Visual SLAM for Autonomous MAVs with Dual Cameras

Shaowu Yang, Sebastian A. Scherer and Andreas Zell

***Abstract*— This paper extends a monocular visual simultaneous localization and mapping (SLAM) system to utilize two cameras with non-overlap in their respective field of views (FOVs). We achieve using it to enable autonomous navigation of a micro aerial vehicle (MAV) in unknown environments. The methodology behind this system can easily be extended to multi-camera rigs, if the onboard computation capability allows this. We analyze the iterative optimizations for pose tracking and map refinement of the SLAM system in multi-camera cases. This ensures the soundness and accuracy of each optimization update. Our method is more resistant to tracking failure than conventional monocular visual SLAM systems, especially when MAVs fly in complex environments. It also brings more flexibility to configurations of multiple cameras used onboard of MAVs. We demonstrate its efficiency with both autonomous flight and manual flight of a MAV. The results are evaluated by comparisons with ground truth data provided by an external tracking system.**

I. Introduction

Monocular vision systems with conventional lenses normally have rather limited FOVs. This is one of their disadvantages when being used for MAV navigation applications, since a larger FOV can provide better environmental awareness. In the context of visual SLAM on MAVs, a larger FOV of the vision system can be more resistant to tracking failure. The FOV can be enlarged by using a wider angle lens (even fish-eye lens), 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 [8]. Another type of omnidirectional vision systems combine multiple cameras into one vision system, maintaining a single-viewpoint projection model. However, these cameras need to be very precisely configured in relatively heavy mechanical systems, in order to preserve this model, and thus are not flexible enough for MAV applications.

In this paper, we focus on achieving autonomous navigation of MAVs by extending the Parallel Tracking and Mapping (PTAM) described in [6] to utilize image features from multiple cameras. We expand the FOV of our MAV vision system by using two cameras mounted looking in two different directions (forwards and downwards) to capture more critical views, as shown in Fig. 1. The choice in the number of cameras is resulted from a compromise between tracking robustness and onboard computation capability. 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 configurations of cameras more flexible. On the other hand, since multiple cameras no longer preserve this model, using features from multiple cameras in PTAM is not trivial: How the features are involved in the iterative optimizations needs to be carefully analyzed. Based on such an 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.

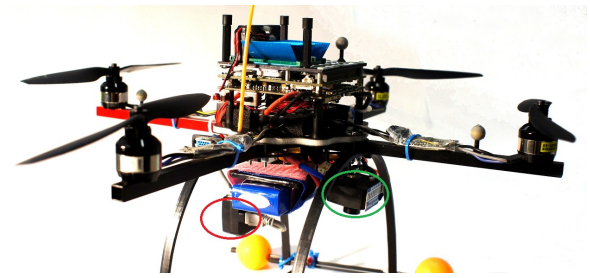


Fig. 1: Our MAV platform, with two cameras mounted looking in two different directions: downwards (green ellipse) and forwards (red ellipse).

II. Related Work

Vision-based onboard solutions are becoming a popular research focus for autonomous navigation of MAVs. Autonomous mapping and exploration of MAVs based on stereo cameras is demonstrated in [4]. The work in [12] features a vision system for autonomous navigation of MAVs using two pairs of stereo cameras, and stereo triangulations serve as constraints in bundle adjustment of PTAM. A stereo setup yields metric scale information of the environment. However, stereo visual odometry and SLAM systems still may degenerate to the monocular case when the distance to the scene is much larger than the stereo baseline. In [1], 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 [15] presents a visual-inertial data fusion method onboard of MAVs for navigation in unknown environments. A vision-based system combining the advantages of monocular vision and stereo vision is developed in [14], which uses a low frame-rate secondary camera to extend the high frame-rate forward facing camera equipped with a fish-eye lens. The resulting

S. Yang and S. A. Scherer are PhD candidates, and A. Zell is full professor, with the Department of Computer Science, University of Tübingen, Tübingen, Germany {shaowu.yang, sebastian.scherer, andreas.zell} @uni-tuebingen.de

©2014 IEEE

vision system relies mainly on monocular vision algorithms, while being able to track metric scale by stereo triangulation.

Pose estimation using multi-camera systems can already be found in literature, e.g. the work in [7] adopts a generalized camera model for a multi-camera system to estimate the ego-motion of a self-driving car. Another work most similar to our current work, is that in [5], which uses PTAM with multiple cameras mounted on a buoyant spherical airship. It employed 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. Our improvements comparing to it are in three aspects. First, we provide a solid mathematical analysis on how measurements from different cameras can be integrated in each optimization process of PTAM. The analysis guarantees soundness and accuracy of the optimizations for pose update and bundle adjustment using measurements from multiple cameras. Second, we make use of the fact that multiple cameras are typically mounted rigidly, and force camera poses to obey their rigid extrinsic calibration in bundle adjustment as will be shown in Sect. III. This ensures a consistent map in multi-camera cases. Furthermore, the resulting pose tracking accuracy in [5] was evaluated only in a manual flight experiment, and the position errors are reported to be higher than our results although stereo cameras were used there. We also demonstrate that our SLAM system can enable autonomous navigation of a MAV.

III. Extending the Monocular Visual SLAM

We implemented our SLAM system based on the open source PTAM system. The reason for this choice is that PTAM provides an efficient tracking module and it is able to generate an accurate map with a large number of map points from the environment. Furthermore, PTAM uses iterative optimizations for both pose tracking and map refinement, which we could extend to incorporating multi-camera image features. In this section, we analyze how image features from different cameras can be integrated in one SLAM system.

A. The Basics of PTAM

In order to achieve real-time operation, the main idea proposed in PTAM is to split tracking and mapping into two separate threads, which can be processed in parallel on a dual-core computer. The first *tracking* thread is responsible for real-time tracking of the camera motion relative to the current map. The second *mapping* thread extends the map, consisting of 3D point features organized in keyframes, and refines it using bundle adjustment.

Within the tracking thread, the FAST corner detector [11] is applied to each image at four pyramid levels, and all map points are projected to the current image plane based on a prior pose estimate. Successful matches between image features and reprojected map points are then used for pose update computation. The mapping thread integrates new keyframes into the map when requested by the tracking thread, and creates new map points by triangulating FAST corner matches between the new keyframe and its closest neighbours. Local bundle adjustment and global bundle adjustment are continuously performed to refine the map for the rest of the time.

For both pose tracking and map refinement (using bundle adjustment), iterative minimization of the reprojection errors of those matched map points serves as a major step in PTAM.

In the context of autonomous navigation of MAVs using PTAM, there are three issues we need to keep in mind: First, as a monocular system, it does not provide metric scale measurements, which will be addressed in Sect. III-G. Second, it is originally designed for augmented reality applications used in small areas, thus not suitable for large-scale SLAM. Third, if MAVs fly in ways which lead to failure in triangulating new map points or tracking the existing points, pose tracking will consequently fail.

B. Camera Projection Model and Pose Update

Using the same calibrated camera projection model as in [6], the image projection of the j^{th} map point to the i^{th} camera is

$$\mathbf{u}_{ji} = \mathbf{P}_i\left(E_{c_iw}\mathbf{p}_j\right), \tag{1}$$

where $\mathbf{P}_i$ is the i^{th} camera projection function considering lens distortion (see [6]), $\mathbf{p}_j$ is world coordinates of the j^{th} map point, and E_{c_iw} is a member of the Lie group $SE(3)$, which 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 specified 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. Following this idea, pose updates of all cameras can be expressed with one single six-element vector μ using the exponential map:

$$E'_{c_iw} = E_{i1} \cdot e^{\mu} \cdot E_{c_1w}, \tag{2}$$

where μ is an element of the $se(3)$ algebra, and E_{i1} is the pose of C_1 in the i^{th} camera coordinate system. The pose tracking (and a part of mapping) problem of the SLAM system now mainly consists of how to obtain an optimized μ as the pose update of C_1. The advantage of the parameterization of the camera pose updates using the six-element vector μ is that it allows a closed form differentiation of Eq. 2.

C. Optimizations for Pose Update and Bundle Adjustment

The camera pose update and bundle adjustment in PTAM 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. In a n-camera (or n-keyframe) system, we need to minimize the function:

$$\sum_{i=1}^{n}\sum_{j\in S_i}\mathrm{Obj}\left(\frac{|\,\mathbf{e}_{ji}\,|}{\sigma_{ji}},\sigma_T\right), \tag{3}$$

where Obj is the Tukey biweight objective function, σ_{ji} is the estimated standard deviation of the image reprojection of point j in pixels, and σ_T is a robust estimate of the standard deviation of all reprojection errors. $\mathbf{e}_{ji}$ is defined as the difference between the image reprojection of map point j and its actual image measurement:

$$\mathbf{e}_{ji} = \mathbf{u}_{ji} - \hat{\mathbf{u}}_{ji}. \tag{4}$$

The minimization problems can be solved by iterations of reweighted least squares. This requires us to differentiate $\mathbf{e}_{ji}$ (i.e. to obtain the Jacobians of $\mathbf{e}_{ji}$) with respect to the estimated camera poses at each iteration step. In bundle adjustment, the differentiation of $\mathbf{e}_{ji}$ with respect to map point j position changes is also required. The work in [2] provides a good tutorial to related mathematics. We will discuss the differentiations of $\mathbf{e}_{ji}$ in multi-camera systems in the following two sub-sections.

D. Camera Pose Update with Multiple Cameras

For the pose update of a n-camera system, the optimization problem is to find the optimal camera C_1 pose update μ:

$$\mu' = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j \in S_i} \operatorname{Obj}\left(\frac{|\, \mathbf{e}_{ji} \,|}{\sigma_{ji}}, \sigma_T \right) \tag{5}$$

Following the discussion in Sect. III-C, we analyze the differentiations required for solving the optimization. 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 μ using the chain rule as

$$\mathbf{J}_{1\mu} = \frac{\partial \mathbf{P}_1\left(e^{\mu} E_{c_1 w} \mathbf{p}_j\right)}{\partial \mu} = \left.\frac{\partial \mathbf{P}_1(\mathbf{c})}{\partial \mathbf{c}}\right|_{\mathbf{c}=E_{c_1 w}\mathbf{p}_j} \cdot \frac{\partial\left(e^{\mu} E_{c_1 w} \mathbf{p}_j\right)}{\partial \mu}. \tag{6}$$

The first term of the above matrix product is the Jacobian of the camera projection model. The last term is:

$$\frac{\partial\left(e^{\mu} E_{c_1 w} \mathbf{p}_j\right)}{\partial \mu} = \begin{pmatrix} \mathbf{I}_3 & -[E_{c_1 w}\mathbf{p}_j]_\times \end{pmatrix}. \tag{7}$$

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

$$\mathbf{J}_{i\mu} = \frac{\partial \mathbf{P}_i\left(E_{i1} e^{\mu} E_{c_1 w} \mathbf{p}_j\right)}{\partial \mu} \tag{8}$$

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

Its difference to Eq. 6 lies in the last term of this equation:

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

where $Rot(E_{i1})$ is the rotation component of E_{i1}.

E. Bundle Adjustment with Multiple Rigid Camera Rigs

Bundle adjustment in PTAM means solving the following minimization problem:

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

where N is the number of keyframes and M is the number of observed map points that need to be updated.

In a multi-camera system, we assume that relative poses of 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 keyframes K_1 obtained by the first camera C_1. The pose of other rigidly connected keyframes are computed based on the updated poses of corresponding keyframes in K_1. This allows a consistent map to be built using multiple cameras.

In this case, to solve bundle adjustment, we differentiate $\mathbf{e}_{ji}$ with respect to the corresponding keyframe (in K_1) pose update μ, which can be obtained in the same way as in Eq. 6 or Eq. 8, depending on the camera identity of the the point j (i.e. by which camera the point is measured). The Jacobian of $\mathbf{e}_{ji}$ with respect to the estimated point j pose can be expressed in a consistent way:

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

The last term simply becomes:

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

F. Some Implementation Details

For autonomous navigation of our MAV, we utilize two cameras with non-overlapping FOVs. This configuration achieves a maximal effective FOV of the vision system.

1) Mapping: Since the two cameras have very different perspectives, we assume they share no common feature points. Then the global map of the SLAM system can be treated as two sub-maps, each corresponding to one camera. Such organization makes map operations more efficient. Note that it is trivial in PTAM to assume multiple cameras can share common points, since both map point triangulation and bundle adjustment are designed to handle multiple observations of one feature point. Then we only triangulate new map points with keyframes obtained by the same camera. However, in bundle adjustment, map points and keyframes from both cameras are involved. To achieve better real-time performance of the SLAM system when operating in large scale, we only retain the local bundle adjustment step and abandon global bundle adjustment in mapping.

2) Pose tracking: Map points in the two sub-maps are reprojected to the corresponding source camera to decide whether they are potentially visible. Successful matches between image features and those potentially visible points

will serve as image measurements and be used in the iterative optimizations for camera pose update. In the optimization, we assume the two cameras produce measurement noises with the same four standard deviations on the four image pyramid levels. This also applies to the optimization in bundle adjustment. These standard deviations for systems using significant different cameras or lenses can be estimated according to the noises of all measurements as did in [13].

G. Automatic Initialization of the SLAM System

Metric scale ambiguity generally exists in monocular camera systems. Our dual-camera system has the same issue since the cameras have no overlap in their corresponding FOVs. We solve it by initializing the metric map of our SLAM system similarly to [16]. We use an initialization module developed in [17] to robustly estimate the downward looking camera (the first camera) pose during the takeoff phase of our MAV. The main idea of the pose estimation method in [17] is to apply a computational geometry method to a known circular pattern which is detected by using an artificial neural network. 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 outdoor experiment in [16]. The sub-map corresponding to the first camera is initialized with those feature points. The sub-map corresponding to the second camera is initialized after two keyframes from this camera are obtained, when 3D feature points can be triangulated with the known keyframe poses.

IV. Experiments

A. Experimental Setup

1) Quadrotor platform: Our MAV is based on the open source quadrotor platform developed by the PIXHAWK project from ETH Zürich, described in [9], as shown in Fig. 1. The onboard computer features an Intel Core 2 Duo 1.86GHz CPU, 2 GB DDR3 RAM and a 32Gb SSD. The pxIMU inertial measurement unit and autopilot board mainly consists of a microcontroller unit (MCU) for position and attitude control, and sensors including a tri-axis accelerometer and a tri-axis gyroscope. The two synchronised cameras utilized on our MAV are two PointGrey Firefly MV monochrome cameras, each of which weighs only 37 grams. Each camera has an image resolution of 640×480 pixels, a maximum frame rate of 60 fps, and both lenses we use have viewing angles of approximately 90 degrees.

2) Extrinsic Calibration of Cameras: We calibrate the extrinsic parameters between the first camera and other cameras (E_{i1}) by utilizing a commercial external tracking system, which we mentioned in [16], and a pattern which is also used for camera intrinsic parameters calibration. Our quadrotor and the pattern poses can be measured by the tracking system. The camera poses with respect to the pattern can be obtained by performing extrinsic calibration of each camera. Then E_{i1} can be obtained after a few coordinate transformations with a similarly pre-calibrated first-camera to quadrotor pose. When an external system is unavailable, an alternative way could be using SLAM-based methods like the one presented in [3].

3) Quadrotor controllers: We use a nested PID pose controller and PD trajectory controller described in previous work [16] for autonomous navigation of our quadrotor. The 3D position estimates and the yaw angle estimates from our visual SLAM system are fed to the position controller at frame rate. The attitude controller runs at a frequency of 200 Hz, using the roll and pitch estimates from the IMU.

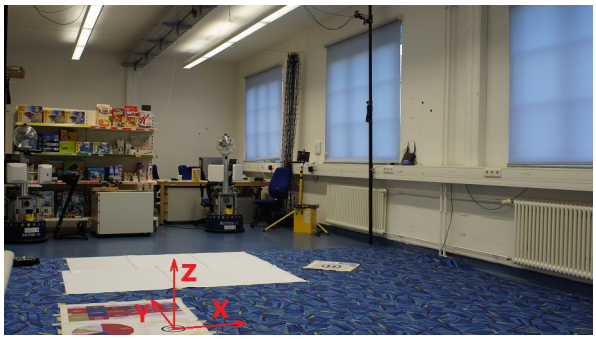


Fig. 2: A scene of our robot lab where we carry out the experiments. The x, y, z axes of the SLAM coordinate system are indicated inside.

TABLE I: MAV pose RMSEs of the whole trajectories in autonomous flight (Auto.) and manual flight (Manual), with position errors in *mm* 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

B. Enabling Autonomous Navigation

In this first experiment, we demonstrate the efficiency of our SLAM system to enable autonomous navigation of MAVs. We further evaluate its accuracy by comparing its pose tracking results to the data provided by the external tracking system. A picture of the experiment environment is shown in Fig. 2, in which we also sketch the SLAM coordinate system. There is a large white area on the desired path of the MAV, where 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. 3b) in a counter-clockwise direction with a commanded forward speed of $v_s = 0.4m/s$, taking off and finally landing above the origin of the SLAM coordinate system. The takeoff phase is controlled by using pose fed from the initialization module. We set the MAV to turn 90 degrees at the first corner, which makes the forward looking camera unable to triangulate new features and track its pose if it is the only camera in the SLAM system.

TABLE II: MAV pose RMSEs during the part of manual flight when the MAV pose can be tracked by the downward looking camera along, with position errors in *mm* 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

The resulting MAV trajectory during an autonomous flight can be found in Fig. 3. The MAV trajectory estimated by our onboard SLAM system using two cameras (SLAM2c) fits well with the ground truth data from the external tracking system (ETS). The SLAM2c attitude estimates are actually less noisy than that of the ETS data. The root-mean-square errors (RMSEs) of the pose estimates of SLAM2c data with respect to the ETS data are listed in Table I (the row of Auto.). Three noise sources which contribute to the errors should be noted. First, slow scale drift still exists in our SLAM system, since we do not use additional sensor data 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 perfectly coincide with the SLAM coordinate frame.

In Fig. 3b, we can find fluctuations in the performed trajectory at the designated path corners. The reasons are that we set the MAV to hover $5sec$ at each corner, and we have not implemented a sophisticated and precise pose controller, which is out of the scope of this paper. 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. 3b.

C. More Evaluation through Manual Flight

We process an image logfile off-board to perform more evaluation to our SLAM system. The onboard computation capability does not allow us to take image logfiles during autonomous navigation. Thus, we manually control the MAV to follow a similar path as in Sect. IV-B, and take a logfile containing images from both cameras and other useful onboard sensor data by utilizing ROS [10].

The MAV trajectory during this manual flight is shown in Fig. 4. When using the proposed SLAM system with two cameras, the MAV pose can be well tracked throughout the flight, which also fits well with the ground truth data. The corresponding RMSEs are listed in Table I (the row of Manual). Fig. 5 shows a view of the final map built by our SLAM system with the dual-camera trajectory. Two minor parts of the ground truth data are missing due to the flight outside the working area of the tracking system, which is found to be two periods of straight dashed lines in Fig. 4c around the time of $17sec$ and $29sec$.

However, if we use PTAM with the downward looking camera alone, pose tracking will fail when the MAV flies above the white area (see the SLAMdc case in Fig. 4). The MAV position where pose tracking fails in this case is marked with black circles in Fig. 4a and Fig. 4b. We mark the time when it fails with a red line in Fig. 4c, where 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 II. Similarly, if we use PTAM with only the forward looking camera after the initialization of its sub-map, pose tracking will fail again when the MAV rotates during hovering. Like in the first failure case, we mark this failure with black crosses in Fig. 4a and Fig. 4b, and a black line in Fig. 4c.

V. Conclusions and Discussions

We present a visual SLAM system which can utilize feature measurements from multiple cameras. We demonstrate the efficiency of the method by enabling a MAV with two cameras to navigate autonomously along a predefined path. The experiment with a logfile taken from a manual flight proves that our proposed method is more resistant to tracking failure than a monocular method. A demonstration of our work can be found in the accompanying video or online at `http://www.youtube.com/channel/UCQd6_G6qyvGHUmz7NUelDZQ/videos`.

For long term pose tracking of MAVs, its metric scale could be better tracked by fusing IMU data. An alternative way is to use more cameras running in an asynchronous way with reasonable areas of overlap in FOVs to get depth constraints of some features. A loop closing method will also be integrated to form a full SLAM system for navigating the MAV in large scale environments.

References

[1] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart. Onboard IMU and Monocular Vision Based Control for MAVs in Unknown in-and Outdoor Environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3056–3063, 2011.

[2] José-Luis Blanco. A tutorial on SE(3) transformation parameterizations and on-manifold optimization. Technical report, University of Malaga, September 2010.

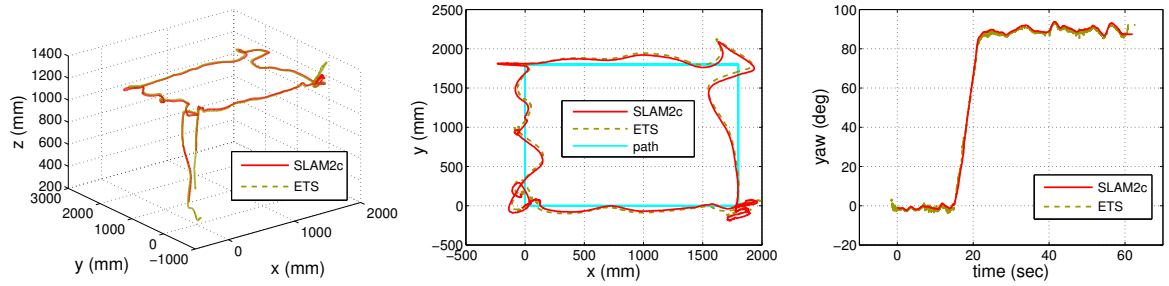
[3] G. Carrera, A. Angeli, and A.J. Davison. SLAM-based automatic extrinsic calibration of a multi-camera rig. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2652–2659, 2011.

[4] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys. Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4557–4564, 2012.

[5] A. Harmat, I. Sharf, and M. Trentini. Parallel Tracking and Mapping with Multiple Cameras on an Unmanned Aerial Vehicle. In *International Conference on Intelligent Robotics and Applications (ICIRA)*, volume 1, pages 421–432. Springer, 2012.

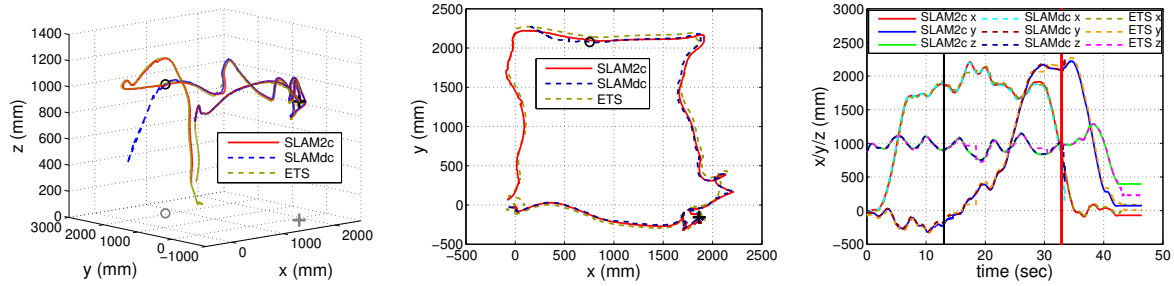
[6] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[7] Gim Hee Lee, Friedrich Faundorfer, and Marc Pollefeys. Motion estimation for self-driving cars with a generalized camera. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2746–2753. IEEE, 2013.



(a) (b) (c)

Fig. 3: MAV poses estimated by our SLAM system (SLAM2c) and the external tracking system (ETS) during autonomous navigation. (a) The trajectory on x, y, z axis in 3D and (b) projected to the $x-y$ plane. (c) The yaw angle of the MAV.



(a) (b) (c)

Fig. 4: 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, y, z axis in 3D, (b) projected to the $x-y$ plane, and (c) with respect to the flight time.

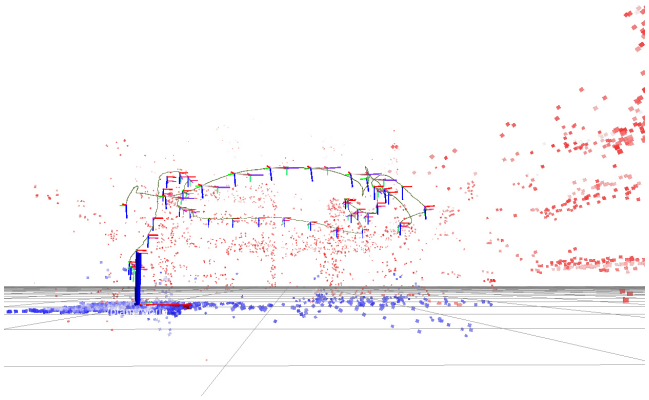


Fig. 5: Built map with the dual-camera trajectory of the manual flight logfile (with better view in color). Map points from the downward looking camera are marked in blue, those from the forward looking camera in red. More details of the results can be found in the accompanying video.

[8] Huimin Lu, Shaowu Yang, Hui Zhang, and Zhiqiang Zheng. A Robust Omnidirectional Vvision Sensor for Soccer Robots . *Mechatronics*, 21(2):373 – 389, 2011.

[9] L. Meier, P. Tanskanen, L. Heng, G. Lee, F. Fraundorfer, and M. Pollefeys. PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight Using Onboard Computer Vision. *Autonomous Robots*, pages 1–19, 2012.

[10] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, 2009.

[11] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443. Springer, 2006.

[12] Konstantin Schauwecker and Andreas Zell. On-board dual-stereo-vision for autonomous quadrotor navigation. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 333–342, 2013.

[13] Sebastian A. Scherer, Daniel Dube, and Andreas Zell. Using depth in visual simultaneous localisation and mapping. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 5216–5221. IEEE, 2012.

[14] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-based State Estimation for Autonomous Rotorcraft MAVs in Complex Environments . In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Karlsruhe, Germany, may 2013.

[15] Stephan Weiss and Roland Siegwart. Real-time metric state estimation for modular vision-inertial systems. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4531–4537. IEEE, 2011.

[16] Shaowu Yang, Sebastian A. Scherer, Konstantin Schauwecker, and Andreas Zell. Autonomous Landing of MAVs on an Arbitrarily Textured Landing Site Using Onboard Monocular Vision. *Journal of Intelligent & Robotic Systems*, 74(1-2):27–43, 2014.

[17] Shaowu Yang, Sebastian A. Scherer, and Andreas Zell. An Onboard Monocular Vision System for Autonomous Takeoff, Hovering and Landing of a Micro Aerial Vehicle. *Journal of Intelligent & Robotic Systems*, 69:499–515, 2013.