Robotics and Autonomous Systems 93 (2017) 116–134

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

# Multi-camera visual SLAM for autonomous navigation of micro aerial vehicles

Shaowu Yang [a,b,*,1], Sebastian A. Scherer [c], Xiaodong Yi [a,b], Andreas Zell [c]

[a] State Key Laboratory of High Performance Computing (HPCL), National University of Defense Technology, China
[b] School of Computer, National University of Defense Technology, Changsha, China
[c] Department of Computer Science, University of Tübingen, Tübingen, Germany

## HIGHLIGHTS

- A SLAM system integrating measurements from multiple cameras for MAVs is proposed.
- No overlap in the respective fields of view of the multiple cameras is required.
- Robust pose-tracking can be achieved in complex environments.
- Mathematical analysis on the iterative optimizations in visual SLAM is provided.
- The efficiency of the proposed visual SLAM system is demonstrated onboard of MAVs.

## ARTICLE INFO

*Article history:*
Available online 12 April 2017

*Keywords:*
Visual SLAM
Multiple cameras
MAVs
Autonomous navigation

## ABSTRACT

In this paper, we present a visual simultaneous localization and mapping (SLAM) system which integrates measurements from multiple cameras to achieve robust pose tracking for autonomous navigation of micro aerial vehicles (MAVs) in unknown complex environments. We analyze the iterative optimizations for pose tracking and map refinement of visual SLAM in multi-camera cases. The analysis ensures the soundness and accuracy of each optimization update. A well-known monocular visual SLAM system is extended to utilize two cameras with non-overlapping fields of view (FOVs) in the final implementation. The resulting visual SLAM system enables autonomous navigation of an MAV in complex scenarios. The theory behind this system can easily be extended to multi-camera configurations, when the onboard computational capability allows this. For operations in large-scale environments, we modify the resulting visual SLAM system to be a constant-time robust visual odometry. To form a full visual SLAM system, we further implement an efficient back-end for loop closing. The back-end maintains a keyframe-based global map, which is also used for loop-closure detection. An adaptive-window pose-graph optimization method is proposed to refine keyframe poses of the global map and thus correct pose drift that is inherent in the visual odometry. We demonstrate the efficiency of the proposed visual SLAM system for applications onboard of MAVs in experiments with both autonomous and manual flights. The pose tracking results are compared with ground truth data provided by an external tracking system.



## 1. Introduction

In the last decade, we have seen a growing interest in micro aerial vehicles (MAVs) from the robotics community. One of the reasons for this trend is that MAVs are potentially able to efficiently navigate in complex 3D environments with different types of terrains, which might be inaccessible to ground vehicles or large-scale unmanned aerial vehicles (UAVs), e.g. in an earthquake-damaged building [1]. A basic requirement for MAVs to autonomously operate in such environments is their robust pose tracking abilities, which is still a challenging task when the environment is previously unknown and external signals for providing global position data are unreliable. Meanwhile, if a map of the environment can be built, it will be able to provide support to path planning of autonomous navigation of the MAV [2]. Recently, more focus has been on using onboard visual solutions to address these issues, especially using visual simultaneous localization and mapping (SLAM) systems.

* Corresponding author at: State Key Laboratory of High Performance Computing (HPCL), National University of Defense Technology, China.
*E-mail address:* shaowu.yang@nudt.edu.cn (S. Yang).

[1] This work was mainly carried out at the Department of Computer Science, University of Tübingen. Project 61403409 and project 91648204 supported by NSFC.

http://dx.doi.org/10.1016/j.robot.2017.03.018
0921-8890/

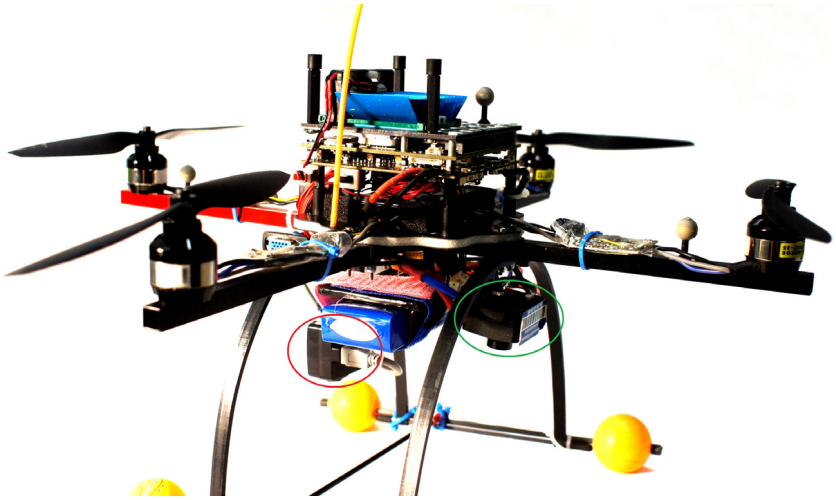

**Fig. 1.** Our MAV platform, with two cameras facing two different directions: downward (in green ellipse) and forward (in red ellipse). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Although we have seen successful applications of visual SLAM on ground vehicles [3,4], there are more challenges in using visual SLAM to enable autonomous navigation of MAVs. First, the payload of an MAV is rather limited. This usually leads to *limited onboard computational capability*, which requires the visual SLAM system to be very efficient, especially in large-scale operations. Second, due to the 3D maneuverability of an MAV, *robustness of pose tracking* is highly desired. When pose tracking fails, unlike a ground vehicle which may be able to maintain stability simply by ceasing moving, an MAV is likely to run into catastrophic situations with its position control getting lost.

In order to achieve more robust pose tracking of visual SLAM for MAVs, previous work has made much effort in fusing data from multi-modal sensors, e.g. fusing inertial measurements [5–8]. Recently, the robotics community has shown a growing interest in improving the performance of visual SLAM by utilizing multiple cameras [9–11]. Pose tracking of a monocular visual SLAM system may easily fail in complex environments when very limited number of visual features can be observed, due to its limited field of view. This also applies to typical stereo visual SLAM, which employs cameras looking in one specific direction. The FOV can be enlarged by using a lens with a wider viewing angle (even fish-eye lens), but at the cost of suffering from larger lens distortion and loss of environmental details, due to a smaller angular camera resolution. This also applies to catadioptric omnidirectional vision systems [12]. Another type of omnidirectional vision systems combines multiple cameras into one vision system, maintaining a single-viewpoint projection model. However, these cameras need to be very precisely configured within relatively heavy mechanical systems in order to preserve this model, and thus are not flexible enough for MAV applications. Obviously, larger effective FOV of a vision system can be obtained by integrating multiple cameras in it. This implies that better pose tracking robustness could be achieved by extending monocular visual SLAM to utilize measurements from multiple cameras. The challenge is how to utilize those cameras in SLAM efficiently and flexibly.

In this paper, we investigate multi-camera visual SLAM for robust pose tracking and environmental mapping. In the proposed visual SLAM system, multiple cameras can be mounted pointing to different directions, so that more reliable visual features can be observed. Our method allows a SLAM system to integrate images captured from various useful perspectives without requiring the cameras to be mounted in a specific way in order to keep a single-viewpoint model. This makes the configuration of cameras very flexible. On the other hand, since multiple cameras no longer preserve this model, using features from multiple cameras in SLAM is not a trivial issue: The question of how features from additional cameras can be used in the iterative optimizations of a SLAM system needs to be carefully analyzed. Based on this analysis, we are able to integrate those image features into a single visual SLAM system. This enables our MAV to achieve more robust pose tracking and to build a map that consists of more interesting regions of the environment. In the final implementation, we expand the FOV of the vision system by using two cameras mounted looking in two different directions (forward and downward) to capture more critical views, as shown in Fig. 1. The choice of the count of cameras is a compromise between tracking robustness and onboard computational capability. We further modify the above multi-camera visual SLAM system to operate as a robust visual odometry with constant-time cost towards large-scale explorations. Moreover, we propose an efficient back-end for loop-closure detection and correcting pose drift that is inherent in the visual odometry by using pose-graph optimization (PGO).

The multi-camera visual SLAM front-end and the back-end were first proposed in our previous conference presentations in [10] and [13], respectively. In this paper, we complete the theory of our multi-camera visual SLAM system with more detailed and systematical analysis, and provide further evaluations of the system in the experimental results.

The remainder of this paper is organized as follows. Related work on visual SLAM for MAVs and multi-camera visual SLAM is reviewed in Section 2. In Section 3, we present the analysis on optimizations in multi-camera SLAM, and the implementation of the visual odometry. Then we present our SLAM back-end for managing the global map and loop closing in Section 4. The performance of the proposed SLAM system is evaluated in the experiments in Section 5. Finally, in the last section, we provide the summary and discussions of this work.

## 2. Related work

In recent years, real-time visual SLAM has been achieved. Two methodologies have become predominant in visual SLAM [14]: filtering methods which fuse information from all past measurements in a probability distribution [15,16], and keyframe-based methods, like parallel tracking and mapping (PTAM) [17] and Linear MonoSLAM [18]. PTAM organizes its map in keyframes and use nonlinear optimizations for pose tracking and map refinement. Linear MonoSLAM adopts an efficient sub-map scheme and utilizes bundle adjustment (BA) only in building initial

sub-maps, resulting a linear approach which can achieve a performance very close to that obtained by using global BA. The advantages and disadvantages of filtering and keyframe-based methods are analyzed in the work in [19,20], suggesting that, in most modern applications, keyframe optimization gives the most accuracy per unit of computing time. The progress in visual SLAM has facilitated its applications to aerial robots.

Autonomous navigation of UAVs/MAVs relying on pose estimates from GPS sensors has been well studied in early researches. Related work usually fuses inertial navigation system (INS) data to aid GPS-sensor data, achieving autonomous navigation in high altitude and long range operations. However, they are not suitable in GPS-denied environments, like indoor or in outdoor urban area. Recently, much effort has been focused on developing visual SLAM systems to enable autonomous flight of MAVs. Related work using stereo cameras, monocular cameras and RGB-D cameras can be found in the literature.

Autonomous mapping and exploration for MAVs based on stereo cameras is presented in [21]. The work in [22] features a vision system for autonomous navigation of MAVs using two pairs of stereo cameras, with stereo triangulation adding constraints to bundle adjustment in PTAM. A stereo setup yields metric scale information of the environment. However, those systems have difficulties in using distant features since they triangulate those feature points based on their short baselines. Stereo visual odometry and SLAM systems may degenerate to the monocular case when the distance to the scene is much larger than the stereo baseline. In this case, stereo vision becomes ineffective and monocular methods must be used [14].

In [23], PTAM is used to provide position estimates for an MAV, while fusing data from an air pressure sensor and accelerometers to estimate the unknown metric scale factor of the monocular vision system. The work in [24] presents a visual–inertial data fusion method based on the EKF. It is further implemented in [25] for autonomous navigation of MAVs using inertial data and visual pose estimates from a modified PTAM system. The scale drift of the monocular PTAM system has been considered in the EKF framework.

A vision-based system combining advantages of both monocular vision and stereo vision is developed in [5], which uses a low frame-rate secondary camera to extend a high frame-rate forward facing camera that is equipped with a fisheye lens. It can provide robust state estimates for a quadrotor by fusing the onboard inertial data. The resulting vision system mainly relies on monocular vision algorithms, while being able to track metric scale by stereo triangulation. Since the two cameras are configured in a stereo setup, the field of view of the vision system is not expanded. The improved work in [26] enables a quadrotor to autonomously travel at speed up to 4 m/s, and allows roll and pitch angles exceeding 20° in 3D indoor environments.

In [27], autonomous flight of an MAV is enabled by the proposed SLAM system using an RGB-D camera. A visual odometry is used for real-time local state estimation, and integrated with the RGBD-Mapping (described in [28]) to form the SLAM system. An efficient RGB-D SLAM system is described in [29], which enables an MAV to autonomously fly in an unknown environment and create a map of its surroundings. In this work, sparse optical flow is used for feature matching, which is of advantage when motion blur may result in very limited number of local features being detected. The work in [30] perform RGB-D visual odometry on an MAV to enable its autonomous flight. 3D occupancy grid map is then built for path planning of the MAV.

Previous work on developing multi-camera systems can mainly be found in applications of surveillance and object tracking [31,32]. More relevant related work appears in the context of structure from motion (SFM). The work in [33] presents a theoretical treatment of multi-camera systems in SFM deriving the generalized epipolar constraint. The work in [34] proposes a virtual camera as a representation of a multi-camera system for pose estimation. A structure-from-motion scheme is achieved using multiple cameras in this work.

A number of multi-camera systems for pose estimation of mobile robots can be found in the literature. In [35], pose estimation of a mobile robot is solved by placing two back-to-back stereo pairs on the robot using the Extended Kalman Filter (EKF). The work in [9] adopts a generalized camera model described in [33] for a multi-camera system, to estimate the ego-motion of a self-driving car using a 2-Point RANSAC algorithm. This system allows point correspondences among different cameras. In [36], pose-graph loop-closure constraints are computed. The relative pose between two loop-closing pose-graph vertices is obtained from the epipolar geometry of the multi-camera system. The work in [37] presents a visual SLAM system with a multi-camera rig using Harris corner detector [38]. In its further work in [39], a Bayesian approach to data association is presented taking into account moving features which can be observed by cameras under robot motion. The work in [40] provides solutions to two different problems in multi-camera visual SLAM: automatic self-calibration of a stereo rig while performing SLAM and cooperative monocular SLAM.

In [41], PTAM with multiple cameras mounted on a buoyant spherical airship is reported in a manual flight experiment. It employs a ground-facing stereo camera pair which can provide metric scale, together with another camera mounted pointing to the opposite direction using a wide-angle lens. In [42], multi-camera visual SLAM is achieved based on a modified version of PTAM. This SLAM system allows convergence in pose tracking and mapping with the absence of accurate metric scale, taking advantages of the Taylor omnidirectional camera model and a spherical coordinate update method.

In order to use multiple cameras for pose estimation, the extrinsic parameters of those cameras need to be calibrated. Here, we are interested in the previous work solving this calibration problem for multiple cameras with non-overlapping fields of view. The work in [43] proposes a SLAM-based automatic calibration scheme for multiple cameras. The scheme uses global bundle adjustment to optimize the alignment of maps built by different visual SLAM instances, each processing images from one corresponding camera. The proposed solution computes the relative 3D poses among cameras up to scale. A more recent work can be found in [44], which uses a computationally expensive SLAM system to accurately map an infrastructure before the extrinsic calibrations of multi-camera systems. During a calibration process, 2D–3D correspondences among visual features in the current scene and in the previously known map are used for tracking the camera poses based on a Perspective-n-point (PnP) method. Then, camera extrinsics are optimized via non-linear refinement. Self-calibration of multiple stereo cameras and IMU is further achieved in [11], which also achieved multi-camera visual SLAM based on the generalized camera model. At lease one set of stereo cameras is required in this work. Real-time loop closing runs on-board of MAVs is achieved in both [13] and [11].

## 3. Multi-camera visual SLAM front-end

### 3.1. Overview of the visual SLAM front-end

The implementation of the visual SLAM front-end is based on the PTAM framework. After we propose the multi-camera visual SLAM scheme, the SLAM system is further reduced to be a constant-time visual odometry. The resulted visual odometry mainly consists of two separate threads, in which images from multiple cameras are utilized: In the first thread responsible for camera pose tracking, fixed-range searches are applied to FAST corners [45]

correspond to potentially observable map points in images from multiple cameras. The resulted 2D–3D correspondences are then used for camera-pose tracking through an iterative optimization process. The second thread integrates new keyframes from multiple cameras to the map and creates new map points. Furthermore, local bundle adjustment is performed to refine the map.

The map of the SLAM front-end consists of a set of keyframes **K** obtained by all the cameras in the vision system at different time-stamps and a set of 3D map points **P** measured by these cameras. A map point $p_j$ is located and added to the map by stereo triangulation of matched FAST corners in successive keyframes from a certain camera. Each keyframe saves its absolute pose and all the observations to the map points. In the SLAM front-end, only a local map containing recent keyframes and their measured map points are maintained, as will be explained in more detail in Section 3.4. A global map of the SLAM system is only implicitly maintained in the back-end, which will be presented in Section 4.1.

### 3.2. Multi-camera projection and pose update

Using the same calibrated camera projection model as in [17], the image projection of a map point $p_j$ to a camera $C_i$ (in the multi-camera system) is

$$\mathbf{u}_{ji} = \mathcal{P}_{C_i}\left(E_{c_i w}\mathbf{p}_j\right), \tag{1}$$

where $\mathcal{P}_{C_i}$ is the mapping of a point in the camera coordinate system $\mathcal{C}_i$ to the image coordinates in camera $C_i$, and $\mathbf{p}_j$ are the world coordinates of the map point $p_j$. $E_{c_i w}$ is chosen as a member of the Lie group **SE**(3), which is a 6D manifold. It represents the $i$th camera pose in the world coordinate system, containing a rotation and a translation component.

Since our goal is to use the SLAM system to track the pose of the MAV, without losing generality, we compute the pose update of one specific camera, which we call the first camera, $C_1$, based on the measurements from all cameras. The pose of other cameras can be updated by assuming a constant transformation relative to $C_1$. Thus, with a calibrated transformation between the first camera and the MAV body coordinate system, the MAV pose can be updated, as illustrated in Fig. 2. Following this idea, pose updates of all cameras can be expressed with one single six-element vector $\mu$:

$$E'_{c_i w} = E_{i1} \cdot E'_{c_1 w} = E_{i1} \cdot e^{\mu} \cdot E_{c_1 w}, \tag{2}$$

where $\mu$ is an element of the Lie algebra $\mathfrak{se}(3)$, and $E_{i1}$ is the pose of $C_1$ in the $i$th camera coordinate system. In this paper, $e^{\mu}$ is defined following the exponential map of an element $\mu$ in the Lie algebra $\mathfrak{se}(3)$ to the manifold **SE**(3):

$$\exp : \mathfrak{se}(3) \mapsto \mathbf{SE}(3). \tag{3}$$

For $C_1$ itself, $E_{i1}$ is simply the identity matrix. The pose tracking (and a part of mapping) problem of the SLAM system now mainly consists of how to obtain an optimized $\mu$ as a pose update for camera $C_1$ by minimizing certain objective function. One of the advantages of the parametrization of camera pose updates using a six-element vector $\mu$ in the Lie algebra $\mathfrak{se}(3)$ is that it allows a closed-form differentiation of Eq. (2), and thus, optimization problems containing camera poses in our SLAM system can be elegantly solved on the manifold **SE**(3).

### 3.3. Optimizations in the multi-camera visual SLAM front-end

Both the camera pose update and map refinement (using bundle adjustment) in the SLAM system are based on iteratively minimizing a robust objective function of the reprojection errors of sets of image measurements $S_i$, which are observed map points in each camera (or keyframe) $i$. It needs to be analyzed that how to correctly use features from different cameras in such optimization processes in order to preserve their mathematical soundness and numerical accuracy. In an $n$-camera (or $n$-keyframe) system, we need to minimize the function:

$$\sum_{i=1}^{n}\sum_{j\in S_i} \mathrm{Obj}\left(\frac{|\mathbf{e}_{ji}|}{\sigma_{ji}}, \sigma_T\right), \tag{4}$$

where Obj is the Tukey biweight objective function [46], $|\mathbf{e}_{ji}|$ is the reprojection error of point $j$ measured in camera (or keyframe) $i$, $\sigma_{ji}$ is the estimated measurement noise of point $j$, and $\sigma_T$ is a median-based robust standard-deviation estimate of the distribution of all reprojection errors [47]. $\mathbf{e}_{ji}$ is defined as the difference between the image reprojection of map point $j$ and its actual image measurement:

$$\mathbf{e}_{ji} = \mathbf{u}_{ji} - \hat{\mathbf{u}}_{ji}. \tag{5}$$

The minimization problem can be solved by iterations of reweighted nonlinear least squares [48]. One fundamental requirement to do this efficiently is to differentiate $\mathbf{e}_{ji}$ (i.e. to obtain the Jacobians of $\mathbf{e}_{ji}$) with respect to those parameters that need to be estimated. In pose tracking, the derivatives of $\mathbf{e}_{ji}$ with respect to the estimated camera pose update need to be computed. In bundle adjustment, the derivatives of $\mathbf{e}_{ji}$ with respect to the map point $j$ position are also required. The work in [49] provides a tutorial to the mathematical background of related derivatives. We analyze the derivatives of $\mathbf{e}_{ji}$ in the multi-camera case in the following two sections.

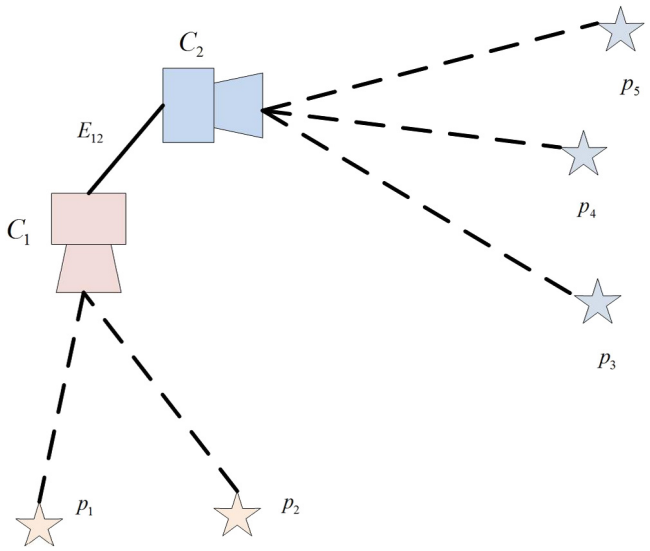


**Fig. 2.** A 2D illustration of MAV and camera pose updates in a dual-camera case. Although points $p_{j\in\{1,2,\ldots,5\}}$ are measured in either camera $C_1$ or $C_2$, these measurements are used to optimize the pose of camera $C_1$ only. The camera $C_2$ pose and the MAV pose are updated by assuming rigid connections to camera $C_1$.

#### 3.3.1. Pose update with multiple cameras

For the pose update of an $n$-camera system, the optimization problem is to find the optimal pose update $\mu$ for camera $C_1$:

$$\mu_1 = \underset{\mu}{\mathrm{argmin}} \sum_{i=1}^{n}\sum_{j\in S_i} \mathrm{Obj}\left(\frac{|\mathbf{e}_{ji}|}{\sigma_{ji}}, \sigma_T\right). \tag{6}$$

Following the above discussions, we analyze the Jacobians required for solving this optimization problem. For a map point $j$ measured by the first camera $C_1$, we can compute the Jacobian matrix of $\mathbf{e}_{ji}$

with respect to the estimated $C_1$ pose update $\mu$ using the chain rule as

$$\mathbf{J}_{1\mu} = \frac{\partial \mathcal{P}_{C_1}\left(e^{\mu} E_{c_1 w}\mathbf{p}_j\right)}{\partial \mu} = \left.\frac{\partial \mathcal{P}_{C_1}(\mathbf{c})}{\partial \mathbf{c}}\right|_{\mathbf{c}=E_{c_1 w}\mathbf{p}_j} \cdot \frac{\partial\left(e^{\mu} E_{c_1 w}\mathbf{p}_j\right)}{\partial \mu}. \quad (7)$$

The first term of the above matrix product is trivially the Jacobian of the camera projection function in Eq. (1), and the last term is:

$$\frac{\partial\left(e^{\mu} E_{c_1 w}\mathbf{p}_j\right)}{\partial \mu} = \begin{pmatrix}\mathbf{I}_3 & -[E_{c_1 w}\mathbf{p}_j]_\times\end{pmatrix}. \quad (8)$$

However, for map points measured by other cameras, with Eq. (2), the differentiation becomes:

$$\mathbf{J}_{i\mu} = \frac{\partial \mathcal{P}_{C_i}\left(E_{i1} e^{\mu} E_{c_1 w}\mathbf{p}_j\right)}{\partial \mu} = \left.\frac{\partial \mathcal{P}_{C_i}(\mathbf{c})}{\partial \mathbf{c}}\right|_{\mathbf{c}=E_{c_i w}\mathbf{p}_j} \cdot \frac{\partial\left(E_{i1} e^{\mu} E_{c_1 w}\mathbf{p}_j\right)}{\partial \mu}. \quad (9)$$

Its difference to Eq. (7) lies in the last term of this equation:

$$\frac{\partial\left(E_{i1} e^{\mu} E_{c_1 w}\mathbf{p}_j\right)}{\partial \mu} = Rot\left(E_{i1}\right) \cdot \begin{pmatrix}\mathbf{I}_3 & -[E_{c_1 w}\mathbf{p}_j]_\times\end{pmatrix}, \quad (10)$$

where $Rot\left(E_{i1}\right)$ represents the rotation component of $E_{i1}$. The Jacobians are all taken at the vicinity of the present estimates of the camera poses or point positions. Detailed deviations of the Jacobians can be found in [49].

#### 3.3.2. Bundle adjustment with multiple cameras

Bundle adjustment in the SLAM system means solving the following minimization problem:

$$\{\{\mu_2 \ldots \mu_N\}, \{p_1 \ldots p_M\}\} = \underset{\{\{\mu\},\{p\}\}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} \mathrm{Obj}\left(\frac{|\mathbf{e}_{ji}|}{\sigma_{ji}}, \sigma_T\right), \quad (11)$$

where $N$ is the number of keyframes and $M$ is the number of observed map points that need to be updated. The first keyframe is normally assumed to be fixed and to function as the reference frame.

In a multi-camera system, we assume that the relative poses among the group of new keyframes obtained at the same time $t$ by different synchronized cameras are constant, since the cameras are mounted rigidly. Thus, in bundle adjustment, we can use image measurements from all cameras to compute the optimal pose updates of the keyframe set $\mathbf{K}_1$ which are obtained by the first camera $C_1$. The poses of other rigidly connected keyframes are computed based on the updated poses of their associated keyframes in $\mathbf{K}_1$. This multi-camera bundle adjustment strategy is illustrated in Fig. 3. Therefore, a consistent map can be built by using multiple cameras. Here, the consistency is in the sense of rigid connectivity among sub sets of the whole map generated by multiple cameras.

In the above case, to solve bundle adjustment, we differentiate $\mathbf{e}_{ji}$ with respect to the corresponding keyframe (in $\mathbf{K}_1$) pose update $\mu_i$, which can be computed in the same way as in Eq. (7) or Eq. (9), depending on the camera identity of the point $j$, i.e. by which camera this point has been measured. The Jacobian of $\mathbf{e}_{ji}$ with respect to the estimated point $j$ pose can be expressed in a consistent way, regardless of the camera used to measure this point:

$$\mathbf{J}_{\mathbf{p}_j} = \frac{\partial \mathcal{P}_{C_i}\left(E_{c_i w}\mathbf{p}_j\right)}{\partial \mathbf{p}_j} = \left.\frac{\partial \mathcal{P}_{C_i}(\mathbf{c})}{\partial \mathbf{c}}\right|_{\mathbf{c}=E_{c_i w}\mathbf{p}_j} \cdot \frac{\partial\left(E_{c_i w}\mathbf{p}_j\right)}{\partial \mathbf{p}_j}. \quad (12)$$

The last term simply becomes:

$$\frac{\partial\left(E_{c_i w}\mathbf{p}_j\right)}{\partial \mathbf{p}_j} = Rot\left(E_{c_i w}\right). \quad (13)$$

### 3.4. From visual SLAM to constant-time visual odometry

In order to achieve constant-time cost in large-scale explorations, we further reduce the complexity of the multi-camera visual SLAM system by fixing the size of keyframes from each camera to be a constant number $n_L$ in the local map. We remove the oldest keyframe when a new keyframe is added to the map. Bundle adjustment is performed within all $m \cdot n_L$ keyframes in a $m$-camera case. This changes our multi-camera SLAM system to be an efficient constant-time visual odometry. An example map built by the visual odometry in our dual-camera setting is shown in Fig. 4.

If the number of keyframes from camera $C_i$ in the local map exceeds $n_L$ after a new keyframe is added, we remove the oldest keyframe $\mathcal{K}_r$ from the local map. Before that, we first update data associations of the map points measured in $\mathcal{K}_r$. Each keyframe $\mathcal{K}_i$ is associated with a set of map points $p_j$. We call $\mathcal{K}_i$ the source keyframe of $p_j$, since after it obtains a measurement to $p_j$, it is then used to triangulate $p_j$ together with one of its neighboring keyframe which also measures $p_j$. When $\mathcal{K}_i$ should be removed, we only remove those map points which are not measured by any other keyframe in the local map. Other map points could be important for later pose tracking. If $p_j$ is measured by another keyframe $\mathcal{K}_m$, we transfer its source keyframe identity to $\mathcal{K}_m$, and update its related data association with the measurement in $\mathcal{K}_m$, which will be used in pose tracking. Meanwhile, we mark $p_j$ indicating that it has been sent to the global map already, in order to avoid resending it with $\mathcal{K}_m$ in the future.

### 3.5. Visual odometry with dual cameras

As explained in Section 1, two cameras are utilized in the implementation of the SLAM front-end, being allowed to share no overlap in their respective fields of view. This flexible configuration can achieve a maximal FOV of the vision system, facing to those most interesting directions.

#### 3.5.1. Organizing the local map

In pose tracking and bundle adjustment with multiple cameras, as proposed in the previous sections, observations of a map point by different cameras are allowed in our method. Thus, if multiple cameras may share common perspectives to the environment, e.g. they are mounted looking in the forward direction or the sides of an MAV, potentially visible map points of each camera could be searched in the whole map generated by the vision system.

However, since we do not expect our MAV to do aggressive maneuvers with extreme roll or pitch rotation, the two cameras mounted on our MAV as shown in Fig. 1 will hardly obtain images with similar perspectives to the environment. Thus, we can assume that the two cameras share no common feature point from the environment. Then, the whole map of the SLAM system can be organized as two sub-maps, each of which corresponds to one camera. We divide the map by giving labels to each keyframe and each map point, indicating from which camera they have been generated.

The advantage of the above organization method is that it can make map operations in both pose tracking and mapping more efficient. When searching for potentially visible points in an image taken by camera $C_1$, now only the map points measured by $C_1$ need to be checked. In the mapping process, neighboring keyframes can be searched in a sub-map only, instead of the whole map. This is done whenever we need to decide whether a new keyframe should be added, or to choose an existing keyframe for triangulating new map points.

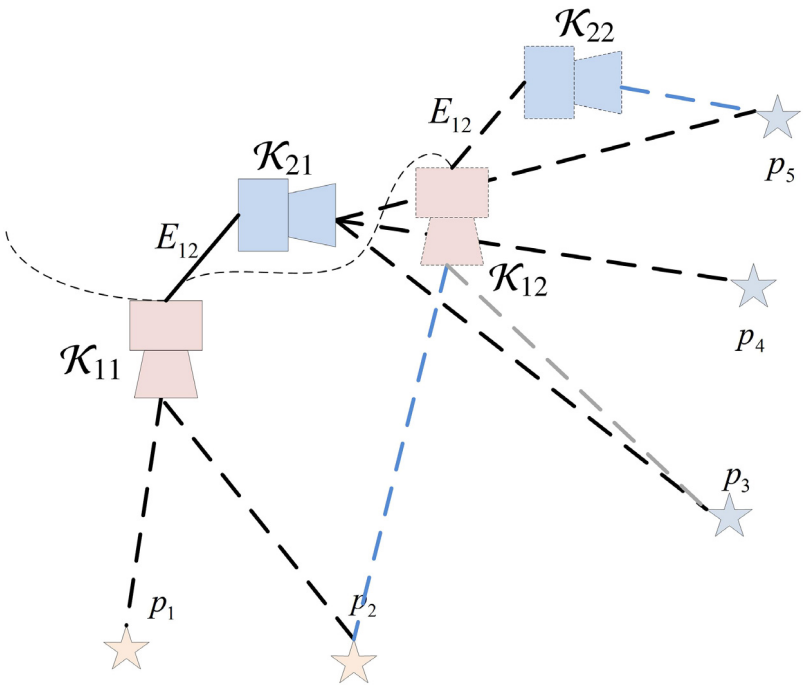


**Fig. 3.** A 2D illustration of bundle adjustment in a dual-camera case with two keyframes from each camera ($\mathcal{K}_{11,12}$ from camera $C_1$ and $\mathcal{K}_{21,22}$ from camera $C_2$). In this bundle adjustment problem, the measurements to points $p_{i\in\{1,2,\ldots,5\}}$ are used to optimize the poses of keyframes $\mathcal{K}_{11,12}$ and the positions of all these points. The poses of keyframe $\mathcal{K}_{21}$ and $\mathcal{K}_{22}$ are updated by assuming their rigid connections to $\mathcal{K}_{11}$ and $\mathcal{K}_{12}$, respectively.

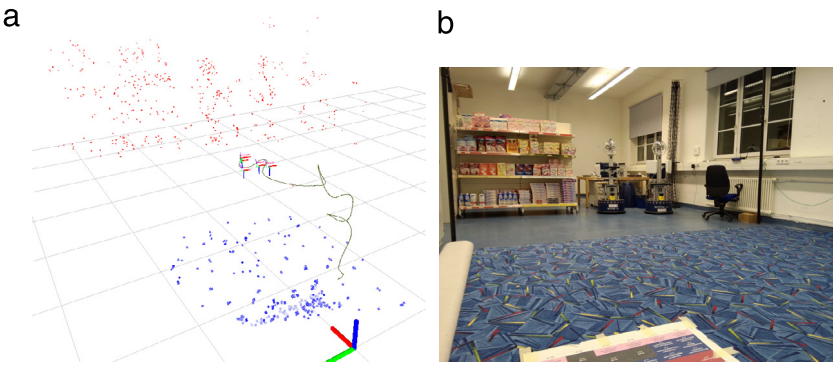


**Fig. 4.** (a) A local map built by our dual-camera visual odometry during exploration, with $n_L = 4$. Map points from the forward-looking camera are marked in red color, and those from the downward-looking camera in blue. The trajectory of the forward-looking camera is plotted in green. (b) A scene of the actual lab environment where this experiment was performed, in a similar perspective. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### 3.5.2. Pose tracking

Map points in the two sub-maps are reprojected to their corresponding source camera to decide whether they are potentially visible. Successful matches between image features and those potentially visible points serve as image measurements which are used in iterative optimizations for the pose update of the camera $C_1$. Then the optimal pose of the camera $C_2$ is computed by using Eq. (2).

In the original PTAM, if map point $j$ is measured in image pyramid level $s$ ($s \in \{0, 1, 2, 3\}$), the measurement noise is estimated to be $\sigma_j = 2^s$. In our dual-camera pose tracking optimizations, we assume that the measurement noises of the two cameras follow the same distribution, i.e. for any camera $i \in \{1, 2\}$ measuring point $j$, we have $\sigma_{ji} = 2^s$. This also applies to the optimizations in bundle adjustment. Measurement noise of each camera in a multi-camera system using significant different cameras or lenses should not be considered to follow the same distribution. Instead, the distributions can be estimated according to the actual performance of each sensor, as proposed in [50].

### 3.5.3. Mapping

New keyframes $\mathcal{K}_{1n}$ and $\mathcal{K}_{2n}$ from both dual cameras are added to the map of the SLAM system, when the geometric distance of $\mathcal{K}_{1n}$ (or $\mathcal{K}_{2n}$) to its closest neighbor obtained by the same camera is larger than a threshold. This means, if we obtain a keyframe from any of the dual cameras that should be added to the map, the keyframe obtained by the other camera at the same time stamp will also be added. Thus, in the global map, each keyframe obtained by the camera $C_1$ is always associated with another keyframe obtained by $C_2$, and vice versa. Here, the geometric distance measures the sum of weighted translation distance and angular difference, as has been done in the original PTAM.

To achieve real-time performance of the visual odometry during its exploration, we only retain local bundle adjustment in the mapping thread. The local bundle adjustment process involves a subset of map points and keyframes in the local map which are generated by both cameras. It computes the pose updates for keyframes and the 3D position updates for map points, which are added to the keyframe set $\mathbf{K}_a$ and the point set $\mathbf{p}_a$, respectively. As explained

in Section 3.3.2, only the keyframes from the camera $C_1$ will be added to $\mathbf{K}_a$. Keyframes which are obtained by the camera $C_2$ and associated (rigidly connected) to $\mathbf{K}_a$ form the set $\mathbf{K}_c$, whose poses can be computed by using the optimized poses of $\mathbf{K}_a$. $\mathbf{p}_a$ consists of all the points which are measured in $\mathbf{K}_a$ or $\mathbf{K}_c$. A further fixed keyframe set $\mathbf{K}_f$ contains any keyframe in which a measurement of any point in $\mathbf{p}_a$ has been made. Then the minimization of the local bundle adjustment becomes

$$\{\{\mu_{i\in\mathbf{K}_a}\}, \{p_{j\in\mathbf{p}_a}\}\} = \underset{\{\{\mu\},\{p\}\}}{\operatorname{argmin}} \sum_{i\in\mathbf{K}_a\cup\mathbf{K}_c\cup\mathbf{K}_f} \sum_{j\in\mathbf{p}_a\cap S_i} \operatorname{Obj}\left(\frac{|\mathbf{e}_{ji}|}{\sigma_{ji}}, \sigma_T\right), \quad (14)$$

which is solved by using Levenberg–Marquardt method [51]. The Jacobians of $\mathbf{e}_{ij}$ are solved as described in Section 3.3.2. We use a similar strategy as in PTAM to define the keyframe set $\mathbf{K}_a$: It consists of $n_k$ keyframes obtained by the camera $C_1$, including the newest keyframe and the other $n_k - 1$ ones nearest to it. Normally, we set $n_k = 5$.

#### 3.5.4. Automatic initialization

Metric scale ambiguity generally exists in monocular camera systems. Our dual-camera system has the same issue when the cameras have no overlap in their respective fields of view, since no stereo triangulation can be used to recover the metric scale factor of the map built by the system. We solve this issue by initializing the metric map of our SLAM system similarly as we did in [52], using a standard helicopter landing pad. We use the initialization module presented in [53] to robustly estimate the pose of the downward-looking camera $C_1$ during the takeoff phase of our MAV.

When the MAV height is larger than a given threshold, the first camera pose and the associated image are sent to the SLAM system. 3D positions of the feature points in the image are obtained by assuming they lie on the ground plane, which does not need to be strictly true, as demonstrated in the outdoor experiment in [52]. The sub-map corresponding to the first camera $C_1$ is initialized with those feature points. The sub-map corresponding to the second camera $C_2$ is initialized after two keyframes from this camera are obtained, when 3D feature points can be triangulated with the known keyframe poses.

## 4. Back-end of the SLAM system

Our multi-camera visual odometry only maintains a local map for pose tracking. We further implement a back-end for the SLAM system to manage and refine a global map during exploration. While integrating the local map, the back-end mainly performs loop-closure detection and pose-graph optimization to refine the global map.

### 4.1. The global map representation

We use a keyframe-based global map representation as proposed in [54]. The representation is similar to the one used in the original PTAM: Each keyframe consists of a four-level image pyramid with FAST corners computed in each level, while each map point storing a reference to its source keyframe in which it is measured. However, the work in [54] adopts a relative representation for the position of each map point and uses local feature descriptors for feature matching: Instead of storing an absolute position of each map point in the world frame as in PTAM, we store the position of each map point relative to its source keyframe. In this way, we can achieve implicit updates of the absolute position of a map point after the pose of its source keyframe being updated in pose-graph optimization as will be described in Section 4.3.

We compute BRIEF descriptors [55] for feature points in all pyramid levels of each keyframe. They are used for wide-baseline matching between keyframes and appearance-based loop-closure detection in the back-end. Compared with some other local feature descriptors, e.g. SIFT [56], SURF [57], the BRIEF descriptor is not scale invariant nor rotation invariant. However, it is more efficient and is already sufficient for our MAV-application scenario. First, since we compute it in four pyramid levels of a keyframe, scale invariance can be achieved in a certain extent without considering further pyramid levels. Second, as will be discussed in Section 4.4.2, only keyframes and map points from the forward-looking camera will be used to build the global map. Then, as our MAV flies in a nearly hovering attitude, we do not expect the image measurements of a map point to be significant rotated among different keyframes. Thus, rotation invariant of feature descriptors is not necessary for feature matching. In general, BRIEF descriptors offer us a good compromise between distinctiveness and computation time. On the other hand, ORB features can be a reasonable alternative to BRIEF if rotation invariance is required.

### 4.2. Loop-closure detection

Loop-closure detection provides additional pose constraints (edges) to the pose graph of the SLAM system as will be described in Section 4.3. It is of vital importance for correcting pose drift and building a consistent map during loopy explorations in large-scale environments. We consider the following two types of loop closures.

#### 4.2.1. Appearance-based loop closure

We use an appearance-based (image-to-image) method for online large-scale loop-closure detection among keyframes in the global map which are obtained from the local map, as explained in Section 4.4.2. More specifically, local features (BRIEF features) are used to represent the image appearance information of a keyframe. The open-source implementation of the bag-of-words method developed in [58] is adopted for this task. In the off-line process, we train the vocabulary tree required by this method using BRIEF descriptors of FAST keypoints extracted from a large number of images taken from various environments.

After a large loop is detected between the current keyframe $\mathcal{K}_A$ and a previous keyframe $\mathcal{K}_B$, we match feature corners in $\mathcal{K}_A$ to map points measured in $\mathcal{K}_B$ to retrieve 2D–3D correspondences. If there are enough 2D–3D correspondences, we estimate the relative pose $E_{AB}$ between these two keyframes. This can be done efficiently using a P3P [59] plus RANSAC method, and further refining the result by robust optimization which minimizes 2D reprojection errors of all inlier correspondences. Then the edge $\mathcal{E}_{AB}$ is added to the pose graph. We do not compute the otherwise relative pose $E_{BA}$, since we expect that the keyframe $\mathcal{K}_A$ would have significant pose drift relative to $\mathcal{K}_B$, and thus positions of map points measured in $\mathcal{K}_B$ should be trusted.

#### 4.2.2. Local loop closure

A potential local loop closure can be detected by trying to register the current keyframe $\mathcal{K}_A$ to its best neighbor $\mathcal{K}_B$ within a certain geometric distance range, and estimating their relative pose in a way similar to the method we used after an appearance-based loop closure is detected. Since the registration process is time cost intensive, the geometric distance range should be rather limited in order to keep real-time performance of the back-end. Here, the best neighboring keyframe is decided by its co-visibility of common features with $\mathcal{K}_A$.

In this case, we compute both $E_{AB}$ (from 2D–3D correspondences matching feature corners in $\mathcal{K}_A$ to map points measured in $\mathcal{K}_B$) and $E_{BA}$ (from 2D–3D correspondences matching feature corners in $\mathcal{K}_B$ to map points measured in $\mathcal{K}_A$). We expect $E_{AB}$ and $E_{BA}$ to agree with each other if a true local loop closure has been detected, as proposed in [54].

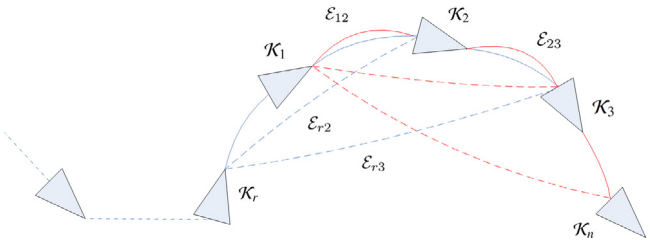


**Fig. 5.** Edges added to the pose graph after a new keyframe $\mathcal{K}_n$ is added, in the case of four keyframes involved in bundle adjustment. Edge $\mathcal{E}_{r2}$ and $\mathcal{E}_{r3}$ will also be added to the graph before $\mathcal{K}_r$ is removed from the local map. Edge $\mathcal{E}_{12}$ and $\mathcal{E}_{23}$ will be replaced by new constraints (in red) after a new bundle adjustment operation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### 4.3. Adaptive-window pose-graph optimization

We apply pose-graph optimization (PGO) at each time when a new keyframe is added to the global map. During this process, we adaptively define a window (sub-graph) of the whole global graph to be adjusted, depending on whether new edges are added from loop closures.

#### 4.3.1. The graph structure

For the purpose of PGO, the graph structure definition of our SLAM system is straightforward: It consists of a set of keyframe-pose vertices ($\mathcal{V}_i$) and relative edges ($\mathcal{E}_{ij}$) describing the relative pose–pose constraints ($E_{ij}$) among those vertices. Each vertex $\mathcal{V}_i$ stores an absolute pose $E_i$ of its corresponding keyframe $\mathcal{K}_i$ in the world frame of the SLAM system. The edges consist of constraints obtained from the bundle adjustment in the visual odometry and the two types of loop closures described in Section 4.2.

We notice that each bundle adjustment operation in the visual odometry produces pose constraints among all $n_L$ keyframes in the local map. When a new keyframe $\mathcal{K}_n$ is added to the local map of the visual odometry, the oldest keyframe $\mathcal{K}_r$ will be removed before the next bundle adjustment operation. As a result, poses of the remaining $n_L - 1$ keyframes will be re-adjusted during the next bundle adjustment step without considering the pose constraints to $\mathcal{K}_r$. This means that pose constraints between $\mathcal{K}_r$ and the other $n_L - 1$ nodes may actually contain useful information and should be considered in PGO. Thus, we not only add pose constraints among consecutive keyframes to the pose graph, but also add those between $\mathcal{K}_r$ and all other keyframes within the current local map, as depicted in Fig. 5.

#### 4.3.2. Defining the sub-graph for optimization

In PGO, we always consider a window of the whole pose graph $\mathcal{G}$ to be adjusted. We perform uniform-cost search for choosing sub-graph vertices, beginning with the latest added vertex. The cost is measured by the geometric distances among vertices. The sub-graph to be adjusted (denoted as $\mathcal{G}_s$) consists of all those sub-graph vertices and constrains among them. We define the size of the sub-graph vertices in the following adaptive way depending on whether there is a loop closure being detected: During regular exploration of the SLAM system without loop closure being detected, only a relatively small window (with no more than $n_s$ vertices) of the whole pose graph is to be adjusted. In this case, PGO is still useful due to the constraints produced by the bundle adjustment as we discussed in the last section. When a loop closure between two keyframes is detected, an edge between their corresponding vertices will be added to the pose graph. Then, we expand the graph window $\mathcal{G}_s$ to contain a large number of vertices until it has included all vertices in the detected loop or a maximal number ($n_m$) of vertices.

#### 4.3.3. Pose-graph optimization

Given a $n$-vertices sub-graph $\mathcal{G}_s$ we previously defined, the pose-graph optimization is to minimize the following cost function:

$$F(\mathbf{E}) = \sum_{\mathcal{E}_{\mathcal{G}_s}} \Delta E_{ij}^T \Omega_{ij} \Delta E_{ij}, \tag{15}$$

with respect to all vertex poses $\mathbf{E} = (E_1, E_2, \ldots, E_n)$, where $\mathcal{E}_{\mathcal{G}_s}$ contains all edges in $\mathcal{G}_s$, $\Delta E_{ij} := \log(E_{ij} \cdot E_j^{-1} \cdot E_i)$ is the relative pose error in the tangent space of SE(3) and $\Omega_{ij}$ is the information matrix of the pose constraint $E_{ij}$. We estimate $\Omega_{ij}$ coarsely as a diagonal matrix in a way proposed in [4]:

$$\Omega_{ij} = \omega_{ij} \begin{bmatrix} \lambda_t^2 I_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & \lambda_r^2 I_{3\times 3} \end{bmatrix}, \tag{16}$$

with the rotation component $\lambda_r$ being a constant and the translation component $\lambda_t$ being inversely proportional to the parallax of $E_{ij}$ which is measured by normalizing the translation $\mathbf{t}_{ij}$ using the average scene depth. Here, $\mathbf{t}_{ij}$ is the translation element of $E_{ij}$. We consider the pose constraints produced in the bundle adjustment and loop closures are similar in accuracy. Thus, we set $\omega_{ij}$ to be a constant $\omega_{ij} = 1$.

### 4.4. Implementation details

Our visual SLAM system mainly consists of three threads: two threads for the visual odometry (the tracking thread and the mapping thread), and a third thread working for the back-end (back-end thread). The mapping thread of the visual odometry manages a local map and handles most of the interactions with the back-end. The global map and the pose graph is managed by the back-end thread. The flowcharts of the mapping thread and the back-end thread are shown in Fig. 6. In this section, we present further implementation details about interactions between the visual odometry and the back-end.

#### 4.4.1. Overview of the back-end

The back-end thread runs in an infinite loop as illustrated in Fig. 6(b). It is activated by the mapping thread whenever a new keyframe is added to the waiting queue of the global map. After a pose-graph optimization operation is done, the global map is updated according to the pose graph: The keyframe poses are updated by directly copying the corresponding vertex poses of the graph, leaving the associated map points only implicitly updated. The implementation of our PGO is based on the open source library $g^2o$ described in [60].

#### 4.4.2. Adding local map to the back-end

In order to construct the global map, we add the local sub-map (including keyframes and map points) built by one specific camera to the back-end. BRIEF descriptors of feature points in the keyframes are computed in the back-end. We set the specific camera to be the forward-looking camera $C_2$, for three reasons. First, since the dual cameras are rigidly connected, pose constraints of one camera are sufficient for pose-graph optimization. Second, we mainly use our MAV in low-altitude applications. Thus, we expect more interesting views taken from the forward-looking camera, which will be used for loop-closure detection. Third, this is again a compromise for real-time performance: Keyframes from the downward-looking camera might later be included in the global map when the onboard computation capability is significantly improved.

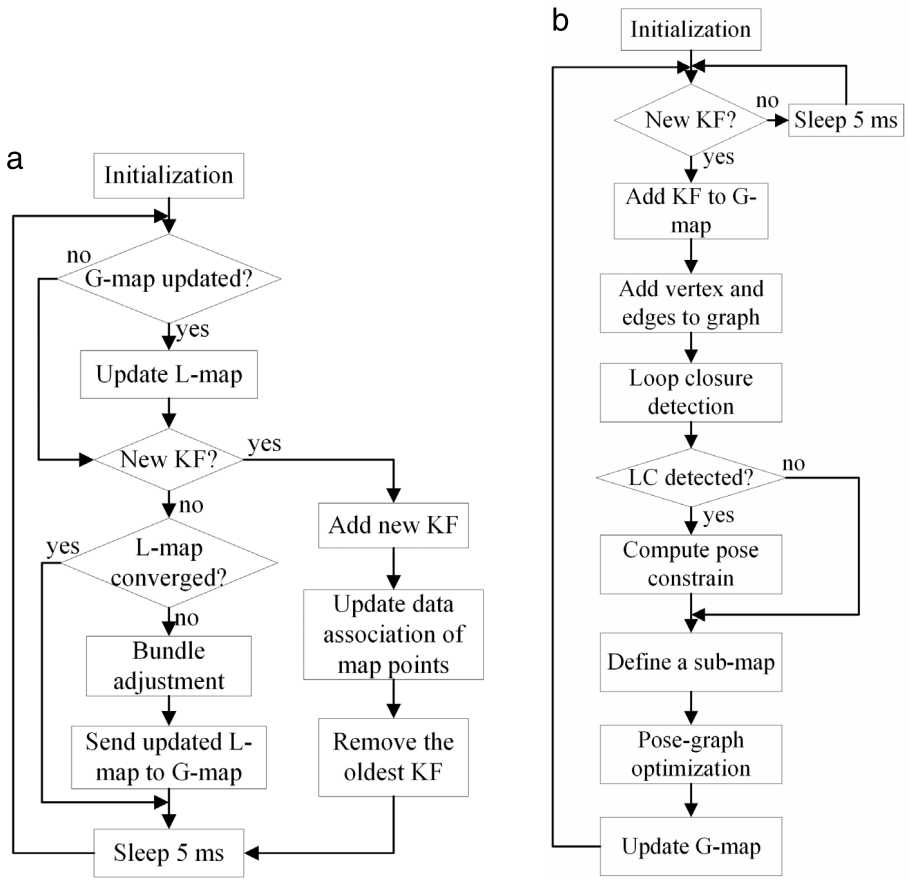


**Fig. 6.** (a) The mapping thread of the visual odometry, and (b) the back-end thread of the SLAM system. Abbreviations: KF (keyframe), L-map (local map), G-map (global map), and LC (loop closure).

*4.4.3. Updating the local map*

After each pose-graph optimization process, we update the local map in the visual odometry according to the global map, including updating keyframe poses and the positions of their measured map points.

Here, we distinguish keyframe $\mathcal{K}_{1j}, j \in \{1, 2, \ldots, n_L\}$ from the downward-looking camera $C_1$ and keyframe $\mathcal{K}_{2j}, j \in \{1, 2, \ldots, n_L\}$ from the forward-looking camera $C_2$. We assume that the pose of $\mathcal{K}_{2j}$ is identical to the pose of its corresponding keyframe in the global map, while the $\mathcal{K}_{1j}$ pose is updated by assuming a calibrated rigid transform to $\mathcal{K}_{2j}$. Since the map points in the local map are stored with absolute coordinates, their positions need to be updated individually. We do this by assuming fixed relative translations to their current source keyframe.

## 5. Experiments

In the experiments, the pose-tracking robustness of our multi-camera system is validated first. Then, the performance of the multi-camera visual SLAM system is evaluated. Experiments with both autonomous flights and manual flights of an MAV are carried out. Furthermore, the accuracy of the SLAM system is evaluated by comparing its pose tracking results with the ground truth data provided by an external tracking system.

### 5.1. Experimental setup

We implemented our software in several modules (nodes) using the open source Robot Operating System (ROS) [61] on Ubuntu Linux 12.04. ROS provides the infrastructure for logging all interested onboard data and for efficient communication among our different vision modules: two ROS nodes for the dual-camera drivers, one node for the initialization module, and another one for the SLAM system. Images and other onboard sensor data are recorded into ROS-bag files if necessary.

*5.1.1. Quadrotor platform*

Two quadrotors are used in our experiments, one for the indoor flights and the other one for outdoor flights, respectively. In the indoor experiments, our MAV is based on the open source and open hardware quadrotor platform developed by the Pixhawk project from ETH Zürich, described in [62], as shown in Fig. 1. The onboard computer is a Kontron microETXexpress computer-on-module (COM) featuring an Intel Core 2 Duo 1.86 GHz CPU, 2 GB DDR3 RAM and a 32Gb SSD. The pxIMU inertial measurement unit and autopilot board mainly consists of a microcontroller unit (MCU), and sensors including a tri-axis accelerometer and a tri-axis gyroscope. The MCU is a 60 MHz ARM7 microcontroller for sensor readout and fusion, as well as position and attitude control. In the outdoor experiments, an Asctec Pelican quadrotor is used, which has a payload of 650 grams. Its onboard computer is Asctec Mastermind, featuring an Intel Core i7 CPU. On both of the quadrotors, we utilize two cameras for our SLAM system: Two PointGrey Firefly MV monochrome cameras, each with an image resolution of 640 × 480 pixels, on the Pixhawk quadrotor, while two Blufox2 cameras, each with an image resolution of 752 × 480 pixels, on the Asctec Pelican. All camera lenses we use have viewing angles of approximately 90°.

*5.1.2. External tracking system*

To measure ground truth data of the 6 DOF quadrotor pose in the indoor experiments, we use an external Optitrack tracking

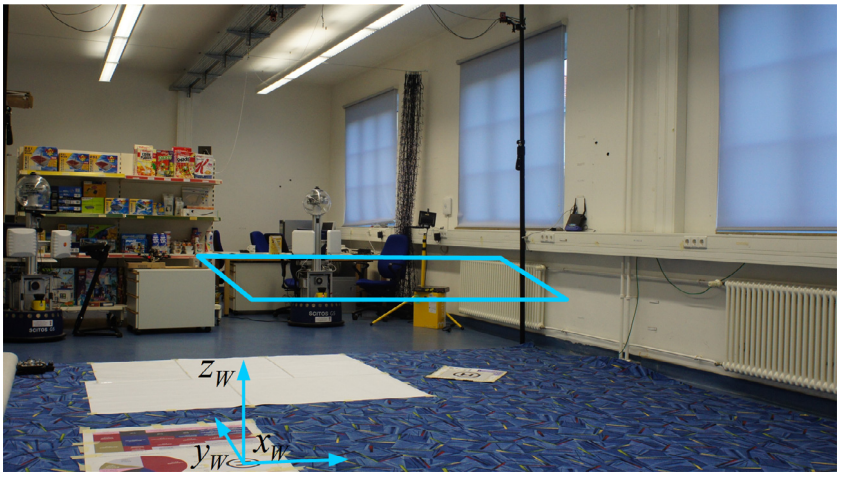


**Fig. 7.** A scene of our robot lab where the indoor experiments were carried out. The world coordinate system and the desired flight path of the MAV are indicated inside.

system manufactured by Naturalpoint,[2] which comprises 9 infrared cameras in our case. After attaching several highly reflective markers to the quadrotor, the tracking system can provide 6 DOF pose estimates of the quadrotor with a frequency of up to 100 fps. According to our tests, the deviation of the position estimates for a static quadrotor is in the order of few millimeters.

#### 5.1.3. Camera synchronization

On the hardware level, we use a master–slave scheme to synchronize different cameras. Camera $C_1$ performs as a master, with an image frequency of $f_m = 30$ fps. Image acquisitions of camera $C_2$ are triggered by the signal sent by $C_1$ at each time it starts to acquire an image. On the software level, we enforce a rule that the time-stamp difference of the two newest images from the two cameras should be smaller than $1/(2 \cdot f_m)$ s. An image obtained by the slave camera $C_2$ which breaks this rule will be dropped to avoid unexpected errors during image transportation between camera drivers and the SLAM system. If this happens, the new image from camera $C_1$ will be used for pose tracking only, and not for the further mapping process.

#### 5.1.4. Extrinsic calibration of cameras

We calibrate the extrinsic parameters between the first camera and other cameras ($E_{i1}$) by utilizing our external tracking system and a pattern which is also used for camera intrinsic parameters calibration. We attach a few markers on both our quadrotor and the pattern. Thus, their corresponding poses in the tracking system can be obtained. The camera poses with respect to the pattern can be obtained by performing extrinsic calibration of each camera based on one image of the pattern. Then $E_{i1}$ can be obtained after a few coordinate transformations with a similar pre-calibrated first-camera to quadrotor pose. When an external system is unavailable, an alternative way is to use a SLAM-based method, like the one presented in [44].

#### 5.1.5. Quadrotor controllers

In the indoor experiments, autonomous flight of the Pixhawk quadrotor is demonstrated. The pose controller and trajectory controller of the quadrotor are similar to those proposed in [63]: A nested PID controller that consists of a separate attitude and a position controller is used for pose control. The 3D position estimates from our visual SLAM system are fed to the position controller at frame rate, which applies a basic Kalman Filter to smooth the pose estimation. The attitude controller runs at a frequency of 200 Hz, using the roll and pitch estimates from the IMU, and only the yaw angle is provided by the visual SLAM system, since the yaw angle estimates from the IMU is normally inaccurate. A trajectory controller is also implemented to enable the MAV to navigate along a predefined path. The basic idea of this controller is to only consider position errors in the normal direction of the path, while ignoring the errors in the tangent direction, which makes a constant expected forward speed possible. The final commanded acceleration is calculated by a PD controller. We simplify it by dropping the feed-forward term of the desired acceleration, and set a constant forward speed, $v_s$, along the tangent direction of the searching path.

### 5.2. Validation of the robustness of the multi-camera scheme

In this section, we employ the dual-camera visual SLAM system (before it is reduced to a visual odometry) presented in Section 3 without the back-end in an indoor environment. We demonstrate that the multi-camera setting proposed in this paper provides more robust pose tracking than using a monocular camera.

#### 5.2.1. Enabling autonomous navigation

In this first experiment, we demonstrate the efficiency of our SLAM system to enable autonomous navigation of an MAV.

The environment in which the indoor experiments are carried out is shown in Fig. 7, in which we also sketch the world coordinate system in visual SLAM and depict the desired path of the MAV. There is a large white area on the desired path, in which no visual feature can be obtained by the downward-looking camera. The MAV autonomously navigates along a predefined rectangular path (plotted in cyan in Fig. 8(b)) in a counter-clockwise direction with a commanded forward speed of $v_s = 0.4$ m/s, taking off and finally landing above the origin of the world coordinate system. The takeoff phase is controlled by using pose estimates fed from the initialization module until the SLAM system is initialized. We set the MAV to turn 90° after it passes one edge of the rectangular path and reaches the first corner. This turn will make the forward-looking camera unable to triangulate new features and track its pose if it is the only camera in the SLAM system. We use the trajectory-following method (described in Section 5.1.5) for the trajectory control of our MAV in the autonomous flight.

The resulting MAV trajectory during an autonomous flight can be found in Fig. 8. The MAV trajectory estimated by our onboard SLAM system (SLAM2c) using two cameras fits well with the ground truth data from the external tracking system (ETS). The

[2] http://www.naturalpoint.com/optitrack/products/tracking-tools-bundles.

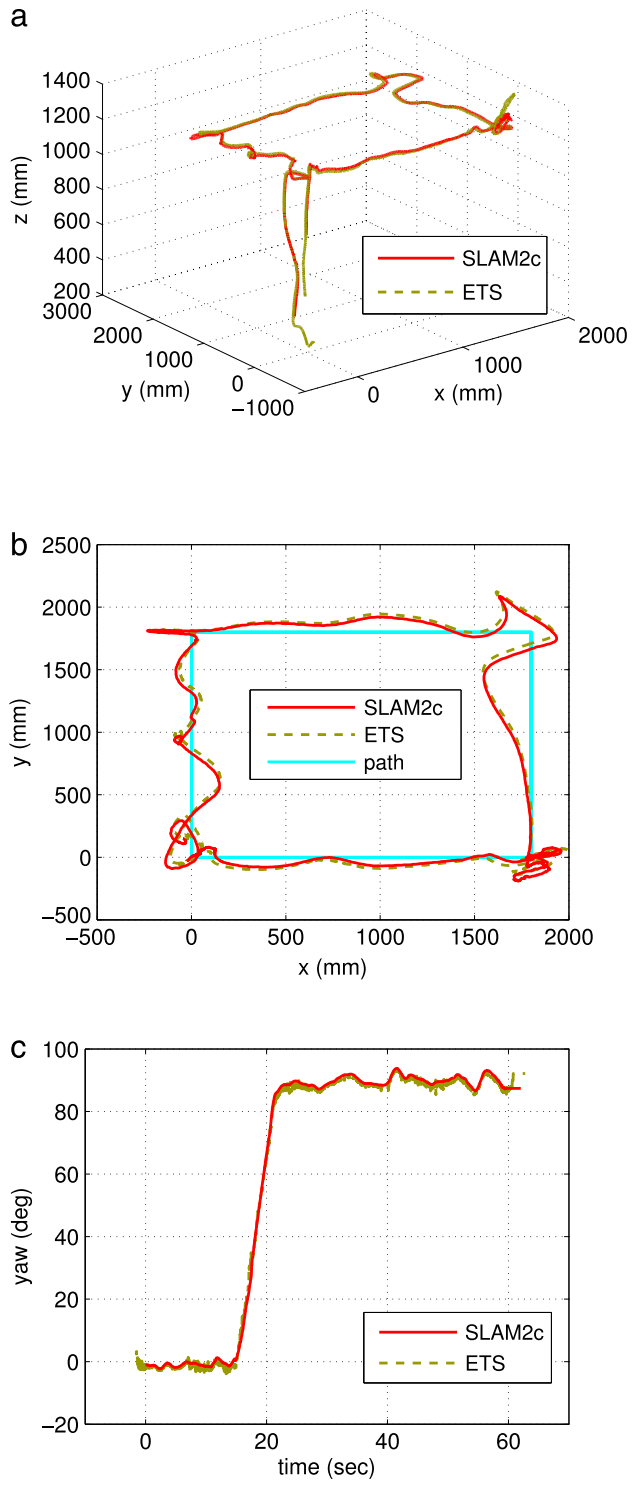


**Fig. 8.** The MAV poses estimated by our visual SLAM system (SLAM2c) and the external tracking system (ETS) during the autonomous navigation. (a) The trajectory on $x_W$, $y_W$, and $z_W$ axes, and (b) projected to the $x_W$–$y_W$ plane. (c) The yaw angle of the MAV.

RMSEs of the pose estimates of SLAM2c data with respect to the ETS data are listed in Table 1 (the row of Auto.). Three sources of noise which contribute to the errors should be noted. First, slow scale drift still exists in our SLAM system, since this system does not have additional depth sensor or stereo triangulation to provide metric scale measurements. Second, the extrinsic camera calibration errors can also affect the pose tracking and mapping accuracy. A last minor factor comes from the ground truth data itself, since it is difficult to set the tracking system coordinate frame to perfectly coincide with the world frame.

**Table 1**
MAV pose RMSEs of the whole trajectories in autonomous flight (Auto) and manual flight (Manual), with position errors in *millimeters* and attitude errors in *degrees*.

| RMSEs | *x* | *y* | *z* | 3D | Roll | Pitch | Yaw |
|---|---|---|---|---|---|---|---|
| Auto. | 23.5 | 37.2 | 15.9 | 46.8 | 0.82 | 0.81 | 1.04 |
| Manual | 23.4 | 43.2 | 12.5 | 50.7 | 1.31 | 1.08 | 1.06 |

**Table 2**
MAV pose RMSEs during the part of manual flight when the MAV pose can be tracked by the downward-looking camera alone, with position errors in *millimeters* and attitude errors in *degrees*.

| RMSEs | *x* | *y* | *z* | 3D | Roll | Pitch | Yaw |
|---|---|---|---|---|---|---|---|
| SLAM2c | 25.5 | 35.4 | 13.1 | 45.6 | 1.02 | 0.94 | 0.91 |
| SLAMdc | 38.2 | 29.7 | 19.1 | 52.0 | 1.30 | 1.27 | 1.37 |

In Fig. 8(b), we can find fluctuations in the resulting trajectory at the designated path corners. The reason is that we set the MAV to hover 5 s at each corner. If the desired pose of the MAV propagates forward when the MAV is still trying to hover back to a corner, the trajectory may form a fluctuation like the one at the top right corner of Fig. 8(b). Accurately identifying the dynamic parameters of the MAV system or implementing a more sophisticated pose controller like the one proposed in [64] could achieve better control performance of the autonomous flight.

### 5.2.2. Further evaluation through manual flight

We process an image logfile off-board to perform further evaluation of our SLAM system. The onboard computational capability does not allow us to perfectly take image logfiles during autonomous navigation. Thus, we manually control the MAV to follow a similar path as in Section 5.2.1, and take a logfile containing images from both cameras and other useful onboard sensor data by utilizing the original ROS functions for recording ROS-bag files.

The MAV trajectory during this manual flight is shown in Fig. 9. When using the proposed SLAM system with two cameras, the MAV pose can be tracked well throughout the flight, and also fits well with the ground truth data. The corresponding RMSEs are listed in Table 1 (the row of Manual). Two minor parts of the ground truth data are missing due to the flight outside the working area of the external tracking system, which is found to be two periods of straight dashed lines in Fig. 9(b) around the time of 17 and 29 s. Fig. 10 shows two views of the final map built by our SLAM system including the dual-camera trajectory.

However, if the SLAM system uses only the downward-looking camera, pose tracking will fail when the MAV flies above the white area (see the SLAMdc case in Fig. 9). The MAV position where pose tracking fails in this case is marked with black circles in Fig. 9(a). We mark the time when it fails with a vertical red line in Fig. 9(b), in which MAV positions on each axis are shown. During the part of flight before tracking failure happens, the RMSEs of pose tracking using the proposed SLAM system (SLAM2c) are a bit smaller than those of using only the downward-looking camera (SLAMdc), as can be found in Table 2. Similarly, if we use the SLAM system with only the forward-looking camera after the initialization of the system, pose tracking will fail again when the MAV turns 90° during hovering. Like in the first failure case, we mark the failure with black crosses in Fig. 9(a) and a vertical black line in Fig. 9(b).

## 5.3. Evaluation of the mult-camera visual SLAM system

In this section, we evaluate the full SLAM system proposed in this paper in two steps: In the indoor experiments, the full visual

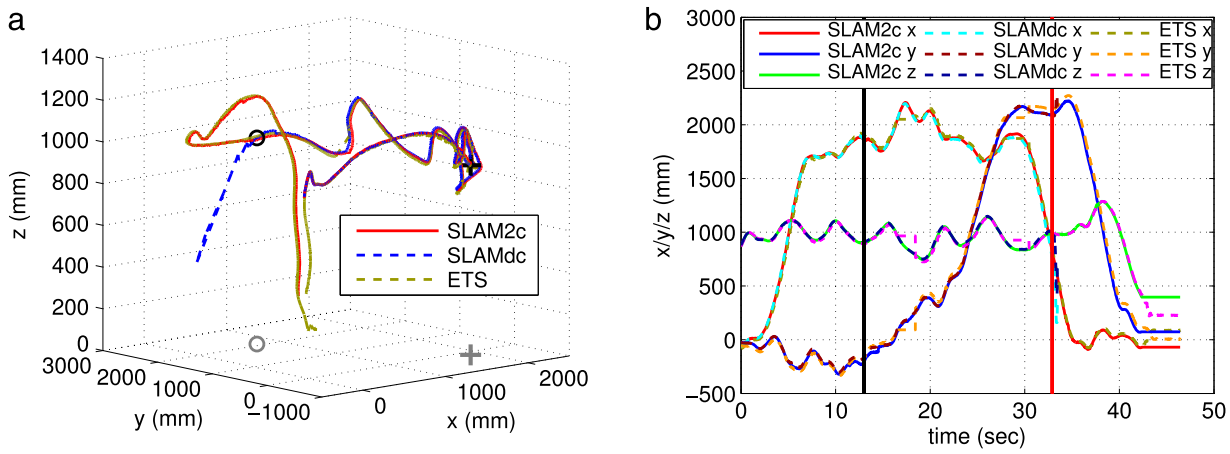


**Fig. 9.** The MAV poses estimated by the proposed SLAM system (SLAM2c), PTAM with only the downward-looking camera (SLAMdc), and the external tracking system (ETS) for the manual flight logfile. (a) The trajectory on $x_W$, $y_W$, and $z_W$ axes, (b) projected to the $x_W$–$y_W$ plane, and (c) with respect to the flight time.

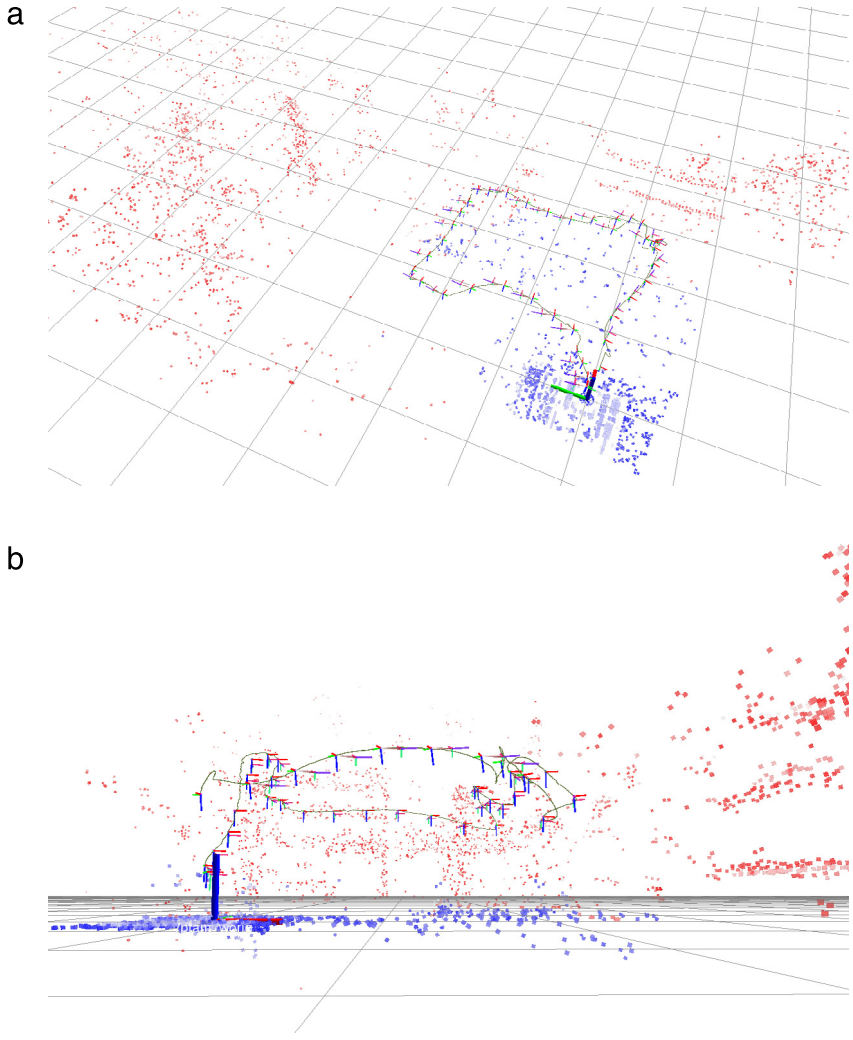


**Fig. 10.** Built map and the forward-looking camera trajectory during the manual flight, in two different perspectives. Map points measured by the downward-looking camera are marked in blue, those measured by the forward-looking camera in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

SLAM system without fusing inertial data is evaluated. Furthermore, we carry out outdoor experiment to show the performance of the full visual SLAM system aided by inertial data in large-scale environment.

*5.3.1. Enabling autonomous navigation indoor*

In this experiment, we demonstrate the efficiency of our full SLAM system to enable autonomous navigation of an MAV. The MAV autonomously navigates along a predefined rectangular path

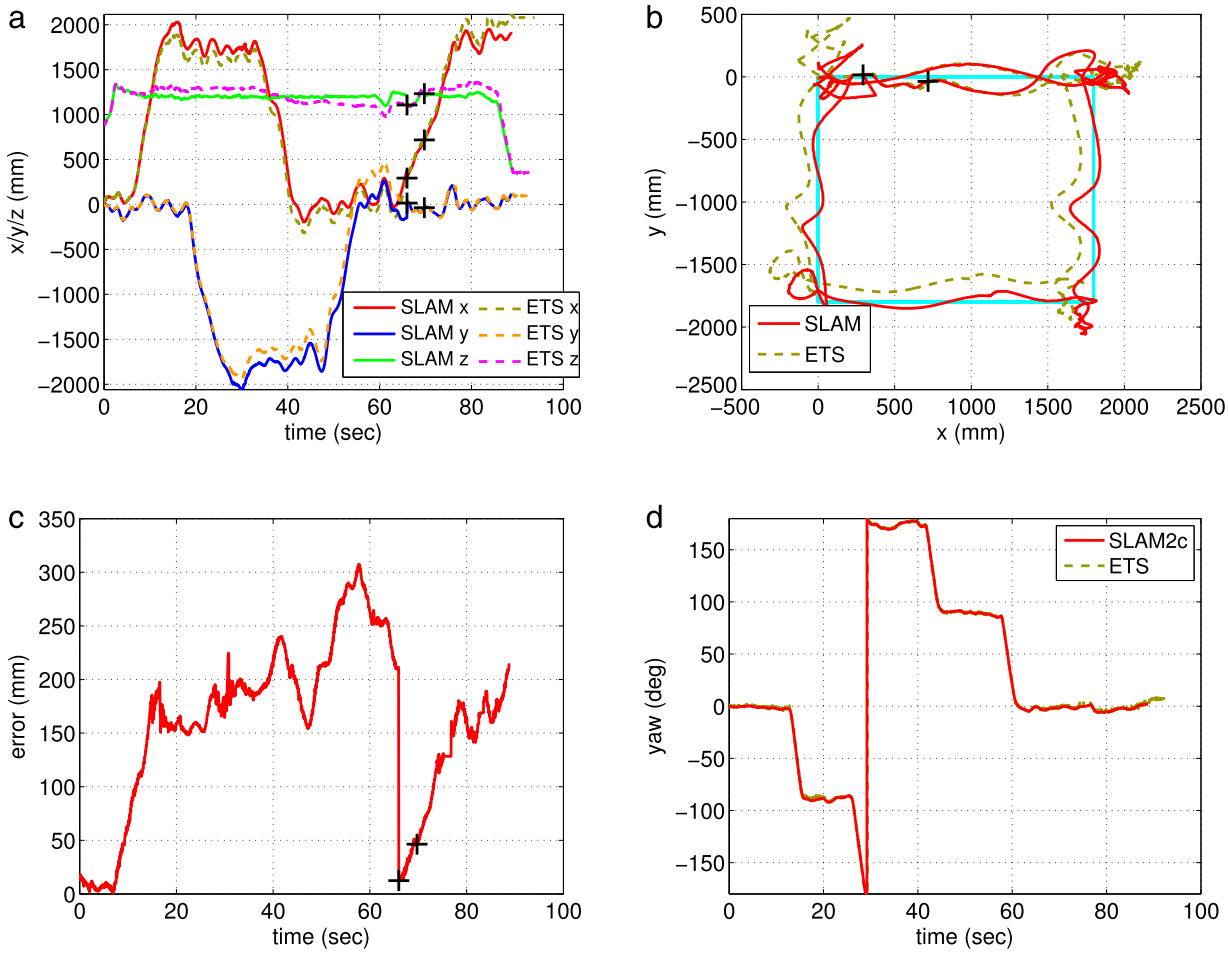


**Fig. 11.** MAV pose-estimation results during the autonomous flight, compared with ground truth data: (a) MAV position estimation on $x_W$, $y_W$, and $z_W$ axes, (b) on $x_W$–$y_W$ plane, (c) the translation error, and (d) the yaw-angle estimation. Pose corrections after loop closures being detected and PGO being processed, are marked with black crosses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 3**
MAV pose tracking RMSEs in the autonomous flight (Auto) and the manual flight (Manual) experiments using the proposed visual SLAM system, and using only the visual odometry (VO), with position errors in *millimeters* and attitude errors in *degrees*.

| RMSEs | $x$ | $y$ | $z$ | 3D | Roll | Pitch | Yaw |
|---|---|---|---|---|---|---|---|
| Auto. | 103.0 | 117.0 | 82.7 | 176.5 | 1.70 | 1.45 | 0.80 |
| Manual | 50.6 | 89.9 | 114.9 | 150.4 | 2.06 | 1.34 | 1.92 |
| VO | 69.2 | 168.2 | 106.7 | 206.9 | 2.37 | 1.35 | 2.28 |

with a height of 1.2 m (plotted in cyan in Fig. 11(b)) in a clockwise direction, taking off above the origin of the world frame and landing on the top-right corner. It turns 90° at each corner in order to head to the forward direction. To keep better real-time performance of the SLAM system onboard MAVs, we set $n_L = 4$ in this work.

Fig. 11 shows the pose tracking results of the SLAM system (SLAM) compared with the ground truth data provided by the external tracking system (ETS). The visual odometry slowly drifts during explorations. However, the local-map update process will correct the drift after a loop closure is detected and the pose-graph optimization is performed. We can clearly recognize the effect of such corrections in Fig. 11: After a loop closure is detected and the local map is updated correspondingly, we mark a black cross to the trajectory where the loop closure happens. The first loop-closure provides an obvious correction to the pose tracking of the visual odometry. Fig. 11(c) shows the 3D translation errors of the pose tracking results compared with the ground truth data during the flight. Those very short horizontal lines in Fig. 11(c) are resulted from missing ground truth data during those periods of time, when the MAV is not flying in the effective field of view of the external tracking system. The RMSEs of the 6DOF pose estimates of the SLAM system to the ground truth data during the whole flight are listed in Table 3 (in the row of Auto.).

### 5.3.2. Evaluations with manual flight data from indoor environment

We manually control the quadrotor to fly in a similar way as we have done in Section 5.3.1, and record all necessary onboard data in a ROS-bag file. We process this logfile in post-processing on the onboard computer, to gain more insights into the performances of both the visual odometry and the back-end of the SLAM system.

Fig. 12(a) shows the pose tracking results of the SLAM system on the $x_W$, $y_W$, and $z_W$ axes of the world frame. Fig. 12(b) provides a top view of the MAV trajectory on the $x_W$–$y_W$ plane. Here, we have processed the ROS-bag file twice using our SLAM system: firstly, using the full SLAM system (SLAM), and secondly, using only the visual odometry without the back-end (VO). The results of the two processes are compared with pose estimates from the external tracking system (ETS). Fig. 12(c) shows the position estimation errors of these two processes. Without the back-end, the visual odometry would result in larger pose drift. Furthermore, loop-closure detection of the back-end can obviously benefit the pose tracking with drift corrections. After a loop closure is detected and the local map is updated correspondingly, we also mark a black cross to the trajectory where the loop closure happens in

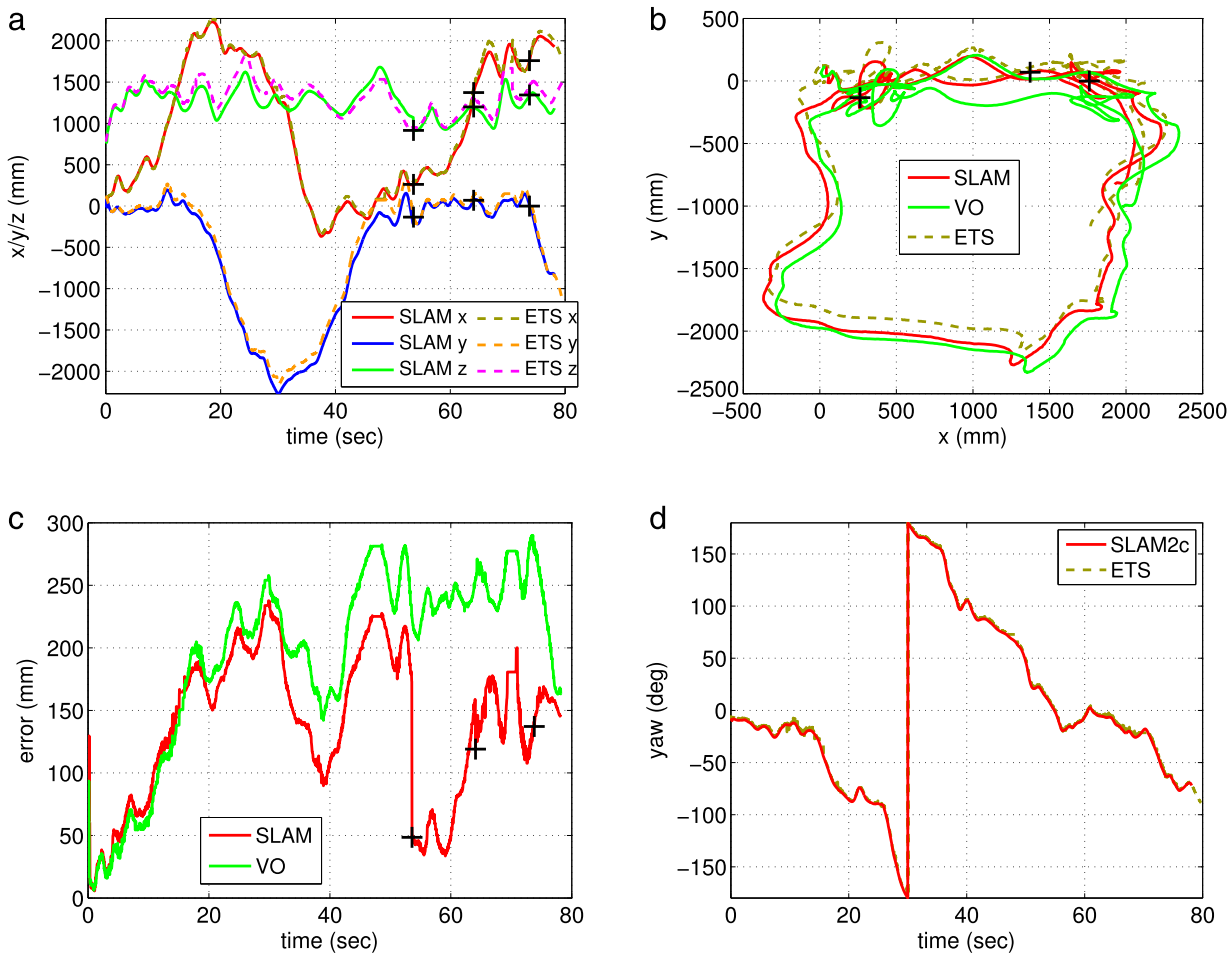


**Fig. 12.** MAV pose-estimation results using the proposed SLAM system and visual odometry during the manual flight compared with ground truth data: (a) MAV position on $x_W$, $y_W$, and $z_W$ axes, (b) on $x_W$–$y_W$ plane, (c) the translation errors, and (d) the yaw-angle estimates.

this experiment. RMSEs of the pose tracking results of the two processes are listed in Table 3 (in the row of Manual and the row of VO, respectively).

Fig. 13(a) and (c) provide more details in the performance of the visual odometry when running our full SLAM system. Time costs of the tracking thread (Tracking) and the mapping thread (LMapping) of the visual odometry are shown in Fig. 13(a), which are largely affected by the number of map points in the local map, as shown in Fig. 13(c). The pose tracking process is rather fast, with an average time cost of 0.0152 s. The bundle adjustment in the mapping thread is the most time intensive process, which only runs when new keyframes are added to the map, however. Time costs of both threads reach peaks when the MAV flies above the place where it takes off (with highly textured background). This is resulted from the way we initialize the SLAM system: We use all feature corners (after non-maximum suppression) of the image which are sent by the initialization module for initialization, resulting in a very large number of map points in the local map if the background is highly textured.

Fig. 13(b) shows the time cost of the back-end process when each new keyframe is added to the global map: the pose-graph optimization process (PGO), and the rest processes of the back-end (rest) which mainly consists of the loop-closure detection and the preparation processes. Since this experiment is performed in a small area, the average time cost of the PGO is only less than one *millisecond*. Both the currently visible map points and the feature corners in the new keyframe can affect the time cost of the local loop closure and the appearance-based loop-closure detection. The time cost of the loop-closure detection and related preparation contributes the most to that of the back-end. Nevertheless, the maximal time cost is less than 0.1 s. Fig. 13(d) shows the total number of map points in the global map (GPoints), as well as the number of feature corners in each new keyframe (corners). The number of map points in the global map increases linearly during the exploration. However, their positions will only be implicitly updated, and thus they do not affect the real-time performance of the SLAM system.

If we roughly compare the statistics in Tables 2 and 3, we can find that, after reducing to be a visual odometry, the dual-camera visual SLAM losses part of its accuracy before loop closures are found. On the other hand, the visual odometry has the advantage of being constant-time consuming, which fits better to large scale operations.

Fig. 14 illustrates two views of the resulting global map of the SLAM system, with references of the real-world pictures. In this figure, we can find the pose graph with nodes (keyframe poses) and edges (in green), and map points at their absolute positions which are only computed for visualization purpose.

### 5.3.3. Further evaluations in outdoor environment

Visual odometry without direct depth measurement suffers from scale drift, especially during large-scale operations. To reduce this effect, and more importantly, achieve more robust pose tracking in complex environments, we fuse inertial data from an XSENS Mti-30 IMU using an Extended Kalman Filter (EKF) presented in [65], in a loosely-coupled way. Inertial data from the

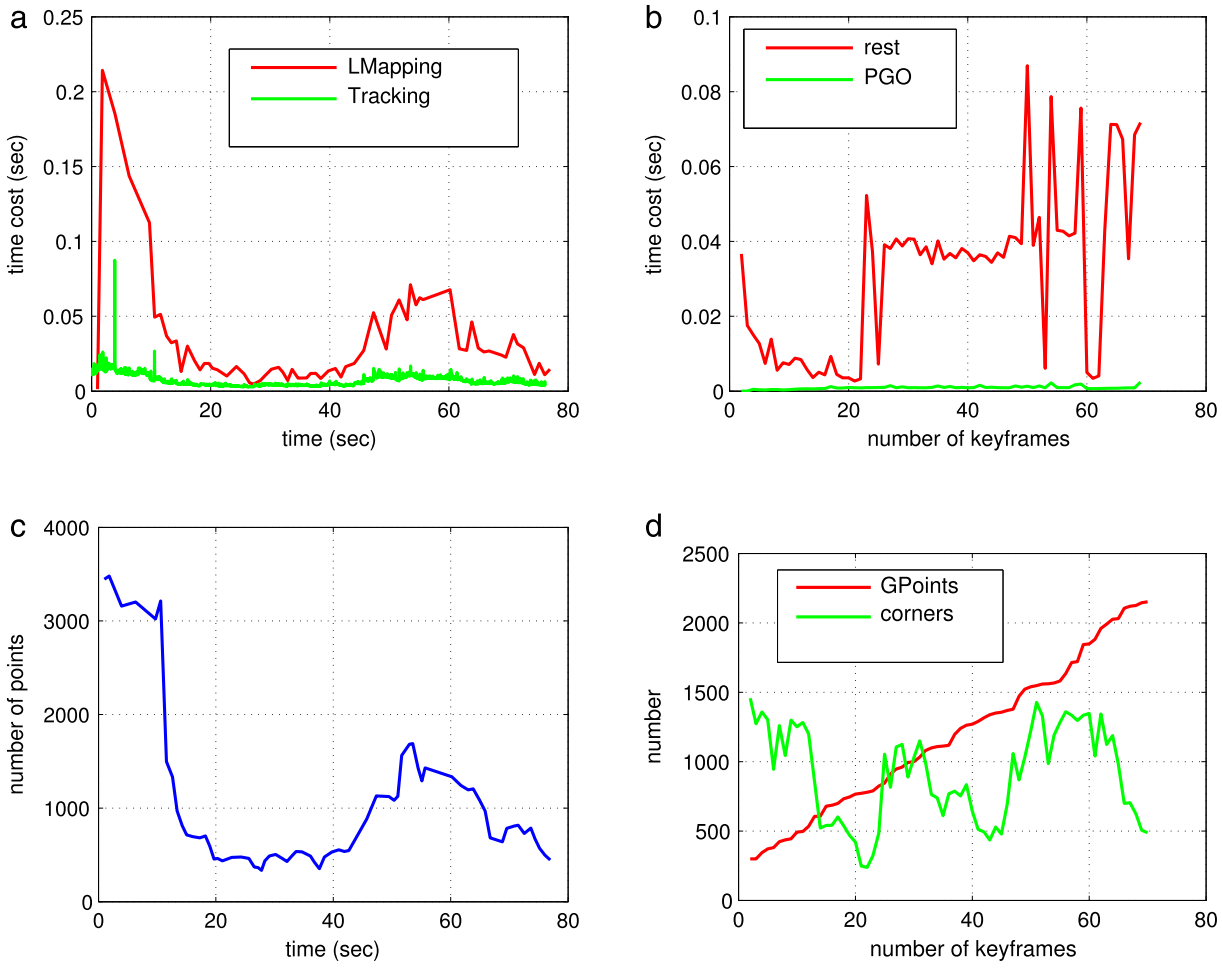


**Fig. 13.** (a) Time cost of the tracking thread (Tracking) and mapping thread (LMapping) of the visual odometry during the exploration, and (c) the number of map points in the local map. (b) Time costs of the PGO process (PGO) and the rest of the processes of the back-end (rest) when each new keyframe is added to the back-end. (d) Total number of map points in the global map (GPoints) and the number of feature corners of each new keyframe (corners).

IMU is acquired in 200 Hz. MAV pose updates are fed back to the SLAM system after each measurement update and pose prediction of the EKF. Pose tracker of the SLAM system uses the pose prediction propagated to the current time-stamp as an initial estimate for more accurate pose tracking. With this implementation, we perform an outdoor experiment with longer trajectory to evaluate the performance of our multi-camera visual SLAM system. In this experiment, the Pelican quadrotor is used to take a logfile of the two cameras and the IMU data during a manual flight. The logfile is processed offline on the onboard computer of Pelican.

Two images of the outdoor environment taken by the onboard cameras are shown in Fig. 15. Similarly to the experiments in Section 5.3.2, Fig. 16 shows the pose tracking results of the SLAM system when used as a full SLAM system (SLAM) and as a visual odometry without the back-end (VO). One loop closure, marked with black crosses, is detected and used to add a pose constraint to the pose graph, during the flight with two loops. More loop candidates are found by the appearance-based method and local loop detection method. However, if we fail to further obtain a correct pose constraint, such a loop will not be assumed to be a final loop closure. This may happen when no enough accurate map points are measured by the loop-candidate keyframes. Nevertheless, less loops are preferred in our system to make sure that the map is not corrupted by false-positive loops. Fig. 17 illustrates two views of the resulting map of the SLAM system and the visual odometry. Although ground truth data from an external tracking system is unavailable, we can still find that the SLAM back-end improves the consistency of the global map, judging from the blue map points which are measured by the downward camera. On the other hand, since we failed to obtain a loop closure during the second loop of the trajectory, the whole map built during the full trajectory is not consistent enough.

## 6. Conclusions and discussions

In this paper, we have presented a visual SLAM system which is able to utilize measurements from multiple cameras. The mathematical analysis on how those measurements should be integrated in the optimization processes of the SLAM system has been provided. We have demonstrated the efficiency of the system by enabling an MAV with two cameras to navigate autonomously along a predefined path. The experiments with a logfile taken from a manual flight prove that our proposed multi-camera system is more resistant to tracking failure than a conventional monocular vision system. It also brings more flexibility to configuring multiple cameras used onboard MAVs. We have shown the accuracy of our system by comparing the pose tracking results with the ground truth data provided by the external tracking system. Our final SLAM implementation utilizes dual cameras to achieve a constant-time visual odometry, which can provide robust pose tracking for autonomous navigation of our MAV. The back-end of the SLAM system performs loop-closure detection and adaptive-window pose-graph optimization to correct pose drift of the visual odometry and to maintain a consistent global map.

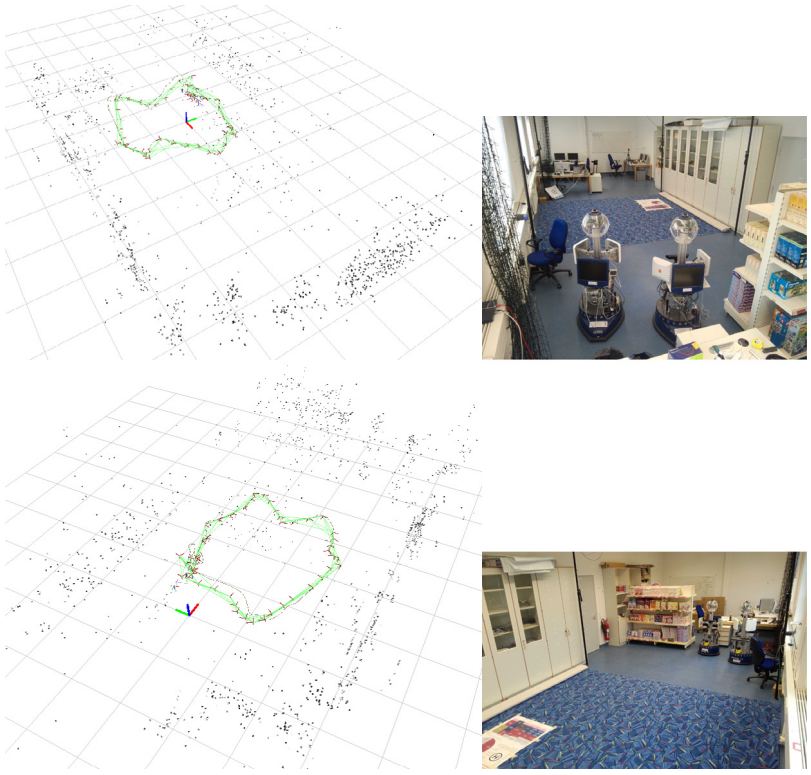

**Fig. 14.** Two views of the global map built by the visual SLAM system during a manual flight in the lab. The grid cell size is 1 m × 1 m. The global map points are shown in gray-scale. The vertices of the pose graph are illustrated with red tri-axes, and the edges are plotted in green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Fig. 15.** Sample images of the outdoor experiment from the forward-looking camera and the downward-looking camera.

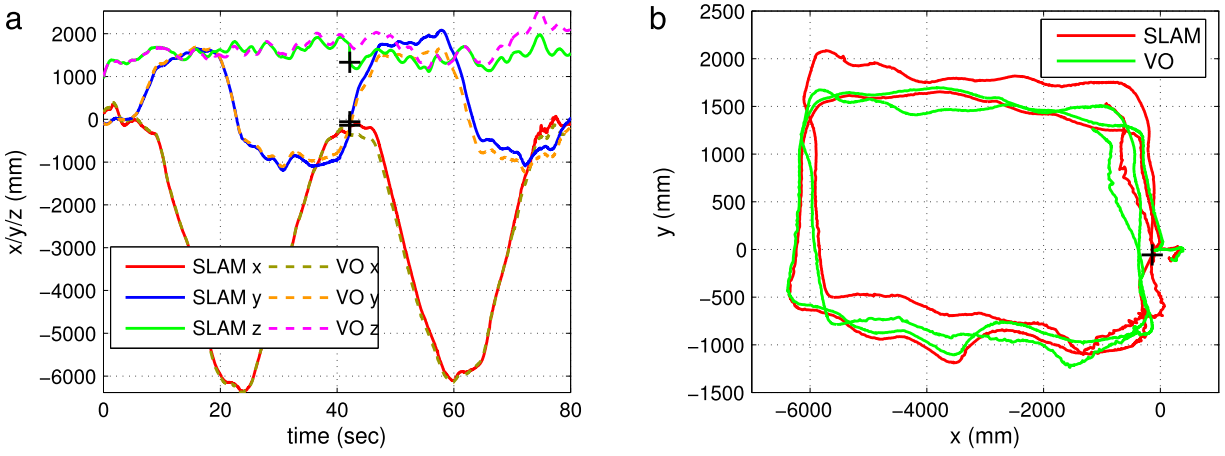


**Fig. 16.** MAV pose-estimation results using the proposed SLAM system and visual odometry during the outdoor experiment (a) MAV position on $x_W$, $y_W$, and $z_W$ axes, (b) on $x_W$–$y_W$ plane.

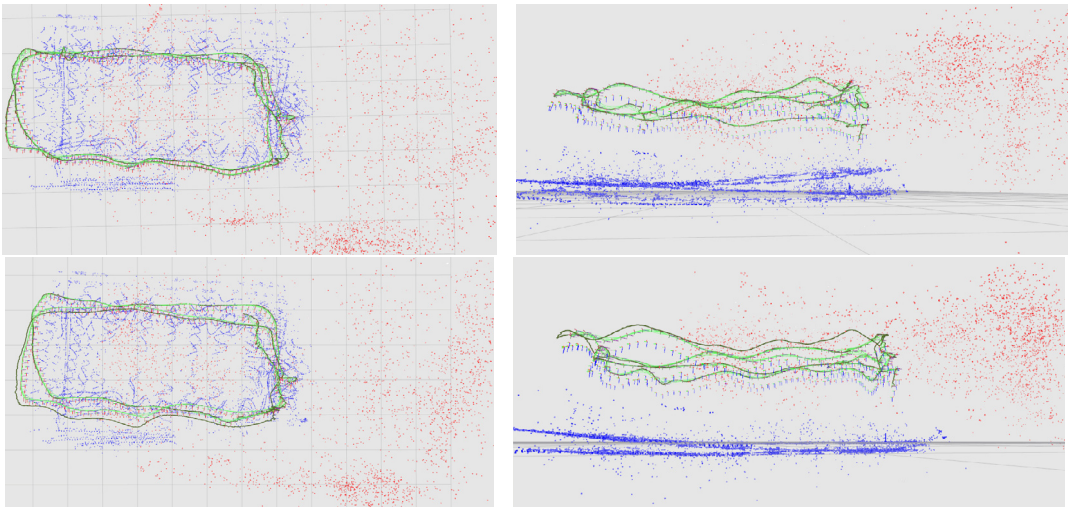

**Fig. 17.** Two views of the map built by the visual odometry (top) and the visual SLAM system (bottom) during the outdoor flight. Map points measured by the downward-looking camera are marked in blue, those measured by the forward-looking camera in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In general, doing PGO in the Similarity space Sim3, instead of SE3, can provide better scale corrections to a SLAM system [66], which could be a near future work. A map merging strategy would also be considered to achieve a more consistent map after loop closures. In [4], bundle adjustment in an inner window and pose graph optimization in an outer window are performed in an unified optimization framework. In our proposed system, only few keyframes are included in the local map, and PGO is performed within a relatively large number of pose graph. Whether the method proposed in [4] could further improve the accuracy of the map could be explored.

In the future, the effect of extrinsic camera calibration errors to the pose tracking and mapping accuracy could be analyzed. Furthermore, it would be interesting to investigate the fact that more parameters could be tracked in the optimizations of the SLAM system, if there is overlap in the fields of view of the multiple cameras: Based on more analysis to the differentiation problem, it is possible to integrate the online changes of camera extrinsic parameters into the optimization problem. Such online changes may happen when the cameras are not rigidly connected, e.g. when they are mounted on different wings of a fixed-wing aerial vehicle. For long term pose tracking of MAVs, an alternative way to better track the metric scale is to use more cameras running in an asynchronous way with reasonable areas of overlap in FOVs to get depth constraints of some features. Such depth constraints can be also utilized in the bundle adjustment process of the visual odometry, as shown in [50].

## References


[1] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, S. Tadokoro, Collaborative mapping of an earthquake-damaged building via ground and aerial robots, J. Field Robot. 29 (5) (2012) 832–841.

[2] K. Schauwecker, A. Zell, Robust and efficient volumetric occupancy mapping with an application to stereo vision, in: 2014 International Conference on Robotics and Automation, ICRA, 2014, pp. 6102–6107.

[3] M. Cummins, P. Newman, FAB-MAP: Probabilistic localization and mapping in the space of appearance, Int. J. Robot. Res. 27 (6) (2008) 647–665.

[4] H. Strasdat, A. Davison, J.M.M. Montiel, K. Konolige, Double window optimisation for constant time visual SLAM, in: Computer Vision, ICCV, 2011 IEEE International Conference on, 2011, pp. 2352–2359.

[5] S. Shen, Y. Mulgaonkar, N. Michael, V. Kumar, Vision-based state estimation for autonomous rotorcraft MAVs in complex environments, in: 2013 International Conference on Robotics and Automation, ICRA, Karlsruhe, Germany, 2013.

[6] S. Weiss, M.W. Achtelik, S. Lynen, M.C. Achtelik, L. Kneip, M. Chli, R. Siegwart, Monocular vision for long-term micro aerial vehicle state estimation: A compendium, J. Field Robot. 30 (5) (2013) 803–831.

[7] M. Li, A.I. Mourikis, High-precision, consistent EKF-based visual–inertial odometry, Int. J. Robot. Res. 32 (6) (2013) 690–711.

[8] J.A. Hesch, D.G. Kottas, S.L. Bowman, S.I. Roumeliotis, Camera-IMU-based localization: Observability analysis and consistency improvement, Int. J. Robot. Res. 33 (1) (2014) 182–201.

[9] G.H. Lee, F. Faundorfer, M. Pollefeys, Motion estimation for self-driving cars with a generalized camera, in: Computer Vision and Pattern Recognition, CVPR, 2013 IEEE Conference on, IEEE, 2013, pp. 2746–2753.

[10] S. Yang, S.A. Scherer, A. Zell, Visual SLAM for autonomous MAVs with dual cameras, in: 2014 International Conference on Robotics and Automation, ICRA, 2014, pp. 5227–5232.

[11] L. Heng, G.H. Lee, M. Pollefeys, Self-calibration and visual SLAM with a multi-camera system on a micro aerial vehicle, in: Proceedings of Robotics: Science and Systems, Berkeley, USA, 2014.

[12] H. Lu, S. Yang, H. Zhang, Z. Zheng, A robust omnidirectional vision sensor for soccer robots, Mechatronics 21 (2) (2011) 373–389.

[13] S. Yang, S. Scherer, A. Zell, Robust onboard visual SLAM for autonomous MAVs, in: Intelligent Autonomous Systems 13, in: Advances in Intelligent Systems and Computing, vol. 302, Springer International Publishing, 2016.

[14] D. Scaramuzza, F. Fraundorfer, Visual odometry [tutorial], IEEE Robot. Autom. Mag. 18 (4) (2011) 80–92.

[15] A. Davison, I. Reid, N. Molton, O. Stasse, MonoSLAM: Real-time single camera SLAM, IEEE Trans. Pattern Anal. Mach. Intell. 29 (6) (2007) 1052–1067.

[16] E. Eade, T. Drummond, Monocular SLAM as a graph of coalesced observations, in: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, 2007, pp. 1–8.

[17] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, in: Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR'07, Nara, Japan, 2007.

[18] L. Zhao, S. Huang, G. Dissanayake, Linear MonoSLAM: A linear approach to large-scale monocular SLAM problems, in: Robotics and Automation, ICRA, 2014 IEEE International Conference on, 2014, pp. 1517–1523.

[19] H. Strasdat, J.M.M. Montiel, A. Davison, Real-time monocular SLAM: Why filter? in: Robotics and Automation, ICRA, 2010 IEEE International Conference on, 2010, pp. 2657–2664.

[20] H. Strasdat, J. Montiel, A.J. Davison, Visual SLAM: Why filter? Image Vis. Comput. 30 (2) (2012) 65–77.

[21] F. Fraundorfer, L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, M. Pollefeys, Vision-based autonomous mapping and exploration using a quadrotor MAV, in: Intelligent Robots and Systems, IROS, 2012 IEEE/RSJ International Conference on, 2012, pp. 4557–4564.

[22] K. Schauwecker, A. Zell, On-board dual-stereo-vision for autonomous quadrotor navigation, in: Unmanned Aircraft Systems, ICUAS, 2013 International Conference on, 2013, pp. 333–342.

[23] M. Achtelik, M. Achtelik, S. Weiss, R. Siegwart, Onboard IMU and monocular vision based control for MAVs in unknown in-and outdoor environments, in: IEEE International Conference on Robotics and Automation, ICRA, 2011, pp. 3056–3063.

[24] S. Weiss, R. Siegwart, Real-time metric state estimation for modular vision-inertial systems, in: Robotics and Automation, ICRA, 2011 IEEE International Conference on, IEEE, 2011, pp. 4531–4537.
[25] S. Weiss, M. Achtelik, S. Lynen, M. Chli, R. Siegwart, Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments, in: Robotics and Automation, ICRA, 2012 IEEE International Conference on, 2012, pp. 957–964.
[26] S. Shen, Y. Mulgaonkar, N. Michael, V. Kumar, Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor, in: Robotics: Science and Systems, RSS, 2013.
[27] A.S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, N. Roy, Visual odometry and mapping for autonomous flight using an RGB-D camera, in: International Symposium on Robotics Research, ISRR, 2011 pp. 1–16.
[28] P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox, RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments, in: O. Khatib, V. Kumar, G. Sukhatme (Eds.), Experimental Robotics, in: Springer Tracts in Advanced Robotics, vol. 79, Springer, Berlin, Heidelberg, 2014.
[29] S. Scherer, A. Zell, Efficient onbard RGBD-SLAM for autonomous MAVs, in: Intelligent Robots and Systems, IROS, 2013 IEEE/RSJ International Conference on, 2013, pp. 1062–1068.
[30] R.G. Valenti, I. Dryanovski, C. Jaramillo, D.P. Strom, J. Xiao, Autonomous quadrotor flight using onboard RGB-D visual odometry, in: Robotics and Automation, ICRA, 2014 IEEE International Conference on, IEEE, 2014, pp. 5233–5238.
[31] R.T. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, et al., A System for Video Surveillance and Monitoring, vol. 2, Carnegie Mellon University, the Robotics Institute Pittsburg, 2000.
[32] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, S. Shafer, Multi-camera multi-person tracking for EasyLiving, in: Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on, 2000, pp. 3–10.
[33] R. Pless, Using many cameras as one, in: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, vol. 2, 2003, pp. –587–593.
[34] J.-M. Frahm, K. Köser, R. Koch, Pose estimation for multi-camera systems, in: C. Rasmussen, H. Bülthoff, B. Schölkopf, M. Giese (Eds.), Pattern Recognition, in: Lecture Notes in Computer Science, vol. 3175, Springer Berlin Heidelberg, 2004.
[35] M.E.M. Ragab, Multiple Camera Pose Estimation (Ph.D. thesis), The Chinese University of Hong Kong (People's Republic of China), 2008.
[36] G.H. Lee, F. Fraundorfer, M. Pollefeys, Structureless pose-graph loop-closure with a multi-camera system on a self-driving car, in: Intelligent Robots and Systems, IROS, 2013 IEEE/RSJ International Conference on, 2013, pp. 564–571.
[37] M. Kaess, F. Dellaert, Visual SLAM With a Multi-Camera Rig, Technical Report, Georgia Institute of Technology, 2006.
[38] C. Harris, M. Stephens, A combined corner and edge detector, in: The 4th Alvey Vision Conference, vol. 15, Manchester, UK, 1988, pp. 147–151.
[39] M. Kaess, F. Dellaert, Probabilistic structure matching for visual SLAM with a multi-camera rig, Comput. Vis. Image Underst. (ISSN: 1077-3142) 114 (2) (2010) 286–296.
[40] J. Solà, A. Monin, M. Devy, T. Vidal-Calleja, Fusing monocular information in multicamera SLAM, IEEE Trans. Robot. (ISSN: 1552-3098) 24 (5) (2008) 958–968.
[41] A. Harmat, I. Sharf, M. Trentini, Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle, in: Proc. 5th International Conference on Intelligent Robotics and Applications, in: Lecture Notes in Computer Science, vol. 7506, Springer Berlin Heidelberg, 2012.
[42] M.J. Tribou, A. Harmat, D.W. Wang, I. Sharf, S.L. Waslander, Multi-camera parallel tracking and mapping with non-overlapping fields of view, Int. J. Robot. Res. (2015).
[43] G. Carrera, A. Angeli, A. Davison, SLAM-based automatic extrinsic calibration of a multi-camera rig, in: Robotics and Automation, ICRA, 2011 IEEE International Conference on, 2011, pp. 2652–2659.
[44] L. Heng, P. Furgale, M. Pollefeys, Leveraging image-based localization for infrastructure-based calibration of a multi-camera rig, J. Field Robot. (ISSN: 1556-4967) (2014) 775–802.
[45] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: European Conference on Computer Vision, ECCV, Springer, 2006, pp. 430–443.
[46] P. Huber, Robust statistics, in: M. Lovric (Ed.), International Encyclopedia of Statistical Science, Springer Berlin Heidelberg, 2011.
[47] Z. Zhang, Parameter estimation techniques: a tutorial with application to conic fitting, Image Vis. Comput. (ISSN: 0262-8856) 15 (1) (1997) 59–76.
[48] S. Wright, J. Nocedal, Numerical Optimization, Springer, New York, 1999.
[49] J.-L. Blanco, A Tutorial on SE(3) Transformation Parameterizations and On-manifold Optimization, Technical Report, University of Malaga, 2010.
[50] S. Scherer, D. Dube, A. Zell, Using depth in visual simultaneous localisation and mapping, in: Robotics and Automation, ICRA, 2012 IEEE International Conference on, 2012, pp. 5216–5221.
[51] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004. http://dx.doi.org/10.1017/CBO9780511811685.
[52] S. Yang, S.A. Scherer, K. Schauwecker, A. Zell, Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision, J. Intell. Robot. Syst. (ISSN: 0921-0296) 74 (1–2) (2014) 27–43.
[53] S. Yang, S.A. Scherer, A. Zell, An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle, J. Intell. Robot. Syst. (ISSN: 0921-0296) 69 (2013) 499–515.
[54] S.A. Scherer, S. Yang, A. Zell, DCTAM: Drift-corrected tracking and mapping for autonomous micro aerial vehicles, in: Unmanned Aircraft Systems, ICUAS, 2015 International Conference on, IEEE, 2015, pp. 1094–1101.
[55] M. Calonder, V. Lepetit, C. Strecha, P. Fua, Brief: Binary robust independent elementary features, in: Computer Vision–ECCV 2010, 2010, pp. 778–792.
[56] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[57] H. Bay, T. Tuytelaars, L. Van Gool, SURF: Speeded up robust features, in: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, in: Lecture Notes in Computer Science, vol. 3951, Springer, Berlin, Heidelberg, 2006.
[58] D. Gálvez-López, J. Tardós, Bags of binary words for fast place recognition in image sequences, IEEE Trans. Robot. 28 (5) (2012) 1188–1197.
[59] X. Gao, X. Hou, J. Tang, H. Cheng, Complete solution classification for the perspective-three-point problem, IEEE Trans. Pattern Anal. Mach. Intell. 25 (8) (2003) 930–943.
[60] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, G2o: A general framework for graph optimization, in: Robotics and Automation, ICRA, 2011 IEEE International Conference on, 2011, pp. 3607–3613.
[61] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.
[62] L. Meier, P. Tanskanen, L. Heng, G. Lee, F. Fraundorfer, M. Pollefeys, PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision, Auton. Robots 33 (1) (2012) 21–39.
[63] D. Mellinger, N. Michael, V. Kumar, Trajectory generation and control for precise aggressive maneuvers with quadrotors, Int. J. Robot. Res. 31 (5) (2012) 664–674.
[64] Y. Liu, J.M. Montenbruck, P. Stegagno, F. Allgoewer, A. Zell, A robust nonlinear controller for nontrivial quadrotor maneuvers: Approach and verification, in: Intelligent Robots and Systems, IROS, 2015 IEEE/RSJ International Conference on, 2015, pp. 5410–5416.
[65] S. Lynen, M. Achtelik, S. Weiss, M. Chli, R. Siegwart, A robust and modular multi-sensor fusion approach applied to MAV navigation, in: Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems, IROS, 2013, pp. 3923–3929.
[66] H. Strasdat, J. Montiel, A.J. Davison, Scale drift-aware large scale monocular SLAM, in: Robotics: Science and Systems, 2010.

**Shaowu Yang** is an assistant professor with the State Key Laboratory of High Performance Computing (HPCL), National University of Defense Technology (NUDT), China. He received the M.Eng. degree in control science and engineering from NUDT, China, and the Ph.D. degree in computer science from the University of Tübingen, Germany, in 2009 and 2014, respectively. His current research interests include SLAM, micro aerial vehicles, environment modeling and geospatial information.

**Sebastian A. Scherer** works as a research engineer for Robert Bosch Gmbh Corporate Research, Future Systems Industrial Technology. Before that he was a research assistant and Ph.D. student at the chair of Cognitive Systems, University of Tübingen. He received his Dipl. Inform. degree from University of Tübingen in 2009 after studying computer science and physics at University of Tübingen and University of Waterloo. His research interests include computer vision, visual SLAM, and autonomous flying robots.

**Xiaodong Yi** is an associate professor in the State Key Laboratory of High Performance Computing. He received the Ph.D. degree in computer science from National University of Defense Technology. He has been devoted to Kylin operating system, which was used in Tianhe supercomputers, for over 10 years. His research interests include operating system, high performance computing, robotics software, etc.

**Andreas Zell** is a full professor at the University of Tübingen where he heads the chair of Cognitive Systems. He received the University diploma degree in computer science from the Univ. of Kaiserslautern, Germany, an M.S. degree from Stanford University, USA, and a Ph.D. degree in computer science from the University of Stuttgart, Germany. His research interests include machine learning methods, evolutionary algorithms, autonomous mobile robots, robot vision and navigation.