Journal of Intelligent & Robotic Systems (2019) 94:389–404
https://doi.org/10.1007/s10846-018-0829-1

Multiple Maps for the Feature-based Monocular SLAM System

Huan Hu[1] · Hanxu Sun[1] · Ping Ye[1] · Qingxuan Jia[1] · Xin Gao[1]

Received: 1 March 2017 / Accepted: 27 March 2018 / Published online: 17 May 2018

Abstract
Monocular visual SLAM has become a popular research area in recent years because of its advantages of requiring low-cost hardware and providing high computational efficiency. This paper presents a multiple maps based SLAM system with four threaded architecture. When multiple maps are introduced the system works consistently in a large area for significant length of time . A new map is created automatically when tracking fails or the size of the map becomes too large. Tracking using multiple small maps reduces the accumulation of errors in visual SLAM to a significant extent when compared with a single map. To enhance the robustness of tracking an algorithm for predicting the camera pose is introduced . Exhaustive evaluations performed on publicly available indoor and outdoor datasets show that our system satisfies real-time requirements for robotics applications while providing superior accuracy to state-of-the-art systems.

Keywords Monocular SLAM · Visual SLAM · Robot vision · Multiple maps

1 Introduction

Monocular visual simultaneous localization and mapping (SLAM) is the problem of localizing the camera pose and building a consistent map at the same time using only a single camera while the robot moves in an unknown environment. In recent years, monocular visual SLAM has become a popular research area due to its low-cost hardware requirements, ease of installation and high computational efficiency.

1.1 Monocular SLAM

Most monocular SLAM methods introduced to date can be classified into two main groups. The first group consists of feature-based methods that estimate the camera pose by minimizing the reprojection errors of the corresponding image features. Two approaches within this group are the filter based approach and the key frame(KF) based bundle adjustment (BA) approach. Strasdat et al. proved that the KF based BA approach outperforms the filter based approach at the same computational cost [1]. Our work focuses only on the KF based BA approach. One of the more promising algorithms in this category is the Parallel Tracking and Mapping (PTAM) proposed by Klein and Murray [2]. The most notable characteristic of this system is that it runs tracking and mapping in two parallel threads to meet the requirements of real-time operation. This algorithm works quite well for small scale deployments. However, it is not suitable for large scale applications as it suffers from several limitations including: 1) relocalization with limited invariance to viewpoint and 2) error accumulation. To improve this, Mur-Artal et al. developed a system called ORB-SLAM [3] which proposed a relocalization algorithm with a high invariance to viewpoint and a loop closing method. The second group consists of direct methods that estimate the camera pose by minimizing the photometric error based on image intensity information. Engel et al. proposed a representative algorithm called LSD-SLAM [4] which was shown to work well in some textureless environments. However Mur-Artal showed that, compared to feature based methods, its accuracy is significantly lower in texture-rich environments [3]. Buyval et al. also verified this in the application of indoor navigation [5]. Recently Forster et al. proposed a semi-Direct Monocular Visual SLAM (SVO) [6] which combined the advantages of both feature based and direct methods. This algorithm

This work was supported by the National Natural Science Foundation of China (61305126).

✉ Huan Hu
wendyhh@bupt.edu.cn

Ping Ye
yeping@bupt.edu.cn

[1] Automation School, Beijing University of Posts and Telecommunications, Beijing, China

is promising because it runs at high frame rates and has been shown to perform well on several MAV(Micro Air Vehicle) datasets. However we encountered some difficulty in running this algorithm on several ground vehicle datasets because tracking failed in a very short time after map initialization. Younes et al. summarized the published research on non-filter based monocular SLAM systems [7]. As a result, we chose the feature-based monocular visual SLAM method for this project.

The key limitation with this method in a large scale application for long-term operation is the accumulation of errors. Various methods have been proposed to improve this through loop closure. Loop detection is the first step of loop closure. To the best of our knowledge, the appearance based loop detecting method is superior to others, such as map-to-map matching or image-to-map matching. The main representative work is FAB-MAP [8, 9] which detects the revisited place in the already mapped area by using image-to-image matching based on bag-of-words techniques. This algorithm has proven to be greatly successful in localization in outdoor areas where there are many point features. Another typical example is SeqSLAM [10] presented by Milford and Wyeth, which matches the current image with a sequences of images. One outstanding characteristic of this system is that it is suitable for use in extreme environmental conditions such as seasonal changes or inclement weather. In our application, we did not address the severe appearance changes. Instead, we continued to follow the approach of matching a single image. After loop detection, the pose-graph based optimization method is key to improving map accuracy for long-term operation. Kummerle et al. put forward a general pose graph method (g2o) [11] which introduced the loop edge into the pose-graph and then solved this nonlinear least squares optimization problem by using iterative techniques. Examining the general pose-graph method, Dubbelman proposed a COP-SLAM algorithm [12] based on a special sparse pose-graph "pose-chain" in which edges only exist between two consecutive camera poses or two loop poses. Another feature of COP-SLAM is the adoption of seven-DOF(degree-of-freedom) similarity transformation which adds one extra constraint to prevent the scale drift. This approach only corrected the camera pose inside the loop. Another similar approach was proposed by Mur-Artal [3] which also introduced scale estimation between two loop frames to remove the scale drift and subsequently propagate this scale to all other camera poses.

1.2 Multiple Maps in SLAM

Studies have shown that loop closure can cope with error accumulation and scale drift when the robot returns to the explored place(a loop). However, if no loop is detected or does not exist in the explored area, the problem of how to deal with the accumulated error is a key issue in increasing the accuracy of localization. It is known that the monocular SLAM easily accumulates error as the map grows. Therefore, we assumed that the bounded map size using multiple maps when doing exploration might be helpful. We verify this assumption in our experiments. Two types of work related to multiple maps are multi-session SLAM and sub-maps based SLAM. The multi-session visual SLAM proposed by MacDonald et al. adopted anchor nodes to combine multiple maps in a common coordinate system [13]. In this system one map corresponds to one single session defined by the user, such as a room, in a single session. Also, there are significant overlaps between maps. Our objective was to construct multiple maps automatically, instead of artificially dividing the environment into sub-regions. We also wanted to reduce the size of overlaps between adjacent maps. Holmes and Murray proposed one sub-mapping approach [14] which split the whole map into sub-maps according to time cost in global BA. If the time used in global BA reaches a particular threshold, a new sub-map is spawned. In their system multiple maps only existed for global BA and the system still maintained one map. Based on our experience, global BA does not always improve system accuracy in large scale applications. In 2015 Bourmaud and Megret released an off-line visual SLAM system with sub-map estimation from some consecutive key frames and the alignment of sub-maps based upon 3D relative similarity estimation [15]. Using the idea that the map is bounded in size, we created sub-maps online when the bounded value was reached. All the above mentioned multiple maps related systems attempt to obtain a globally consistent map. However we focused on constructing a system in which multiple maps coexist and the camera is localized automatically in one map. Another reason for using multiple maps is to maintain the consistency of tracking. After testing several open-source SLAM systems, we found that tracking failures were inevitable even when recovery strategies were applied. To cope with this issue, we built a new map when tracking failed then inserted this new map into the last one if the overlap was detected in a short time.

1.3 Recommendation

We developed this software to improve upon the state-of-art monocular SLAM system called ORB-SLAM [3]. Our recommended modifications to ORB-SLAM are:

– Build multiple maps instead of a single one, ensuring the persistency of localization. When tracking fails a new map is created automatically from the overlapped frames which were localized in the last map.

– Introduce a new thread for managing multiple maps. This separated thread maintains similar performance to that of the ORB-SLAM, because the extra function is running in a different thread. At any given time, only one map is activated. After each new key frame is inserted into the current activated map, it is relocalized in the other unactivated maps. If an overlap is detected, the map switch/fusion process is triggered based on the size of the current map.
– Improve the robustness of tracking. The tracker developed in paper [3] is easily lost especially when the camera undergoes rotation. Our system predicts the camera pose based on the decaying velocity model for translation estimation and the template-based image tracking method for rotation estimation.
– Enhance the accuracy of the map. In order to cope with the measurement error, the corresponding features are refined before triangulation. Forward sparse global bundle adjustment is performed to optimize a map in the loop closure thread.
– Compensate for the error accumulation by using multiple small maps. When the system explores an unknown scene for a long period of time, error accumulation significantly decreases the accuracy of localization. If loop closure is detected, scale consistency is checked to reduce the error. However if the camera keeps exploring the new area without going back to the previous area, this regular approach fails because there is no loop detected. The detrimental effect of the error accumulation worsens as the map gets larger. Therefore, our system limits the size of the map by constructing a new map if the current one has too many key frames.

1.4 Outline

The rest of the paper is structured as follows. We begin with an overview of our multiple maps based SLAM system in Section 2. The system consists of four core components which are tracking, single mapping, loop closing and multiple mapping. In Section 3, the method for localizing the camera pose is introduced. In Section 4 the proposed mapping algorithms are explained. A loop closing method (loop correction) to improve the system robustness is discussed in Section 5. Management of multiple maps is provided in Section 6. Finally, extensive evaluation of how our system interprets popular outdoor and indoor datasets was performed. The results are reported in Section 7. Concluding remarks are presented in Section 8.

2 System Overview

As previously mentioned, this paper proposes a monocular visual SLAM system that can localize the robot pose up to a scale factor and construct a map once the robot begins to explore the environment. When localization fails, a new map is automatically created. This new map initialization procedure starts with several explored image frames which have already been added to the last map. If there are enough features and good matching feature pairs in these past images, this new map is probably created successfully from these images, thus overlaps exist between these two adjacent maps. When the robot returns to a previously explored place, it is recognized as a loop and used to correct the map.

Figure 1 illustrates the system structure showing four threads running in parallel: tracking, single mapping, loop closing and multiple mapping. The single mapping thread maintains an activated map which is used for localizing. The tracking thread estimates the camera pose corresponding to each image frame and decides which ones can be defined as key frames. Once an image frame is considered to be a key frame, the single mapping thread inserts it into the currently activated map and extracts new map points observed in this key frame. Subsequently, the loop closing thread starts performing loop detection in the currently activated map with the goal of finding revisited places to correct the map.

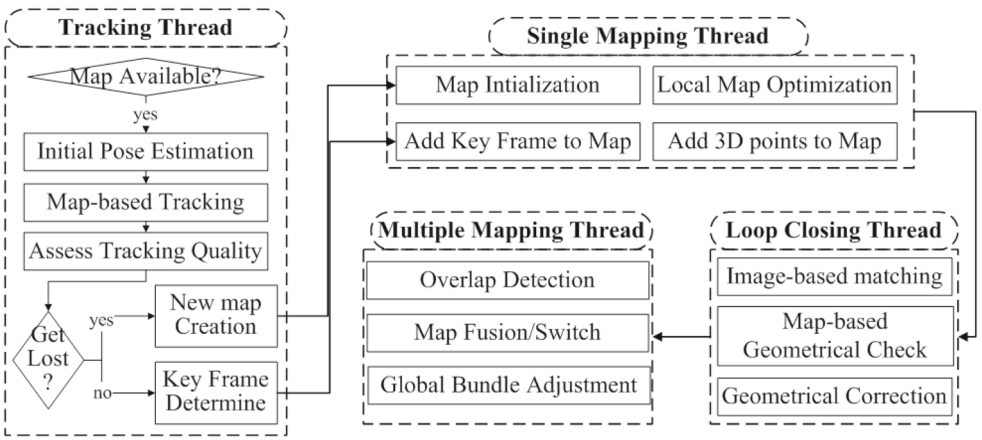


Fig. 1 The structure of multiple maps visual SLAM system. The four threads are run in parallel

If no loop is detected, the multiple mapping thread tries to find overlaps in the unactivated maps. If an overlap between the two maps is detected, this thread starts the map switch/fusion process. The tracking thread also determines when a new map is to be created. At any time only a single map is activated for tracking and single mapping purposes. Other unactivated maps are used for overlap detection in the multiple mapping thread. Among these four threads, the tracking thread has the highest priority and the multiple mapping thread has the lowest priority. This ensures that tracking meets the requirement of real time operation.

Our software was developed using several open source libraries. A binary feature [16] presented by Rublee et al. (ORB) is similar to an oriented multiple scales FAST(features from accelerated segment test) corner detector and an associated 256 bit descriptor. The ORB features are detected in each frame. This feature is highly invariant to viewpoint which is beneficial for matching from wide baselines. Further, the bag-of-words(BoW) [17] representation for an image proposed by Galvez-Lopez and Tardos is used to carry out feature matching. The visual words vocabulary with several levels is built off-line using the ORB descriptors. The vocabulary was extracted from a large number of images in various indoor and outdoor environments to ensure the generality of the vocabulary. Matching is carried out by first searching through the BoW representation and then performing ORB feature matching in the same node at a medium level of the vocabulary tree. This makes the matching process more efficient. An inverted index list for each visual word is used for storing key frames in which that word has been detected. This increases the speed of image matching during relocalization and loop/overlap detection. The key frames are connected as a pose-graph. The weight of each edge in the graph is dependent on how many visual words are shared by the two key frames connected by that edge. BA is used for map optimization where the pose-graph optimization technique implemented in g2o library [11] is adopted.

3 Tracking

The tracking thread is primarily responsible for estimating the camera pose in real-time, with the assumption that a map has already been built. The camera pose is initially predicted from the last camera pose and then refined by using the map. If the tracking quality in a particular frame is deemed to be good, this frame will probably be inserted into the map as a key frame. If tracking fails in less than five frames, global relocalization is used to predict the camera pose. If tracking fails for more than five consecutive frames, the tracker is considered to be lost and a new map is created.

3.1 Initial Pose Estimation

An initial camera pose is calculated when a new image frame arrives. The initial camera pose is only a rough estimate and is refined later by using the map-based tracking method. The following two methods shown in Fig. 2 are used for obtaining the initial camera pose.

If tracking quality of the last image frame was good, the initial pose is predicted based on the camera motion model and image rotation in 2D space. The camera motion model is used for estimating camera translation. The 2D image rotation is employed to estimate camera rotation in 3D space.

In our application the camera is mounted on a robot, so the camera movement is supposed to be smooth most of time. The camera motion model in our system is based on the decaying velocity motion model, expressed as:

$$V_t = 0.9 * \left(V_{t-1} + \frac{log_{SE3}\,(T_t \cdot T_{t-1})}{d_t} \right) /2 \tag{1}$$

where V_t and V_{t-1} are the velocity of camera at time t and t-1 respectively, T_t and T_{t-1} are the estimated camera pose at time t and t-1 respectively, d_t is the time interval between these two consecutive image frames. T is represented using the Lie Group SE3 and log_{SE3} is the logarithmic mapping

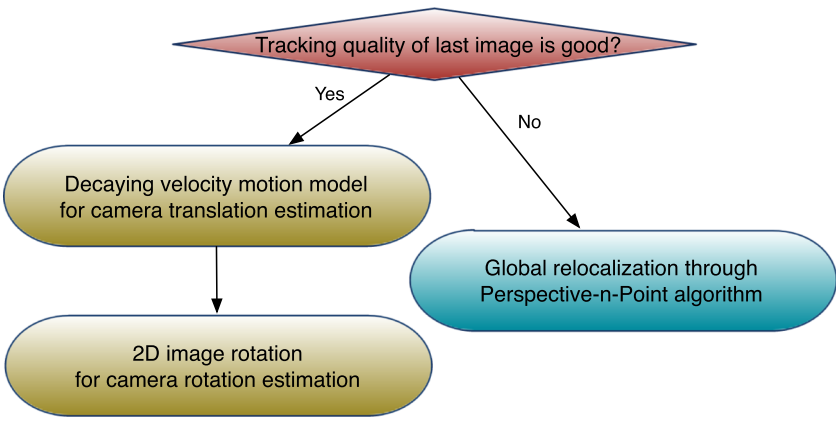


Fig. 2 The flowchart for roughly estimating the initial camera pose for a new image

operation in the corresponding Lie algebra [18], which converts a 4*4 rigid transformation matrix to a six degree vector. The average of the instantaneous velocity from the last two camera poses and the last velocity V_{t-1} is calculated to reduce the effect of camera shake. Considering that the estimated camera pose will slow down and eventually stop, this average is multiplied by a velocity damping factor (*0.9*).

The predicted camera pose is

$$\widetilde{T}_{t+1} = exp_{SE3}\left(d_{t+1}V_t\right) \cdot T_t \tag{2}$$

where exp_{SE3} is the exponential map that converts a six degree vector to a 4*4 rigid transformation matrix.

Because tracking fails easily when the camera is rotated, a rotation estimator is introduced into our system. This first estimates the 2D transformation in the image plane and then converts it to the 3D camera rotation matrix. The 2D transformation is estimated by using the template-based image tracking method (ESM) proposed by Benhimane and Malis [19]. This method uses an efficient second order approximation to minimize the photometric error. The transformation is initially set as an identity and then updated by the following equation:

$$\delta x \approx -2\left(J\left(e\right) + J\left(x_c\right)\right)^{+} \cdot \left(I\left(x_c\right) - I\left(e\right)\right) \tag{3}$$

where x represents the transformation in the image plane. The transformation consists of translation and rotation. e represents the identity transformation. The Jacobian J is the derivative of image intensity with respect to the transformation. $J\left(e\right)$ is constant and computed only once since it refers to the reference image. The Jacobian $J\left(x_c\right)$ which represents the first order derivative of the current image, is updated at each iteration. The initial estimation of x_c is the identity transformation matrix, $(\cdot)^{+}$ is pseudo-inverse, $I\left(x_c\right)$ and $I\left(e\right)$ are the image intensities in the current and reference images respectively.

The camera rotation in 3D space which causes the above 2D transformation is estimated by minimizing the reprojection error

$$\min f(\theta) = \sum_{i}^{N} \left(\delta u_i\right)^2 = \sum_{i}^{N} \left(u_1(p_i) - u_2(p_i)\right)^2 \tag{4}$$

The rotation is represented by a 3 degrees vector θ as in Lie algebra [18]. This minimization problem is solved by the Gauss-Newton method which iteratively updates $\delta\theta$:

$$\delta\theta = \left(J^T J\right)^{-1} J \cdot \delta u \tag{5}$$

where the Jacobian $J\left(\theta\right)$ is

$$J(\theta) = \frac{\partial u}{\partial \theta} = \begin{bmatrix} -\frac{xy}{z^2} f_x & f_x + \frac{x^2}{z^2} f_x & -\frac{y}{z} f_x \\ -f_y - \frac{y^2}{z^2} f_y & \frac{xy}{z^2} f_y & \frac{x}{z} f_y \end{bmatrix} \tag{6}$$

with f_x and f_y being the focal lengths of camera at x and y respectively. (x, y, z) is the 3D point coordinate in the camera reference frame. Our system selects 4 points around the image center in the template image and the matching points in the reference image are obtained by using the estimated 2D transformation. These points are back-projected into the camera coordinates to estimate the 3D camera rotation. Due to the requirement of real-time operation, the rotation estimator uses a down-sampled image whose resolution is less than 100 pixels in both x and y directions. The image is also normalized to have zero mean and blurred using Gaussian filter in order to reduce the influence of noise.

If the current tracking quality is evaluated as being poor, global relocalization is used to predict the new camera pose. The new image is represented using BoW feature representation. The candidate key frames are selected according to an inverted index list of the visual words. The correspondences are searched from candidate key frames by combining visual word matching with feature matching as described in Section 2. The predicted camera pose is calculated by using the Perspective-n-Point(PnP) algorithm [20] proposed by Lepetit et al. which estimates the camera pose from n 3D-to-2D point correspondences.

3.2 Coarse-to-Fine Map-based Tracking

Assuming that the initial camera pose has already been estimated, the map-based tracking procedure is to project all map points under this predicted camera view and then search for more point correspondences in adjacent areas to update the camera pose. Since searching on a large map is expensive, we maintain a local map which only contains key frames connected to the latest key frame in the pose graph. A two stage tracking approach is adopted in our system.

In the coarse tracking stage, the correspondences are looked for in a relatively large window. Only map points in the reference frame are used for coarse tracking. If the predicted camera pose is calculated from the motion model, the last frame is used as the reference frame. If it is calculated using PnP, the reference frame is the previous key frame used for relocalizing the current image frame.

In the fine tracking stage, local map points without matched ORB features are projected into the current frame. If enough inliers are found in the first tracking stage, searching is performed in a small search window. Otherwise the correspondences are looked for in a relatively large window.

The camera pose is finally optimized from all correspondences found in both stages by using motion-only BA which assumes that the estimations of map points are accurate.

3.3 Key Frame Determination

We tried to insert key frames as rapidly as possible to ensure that tracking is more robust to deal with challenging camera

movements such as rotations and fast motion. A frame is considered to be a key frame when the following conditions are met:

- Tracking quality is good. If less than 10 points are tracked in the current frame, tracking quality is considered to be poor. Otherwise, tracking quality is defined by the percentage of successfully tracked features at all possible map points. This percentage is formulated as $s = n/m$, where m is the total number of map points which are used for pose optimization and n is the number of inliers. If $s > 0.7$ the tracking quality is considered to be good.
- The camera is a certain distance away from other key frames already in the map. The distance between two key frames is normalized by the mean depth of observed features. The distance between the first two key frames is used as the threshold for determining if a frame is far enough away from the nearest key frame. Therefore, this rule is formulated as:

$$\frac{s_{ci}}{d_c} > \frac{s_{01}}{d_0} \tag{7}$$

  where s_{ci} is the linear distance between the current frame and the nearest key frame in space, s_{01} is the linear distance of the first two key frames while d_c and d_0 are the mean depths of the current frame and the first key frame.
- The key frame queue in the single mapping thread is empty. This ensures that key frames can be quickly inserted into the map. If this is the only condition that is not met, the tracking thread sends a signal to inform the mapping thread to stop the time-consuming map optimization process.

3.4 New Map Creation

Despite many efforts being made to develop a robust tracking system, in some challenging environments, tracking is inevitably lost. When tracking is lost, a new map is created in our system. If there are more than five frames which cannot be tracked correctly, tracking is defined as being lost and the signal to create a new map is sent. Once the signal is received the mapping thread immediately stops all other processing to start a new map creation procedure. Moreover, when the size of the map has reached the threshold, the new map creation procedure is also triggered. The initialization of a new map begins with several past image frames that were explored in the last map. If there are enough features in those images, this is done to ensure that an overlap exists between adjacent maps. Otherwise, our system keeps trying to build the new map until there are enough features detected in the images.

4 Single Mapping

The single mapping thread incrementally constructs a map which consists of key frames and 3D map points. The map is refined and expanded once a new key frame is received from the tracking thread. The map also maintains an undirected weighted pose graph which describes the connections between key frames. The weight of each edge in the graph depends on the number of common map points that are visible in the two key frames connected by that edge. After the new key frame and map points are inserted into the map, BA is performed from all local key frames where the map points in the new key frame were also observed. This module is summarized as following two parts: map initialization and map extension.

4.1 Map Initialization

The goal of map initialization is to obtain an initial set of 3D points which will be used for tracking. This work uses the method which estimates the fundamental matrix between two image correspondences and then triangulates 2D features into 3D map points. In order to realize the automatic initialization, the first image frame is considered as the first key frame. Four steps are performed for each subsequent incoming frame.

Firstly, a search is performed for the second key frame that meets the following conditions:

- There are enough correspondences within the first key frame. If there are insufficient correspondences the current frame is reset as the first key frame.
- There exists enough baseline distance between the current and the first camera pose for good depth estimation to be performed. Since base line distance is related to translation, the first step is to eliminate the influence of rotation in the image motion. Rotation is estimated from a downsampled image by using the rotation estimator described in Section 3.1. Then the features in the current image are reprojected as:

$$p_2' = K \cdot R_{cur}^{first} \cdot \left(K^{-1} \cdot p_2\right) \tag{8}$$

  where R_{cur}^{first} is the estimated current camera rotation relative to the first camera pose and K is the intrinsic camera matrix. This is followed by computing the reprojection error $\left|p_2' - p_1\right|$ which gives the image motion caused by camera translation. If the median of all reprojection errors from the correspondences is larger than 20 pixels, the current frame is considered to have enough base line distance relative to the first key frame.

Secondly, the fundamental matrix F_r^c is estimated based on epipolar constraint theory shown as $x_c^T \cdot F_r^c \cdot x_r = 0$. Here x_c and x_r are image coordinates of ORB features in the current frame and the first key frame respectively. In our implementation, an 8-point algorithm [21] is adopted with RANSAC(random sample consensus) search.

Thirdly, rotation and translation transformation are estimated from an essential matrix:

$$E_r^c = K^T \cdot F_r^c \cdot K \tag{9}$$

where K is the intrinsic camera matrix which is known through the offline calibration. Four possible rigid transformations are obtained through the singular value decomposition of E_r^c. For each such transformation, the 2D features are triangulated into 3D points and the reprojection error is computed and used to label inliers. The rigid transformation with the most inliers is considered as the current camera pose.

Finally, the initial map is constructed by triangulating the correspondences based on the above transformation matrix. Bundle adjustment is also performed to refine this map.

4.2 Map Extension

The map initially only contains two key frames. As the camera moves away from the initial area, the map needs to be continually extended in order to continue tracking. This extension is performed when a new key frame is provided by the tracking thread. Along with adding this key frame into the current map, the new map points also need to be put into the map. The ORB features in the new key frame which have not been matched with known map points are considered as candidates for the new map points. In order for these candidates to be added to the map, their correspondences must be found in another camera view. The closest key frame is selected as this second view and correspondences are established using epipolar search. The ORB features which lie along the epipolar line in the second view are compared with the candidate map points using the Hamming Distance Test on ORB descriptors [16]. If a match is found, the corresponding point is triangulated and inserted into the map.

Triangulation is based on the hypothesis that the two rays back-projected from the corresponding points intersect. This hypothesis is valid if and only if the correspondence satisfies the epipolar constraint. Since a measurement error often exists in practice, the ray tends to be skewed. Therefore, before performing triangulation, the measured locations of the corresponding points are corrected using a non-iterative algorithm [21] to minimize :

$$mins(t) = d(x, l(t))^2 + d(x', l'(t))^2 \tag{10}$$

where x and x' are the corresponding points in the two images, $l(t)$ and $l'(t)$ range over all choices of corresponding epipolar lines parametrized as a one-parameter family of lines and $d(\cdot)$ are the shortest distances from the points to the epipolar line. The foot of the perpendicular thus obtained is the corrected point which is substituted for the measured point. Finally, the 3D point is obtained using the direct linear transformation (DLT) [21] method which finds the solution to

$$\begin{bmatrix} u_1 \cdot t_1^{(3)T} - t_1^{(1)} \\ v_1 \cdot t_1^{(3)T} - t_1^{(2)} \\ u_1 \cdot t_1^{(2)T} - v_1 \cdot t_1^{(1)} \\ u_2 \cdot t_2^{(3)T} - t_2^{(1)} \\ v_2 \cdot t_2^{(3)T} - t_2^{(2)} \\ u_2 \cdot t_1^{(2)T} - v_2 \cdot t_1^{(1)} \end{bmatrix} \cdot P = 0 \tag{11}$$

where $x = (u_1, v_1)$ and $x' = (u_2, v_2)$ are the corrected projection positions of the 3D point $P = \begin{bmatrix} x & y & z \end{bmatrix}^T$ in the two images, $T_1 = \begin{bmatrix} t_1^{(1)} & t_1^{(2)} & t_1^{(3)} \end{bmatrix}^T$ and $T_2 = \begin{bmatrix} t_2^{(1)} & t_2^{(2)} & t_2^{(3)} \end{bmatrix}^T$ are the projection transformations.

BA is used to maintain map quality. Full BA, which processes all key frames and map points, becomes an increasingly expensive computation as map size increases. Significant time (in the order of tens of seconds) is required for a map with more than 150 key frames to converge. Therefore we adopted a local variant of BA. This variant only processes the key frames connected to the current key frame in the pose graph and map points seen by those key frames.

When the camera moves in the well-explored portions of the map, the mapping thread is relatively idle. It can then be used to improve the map by: 1) creating new measurements corresponding to the new features; 2) pruning redundant key frames in which 90% of map points have been observed in at least three other key frames; 3) dropping unreliable map points which have probably been poorly estimated. The unreliable map points have only two measurements, but are predicted to be visible in more than 10 key frames.

5 Loop Closing

Loop closing is a crucial process for enhancing the robustness of a SLAM system. Its aim is to detect whether or not the camera has returned to a previously visited location(marked as a loop) that is already present on the current map. If the loop is found, it tries to use this information to compensate for the accumulated error which usually exists in visual tracking. The loop closing algorithm that we used is summarized in the following three steps:

Firstly, the image-to-image matching process is run to detect revisited areas by using appearance information only. This procedure tries to find overlapping key frames which are not connected to the current key frame in the pose graph. Key frames that share any visual words with the current key frame are initially scored by the number of shared words. This score is then recomputed according to the sum of the scores of all connected key frames. Key frames whose scores are higher than 75% of the best score are selected as candidates for further validation.

Secondly, a map based geometrical check is carried out by computing the 3D similarity transformation between the current key frame and candidates from the previous step. The geometrical error is computed based on the corresponding map points. They are searched by employing the ORB-based matching method and are aligned by a similarity transformation. Compared to the 6 DoF(Degree of Freedom) rigid transformation, the similarity transformation factors in the scale drift, which can be expressed as

$$S = \begin{bmatrix} sR & t \end{bmatrix} \tag{12}$$

where $s \in \mathbb{R}^+$ is a positive scale factor while R and t respectively represent the rotation matrix and the translation vector in the rigid transformation $T = \begin{bmatrix} R & t \end{bmatrix}$. Given a set of 3D-to-3D correspondences, the similarity transformation is computed by using the closed-form solution presented by Horn [22]. In this work, the RANSAC method is used to find the best solution then it performs a guided search to optimize similarity transformation further. If a candidate key frame has enough inliers, it is considered to overlap with the current key frame and the loop closure is thus detected successfully. Otherwise, this candidate is considered as false loop.

Finally, geometrical correction is performed to improve the map if a loop is detected in the previous step. The current key frame is corrected by adding the scale factor into the translation vector as:

$$\widetilde{T}_c^w = \begin{bmatrix} R & \frac{1}{s}t \end{bmatrix} \tag{13}$$

After correcting the current key frame, one local BA, which only optimizes the connected key frames in the pose graph, is performed. The similarity transformation for each connected key frame i can be calculated using $S_i^w = S_i^c \cdot S_c^w$. Here, S_i^c is the similarity transformation between the current key frame and the connected key frame i which is computed using a rigid transformation T_i^c by leaving rotation and translation unchanged and setting the scale as $s = 1$. The rigid transformation is obtained by using $T_i^c = T_i^w \cdot \left(T_c^w\right)^{-1}$. The corrected pose $\widetilde{T}_i^w$ for key frame i is calculated by using Eq. 13.

Further, all associated map points existing in the current and connected key frames need to be corrected. The corrected map point j in the key frame i is calculated as follows:

$$\widetilde{p}_j^w = \left(S_i^w\right)^{-1} \cdot \left(T_i^w \cdot p_j^w\right) \tag{14}$$

where first project map points by using the non-corrected camera pose T_i^w and then projecting it back with the corrected similarity transformation S_i^w. After the above correction has been performed, the local map, which only includes the above key frames and map points, is optimized by using the full BA process. A new thread is also triggered to perform global bundle adjustment.

Due to the sparsity of loop closure detections, the loop closing thread is free for a large portion of the time. The map optimization is performed during this free time. In the single mapping thread, only the local map is optimized. To increase the accuracy of the map, global bundle adjustment is usually performed as the last step. Since global bundle adjustment is time consuming, the forward sparse global bundle adjustment (FBA) method is used instead. Compared to global BA, FBA only optimizes the forward pose tree which is extracted from the pose graph. This tree starts from the current key frame while the next key frame is at least five key frames apart from the last one and has more than 50 common words. The tree has been successfully created when no key frames meet the above criteria. FBA is performed once when no loop has been detected and several key frames have been added after the last time it was performed.

6 Multiple Mapping

The multiple mapping thread is responsible for detecting the overlaps and switching between maps. If the camera comes back to an area that has already been seen in an unactivated map, this is set as the currently activated map. The current map is either fused into the previous one or set as unactivated. This process consists of the following three steps:

1. Search all unactivated maps for overlaps: The method used for overlap detection is similar to that for loop detection in loop closing thread. Since the detection is performed in all previous maps, it becomes very time consuming as the number of maps increases. A map graph is therefore adopted in the proposed system to store correlations between the maps. These correlations depend on the number of mutual activities between the maps. When the new map is created, the correlations with the adjacently created map is initialized as one, the correlations with other maps is initialized as zero. Whenever a map ‘A’ is switched to or from a map ‘B’, the number of mutual activities between ‘A’ and ‘B’ is

increased by one. The search for overlap in previous maps is performed in descending order based on this map graph.

2. Switch to the previous map: If the number of key frames in the current map is larger than 50, the overlapped previous map is set as activated while the current one is set as unactivated. The remaining threads reset automatically and use the overlapped map. On the other hand, if the current map is small, it is fused into the overlapped map. The fusion method is similar to the correction method in the loop closing thread, with the main difference being that *all* key frames and map points in the current map are corrected, not only the connected ones.
3. Perform global BA in a new thread to optimize the merged map: Since global BA is a time consuming process, a new thread is created for this step. Since other threads are still running in parallel, some new key frames are probably inserted after global BA. These are updated with respect to the connected key frame which has already been optimized in the process of global BA.

Our system builds multiple maps to cope with tracking loss or the accumulation errors on the large map. Only one map at a time is activated and it is used for tracking and single mapping. In the case of tracking being lost, multiple maps improve the robustness and accuracy of the SLAM system. A small map has relatively small errors. But the accumulation errors on the large map may lead to the map becoming useless. Instead of building a large global map, our system constructs several small maps in order to limit accumulation errors and improve SLAM accuracy. The errors in the newly constructed map are not related to the last map. As the camera is localized in one single map, the errors in one map don't affect any other map.

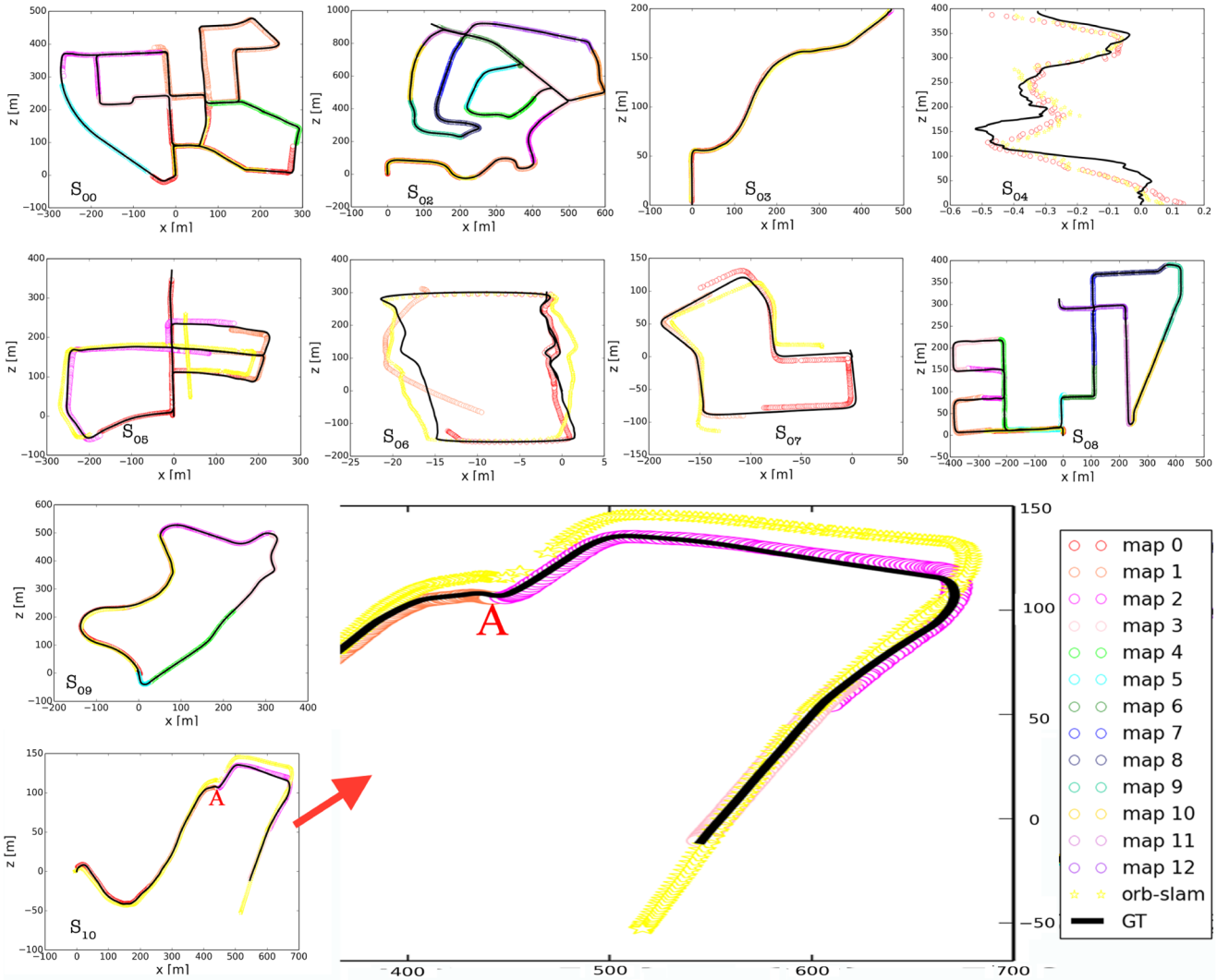


Fig. 3 Map Fusion results for all KITTI sequences, compared with ground truth. The trajectory from ORB-SLAM algorithm is also shown as yellow stars

7 Experiments

7.1 Setup

The proposed system runs on a desktop equipped with an Intel Core i7-3770K 3.5 GHz and a NVIDIA GeForce 7800GT graphic card. All tests were conducted using ROS (Robot Operating System) Indigo framework on Ubuntu 14.04. In the following experiments, 2000 ORB features [16] were extracted in the image at 8 different scales with a scale factor of 1.2.

7.2 System Verification on Outdoor Dataset

The KITTI dataset [23] provided by Geiger et al. was used to evaluate our system. The eleven sequences in this dataset are captured from a car equipped with a stereo camera driving around a residential area. The sequences have corresponding ground truth calculated by using GPS and a laser scanner. The camera trajectories estimated in our multiple maps based SLAM system are shown as Fig. 3 along with the ground truth on ten of eleven sequences.

Sequence 01 includes a highway which has few image features and a similar structure throughout. Consequently our system did not track well. The camera poses are aligned with the ground truth through a similarity transformation. It can be seen from the results that our system can maintain tracking during the entire sequence through restarting a new map automatically. The trajectories from running the ORB-SLAM algorithm on our setup (see Section 7.1) is shown by yellow stars in Fig. 3. For our hardware setup, ORB-SLAM only managed to complete a portion of the trajectory in most of the sequences such as S_{00}, S_{02}, S_{05}, S_{07}, S_{08}, S_{09}. However our system worked for long periods of time because multiples maps were created automatically. Even when the ORB-SLAM algorithm did manage to function for an entire sequence, the performance of our algorithm was superior. The reason is that our system can restrain the cumulative error by restarting a new map when the threshold has been reached. For example in S_{10} in Fig. 3, significant error accumulation exists in the latter part of the sequence when using ORB-SLAM. The enlarged view highlights this trajectory error in the latter part of the sequence. It shows that the trajectory estimated by the ORB-SLAM algorithm

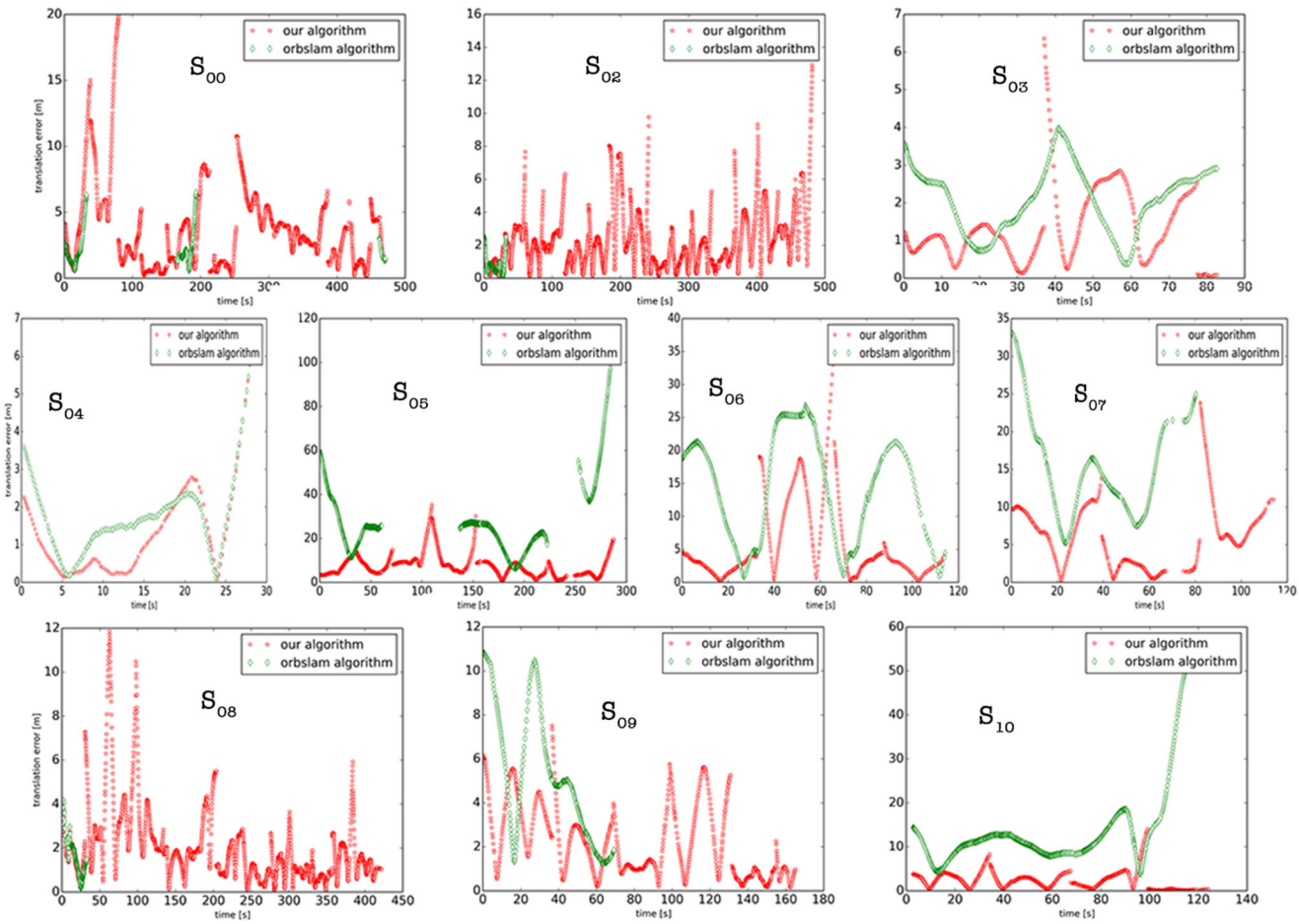


Fig. 4 Translation error for all KITTI sequences (*red*) comparing with ORB-SLAM algorithm (*green*)

Table 1 Translation error using multiple maps of the proposed system and a single map used by ORB-SLAM

	Multi-Maps					Single-Map				
	KFs	RMSE	mean	median	std	KFs	RMSE	mean	median	std
S00	1848	4.909	3.674	2.97	3.255	246	2.642	2.251	1.797	1.382
S02	2314	3.012	2.337	1.910	1.901	170	0.990	0.788	0.629	0.598
S03	317	1.734	1.347	1.093	1.091	286	2.336	2.161	2.277	0.887
S04	122	1.828	1.364	1.037	1.217	121	2.107	1.802	1.693	1.091
S05	1071	9.577	7.766	6.765	5.605	691	36.800	30.435	25.185	20.687
S06	487	8.921	6.172	3.421	6.441	418	16.697	14.682	15.975	7.953
S07	417	7.822	6.192	5.699	4.779	292	17.472	15.845	14.292	7.364
S08	1950	2.644	1.982	1.555	1.749	157	1.833	1.593	1.461	0.907
S09	800	2.656	2.097	1.400	1.631	291	5.843	5.011	4.794	3.005
S10	544	3.286	2.463	2.221	2.174	505	18.317	14.487	11.225	11.209

(shown as yellow star in Fig. 3) strays away from the ground truth as the map grows. However our system would have already started a new map from point 'A' that inhibited the accumulation of localization error.

Figure 4 shows the translation errors that occur when using our multi-maps algorithm and the ORB-SLAM algorithm. In some sub-figures, ORB-SLAM errors are not complete because of the loss of tracking. This comparison shows that our algorithm performs better in terms of accuracy.

Further, the absolute trajectory error (ATE) was computed. Table 1 shows the number of key frames and the RMSE(root-mean-square-error), mean, median and standard (std) error of the proposed system compared with ORB-SLAM. ORB-SLAM was only able to track the entire sequence in only four of ten cases (shown as green lines in Table 1) and even there its RMSE is higher than the multi-maps algorithm. The results show that error accumulates quickly as the number of key frames in the map increases. In order to improve the tracking accuracy so it can be used for a long term task, the proposed algorithm builds a new map whenever the existing map gets larger. For example in S_{10}, two maps are built in our system instead of a single one as in the case of ORB-SLAM. The localization error is much higher (up to 18.317m) mainly because of the error accumulation after 100 seconds when running the ORB-SLAM algorithm. Our algorithm, however, starts the second map at approximately 100 seconds and runs the localization within it. For the remaining six sequences, ORB-SLAM only managed to track a part of the sequence. The purpose of the proposed system is for it to be usable for a large scale robotics application and for it to be able to perform persistent long term tracking and mapping. Our results show its superiority over ORB-SLAM in this domain.

Figure 5 shows the feature points for initializing multiple maps in sequence S_{02}. The pairs shown in green are used for computing the initial pose. The ones shown in red are discarded when computing the fundamental matrix.

Figure 6 shows map fusion results from running sequences S_{00}, S_{02} and S_{05} respectively. In our system

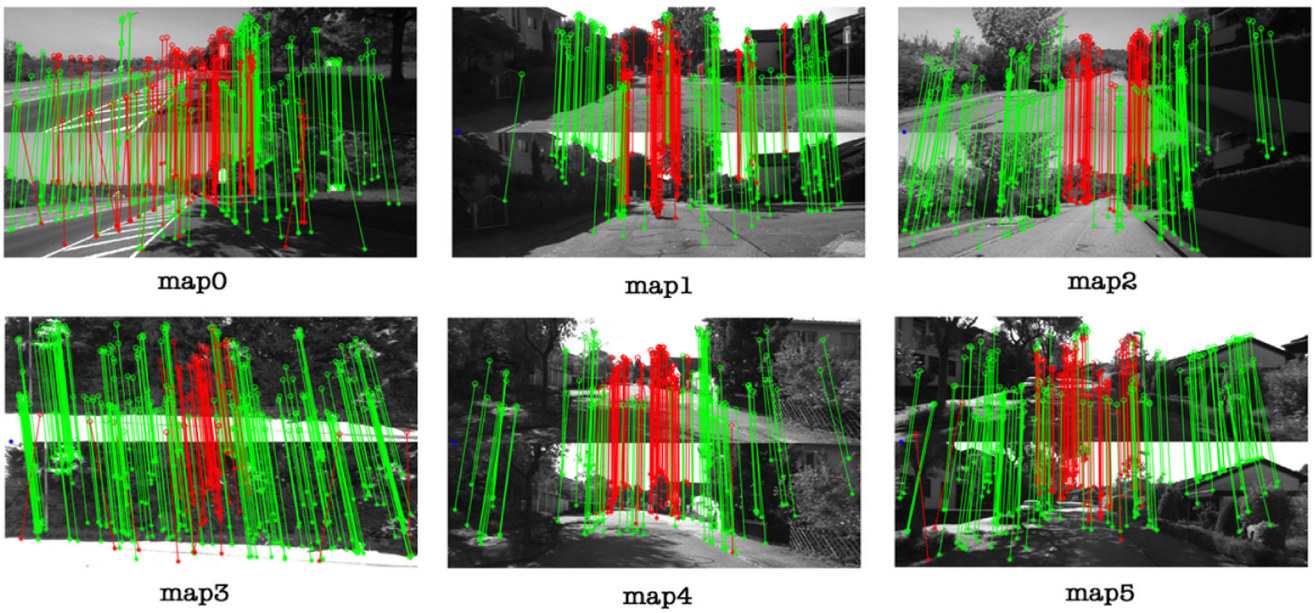


Fig. 5 Feature pairs used for building multiple maps. The green pairs are used to build the initial map

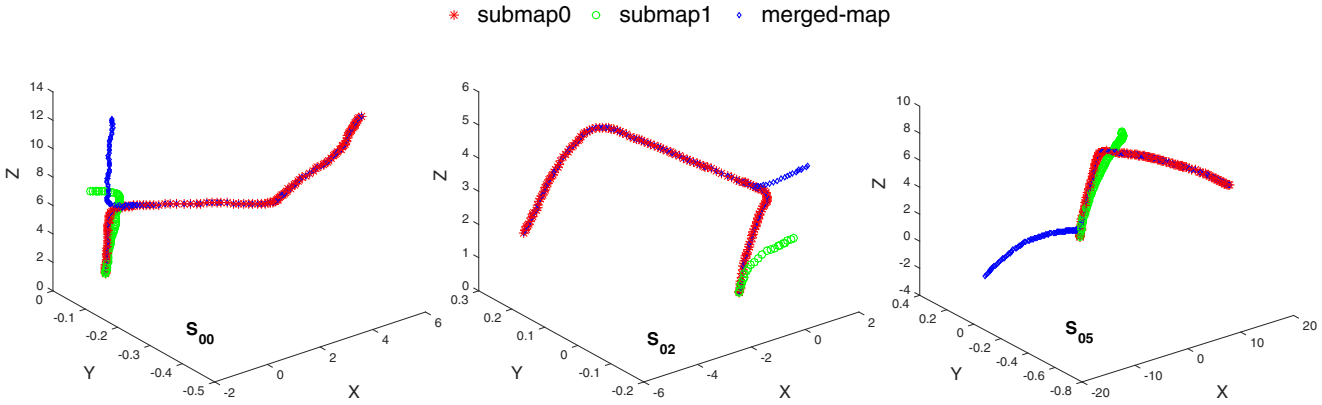


Fig. 6 Map Fusion. The map shown in green is merged into the map shown in red. The merged map is shown in blue

only the small map shown in green in Fig. 6 with less than 50 keyframes is considered for merging into another overlapped map shown in red in Fig. 6.

Figure 7 shows the map switch results from running a larger dataset which combines three longest KITTI sequences twice in the following oder: S_{00}-S_{02}-S_{08}-S_{00}-S_{02}-S_{08}. In one run, there are 7932 key frames in total and 21 maps are created. Loop exists in both sequences S_{00} and S_{02}. There is no loop in sequences S_{08}. For sequence S_{02} and S_{08}, maps switch for the second running is the same as it is in the first running. In sequence S_{00}, the map switch is slightly different between the first run and the second run because of the wider overlap between maps. This result verified that the system is capable of switching to the previous map when the camera returns to a revisited place.

The proposed algorithm significantly improves the initial pose estimation when compared to ORB-SLAM. This, in turn, is beneficial to the robustness of tracking. To verify this, we counted the number of created maps by using different initial pose estimation methods (shown in Table 2). In this experiment, the map size is set as unlimited. The higher the number of maps created, the less robust was the tracking. This conclusion is based on twenty runs for each sequence. The results show that our new initial pose estimation method is more robust than ORB-SLAM.

Figure 8 shows the time spent in running each individual thread in the proposed system. Events that are more time consuming are labeled as 'e0-3'. For sequences S_{00} and S_{08}, the map size is not limited and only one map is created so that the multiple mapping thread is not run. e0 is the event for closing a loop. A time-consuming process in loop closing is FBA which only chooses a few key frames for optimization. For sequence S_{02} three maps are created. e1 is the event for map switching which includes performing corrections to the current map. e3 is the event for creating a new map which is constructed from the overlapped frames seen in the last map. The bottom right figure shows the time cost of sequence S_{09} which we ran three times. e2 is the event for map fusion which merges the current small map to the overlapped map. Since tracking was run in parallel to these events, the results show that tracking can still meet the requirements of real-time application.

Figure 9 shows the distribution of time cost for estimating camera poses in tracking for all test sequences. The figure on the left shows the time distribution obtained from all data. Occasionally, tracking takes much longer because other threads are correcting map data which is shared by the four threads. The figure on the right shows the time distribution from part of the data in which time cost is less than 100 seconds. From the comparison, we can see that

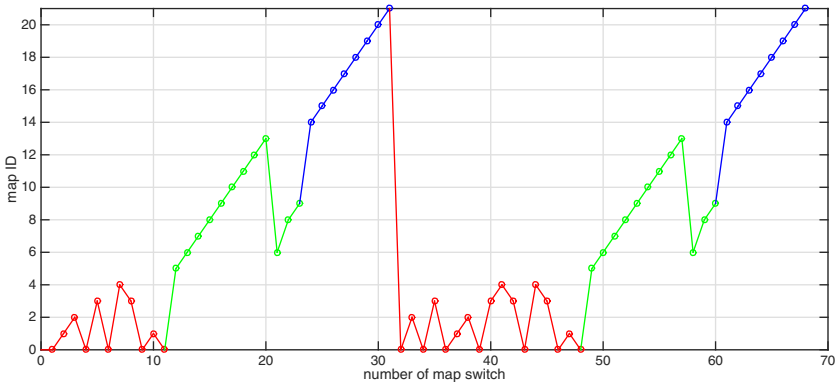


Fig. 7 Map Switch. Red for running sequence S_{00}. Green for running sequence S_{02}. Blue for running sequence S_{08}

Table 2 Average number of created maps using different initial pose estimation methods

	S00	S02	S03	S04	S05	S06	S07	S08	S09	S10
this paper	1	2.1	1	1	1.5	1	1	2.3	1.9	1
ORB-SLAM	3	7.9	1	1	2	1	2	4.8	2.2	1

the performance of our system is close to the ORB-SLAM system that only worked for a small part of long video sequences when running on our machine setup. Note that the sample size for our system is 22007, compared to 7144 for ORB-SLAM. This is because often ORB-SLAM only ran in parts of video sequences (shown as Fig. 3). Generally, this distribution shows that tracking could maintain around 15-30 Hz, which is suitable for most robotics applications.

7.3 System Verification on Indoor Dataset

The TUM RGBD benchmark [24] provides a good indoor dataset for evaluating the accuracy of camera localization along with the ground-truth camera poses obtained from a high-accuracy motion-capture system. In the light of our focus on robotics application, we choose four sequences which are recorded by a Kinect camera mounted on top of a Pioneer robot. The robot is manually controlled by a joystick to move through a large hall. It is more challenging to use this dataset because of the serious camera shake and many more poor-feature areas such as blank walls. Figure 10 shows the translation error for every sub-map. From this figure we can easily see that tracking fails when the translation errors are increasing quickly. Our system built a new map automatically when tracking failure had been detected. In this experiment a blank space often exists between the adjacent maps, such as the significant interrupt between sub-map 1 and sub-map 2 in

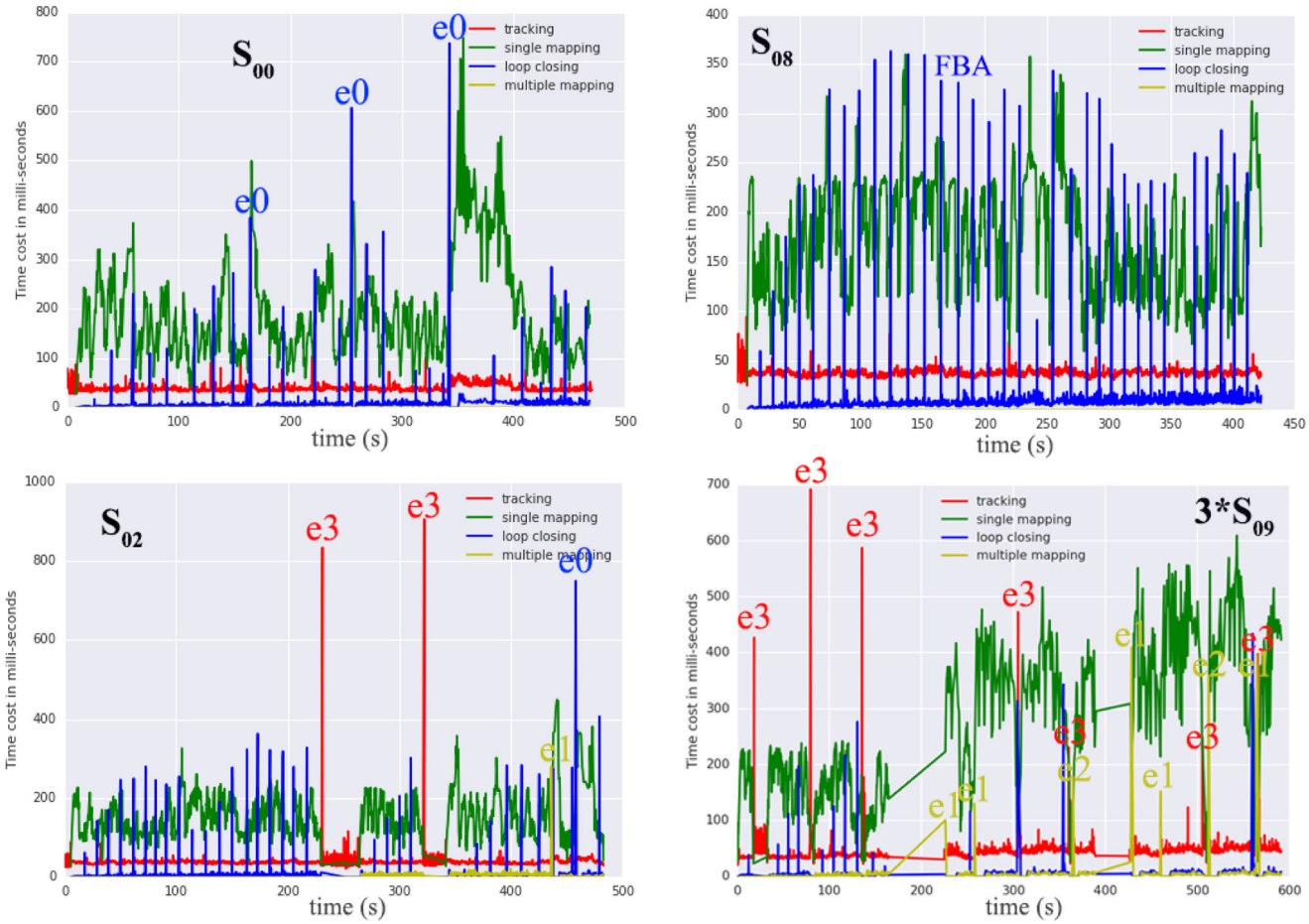


Fig. 8 Time cost for tracking, single mapping, loop closing and multiple mapping threads. e0: loop closure in loop closing thread. e1: map switch in multiple mapping thread. e2: map fusion in multiple mapping thread. e3: new map creation in tracking thread

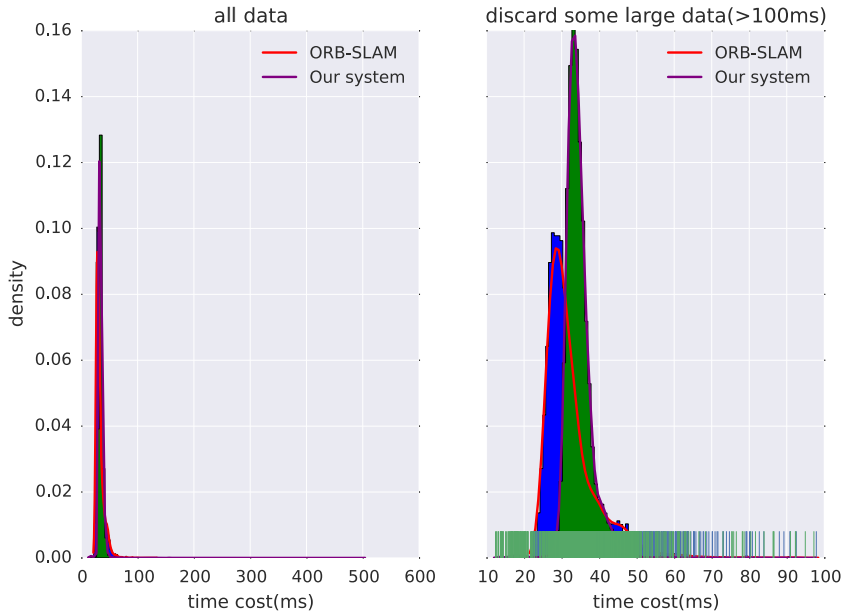


Fig. 9 Distribution of time cost for tracking

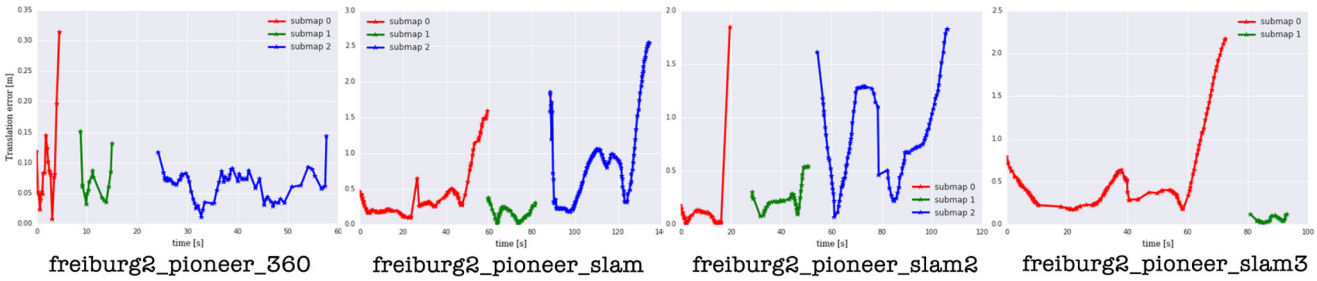
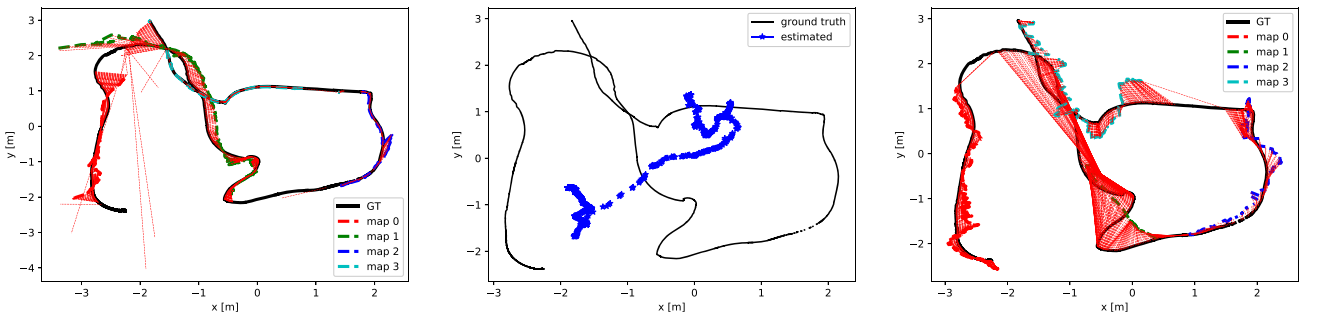


Fig. 10 Translation error in four sequences recorded by a Kinect camera mounted on a moving Pioneer robot

Table 3 The RMSE error of the proposed system compared with RGBD-SLAM algorithm

	Dimension	Proposed System	RGBD-SLAM
fr2_pioneer_360	4.24m*4.38m*0.06m	0.080536	0.912504
fr2_pioneer_slam	5.50m*5.94m*0.07m	0.805553	1.165072
fr2_pioneer_slam2	5.29m*5.25m*0.07m	0.676970	1.723923
fr2_pioneer_slam3	4.98m*5.34m*0.07m	0.669821	0.908619



(a) multi-maps in our system (b) the map in RGBD-SLAM (c) divided-maps in RGBD-SLAM

Fig. 11 Trajectory of camera estimated by our system comparing with the RGBD-SLAM system

the 'freiburg2_pioneer_360' dataset. This is mainly because there were not enough features detected.

Table 3 shows the RMSE error of camera poses estimated by our system compared with the estimate from the RGBD-SLAM system [24] which is provided along with the dataset. The RGBD-SLAM algorithm first searches matching feature points from color images and then combines the depth information to compute a set of point-wise 3D correspondences. The camera poses are estimated from these 3D correspondences by using RANSAC and ICP algorithms. The camera poses for the RGBD-SLAM algorithm were obtained from the authors' website. However, the camera poses estimated from our algorithm are unscaled. In comparison, the absolute scale is found through an exhaustive trial and error process to iteratively search for the minimum trajectory error. The result shows that the accuracy of our algorithm is higher than the one in the RGBD-SLAM. The RGBD-SLAM algorithm estimates the camera pose based on the previous image frames, but our algorithm uses the map-based method which re-projects the explored map points into the current image and then computes the camera pose by optimizing the re-projection error. The former often causes significant error accumulation. However, our tracking algorithm is recognized as being lost and simultaneously a new map is created to maintain system function when the errors have severely accumulated.

Taking the 'fr2_pioneer_slam2' dataset as an example, Fig. 11a shows the camera trajectory estimated by our system, which built four maps. The RGBD-SLAM system only built one map as shown in Fig. 11b. Because of the error accumulation, the estimated camera pose often drifts. This drift is also induced by the scale change which is not estimated inside the SLAM algorithm. In order to compare the RGBD-SLAM system with our multiple maps, we manually divided the whole map into four sub-maps which have different scales when aligned with the ground truth. However the error in the RGBD-SLAM system is still larger than in our proposed system. This result verified that the introduction of a new map can efficiently inhibit error accumulation.

8 Conclusions

This paper proposes a multiple maps based monocular SLAM system which can automatically create a map when the current map becomes too large or the tracking quality becomes poor. This system was evaluated on publicly available indoor and outdoor datasets. Our results show that our system can localize the robot in real-time. The advantage of our system is that localization works on a long term basis by automatically restarting a new map to cope with the loss of tracking. The accuracy of our system was also shown to be superior to the state-of-art monocular SLAM systems.

Acknowledgements We thank the authors of [23] for releasing the KITTI dataset and the authors of [24] for sharing their dataset and experiment results.

References

1. Strasdat, H., Montiel, J.M., Davison, A.J.: Visual SLAM: why filter? Image Vis. Comput. **30**(2), 65–77 (2012)
2. Klein, G., Murray, D.: Parallel Tracking and Mapping for Small AR Workspaces. In: The 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007), Nara, Japan, pp. 225–234 (2007)
3. 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)
4. Engel, J., Sch, T., Cremers, D.: LSD-SLAM: large-Scale Direct Monocular SLAM. In: 13th European Conference on Computer Vision (ECCV), Zurich, Switzerland, pp. 834–849 (2014)
5. Buyval, A., Afanasyev, I., Magid, E.: Comparative analysis of ROS-based monocular SLAM methods for indoor navigation. International Conference on Machine Vision (2017)
6. Forster, C., Pizzoli, M., Scaramuzza, D.: SVO: fast semi-direct monocular visual odometry. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 15–22 (2014)
7. Younes, G., Asmar, D., Shammas, E.: A survey on non-filter-based monocular Visual SLAM systems (2016)
8. Cummins, M., Newman, P.: Appearance-only SLAM at large scale with FAB-MAP 2.0. Int. J. Robot. Res. **30**(9), 1100–1123 (2011)
9. Glover, A., et al.: OpenFABMAP: an Open Source Toolbox for Appearance-Based Loop Closure Detection. In: IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, USA, pp. 4730–4735 (2012)
10. Milford, M., Wyeth, G.: SeqSLAM: visual route-based navigation for sunny summer days and stormy winter nights. In: IEEE International Conference on Robotics and Automation, pp. 1643–1649 (2012)
11. Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o: a general framework for graph optimization. In: IEEE International Conference on Robotics and Automation, pp. 3607–3613 (2011)
12. Dubbelman, G., Browning, B.: COP-SLAM: closed-Form online Pose-Chain optimization for visual SLAM. IEEE Trans. Robot. **31**(5), 1194–1213 (2015)
13. McDonald, J., et al.: Real-time 6-DOF multi-session visual SLAM over large-scale environments. Robot. Auton. Syst. **61**(10), 1144–1158 (2013)
14. Holmes, S.A., Murray, D.W.: Monocular SLAM with conditionally independent split mapping. IEEE Trans. Pattern Anal. Mach. Intell. **35**(6), 1451–63 (2013)
15. Bourmaud, G., Megret, R.: Robust large scale monocular visual SLAM. Computer Vision and Pattern Recognition, 1638–1647 (2015)
16. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: IEEE International Conference on Computer Vision, pp. 2564–2571 (2011)

17. Galvez-Lopez, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Trans. Robot. **28**(5), 1188–1197 (2012)
18. Blanco, J.L.: A tutorial on se3 transformation parameterizations and on-manifold optimization. In: University of Malaga, Spain (2010)
19. Benhimane, S., Malis, E.: Real-time image-based tracking of planes using efficient second-order minimization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 943–948 (2004)
20. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPNp: an accurate O(n) solution to the PnP problem. Int. J. Comput. Vis. **81**(2), 155–166 (2009)
21. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2003)
22. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. J. Opt. Soc. Am. A **4**(4), 629–642 (1987)
23. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013)
24. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 573–580 (2012)

Huan Hu is a Ph.D. student from Beijing University of Posts and Telecommunications. Her major is Mechanical and Electronic Engineering. Her research focuses on robotics, robot vision, virtual reality and augmented reality.

Hanxu Sun is a Professor from Beijing University of Posts and Telecommunications. He is a director of space robot research center hold by Chinese education ministry, also a director of Chinese Journal of Mechanical Engineering. His research areas are space robot technology and mobile robot technology.

Ping Ye is an Associate Professor from Beijing University of Posts and Telecommunications. He received his doctor's degree in mechanical engineering and automation from Beijing University of Aeronautics and Astronautics, in 2007. His research interests include the embedded image processing, visual localization, and navigation.

Qinxuan Jia is a Professor from Beijing University of Posts and Telecommunications. He has the membership of IEEE, Chinese Institute of Aeronautics, Chinese mechanical engineering society. His research areas are robot technology, the embedded control technology, virtual reality.

Xin Gao is an Associate Professor from Beijing University of Posts and Telecommunications. He received his doctor's degree from Beijing University of Aeronautics and Astronautics, in 2007. His reasearch focus on robot control, teleoperation, machine learning and artificial intelligence.