbf{T}^*_s\left(P^{\mathcal{F}^i_s}_{s(i)}\right).$$

1) Alignment of Point Clouds: After scaling the point cloud of source agent $P^{\mathcal{F}^i_s}_{s(i)}$ by σ^*, both the point clouds $P^{\mathcal{F}^j_t}_{t(j)}$ and $P^{\mathcal{F}^i_s}_{s(i)}$ are aligned before applying the ICP. Therefore, we transform the point cloud $P^{\mathcal{F}^j_t}_{t(j)}$ from $\mathcal{F}^j_t$ to $\mathcal{F}_w$

$$P^{\mathcal{F}_w}_{t(j)} = {}^{t(j)}\mathbf{T}^{-1}_w\left(P^{\mathcal{F}^j_t}_{t(j)}\right) = {}^{w}\mathbf{T}_{t(j)}\left(P^{\mathcal{F}^j_t}_{t(j)}\right). \tag{6}$$

Then, we transform the point cloud $P^{\mathcal{F}^j_t}_{t(j)}$ from $\mathcal{F}_w$ to $\mathcal{F}^i_s$

$$P^{\mathcal{F}^i_s}_{t(j)} = {}^{s(i)}\mathbf{T}_w\left(P^{\mathcal{F}_w}_{t(j)}\right). \tag{7}$$

Algorithm 1: Finest Tuning for Optimal Scale Estimation

Input: Matched keyframes $\mathbf{K}_{r_{ID}} = \{\mathbf{K}_s^i, \mathbf{K}_t^j\}$, poses ${}^{w}\mathbf{T}_{r_{ID}} = \{{}^{w}\mathbf{T}_s, {}^{w}\mathbf{T}_t\}$, point clouds $P^{\mathcal{F}}_{r_{ID}} = \{P^{\mathcal{F}^i_s}_{s(i)}, P^{\mathcal{F}^j_t}_{t(j)}\}$ with $i, j \in \mathbb{Z}^+$

Output: Optimal scale difference σ^*, initial guess relative transformation ${}^{si}\mathbf{T}^{IG}_{ti}$

initialization;

for $z = -1{:}1$ **do**

$P^{\mathcal{F}_w}_{s(i+z)} = {}^{w}\mathbf{T}_{s(i+z)}(P^{\mathcal{F}_{s(i+z)}}_{s(i+z)})$;

$P^{\mathcal{F}_w}_{t(j+z)} = {}^{w}\mathbf{T}_{t(j+z)}(P^{\mathcal{F}_{t(j+z)}}_{t(j+z)})$;

Function `PCR-Pro [?]` $(\mathbf{K}_{r_{ID}}, P^{\mathcal{F}_w}_{r_{ID}})$**:**

Estimate volume ratio r_{vol} of $P^{\mathcal{F}_w}_{s(i+z)}, P^{\mathcal{F}_w}_{t(j+z)}$;

${}^{s(i+z)}\mathbf{T}^{RC}_{t(j+z)} \longleftarrow \gamma_z \longleftarrow \mathbf{K}_s^{i+z}, \mathbf{K}_t^{j+z}$;

$\sigma_z \longleftarrow {}^{s(i+z)}\mathbf{T}^{RC}_{t(j+z)}, \gamma_z, {}^{w}\mathbf{T}_{s(i+z)}, {}^{w}\mathbf{T}_{t(j+z)}$;

${}^{s(i+z)}\mathbf{T}^{IG}_{t(j+z)} \longleftarrow \sigma_z, P^{\mathcal{F}_w}_{s(i+z)}, P^{\mathcal{F}_w}_{t(j+z)}$;

return $\sigma_z, {}^{s(i+z)}\mathbf{T}^{IG}_{t(j+z)}$;

if $r_{vol} > 0.5$ **then**

$\Delta^* = 5$;

for $x = -1{:}1$ **do**

for $y = -1{:}1$ **do**

if $x \neq y$ **then**

$\Delta = |\sigma_x - \sigma_y|$;

if $\gamma^* < \gamma_{xy}$ && $\Delta^* > \Delta$ && $\Delta^* \neq 0$ **then**

$\sigma^* = avg(\sigma_x, \sigma_y)$;

$\Delta^* = \Delta$;

$\gamma^* = \gamma_{xy}$;

else

$\sigma^* = \sigma_{xy=00}$;

Finally, initial guess transformation is applied to align them further with ${}^{*}P^{\mathcal{F}^i_s}_{s(i)}$

$$ {}^{+}P^{\mathcal{F}^i_s}_{t(j)} = {}^{s(i)}\mathbf{T}^{\mathrm{IG}}_{t(j)}\left(P^{\mathcal{F}^i_s}_{t(j)}\right). \tag{8}$$

2) Transformation Computation: After transforming $P^{\mathcal{F}^i_t}_{t(j)}$ to ${}^{+}P^{\mathcal{F}^i_s}_{t(j)}$, both ${}^{+}P^{\mathcal{F}^i_s}_{t(j)}$ and ${}^{*}P^{\mathcal{F}^i_s}_{s(i)}$ are aligned and are almost the same scale. By using the filtered point clouds of both agents, the libpointmatcher [26] approach is applied to find the final transformation ${}^{s(i)}\mathbf{T}^{\mathcal{F}_{\mathrm{trans}}}_{t(j)}$, where the entire fusion converges to the global minima. We use the point-to-point matrix-based error method. The system applies this transformation to the target point cloud that fuses with the source point clouds and gives excellent map merging.

E. Update of Poses and 3-D Map

If we write the compact form of all the transformations, the final optimal transformation can be written as

$$ {}^{s}\mathbf{T}^{*}_{t} = {}^{s(i)}\mathbf{T}^{\mathcal{F}_{\mathrm{trans}}}_{t(j)} * {}^{s(i)}\mathbf{T}^{\mathrm{IG}}_{t(j)} * {}^{s(i)}\mathbf{T}_{w} * {}^{t(j)}\mathbf{T}^{-1}_{w}. \tag{9}$$

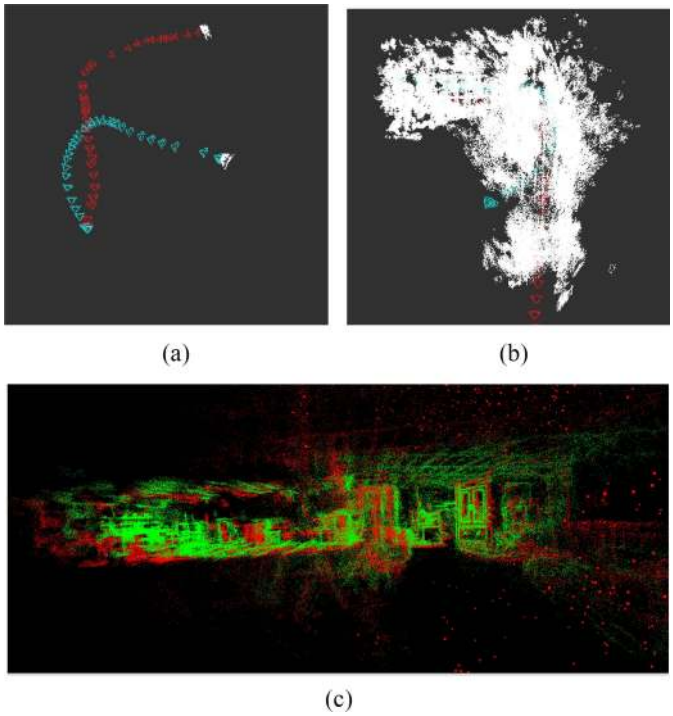

Fig. 5. *Ri* dataset results: live multiagent SLAM after applying the Loop-box method. The poses and 3-D maps of the source slave are shown in red, while the target slave poses are shown in cyan, with the 3-D map in green. (a) Original visual odometry of both agents starts from the origin. (b) Visual odometry poses of the target robot are synchronized to the source robot's frame of reference. (c) Detailed insight of excellent 3-D map merging using Loop-box.

For the global mapping, all the poses and 3-D maps are updated using the ${}^{\sigma}\mathbf{T}^{*}_{s}$ and ${}^{s}\mathbf{T}^{*}_{t}$ transformation to the source and target slaves, respectively, and the multiagent SLAM starts working as shown in Fig. 5.

V. Loop-Box Results and Comparison

This work addresses the compatibility and flexibility of multiagent systems by introducing a robust and highly relaxed environment for a multiagent SLAM framework. In comparison to the existing systems, Loop-box can localize the agents and fuse their maps effectively without using the feature keypoints' history. It efficiently performs map merging and reliable relocalization of agents at the first loop closure.

A. Datasets

In this article, we assess the performance of Loop-box on five different datasets. The *Ri* dataset consists of the indoor movements of two agents but the indoor environment is feature rich compared to the outdoor environment. Therefore, we further test our framework on four outside datasets *Academic building* and *Parkings* 1, 2, and 3. The *Ri* and *Parking* 1 datasets were taken by using a monocular camera connected with a laptop, carried by a human as an agent. Meanwhile, the *Academic building*, *Parking* 2, and *Parking* 3 bag files were captured using an unmanned ground vehicle (UGV), which is shown in Fig. 4(b). For the multiagent scenario, the target agent of the *Parking* 2 dataset is the source agent of the *Parking* 3 dataset.

TABLE I
RELATIVE POSE RMSE OF THE PROPOSED METHOD COMPARED TO PCR-PRO AND PGO

Dataset	Type	Agents	Direction	Total Time Duration (sec)		Scale		PCR-Pro [14]		Pose Graph Optimization [27]		Loop-box (Proposed)	
				Source Agent	Target Agent	Estimation Time (sec)	Scale Difference	Time (sec)	RMSE (m)	Time (sec)	RMSE (m)	Time (sec)	RMSE (m)
Ri	Indoor	2	Same	53.7	69	0.98	2.4922	7.6	0.1503	5.1	0.0903	**2.53**	**0.0290**
Academic building	Outdoor	2	Same	73	74	0.89	2.83519	8.8	1.2453	5.5	0.9832	**2.93**	**0.0323**
Parking 1	Outdoor	2	Same	104	91	0.94	1.7857	7.4	1.5594	5.04	0.0883	**2.47**	**0.0486**
Parking 2	Outdoor	3	Opposite	122	117	0.92	1.2786	9	1.6	5.57	0.207	**3**	**0.175**
Parking 3			Same	122	112	0.867	1.15	11	0.3613	6.24	0.3073	**3.67**	**0.0712**

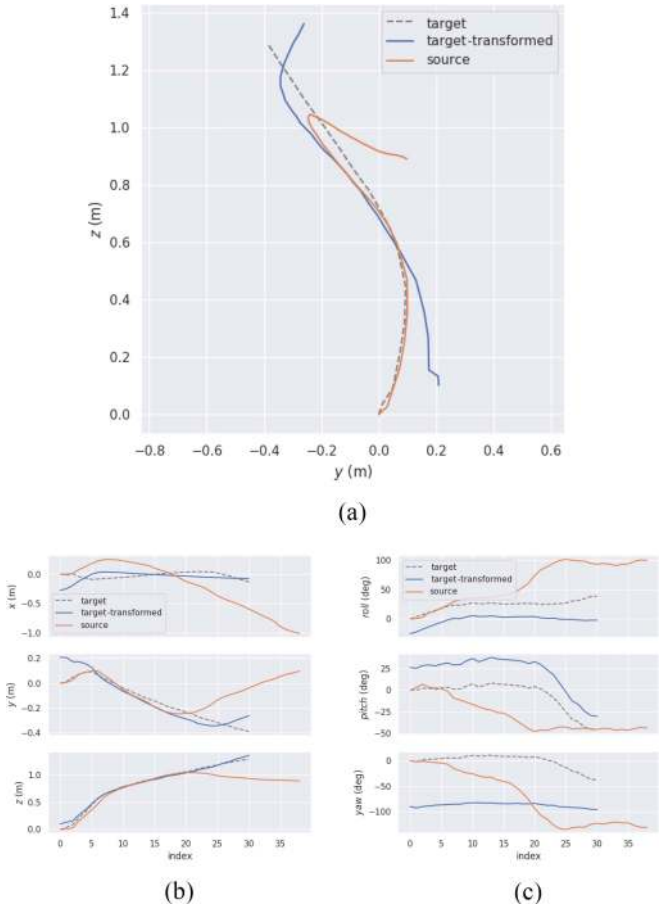

Fig. 6. RTT for the enhanced loop closure evaluation on the *RI* dataset. The final trajectories of the source and target agents are shown in orange and blue, respectively. (a) Visual odometry poses of the transformed target agent to the source agent's frame of reference. (b) After the transformation, visual odometry comparison on each axis. (c) Roll, pitch, and yaw results after the transformation.

B. Quantitative and Qualitative Analysis

We qualitatively and quantitatively evaluate the Loop-box framework for both online and offline testing. We record the bags files at different times to capture various conditions. For instance, some objects such as cars are present in the *Parking* 2 dataset, whereas they are missing in the *Parking* 3 dataset. One of the numerous benefits of using the one loop-closure-based method is that it can easily estimate relative transformation in those environments where agents meet for a short interval.

1) Qualitative Results: The *Ri* dataset was recorded inside the Robotics Institute of HKUST. We captured it using a monocular camera connected to a laptop at a normal speed of human motion. We apply the ${}^{\sigma}\mathbf{T}_{s}^{*}$ and ${}^{s}\mathbf{T}_{t}^{*}$ transformation estimated in the last section to the poses and point clouds of both agents. It transforms the poses of the target slave to the source slave frame of reference $\mathcal{F}_s$, as shown in Fig. 5(b). The SLAM and map merging results are shown in Fig. 5. The original trajectory of the target agent starts with the source agent from the origin $[0, 0, 0]$. For the multiagent case, SLAM systems have an optimal relative transformation to another agent's frame of reference.

After applying the transformation estimated using Loop-box, the relative trajectory transformation (RTT) is as shown in Fig. 6(a). We further analyze the behavior from each axis, as shown in Fig. 6(b). In each direction, we can observe that they both start at the same point. Furthermore, there is a significant movement of the target agent in the x- and z-direction. But in the y-direction, few variations of poses are detected, as the camera height is almost the same throughout the agents' movement. We also observe the changes in terms of yaw, pitch, and roll, as shown in Fig. 6(c). After applying the Loop-box method, the target rotation is corrected and transformed according to the source agent reference frame. Large variations of poses are observed in the roll and pitch axis. Furthermore, about a two-times shift in the yaw axis is also detected.

2) Quantitative Results: In Table I, we compare the relative pose root-mean-square error (RMSE) of Loop-box with that of PCR-Pro [14] and pose graph optimization (PGO) [27]. Note that both the Loop-box and PGO methods use a scale difference and relative transformation first computed by PCR-Pro. This is why their results are better as compared to PCR-Pro's. Furthermore, we separately analyze the relative pose error (RPE) of PCR-Pro on the three parking datasets. For *Parking* 1 and *Parking* 2, the RPE is much higher than for the *Parking* 3 dataset, as shown in Fig. 7(a). If we look at the performance of bundle adjustment using PGO, the RPE is small. The interesting factor to note is that Loop-box outperforms PGO further. We evaluate the Loop-box performance compared to PGO's, as shown in Fig. 7(b). Moreover, the

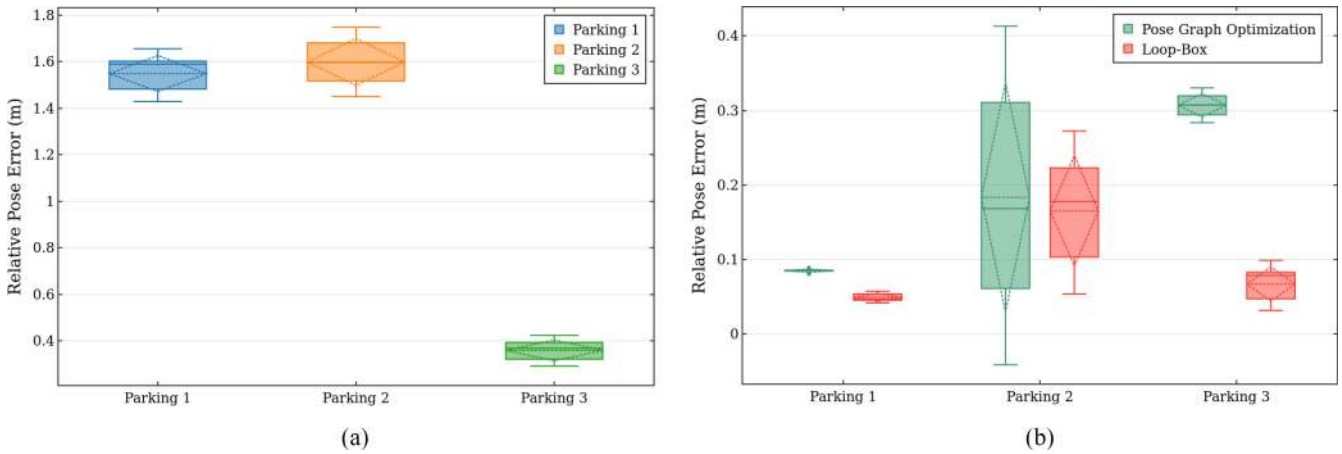

Fig. 7. RPE (a) PCR-Pro applied directly to the first match corresponding to each dataset. The RPE results show that the error varies due to the number of matched keypoints of each matching. (b) Evaluation of the Loop-box performance compared to state-of-the-art PGO [27] for each dataset.

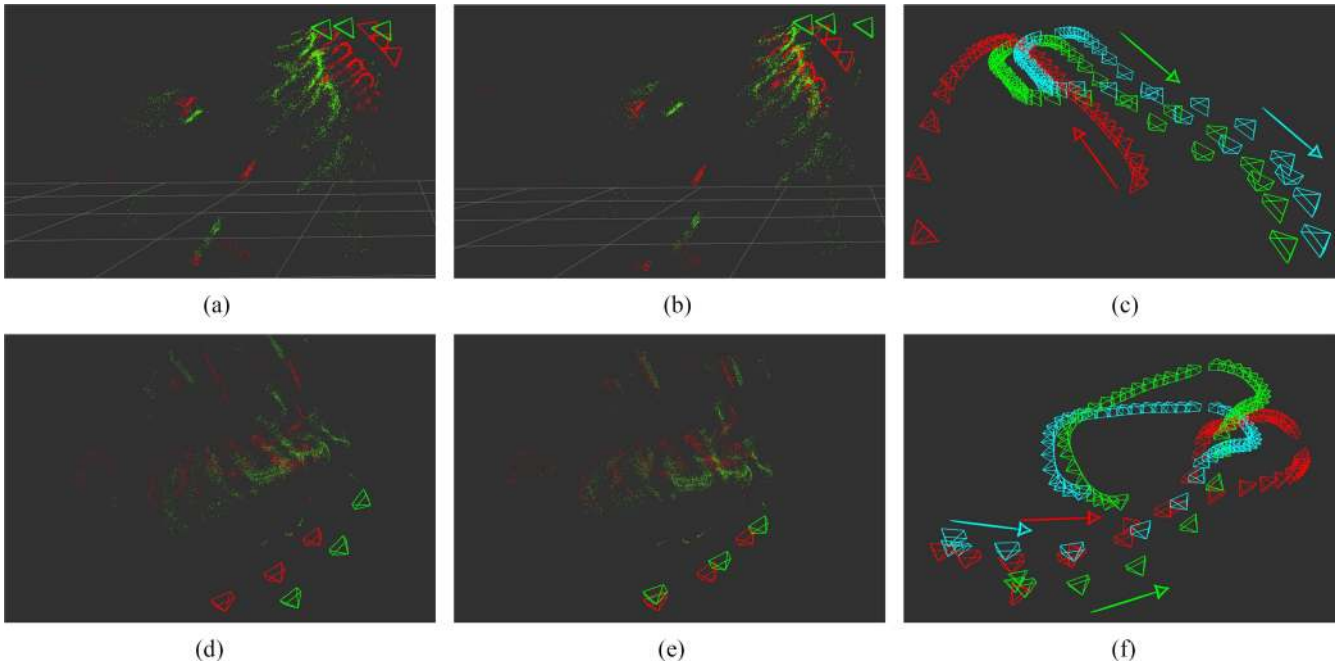

Fig. 8. (a), (b), and (c) correspond to the *Parking* 2 dataset and (d), (e), and (f) correspond to the *Parking* 3 dataset. Travelling directions of the agents are shown with arrows in (c) and (f). In (a), (b), (d), and (e), red represents the source poses and point clouds and green represents the target agent poses and point clouds. In (c) and (f), the source is in red, while the green camera poses are those estimated by the PGO method, and cyan represents the camera poses transformed by Loop-box.

estimation time of Loop-box is much lower than that of PGO, which requires additional nonlinear optimization for the bundle adjustment. The computation time and RMSE are compared in Table I. To show the robustness of Loop-box according to the motivation mentioned in Section I, we discuss the results in two sections: 1) unidirectional and bidirectional results and 2) multiagent 3-D mapping results.

C. Unidirectional and Bidirectional Results

For the unidirectional and bidirectional results, the *Parking* 2 and *Parking* 3 datasets were recorded using the UGV shown in Fig. 4(b). These datasets consist of three bag files. The first two bags have the same path direction for loop closure, and the third bag has the opposite direction. We compare the map merging results of PGO and Loop-box in Fig. 8 since both perform better than PCR-Pro. The results show that the map merging by Loop-box converges to the global optimum in contrast to the bundle adjustment using PGO. The difference is viewed best in the matched poses location. All camera poses along with 3-D maps transformed by bundle adjustment using the PGO and Loop-box methods are shown in Fig. 8.

D. Multiagent 3-D Mapping

To have multiagent 3-D mapping results, we combined the results of the *Parking* 2 and *Parking* 3 datasets. In the *Parking* 3 dataset, both agents are moving in the same direction, whereas in the *Parking* 2 dataset, the agents are moving in opposite directions. If the system can detect merely one loop closure, then the 3-D maps can easily be merged using Loop-box. All three agents originally had different scales and

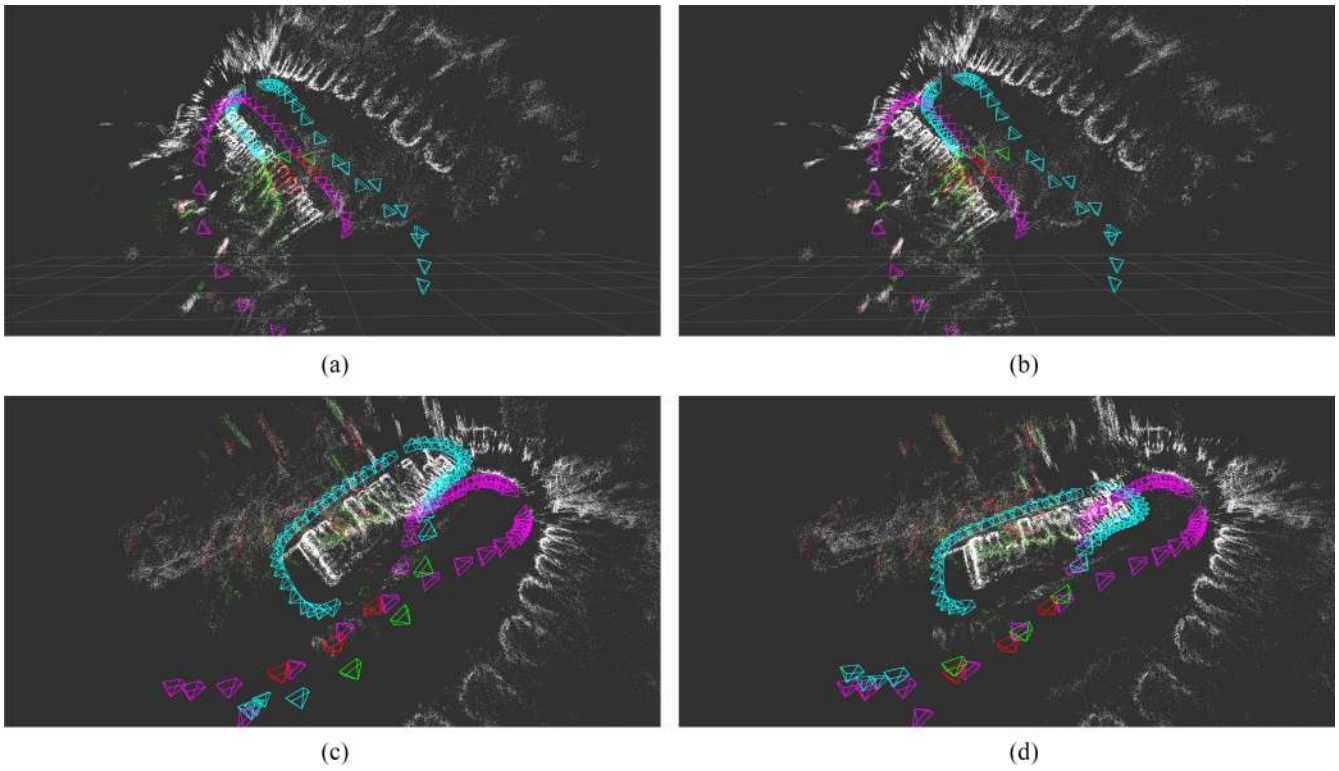

Fig. 9. Multiagent SLAM including full map merging comparison of Loop-box with PGO. Poses of the source agent and target are shown in magenta and cyan, respectively. The loop closure matches of the source and target agent are shown in red and green, respectively. (a) *Parking* 2 dataset: SLAM results after bundle adjustment by PGO. (b) *Parking* 2 dataset: SLAM results using Loop-box. (c) *Parking* 3 dataset: SLAM results after bundle adjustment by PGO. (d) *Parking* 3 dataset: SLAM results using Loop-box.

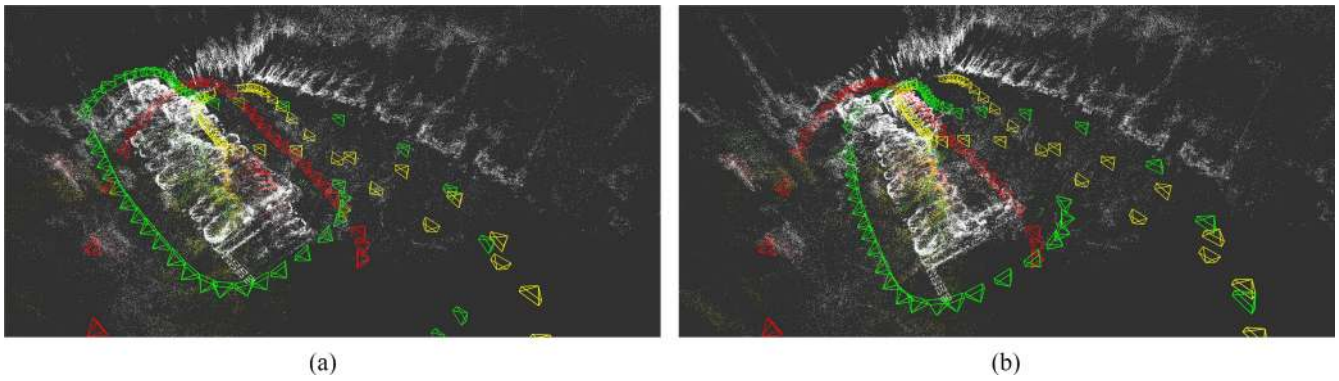

Fig. 10. Multiagent 3-D map fusion. The target agent of the *Parking* 2 dataset is the source agent of the *Parking* 3 dataset. The odometry of this connecting agent is shown in yellow. The source agent of the *Parking* 2 dataset and target agent of the *Parking* 3 dataset are shown in red and green, respectively. (a) *Parking* 2 and *Parking* 3 datasets: PGO-based multiagent results. (b) *Parking* 2 and *Parking* 3 datasets: Loop-box-based multiagent results.

camera odometries. After estimating the optimal scale, we processed the three adjacent matches using Loop-box and PGO, as shown in Fig. 9. Since the target agent of the *Parking* 2 dataset is the source agent of the *Parking* 3 dataset, we name the agent a connecting agent. First, we scale the source agent of the *Parking* 2 dataset according to the connecting agent. This transformation was further scaled according to the target agent of the *Parking* 3 dataset. Next, the target agent of the *Parking* 3 dataset was transformed according to the merged *Parking* 2 dataset. We tested Loop-box compared to the PGO method to merge the 3-D map with the connecting agent. The full map merging results are shown in Fig. 10.

Material related to this work is available at https://usmanmaqbool.github.io/loop-box.

VI. Discussion

A challenging application of the multiagent SLAM system is a crowd-based mapping system where different people use their camera phones to capture the 3-D environment and then collaborate to yield large-scale 3-D mapping. After recognizing the first loop closure, our Loop-box framework is successfully able to find the relative pose transformation regardless of the scale difference between them. For the

PCR-Pro results, we estimated the relative transformation individually. In the PGO, we selected the top three methods, which are discussed in detail in the Appendix. In the end, we used our proposed method, which needs only one match out of three based on the keypoints. For all the datasets, 3-D map merging accuracy describes the efficiency of the Loop-box method where the system converges to the global minimum, as presented in Table I. We discuss the limitation of multiagent SLAM systems and specifications for improving Loop-box as follows.

A. Place Recognition

With the short-interval loop closure, a number of scenarios can happen. If both agents are moving in the same direction, the system does not need any modification. But if both agents are coming from opposite directions, as in the *Parking* 2 dataset, the camera system should capture the side view of the agent path so both agents can accurately detect the loop closure. After the loop closure, the adjacent keyframe matching will not be in a direct but in a crossed manner, as shown in red in Fig. 4(a). For the distributed multiagent SLAM system, if two agents detect that they are in range of each other, they can quickly establish relative pose connection using Loop-box.

In this work, we use FABMAP [25] and create the visual vocabulary for loop closure detection. Preparing a visual vocabulary is time consuming and even not recommended for large-scale mapping. Some methods [23] also offer a prebuilt dictionary for place recognition. This enables the system to be used without training the environment. Loop-box can be further improved by incorporating NetVLAD features [24]. For all, the features will be calculated and uploaded to the server instead of all the keyframes. This can give a significant improvement in the detection time of loop closures. The NetVLAD features handling is also simpler than vocabulary handling. Sometimes, biologically inspired visual SLAM systems [28], [29] also help in persistent navigation and mapping in a dynamic environment.

B. Drift

In our results, we can see excellent transformations at the loop closure area, but a small drift is clearly visible afterward. Since LSD-SLAM is based on a monocular camera only, drift can affect local mapping. The Loop-box method allows monocular camera-based SLAM systems to connect and find the exact transformation. These systems can have drift problems, which can be solved using additional sensors. For example, OKVIS can track a local map built from several recently captured keyframes, which significantly minimizes the local drifts [30]. It can also be improved by a tightly coupled configuration for sensor fusion of the camera with an IMU. This helps in avoiding the local drift in the visual odometry of the agent.

VII. Conclusion

This article has presented a framework that can estimate real-time 3-D relative transformation of different agents for large-scale SLAM and 3-D map-merging applications. The results have shown admirable performance regarding map merging and accurate estimation of the relative pose transformation after only a single loop closure. After the first matches, our system has the ability to localize each agent with respect to the global reference frame while keeping track of and continuously adding keyframes and point cloud data to the global map. This work can be extended to n agents of any kind, for instance, unmanned air and ground vehicles. Features of this research are its computationally robust framework, and its possible applicability to swarm robotics in map exchange, and task allocation applications.

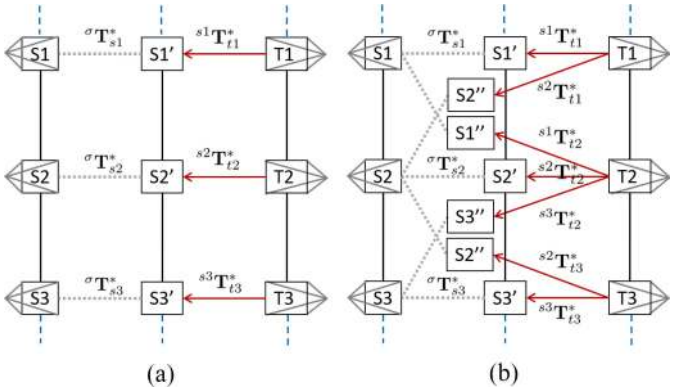

Fig. 11. Different pose configurations for PGO. (a) Direct match configuration. (b) Fully connected configuration.

Appendix
Bundle Adjustment Study for Multiagent SLAM

Several studies suggest that bundle adjustment using PGO performs excellently in SLAM systems. To compare Loop-box with bundle adjustment, we, in addition, estimated the covariance among edges. PCR-Pro [14] along with libpointmatcher [26] were used to achieve high-level optimal transformation. Using the estimated covariance based on the sets of correspondence and relative transformation, we can easily determine the information matrix, enabling the system for PGO. We considered three different cases, and applied the PGO to the edge transformations, which are shown by the gray dotted lines in Fig. 11.

Straight Configuration: For the straight configuration, we examined directly matched camera poses, as shown in Fig. 11(a). $S1$, $S2$, and $S3$ are source poses, while $T1$, $T2$, and $T3$ are the corresponding target poses. We applied the transformation ${}^{s1}\mathbf{T}^*_{t1}$, ${}^{s2}\mathbf{T}^*_{t2}$, and ${}^{s3}\mathbf{T}^*_{t3}$ estimated in (9) to $T1$, $T2$, and $T3$ poses, respectively. Moreover, we scaled the $S1$, $S2$, and $S3$ poses by ${}^{\sigma}\mathbf{T}^*_{s1}$, ${}^{\sigma}\mathbf{T}^*_{s2}$, and ${}^{\sigma}\mathbf{T}^*_{s3}$. Then, we applied the bundle adjustment for overall transformation optimization.

Fully Connected Configuration: After the loop closure, cross matched pairs $\{T1, S2''\}$, $\{T2, S1''\}$, $\{T2, S1''\}$, $\{T2, S3''\}$, and $\{T3, S2''\}$ of three adjacent poses were also selected along with the directly matched pairs for the full PGO, as shown in Fig. 11(b).

Top Matches Configuration: This configuration is similar to the straight configuration as explained above. The difference

is that we use a single transformation ${}^{s(j)}\mathbf{T}^{*}_{t(i)}$, where i to j corresponds to those poses that have large matched keypoints γ. For two agents, there is one optimal transformation for all the poses.

In all the above configurations, the top matches configuration gives much better results than the direct and fully connected setup. After the PGO using this best pose solution, we use the middle poses of $S2$ and $T2$. Another good factor in using this for bundle adjustment is that it not only estimates better relative transformation but also takes less time in computation. The top matches configuration results have been compared with the Loop-box method in Section V.

REFERENCES

[1] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *J. Field Robot.*, vol. 36, no. 4, pp. 763–781, 2019.

[2] P. Schmuck and M. Chli, "Multi-UAV collaborative monocular SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, 2017, pp. 3863–3870.

[3] T. Schneider *et al.*, "Maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1418–1425, Jul. 2018.

[4] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework," *Int. J. Robot. Res.*, vol. 23, no. 12, pp. 1113–1139, Dec. 2004.

[5] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[7] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[8] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 796–803, Apr. 2017.

[9] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.

[10] I. Deutsch, M. Liu, and R. Siegwart, "A framework for multi-robot pose graph SLAM," in *Proc. IEEE Int. Conf. Real Time Comput. Robot. (RCAR)*, Angkor Wat, Cambodia, Jun. 2016, pp. 567–572.

[11] X. Chen, H. Lu, J. Xiao, and H. Zhang, "Distributed monocular multi-robot SLAM," in *Proc. IEEE 8th Annu. Int. Conf. CYBER Technol. Autom. Control Intell. Syst. (CYBER)*, Tianjin, China, 2018, pp. 73–78.

[12] A. Gil, Ó. Reinoso, M. Ballesta, and M. Juliá, "Multi-robot visual SLAM using a Rao-Blackwellized particle filter," *Robot. Auton. Syst.*, vol. 58, no. 1, pp. 68–80, Jan. 2010.

[13] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Proc. IEEE Int. Conf. Robot. Autom.*, Roma, Italy, 2007, pp. 3167–3172.

[14] M. U. M. Bhutta and M. Liu, "PCR-Pro: 3D sparse and different scale point clouds registration and robust estimation of information matrix for pose graph SLAM," in *Proc. IEEE 8th Annu. Int. Conf. CYBER Technol. Autom. Control Intell. Syst. (CYBER)*, Tianjin, China, 2018, pp. 354–359.

[15] S. M. Prakhya, L. Bingbing, Y. Rui, and W. Lin, "A closed-form estimate of 3D ICP covariance," in *Proc. 14th IAPR Int. Conf. Mach. Vis. Appl. (MVA)*, Tokyo, Japan, May 2015, pp. 526–529.

[16] L. Kneip and P. Furgale, "OpenGV: A unified and generalized approach to real-time calibrated geometric vision," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, China, May 2014, pp. 1–8.

[17] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, vol. 3. Kobe, Japan, 2009, p. 5.

[18] B. Jacob and G. Gunnebaud. (2010). *Eigen V3*. [Online]. Available: http://eigen.tuxfamily.org

[19] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, Nov. 2000. [Online]. Available: https://github.com/opencv/opencv/wiki/CiteOpenCV

[20] S. Agarwal *et al.* (2012). *Ceres Solver*. [Online]. Available: http://ceres-solver.org/#cite-us

[21] M. Grupp. (2017). *EVO, Python Package for the Evaluation of Odometry and SLAM*. [Online]. Available: https://github.com/MichaelGrupp/evo

[22] S. O'Hara and B. A. Draper, "Introduction to the bag of features paradigm for image classification and retrieval," 2011. [Online]. Available: http://arxiv.org/abs/1101.3354.

[23] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

[24] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 40, Jun. 2018, pp. 5297–5307.

[25] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *Int. J. Robot. Res.*, vol. 30, no. 9, pp. 1100–1123, 2011.

[26] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart, "Tracking a depth camera: Parameter exploration for fast ICP," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, CA, USA, Sep. 2011, pp. 3824–3829.

[27] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, 2011, pp. 3607–3613.

[28] M. Milford and G. Wyeth, "Persistent navigation and mapping using a biologically inspired SLAM system," *Int. J. Robot. Res.*, vol. 29, no. 9, pp. 1131–1153, 2010.

[29] J. Ni, L. Wu, X. Fan, and S. X. Yang, "Bioinspired intelligent algorithm and its applications for mobile robot control: A survey," *Comput. Intell. Neurosci.*, vol. 2016, Jan. 2016, Art. no. 3810903. [Online]. Available: https://www.hindawi.com/journals/cin/2016/3810903/

[30] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, 2015.