# Collaborative Visual SLAM Framework for a Multi-UAVs System Based on Mutually Loop Closing

Haifeng Yu[1], Hao Li[1], and Zhihua Yang[1,2(✉)]

[1] Communication Engineering Research Center, Harbin Institute of Technology (Shenzhen), Shenzhen, Guangdong, China
mryuhaifeng@163.com, lihao_hitsz@163.com, yangzhihua@hit.edu.cn
[2] Pengcheng Laboratory, Shenzhen, Guangdong, China

**Abstract.** In a typical visual Simultaneous Localization and Mapping (SLAM) algorithm integrated within a single unmanned aerial vehicle (UAV), the positioning drift will increase cumulatively due to the motion and dynamic of UAV platform, which could be efficiently alleviated by the introduction of loop detection. However, a large number of loop movements by a single UAV will result in too many turns of the drone with significant reduction of coverage area per unit time. Therefore, in this paper, we propose a collaborative framework of visual SLAM algorithm with a multiple UAVs system. In the proposed framework, a series of mutually closed loop will be detected and executed by multiple UAVs within the position map. By this coordination method, the position accuracy of the system could be obviously improved, through the experimental results compared with a single UAV SLAM system.

**Keywords:** Visual SLAM · Multi-UAVS · Loop closing · Mapping merge

## 1 Introduction

Simultaneous localization and mapping (SLAM) is an effective method to solve the problem of synchronous localization and map reconstruction in unknown environments. In recent years, the SLAM algorithm is attractively applied into the system of unmanned aerial vehicle (UAV), in order to locate the position of the UAV while generating the map especially in an unknown environment. One broad category of SLAM algorithms attempts to build and maintain a pose (position and orientation) graph, which consists of a group of UAV states (represented as nodes), and pose transformations (as edges, i.e., constraints). For each state, related poses and sensor data, e.g., camera images or laser scans, are contained. Generally, the drifting error from odometry can be reduced by detecting re-visited places and introducing loop closure constraints, with a global


M. Jia et al. (Eds.): WiSATS 2019, LNICST 280, pp. 653–664, 2019.
https://doi.org/10.1007/978-3-030-19153-5_64

graph optimization. In particular, the goal of graph optimization in this context is to find a configuration that minimizes the errors of the constrained pose estimations [1]. As a result, the expected map can be reconstructed at any time from the graph by superimposing the associated sensor data with a consistent configuration.

Typically, compared with the terrestrial robot system integrated with a SLAM, a drone platform has a faster and more flexible flight as making an exploration mission in a wide-range environment, leading to an easily loss of the feature information of objects and thus a sparse density of the resulted point cloud map. In recent days, a collaborative method by using multiple UAVs is elaborately employed for the SLAM, which can improve the robustness of positioning and the accuracy of the map with a less time. In particular, in a large-scale environment, a collective use of multiple UAVs can not only speed up the reconstruction of the map, but also improve the accuracy through multi-machine map fusion. In specific, during the mission, the UAVs team can branch off the cruise path and meet again at some place where they share their maps. Another advantage of multi-UAV collaboration is the detection capability on the overlap area. For example, a required redundancy of the data can be provided by other UAVs in the event of a loss of positioning of one certain UAV, which could help locating its own exact location. Therefore, in this paper, we propose a collaborative visual SLAM algorithmic framework for a multi-UAVs system based on mutually loop closing. In the proposed algorithm, a dense point cloud map is constructed by fusing multiple pose maps from different drones with detection on a series of key frames (site or scene identification) from different drone shooting scenes. The proposed algorithm is designed on the basis of a well-tailored ORB-SLAM2 framework and verified by a group of experiments in real scenes.

The remainder of this paper is structured as follows. Section 2 lists related research. Section 3 particularizes the posed problem and explains the proposed methods in details. In Sect. 4, we present experiment results and discussion. Finally, we summarize our work in Sect. 5.

## 2 Related Work

In the early SLAM, laser scanners were used as the main sensors of SLAM for estimating poses. In [2], an extended Kalman filter (EKF) method is exploited for relative observation between multiple robots. With the developments of camera technology, a variety of sensors have gradually replaced inexpensive lasers with inexpensive costs. In [3], the monocular cameras and the IMU are cooperatively employed to form a flexible three-dimensional platform of UAVs team, which is used to estimate the relative poses of UAVs on a simple experimental setup. In [4], Piasco et al. use a distributed stereo-vision system of multiple UAVs for collaborative localization with an EKF filtering algorithm. In [5], a UAV equipped with multiple cameras is used to estimate the trajectory of moving objects in the scene, while creating a 3-D map of a static object.

In order to improve the running speed, at present, non-linear optimization methods are gradually exploited in the SLAM to replace traditional filtering

methods. In [6], a fully distributed system is proposed to realize collaborative SLAMs and evaluate them in a simulated environment. The main challenge for decentralized systems is more difficult to ensure data consistency and avoid duplicate calculations, compared with a centralized client-server architecture. In addition, in a central server-assisted system, computationally expensive algorithms can be handled on the server side, which does not necessarily limit real-time operations such as bundling adjustment and location identification. As a result, this allows a flying agent to use its only potentially limited resources for performing the most critical task, such as real-time visual ranging. In several cases, a centralized SLAM would utilize a cloud server instead of a central server to exchange data. In [7], the ground agent will update the related information of key-frames in the local map to the cloud system without checking the global map stored in the cloud. A centralized framework is proposed for a group of UAVs equipped with monocular cameras, in which each UAV performs visual odometry on its on-board processor.

## 3 Methodology

### 3.1 Algorithmic Framework

The method is applicable to a quadrotor UAV. The camera pose is consistent with the position of the rotor unmanned aerial vehicle. The environment map is constructed indoors by three or more drones. The maximum speed of the drone flight is 1 m/s, the turning angle is less than 30°.

In this section, we devise a loop-closing enabled algorithmic framework of multi-UAVs SLAM system, as shown in Fig. 1. In specific, the proposed algorithm consists of two parts, which are separately executed at multi-UAVs system and central server. The proposed algorithm system use ORB-SLAM2 for locating and mapping at each UAV, meanwhile construct a complete map at central server by merging those sectional maps from different agent. Without loss of generality, in this paper, we assume that a reliable and real-time data transmission is ensured between UAVs and central server. In each UAV, a RGB Depth cameras and

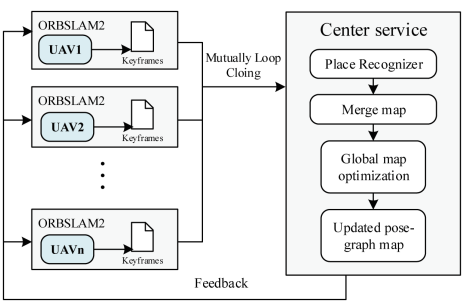



**Fig. 1.** Collaborative visual SLAM framework based on loop closing.

on-board processors are equipped for visual mileage calculations, which provides an estimate of the UAV platform attitude, with a 3D map of the environment in the local coordinate system.

**Algorithm 1.** Algorithm in Service

**Begin**
$msgFlag \longleftarrow False$;
// receive the key frame from different UAVs
**while** $msgFlag \longleftarrow if\_connUAV()$ **do**;
$[uavLabel, keyFrame] \longleftarrow recv\_keyFrame()$;
**end while**
// merge the maps
$subMap \longleftarrow find\_olpregion(keyFrame, uavLabel)$;
$mergeMap \longleftarrow merge\_submaps(submaps)$;
$optMap \longleftarrow optimal\_Map(mergeMap)$;
$optMap \longleftarrow update\_Map(optMap)$;
$msgFlag \longleftarrow False$;
// feedback the optiaml maps to the different UAVs
**while** $msgFlag \longleftarrow if\_connUAV()$ **do**;
$send\_map(uavLabel, optMap)$
**end while**
**End**

**Algorithm 2.** Algorithm in UAVs

**Input:** task
$keyframe \longleftarrow run\_orbslam()$;
$send\_keyFrame(keyFrame, centerService)$;
**Output:**

In the proposed algorithm, each UAV transmits a series of key frames (posture and shooting picture information in the local coordinates system) to the central server. Meanwhile, the central server will continuously monitor all observations and detect those same shooting scenes from different UAVs. In particular, we use the technology of Bag of Word (BOW) to quickly compare their image features and perform overlapping detection algorithms on the appearance. Once overlapping scenes from different UAVs are detected, a well-designed map fusion algorithm is started to extract the ORB features of the related two images, and then feature matching is performed to remove false matches. An iterative closest point algorithm is performed to calculate the two key frames for similarity transformation. Then, the two maps are merged into a single global map. At the same time, a bundle optimization algorithm is run on the global map. The updated pose of the keyframe image is fed back to each UAV to improve its local estimation and local map in turn.

### 3.2 Improved ORB-SLAM2

In this paper, we built a feature-based ORB-SLAM2 framework integrated with a stereo and RGB-D cameras based on the typical version of ORB-SLAM2 [8]. For reader convenience, the main components are summarized here, which includes three main parallel threads, as shown in Fig. 2.

(1) Tracking thread. It will localize the camera with every frame by finding those features matching the local map, and minimizing the re-projection error with motion-only Bundle Adjustment (BA).
(2) Local mapping thread. This thread will optimally manage the local map with local BA.
(3) Loop closing thread. It attempts to detect large loops and correct the accumulated drift by performing a pose-graph optimization.

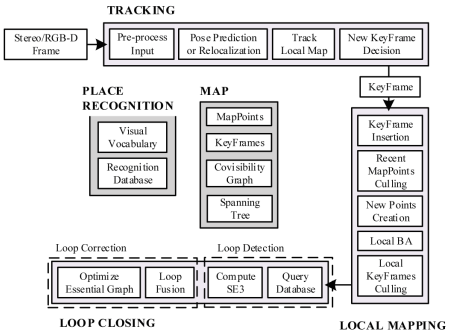



**Fig. 2.** The frame of ORBSLAM2

Compared with typical version of ORB-SLAM2, a dense point cloud map is constructed by utilizing a sparse point cloud with a RGD image to remap the point cloud map, and added into the proposed framework in this paper, which has a large amount of information for restoration on the environments more sparse point clouds in typical ORB-SLAM2. In particular, here, we use an incremental method to generate a dense surface model from a semi-dense point cloud. The proposed method first generates a dense surface mesh for every single scan, and creates corresponding image by doing edge-aware resampling and triangulating the surface points.

### 3.3 Place Recognizer

In the proposed framework, loop closing is performed in two steps. Firstly, a loop has to be detected and validated, which is called place recognizer. Secondly, the loop is corrected by optimizing the fused attitude map of pose-graph. In this framework, we incorporated a full bundle adjustment (BA) optimization after the maps are fused. Considering that this optimization might be very

costly, we perform it in a separate thread, allowing the system to continue creating maps and detecting loops. However, this brings the challenge of merging the BA output with the current state of the map. If a new loop is detected while the optimization is running, the algorithm aborts the current optimization and proceeds to close the new loop, which will launch the full BA optimization again. Once the full BA finishes, the algorithm will merge the updated subset of keyframes and points, with the non-updated keyframes and points inserted while the optimization was running. This is done by propagating the correction of updated keyframes (i.e. the transformation from the non-optimized to the optimized pose) to non-updated keyframes through the spanning tree. Then, non-updated points are transformed according to the above correction. In this collaborative framework, the identification of the location and the fusion of those separated maps are handled in a central server, in which local maps in different coordinate systems are merged into a global map by finding similar scenes from different UAVs. In this framework, we extract ORB features from all keyframe images and use Bag of Words (BOW) to represent keyframe features as descriptors. We use the DBOW3 open source software package to create the dictionary and finally generate a descriptor that facilitates location detection. In specific, DBOW3 uses the k-means++ method to create visual dictionaries. Each frames picture is described by dictionary information. Finally, the same place is detected by comparing the degree of similarity of descriptors of each frame.

### 3.4 Map Merge

In order to achieve map merge, we use a iterative closest point (ICP) algorithm in [9], which tries to find a transformation that minimizes the distance between a set of corresponding points in two clouds. In this section, we tailored an augmented version of ICP which also includes surface normal and tangent information to improve the estimate results. Note that both the Sim3 [10] tracker and the ICP procedure use a gradient approach to find the solution. In theory, both methods could be used in any order. However, the ICP procedure was found to be most accurate if starting from a better estimation of the scale factor.

When used to register and align two sets of two-dimensional or three-dimensional points with relative motion, the ICP can continuously solve the optimal value by iterations. This issue could be modeled by finding the optimal solution of the least-squares problem as shown in (1).

$$\min E = \sum_{i=1}^{n} (q_i - (Rp_i + t))^2 \tag{1}$$

in which $P = \{p_i | i = 1, \ldots, n\}$ and $Q = \{q_i | i = 1, \ldots, n\}$ are respectively 3D feature point sets from different UAVs with the 3D coordinates of $(x, y, z)_i^T$. By continuously iterating the rotation matrix $\boldsymbol{R}$ and the translation matrix $\boldsymbol{t}$, we get a motion transformation matrix $\boldsymbol{T} = [\boldsymbol{R}, \boldsymbol{t}]$. Trough the motion transformation matrix $\boldsymbol{T}$, we can get the merge map form the source point set $Q$ to the target point set $P$.

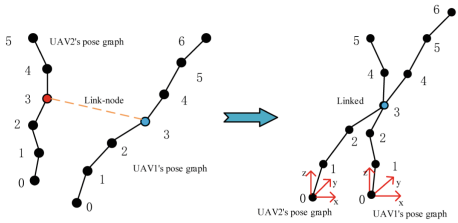



**Fig. 3.** Merged diagram of multi-UAVs pose graph.

The merged process of pose graph of multiple UAVs is shown in Fig. 3. In the figure, two third nodes are the separated pose nodes from two individual UAV paths for the same environment identified by the BOW algorithm. By the ICP, a transformation matrix T could be calculated and utilized for merging into a global pose graph in a unified coordinate system with the two different pose diagrams.

### 3.5 Global Map Optimization

Once the transformation is computed, new similarity constraints are added between the matched keyframes. The corresponding local maps are transformed into the global coordinate system by considering one of the two as reference (if it is the first map to be merged) or using the existing reference otherwise. After the new constraints have been added, a BA step is performed over the merged graph.

After the ICP procedure, we establish the relationships between different UAVs maps. Then, we use these images to triangulate some feature points with geometry. The constructed triangle could determine the position of the three-dimensional point. Finally, the coordinates of the three-dimensional point are calculated to re-project the camera matrix. This issue could be shown in (2).

$$\{R, t\} = \arg\min_{R,t} \sum_{i \in \chi} \rho(\|x^i_{(\cdot)} - \pi_{(\cdot)}(RX^i + t)\|^2_{\Sigma} \tag{2}$$

In which $\boldsymbol{R} \in SO(3)$ is the camera orientation of BA optimizes and $\boldsymbol{t} \in \mathbb{R}^3$ is position. $\boldsymbol{X}^i \in \mathbb{R}^3$ and $\boldsymbol{x}^i_{(\cdot)}$ expresses respectively the minimizing reprojection error between matched 3D points in world coordinates and keypoints. Either monocular $\boldsymbol{x}^i_m \in \mathbb{R}^2$ or stereo $\boldsymbol{x}^i_s \in \mathbb{R}^3$, with $i \in \chi$ the set of all matches.

We use general graph optimization (g2o) to solve the bundle adjustment. g2o is an open source framework that is mainly used to solve the problem based on graph optimization. The optimization problem is described the following formulation:

$$\{X^i, R_l, t_l \mid i \in p_L, l \in \kappa_L\} = \arg\min_{X^i, R_L, t_l} \sum_{k \in \kappa_L \cup \kappa_F} \sum_{j \in \chi_k} \rho(E_{k,j}) \tag{3}$$

where

$$E_{kj} = \|x^j_{(\cdot)} - \pi_{(\cdot)}(R_k X^i + t_k)\|^2_{\sigma} \tag{4}$$

In which, local BA optimizes a set of covisible keyframes $\boldsymbol{\kappa}_L$ and all points seen in those keyframe is $p_L$. All other keyframes $\boldsymbol{\kappa}_F$, not in $\boldsymbol{\kappa}_L$, observing points in $p_L$ contribute to the cost function but remain fixed in the optimization. Defining $\chi_k$ as the set of matches between points in $p_L$ and keypoints in a keyframe $k$.

The conversion relationship between the dense point cloud maps is same with the pose graph. Finally, according to the above algorithm, the two point cloud maps are merged. Finally, the overall map, consisted of keyframe poses as vertices and BA constraints as edges, is continuously optimized in parallel by using a general graph optimization framework in [10] for reducing the drifts both in scale and pose estimates.

### 3.6 Updated Pose Graph

When different UAVs visit the same place in the environment, new constraints are created in the global graph. If they cross each other path by multiple times, these resulted loop closure constraints will help in reducing the overall drifts.

Finally, the central server communicates the updated pose graph to individual UAVs, which can then use this information to update their localization estimate and the local maps. This overall feedback mechanism facilitates the extension of sensing capability of an individual UAV beyond the direct reach of their respective on-board sensors.

## 4 Experimental Results

In this experiment, the multi-UAV map fusion algorithm based on loopback detection is applied to three separate quantity acquisition and processing modules, and the generated keyframe information is merged to finally realize the fusion of 3D point cloud maps. The camera is ASUS's Xtion2. The stand-alone platform is a great deal for Dajiang, and the central processor is Lenovo's notebook. Finally, it analyzes the number of keyframes, algorithm implementation complexity, and compositional integrity. Verify the feasibility and advantages of the algorithm.

On each stand-alone platform, we use the ORB-SLAM2 algorithm to achieve stand-alone SLAM positioning and composition, as shown in Fig. 4, which is the RGB picture in left and ORB-SLAM2's feature tracking process in right. The blue and green colors in the picture indicate the extracted feature points and the matched feature points respectively.

A single agent runs ORB-SLAM2 to estimate is own pose, and builds a 3-dimensional point cloud map of the environment. Figure 5 shows the path maps and point cloud maps of the three agents. As can be seen from the figure, the dense point cloud map constructed by a single machine is relatively vague and has a relatively small amount of information. It can be seen from the path in the sparse map that no large loopback detection occurs.

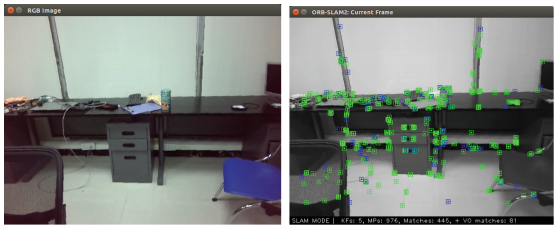
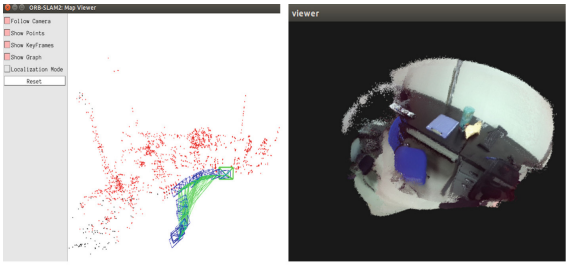
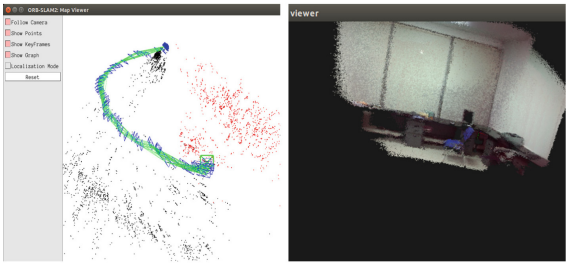


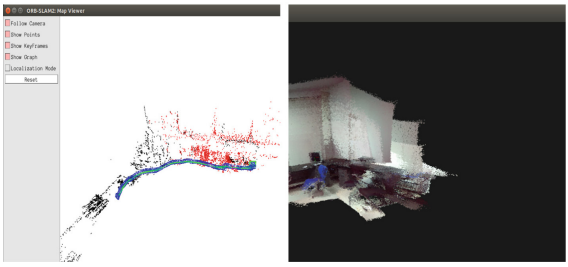

**Fig. 4.** The features of ORB. (The bule points is all the ORB features and the green points is matched features). (Color figure online)




(a) The pose graph and point cloud map of UAV1.




(b) The pose graph and point cloud map of UAV2.




(c) The pose graph and point cloud map of UAV3.

**Fig. 5.** The pose graph and point cloud map.

According to the 3.2 map fusion algorithm, the keyframe data of the two platforms, based on the mutual loop detection, the fusion results of the map are shown in Fig. 6, and it can be clearly seen that the map obtained by the multi-machine fusion is better than the stand-alone, and the map can be used. Observed environmental information.

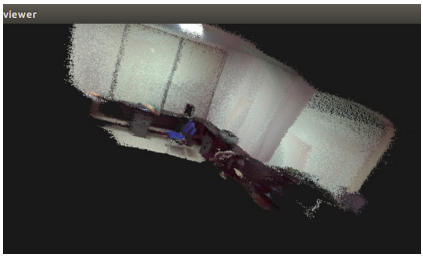



**Fig. 6.** The merged point cloud map of UAV1 and UAV2.

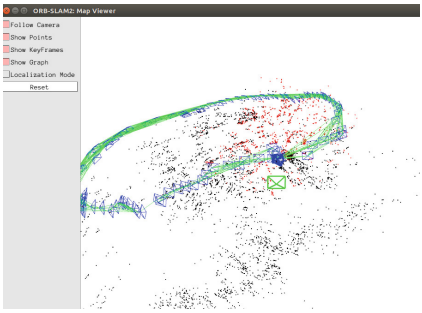



**Fig. 7.** The merged pose gragh.

It is shown the merged point cloud map in Fig. 6. According to the Collaborative Visual SLAM Framework merge Algorithm the point cloud maps from different agencies are merged into a relatively complete global point cloud map. In Fig. 7, we can see that through the fusion of information, the path map increases the number of loops, and the positioning accuracy is improved.

In Fig. 8, It describes the number of map points and the number of generated key frames in a stand-alone system and a multi-machine system, respectively. In the same environment, the number of map points of a multi-UAVs system is more than that of a stand-alone system. On the contrary, the number of key frames is less than that of a stand-alone system. It can be concluded that the multi-machine system can reduce the number of extracted keyframes and the complexity of the algorithm. However, the multi-machine system has the more map points, the more fusion map information, and the higher map accuracy.

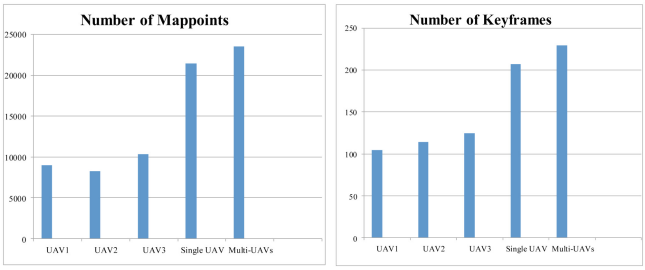



**Fig. 8.** The number of mappoints and keyframes.

**Table 1.** The number of looping closing

| | Single UAV | MultI-UAV |
|---|---|---|
| Number of LC | 6 | 13 |
| Number of mutually LC | – | 7 |

Table 1 describes the number of loop closing, in which, the number of reciprocating loops can be increased by multi-machine information fusion in a multi-machine system, thereby improving the positioning accuracy and robustness of the entire system.

## 5 Conclusions

We propose a collaborative framework of visual SLAM algorithm with a multiple UAVs system. While the small UAVs run real-time visual odometry onboard independently, the computationally more powerful central server aggregates their experiences, searches for loop closing and merges maps. Communicating merged and optimized information back to the agents, the agents are able to operate on updated information, enabling better and more consistent estimates.

In the proposed framework, a series of mutually closed loop will be detected and executed by multiple UAVs within the position map. Compared with some existing frameworks, the final feedback can improve the positioning accuracy of each agent. At the same time, there are also problems. The processing capability of the central server directly affects the real-time nature of map fusion and feedback. In practical applications, communication is used. The real-time nature of the entire framework is also an important part of it. A good communication environment, the availability and reliability of transmission are also important factors that determine the quality of the final map fusion.

**Acknowledgments.** The authors would like to express their high appreciations to the supports from the National Natural Science Foundation of China (61871426) and Basic Research Project of Shenzhen (JCYJ20170413110004682).

## References

1. Deutsch, I., Liu, M., Siegwart, R.: A framework for multi-robot pose graph SLAM. In: IEEE International Conference on Real-Time Computing and Robotics, pp. 567–572 (2016)
2. Martinelli, A., Pont, F., Siegwart, R.: Multi-robot localization using relative observations. In: IEEE International Conference on Robotics and Automation, pp. 2797–2802 (2006)
3. Achtelik, M.W., Weiss, S., Chli, M., Dellaert, F.: Collaborative stereo. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2242–2248 (2011)
4. Piasco, N., Marzat, J., Sanfourche, M.: Collaborative localization and formation flying using distributed stereo-vision. In: IEEE International Conference on Robotics and Automation, pp. 1202–1207 (2016)
5. Mohanarajah, G., Usenko, V., Singh, M., D'Andrea, R., Waibel, M.: Cloud-based collaborative 3D mapping in real-time with low-cost robots. IEEE Trans. Autom. Sci. Eng. **12**(2), 423–431 (2015)
6. Kaess, M., Dellaert, F.: Probabilistic structure matching for visual slam with a multi-camera rig. Comput. Vis. Image Underst. **114**(2), 286–296 (2010)
7. Riazuelo, L., Civera, J., Montiel, J.M.M.: C2TAM: a cloud framework for cooperative tracking and mapping. Robot. Autonom. Syst. **62**(4), 401–413 (2014)
8. Mur-Artal, R., Montiel, J.M.M., Tards, J.D.: ORB-SLAM: a versatile and accurate monocular slam system. IEEE Trans. Robot. **31**(5), 1147–1163 (2017)
9. Holz, D., Behnke, S.: Sancta simplicitas - on the efficiency and achievable results of SLAM using ICP-based incremental registration. In: IEEE International Conference on Robotics and Automation, pp. 1380–1387 (2010)
10. Kim, A., Eustice, R.M.: Active visual SLAM for robotic area coverage: theory and experiment. Int. J. Robot. Res. **34**(4–5), 457–475 (2015)