CORB-SLAM: A Collaborative Visual SLAM System for Multiple Robots

Fu Li[1,2], Shaowu Yang[1,2(✉)], Xiaodong Yi[1,2], and Xuejun Yang[1,2]

[1] State Key Laboratery of High Performance Computing (HPCL), National University of Defense Technology, Changsha, China
shaowu.yang@nudt.edu.cn

[2] College of Computer, National University of Defense Technology, Changsha, China

Abstract. With the single-robot visual SLAM method reaching maturity, the issue of collaboratively exploring unknown environments by multiple robots attracts increasing attention. In this paper, we present CORB-SLAM, a novel collaborative multi-robot visual SLAM system providing map fusing and map sharing capabilities. Experimental results on popular public datasets demonstrate the performance of the CORB-SLAM. Furthermore, we make the source code of CORB-SLAM to be publicly available (https://github.com/lifunudt/CORB-SLAM.git).

Keywords: Collaborative visual SLAM · Map fusing · Map sharing

1 Introduction

Simultaneous localization and mapping (SLAM) is originally proposed to locate a robot while simultaneously building a consistent map of an unknown environment. It has been a fundamental technology for autonomous navigation of robots in the past decades. Researchers have proposed a number of SLAM methods using different types of sensors, e.g. cameras or 2D/3D laser scanners. Specifically, the visual SLAM systems that utilize cameras to obtain sensor-data have become an attractive research focus in robotics.

Some milestone visual SLAM systems have been proposed, such as PTAM [1], and ORB-SLAM2 [2]. Most of these visual SLAM methods focus on the single-robot applications. However, because of the increasing demands for robotic applications in large-scale environments, multi-robot systems are expected to work collaboratively to resolve complex tasks that cannot be handled by a single robot. For instance, some emergency situations, e.g. earthquakes and conflagrations, would require a team of rescue robots to work in a cooperative way to efficiently and robustly explore damaged scenes. In those cases, collaborative visual SLAM can significantly improve the capability of multi-robot explorations.

Project 91648204 supported by NSFC, project ZDYYJCYJ20140601 supported by NUDT, and project 201602-01 supported by HPCL.

I. Romdhani et al. (Eds.): CollaborateCom 2017, LNICST 252, pp. 480–490, 2018.
https://doi.org/10.1007/978-3-030-00916-8_45

Existing works on collaborative visual SLAM methods for multi-robots have generally fallen into the centralized and decentralized types. The centralized method tries to process and fuse all the local maps from the individual robots in a central server. For example, [3] proposes the visual C^2SLAM framework that provides the services of map optimization and storage in the cloud infrastructure. However, the clients cannot process the essential tasks of environmental mapping and highly rely on the central server. [4] proposes the Team SLAM method that aggregates local pose graphs from multiple robots into a global pose graph and feeds it back to the robots. However, it lacks global map optimization and may cause larger drifting errors. [5] proposes a monocular SLAM system employed on multiple unmanned aerial vehicles (UAVs). Its server manages the maps of all UAVs, and handles map fusing. The monocular cameras utilized in the system may cause larger errors due to the lack of scale information. On the other hand, decentralized methods reach global map consensus by sharing the local map of each individual robot via communication. [6] proposes the D-RPGO algorithm (distributed riemannian pose graph optimization) to perform collaborative localization by fusing the inter-time and inter-robot relative measurements to obtain an estimate of the absolute pose of each robot. [7] presents DDF-SAM, a decentralized data fusing approach for multi-robot SLAM by performing local SLAM and sharing a subset of map variables with neighbors in the form of summarized maps. [8] presents a frontier-based approach that uses a utility function to guide explorations, which takes the information gain and the distance costs of the robots into consideration. [9,10] use a distributed Gauss-Seidel (DGS) algorithm to reduce the information exchange among robots without a reliable communication infrastructure. Comparing to decentralized methods, centralized methods can support more complex operation in the central server.

In this paper, we present CORB-SLAM (collaborative ORB-SLAM), a centralized multi-robot visual SLAM system based on ORB-SLAM2 [2]. The ORB-SLAM2 is a versatile visual SLAM method that has been popularly applied in single-robot applications. However, this method cannot provide support to multi-robot cooperation in environmental mapping. The CORB-SLAM system consists of multiple ORB-SLAM2 clients for local mapping and a central server for global map fusing. Specifically, we extend each of the ORB-SLAM2 clients with a memory management module that organizes the local map and communicates with the central server. In the central server, we detect the overlaps of multiple local maps by the DBoW method [11], and fuse these maps by utilizing the Perspective-n-Point (PnP) [12] method and global optimization through bundle adjustment. We have evaluated the performance of the CORB-SLAM system with several systematic experiments on a public dataset KITTI [13].

2 The CORB-SLAM System

Considering scenarios of multiple robots collaboratively exploring an unknown environment, each robot agent in the CORB-SLAM system is designated to explore parts of the environment and build the local map.

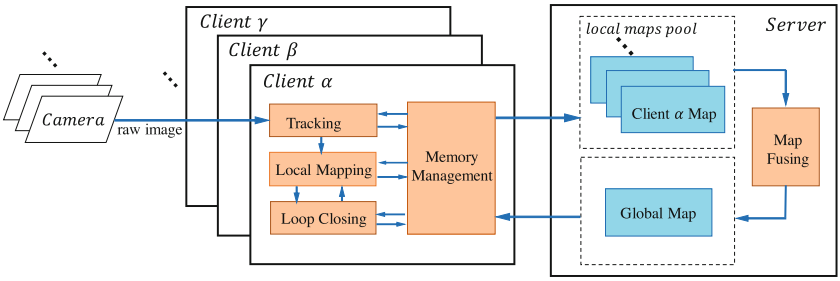


Fig. 1. The framework of CORB-SLAM system.

Figure 1 presents the framework of the CORB-SLAM system which consists of multiple ORB-SLAM2 clients and a server. Each robot is deployed with one SLAM client instance that builds a local map independently as the robot explores. The ORB-SLAM2 client is performs pose tracking, local mapping, and loop closing. We extend the original ORB-SLAM2 system by incorporating a novel memory management module for local map organization. All the modules in the SLAM client run in separate threads.

The central server receives the local maps from multiple clients and put them into the local-map pool. Global map fusing is achieved by three steps in a separate thread. Firstly, overlaps between two local maps are detected by the DBoW method to establish the spatial connections of all the local maps, when overlaps become available. Secondly, the local maps are merged into a global map through PnP and global BA. After the map fusing, the global map can be shared to each individual robot for further explorations.

2.1 The Robot-End SLAM Client

The robot-end SLAM client extends the ORB-SLAM2 to independently build the map of the environment around the robot, and thus, to facilitate autonomous exploration of the robot.

Tracking: The tracking module processes each raw image frame for pose tracking, and decides which frame should be inserted into the local map as a keyframe. ORB features [16] in raw images are used to match to existing map points to achieve pose tracking. When a new keyframe should be added, it will be transferred to the memory management module which schedules all map data.

Local Mapping: The local mapping module triangulates new map points from ORB features in keyframes. Moreover, it refines keyframe and map points in the local map, which is a sub-map of the client global map that affects current pose tracking and mapping, by using local bundle adjustment (BA).

Loop Closing: The loop closing module detects loop closures among keyframes, and refines the client global map using BA whenever a loop closure is found. All keyframes sharing similar ORB features with the new keyframe are used

to compute similarity scores to the new keyframe. Several keyframes with the highest scores are selected as loop candidates. The candidates whose similarity transformations are supported by enough inliers will be accepted as loop closures.

Memory Management: The memory management module stores local map and transfers the map data between the SLAM clients and the server in a bi-direction way. Specifically, when new keyframes and map points are created, this module transfers the new map data to the local-map pool within the server. When the poses in the local map are updated, all updated pose data will be transferred to the server in the forms of light keyframe and light map point, which only store the pose and identity data to reduce the communication bandwidth. The memory management module also receives the global map data that shared by the server. The global map is built in the server-end as will be described in Sect. 2.2. The SLAM client can use these map data to better facilitate exploring unknown environments, but is not allowed to update them in case of causing data inconsistency in other clients.

2.2 Map fusing in the server end

In the server, the map fusing module fuses local maps received from the SLAM clients, achieving an optimized global map. The map fusing algorithm is shown in the Fig. 2. It includes two main parts: map overlap detection and local-map fusion.

Initializing Global Map. Initially, the global map is set empty as the server system starts. The SLAM clients build the local maps independently without any initial information from other clients and the server. The local maps from different clients have the different reference coordinate system. When the server receives local maps from the SLAM clients, the first local map will be directly inserted into the global map, and thus, the global reference coordinate system can be decided.

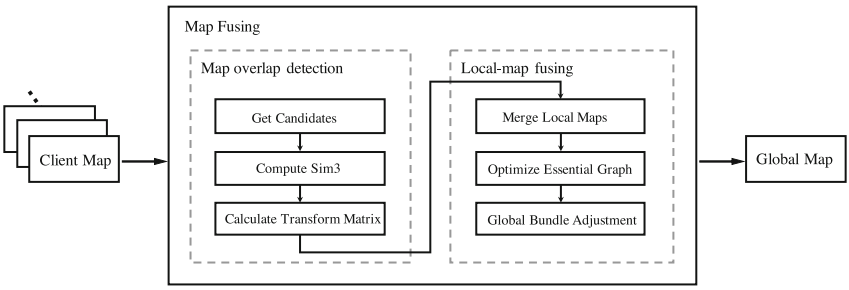


Fig. 2. The flowchart of Map Fusing module.

Map Overlap Detection. To determine the spatial connections among local maps in the server, we detect the overlaps among the local maps and further calculate the transform matrices using PnP method.

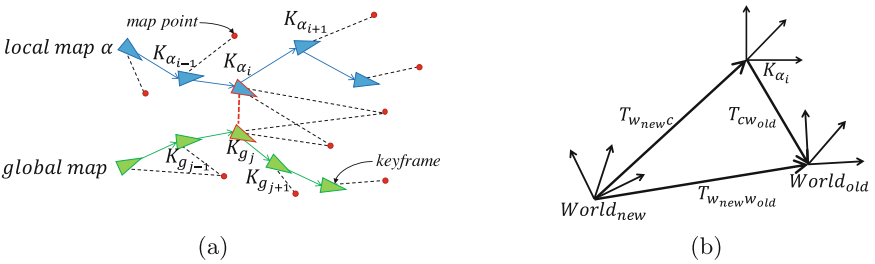


Fig. 3. In (a), the server detects the overlap between the *local map* α and the *global map*. (b) shows the transform relationships among the keyframe pose, the original and new world coordinate systems.

This module employs a bags of binary words (DBoW) approach [11] for map overlaps detection. The vocabulary is created offline with the ORB descriptors extracted from a large set of images and is loaded while system initializing [18]. The server incrementally builds a keyframe database for each local map and the global map. The keyframe database contains an invert index, which stores the ID of the keyframe in which each visual word in the vocabulary has been seen. Thus, querying the keyframe database can be done very efficiently. The keyframe database is updated when a keyframe is inserted or deleted from the map.

This module detects overlaps between local maps and the global map. It obtains a list of keyframes from the keyframe database of the global map according to the ORB features of the traversed keyframe in the operated local map. The keyframes in the list which have enough ORB feature matches will be accepted as rough candidates, as shown in the Fig. 3(a).

Then, it performs RANSAC iterations with all the rough candidates to calculate the 7 degrees-of-freedom transform matrix between the keyframe and the candidate, which is done in 3D similarity space (Sim3) by the method of Horn [14]. If we find a similar candidate, K_{α_i}, with enough inliers, we optimize it and perform a guided search of more correspondences. We optimize K_{α_i} again and, if it is supported by enough inliers, the overlap with K_{α_i} is accepted. Furthermore, the pose of the K_{α_i} in the global map, $T_{w_{new}c}$, is also calculated at the same time. As shown in the Fig. 3(b), if we know the pose of one keyframe in the global map (i.e., $T_{w_{new}c}$), the poses of all other keyframes in the local map of *client* α can also be calculated. This transform matrix is used to merge the local map of the *client* α to the global map.

Local-Map Fusing. To fuse local maps into a consistent global map, we optimize the essential graph in the server, and further optimize the global map by bundle adjustment (BA).

The local maps are merged into the global map by applying the previously calculated transformations to the keyframes and map points. Then, we perform a global 7 DoF optimization using BA to reduce map errors caused by different SLAM clients. Since global BA is time consuming, the optimization process only optimizes essential graph in the main thread, and performs global BA in a separate thread. Optimizing essential graph optimizes the poses of the keyframes in the global map and adds constraints by merging the duplicate map points in the overlapped maps. Global BA optimizes a graph by minimizing the re-projection error for all keyframes and map points that have been taken into account for the optimization. The implementation of the global BA uses the Levenberg-Marquardt implementation of g^2o [15].

2.3 Sharing the Globally Consistent Map

In the CORB-SLAM, the local maps built by the SLAM clients can be shared with the server and other clients. The map data, including keyframes and map points, will be transferred between clients and the central sever module when new map data is inserted or old map data is updated. Each keyframe mainly stores the camera pose, ORB features of the image frame, the IDs of all map points it observes and IDs of its co-related keyframes in the map. Each map point mainly stores its 3D position, its representative ORB feature, the IDs of keyframes that able to observe this map point. When new keyframes and map points are transferred, all the data in the keyframe and map point is packaged and transferred to the server or other clients. Furthermore, to reduce the bandwidth of updating poses in the map, we only transfer the light keyframe and light map point data which only stores the ID of the keyframe or map point and the pose or position of it.

When sharing map data among multiple robots, map data consistency, i.e. data values being consistent among robots, is taken into consideration. In this paper, we keep data consistency in all SLAM clients and the server by the following rules: (1) The SLAM clients share the newly created and updated data in local maps to the server in a particular rate. (2) The server share the newly created and updated data in the global map to all SLAM clients in a particular rate. (3) The SLAM clients can only update the map data created by themselves and keep the map data from the server and other clients fixed.

3 Experimental Results

The CORB-SLAM system is implemented in the Robot Operating System (ROS) [17]. The ROS provides a run-time environment that facilitates the client-server communication. Experiments are conducted to evaluate the performance of our CORB-SLAM system on the KITTI dataset in two different robot setup, i.e., two robots and three robots. The KITTI dataset contains stereo data sequences recorded from a car in different urban environments. The computer that runs the experiments is deployed an Ubuntu 14.04 64-bit operating system, an Intel core

i7-4790 (8 cores @ 3.6 GHZ) CPU and 8GB RAM. Although in the experiments of this paper all SLAM clients and the server run on one computer, each SLAM client is free to run on an individual robot, owing to the distributed communication mechanism of ROS. On the other hand, running all those processes on one computer can prove the real-time performance of the proposed system.

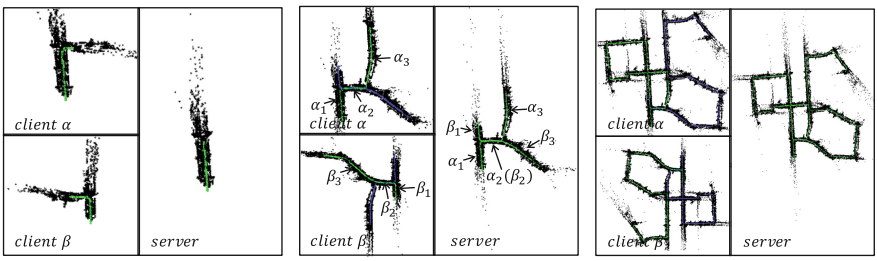
3.1 Customized Datasets

Without popular public datasets specially designed for multi-robot SLAM being available, we build our multi-robot datasets based on the single-robot KITTI dataset. We select the sequences 00 and 05 of the KITTI dataset and divide them into several sub-sequences according to the number of the robots involved in our CORB-SLAM system. Each SLAM client runs with a sub-sequence, and each sub-sequence should have overlaps with at least one other sub-sequence to support the map fusing algorithm.

We use the following method to produce the sub-sequences. We assume the time period of a KITTI sequence is $Seq.[0, t]$. For the two-robot case, we set $Seq.[0, \frac{t}{2} + \delta t]$, and $Seq.[\frac{t}{2} - \delta t, t]$ as two sub-sequences. For the three-robot case, we set $Seq.[0, \frac{t}{3} + \delta t]$, $Seq.[\frac{t}{3} - \delta t, \frac{2t}{3} + \delta t]$, and $Seq.[\frac{2t}{3} - \delta t, t]$ as three sub-sequences. The δt ensures the overlaps among the sub-sequences.

3.2 Experiments with Two Robots

In this experiment, the sequence 00 of the KITTI dataset is used for two-robot collaborative visual SLAM, after being divided into two sub-sequences. Figure 4 shows the procedures of exploring the sequence 00 on two SLAM clients.



(a) 10s (b) 80s (c) 400s

Fig. 4. The running snapshots of CORB-SLAM at different timestamp. In (a), (b), and (c), the green lines mark the poses of keyframes built by the clients themselves, and the blue lines mark the poses of keyframes from other clients. The black points are the map points. (Color figure online)

Figure 4(a) shows initial map status of CORB-SLAM at $t = 10s$. *Client* α and *client* β initialize their local maps independently in their own local coordinate

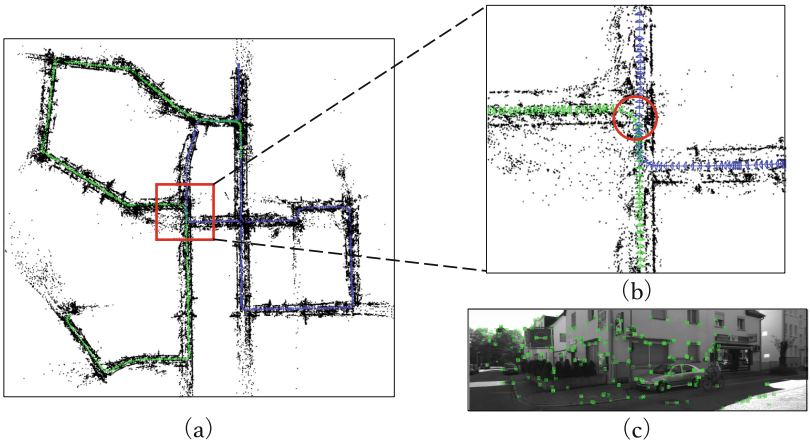


Fig. 5. (a) is the global map in *client* β. (b) is the detailed local zoom-in of the global map. In the street corner (the red circle area of (b)), *client* β builds the local map (green lines) and detects the loop closure in the map from the server (blue lines). (c) is the street-corner image from the camera in the red circle area of (b). (Color figure online)

system. The server has not detected an overlap between the two local maps, and thus, the two clients explore the environment independently.

In Fig. 4(b), the server has detected the overlap at $t = 80s$. *client* α has built the local map $\alpha_1\alpha_2\alpha_3$, and *client* β has built $\beta_1\beta_2\beta_3$. The server detects the overlap $\alpha_2 \cong \beta_2$, and fuses the local maps $\alpha_1\alpha_2\alpha_3$ and $\beta_1\beta_2\beta_3$ to global map $\alpha_1\beta_1\alpha_2(\beta_2)\alpha_3\beta_3$. Then, the server shares the global map with *client* α and *client* β. *client* α transforms the new map data $\beta_1\beta_2\beta_3$ to its own reference coordinate system and inserts $\beta_1\beta_2\beta_3$ to its local map as shown in the blue lines of *client* α, so does *client* β. Figure 4(c) shows the resulting map at $t = 400s$, and we present the detailed global map of *client* β in Fig. 5 as an example.

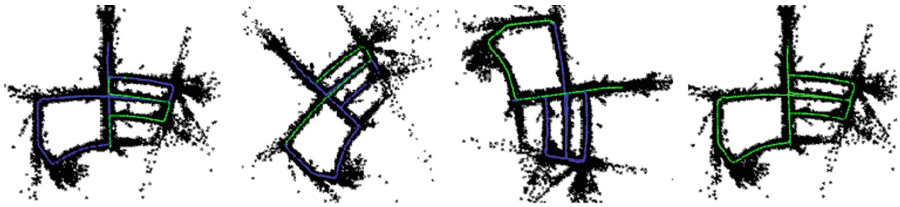
3.3 Experiment with Three Robots

In this experiment, the sequence 05 of the KITTI dataset is used for three-robot collaborative visual SLAM, after being divided into three sub-sequences.

The resulting maps in Fig. 6 show that, in the three SLAM clients case, each client can build its own local map and the local maps from the three different clients can be fused in the server and be shared back to different clients. The experiments with different numbers of robots also prove that the CORB-SLAM system is extendable for different scales of robot clients.

3.4 System Performance

There are some factors that can limit the scalability and the performance of the CORB-SLAM system, such as the communication, algorithm complexity in the



(a) *client* α (b) *client* β (c) *client* γ (d) *server*

Fig. 6. The resulting maps on the KITTI sequence 05.

server end, and robot hardware resource. In this paper, we evaluate time costs of the client-server communication and the map-fusion algorithm.

Table 1. Total time costs for clients-server communications on different datasets.

	Full Keyframe		Full Map Point		Light Keyframe		Light Map Point	
	Num.	Time	Num.	Time	Num.	Time	Num.	Time
Seq. 00	937	98.15 s	499681	14.02 s	1083	2.570 s	139418	6.045 s
Seq. 05	818	89.97 s	429938	12.10 s	1375	2.889 s	156466	7.969 s

Table 2. Time costs on merging local map and global BA

	Keyframes in map	Merging local map	Global bundle adjustment
Seq. 00	136	0.5346 s	1.955 s
Seq. 05	456	2.835 s	4.535 s

As shown in Table 1, the total time cost for transferring full keyframes is about 35 times more than that of the light keyframes, although the number of light keyframes is much larger than that of full keyframes. The average transferring time cost for one full keyframe is about 45 times more than that of the light keyframe. The reason behind these is that the ORB features in the full keyframe occupy most of the keyframe storage. We find that the average time cost of transferring the light map point is more than that of the full map point, which is resulted from the fact that the updating operation is more complex than the inserting operation.

As shown in Table 2, the time cost of map fusing in the server is mainly spent on the map merging operation and the global BA. The global BA costs much more time than the merging operation which can be indicated from Table 2.

4 Conclusions

This paper proposes the CORB-SLAM system, a collaborative multiple-robot visual SLAM for unknown environment explorations. Experimental results with

public KITTI dataset demonstrate that the CORB-SLAM system can perform SLAM collaboratively with multiple clients and a server end. The experiments are also shown in a video online[1].

References

1. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2007, pp. 225–234. IEEE (2007)
2. Mur-Artal, R., Tardos, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. arXiv preprint arXiv:1610.06475 (2016)
3. Riazuelo, L., Civera, J., Montiel, J.: C2TAM: a cloud framework for cooperative tracking and mapping. Robot. Auton. Syst. **62**(4), 401–413 (2014)
4. Deutsch, I., Liu, M., Siegwart, R.: A framework for multi-robot pose graph SLAM. In: IEEE International Conference on Real-time Computing and Robotics (RCAR), pp. 567–572. IEEE (2016)
5. Schmuck, P., Chli, M.: Multi-UAV Collaborative Monocular SLAM (2017)
6. Knuth, J., Barooah, P.: Collaborative localization with heterogeneous inter-robot measurements by Riemannian optimization. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 1534–1539. IEEE (2013)
7. Cunningham, A., Paluri, M., Dellaert, F.: DDF-SAM: Fully distributed SLAM using constrained factor graphs. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3025–3030. IEEE (2010)
8. Colares, R.G., Chaimowicz, L.: The next frontier: combining information gain and distance cost for decentralized multi-robot exploration. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 268–274. ACM (2016)
9. Choudhary, S., Carlone, L., Nieto, C., Rogers, J., Christensen, H.I., Dellaert, F.: Distributed trajectory estimation with privacy and communication constraints: a two-stage distributed Gauss-Seidel approach. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 5261–5268. IEEE (2016)
10. Choudhary, S., Carlone, L., Nieto, C., Rogers, J., Christensen, H.I., Dellaert, F.: Distributed mapping with privacy and communication constraints: lightweight algorithms and object-based models. arXiv preprint arXiv:1702.03435 (2017)
11. Gálvez-López, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Trans. Robot. **28**(5), 1188–1197 (2012)
12. Moreno-Noguer, F., Lepetit, V., Fua, P.: Accurate non-iterative o (n) solution to the PnP problem. In: IEEE 11th International Conference on Computer vision, ICCV 2007, pp. 1–8. IEEE (2007)
13. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013)
14. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. JOSA A **4**(4), 629–642 (1987)
15. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: a general framework for graph optimization. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 3607–3613. IEEE (2011)
16. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571. IEEE (2011)

[1] http://v.youku.com/v_show/id_XMjg5MTkyMzY3Ng.

17. Quigley, M., et al.:. ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5. Kobe (2009)
18. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans. Robot. **31**(5), 1147–1163 (2015)