*Research Article*

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

*International Journal of Advanced Robotic Systems*
May-June 2018: 1–20

DOI: 10.1177/1729881418780178
journals.sagepub.com/home/arx
SAGE

# Distributed and collaborative monocular simultaneous localization and mapping for multi-robot systems in large-scale environments

**Hui Zhang, Xieyuanli Chen, Huimin Lu and Junhao Xiao**

**Abstract**

In this article, we propose a distributed and collaborative monocular simultaneous localization and mapping system for the multi-robot system in large-scale environments, where monocular vision is the only exteroceptive sensor. Each robot estimates its pose and reconstructs the environment simultaneously using the same monocular simultaneous localization and mapping algorithm. Meanwhile, they share the results of their incremental maps by streaming keyframes through the robot operating system messages and the wireless network. Subsequently, each robot in the group can obtain the global map with high efficiency. To build the collaborative simultaneous localization and mapping architecture, two novel approaches are proposed. One is a robust relocalization method based on active loop closure, and the other is a vision-based multi-robot relative pose estimating and map merging method. The former is used to solve the problem of tracking failures when robots carry out long-term monocular simultaneous localization and mapping in large-scale environments, while the latter uses the appearance-based place recognition method to determine multi-robot relative poses and build the large-scale global map by merging each robot's local map. Both KITTI data set and our own data set acquired by a handheld camera are used to evaluate the proposed system. Experimental results show that the proposed distributed multi-robot collaborative monocular simultaneous localization and mapping system can be used in both indoor small-scale and outdoor large-scale environments.

**Keywords**

Multi-robot collaborative SLAM, monocular SLAM, relocalization, large-scale SLAM

Date received: 8 October 2017; accepted: 5 May 2018

Topic: Special Issue - 3D Vision for Robot Perception
Topic Editor: Antonio Fernandez-Caballero
Associate Editor: Tiziana D'Orazio

## Introduction

As the monocular camera is much cheaper, physically smaller and lower powered than other vision systems, for example, stereo and RGB-D cameras, it has been widely applied in the fields of computer vision and robotics. The state-of-the-art monocular simultaneous localization and mapping (SLAM) algorithms have achieved remarkable performance in rich featured, static indoor environments. However, when it moves to multi-robot platforms carrying

Department of Automation, National University of Defense Technology, Changsha, China

**Corresponding author:**
Xieyuanli Chen, Department of Automation, National University of Defense Technology, 137 Yanwachi Street, Changsha, Hunan 410073, China.
Email: chenxieyuanli@hotmail.com

out long-term SLAM in large-scale environments, there are still many challenges to be addressed. The first of all is the tracking failure problem. Since it is difficult to calculate the accurate depth of observed features using only one camera, most existing monocular SLAM systems are less robust than those based on stereo or RGB-D cameras.

When tracking failure happens, the most effective solution is to add an image-to-map relocalization module, because a large amount of measurements in the map can be used to mitigate the side effect of outliers and higher accuracy can be achieved in pose estimation.[1] However, most of the image-to-map approaches are resource-consuming and normally designed for small workplace or indoor scenarios.[1–3] They are obviously inapplicable for mobile robots to perform long-term tasks. Inspired by Strasdat et al.[4] and Mur-Artal et al.,[5] our system only performs feature matching in a local map when the tracking fails, so the computational complexity is up-bounded. Thus, real-time relocalization can be realized in relative larger environments.

After the relocalization, the drift of SLAM may increase, since during the tracking failure period, the robot is not able to collect enough information to accurately localize itself and build the map. It is well known that loop closure can effectively eliminate the accumulated error. However, most current research only focus on how to detect and close a loop passively but not to control robots to actively create loop closures to correct the SLAM drift. In this article, we propose an active loop closure approach, which uses the information of the robot/camera pose obtained from the relocalization to navigate robots finding a loop actively and, therefore, eliminate the accumulated drift.

Another problem of robots carrying out large-scale SLAM is that the computing capacities of a single robot are normally limited. Naturally, we think of using a multi-robot system to solve this problem. A multi-robot system allows parallel execution of tasks, and, in addition, is more efficient and robust compared against a single-robot system. Considering a scenario of employing a multi-robot team to map a large unknown environment, the task can be divided among all the team members who can collaboratively build a global map to reduce the overall execution time. In a group of robots, each robot should not only rely on its own information but also the information provided by the others. This is not an easy task since the robots in general do not have any prior information about each other's location. To allow the robots working cooperatively, relative pose estimation and map merging are two fundamental issues.

To calculate the relative pose of the robots, relative localization methods impose the least limitation on both the motion of the robots and the prior information.[6] In addition, this method can realize the merging of local maps without large overlaps. In this article, a cooperative appearance-based place recognition method is employed to enable the robots to identify similar scenes in their visual perception and then compute their relative poses. The similar scenes are identified by performing image-to-image place matching, which is usually employed for loop closure in single-robot SLAM.

The main contributions of our work can be concluded as follows:

- An image-to-map relocalization method based on local map is proposed, which limits the calculation burden and realizes the unvisited place relocalization in large-scale scenes.
- A robust relocalization system based on active loop closure is proposed, where the pose information obtained from the latest relocalization is utilized to navigate the robot to find a loop actively and in turn eliminate the drift caused by the tracking failure.
- A distributed multi-robot collaborative SLAM based on the robust monocular SLAM is proposed, by which a team of robots can cooperatively map the large-scale environment with high efficiency.
- A relative pose calculation and map merging method is proposed, by which the multi-robot collaborative SLAM can be realized without any prior knowledge and large map overlaps.

## Related work

### Monocular SLAM

Monocular SLAM was initially solved by filtering,[7] and now it has been developed into two main branches: feature-based approaches and direct approaches.

*Feature-based approaches.* These methods include two steps: features are first extracted from the images, and then the robot pose is calculated and the environment is mapped based on these features. Early versions of these approaches are based on filters,[7,8] using all the measurements to update the probability distributions of features and poses, which are less accurate and can only be used in small-scale scenes. In order to better realize modern applications, feature-based monocular SLAM methods are now mostly based on keyframes and bundle adjustment (e.g. ORB-SLAM[5]), which can have high accuracy and low computation cost.

*Direct approaches.* Direct methods can achieve high accuracy and robustness by optimizing the geometry directly on the image intensities.[9–11] However, either some of the direct methods mainly focus on visual localization (e.g. DSO[11]), or some others' maps are too dense to be used for outdoor large-scale multi-robot collaborative mapping system (e.g. LSD-SLAM[9] and DTAM[10]). Considering the limited bandwidth of robots' communication, we, therefore, choose the feature-based ORB-SLAM as the

implementation basis, which has a lightweight map representation and, meanwhile, can also achieve high accuracy and robustness after adding the proposed relocalization module.

### Relocalization

The relocalization problem can be traced back to global self-localization in given maps and was first realized by Dellaert et al.[12] With robots gradually applied to real scenes, the relocalization became a tracking recovery problem, and three types of approaches were proposed[13]: map-to-map, image-to-image and image-to-map.

*Map-to-map.* It was first implemented by Clemente et al.[14] They found matches between landmarks in two submaps, which are very complex and time-consuming. Also, because the sparse map built by monocular SLAM does not always provide enough mappoints, this method is less accurate and unreliable for monocular SLAM.

*Image-to-image.* With the monocular SLAM developed from the initial filtering approaches to keyframe-based approaches, the image-to-image method has been explored rapidly. It was first realized by Reitmayr and Drummond[15] and Klein and Murray.[16] Then, Sivic and Zisserman[17] proposed bag of words (BoWs) using visual words for place recognition, which was widely used in the relocalization and loop closure for its high accuracy. For example, Eade and Drummond[18] used BoW with SIFT[19] to determine the current submap. Cummins and Newman[20] used it with SURF[21] to achieve high robustness to perceptual aliasing. Mur-Artal and Tardós[22] used it with ORB[23] and built the well-known ORB-SLAM.[5] Although it has become popular to use image-to-image method in the relocalization, this method is only effective when robots run in the previously visited places where keyframes exist. This assumption does not hold for mobile robots when running in large-scale environments exploring unvisited places.

*Image-to-map.* This problem can be solved by image-to-map approaches which were first proposed by Pupilli and Calway.[24] They built a short-distance tracking recovery system by exploring multiple hypotheses with a particle filter. To realize a global image-to-map approach, Se et al.[25] used SIFT features in two-dimensional (2D) scenario. Williams et al. proposed a three-dimensional (3D) relocalization based on a filtering approach and feature recognition.[1] They published a survey comparing different types of relocalization and loop closing approaches.[13] According to the survey, to deal with the relocalization in monocular SLAM, image-to-map approaches perform best for higher accuracy, speed and robustness. Afterwards, Straub et al.[3] used locality sensitive hashing (LSH)[26] to speed up the nearest neighbour searching. Feng et al.[2] improved this work using online learning process to construct the hash key, which made the relocalization more robust.

*Active loop closure.* Loop closure is another important problem in SLAM, which can correct the accumulated error of the map and the robot trajectory. Traditional loop closure methods are passive and normally performed by human or on the pre-planned trajectories. However, robots' motions can strongly affect the SLAM performance, which has become a hot topic named active SLAM coined by Feder et al.[27] Active SLAM can be described as controlling the robot's motion to minimize the uncertainty of its map and localization. Our system introduces this idea into the relocalization, which aims to correct the drift and finally achieve higher accuracy and robustness.

## Multi-robot collaborative monocular SLAM

There are many solutions for the single-robot SLAM, but when it moves to multiple robots platforms, another layer of challenges will be introduced. Especially for the monocular SLAM, the estimation of relative poses and the map merging problems are very challenging, as the monocular vision cannot obtain the accurate depth information and the scales of robots' maps are usually different from each other.

*Relative poses estimation and map merging.* One of the most important steps to realize multi-robot collaborative SLAM based on monocular vision is to calculate the relative poses of the robots and then consolidate their local maps into a large-scale global map. According to Rone and Ben-Tzvi,[28] the proposed approaches can be classified into the following four categories.

*Known initial poses.* In this case, the initial poses of the robots are known, and thus the relative poses of the robots can be determined at any time. The assumption of having knowledge of the initial poses limits the application of multi-robot SLAM.

*Rendezvous.* In the rendezvous method,[29] robots meet at a point. At the meeting point, the relative poses can be calculated through the line-of-sight measurements. Once the relative poses are known, maps can be merged. However, this approach will bring another challenge, which is the coordination for the meeting.

*Relative localization.* A more advanced form of the rendezvous is the relative localization. In this method, one robot localizes other robots in its own map[30]; thus, without rendezvous, the relative poses of the robots can be determined.[31]

*Based on overlaps.* In this approach, the overlaps between the maps are used to calculate the relative transformation between the maps and the poses.[32] The challenge of this approach is finding the overlaps; however, this method does not need rendezvous and robots can be out of teammates' maps at any time.

*Multi-robot collaborative monocular SLAM.* Recently, most of multi-robot SLAM algorithms use combinations of various

types of sensors such as laser range finder and IMU.[33–36] Very little work based on monocular vision has been done, since it may cause many problems such as the lack of depth, different scales and scale drift. It first appeared as the multi-robot visual localization problem proposed by Fox et al.[37] and Martinelli et al.[38] or multi-robot mapping problem proposed by Vidal-Calleja et al.[39] They proved that the overall localization accuracy and mapping efficiency of cooperative multi-robot SLAM are much higher than that of single-robot SLAM. Later, Kaess and Dellaert[40] realized a system of multi-camera structure from motion (SfM), while Sola et al.[41] realized a multi-camera SLAM system, both of which are the rudimentary form of multi-robot monocular SLAM. Doitsidis et al.[42] presented a two-step centralized algorithm for surveillance coverage for multiple micro aerial vehicles (MAVs), but the initial poses of the robots are assumed to be known. Forster et al.[43] first proposed a real-time collaborative monocular SLAM algorithm called collaborative SfM (CSfM). In CSfM, each MAV generates a six degree of freedom (DOF) estimation of its own pose using monocular visual odometry. Features of selected keyframes are sent to a centralized ground station where a global map is generated when the overlap between the maps is detected. Based on CSfM, Chebrolu et al.[44] presented a similar framework for multi-robot SLAM, but each robot is capable of performing complete SLAM individually using full image information instead of using only features. In addition, a feedback mechanism is used to correct the local estimates continuously. Similar to them, Schmuck[45] recently proposed a multiple MAVs collaborative monocular SLAM, which employs MAVs as agents to independently explore the environment running limited-memory SLAM onboard.

The proposed robust relocalization method is based on our previous work,[46,47] and we give a more exhaustive explanation in this article. Our work is similar to these works,[1–3] for using the image-to-map relocalization method. However, their works are based on pre-trained classifiers or hash methods, which need extra time to train the classifiers and can only be used in small workplaces or indoor scenarios. Differently, based on a local map and ORB features, our work can be used in outdoor large-scale environments with higher accuracy and efficiency. The local map we used is similar to that of Strasdat et al.[4] and ORB-SLAM,[5] while our method can be applied in unvisited environments. Stachniss et al.[48] and Rahimi et al.[49] also use active loop closure to reduce the drift. Different from them, we here propose to combine the relocalization and active loop closure, which can simultaneously raise the accuracy of the relocalization result and solve the well-known 'when to use' problem in active SLAM. The combination system can overcome the shortcomings of both parts and achieve a good performance as an integrated system.

Based on the proposed robust monocular SLAM, we realize a novel multi-robot collaborative SLAM system. The proposed multi-robot SLAM is different from these works,[37–39,42] because we do not need the initial relative poses of the robots and not rely on the maps to determine the relative poses. We utilize a place recognition method based on image-to-image feature matching to identify similar scenes and then calculate the relative poses. Furthermore, different to the MAV-based server-agent like centralized architectures,[43–45] our system is fully distributed, which has higher reliability and better overall performance when fulfilling an outdoor large-scale SLAM task.

## Robust monocular SLAM

### *The overview of the robust monocular SLAM*

The proposed monocular SLAM uses an active loop closure-based relocalization system to make itself more robust, which can detect and recover from tracking failures automatically even in previously unvisited areas where no keyframe exists. The proposed system is based on the state-of-the-art real-time monocular SLAM, ORB-SLAM,[5] which has a lightweight representation of the environmental map based on sparse feature points. This makes it possible for multiple robots to share their maps via wireless communication. Furthermore, after adding the proposed relocalization module, the improved monocular SLAM can also be very robust and accurate when used in outdoor environments. Figure 1 demonstrates the structure of the proposed robust monocular SLAM, which is composed of five parts: tracking, loop closing, local mapping, relocalization and active loop closure. The first three modules form the paradigm of the current appearance-based monocular SLAM. Following the original ORB-SLAM, the tracking thread localizes the camera in every frame and decides when to insert a new keyframe; the local mapping thread processes new keyframes and maintains the local map; and the loop closing thread searches loops and corrects the accumulated drift. The details of the monocular SLAM implementation are introduced in 'Experiments' section. When the tracking fails, the proposed robust monocular SLAM will employ the relocalization module on each incoming frame, which is explained in detail in 'Relocalization module' section. To solve the drift problem caused by tracking failures, the proposed system can actively create loop closure to reduce the drift through path planning and robot movements, which is explained in 'Active loop closure module' section.

### *Relocalization module*

In our relocalization module, the image-to-map method is used based on local maps. Similar to most image-to-map approaches, we estimate a six-dimensional pose from 2D image to 3D maps, but the difference is that our method only uses the simplest exhaustive nearest neighbour

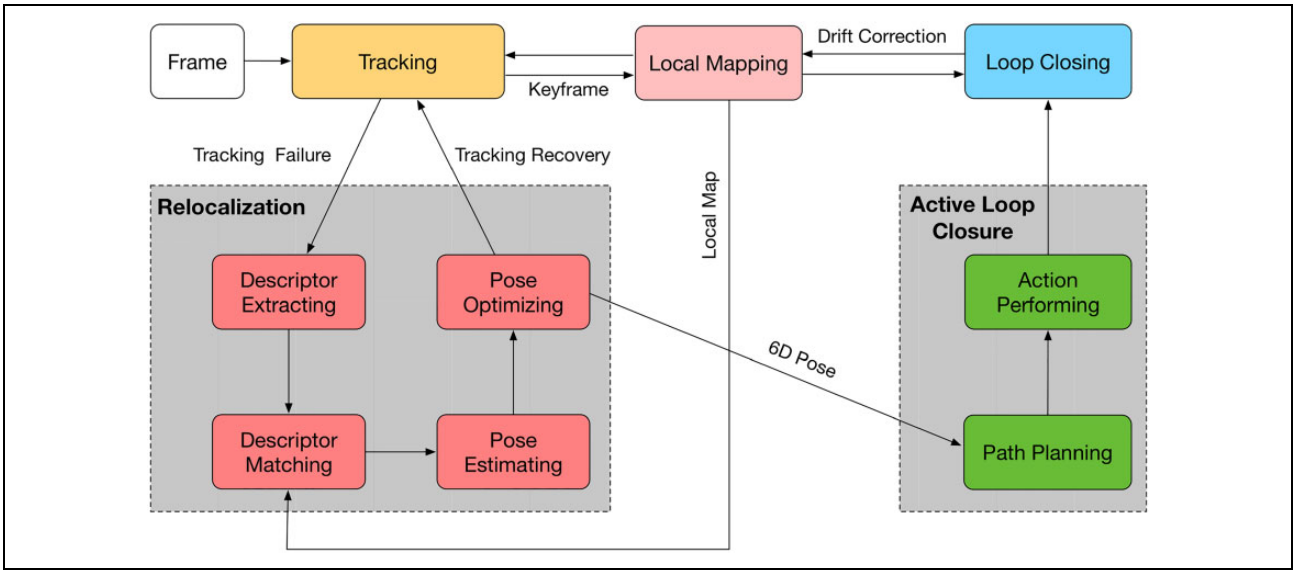


**Figure 1.** The overview of the proposed robust relocalization system based on active loop closure. It includes five main parts: tracking, local mapping, loop closing, relocalization and active loop closure.

searching method to establish the associations between the descriptors extracted from the current frame and the mappoints stored in the local map, while most other methods are classifier-based, which are time-consuming and sometimes inaccurate.

Our method benefits from the ORB features and the local map, as ORB features allow us to directly compute the hamming distance between the respective binary descriptors, very efficiently using low-level hardware operations; and the local map helps us reduce the searching space in an efficient way. Once the tracking fails, the proposed relocalization method will try to recover the tracking process by finding the robot/camera pose in every incoming frame which is summarized in algorithm 1. The steps of the proposed relocalization method are described as follows.

**Algorithm 1.** Relocalization algorithm.

**Require:** Current frame: $f$, local map: $map_l$.
**Ensure:** The updated map and the keyframe of the robot.
1: **procedure** RELOCALIZATION($f, map_l$)
2: **if** $tracking_{lost}$ = *true* **then**
3: $descriptors \leftarrow Extraction(f)$
4: $(matches, N) \leftarrow Matching(descriptors, map_l)$
5: **if** $N > th_N$ **then**
6: $pose_{raw} \leftarrow RANSAC(PnP(matches))$
7: $pose \leftarrow g2o(pose_{raw})$
8: $relocalization \leftarrow true$
9: **end if**
10: **end if**
11: **end procedure**

*Feature extracting.* The first step is to extract ORB features in the current frame (line 3 of algorithm 1). Like ORB-SLAM, we use the image pyramid of eight scale levels with a scale factor of 1.2 and detect FAST keypoints in each level to make the features invariant to the scale in a certain degree. Furthermore, since ORB feature[23] is made up of oriented FAST corner and rotated BRIEF descriptor, it can be extracted very fast and has a certain level of rotational invariance. Therefore, we extract ORB features in every incoming frame and utilize them in every module of our SLAM system, which makes it possible to be used in large-scale environments.

*Descriptor matching.* The second step is to find the matches between the extracted features and mappoints stored in the local map (line 4 of algorithm 1). Mappoints form the 3D reconstruction of the environment. They are defined by two coordinates $X_i = (u_i, v_i)$ and correspond to textured planar patches in the world whose position has been triangulated from multiple views and refined by bundle adjustment. In this work, the mappoints actually correspond to the ORB features in several keyframes. Instead of using complex classifier-based methods, we employ the simplest exhaustive nearest neighbour searching built in FLANN[50] to establish the associations based on the local map, which is fast and accurate. Once the number of matches exceeds a certain threshold $\boldsymbol{th_N}$ (line 5 of algorithm 1), it then enter the next step.

*Pose estimating.* The third step is to estimate the robot/camera pose from the set of 2D to 3D matches by solving a PnP problem using an RANSAC scheme (line 6 of algorithm 1). In practice, we use EPnP[51] to randomly select four non-coplanar points to calculate the camera pose and then use RANSAC to iteratively verify the estimated camera pose.

Once the number of the inliers exceeds a certain threshold, we accept this pose as the new camera pose and recover the SLAM process. Specifically, let the reference points, that is, the $n$ points whose 3D coordinates are known in the world coordinate system, be $\boldsymbol{p}_i, i = 0, ..., n$ and the selected four points be $\boldsymbol{c}_j, j = 1, ..., 4$. Let $\boldsymbol{A}$ be the matrix of camera intrinsic parameters and $\boldsymbol{u}_{i=1,...,n}$ be the 2D projections of $\boldsymbol{p}_i$. Then, we have

$$w_i \begin{bmatrix} \boldsymbol{u_i} \\ 1 \end{bmatrix} = \boldsymbol{A}\boldsymbol{p}_i^{\boldsymbol{c}} = \boldsymbol{A} \sum_{j=1}^{4} \alpha_{ij} \boldsymbol{c}_j^{\boldsymbol{c}}, \forall i \tag{1}$$

where $w_i$ is the vector of scalar projective parameters, and we express each reference point as a weighted sum of the selected points. $\alpha_{ij}$ is the vector of homogeneous barycentric coordinates, and $\sum_{j=1}^{4} \alpha_{ij} = 1$. We now expand this expression by considering the specific 3D coordinate $[x_j^c, y_j^c, z_j^c]^\top$ of each $\boldsymbol{c}_j$, the 2D coordinate $[u_i, v_i]$ of $\boldsymbol{u_i}$, the focal length coefficients $[f_u, f_v]$ and the principal point $[u_{\boldsymbol{c}}, v_{\boldsymbol{c}}]$ that appear in the $\boldsymbol{A}$ matrix. Equation (1) then becomes

$$w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_{\boldsymbol{c}} \\ 0 & f_v & v_{\boldsymbol{c}} \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^{\boldsymbol{c}} \\ y_j^{\boldsymbol{c}} \\ z_j^{\boldsymbol{c}} \end{bmatrix}, \forall i \tag{2}$$

The unknown parameters of this linear system are the 12 control point coordinates and the $n$ projective parameters. The last row of equation (2) implies that $w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^{\boldsymbol{c}}$. Two linear equations for each reference point can be acquired by substituting this expression in the first two rows

$$\sum_{j=1}^{4} \alpha_{ij} f_u x_j^{\boldsymbol{c}} + \alpha_{ij}(u_{\boldsymbol{c}} - u_i) z_j^{\boldsymbol{c}} = 0 \tag{3}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_v y_j^{\boldsymbol{c}} + \alpha_{ij}(v_{\boldsymbol{c}} - v_i) z_j^{\boldsymbol{c}} = 0 \tag{4}$$

Let $\boldsymbol{x} = [c_1^{c\top}, c_2^{c\top}, c_3^{c\top}, c_4^{c\top}]^\top$, which is a 12-dimensional vector made up of the four selected points' coordinates. Arranging the coefficients of equations (3) and (4), we can generate a $2n \times 12$ matrix $\boldsymbol{M}$, and for each reference point, we have a linear system

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{0} \tag{5}$$

The solution, therefore, belongs to the null space or kernel of $\boldsymbol{M}$ and can be expressed as

$$\boldsymbol{x} = \sum_{i=1}^{N} \beta_i \boldsymbol{v}_i \tag{6}$$

where the set $\boldsymbol{v}_i, i = 1, ..., N$, is made up of the columns of the right-singular vectors of $\boldsymbol{M}$ corresponding to the $N$ null singular values of $\boldsymbol{M}$. They can be found efficiently as the null eigenvectors of matrix $\boldsymbol{M}^\top \boldsymbol{M}$, which is of small constant $(12 \times 12)$ size. Given that the solution can be expressed as a linear combination of the null eigenvectors, we can solve this problem by computing the appropriate values for $\beta_i$ and then get four selected points' coordinates. The EPnP method uses the efficient Gauss–Newton method to optimize $\beta_i$, which can efficiently reduce the calculation error. After that, it becomes a question of having the coordinates of a set of points in two coordinate systems and finding the pose transformation of these two coordinate systems.

Let the transformation be

$$\boldsymbol{p}_w^i = \boldsymbol{R}\boldsymbol{p}_{\boldsymbol{c}}^i + \boldsymbol{T}, i = 1, ..., n \tag{7}$$

To calculate the rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$, we first centralize every point to the barycentre

$$\boldsymbol{p}_{wo}^i = \boldsymbol{p}_w^i - \frac{\sum_{i=0}^{N} \boldsymbol{p}_w^i}{N}, \boldsymbol{p}_{\boldsymbol{c}o}^i = \boldsymbol{p}_{\boldsymbol{c}}^i - \frac{\sum_{i=0}^{N} \boldsymbol{p}_{\boldsymbol{c}}^i}{N} \tag{8}$$

Then, we calculate the homography matrix $\boldsymbol{H}$ and solve this problem by singular value decomposition

$$\boldsymbol{H} = \sum_{i=0}^{N} \boldsymbol{p}_{\boldsymbol{c}o}^i \boldsymbol{p}_{wo}^{i\top} = \boldsymbol{U}\Lambda\boldsymbol{V}^\top \tag{9}$$

The rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ can be calculated by

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^\top, \boldsymbol{t} = \boldsymbol{p}_{\boldsymbol{c}}^i - \boldsymbol{R}\boldsymbol{p}_w^i \tag{10}$$

*Pose optimizing.* After estimating a raw camera pose, we use the Levenberg–Marquardt-based bundle adjustment method to refine the pose (line 7 of algorithm 1). In our image-to-map relocalization module, the actual optimization objects are the reprojection errors, and we use the famous optimization library g2o[52] to implement this method. Taking two views as an example, also let the $n$ points whose 3D coordinates are known in the world coordinate system, be $\boldsymbol{p}_i$, $i = 0, ..., n$ and $\boldsymbol{A}$ be the matrix of camera intrinsic parameters. Let $\boldsymbol{u}_{i=1,...,n}$ be the 2D projections of $\boldsymbol{p}_i$ and $\boldsymbol{\xi} = [p, \phi]^\top \in \mathbb{R}^6$ an element of $\mathfrak{se}(3)$ be the camera model, where $p \in \mathbb{R}^3$ represents the translation and $\phi \in \mathbb{R}^3, \mathfrak{so}(3)$ represents the rotation. We define $\xi^\wedge$ as

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \phi^\wedge & p \\ \boldsymbol{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{11}$$

where $\phi^\wedge \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix of $\phi$. Then, we have

$$w_i \boldsymbol{u}_i = \boldsymbol{A} \exp(\xi^\wedge) \boldsymbol{p}_i, \forall i \tag{12}$$

where $w_i$ is the scalar projective parameters. In linear optimization, people usually calculate the camera pose first and then use the mappoints to refine the pose. However, in the actual implementation of the nonlinear optimization, we set

**Algorithm 2.** Active loop closure algorithm.

**Require:** Current frame: $f$, Pose calculated by the relocalization module: $pose$, Pose before the tracking failure: $pose^*$.
**Ensure:** The updated map and the keyframe of the robot.
1: **procedure** ACTIVE LOOP CLOSURE ($pose, pose^*$)
2: **if** *relocalization* = *true* **then**
3: $direction \leftarrow Calculate(pose, pose^*)$
4: **while** $loop_{closing}$ = *true* **do**
5: $path \leftarrow Planning(direction)$
6: $Action(path)$
7: **end while**
8: **end if**
9: **end procedure**

both of them the optimization variables and optimize them at the same time. Since there are noises in the measurements, we can calculate the reprojection errors as

$$\boldsymbol{e_i} = \boldsymbol{u_i} - \frac{1}{w_i}\boldsymbol{A}\exp(\xi^\wedge)\boldsymbol{p}_i, \forall i \tag{13}$$

Thus, we sum the reprojection errors and build a least square problem

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} \| \boldsymbol{u_i} - \frac{1}{w_i}A\exp(\boldsymbol{\xi}^\wedge)\boldsymbol{p}_i \|_2^2 \tag{14}$$

By minimizing the sum of reprojection errors, we can find a more accurate camera pose $\boldsymbol{\xi}^*$ to recover the tracking process.

### Active loop closure module

As discussed above, fast and long-distance tracking recovery can be achieved using the proposed relocalization module. However, because of the information missing during the tracking failure, it is inevitable to have a larger drift after the relocalization. We, therefore, employ an active loop closure module to reduce the drift after the relocalization and acquire more accurate maps, which is summarized in algorithm 2. Once relocalizing successfully, the system will enable the active loop closure module which includes three steps.

*Direction computing.* We first calculate the robots movement direction (line 3 of algorithm 2). After the relocalization, we get an optimized *pose* and a camera $pose^*$ stored in the last keyframe before the tracking failure. Their absolute transformations in SE(3) are

$$pose : \boldsymbol{T}_w = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}, \; pose^* : \boldsymbol{T}_w^* = \begin{bmatrix} \boldsymbol{R}^* & \boldsymbol{t}^* \\ 0 & 1 \end{bmatrix} \tag{15}$$

where for simplicity, we assume the scale factor to be 1, which does not affect the calculation of the direction. We then compute the relative transformation $\Delta\mathbf{T}$ between the current *pose* and the last $pose^*$

$$\Delta\boldsymbol{T} = \begin{bmatrix} \Delta\boldsymbol{R} & \Delta\boldsymbol{t} \\ 0 & 1 \end{bmatrix} = \boldsymbol{T}_w^{*-1} \cdot \boldsymbol{T}_w \tag{16}$$

where $\Delta\boldsymbol{R}$ and $\Delta\boldsymbol{t}$ are, respectively, the differences of the robots rotation and translation between the current *pose* and the last $pose^*$.

In this article, we only consider the case that robots conduct the relocalization when travelling forwards into unvisited environments, so the camera's orientation is almost constant. We, therefore, only take the translation into consideration and compute the direction as follows

$$direction : \boldsymbol{d} = \Delta\boldsymbol{t} \tag{17}$$

*Path planning.* The second step is to plan a path using the computed direction to realize a loop closure (line 5 of algorithm 2). In fact, any path-planning algorithm which can find a loop between the current *pose* and the last $pose^*$ based on the calculated direction can be used in our system. It is worth mentioning that only loop closure can eliminate the accumulated error and the scale drift. Other path-planning algorithms cannot deal with this situation even thought they can find paths and re-collect the missing environment information. For example, it is not a best solution for the robot just moving backwards a few meters to revisit previously seen areas, because the mappoints and the camera poses stored in each node/keyframe are consistent. Here, we give a simple example of using the calculated direction and the tracking failure distance to realize the active loop closure. Assuming the robot moving forwards into unvisited environments with a constant speed $V_{robot}$ during the tracking failure time $T_{failure}$, we can compute the tracking failure distance $D_{failure}$ as follows

$$D_{failure} = V_{robot} \times T_{failure} \tag{18}$$

Taking $D_{failure}$ as the radius and the computed direction as the initial normal vector, we can plan a semicircle path for the robot to realize loop closure.

*Action performing.* The third step is to control the robot moving along the planned path (line 6 of algorithm 2). The active loop closure module will be performed by repeating steps 2 and 3 until a loop closure is detected successfully.

## Multi-robot collaborative SLAM

Multi-robot SLAM is motivated by the fact that exploration and mapping tasks can be done faster and more accurately by multiple robots than by a single robot. In addition, in a distributed system, the whole team is more robust since the failure of one of the robots does not halt the entire mission.[53] Based on the proposed robust monocular SLAM, we realize a novel distributed multi-robot collaborative SLAM system, which can employ a team of robots to map a large-scale unknown environment.

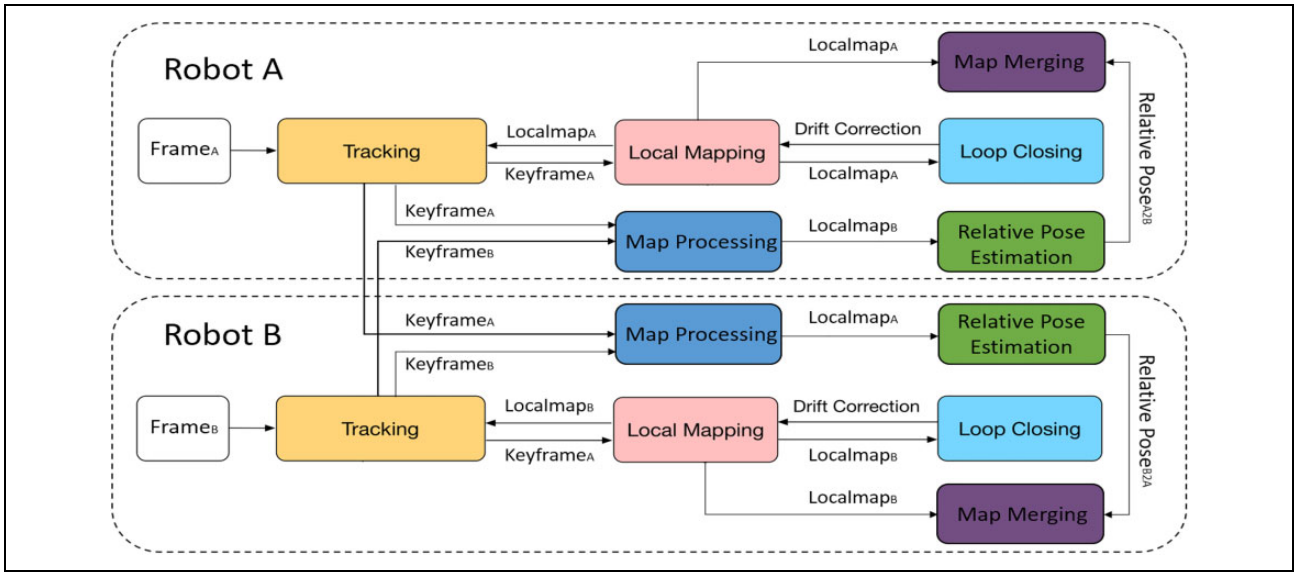


**Figure 2.** Taking two robots as an example, each robot conducts the same monocular SLAM, in which we add another three new modules, map processing, relative pose estimation and map merging, to realize a distributed multi-robot collaborative SLAM. SLAM: simultaneous localization and mapping.

### *The overview of the multi-robot collaborative SLAM*

In this distributed multi-robot collaborative SLAM, we mainly solve two problems. One is the message format of real-time multi-robot communication, and another is the problem of relative poses estimation and map merging. For the communication issue, we utilize bridges with robot operating system (ROS) to realize a real-time AP-to-AP wireless multi-robot communication. We use the ROS *message* to design our own map representation, through which the robots can incrementally share their maps among the group. The details of multi-robot communication implementation are introduced in 'Experiments' section. For the problem of estimating unknown relative poses, we utilize the same method that is used for loop detection of single-robot SLAM, which is exactly a kind of place recognition method. Based on the proposed robust monocular SLAM, we add another three modules, map processing, relative pose estimation and map merging, to realize this multi-robot collaborative SLAM, as shown in Figure 2. Each robot conducts the same monocular SLAM, meanwhile sharing their incremental maps with each other via wireless network and ROS messages. The robots in the team use map processing thread to maintain the maps received from other robots. Once a robot creates a new keyframe, it will traverse these maps to find whether it is in the same places where other robots have visited. Once the similarity between the current keyframe and another keyframe stored in those maps is higher than a certain threshold, it then confirms they are describing the same place. Subsequently, the robot will use the relative pose estimation thread with these two keyframes to calculate the relative pose and combine the maps together. The map processing module is described in 'Map processing module' section, the relative pose estimation module is explained in 'Relative pose estimation' section and the multi-robot map merging module is explained in 'Map merging' section.

### *Map processing module*

For simplicity, we assume that there are two robots involved in this cooperative SLAM task, $robot_A$ and $robot_B$ (see Figure 2). When $robot_A$ receives a keyframe, called $keyframe_B$, if it is the first keyframe received from $robot_B$, $robot_A$ will create a new map structure $map_B$ and add this keyframe to the map as the start point. Otherwise, $robot_A$ will add a new node for $keyframe_B$ in $map_B$ and update the edges resulting from the shared mappoints with other previous keyframes. Then, we use the same local bundle adjustment, which has been described in 'Relocalization module' section, to optimize the current $keyframe_B$, all the keyframes connected to it in the covisibility graph and all the map points seen by those keyframes. In order to realize a real-time and life-long multi-robot collaborative SLAM system, we must reduce the amount of the maps' calculation and storage. Therefore, we will not maintain all the messages received from other robots. The map processing module will try to detect the redundant keyframes and mappoints and delete them. We discard all the keyframes where 90% of the map points have been seen in at least other three keyframes in the same or finer scale and discard the mappoints if less than three keyframes have observed them.

### *Relative pose estimation*

From the perspective of $robot_A$, after it has received keyframes from other robots and built the map structures, it will try to match its every incoming keyframe with the

**Algorithm 3.** Relative pose estimation algorithm.

**Require:** Current frame: $keyframe_A(K_A)$, Maps of $robot_A$ and $robot_B$: $map_A(M_A)$ and $map_B(M_B)$.
**Ensure:** The updated map and the keyframe of the $robot_A$.
1: **procedure** RELATIVE POSE ESTIMATION ($K_A$, $M_A$, $M_B$ )
2: $K_B^* \leftarrow \arg\min_{K_B \in M_B} BoW(K_A, M_B)$
3: $(\lambda_{AB}, N_{inliers}) \leftarrow SIM3(K_B^*, K_A)$
4: **if** $N_{inliers} \geq th_N$ **then**
5: $Merge(M_A, M_B)$
6: $g2o(M_A)$
7: **end if**
8: **end procedure**

keyframe databases of all other robots' maps. The keyframe matching is done using the place recognition method which can recognize the same place that has been observed by other robots. In this work, we use a specific BoW method, DBoW2,[54] which uses the bags of binary words to realize fast place recognition in image sequences. Algorithm 3 summarizes the proposed method which is conducted to determine the relative poses and merge the maps.

For the first step, we calculate the visual word of the current $keyframe_A$ and try to find the candidates of the relative pose estimation in other robots' maps using DBoW2 (line 2 of algorithm 3). In DBoW2, a vocabulary tree is built by discretizing the binary descriptor space to speed up the correspondences for geometrical verification. We traverse all the keyframes in each map comparing their visual words to those of $keyframe_A$ and select the best match, which can be described as follows

$$K_B^* = \underset{K_B \in M_B}{\arg\min}\ \mathrm{BoW}(K_A, M_B) \tag{19}$$

where $BoW(K_A, M_B)$ means searching the match candidates of $keyframe_A$ in $map_B$.

Wrong place match hypotheses can lead to serious failures to the multi-robot collaborative SLAM. In practice, we employ a spatial continuity check to provide a constraint, which requires several continuous matches to find stronger matching hypotheses. Empirically, we find that it is suitable to confirm a candidate of relative pose estimation after three continuous matches, which can not only raise the robustness of the whole SLAM system against the wrong matches but also not too strict to find a candidate.

Besides the continuous matches check, we also verify each candidate in the geometry level. We compute the transformation from the current keyframe camera coordinate system to the loop candidate one. Using this verification, we can compute the relative scale factors for each robot and correct the drift of accumulated error. Since the monocular SLAM has no absolute scale and the scale factors in each robot's SLAM process are different from others, we use the geometry verification to uniform the scale factors, which is the core of the proposed relative pose estimation algorithm. We compute a similarity transformation, Sim(3),[55] from the current keyframe $K_A$ to loop candidate $K_B$ (line 3 of algorithm 3)

$$\boldsymbol{S}_{K_A,K_B} = \begin{bmatrix} \lambda_{K_A,K_B}\boldsymbol{R}_{K_A,K_B} & \boldsymbol{t}_{K_A,K_B} \\ 0 & 1 \end{bmatrix} \tag{20}$$

where $\lambda_{K_A,K_B} \in \mathbb{R}^+$ is the scale factor, $\boldsymbol{R}_{K_A,K_B} \in \mathrm{SO}(3)$ is a rotation matrix and $\boldsymbol{t}_{K_A,K_B} \in \mathbb{R}^3$ is a translation vector. If it is successful, we continue to calculate the relative pose; otherwise, we reject this candidate.

To compute the transformation, we also need to solve a PnP problem. Different from the method used for the *relocalization module*, in the relative pose estimation module, we use an image-to-image method with an RANSAC scheme to calculate the transformation. We first compute correspondences between ORB features associated with the mappoints connected to the current keyframe $K_A$ and the candidate $K_B$. Then, we have 3D to 3D correspondences for each candidate and can directly use the RANSAC scheme to solve a P3P problem, which is very fast and has a certain degree of rotational invariance. Once the number of inliers is beyond a certain threshold (line 4 of algorithm 3), we stop the RANSAC process and enter the next step.

## Map merging

Based on the relative pose estimation, we can then combine the maps of multiple robots (line 5 of algorithm 3). First, we convert all keyframes' poses from SE(3) absolute transformation $\boldsymbol{T}_{i,A}$ into a similarity Sim(3), $\boldsymbol{S}_{i,A}$, maintaining the rotation and translation and setting the scale to be 1. Then, we compute the transformation between one keyframe and the next keyframe, merging the map with the relative pose $\boldsymbol{S}_{K_A,K_B}$ calculated in last section. Next, we use a global bundle adjustment, to minimize the residual error $\boldsymbol{r}_{i,j}$ between the keyframe $i$ with its pose $\boldsymbol{S}_{i,A}$ and keyframe $j$ with its pose $\boldsymbol{S}_{j,A}$. The relative transformation between $\boldsymbol{S}_{i,A}$ and $\boldsymbol{S}_{j,A}$ is $\Delta\boldsymbol{S}_{i,j}$. We use it as the constraint, and the optimization process in the tangent space $\mathfrak{sim}(3)$ can be represented as follows

$$\boldsymbol{r}_{i,j} = (\log_{\mathrm{Sim}(3)}(\Delta\boldsymbol{S}_{i,j}\cdot)\boldsymbol{S}_{j,A}\cdot\boldsymbol{S}_{i,A}^{-1}))^{\vee}_{\mathfrak{sim}(3)} \tag{21}$$

where $\log_{\mathrm{Sim}(3)} : \mathrm{Sim}(3) \to \mathfrak{sim}(3)$ maps from the over-parameterized representation of the transformation to the tangent space, and $(\cdot)^{\vee}_{\mathfrak{sim}(3)} : \mathfrak{sim}(3) \to \mathbb{R}^7$ is the vee-operator that maps from the tangent space to the minimal representation with the same elements as the DOFs of the transformation.[56]

Initially, all the residuals are zero, except for the map junction part. We then optimize the poses of keyframes to distribute this error along the graph. The cost function to minimize is defined as follows

$$\chi^2 = \sum_{i,j} \boldsymbol{r}_{i,j}^{\top}\Lambda_{i,j}\boldsymbol{r}_{i,j} \tag{22}$$

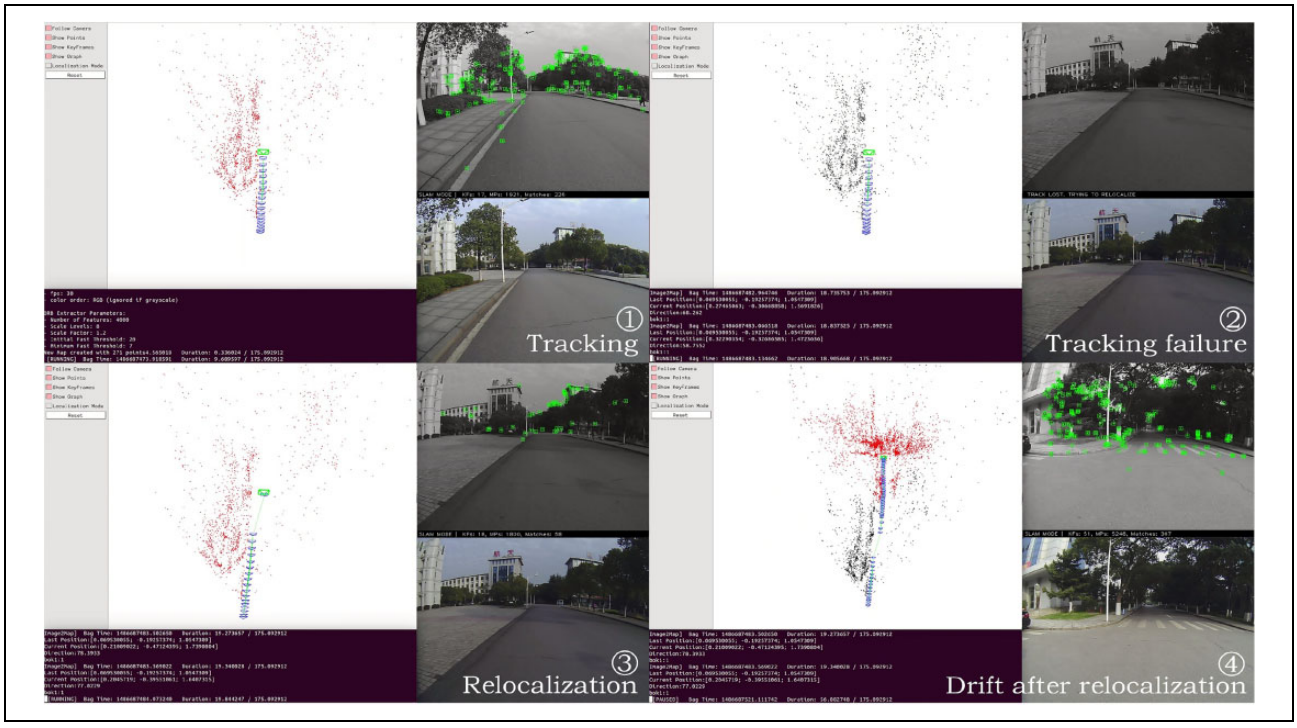


**Figure 3.** The proposed relocalization system dealing with tracking failure caused by image interruption. While conducting SLAM ①, we introduced a tracking failure manually by suspending the tracking thread and keeping on moving for 15 s (25 m). After that, we resumed the tracking thread and the tracking failure happened ②. The proposed system enables the relocalization module automatically and recovered the tracking successfully ③, where a pose was computed and a keyframe was inserted. However, due to the missing data, the scale drift became larger (the space between keyframes became smaller) and the accuracy of pose estimation decreased greatly (there was an obvious deviation between the newly computed camera pose and the previous trajectory) ④. SLAM: simultaneous localization and mapping.

where $\Lambda_{i,j}$ is the inverse covariance of the residual $\boldsymbol{r}_{i,j}$ and is set to be the identity. We exclude the loop keyframe pose from the optimization to fix the seven DOFs of the solution. We also use the Levenberg–Marquardt method implemented in the graph optimizer g2o to solve this problem (line 6 of algorithm 3).

After poses have been optimized, we need to correct the 3D points associated with them. For each point $x_j$, we select a source keyframe $\boldsymbol{T}_{i,A}$ and map the point using the optimized $\boldsymbol{S}_{i,A}^{cor}$ as follows

$$x_j^{cor} = (\boldsymbol{S}_{i,A}^{cor})^{-1} \cdot \boldsymbol{T}_{i,A} \cdot x_j \tag{23}$$

The last step is to convert the corrected similarity transformations $\boldsymbol{S}_{i,A}^{cor}$ back to 3D rigid body transformations $\boldsymbol{T}_{i,A}^{cor}$. Map point scale has been corrected in equation (23), so we just get rid of the scale factor in the similarity and maintain the rotation as it is not affected by the scale. However, the translation is computed for the scale factor of the similarity and we need to rescale it

$$\boldsymbol{S}_{i,A}^{cor} = \begin{bmatrix} \lambda \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \rightarrow \boldsymbol{T}_{i,A}^{cor} = \begin{bmatrix} \boldsymbol{R} & \dfrac{\boldsymbol{t}}{\lambda} \\ 0 & 1 \end{bmatrix} \tag{24}$$

The robots' maps can be merged into a global map, so the multi-robot collaborative SLAM is realized.

## Experiments

We test our system using KITTI data set[57] and our own data set acquired by a handheld camera in outdoor large-scale and indoor small-scale real-world environments where man-made shakes and interruptions were added. For KITTI data set, we use a part of its 3D visual odometry/SLAM data set, which contains six sequences with a total length of 14.6 km and GPS-IMU-derived ground truth. Rectified images are provided in each sequence with 10 Hz and the resolution of $1240 \times 376$ pixels. These sequences are recorded in real-world large-scale driving situations along urban, residual and countryside roads, with driving speeds up to 80 km/h. KITTI data set is very challenging because of its low frame rate, fast driving speed, various scenarios and outdoor large-scale environments, so we can test the proposed method comprehensively. Our experiments can be divided into two categories. One is conducted to test the robustness of the proposed monocular SLAM, and the other is to realize a distributed multi-robot collaborative SLAM based on this proposed robust monocular SLAM. All the experiments were conducted on laptop computers equipped with a 2.4 GHz i7 CPU and 4 GB memory. The bridges we used are LF-P681, which follow the IEEE 802.11 b/g/n/ac 5.8 GHz standard and can realize long-distance communication up to 8 km. The cameras we used are

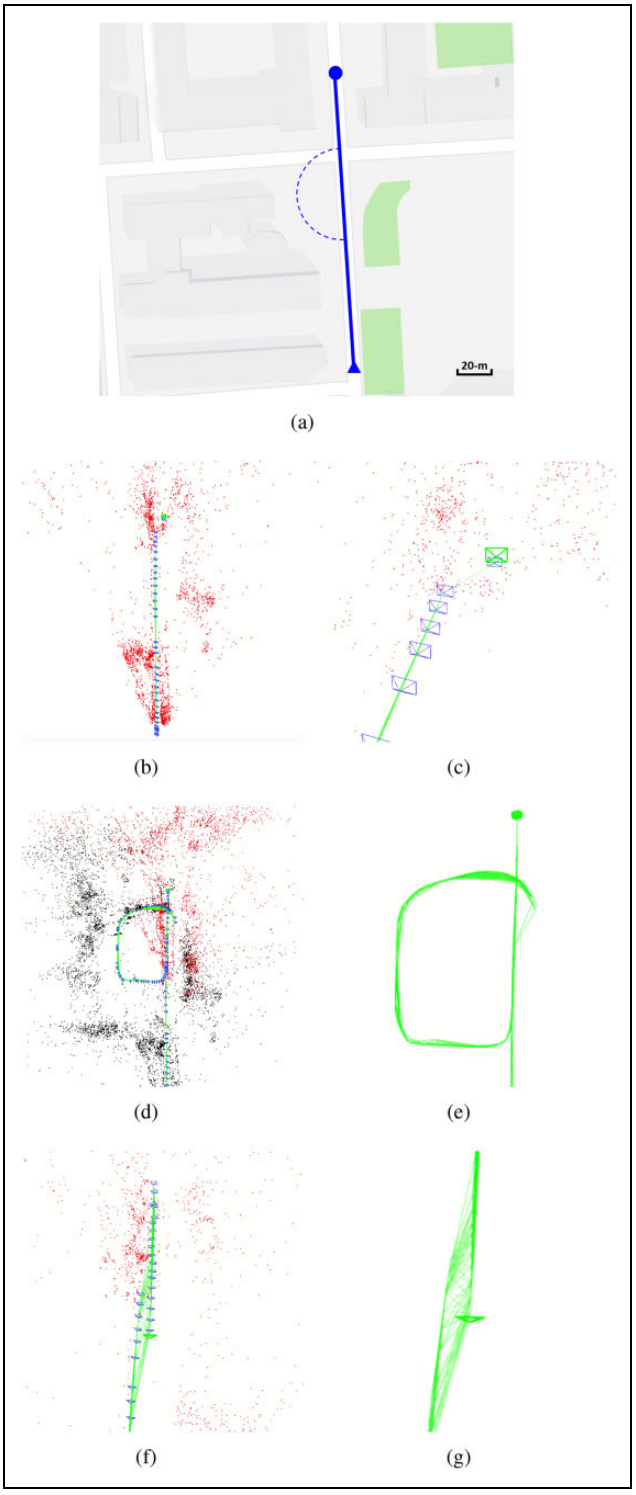


**Figure 4.** Drift correction results by active loop closure. (a) The corresponding Google map of this outdoor environment. (b) The mapping results of the monocular SLAM after the relocalization, where the black points represent the fixed mappoints, red points represent the mappoints which are currently seen by the robot, blue boxes represent the keyframes and the green line represents the trajectories of the robot. After the relocalization (b), we can see an error of the camera's pose estimation, zoomed in (c), where there is an obvious deviation between the newly computed camera pose and the previous trajectory. Then, we compared active loop closure (d) (zoomed in (e)) and moving backwards a few meters (f) (zoomed in (g)). We can see that the trajectory of active loop closure shown in (e) is straight and corrected without any deviation, while in the trajectory of just moving backwards a few meters shown in (g), the deviation still exists. SLAM: simultaneous localization and mapping.

ELP-USBFHD01 M, which can provide HD images in MJPEG at 30 fps.

## Robust monocular SLAM

In the real application of monocular SLAM, the localization or the robot pose tracking is easy to fail, because not enough correspondences of the landmarks can be extracted between the current frame and that stored in keyframes or maps. This situation is usually caused by the image interruption, image blur and sudden motion of cameras. In this section, we conducted several experiments using handheld cameras and KITTI data set to simulate these situations and validated whether the proposed robust monocular SLAM could deal with them or not. We used a handheld camera to simulate the robot, which is actually more challenging because of strong shakes and high speeds. The real-time performance was also tested and compared with other relocalization methods.

*Monocular SLAM implementation.* The state-of-the-art appearance-based monocular SLAM normally includes three main threads: tracking, local mapping and loop closing. The underlying SLAM system used in this article is based on ORB-SLAM with the following differences:

*The tracking.* The tracking thread contains five steps: ORB features extraction, tracking initialization, pose tracking, relocalization and new keyframe decision. The resolution of the image used is $640 \times 480$; 2000 corners for indoor environments, and 4000 corners for outdoor environments are extracted. The original relocalization step is a kind of image-to-image method, which can only be used in the previously visited places. Therefore, it is replaced with an image-to-map relocalization module (see details in section 'Relocalization module'), which enables the monocular SLAM to detect and recover from tracking failures automatically even in previously unvisited areas where no keyframe exists.

*The local mapping.* The local mapping thread also contains five steps, including inserting keyframes, selecting mappoints, creating new mappoints, local bundle adjustment and selecting local keyframes. After these five steps, the local mapping thread will maintain a local map, which in fact consists of two sets of keyframes K1 and K2. The keyframes in K1 share the same mappoints with the current frame, while the keyframes in K2 are the neighbours of those in K1. Different to tracking the whole local map,

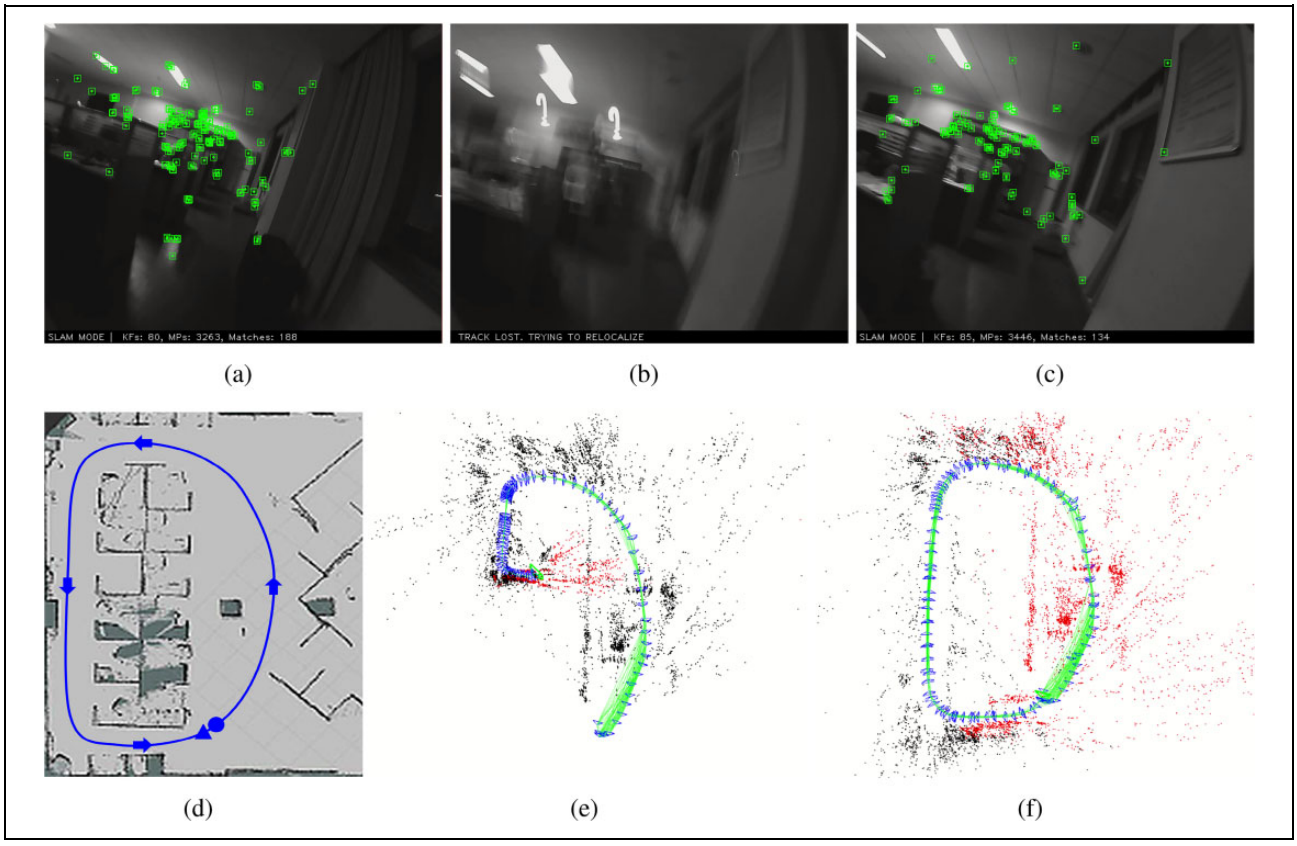


**Figure 5.** The SLAM results with strong shakes caused by camera's high-speed motion. While conducting SLAM (a), tracking failure happened due to the strong shake caused by camera's high-speed motion (b). Our system can rapidly recover the tracking (c), which illustrates that our system has strong robustness even in such challenging situations. (d) The corresponding LiDAR map of the office environment, where the triangle represents the start points, the circle represents the endpoints and the arrows represent the robot's moving directions. (e) and (f) The mapping results of the monocular SLAM, where the black points represent the fixed mappoints, red points represent the mappoints which are currently seen by the robot, blue boxes represent the keyframes and the green line represents the trajectories of the robot. (e) The result without loop closure. (f) The result with a loop closure. The better performance shown in (f) verifies again that our system can perform better after integrating active loop closure module. SLAM: simultaneous localization and mapping.

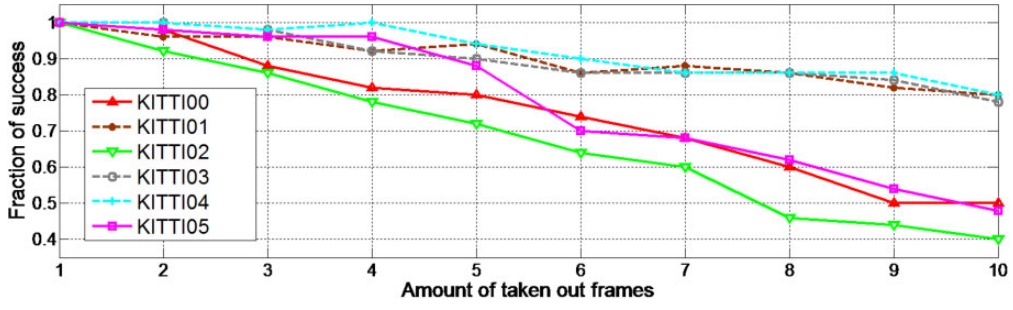


**Figure 6.** Successful rates of the relocalization on the KITTI data set. The solid lines represent the results in large-scale environments with a lot of bends, while the dashed lines represent the results in the environments without many bends.

we only employ K1 as the map for our image-to-map relocalization module, which can reduce the searching time and will not affect the accuracy and success rate.

*The loop closing.* The loop closing module is made up of four steps, including loop candidate detection, similar transformation computing, loop fusion and essential map optimization. Since the proposed active loop closure module will try to detect loops within every incoming frame after the relocalization, it is vital to have a good keyframe selection policy, which should not ignore any potential

match but also not be too open to raise the computation burden. This policy checks four conditions: (1) more than 20 frames must have passed after the latest relocalization, (2) at least 50 points are tracked in the current frame, (3) less than 90% mappoints are tracked in the current frame in comparison to last keyframe and (4) at least 10 frames have passed from last keyframe insertion; local mapping has processed the last keyframe; and a signal is sent to local mapping to finish local bundle adjustment.

To keep the localization accuracy, we use a more strict policy which requires the tracked points doubled from 50 to 100, since we increase the number of extracted features. Meanwhile, we raise the similarity rate from 90% to 95% to keep the same inserting threshold, which in fact makes it easier to realize the active loop closure.

*Dealing with tracking failure caused by image interruption.* The first experiment was conducted in an outdoor campus road. As illustrated in Figure 3, we introduced a tracking failure manually by suspending the tracking thread and keeping on moving for a distance. After that, we resumed the tracking thread and validated whether the proposed monocular SLAM could recover the SLAM process or not. What we want to simulate in this experiment is the tracking failure caused by the image interruption, which can normally interrupt the SLAM from feature extraction for a while. Combining Figure 3-③ and Figure 3-④, we can see that the tracking can be recovered using the proposed system after a long distance failure (about 25 m with 1.67 m/s walking speed and 15 s interruption) in previously unvisited environments where no keyframe exists. After the relocalization, robots can continue the SLAM process. However, we can also see that we conducted the SLAM on a straight campus road (the reference Google map shown in Figure 4(a)), but there is a huge bias in the trajectory shown in Figure 3-④. It is because even though our algorithm can recover the SLAM process quickly, the data missing during the tracking failure is irreversible, which will cause the drift becoming larger and the accuracy of pose estimation decreasing greatly.

To solve the accumulated error and scale drift problem, we proposed an active loop closure method, which has been described in the 'Active loop closure module' section. Our system realized the active loop closure using the pose information computed by the relocalization module, and the robot can find a loop actively to correct the drift. We perform the active loop closure module after each relocalization and stop it when finding a loop closure. As a comparison, we also performed the experiment of moving the robot backwards a few meters to revisit missing areas after the relocalization. Comparing the experimental results of these two methods (shown in Figure 4(d) and (f)), we can conclude that active loop closure performs much better in correcting the drift than just moving backwards a few meters.

**Table 1.** Results on the KITTI data set.

| Sequence | Length of trajectory (km) | Maximal taken out frames (distance, m)[a] |
|---|---|---|
| KITTI00 | 3.7 | 61 (29) |
| KITTI01 | 2.6 | 90 (240)[b] |
| KITTI02 | 5.1 | 28 (37) |
| KITTI03 | 0.4 | 46 (38) |
| KITTI04 | 0.6 | 31 (46) |
| KITTI05 | 2.2 | 104 (9)[c] |

[a]The distance means the length of the trajectory during the tracking failures computed from ground truth.
[b]In sequence 01, the car runs on a highway environment with small changes, which is not a common situation.
[c]In sequence 05, since the car runs very slowly, it moves only 9 m during the taken out 104 frames.

**Table 2.** The computation time needed in relocalization process.

| | Different methods (number of features, ms)[a] | | | | |
|---|---|---|---|---|---|
| Steps | Ours (2000) | Ours (4000) | ORB-SLAM (1000)[5] | Williams (145)[1] | LSH-based (216)[3] |
| Feature extraction[b] | 18.2 | 29.0 | 14.4 | 14.0 | 122.9 |
| Matching | 2.1 | 4.7 | 6.4 | 0.3 | 50.0 |
| Pose estimation | 2.1 | 2.5 | 14.9 | 0.7 | 10.0 |
| Pose optimization | 0.8 | 1.3 | / | 4.0 | 4.0 |
| Total | 23.2 | 37.5 | 35.7 | 19.0 | 183.4 |

LSH: locality sensitive hashing.
[a]We quoted the data of other methods from their publications.[1,3,5]
[b]The time consumed by the step of feature extraction includes the time of feature classification.

*Dealing with tracking failure caused by image blur.* In this experiment, we moved the camera at a high speed and shook it greatly to make the tracking fail during the SLAM process in an indoor office environment to validate that whether the proposed monocular SLAM is robust to the tracking failures caused by the image blur. The experimental results (in Figure 5) show that after combining the relocalization module and active loop closure module, the proposed system works well even in such challenging situation, while the original ORB-SLAM cannot deal with it.

*Dealing with tracking failure caused by camera sudden motion.* In this experiment, we used the image sequence from 00 to 05 from the KITTI data set to test the proposed system. To simulate the tracking failures caused by sudden motion of camera, we randomly chose 50 positions in each sequence, took out image frames close to them and then set the tracking failure manually (by setting the flag tracking lost = true). We tested the relocalization performance on each position by taking out different numbers of frames from 1 to 10. The test results (shown in Figure 6 and Table 1) illustrate that our system can be employed on high-speed

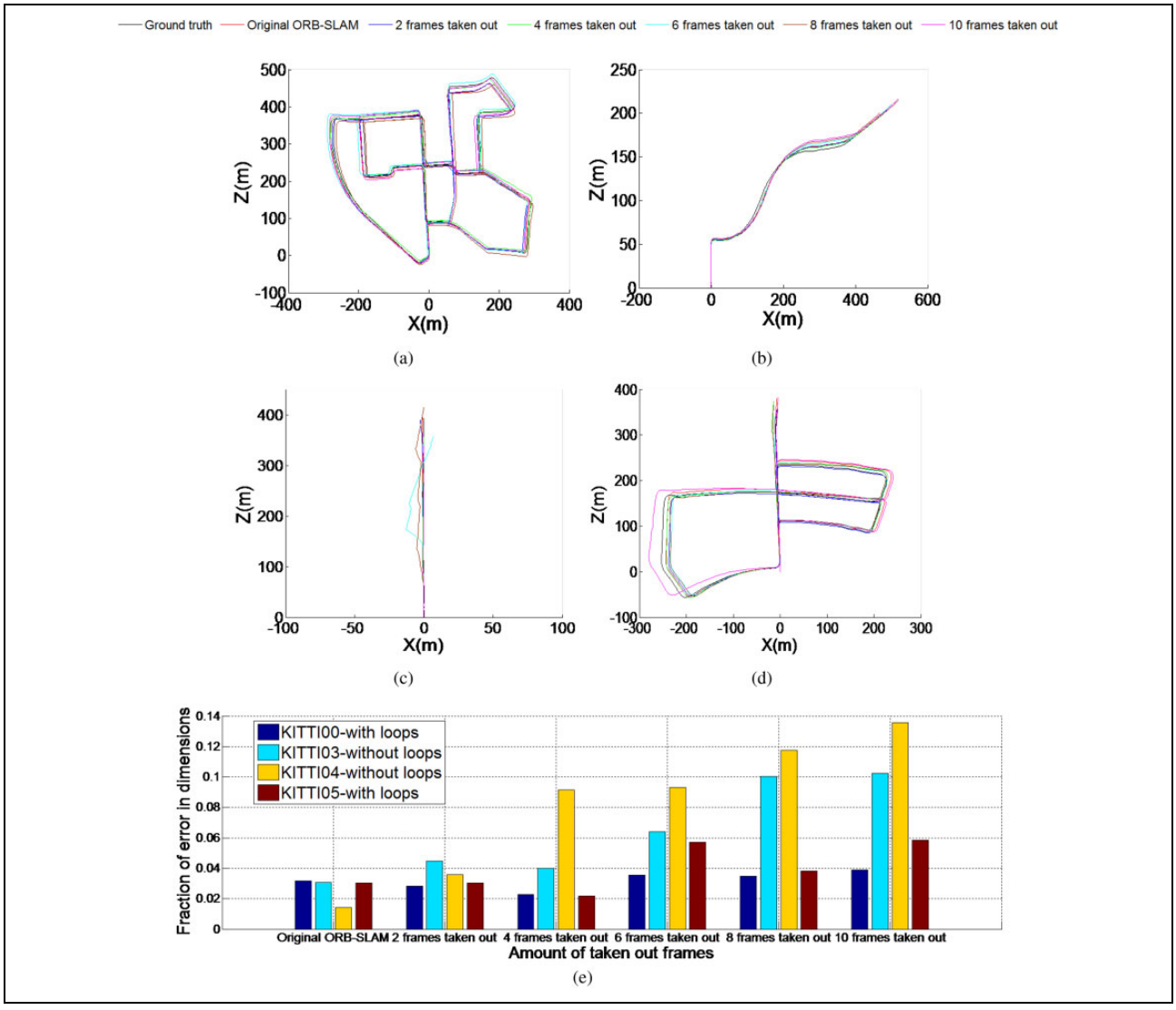


**Figure 7.** (a) to (d) The experimental results of our system conducted on the KITTI sequences 00, 03, 04 and 05 when different numbers of frames from 1 to 10 were taken out, respectively. (e) The comparison of the fraction of their errors (the estimated position error comparing to the given ground truth) in the context of space scales.

cars in large-scale environments (with the total length of 14.6 km) with longest success distance (up to 46 m) and high successful rate of the relocalization (higher than 40%).

*Real-time performance.* We evaluated the real-time performance of the proposed system and other systems by comparing the computation time needed in each step of the relocalization process. As shown in Table 2, our system (23.2 ms) is faster than those in LSH-based method[3] (183.4 ms) and ORB-SLAM[5] (35.7 ms) and is slightly slower than that in Williams et al.'s work.[1] Furthermore, considering that 2000 features were extracted in our system while only 145 features were extracted in Williams et al.'s work, the real-time performance of our system is quite good. Apart from feature extraction, our system only needs 5.0 ms (the same as Williams et al.'s work) for the relocalization, which is the best among all relocalization methods.

*Precision analysis.* Following the original ORB-SLAM, we used the KITTI data set with the relative position error (putting errors in context of space scales) comparing to the given ground truth to evaluate the relocalization accuracy of the proposed system, as shown in Figure 7. We can see that the errors on the KITTI sequences 00 and 05 are obviously smaller than those on the sequences 03 and 04, because there are many loops in the former two sequences. We are surprised to find that with loop closure, our system has the same accuracy (within 6%) as the original ORB-SLAM, even when hundreds of frames were taken out from the image sequence. It verifies again that by combining active loop closure, our system can achieve more robust

and accurate relocalization for the monocular visual SLAM.

### Multi-robot collaborative SLAM.

In this section, two data sets were used: indoor office data set collected by ourselves using handheld cameras and KITTI data set. We used two ELP-USBFHD01 M monocular cameras to perform SLAM simultaneously in each half of the office environment. For the KITTI data set, although it is usually used for single-robot SLAM, concerning the multi-robot mission, we divided the image sequences into several smaller segments and simulated the multi-robot collaborative SLAM by performing SLAM within each segment simultaneously. Map merging cannot be realized with these segments only using overlap-based methods.

### Multi-robot communication

Sharing data among robots is a key requirement in multi-robot collaborative SLAM.[58] In this work, we use bridges with ROS to realize a real-time AP-to-AP wireless multi-robot communication. The bridges have many advantages, such as small size and strong penetration, which enable them to achieve long-distance communication and can be used on mobile robots with limited loading capacity. For the format of the shared information, there are two main approaches, sharing raw sensor data[59] and processed data.[32] Sharing raw sensor data is more flexible but requires high bandwidth and reliable communication between robots as well as more processing power. In contrast, sharing maps uses less bandwidth and there is no need to process raw data redundantly. Therefore, we choose to share maps among the group of robots. Considering that the proposed robust monocular SLAM is based on keyframes, we use the basic ROS message elements to design a new map representation, which contains all the information of a keyframe and the relative mappoints. Each robot in the team conducts the same monocular SLAM and, at the same time, shares this kind of message through the wireless network once they create a new keyframe. The communication of multiple robots is shown in Figure 8. From the perspective of $robot_A$, it receives the environment information from $Frame_A$ and other robots' incremental maps, and meanwhile, sharing its own incremental map through $keyframe_A$. As a distributed system, all the robots in the proposed collaborative SLAM system are conducting the same process and sharing the same format of map information. Therefore, every robot can have input topics for all robots in the network, or just communicate with a subset of the robot team. It depends on the number of the robots and the trade-off between the bandwidth and efficiency. We can set the communication topological relations using the ROS topics before each mission according to the demand. We only give an example of realizing the proposed distributed framework using ROS system. However, it does not rely on a specific communication mechanism and, of course, can also be realized on other systems like ROS2.

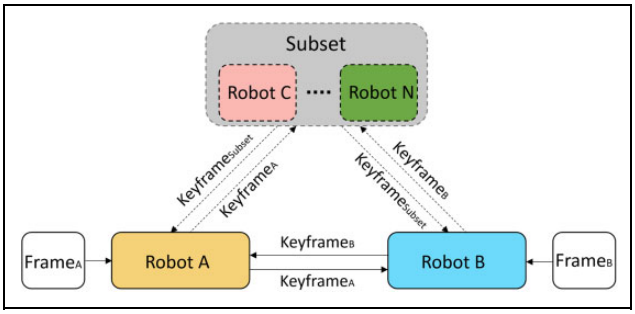


**Figure 8.** The wireless communication between multiple robots. From the perspective of $robot_A$, it is receiving the environment information from $Frame_A$ and other robots' incremental maps, meanwhile, sharing its own incremental map through $keyframe_A$. As a distributed system, all the robots in the proposed collaborative SLAM system are conducting the same process and sharing the same format of map information, which enable the robot not only communicate with another robot but also with a subset of the robot group. SLAM: simultaneous localization and mapping.

*Realizing multi-robot SLAM in indoor scenarios.* In the first experiment, we used two monocular cameras to simulate two robots performing collaborative SLAM in the office environment. The local maps of each robot were shared through the wireless network using bridges and ROS messages. Figure 9 shows the perspective of $robot_A$. Figure 9(a) is the SLAM result of $robot_A$, while Figure 9(b) is the map received from $robot_B$ maintained in the map processing module before map merging. In order to show the difference, we used black and red points to represent the mappoints created by $robot_A$ itself and used purple points to represent the mappoints received from other robots.

Once the same place is detected, the relative pose estimation module will determine the relative poses, and the map merging module will combine these local maps into a global map, as shown in Figure 10. Figure 10(a) illustrates the general map of the indoor office built by a 2D LiDAR, and Figure 10(b) shows the SLAM result after merging the maps of these two robots. In Figure 10(a), the blue line and the red line represent the approximate trajectories of $robot_A$ and $robot_B$, respectively. The triangles represent the start points, and the circles represent the endpoints. The overlaps between these two maps are very small, and we can see that the two robots never see each other directly. In other words, there is no direct line-of-sight observations. These experimental results show that our multi-robot SLAM system can be realized in the indoor office environment using two handheld monocular cameras with little overlaps.

*Realizing multi-robot SLAM in outdoor large-scale scenarios.* In this experiment, we used KITTI data set to realize a large-scale multi-robot collaborative SLAM. Although the data

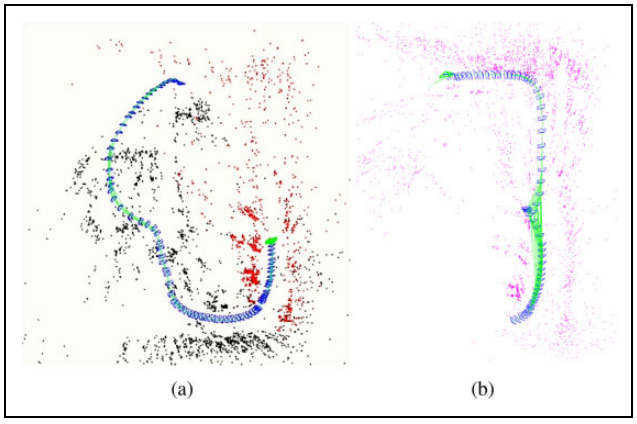


**Figure 9.** The perspective of $robot_A$ conducting multi-robot SLAM in the indoor environment. (a) The SLAM result of $robot_A$. (b) The map received from $robot_B$ maintained in the map processing module. In (a), black points represent the fixed mappoints while red points represent the mappoints which are currently seen by the robot. In (b), the purple points represent the mappoints received from $robot_B$. The blue boxes represent the keyframes, while the green lines represent the edges of the pose graph. SLAM: simultaneous localization and mapping.

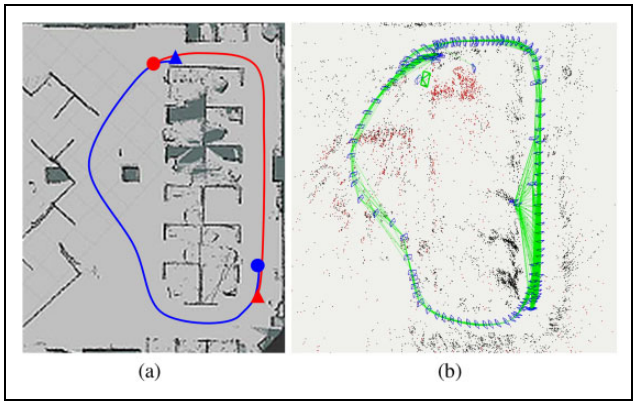


**Figure 10.** The experimental result of the multi-robot collaborative SLAM conducted in the indoor environment. (a) The general map of the indoor office in 2D. (b) The SLAM result after merging the maps of these two robots. In (a), the blue and red lines represent the trajectories of $robot_A$ and $robot_B$, respectively. The triangles represent the start points, and the circles represent the endpoints. SLAM: simultaneous localization and mapping.

set was collected by one vehicle, we split the sequences into several segments and used two computers to simulate two robots performing collaborative SLAM at the same time, or simultaneously conducted SLAM on different sequences and eventually obtained a global map. The images in the KITTI data set were converted into ROS bag files at 10 fps and played in real time.

*Experiment on KITTI sequence 00.* Taking two robots as an example, Figure 11(a) and (b) shows the SLAM results from the perspective of $robot_A$ before map merging. Figure 11(a) shows the local map built by $robot_A$ itself and

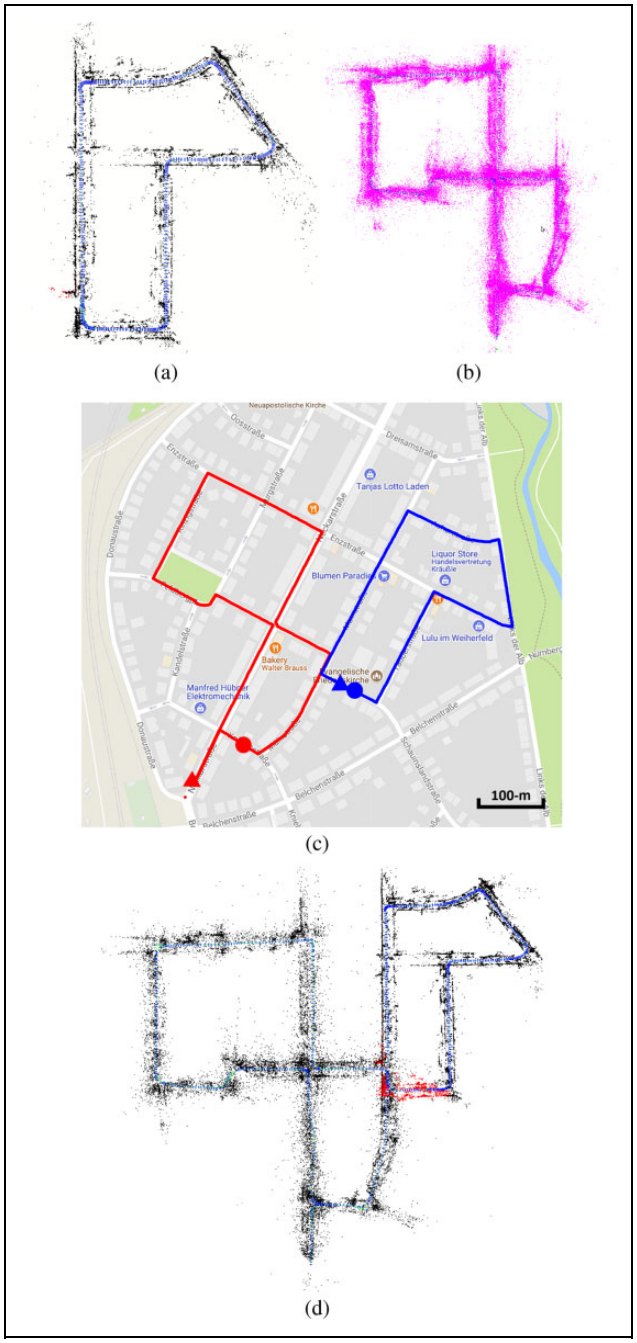


**Figure 11.** Experimental results on KITTI sequence 00. (a) and (b) The SLAM results from the perspective of $robot_A$ before map merging. (a) The local map built by $robot_A$ itself and (b) the local map received from $robot_B$. (c) The corresponding Google map of this experiment. The blue and red lines represent the approximate trajectories of $robot_A$ and $robot_B$, respectively. (d) The final global map after map merging. SLAM: simultaneous localization and mapping.

Figure 11(b) shows the local map received from $robot_B$. Figure 11(c) shows the corresponding Google map of this experimental environment. Approximately, $robot_A$ travels 750 m (shown in blue) and $robot_B$ travels more than 1 km (shown in red). Figure 11(d) shows the final global map

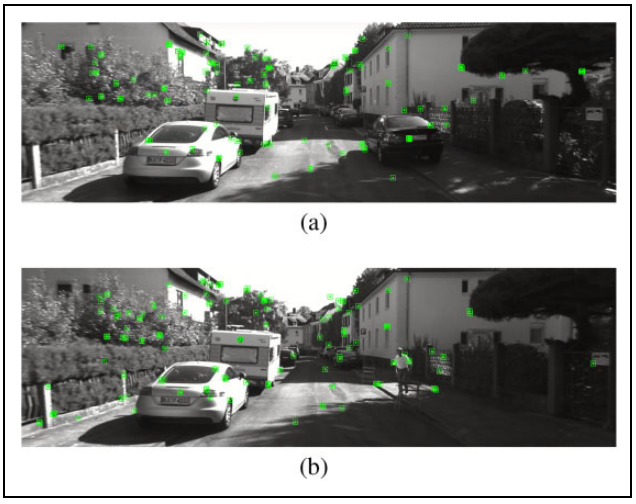


**Figure 12.** Frames which were successfully used for the appearance-based place recognition. (a) The current keyframe of $robot_A$. (b) The matched keyframe of $robot_B$.

acquired by the proposed multi-robot collaborative SLAM. This experiment shows the advantage of the proposed method over other methods that rely on the overlaps of the maps to determine the relative poses. Clearly, in this experiment, the overlap of the maps is very small compared to the size of the maps.

*Experiment on KITTI sequences 00 and 07.* In this experiment, we simultaneously conducted the SLAM on KITTI sequences 00 and 07 to simulate two robots. The image sequences were collected by one vehicle at different time. Therefore, the environment is dynamic which makes it more challenging to realize an outdoor large-scale collaborative SLAM. Figure 12 shows the frames we used to determine the relative poses in these two sequences. It is clear that there is a significant difference between these two images. Figure 13(a) shows the Google map related to these two sequences. The red line represents the approximate trajectory of $robot_A$ carrying out the SLAM on sequence 00, while the blue line represents the approximate trajectory of $robot_B$ carrying out the SLAM on sequence 07. $robot_A$ travels about 3.7 km, while $robot_B$ travels 700 m. Figure 13(b) and (c) shows the maps of sequence 07 maintained in the map processing module of $robot_A$ and the original map built by $robot_B$, respectively. Figure 13(d) shows the map of sequence 00 built by $robot_A$ before map merging, and Figure 13(e) shows the global maps after merging the maps of sequences 00 and 07. This experiment also confirms the ability of the proposed algorithm in performing collaborative SLAM in large-scale environments with little overlaps. It also validates that the proposed system can be used in the real world with moving pedestrians and vehicles.

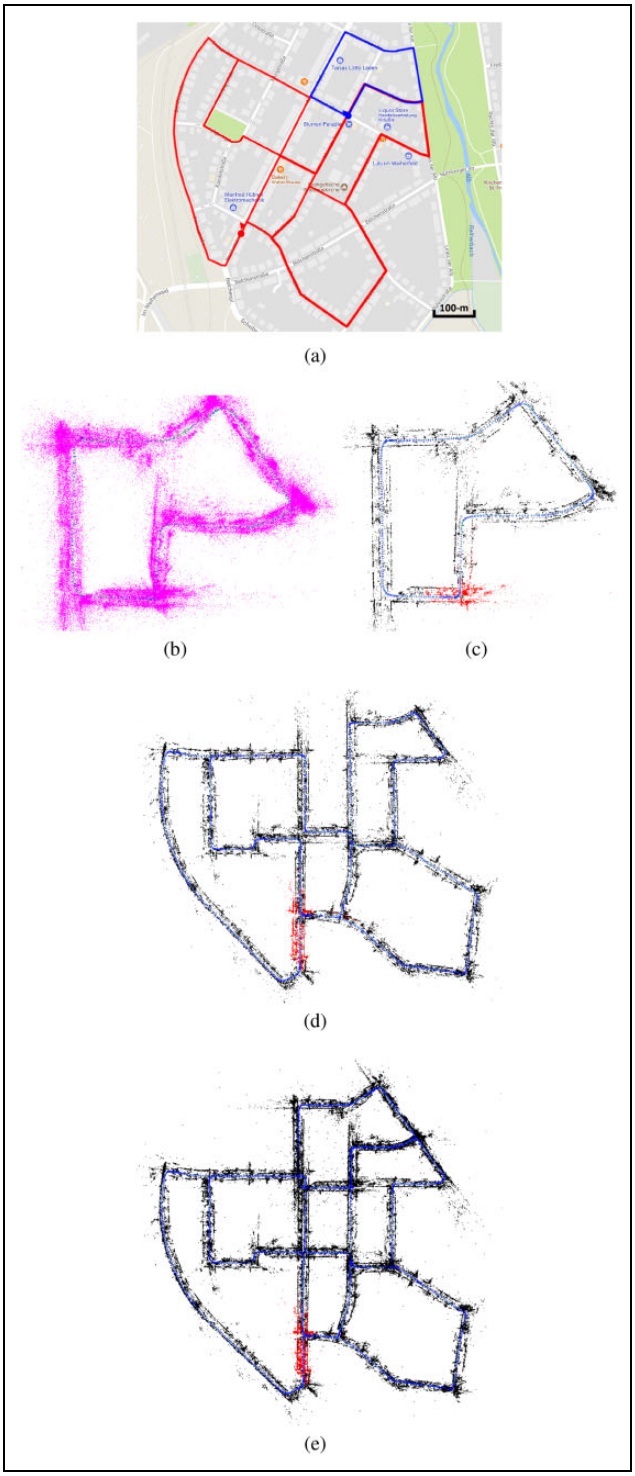


**Figure 13.** Experimental results on KITTI sequences 00 and 07. (a) The Google map related to these two sequences. (b) and (c) The maps of sequence 07 maintained in map processing module of $robot_A$ and the original map built by $robot_B$. (d) The map of sequence 00 built by $robot_A$ before map merging. (e) The global map after merging the maps of sequences 00 and sequence 07.

## Conclusion

In this article, we proposed a robust monocular SLAM based on image-to-map relocalization and active loop closure to solve the well-known tracking failure problem in the monocular SLAM. Based on the robust monocular SLAM, we realized a distributed multi-robot collaborative SLAM system in both indoor small-scale and outdoor large-scale environments. The experimental results show that better robustness can be achieved by our system than other relocalization systems, and the proposed robust monocular SLAM system can be employed in most challenging situations even in previously unvisited places. By combining the active loop closure with the relocalization, high accuracy and efficiency can be achieved simultaneously, which also provides a good example to answer the 'when to use' problem in the active SLAM. Based on the robust monocular SLAM, the proposed multi-robot collaborative SLAM system can well address the relative pose estimation and map merging problem with least requirements on the motion of the robots and the overlaps of the local maps obtained by each robot. The experimental results show that our multi-robot collaborative SLAM system can be realized in real time in both indoor and outdoor large-scale environments.

In this work, we just give a simple example of using the pose information estimated by the proposed relocalization method to realize an active loop closure. In the future, it would be desirable to run our system on real robots with advanced path planning and robot controlling algorithms. Also, we will use more real robots to test our system and find a better way to represent the map to further reduce the amount of calculation and storage.

### Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by National Science Foundation of China (nos 61503401 and 61773393).

### ORCID iD

Xieyuanli Chen http://orcid.org/0000-0003-0955-6681

### References

1. Williams B, Klein G, and Reid I. Automatic relocalization and loop closing for real-time monocular SLAM. *IEEE Trans Pattern Anal Mach Intell* 2011; 33(9): 1699–1712.
2. Feng Y, Wu Y, and Fan L. Online learning of binary feature indexing for real-time SLAM relocalization. In: *Asian conference on computer vision*. pp. 206–217. Springer.
3. Straub J, Hilsenbeck S, Schroth G, et al. Fast relocalization for visual odometry using binary features. In: *2013 20th IEEE international conference on image processing (ICIP)*, Melbourne, VIC, Australia, 15–18 September 2013, pp. 2548–2552. IEEE.
4. Strasdat H, Davison AJ, Montiel JM, et al. Double window optimisation for constant time visual SLAM. In: *2011 IEEE international conference on computer vision (ICCV)*, Barcelona, Spain, 6–13 Novrmber 2011, pp. 2352–2359. IEEE.
5. Mur-Artal R, Montiel JMM, and Tardos JD. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans Robot* 2015; 31(5): 1147–1163.
6. Saeedi S, Trentini M, Seto M, et al. Multiple-robot simultaneous localization and mapping: a review. *J Field Rob* 2016; 33(1): 3–46.
7. Davison AJ. Real-time simultaneous localisation and mapping with a single camera. In: *Proceedings of the ninth IEEE international conference on computer vision, ICCV '03*, 13–16 October 2003, Vol. 2, p. 1403. Washington, DC, USA: IEEE Computer Society. Available at: http://dl.acm.org/citation.cfm?id=946247.946734.
8. Eade E and Drummond T. Monocular SLAM as a graph of coalesced observations. In: *ICCV 2007. IEEE 11th international conference on computer vision*, Rio de Janeiro, Brazil, 14–21 October 2007, pp.1–8. IEEE.
9. Engel J, Schöps T, and Cremers D. LSD-SLAM: large-scale direct monocular SLAM. In: *European conference on computer vision*. pp. 834–849. Springer.
10. Newcombe RA, Lovegrove SJ, and Davison AJ. DTAM: dense tracking and mapping in real-time. In: *2011 IEEE international conference on computer vision (ICCV)*, Barcelona, Spain, 6–13 November 2011, pp. 2320–2327. IEEE.
11. Engel J, Koltun V, and Cremers D. Direct sparse odometry. *IEEE Trans Pattern Anal Mach Intell* 2018; 40(3): 611–625.
12. Dellaert F, Fox D, Burgard W, et al. Monte carlo localization for mobile robots. In: *1999 Proceedings* of *IEEE international conference on robotics and automation*, Vol. 2, Detroit, MI, USA, 10–15 May 1999, pp. 1322–1328. IEEE.
13. Williams B, Cummins M, Neira J, et al. A comparison of loop closing techniques in monocular SLAM. *Robot Auto Syst* 2009; 57(12): 1188–1197. Available at: http://www.sciencedirect.com/science/article/pii/S0921889009000876.
14. Clemente LA, Davison AJ, Reid ID, et al. Mapping large loops with a single hand-held camera. *Robot Sci Syst* 2007; 2.
15. Reitmayr G and Drummond T. Going out: robust model-based tracking for outdoor augmented reality. In: *Proceedings of the 5th IEEE and ACM international symposium on mixed and augmented reality*. pp. 109–118. IEEE Computer Society.
16. Klein G and Murray D. Improving the agility of keyframe-based SLAM. In: *European conference on computer vision*. pp. 802–815.
17. Sivic J and Zisserman A. Video Google: a text retrieval approach to object matching in videos. In: *null*. pp. 1470–1477. IEEE.
18. Eade E and Drummond T. Unified loop closing and recovery for real time monocular SLAM. *BMVC* 2008; 13: 136.
19. Lowe DG. Distinctive image features from scale-invariant keypoints. *Int J Comput Vision* 2004; 60(2): 91–110.

Available at: https://doi.org/10.1023/B:VISI.0000029664.99615.94.
20. Cummins M and Newman P. Appearance-only SLAM at large scale with fab-map 2.0. *Int J Rob Res* 2011; 30(9): 1100–1123. Available at: https://doi.org/10.1177/0278364910385483.
21. Bay H, Tuytelaars T, and Van Gool L. SURF: speeded up robust features. *Comput Vision ECCV* 2006; 2006: 404–417.
22. Mur-Artal R and Tardós JD. Fast relocalisation and loop closing in keyframe-based SLAM. In: *2014 IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, 31 May–7 June 2014, pp. 846–853. IEEE.
23. Rublee E, Rabaud V, Konolige K, et al. ORB: an efficient alternative to SIFT or SURF. In: *2011 IEEE international conference on computer vision (ICCV)*, Barcelona, Spain, 6–13 November 2011, pp. 2564–2571. IEEE.
24. Pupilli M and Calway A. Real-time camera tracking using a particle filter. *BMVC*. Available at: https://doi.org/10.5244/C.19.50.
25. Se S, Lowe DG, and Little JJ. Vision-based global localization and mapping for mobile robots. *IEEE Trans Robot* 2005; 21(3): 364–375.
26. Gionis A, Indyk P, Motwani R, et al. Similarity search in high dimensions via hashing. *VLDB* 1999; 99: 518–529. Available at: http://dl.acm.org/citation.cfm?id=645925.671516.
27. Feder HJS, Leonard JJ, and Smith CM. Adaptive mobile robot navigation and mapping. *Int J Robot Res* 1999; 18(7): 650–668. Available at: https://doi.org/10.1177/027836499922066484.
28. Rone W and Ben-Tzvi P. Mapping, localization and motion planning in mobile multi-robotic systems. *Robotica* 2013; 31(1): 1–23.
29. Zhou XS and Roumeliotis SI. Multi-robot SLAM with unknown initial correspondence: the robot rendezvous case. In: *2006 IEEE/RSJ international conference on intelligent robots and systems*, Beijing, China, 9–15 October 2006, pp. 1785–1792. IEEE.
30. Ko J, Stewart B, Fox D, et al. A practical, decision-theoretic approach to multi-robot mapping and exploration. In: *2003. (IROS 2003). Proceedings of 2003 IEEE/RSJ international conference on intelligent robots and systems*, Vol 4, Las Vegas, NV, USA, 27–31 October 2003, pp. 3232–3238. IEEE.
31. Fox D, Ko J, Konolige K, et al. Distributed multirobot exploration and mapping. *Proc IEEE* 2006; 94(7): 1325–1339.
32. Birk A and Carpin S. Merging occupancy grid maps from multiple robots. *Proc IEEE* 2006; 94(7): 1384–1397.
33. Indelman V, Nelson E, Michael N, et al. Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization. In: *2014 IEEEinternational conference on robotics and automation (ICRA)*, Hong Kong, China, 31 May–7 June 2014, pp. 593–600. IEEE.
34. Kim B, Kaess M, Fletcher L, et al. Multiple relative pose graphs for robust cooperative mapping. In: *2010 IEEE international conference on robotics and automation (ICRA)*, Anchorage, AK, USA, 3–7 May 2010, pp. 3185–3192. IEEE.
35. Dong J, Nelson E, Indelman V, et al. Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach. In: *2015 IEEE international conference on robotics and automation (ICRA)*, Seattle, WA, USA, 26–30 May 2015, pp. 5807–5814. IEEE.
36. Saeedi S, Paull L, Trentini M, et al. Map merging for multiple robots using hough peak matching. *Robot Auto Syst* 2014; 62(10): 1408–1424. Available at: http://www.sciencedirect.com/science/article/pii/S0921889014001134.
37. Fox D, Burgard W, Kruppa H, et al. A probabilistic approach to collaborative multi-robot localization. *Auton Robot* 2000; 8(3): 325–344. Available at: https://doi.org/10.1023/A:1008937911390.
38. Martinelli A, Pont F, and Siegwart R. Multi-robot localization using relative observations. In: *ICRA 2005. Proceedings of the 2005 IEEE international conference on robotics and automation*, Barcelona, Spain, 18–22 April 2005, pp. 2797–2802. IEEE.
39. Vidal-Calleja TA, Berger C, Solà J, et al. Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robot Auto Syst* 2011; 59(9): 654–674. Available at: http://www.sciencedirect.com/science/article/pii/S0921889011000923.
40. Kaess M and Dellaert F. Probabilistic structure matching for visual SLAM with a multi-camera rig. *Comput Vis Image Underst* 2010; 114(2): 286–296. Available at: http://www.sciencedirect.com/science/article/pii/S1077314209001283.
41. Sola J, Monin A, and Devy M. BiCamSLAM: two times mono is more than stereo. In: *2007 IEEE international conference on robotics and automation*, Roma, Italy, 10–14 April 2007, pp. 4795–4800. IEEE.
42. Doitsidis L, Renzaglia A, Weiss S, et al. 3D surveillance coverage using maps extracted by a monocular SLAM algorithm. In: *2011 IEEE/RSJ international conference on intelligent robots ansystems (IROS)*, San Francisco, CA, USA, 25–30 September 2011, pp. 1661–1667. IEEE.
43. Forster C, Lynen S, Kneip L, et al. Collaborative monocular SLAM with multiple micro aerial vehicles. In: *2013 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Tokyo, Japan, 3–7 November 2013, pp. 3962–3970. IEEE.
44. Chebrolu N, Marquez-Gamez D, and Martinet P. Collaborative visual SLAM framework for a multi-robot system.
45. Schmuck P. Multi-uav collaborative monocular SLAM. In: *2017 IEEE international conference on robotics and automation (ICRA)*, Singapore, 29 May–3 June 2017, pp. 3863–3870. IEEE.
46. Chen X, Lu H, Xiao J, et al. Robust relocalization based on active loop closure for real-time monocular SLAM. In: *2017 Springer international conference on computer vision systems (ICVS)*, pp. 131–143. Springer.
47. Chen X, Zhang H, Lu H, et al. Robust SLAM system based on monocular vision and lidar for robotic urban search and rescue. In: *2017 IEEE international symposium on safety,*

*security and rescue robotics (SSRR)*, Shanghai, China, 11–13 October 2017, pp. 41–47. IEEE. DOI: 10.1109/SSRR.2017.8088138.

48. Stachniss C, Hahnel D, and Burgard W. Exploration with active loop-closing for fastSLAM. In: *2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ international conference on intelligent robots and systems*, Vol. 2, Sendai, Japan, 28 September–2 October 2004, pp. 1505–1510. IEEE.
49. Rahimi A, Morency LP, and Darrell T. Reducing drift in parametric motion tracking. In: *ICCV 2001. Proceedings. Eighth IEEE international conference on computer vision, 2001*, Volume 1, Vancouver, BC, Canada, 7–14 July 2001, pp. 315–322. IEEE.
50. Muja M and Lowe DG. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)* 2009; 2(331–340): 2.
51. Lepetit V, Moreno-Noguer F, and Fua P. Epnp: an accurate o(n) solution to the pnp problem. *Int J Comput Vision* 2009; 81(2): 155–166. Available at: https://doi.org/10.1007/s11263-008-0152-6.
52. Kümmerle R, Grisetti G, Strasdat H, et al. g2o: a general framework for graph optimization. In: *2011 IEEE international conference on robotics and automation (ICRA)*, Shanghai, China, 9–13 May 2011, pp. 3607–3613. IEEE.
53. Chen X, Lu H, Xiao J, et al. Distributed monocular multi-robot SLAM. In: *The 8th Annual IEEE international conference on CYBER technology in automation, control, and intelligent systems (IEEE-CYBER)*. IEEE.
54. Galvez-Lpez D and Tardos JD. Bags of binary words for fast place recognition in image sequences. *IEEE Trans Robot* 2012; 28(5): 1188–1197.
55. Strasdat H, Montiel J, and Davison AJ. Scale drift-aware large scale monocular SLAM. *Robot Sci Syst VI* 2010; 2. Available at: http://www.roboticsproceedings.org/rss06/p10.html.
56. Strasdat H. Local accuracy and global consistency for efficient SLAM 2012. Available at: https://books.google.com/books?id=4cR4mwEACAAJ.
57. Geiger A, Lenz P and Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *2012 IEEE conference on computer vision and pattern recognition (CVPR)*, Providence, RI, USA, 16–21 June 2012, pp. 3354–3361. IEEE.
58. Leung KY, Barfoot TD, and Liu HH. Decentralized cooperative SLAM for sparsely-communicating robot networks: a centralized-equivalent approach. *J Int Rob Syst* 2012; 66(3): 321–342. Available at: https://doi.org/10.1007/s10846-011-9620-2.
59. Howard A. Multi-robot simultaneous localization and mapping using particle filters. *Proc IEEE* 2005; 94(7): 1360–1369. Available at: https://doi.org/10.1177/0278364906072250.