Autonomous Robots manuscript No.
(will be inserted by the editor)

Self-Calibration and Visual SLAM with a Multi-Camera System on a Micro Aerial Vehicle

Lionel Heng · **Gim Hee Lee** · **Marc Pollefeys**

Received: date / Accepted: date

Abstract The use of a multi-camera system enables a robot to obtain a surround view, and thus, maximize its perceptual awareness of its environment. If vision-based simultaneous localization and mapping (vSLAM) is expected to provide reliable pose estimates for a micro aerial vehicle (MAV) with a multi-camera system, an accurate calibration of the multi-camera system is a necessary prerequisite. We propose a novel vSLAM-based self-calibration method for a multi-camera system that includes at least one calibrated stereo camera, and an arbitrary number of monocular cameras. We assume overlapping fields of view to only exist within stereo cameras. Our self-calibration estimates the inter-camera transforms with metric scale; metric scale is inferred from calibrated stereo. On our MAV, we set up each camera pair in a stereo configuration which facilitates the estimation of the MAV's pose with metric scale. Once the MAV is calibrated, the MAV is able to estimate its global pose via a multi-camera vSLAM implementation based on the generalized camera model. We propose a novel minimal and linear 3-point algorithm that uses relative rotation angle measurements from a 3-axis gyroscope to recover the relative motion of the MAV with metric scale from 2D-2D feature correspondences. This relative motion estimation does not involve scene point triangulation. Our constant-time vSLAM implementation with loop closures runs on-board the MAV in real-time. To the best of our knowledge, no published work has demonstrated real-time on-board vSLAM with loop closures. We show experimental results from simulation experiments, and real-world experiments in both indoor and outdoor environments.

Keywords Micro Aerial Vehicles · Multi-Camera Systems · Self-Calibration · Simultaneous Localization and Mapping

L. Heng
Information Division, DSO National Laboratories, 20 Science Park Drive, Singapore 118230, Singapore
E-mail: hjianyon@dso.org.sg

G.H. Lee
Mitsubishi Electric Research Laboratories, 201 Broadway, 8th Floor, Cambridge, MA 02139, USA
E-mail: lee@merl.com

M. Pollefeys
Computer Vision and Geometry Group, ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland
E-mail: marc.pollefeys@inf.ethz.ch

1 Introduction

Vision-based MAVs are more versatile than laser-based MAVs. Whereas a laser only provides geometry data, a camera can provide both geometry data via stereo and structure-from-motion techniques, and appearance data. A camera is a passive sensor while a laser is an active sensor and is thus susceptible to interference. Furthermore, a camera is lighter and has a smaller footprint. However, utmost care has to be taken when choosing the camera configuration for a vision-based MAV expected to operate robustly in challenging environments. A single-camera configuration introduces limited perceptual awareness, and in turn, flight constraints because if the camera observes too few features for some time, localization can fail and lead to a crash. In the case of a single downward-looking camera (Weiss et al., 2013), the MAV cannot fly too close to the ground which often has little texture, and at the same time, it cannot perform obstacle avoidance due to the absence of a forward-looking camera. In the case of a forward-looking camera, constraints are imposed on the path planning. For example, in Prentice and Roy (2009), any planned path ensures that there are a sufficient number of features for localization. In addition, in Heng et al. (2011), a path is planned such that the MAV does not move outside the camera's field of view, and inadvertently crash into an unseen obstacle.

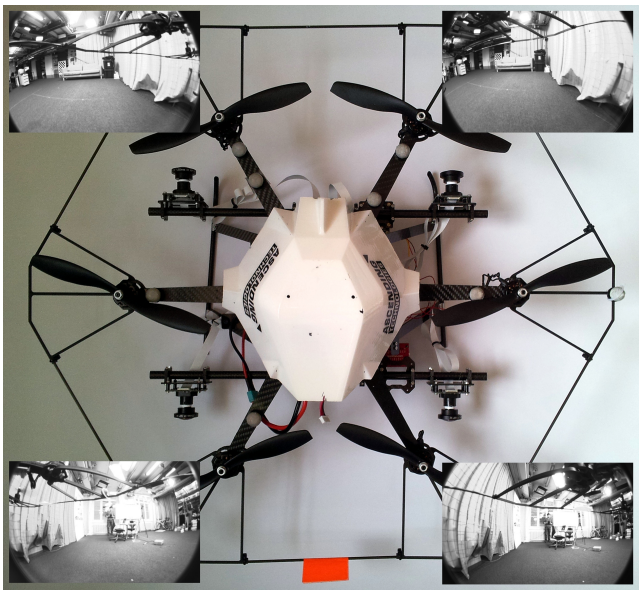


Fig. 1 We show an image from each of the four fish-eye cameras on our MAV platform. Our camera configuration provides a surround view. Each pair of cameras is arranged in a stereo configuration so that 3D scene information is available at all times.

In this paper, we use a multi-camera system on a MAV; together with the use of fish-eye lenses, this camera configuration provides a surround view of the vicinity. Our multi-camera approach is similar in spirit to Furgale et al. (2013b) who uses a car equipped with multiple cameras configured to provide a surround view. The use of such a multi-camera system eliminates the constraints on path planning associated with a single camera as discussed earlier, and allows for immediate path planning in all directions. Furthermore, localization is more robust as there is always at least one camera that observes a sufficient number of features. In addition, the use of fisheye lenses increases the robustness of localization as it precludes the possibility that a large object will occupy a large portion of the field of view of any camera, and cause a complete loss of feature tracks for that camera. Another benefit of using multiple cameras is that we directly infer metric scale.

For autonomous MAV flight to succeed, we need to estimate the MAV's pose with metric scale. Pose estimates with incorrect metric scale can cause control stability issues and incorrectly-scaled maps. Metric scale can be inferred from either accelerometer measurements (Weiss et al., 2013) or the use of multiple cameras. However, the significant measurement noise associated with MEMS accelerometers which are commonly present on MAVs limits the scale accuracy, and to maximize the scale accuracy, occasional excitation of the MAV is needed. In contrast, the use of multiple cameras allows direct inference of metric scale.

Our MAV is self-contained in the sense that all algorithms necessary for autonomous flight are run on-board. With this self-contained MAV, we circumvent latency and reliability issues related to off-board computing, especially in areas where Wi-Fi reception is poor. On the other hand, more cameras correlate to a requirement for more computational resources, but with recent advances in computing hardware, especially in multi-core processors, it is now computationally feasible to run image processing algorithms on-board a MAV that uses multiple cameras.

For robust operation in the field, we assume that the multi-camera system includes at least one calibrated stereo camera; in this way, 3D scene information is always available, and we can directly infer metric scale from calibrated stereo. Thus, we avoid the use of failure-prone map initialization methods required for monocular cameras and multi-camera systems with non-overlapping fields of view.

The use of multiple cameras on a MAV raises two issues we have to resolve before we can realize autonomous flight: finding the inter-camera transforms, and estimating the MAV's pose with this synchronized multi-camera system. Inaccurate inter-camera transforms will cause vSLAM to wrongly estimate the MAV's pose. Hence, it is imperative that any calibration method must be able to estimate the inter-camera transforms with high precision. Furthermore, it is ideal that the calibration method be unsupervised. As calibration parameters drift over time due to wear and tear, calibration has to be carried out repeatedly, and tends to be a tedious process, especially if operator involvement is required. We want calibration to be a process that can be easily carried out, does not require special calibration setups, and minimizes the need for supervision by an operator.

We come up with a novel method for self-calibration of a multi-camera system with at least one calibrated stereo camera and multiple calibrated monocular cameras. For clarity, we refer to either a monocular camera or a stereo camera as a camera entity. Each of the two monocular cameras that make up a stereo camera is not considered as a camera entity. We do not assume overlapping fields of view between any pair of camera entities. Our self-calibration method is based on vSLAM, only assumes that each monocular and stereo camera is pre-calibrated, and does not require operator input, fiducial markers or external positioning systems. We are able to estimate the extrinsic calibration parameters with metric scale which is inferred from calibrated stereo. We do not know of other existing vSLAM-based self-calibration methods for a multi-camera system with at least one calibrated stereo camera that is able to estimate the extrinsic calibration parameters with metric scale and without requiring an additional sensor for inferring metric scale. Optionally, our self-calibration method finds the rotation between the multi-camera system and the gyroscope which is required for rotating gyroscopic measurements from the gyroscope's reference frame into the multi-camera system's reference frame. We obtain this rotation by solving the rotational part

of the hand-eye equation (Daniilidis, 1999) based on gyroscopic measurements and the orientations of the multi-camera system which are estimated as a by-product by the self-calibration method.

We also come up with a novel method for motion estimation with a multi-camera system based on the generalized camera model (Pless, 2003). We propose a 3-point minimal and linear algorithm for motion estimation that uses relative rotation angle measurements to recover the relative motion with metric scale. Scene point triangulation is not required here as we directly estimate the relative motion from 2D-2D feature correspondences. This algorithm is based on the generalized camera concept that treats all cameras as a single object. By assuming that the relative orientation is known from relative rotation angle measurements, we simplify our motion estimation algorithm such that we require 3 point correspondences instead of 17 point correspondences to find a linear solution. As a result, our 3-point motion estimation algorithm is able to compute the relative motion with a small number of computations. We obtain relative rotation angle measurements from a 3-axis gyroscope. We do not see the requirement of a 3-axis gyroscope as restrictive as this gyroscope is typically part of the inertial measurement unit (IMU) commonly found on a MAV. To obtain relative rotation angle measurements with respect to the multi-camera system, we synchronize the gyroscope with the multi-camera system, and obtain the rotation between the reference frames of the multi-camera system and the gyroscope from our self-calibration method.

We incrementally build a graph of keyframes and constraints, and use the double-window optimization method (Strasdat et al., 2011) to optimize this graph. This graph optimization enables real-time on-board SLAM with loop closures on a vision-based MAV, which to the best of our knowledge, has not been shown before in published works.

This paper is an expanded version of our work in Heng et al. (2014b). We provide additional details on our self-calibration method and vSLAM implementation. Furthermore, we extend our self-calibration method for a multi-stereo-camera system to a multi-camera system with at least one calibrated stereo camera, and an arbitrary number of calibrated monocular cameras. Additional experiments are carried out to validate our self-calibration method for both types of multi-camera systems, evaluate alternatives to our self-calibration methods, and look at the impact of calibration errors on our vSLAM implementation.

2 Related Work

There is an extensive body of work on calibration for multi-sensor systems. For tractability, we look at current state-of-the-art methods that involve cameras. Kelly and Sukhatme (2011) perform simultaneous self-calibration and visual-inertial SLAM with a camera-IMU system. The calibration parameters are only accurate as long as the camera-IMU system experiences constant excitation about all rotation and acceleration axes of the IMU. Compared to online methods, offline methods improve the accuracy of visual-inertial calibration. Furgale et al. (2013a) proposes an offline method to find an accurate estimate of the transform and the temporal offset between a camera and an IMU using a calibration pattern. This method makes use of temporal basis functions for parameterization of continuous-time variables, and continuous-time batch estimation. Similarly to Kelly and Sukhatme (2011), the method requires excitation about all axes in order for the calibration parameters to be accurate. Brookshire and Teller (2012) adopt a hand-eye calibration approach in which they calibrate a multi-sensor system based on relative motion measurements with metric scale for each sensor. In the case of the visual-inertial calibration approach applied to multi-camera systems, we can calibrate each camera-IMU pair, and infer the inter-camera transforms from the set of camera-IMU transforms. Similarly, in the case of the hand-eye calibration approach applied to multi-camera systems, we can calibrate each pair of cameras, and infer the inter-camera transforms. However, the estimated inter-camera transforms may not be accurate as the calibration does not consider explicit constraints between camera entities with non-overlapping fields of view; these constraints are in the form of 3D scene points mutually observed by such camera entities at different times. In contrast, in our self-calibration, we exploit these 3D scene points to obtain a more accurate estimate of inter-camera transforms.

Carrera et al. (2011b); Harmat et al. (2012); Heng et al. (2013) develop self-calibration methods for multi-camera systems that are based on vSLAM. However, Carrera et al. (2011b) only estimates the inter-camera transforms up to scale. Harmat et al. (2012) solves the scale issue by using a calibration pattern to infer metric scale. Their SLAM implementation tracks a mixture of calibration pattern landmarks and natural features; the calibration pattern landmarks have known 3D coordinates, and hence, are used as a bootstrapping step for estimating the inter-camera transforms with metric scale. Heng et al. (2013) uses odometry data to infer metric scale; this method is restricted to ground robots with odometry sensors, and is not applicable to a MAV. In contrast, our self-calibration method requires neither a calibration pattern nor odometry data, and yet, is able to estimate the inter-camera transforms with metric scale. This is made possible by utilizing calibrated stereo. With our self-calibration method, a robot can self-calibrate anywhere in the field without the need for special calibration setups as long as the stereo calibration parameters remain stable.

We explore existing work on SLAM with a multi-camera system (Kaess and Dellaert, 2006; Kim et al., 2008; Carrera et al., 2011a; Harmat et al., 2012; Tribou, 2014). Kaess

and Dellaert (2006) solve an optimization problem comprising pose-point constraints and odometry constraints in order to obtain the pose of the multi-camera system, and do not perform loop closure detection. Kim et al. (2008) models a multi-camera system as a spherical camera. The drawback with this spherical model is that the relative motion of the system can only be estimated up to scale. Furthermore, they do not show experiments with their multi-camera system on unmanned aerial vehicles. Carrera et al. (2011a) uses a forward-looking camera and a backward-looking camera on a ground robot, and implements pose-graph SLAM based on odometry data and loop closure detection. The absence of bundle adjustment in any form limits the metric accuracy of the map. Harmat et al. (2012) implements a multi-camera version of PTAM on an airship equipped with three cameras; two of these cameras have overlapping fields of view. At the beginning, they initialize a map with metric scale using triangulated points from stereo. Subsequently, they incrementally build the map, and at the same time, perform map-based localization. Tribou (2014) removes the requirement for overlapping fields of view from the multi-camera implementation of PTAM. They rely on bundle adjustment running in a dedicated mapping thread to gradually recover metric scale. Initial scene point depths are not known with high certainty, and hence, an initialized map does not have accurate metric scale. Due to the cubic complexity of bundle adjustment used by PTAM, the approaches of Harmat et al. (2012); Tribou (2014) do not scale to large environments. In addition, both approaches do not perform loop closures.

The described existing work on SLAM models the multiple cameras as separate entities. There is a body of work (Grossberg and Nayar, 2001; Pless, 2003) that proposes a single general camera model to model multiple cameras as a single entity. In this model, pixels observed in a camera are represented as lines that intersect the projection center and the 3D point corresponding to the pixel. In Schweighofer et al. (2008), an alternative and more efficient approach to bundle adjustment is proposed for a multi-camera system using the general camera model proposed by Grossberg and Nayar (2001). Given initial estimates of the multi-camera system poses, the 3D feature points are linearly estimated in closed-form. In turn, they optimize the translational components of the multi-camera system poses by solving a closed-form linear problem, before optimizing the rotational components by solving the absolute orientation problem. This approach requires initial estimates of the multi-camera system poses, and if the initial estimates are poor, many iterations are needed to reach convergence. In contrast, the generalized camera model proposed by Pless (2003) is more well-established; a considerable number of works (Stewénius et al., 2005; Li et al., 2008; Lee et al., 2013a,b) use the generalized camera model. In our work, we use this generalized camera model.

We choose between motion estimation and pose estimation techniques when computing the current pose of the MAV. Motion estimation involves computing the relative motion between two frames from 2D-2D feature correspondences. Pose estimation involves computing the pose with respect to either a past frame or a map using 2D-3D feature correspondences. For motion estimation with a multi-camera system, we can estimate the relative motion with metric scale as long as there are either inter-camera feature correspondences from overlapping fields of view or significant inter-frame rotations without overlapping fields of view. Li et al. (2008) shows that in the case of only intra-camera feature correspondences, and the relative rotation being an identity matrix, the generalized epipolar constraint reduces to the epipolar constraint which only allows us to recover the relative pose up to scale. In contrast, wide overlapping fields of views for each stereo camera on our MAV ensure that we always have inter-camera feature correspondences, and thus, are able to obtain pose estimates with metric scale all the time. For pose estimation with a multi-camera system, given a map with correct metric scale, we can estimate the pose with metric scale from 2D-3D feature correspondences. Lee et al. (2013a) proposes a 3-point non-linear algorithm for pose estimation that returns up to 8 solutions. In contrast, our 3-point motion estimation algorithm is significantly more computationally efficient as this algorithm is linear and returns an unique solution. Although our 3-point motion estimation algorithm requires relative rotation angle measurements, we do not see this requirement as restrictive as an IMU is generally available on a MAV and provides relative rotation angle measurements via a gyroscope. A number of works (Lee et al., 2014) exploit the vertical direction information provided by the IMU to simplify motion estimation algorithms; the vertical direction is not susceptible to drift unlike gyroscopic measurements. However, the IMU present on MAVs is of the MEMS type. We used a Vicon motion capture system to compute the maximum error of the vertical direction measured by the MEMS IMU on-board our MAV platform; this maximum error is 3 degrees. As a result, we did not get accurate relative pose estimates from using the known vertical in our motion estimation algorithm.

We then look at vision-based MAVs that run real-time visual odometry algorithms on-board. Schauwecker and Zell (2014) deploy one downward-looking stereo camera and one forward-looking stereo camera on a MAV, independently estimate the MAV's pose from each camera, and fuse both pose estimates. In contrast, we use all cameras to obtain a single pose estimate. Weiss et al. (2013) utilizes both an IMU and downward-looking camera together with a modified version of PTAM to estimate the MAV's pose, and infer metric scale from accelerometer measurements. Schmid et al. (2013) uses a FPGA board to compute depth maps from a forward-looking stereo camera and relies on stereo

visual odometry for pose estimation. Shen et al. (2013) uses a forward-looking stereo camera with fish-eye lenses. Here, they use gyroscopic measurements to filter out incorrect feature correspondences, and uses a local map to estimate the MAV's pose. We note that the visual odometry in all discussed works is susceptible to drift as loop closure detection is not carried out. To the best of our knowledge, there is no published work on real-time on-board SLAM with loop closures for vision-based MAVs.

3 MAV Platform

We use a AscTec Firefly equipped with an Intel Core i7 single computer board and the ROS[1] framework for message transmission. The IMU on the AscTec Firefly is time-synchronized to the single computer board. A VRmagic D3 four-camera system is mounted on the AscTec Firefly. In this multi-camera system, each camera is connected to an ARM Cortex A8 board via a proprietary cable. A Lensagon 1.5 mm f/2.0 fish-eye lens with a 185° field of view is fitted to each camera. The IMU on the AscTec Firefly sends a periodic trigger signal at 15 Hz to the ARM board which then grabs pixel-synchronous 754×480 monochrome images from all four cameras. Each time a trigger is sent, the IMU publishes inertial data. In this way, we facilitate inertial-visual fusion by making inertial information available for each image. Furthermore, all sensor measurements are timestamped with respect to the system clock on the single board computer. On the ARM board, we use the RTI Connext DDS middleware[2] to transmit image data over an Ethernet connection to the single computer board. We note that this high-bandwidth image transmission is not possible with ROS as ROS is only able to publish a maximum of 15 images per second on the ARM board given the computational constraints of the ARM processor. Still, we leverage ROS for all other message transmissions due to the wide variety of libraries and ease of use that ROS offers for message data manipulation.

To enable autonomous flight, we use the state estimation framework from Weiss et al. (2013) for robust pose estimation by fusing both inertial data from the IMU and pose data from vSLAM.

For the purpose of brevity, we define the following symbols to be used in this paper. We define a n-camera system to contain cameras $C_1,\ldots,C_n$. In this multi-camera system, there are s stereo cameras and m monocular cameras such that $2s+m=n$. Each stereo camera S_i comprises the camera pair $\{C_{2i-1},C_{2i}\}$ for $1\leq i\leq s$. Likewise, each monocular camera M_j corresponds to camera C_{2s+j} for $1\leq j\leq m$. For each camera C_i, we denote its intrinsics as K_{C_i}, and its extrinsics with respect to the IMU's reference frame V as $[R_{C_i},t_{C_i}]$. No overlapping fields of view are assumed to exist between any two camera entities belonging to the set $\{S_1,\ldots,S_s,M_1,\ldots,M_m\}$.

[1] `http://www.ros.org`

[2] `https://www.rti.com/products/dds/index.html`

4 Self-Calibration

Our self-calibration method utilizes vSLAM with natural features, and accurately estimates the inter-camera transforms for a multi-camera system with at least one calibrated stereo camera and any number of calibrated monocular cameras. Furthermore, our self-calibration method is able to estimate the rotation between the reference frames of the multi-camera system and a 3-axis gyroscope. This rotation is required for rotating the gyroscopic measurements from the gyroscope's reference frame into the multi-camera system's reference frame. One by-product of the self-calibration is a metrically accurate and globally consistent map which can be used for map-based localization (Sattler et al., 2011; Lim et al., 2012) and infrastructure-based calibration (Heng et al., 2014a).

Figure 2 shows the pipeline underlying the self-calibration method. For all stereo and monocular cameras, we perform stereo and monocular visual odometry (VO) respectively. Using the poses of stereo camera S_1 estimated by stereo VO as a reference, and the poses of all other cameras estimated by VO, step 1 gives us an initial estimate of the transform between S_1 and each of all other cameras. In steps 2-5, we improve these initial estimates by finding 3D scene points mutually observed by camera entities, and using these scene points as a prior to recover accurate inter-camera transforms. Step 2 merges maps from all cameras into a single map. In steps 3-4, we obtain globally consistent pose estimates for the multi-camera system. In addition, we obtain inter-camera feature correspondences in the form of loop closures classified as correct. These feature correspondences provide a strong prior for accurate inter-camera transforms, and in step 5, allow us to recover an accurate estimate of the inter-camera transforms. Step 6 which is an optional step yields the rotation between the reference frames of the multi-camera system and the gyroscope.

4.1 Monocular and Stereo Camera Calibration

Our self-calibration method assumes that each camera entity is calibrated. In this step, we calibrate all monocular and stereo cameras that constitute the multi-camera system. We use the unified projection model and plumb bob distortion model (Mei and Rives, 2007) to model the camera intrinsics. In our chessboard-based calibration, we detect the chessboard in each image, and after a minimum number of chessboards is detected, we use the method in Mei and Rives

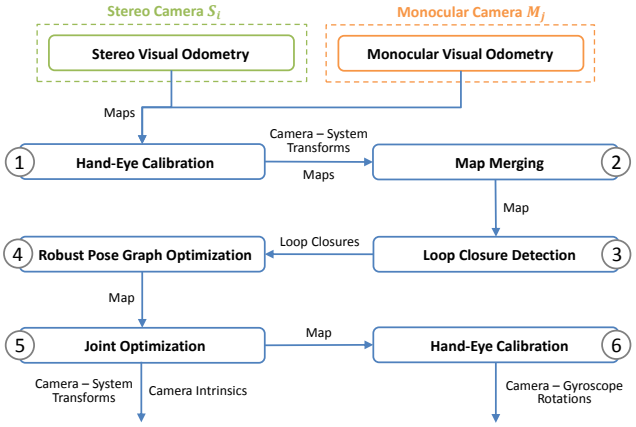


Fig. 2 Our self-calibration pipeline estimates the inter-camera transforms and camera-gyroscope rotations.

(2007) to find the initial values for the camera poses and intrinsic parameters. In the case of a monocular camera, we subsequently use non-linear refinement to optimize the intrinsic parameters and monocular camera poses. In the case of a stereo camera, from the camera poses, we infer the stereo camera poses and the stereo transform between the two cameras, and subsequently, use non-linear refinement to optimize the intrinsic parameters, stereo camera poses, and the stereo transform.

4.2 Extrinsic Self-Calibration

4.2.1 Stereo VO

In this step, each stereo camera builds its own map. For each stereo frame, we use the CenSurE feature detector (Agrawal et al., 2008) implemented in OpenCV to detect features in each image. For each feature, we compute both the ORB feature descriptor (Rublee et al., 2011) and the backprojected unitary ray. We match features between the two images, and for each feature match with corresponding backprojected rays $(\mathbf{r}_1, \mathbf{r}_2)$, if ${\mathbf{r}_2}^T \mathbf{E} \mathbf{r}_1$ is less than a threshold where $\mathbf{E}$ is the essential matrix representing the stereo transform, we mark the feature match as valid and triangulate the feature match.

We define the reference frame of the stereo camera pair $\{C_i, C_{i+1}\}$ to be that of camera C_i. To compute the current pose of the stereo camera, we find 2D-3D correspondences between the first images of the previous and current stereo frames, and use the P3P method (Kneip et al., 2011) together with RANSAC to find the pose that corresponds to the highest number of inlier correspondences. We optimize the pose via sliding window bundle adjustment. Here, each error residual is equivalent to the dot product between the observed backprojected ray and the ray passing through the camera center and the 3D scene point. This dot product is faster to compute compared to the image reprojection error.

4.2.2 Monocular VO

As in the previous step, each monocular camera builds its own map. However, this map is only up to scale. For each frame, we use the CenSurE feature detector and ORB feature descriptor to detect features and compute their descriptors respectively. At the same time, we compute the backprojected unitary ray for each feature. We use the five-point algorithm (Nister, 2004) together with RANSAC to compute the relative camera motion between the first two frames, and then, use the estimated relative camera motion to triangulate the inlier feature correspondences found from the five-point algorithm coupled with RANSAC. At each subsequent frame, we use the P3P method (Kneip et al., 2011) together with RANSAC to compute the camera pose and identify inlier feature correspondences, and subsequently, apply sliding window bundle adjustment.

4.2.3 Hand-Eye Calibration

In this step, we find an initial estimate of the transforms between S_1 and each of all other camera entities. Without loss of generality, we assume that the reference frame of the multi-camera system is the same as that of stereo camera S_1, which in turn, is the same as that of camera C_1. Using the optimized stereo camera poses for S_1 and the poses of all other camera entities from VO as input, we use the hand-eye calibration method (Daniilidis, 1999) to find the transforms between S_1 and S_i for $2 \leq i \leq s$ and the extended hand-eye calibration method (Schmidt et al., 2005) to find the transforms between S_1 and M_j for $1 \leq j \leq m$. As a result, we have an initial estimate for the pose of each camera entity with respect to the multi-camera system's reference frame.

4.2.4 Map Merging

In this step, we merge the maps from all cameras. At the beginning, we set the poses of the multi-camera system to be the same as those of S_1 which were computed by stereo VO described in Section 4.2.1. For each stereo and monocular camera, we infer the camera poses by multiplying the inverse of the poses of S_1 and the initial estimate of the transform between S_1 and that camera as computed by the hand-eye calibration step described in Section 4.2.3. For each scene point observed by a stereo camera S_i for $2 \leq i \leq s$, we recompute its 3D coordinates by multiplying the pose of S_i at which the scene point was first observed, and the stereo 3D coordinates with respect to S_i and which were computed during stereo triangulation in the VO step described in Section 4.2.1. For each scene point observed by a monocular

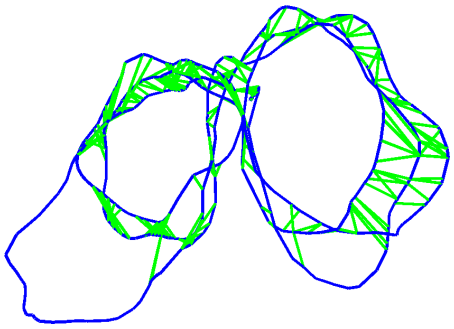


Fig. 3 We illustrate an example of loop closure detection between different camera entities for a multi-camera system comprising one stereo camera and one monocular camera. A blue line represents the pose graph while green lines represent loop closures.

camera M_j for $1 \leq j \leq m$, we recompute its 3D coordinates by using stereo triangulation based on the poses associated with the first two frames the scene point was observed in. Non-linear refinement is used to optimize the inter-camera transforms and scene points; here, the poses of the multi-camera system are kept fixed.

4.2.5 Loop Closure Detection

The loop closure detection step is the first of the next three steps that together make the merged map built by the multi-camera system globally consistent. Furthermore, we use detected loop closures as a strong prior for recovering accurate inter-camera transforms in the joint optimization step. In this loop closure detection step, we detect loop closures within stereo cameras S_i for $1 \leq i \leq s$, within monocular cameras M_j for $1 \leq j \leq m$, and between different camera entities. For the first image in each frame associated with a camera entity, we find the n most similar images, and we filter out matched images whose timestamps are too close to the that of the query frame. This is to avoid unnecessary linking of adjacent frames. Again, we use the P3P method (Kneip et al., 2011) together with RANSAC to find an inlier set of 2D-3D correspondences for each of the n images. We add a loop closure between the query frame and the frame which has the highest number of 2D-3D correspondences provided that this number exceeds a threshold. Figure 3 shows an example of detected loop closures between different camera entities that make up a multi-camera system containing one stereo camera and one monocular camera.

4.2.6 Robust Pose Graph Optimization

This step finds correct loop closures. From loop closures identified as correct, we obtain feature correspondences between different camera entities, and these feature correspondences allow the joint optimization to recover an accurate estimate of the transforms between the camera entities. At the same time, we obtain globally consistent estimates of the poses of the multi-camera system which are a better initial guess for bundle adjustment compared to the reference poses of S_1 estimated by stereo VO and which have drift. Hence, better convergence is achieved in bundle adjustment. We build a pose graph in which the nodes are the poses of the multi-camera system, and edges between nodes correspond to measurements of relative 6D transforms obtained from either VO or loop closures. We use the approach in Lee et al. (2013c) to simultaneously optimize this pose graph, and classify loop closures as either correct or wrong. During the pose graph optimization, each scene point is rigidly attached to the first pose at which the scene point was first observed. At the end of the optimization, the scene points are consistent with the new poses. Furthermore, we merge pairs of duplicate 3D scene points corresponding to correct loop closures.

4.2.7 Joint Optimization

We jointly optimize the camera extrinsics, camera poses, and 3D scene points while keeping the stereo transforms fixed. In the joint optimization, we minimize a cost function which is the sum of squared image reprojection errors of the 3D scene points:

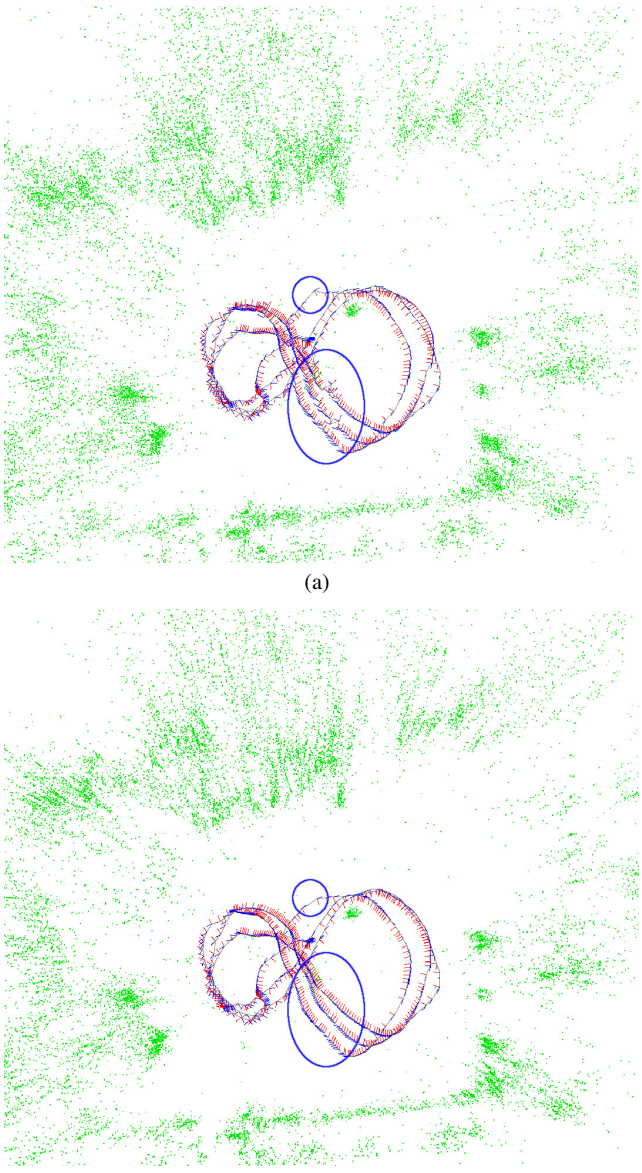
$$\min_{\mathbf{K}_c,\mathbf{P}_i,\mathbf{T}_c,\mathbf{X}_p} \quad \sum_{c,i,p} w_p \rho \left(\|\Delta \mathbf{z}_{c,i,p}\|^2 \right), \tag{1}$$

where

$$\Delta \mathbf{z}_{c,i,p} = \pi(\mathbf{K}_c, \mathbf{P}_i, \mathbf{T}_c, \mathbf{X}_p) - \mathbf{p}_{cip}. \tag{2}$$

π is a projection function that predicts the image coordinates of the scene point $\mathbf{X}_p$ seen in camera c given the camera's intrinsic parameters $\mathbf{K}_c$, the multi-camera system pose $\mathbf{P}_i$, and the transform from the camera frame to the multi-camera system frame $\mathbf{T}_c$. $\mathbf{p}_{cip}$ is the observed image coordinates of $\mathbf{X}_p$ seen in camera c with the corresponding multi-camera system pose $\mathbf{P}_i$. ρ is a robust cost function used for minimizing the influence of outliers.

3D scene points observed by multiple camera entities usually make up a small percentage of all scene points in the map. In our experiments, scene points observed by multiple camera entities make up approximately 10% of all scene points. To ensure that these scene points make a significant contribution to accurate estimation of the inter-camera transforms, we assign a higher value to w_p for feature observations that are associated with such scene points. Figure 4 illustrates an example of joint optimization for a two-stereo-camera system. We observe in this figure that the joint optimization correctly smooths out the jerky portions of the MAV's trajectory which are enclosed in blue circles. The jerky portions occur because of few features observed by the reference stereo camera S_1 in the area which lead to the estimation of poses with high uncertainty for S_1.



(a) (b)

Fig. 4 We show an example of joint optimization for a two-stereo-camera system. (a) and (b) show the map before and after joint optimization respectively. Blue circles enclose portions of the MAV's trajectory which were jerky before the joint optimization and smooth after the joint optimization.

4.2.8 Hand-Eye Calibration

We find the rotation between the reference frames of the gyroscope and the multi-camera system by linearly solving for the rotational component of the hand-eye equation (Daniilidis, 1999). After the hand-eye calibration, we have the estimates of the camera-gyroscope rotations.

5 Visual SLAM

In this section, we describe the algorithms used in our keyframe-based vSLAM implementation. We propose a novel 3-point algorithm to estimate the relative motion of the MAV with metric scale and with respect to the current keyframe. As this 3-point motion estimation algorithm makes use of the relative rotation measurement from the gyroscope via short-term integration of gyroscopic measurements, the accuracy of this relative rotation measurement with respect to the current keyframe drops over time due to gyroscopic drift. Hence, after a certain period of time during which there is no new keyframe, we switch to the pose estimation method (Lee et al., 2013a) which is also based on the generalized camera model and uses 3 correspondences. This period of time depends on the maximum gyroscopic drift between two frames and at which the motion estimation algorithm still performs reasonably well. In our case of a MEMS gyroscope with considerable drift, the time period for switching over from motion estimation to pose estimation in the absence of new keyframes is 1 second. This time period can be longer if we use a better-quality gyroscope such as a fiber-optic or ring-laser gyroscope, or use a filter on-board the MAV to estimate the gyroscopic drift and corrected gyroscopic measurements. Our motion estimation technique is far more computationally efficient than the pose estimation technique. Furthermore, our motion estimation algorithm is linear and computes one unique solution, while the pose estimation algorithm is non-linear and returns up to 8 solutions. A computational analysis reveals that the number of arithmetic operations required by the pose estimation algorithm is a very high multiple of that required by the motion estimation algorithm on the order of ten thousands. In addition, our motion estimation algorithm does not require scene point triangulation unlike the pose estimation algorithm which requires a set of 3D point landmarks. However, the pose estimation algorithm does not assume prior knowledge of the inter-frame rotation unlike the motion estimation algorithm.

We mark the current frame as a keyframe if the number of correspondences falls below a threshold. Over time, we incrementally build a graph of keyframes and constraints obtained from both visual odometry and loop closures. We choose the double-window optimization method (Strasdat et al., 2011) to optimize the graph as the ability of this method to run in constant time makes real-time vSLAM on-board a MAV feasible.

5.1 Motion Estimation

Here, we describe in depth our novel 3-point algorithm for motion estimation based on the generalized camera model.

5.1.1 Generalized Epipolar Constraint (GEC)

Pless (2003) introduced the generalized camera model for a multi-camera system which allows for non-central projection. In this model, we replace each image pixel $\mathbf{x}$ in camera C_i with a ray expressed as a Plücker line 6-vector $\mathbf{L}$ that passes through the camera center of C_i and the normalized image point $\hat{\mathbf{x}} = \mathbf{K}_{C_i}^{-1}\mathbf{x}$:

$$\mathbf{L} = \begin{bmatrix} \mathbf{q}^T & \mathbf{q}'^T \end{bmatrix}^T, \tag{3}$$

where $\mathbf{q}$ and $\mathbf{q}'$ are the direction and moment vectors:

$$\mathbf{q} = \mathbf{R}_{C_i}\hat{\mathbf{x}}, \quad \mathbf{q}' = \mathbf{t}_{C_i} \times \mathbf{q}. \tag{4}$$

We write the GEC (Pless, 2003) as

$$\mathbf{L}_2{}^T \underbrace{\begin{bmatrix} \mathbf{E} & \mathbf{R} \\ \mathbf{R} & 0 \end{bmatrix}}_{\mathbf{E}_{GC}} \mathbf{L}_1 = 0, \tag{5}$$

where $\mathbf{L}_1 \leftrightarrow \mathbf{L}_2$ are two Plücker line vectors representing a ray correspondence between two generalized camera frames V_1 and V_2, and $\mathbf{E}_{GC}$ is the 6×6 generalized essential matrix in which the first item is the conventional essential matrix $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$ where $\mathbf{R}$ and $\mathbf{t}$ are the rotation and translation from V_1 to V_2. We note that for the conventional essential matrix, $\mathbf{t}$ is computed only up to scale, but for a generalized camera, the scale of $\mathbf{t}$ can be computed as shown in Li et al. (2008). In the degenerate case of no inter-camera feature correspondences and $\mathbf{R}$ being an identity matrix, the scale of $\mathbf{t}$ cannot be recovered. However, the assumption that the multi-camera system on a MAV includes at least one calibrated stereo camera ensures that there exists at least one wide overlapping field of view, and that, in turn, inter-camera feature correspondences are always present. Thus, we are always able to compute the scale of $\mathbf{t}$.

5.1.2 Minimal 3-Point Algorithm

We estimate R from short-term integration of gyroscopic measurements between the two frames V_1 and V_2. Substituting $\mathbf{L}_1 = [\mathbf{q}_1^T \ \mathbf{q}_1'^T]^T$ and $\mathbf{L}_2 = [\mathbf{q}_2^T \ \mathbf{q}_2'^T]^T$ into and rearranging equation 5 in the form $\mathbf{At} = \mathbf{b}$, we get

$$\underbrace{\mathbf{q}_1^T \mathbf{R}^T [\mathbf{q}_2]_\times}_{\mathbf{A}} \mathbf{t} = \underbrace{-\mathbf{q}_1^T \mathbf{R}^T \mathbf{q}_2' - \mathbf{q}_1'^T \mathbf{R}^T \mathbf{q}_2}_{\mathbf{b}}. \tag{6}$$

As $\mathbf{t}$ has three unknown variables, we require 3 Plücker line correspondences to solve for $\mathbf{t}$. Given 3 Plücker line correspondences, we construct the 3×3 matrix

$$\mathbf{C} = \begin{bmatrix} \mathbf{A}_1^T & \mathbf{A}_2^T & \mathbf{A}_3^T \end{bmatrix}^T, \tag{7}$$

and the 3-vector

$$\mathbf{D} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix}^T, \tag{8}$$

Table 1 Comparisons of total number of iterations needed for RANSAC ($w = 0.5$ and $p = 0.99$)

Algorithm	# RANSAC Iterations	# Solutions	Total
Minimal 3-Point	34	1	34
Minimal 6-Point (Stewénius et al., 2005)	292	64	18688
Linear 17-Point (Pless, 2003)	603606	1	603606

and solve the system of linear equations $\mathbf{Ct} = \mathbf{D}$ to obtain $\mathbf{t}$. We note that this 3-point algorithm is linear and does not require scene point triangulation.

5.1.3 Robust Estimation

We make our 3-point algorithm robust to outliers by implementing it within RANSAC. We determine the best solution by choosing the solution that has the highest number of inliers.

The number of iterations k needed in RANSAC for a n-point algorithm is given by $k = \frac{\ln(1-p)}{\ln(1-w^n)}$, where n is the number of correspondences, w is the probability that any selected correspondence is an inlier, and p is the probability that all the selected correspondences are inliers. The total number of iterations m needed to run RANSAC while evaluating s solutions is given by $m = k \times s$. Table 1 shows the number of iterations required by our 3-point algorithm; for comparison, we include the 6-point minimal solution (Stewénius et al., 2005) and linear 17-point solution (Pless, 2003) to the GEC problem. We observe that our 3-point algorithm requires much fewer iterations compared to the 6-point and 17-point solutions; in other words, our 3-point algorithm is computationally efficient to a large degree when estimating the relative motion. We then optimize the relative motion estimate returned by RANSAC using non-linear refinement and the inlier set of correspondences associated with the relative motion estimate.

5.2 Pose Estimation

When there is no new keyframe for some time, the relative rotation measurement between the current keyframe and the current frame becomes inaccurate due to gyroscopic drift, and using such rotation measurements will cause our 3-point algorithm for motion estimation to return inaccurate estimates. In this case, we switch to the pose estimation algorithm (Lee et al., 2013a) based on 3D scene points observed in the current keyframe. In our implementation, we switch to the pose estimation algorithm if there is no new keyframe for 1 second. In this pose estimation algorithm,

we require only 3 correspondences to recover the absolute pose associated with the current frame. The pose estimation algorithm returns up to 8 solutions, but in most cases, 2 solutions are returned. As in motion estimation, we use the RANSAC framework for robustness to outliers, and to find the best solution that corresponds to the highest number of inliers.

5.3 Location Recognition

To ensure that the map in the immediate vicinity of the MAV is globally consistent, we find loop closures. We use the same loop closure detection technique discussed in Section 4.2.5 to find loop closures.

5.4 Optimization

For graph optimization, we use the double-window optimization method (Strasdat et al., 2011) which we implement using Google's Ceres Solver (Agarwal et al., 2013). In double-window optimization, we simultaneously optimize both an inner window and outer window which correspond to sets of pose-point constraints and sets of pose-pose constraints respectively. For robustness to outliers, we use the Huber and Cauchy robust cost functions respectively for the inner and outer windows. We make one modification to the double-window optimization such that we include residuals, each of which corresponds to the error between the vertical direction associated with the MAV's estimated pose and the vertical direction measurement from the accelerometer. This modification ensures that the map is aligned with the ground plane.

6 Experiments and Results

We conduct experiments to evaluate the accuracy of our self-calibration method and vSLAM implementation. We use a Vicon motion capture system in most experiments for the sole purpose of collecting ground truth data. The results of these experiments are described in detail in this section.

6.1 Self-Calibration

We use a Vicon motion capture system to evaluate the accuracy of the parameters estimated by our self-calibration method. To obtain ground-truth estimates, we devise a Vicon-based calibration method in which we use the Vicon system to track the pose of both the MAV and a chessboard moving across the field of view of each camera. Hence, the transform between the chessboard and MAV is known at any time step. From intrinsic camera calibration which also computes the camera poses with respect to the chessboard, we obtain an initial guess of both the intrinsic camera parameters and the transform between each camera and the MAV. We optimize these intrinsic parameters and camera-MAV transforms via non-linear refinement. From Vicon pose measurements and gyroscopic measurements, we perform a hand-eye calibration to find a ground-truth estimate of the rotation between the reference frames of the MAV and the gyroscope.

We apply our self-calibration method to varying multi-camera system configurations, and compare the estimated calibration parameters against the ground-truth calibration parameters estimated by the Vicon-based calibration. Prior to running our self-calibration method, we separately calibrate each camera entity on the MAV. For each run of self-calibration, we carry the MAV along a figure-8 path several times while varying the orientation of the MAV. During this manoeuvre, the MAV's heading is roughly parallel to the direction of motion. This figure-8 manoeuvre ensures that we get a high number of loop closures, and in turn, feature correspondences between different camera entities, and thus, ensures that the self-calibration method produces accurate calibration parameters. Any other manoeuvre is useful provided that each camera entity has observed most parts of the environment by the end of the manoeuvre, and there are many loop closures as a result. By varying the orientation of the MAV continuously, we ensure that the rotation between the reference frames of the multi-camera system and gyroscope is fully observable. The calibration parameters are not fully observable if there are no or few loop closures between camera entities or the 3-axis gyroscope is not sufficiently excited around all 3 rotation axes.

In addition, we evaluate alternatives to our self-calibration method: hand-eye calibration and camera-IMU self-calibration. We compare our self-calibration method to these methods in terms of the accuracy of the estimated inter-camera transforms.

6.1.1 Multi-Camera System With Two Stereo Cameras

We apply our self-calibration method to our MAV whose multi-camera system comprises two stereo cameras. The figure-8 manoeuvre took 90 seconds and the travelled distance was 54.16 m. Our self-calibration took 16 minutes, and the resulting average reprojection error associated with the generated map was 0.659 pixels. The map had 23957 scene points with an average scene point depth of 4.36 m.

We report in the first column of Table 2 the estimated rotation in terms of roll, pitch, and yaw angles, and translation of camera C_i with respect to camera C_1 for $i = 2,3,4$. We also report the estimated rotation of camera C_1 with respect to the gyroscope. The second column of Table 2 shows the differences between the estimated rotations/translations

Table 2 Comparison of our estimated inter-sensor transforms from the self-calibration method with ground-truth estimates from the Vicon-based calibration method. ${}^{i}\mathbf{R}_{j}$ is the rotation given in roll, pitch, and yaw angles between sensors i and j. ${}^{i}\mathbf{t}_{j}$ is the translation between sensors i and j.

	Our self-calibration	Difference with Vicon-based calibration
${}^{C_1}\mathbf{R}_{C_2}$	$[-0.06^\circ\ 0.20^\circ\ -1.39^\circ]$	$[0.001^\circ\ 0.035^\circ\ 0.016^\circ]$
${}^{C_1}\mathbf{t}_{C_2}$	[31.89 −0.10 0.04] cm	[0.15 0.01 0.11] cm
${}^{C_1}\mathbf{R}_{C_3}$	$[177.41^\circ\ 0.85^\circ\ 176.59^\circ]$	$[0.177^\circ\ 0.025^\circ\ 0.029^\circ]$
${}^{C_1}\mathbf{t}_{C_3}$	[1.38 3.31 −28.21] cm	[0.67 0.14 0.55] cm
${}^{C_1}\mathbf{R}_{C_4}$	$[176.20^\circ\ 0.34^\circ\ 177.63^\circ]$	$[0.183^\circ\ 0.060^\circ\ 0.086^\circ]$
${}^{C_1}\mathbf{t}_{C_4}$	[32.66 1.47 −27.96] cm	[0.73 0.06 0.61] cm
${}^{GYRO}\mathbf{R}_{C_1}$	$[-94.09^\circ\ 2.42^\circ\ -92.07^\circ]$	$[0.231^\circ\ 0.050^\circ\ 0.187^\circ]$

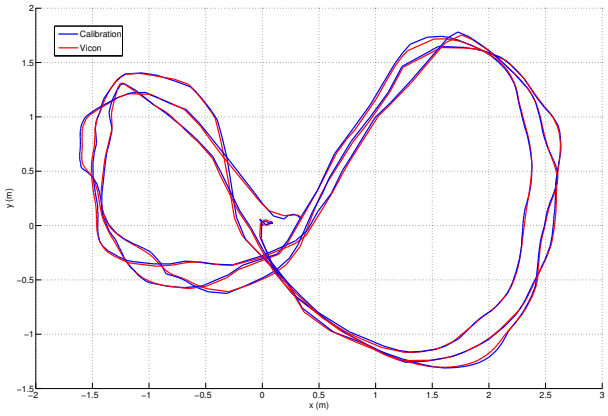


Fig. 5 We plot the robot's (x,y)-position estimated by the self-calibration against that measured by the Vicon motion capture system. The calibration-estimated positions are colored in blue while the Vicon-measured positions are colored in red.

and those estimated by the Vicon-based calibration method. We observe that our estimated inter-camera transforms are accurate with rotation and translation errors not exceeding 0.183° and 0.73 cm respectively.

We plot in Figure 5 the robot's (x,y)-position both estimated by our self-calibration method (colored in blue) and measured by the Vicon motion capture system (colored in red). Similarly, in Figure 6, we plot over time the robot's z-position both estimated by our self-calibration method (colored in blue) and measured by the Vicon motion capture system (colored in red). The average 3D position error is 1.28 cm, and this low error indicates that our self-calibration produces an accurate map, and in turn, accurate calibration parameters.

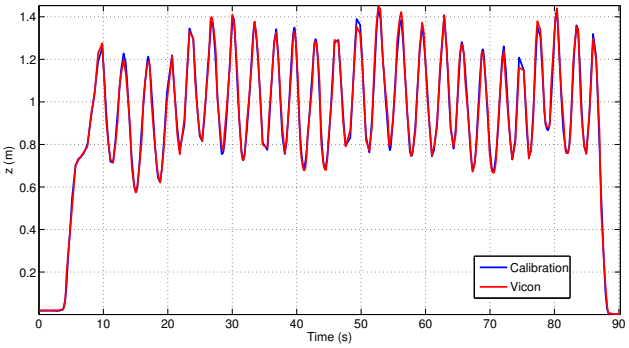


Fig. 6 We plot over time the robot's $z-$position estimated by the self-calibration against that measured by the Vicon motion capture system. The calibration-estimated positions are colored in blue while the Vicon-measured positions are colored in red.

Table 3 Comparison of our estimated inter-sensor transforms for the three-camera system configuration $\{C_1, C_2, C_3\}$ with ground-truth estimates from the Vicon-based calibration method. ${}^{i}\mathbf{R}_{j}$ is the rotation given in roll, pitch, and yaw angles between sensors i and j. ${}^{i}\mathbf{t}_{j}$ is the translation between sensors i and j.

	Our self-calibration	Difference with Vicon-based calibration
${}^{C_1}\mathbf{R}_{C_2}$	$[-0.06^\circ\ 0.20^\circ\ -1.39^\circ]$	$[0.001^\circ\ 0.035^\circ\ 0.016^\circ]$
${}^{C_1}\mathbf{t}_{C_2}$	[31.89 −0.10 0.04] cm	[0.15 0.11 0.11] cm
${}^{C_1}\mathbf{R}_{C_3}$	$[177.68^\circ\ 1.12^\circ\ 176.40^\circ]$	$[0.091^\circ\ 0.294^\circ\ 0.224^\circ]$
${}^{C_1}\mathbf{t}_{C_3}$	[0.69 2.97 −28.66] cm	[0.03 0.20 0.10] cm

6.1.2 Multi-Camera System With One Stereo Camera and One Monocular Camera

We then verify our self-calibration algorithm for two different multi-camera system configurations, each of which contains one stereo camera and one monocular camera. We carry out two runs in which we self-calibrate a subset of the four-camera system on the MAV. In the first run, we calibrate the multi-camera system containing cameras C_1, C_2, and C_3, while in the second run, we calibrate the multi-camera system containing cameras C_1, C_2, and C_4.

For the first run with camera C_4 excluded, we report in the first column of Table 3 the estimated rotation in terms of roll, pitch, and yaw angles, and translation of camera C_i with respect to camera C_1 for $i = 2,3$. The second column of Table 3 shows the differences between the estimated rotations/translations for the three-camera system configuration and those estimated by the Vicon-based calibration method. We observe that the rotation and translation differences are minimal as they do not exceed 0.294° and 0.20 cm respectively.

For the second run with camera C_3 excluded, we report in the first column of Table 4 the estimated rotation in terms of roll, pitch, and yaw angles, and translation of camera C_i

Table 4 Comparison of our estimated inter-sensor transforms for the three-camera system configuration $\{C_1, C_2, C_4\}$ with ground-truth estimates from the Vicon-based calibration method. ${}^{i}\mathbf{R}_j$ is the rotation given in roll, pitch, and yaw angles between sensors i and j. ${}^{i}\mathbf{t}_j$ is the translation between sensors i and j.

	Our self-calibration	Difference with Vicon-based calibration
${}^{C_1}\mathbf{R}_{C_2}$	$[-0.06^\circ\ 0.20^\circ\ -1.39^\circ]$	$[0.001^\circ\ 0.035^\circ\ 0.016^\circ]$
${}^{C_1}\mathbf{t}_{C_2}$	[31.89 −0.10 0.04] cm	[0.15 0.01 0.11] cm
${}^{C_1}\mathbf{R}_{C_4}$	$[176.51^\circ\ 0.30^\circ\ 177.46^\circ]$	$[0.135^\circ\ 0.027^\circ\ 0.259^\circ]$
${}^{C_1}\mathbf{t}_{C_4}$	[31.67 0.42 −28.49] cm	[0.26 0.98 0.07] cm

Table 5 Comparison of our estimated stereo transform for the stereo pair $\{C_3, C_4\}$ with ground-truth estimates from the chessboard-based stereo calibration method. ${}^{i}\mathbf{R}_j$ is the rotation given in roll, pitch, and yaw angles between sensors i and j. ${}^{i}\mathbf{t}_j$ is the translation between sensors i and j.

	${}^{C_3}\mathbf{R}_{C_4}$	${}^{C_3}\mathbf{t}_{C_4}$
Difference	$[0.04^\circ\ 0.264^\circ\ 0.046^\circ]$	[0.19 0.61 0.15] cm

with respect to camera C_1 for $i = 2,4$. The second column of Table 4 shows the differences between the estimated rotations/translations for the three-camera system configuration and ground-truth estimates from the Vicon-based calibration method. We again observe that the rotation and translation differences are minimal as they do not exceed 0.259° and 0.98 cm respectively.

As cameras C_3 and C_4 are a stereo pair, we compute the transform between these two cameras using the estimated transform ${}^{C_1}\mathbf{T}_{C_3}$ between cameras C_1 and C_3 from the first self-calibration run, and the estimated transform ${}^{C_1}\mathbf{T}_{C_4}$ between cameras C_1 and C_4 from the second self-calibration run. We then compare this estimated transform with that estimated by the chessboard-based stereo calibration, and tabulate the differences between the rotations/translations in Table 5. The rotation and translation differences do not exceed 0.264° and 0.61 cm respectively. These small differences show that our self-calibration method is able to accurately calibrate a multi-camera system that comprises both stereo and monocular cameras.

6.2 Reduced Self-Calibration

We evaluate a reduced version of our self-calibration method in which steps 2-5 from map merging to joint optimization as shown in Figure 2 are not carried out. This reduced self-calibration method is simply a hand-eye calibration which uses relative pose estimates from visual odometry with each camera entity to find the transform between each pair of camera entities. We wish to compare the accuracy of our self-calibration method with that of the reduced self-calibration method, and determine whether steps 2-4 significantly increase the accuracy of the calibration parameters at the cost of added complexity. We apply the reduced self-calibration method to each of the three multi-camera system configurations $\{C_1, C_2, C_3, C_4\}$, $\{C_1, C_2, C_3\}$, and $\{C_1, C_2, C_4\}$ that were used in the previous evaluation of the self-calibration method in Section 6.1. For each multi-camera system configuration, we tabulate the differences between the estimated rotations/translations and those estimated by the Vicon-based calibration method in Table 6.

For the case of the multi-camera system configuration with two stereo cameras $\{C_1, C_2\}$ and $\{C_3, C_4\}$, we observe that the maximum rotation and translation errors associated with the inter-camera transforms estimated by the reduced self-calibration method are approximately and respectively 200% and 40% higher than those associated with the self-calibration method. For the case of the multi-camera system configuration with one stereo camera and one monocular camera, we observe that the rotation and translation errors are much higher than those for the two-stereo-camera configuration.

With depth and metric scale from stereo, stereo visual odometry produces extremely accurate relative pose estimates. Hence, hand-eye calibration with stereo visual odometry poses as input will produce calibration parameters that are slightly less accurate than ours. Steps 2-5 of the self-calibration pipeline as shown in Figure 2 further optimize these calibration parameters by utilizing loop closures between stereo cameras as a strong prior to recover accurate inter-camera transforms during the joint optimization. In contrast, monocular visual odometry produces pose estimates whose scale drifts over time. As hand-eye calibration estimates a single scale value, hand-eye calibration with monocular visual odometry poses whose scale drifts over time will produce rather inaccurate calibration parameters. These calibration parameters require further optimization in the forms of steps 2-5 in the self-calibration pipeline. Step 2 locks the poses of the monocular cameras to those of the reference stereo camera. As a result, the initial unscaled poses from monocular visual odometry now have metric scale. With these scaled poses and loop closures between different camera entities as a strong prior, accurate inter-camera transforms can be recovered during the joint optimization.

6.3 Camera-IMU Self-Calibration

We use the implementation provided by Lynen et al. (2013) to self-calibrate the two-stereo-camera configuration $\{C_1, C_2, C_3, C_4\}$. To the best of our knowledge, Lynen et al. (2013) provides the only publicly available implementation of camera-IMU self-calibration which does not require a calibration target. They use an extended Kalman filter to carry out online state

Table 6 Differences between our estimated inter-sensor transforms from the reduced self-calibration method with ground-truth estimates from the Vicon-based calibration method. ${}^{i}\mathbf{R}_{j}$ is the rotation given in roll, pitch, and yaw angles between sensors i and j. ${}^{i}\mathbf{t}_{j}$ is the translation between sensors i and j.

	Configuration $\{C_1, C_2, C_3, C_4\}$	Configuration $\{C_1, C_2, C_3\}$	Configuration $\{C_1, C_2, C_4\}$
${}^{C_1}\mathbf{R}_{C_2}$	[0.001° 0.035° 0.016°]	[0.001° 0.035° 0.016°]	[0.001° 0.035° 0.016°]
${}^{C_1}\mathbf{t}_{C_2}$	[0.15 0.01 0.11] cm	[0.15 0.01 0.11] cm	[0.15 0.01 0.11] cm
${}^{C_1}\mathbf{R}_{C_3}$	[0.177° 0.570° 0.349°]	[0.003° 1.072° 0.378°]	-
${}^{C_1}\mathbf{t}_{C_3}$	[0.98 0.42 0.23] cm	[10.9 12.60 16.96] cm	-
${}^{C_1}\mathbf{R}_{C_4}$	[0.181° 0.598° 0.409°]	-	[0.134° 2.552° 0.565°]
${}^{C_1}\mathbf{t}_{C_4}$	[1.02 0.67 0.59] cm	-	[29.89 17.72 21.15] cm

Table 7 Comparison of the transform between stereo cameras S_1 and S_2 with the ground-truth estimate from the Vicon-based calibration method. ${}^{i}\mathbf{R}_{j}$ is the rotation given in roll, pitch, and yaw angles between sensors i and j. ${}^{i}\mathbf{t}_{j}$ is the translation between sensors i and j.

	${}^{S_1}\mathbf{R}_{S_2}$	${}^{S_1}\mathbf{t}_{S_2}$
Difference	[1.622° 0.277° 1.285°]	[5.87 3.00 3.10] cm

estimation on a MAV. The MAV's state includes the camera-IMU transform. As stereo VO provides pose estimates with metric scale, and to remove the effect of scale estimation on the calibration parameters, we fix the scale component of the state. Unlike our self-calibration method, an initial estimate of the state has to be provided beforehand to ensure convergence.

Using the same dataset used by our self-calibration method in Section 6.1.1, we use the online camera-IMU self-calibration implementation to compute the transform between each stereo camera and the IMU. In turn, we infer the transform between S_1 and S_2, and based on the ground-truth transform estimated by the Vicon-based calibration method, we compute the associated rotation and translation errors. We tabulate these errors in Table 7.

The rotational and translation errors associated with the transform between S_1 and S_2 inferred from the estimated stereo-camera-IMU transforms are considerably higher than those associated with the estimates from our self-calibration method. In Section 6.4, we evaluate the impact of the increased calibration errors on the vSLAM performance, and subsequently, discuss whether the impact is insignificant enough to warrant the use of a camera-IMU calibration method in place of our self-calibration method.

6.4 vSLAM

We conduct a combination of both simulation and real-world experiments. The simulation experiments are designed to provide a quantitative analysis of the accuracy of our 3-point algorithm for motion estimation. Two real-world experiments are carried out in different settings to verify the accuracy of the poses estimated by our vSLAM implementation. In both real-world experiments, the MAV flies autonomously by relying on pose estimates from vSLAM which are input to the state estimator, and we manually send velocity commands to the MAV via a remote control. The MAV is constantly on the move such that we rely heavily on our 3-point motion estimation algorithm rather than the pose estimation algorithm. In the first experiment, the MAV moves along multiple loops in an indoor environment, and we use the Vicon motion capture system to record the MAV poses. In the second experiment, the MAV flies in a horizontal loop in an outdoor environment, and we use the loop closure error metric to evaluate the pose accuracy. Furthermore, we evaluate the impact of calibration errors on the vSLAM implementation. We carry out a simulation experiment to evaluate the accuracy of our 3-point motion estimation algorithm against a range of calibration errors, and a real-world experiment to evaluate the pose accuracy of the vSLAM implementation with calibration parameters from the camera-IMU self-calibration method described in Section 6.3.

In all real-world experiments, the input to our vSLAM implementation consists of 754×480 images from the four-camera system together with inertial data, and our vSLAM implementation runs at 7-12 Hz with the inner window size and outer window size for the double-window optimization set to 15 and 50 respectively.

6.4.1 Simulation Experiments

Here, we run simulations to quantify the accuracy of our 3-point algorithm for motion estimation, and compare the accuracy against that of the 2-point algorithm based on the Ackermann motion model (Lee et al., 2013b) and that of the linear 17-point algorithm (Pless, 2003). In each simulation, we use the same multi-camera system setup on the MAV. For each trial, we generate a random relative motion (θ, ρ) where θ and ρ are the relative yaw and scale respectively of the Ackermann motion defined in Lee et al. (2013b). θ and

ρ are assigned random values in the ranges of $[0.05, 0.15]$ radians and $[0.25, 0.75]$ m respectively. The 3D scene points are randomly sampled from the range of $[-10, 10]$ m with respect to the world reference frame. Point correspondences are obtained by reprojecting the 3D scene points into the cameras, and we ensure that each 3D scene point is seen by at least one camera over two consecutive frames.

Figures 7(a) and 7(b) show the average rotation and translation errors over 1000 trials, and a range of pixel noise levels between 0 and 1 pixels and with a 0.1 pixel interval. The rotation error is defined as the norm of the Euler angles from $\mathbf{R}\tilde{\mathbf{R}}^T$, where $\mathbf{R}$ and $\tilde{\mathbf{R}}$ are the estimated and ground truth rotation matrices. Following Quan and Lan (1999), we define the translation error as $2||\mathbf{t} - \tilde{\mathbf{t}}||/(||\mathbf{t}|| + ||\tilde{\mathbf{t}}||)$, where $\mathbf{t}$ and $\tilde{\mathbf{t}}$ are the estimated and ground truth translations. We can see that the errors from the linear 17-point algorithm are significantly higher, while the 2-point and our 3-point algorithms similarly show low errors.

Figures 7(c) and 7(d) show the rotation and translation errors over 1000 trials, and a range of relative rotation measurement noise levels between 0 and 0.6 degrees and with an interval of 0.1 degrees. The pixel noise is kept fixed at 0.5 pixels. At each relative rotation measurement noise level, we collectively corrupt the relative roll, pitch, and yaw angles with noise. We can see that the error from the linear 17-point algorithm is significantly higher than our 3-point algorithm despite the fact that the relative rotation measurement from the gyroscope is corrupted with noise. It is to be noted that the gyroscope used on our MAV has a maximum error of 0.1 degrees over two consecutive frames.

We also look at how each algorithm performs when we allow relative motion along the z-axis, and here, the Ackermann motion constraint is violated. Figures 7(e) and 7(f) show the rotation and translation errors as we set the z-component of the relative translation from 0 to 0.6 m with a step of 0.1 m. The pixel noise is kept fixed at 0.5 pixels. We can see that the errors from the 2-point algorithm increase as the Ackermann constraint is increasingly violated. In contrast, the errors from our 3-point algorithm remain relatively constant, and at the same time, they are significantly lower than the errors from the linear 17-point algorithm.

6.4.2 Indoor Experiment

In this indoor experiment, the MAV flies 3 loops over a distance of 45.48 m, starting each loop at an increasing height. Figure 8 shows the state of the keyframe graph at the end of the flight. In this figure, the keyframes belonging to the inner and outer windows are marked with red and blue spheres respectively. Green lines between keyframe pairs indicate loop closures. Green points represent 3D scene points in the map while yellow points represent 3D scene points observed in the inner window. A blue line traces the keyframe positions as estimated by our vSLAM implementation. A red line traces the corresponding Vicon-measured keyframe positions.

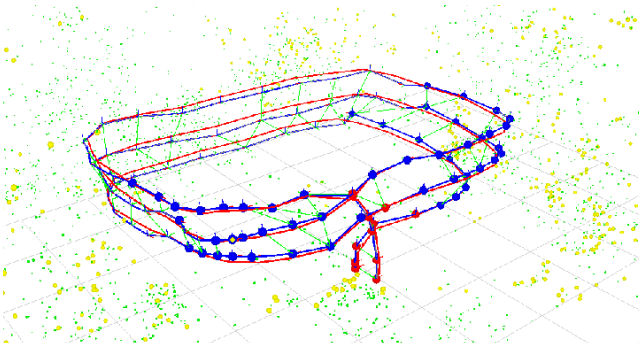


Fig. 8 We show the keyframe graph at the end of a 3-loop flight in an indoor environment. The red and blue spheres represent the keyframe poses belonging to the inner and outer windows respectively. A blue line traces the keyframe positions while a red line traces the corresponding Vicon-measured positions.

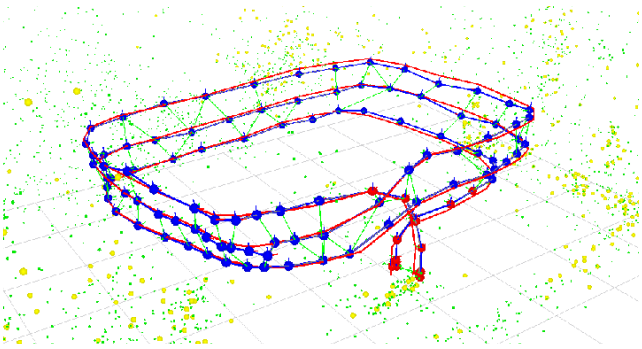


Fig. 9 We increase the size of the outer window to include all keyframes except those in the inner window. As a result, the keyframe position accuracy increases, but the running time of the double-window optimization now scales with the number of keyframes instead of being constant.

Based on Vicon measurements, the average error of the 110 keyframe positions is 6.34 cm. We observe that the errors associated with the keyframes not belonging to either the inner or outer windows, and especially at the boundary of the outer window, are higher as these keyframes are not included in the optimization.

We re-run the vSLAM implementation with real-time playback of the data logged from this experiment, and make one change to the double-window optimization such that the outer window includes all keyframes except those in the inner window. Figure 9 shows the state of the keyframe graph at the end of the flight. As expected, the average error of the keyframe positions decreases to 4.57 cm. However, with this outer window setting, the double-window optimization no longer runs in constant time, and thus, is not capable of real-time performance.

The results show that by switching from optimization over all keyframes to optimization with fixed-size inner and

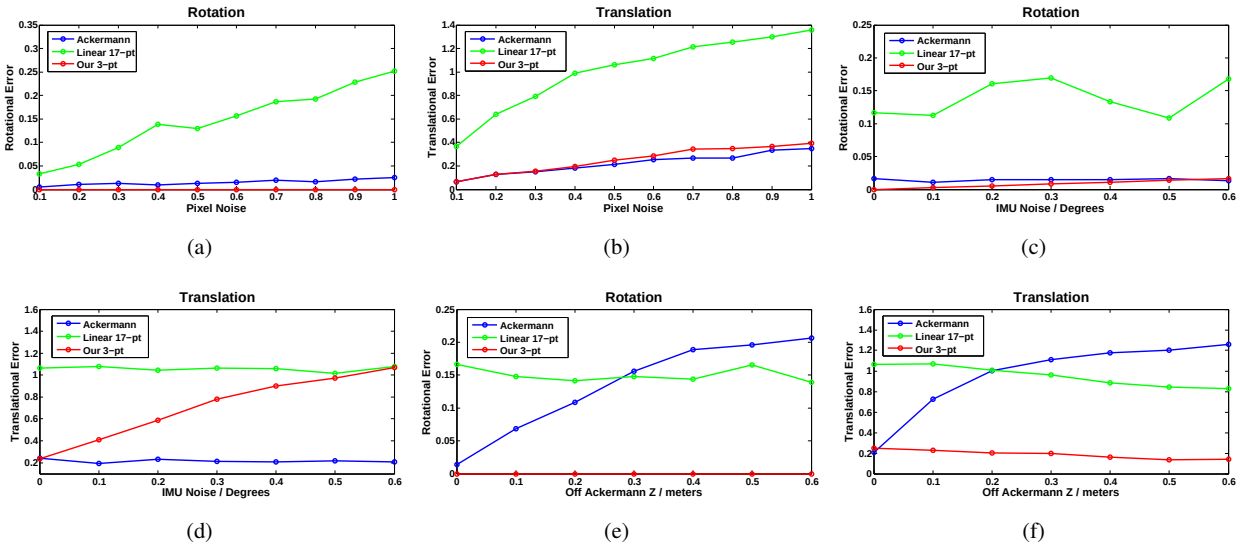


Fig. 7 Comparisons of rotation (rad) and translation (no units) errors from the Ackermann 2-pt, linear 17-point, and our 3-point algorithms in simulation over (a)-(b) image pixel noise, (c)-(d) relative angular measurement noise with pixel noise fixed at 0.5 pixels, and (e)-(f) off Ackermann motion along the z-axis with pixel noise fixed at 0.5 pixels.

outer windows, the accuracy of the keyframe positions slightly decreases. However, the keyframe poses are still reasonably accurate for tasks such as mapping and path planning.

6.4.3 Outdoor Experiment

In this outdoor experiment, we disable the loop closure detection as we want to evaluate the pose drift in the absence of loop closures. Here, the MAV flies one large loop on hilly terrain, and both the start and end points are the same. The flight time is 185 seconds, and the travelled distance is 112.98 m. At the end of the flight, the resulting keyframe graph has 820 keyframes. Figure 10 shows the MAV's path estimated by our vSLAM implementation. In this figure, a purple square and red circle mark the start and end points of the flight. The distance between the start and end point is 3.31 m, and thus, the loop closure error is 2.93%.

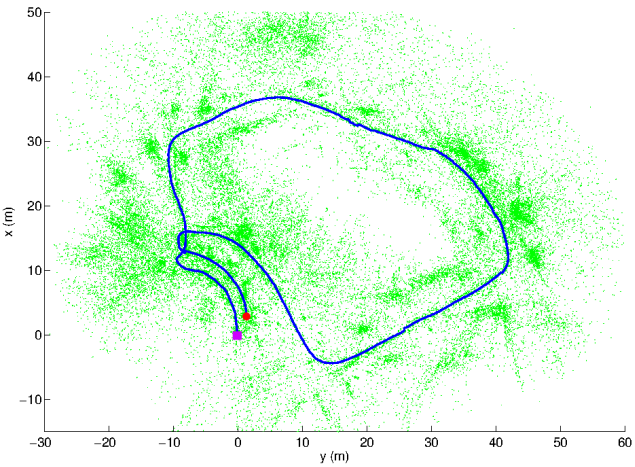


Fig. 10 In an outdoor experiment, we plot the (x,y) positions of the MAV estimated by our vSLAM implementation. A purple square and red circle mark the start and end points which are the same. Green dots represent 3D scene points.

6.4.4 Impact of Calibration Errors on vSLAM

Here, we evaluate the impact of calibration errors on our vSLAM implementation via simulation and real-world experiments. In a simulation experiment, we look at the effect of calibration errors on the pose accuracy of our 3-point motion estimation algorithm. In a real-world experiment, we look at the effect of calibration errors on the pose accuracy of the overall vSLAM implementation that includes both motion estimation and loop closures.

In the simulation experiment, we use the same two-stereo-camera system setup on the MAV and the ground-truth calibration generated by the Vicon-based calibration method. For each trial, we generate a random relative motion where the Euler angles are randomly sampled from a uniform distribution of $[0.05, 0.15]$ radians and each translation component is randomly sampled from a uniform distribution of $[0.25, 0.75]$ m. 100 3D scene points are randomly sampled from the range of $[-10, 10]$ m with respect to the world reference frame. Point correspondences are obtained by reprojecting the 3D scene points into the cameras, and we ensure that each 3D scene point is seen by at least one camera over two consecutive frames.

We use the same rotation and translation error metrics used in the earlier simulation experiment in Section 6.4.1 to quantify the effect of the calibration errors on the pose accu-

racy of our 3-point motion estimation algorithm. We compute a range of pitch errors in the rotation ${}^{S_1}\mathbf{R}_{S_2}$ between the two stereo cameras. This range is between 0 and 2 degrees with an interval of 0.1 degrees. Figures 11(a) and 11(b) show the average rotation and translation errors over 1000 trials, and the range of pitch errors. Similarly, we compute a range of y-translation errors in the translation ${}^{S_1}\mathbf{t}_{S_2}$ between the two stereo cameras. This range is between 0 and 10 cm with an interval of 0.5 cm. Figures 11(c) and 11(d) show the average rotation and translation errors over 1000 trials, and the range of y-translation errors. We observe from these figures that the pose accuracy of our 3-point motion estimation algorithm is significantly more sensitive to rotation errors in the calibration than translation errors in the calibration. Furthermore, we observe that a given rotation error in the calibration approximately corresponds to the same rotation error in the relative pose estimated by the 3-point motion estimation algorithm. In general, higher calibration errors lead to reduced accuracy of motion estimation. Hence, it is important that the calibration errors are kept to a minimum so that the motion estimation algorithm can compute relative poses with high accuracy; in turn, drift is minimized in the absence of loop closures.

Using the ground-truth calibration generated from the Vicon-based calibration method as the reference calibration in simulations, the estimated relative pose using the calibration generated by our self-calibration method has a rotation error of 0.00274 radians and a translation error of 0.0158 averaged over 1000 trials. The estimated relative pose using the calibration generated by the camera-IMU self-calibration method described in Section 6.3 has a rotation error of 0.035 radians and a translation error of 0.115 averaged over 1000 trials. It is apparent that the more accurate calibration associated with our self-calibration method corresponds to more precise motion estimation.

Calibration errors affect not only motion estimation but also loop closures. Calibration errors lead to a globally inconsistent map, and thus, leads to the inaccurate estimation of loop closure transforms. In turn, the overall accuracy of vSLAM is reduced. We look the impact of calibration errors on the overall vSLAM performance in real-world settings. We repeat the indoor experiment described in Section 6.4.2 and with constant-time double window optimization, and re-use the data logged from that experiment. We use the calibration parameters estimated by the camera-IMU self-calibration method. Based on Vicon measurements, the average error of the keyframe positions is 11.83 cm which is significantly higher than the average error of 6.34 cm associated with our self-calibration method.

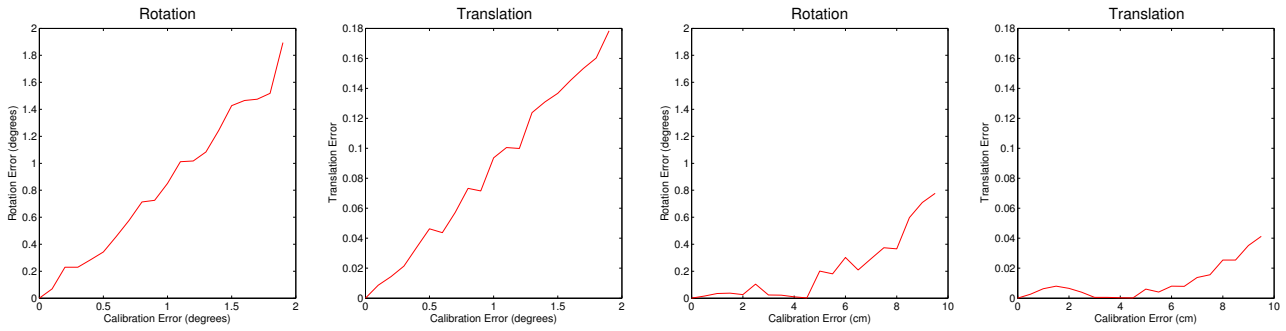
7 Conclusions

Our vSLAM-based self-calibration method produces accurate extrinsic calibration parameters with metric scale for a MAV with at least one stereo camera and any number of monocular cameras. Overlapping fields of view are not assumed to exist between any two camera entities. In addition, we estimate the rotation between the reference frames of the gyroscope and multi-camera system. This rotation facilitates the use of gyroscopic measurements in our motion estimation algorithm. We use real-world experiments to validate our self-calibration method.

We quantify the accuracy of our 3-point motion estimation algorithm through simulation experiments in which we vary factors such as gyroscopic noise and pixel noise. Through real-world experiments in both indoor and outdoor experiments, we demonstrate real-time on-board vSLAM with loop closures on a MAV with two calibrated stereo cameras. There are no overlapping fields of view between these two stereo cameras. The MAV is able to perform autonomous flight based on the pose estimates from vSLAM. Furthermore, via both simulation and real-world experiments, we show how an inaccurate calibration causes our vSLAM implementation to produce degraded estimates of the MAV's pose. This demonstration supports the fact that our self-calibration method outputs calibration parameters with the necessary level of precision that is required for vSLAM with a multi-camera system to estimate accurate poses.

There are several common factors that affect the accuracy of our self-calibration method and vSLAM implementation. We discuss these factors:

1. Number of features in the environment: few features limit the accuracy of visual odometry and estimated loop closure transforms, and in turn, both the self-calibration and vSLAM.
2. Number of temporal feature correspondences between camera entities in the form of loop closures: the more these correspondences, the more accurate the self-calibration and the more globally consistent the map built by vSLAM. In the case of self-calibration, temporal feature correspondences between camera entities act as a prior for recovering accurate transforms between these camera entities.
3. Distance of the scene points from the cameras: the nearer the scene points to the cameras, the lower the uncertainty of the estimated 3D coordinates, and the more accurate the self-calibration and vSLAM.
4. Image resolution: The higher the image resolution, the smaller the angle subtended by a pixel in a camera image, and the more accurate the self-calibration and vSLAM. The 3D coordinates of scene points can be estimated more precisely with a higher image resolution.



(a) (b) (c) (d)

Fig. 11 Comparisons of rotation (degree) and translation (no units) errors from our 3-point motion estimation algorithm in simulation over (a)-(b) error in ${}^{S_1}\mathbf{R}_{S_2}$, and (c)-(d) error in ${}^{S_1}\mathbf{t}_{S_2}$.

5. Type of lens used: A wider-field-of-view lens such as a fisheye lens increases the angle subtended by a pixel in a camera image. As a result, scene points are triangulated with higher uncertainty, and in turn, the self-calibration and vSLAM are less accurate.

Future work will focus on improving the robustness of vSLAM, and adding capabilities such as dense mapping and 3D exploration to our MAV platform. We plan to improve the robustness of vSLAM by switching from motion estimation to pose estimation in the event that there are no inter-camera feature correspondences for the case of a MAV with one stereo camera and one monocular camera. If the stereo camera's field of view is blocked by an object, there will be no inter-camera feature correspondences. As a result, our motion estimation method will only estimate the relative motion up to scale if the relative rotation is close to identity. However, the monocular camera may continue to track many features on the current map, meaning that 2D-3D feature correspondences are still available for the monocular camera. To circumvent a possible failure of the motion estimation method in the absence of inter-camera correspondences, we can use our pose estimation method to continue localizing the MAV.

Acknowledgements The first author is funded by the DSO National Laboratories Postgraduate Scholarship. This work is partially supported by the SNSF V-MAV grant (DACH framework).

References

Agarwal, S., Mierle, K., and Others (2013). Ceres solver. `https://code.google.com/p/ceres-solver/`.

Agrawal, M., Konolige, K., and Blas, M. (2008). Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision ECCV 2008*, volume 5305 of *Lecture Notes in Computer Science*, pages 102–115. Springer Berlin Heidelberg.

Brookshire, J. and Teller, S. (2012). Extrinsic calibration from per-sensor egomotion. In *Robotics: Science and Systems (RSS)*.

Carrera, G., Angeli, A., and Davidson, A. (2011a). Lightweight slam and navigation with a multi-camera rig. In *European Conference on Mobile Robots (ECMR)*.

Carrera, G., Angeli, A., and Davison, A. (2011b). Slam-based automatic extrinsic calibration of a multi-camera rig. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Daniilidis, K. (1999). Hand-eye calibration using dual quaternions. *International Journal of Robotics Research (IJRR)*, 18(3):286–298.

Furgale, P., Rehder, J., and Siegwart, R. (2013a). Unified temporal and spatial calibration for multi-sensor systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Furgale, P., Schwesinger, U., Rufli, M., Derendarz, W., Grimmett, H., Muhlfellner, P., Wonneberger, S., Timpner, J., Rottmann, S., Li, B., Schmidt, B., Nguyen, T., Cardarelli, E., Cattani, S., Bruning, S., Horstmann, S., Stellmacher, M., Mielenz, H., Koser, K., Beermann, M., Hane, C., Heng, L., Lee, G. H., Fraundorfer, F., Iser, R., Triebel, R., Posner, I., Newman, P., Wolf, L., Pollefeys, M., Brosig, S., Effertz, J., Pradalier, C., and Siegwart, R. (2013b). Toward automated driving in cities using close-to-market sensors: An overview of the v-charge project. In *IEEE Intelligent Vehicles Symposium (IV)*.

Grossberg, M. and Nayar, S. (2001). A general imaging model and a method for finding its parameters. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 108–115 vol.2.

Harmat, A., Sharf, I., and Trentini, M. (2012). Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle. In *Intelligent Robotics and Applications*, volume 7506 of *Lecture Notes in Computer Science*, pages 421–432.

Heng, L., Bürki, M., Lee, G. H., Furgale, P., Siegwart, R., and Pollefeys, M. (2014a). Infrastructure-based calibration of a multi-camera rig. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Heng, L., Lee, G. H., and Pollefeys, M. (2014b). Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle. In *Robotics: Science and Systems (RSS)*.

Heng, L., Li, B., and Pollefeys, M. (2013). Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Heng, L., Meier, L., Tanskanen, P., Fraundorfer, F., and Pollefeys, M. (2011). Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Kaess, M. and Dellaert, F. (2006). Visual slam with a multi-camera rig. Technical report.

Kelly, J. and Sukhatme, G. (2011). Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research (IJRR)*, 30(1):56–79.

Kim, J., Hwangbo, M., and Kanade, T. (2008). Motion estimation using multiple non-overlapping cameras for small unmanned aerial vehicles. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2013a). Minimal solutions for pose estimation of a multi-camera system. In *International Symposium on Robotics Research (ISRR)*.

Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2013b). Motion estimation for self-driving cars with a generalized camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2013c). Robust pose-graph loop-closures with expectation-maximization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Lee, G. H., Pollefeys, M., and Fraundorfer, F. (2014). Relative pose estimation for a multi-camera system with known vertical direction. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, H., Hartley, R., and Kim, J. (2008). A linear approach to motion estimation using generalized camera models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lim, H., Sinha, S., Cohen, M., and Uyttendaele, M. (2012). Real-time image-based 6-dof localization in large-scale environments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1043–1050.

Lynen, S., Achtelik, M., Weiss, S., Chli, M., and Siegwart, R. (2013). A robust and modular multi-sensor fusion approach applied to mav navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3923–3929.

Mei, C. and Rives, P. (2007). Single view point omnidirectional camera calibration from planar grids. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Nister, D. (2004). An efficient solution to the five-point relative pose problem. 26(6):756–770.

Pless, R. (2003). Using many cameras as one. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Prentice, S. and Roy, N. (2009). The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research (IJRR)*, 28(11-12):1448–1465.

Quan, L. and Lan, Z. D. (1999). Linear n-point camera pose determination. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 21, pages 774–780.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: an efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*.

Sattler, T., Liebe, B., and Kobbelt, L. (2011). Fast image-based localization using direct 2d-to-3d matching. In *IEEE International Conference on Computer Vision (ICCV)*, pages 667–674.

Schauwecker, K. and Zell, A. (2014). On-board dual-stereo-vision for the navigation of an autonomous mav. *Journal of Intelligent and Robotic Systems*, 74(1-2):1–16.

Schmid, K., Tomic, T., Ruess, F., Hirschmuller, H., and Suppa, M. (2013). Stereo vision based indoor/outdoor navigation for flying robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Schmidt, J., Vogt, F., and Niemann, H. (2005). Calibration-free hand-eye calibration: A structure-from-motion approach. In *Pattern Recognition*, volume 3663 of *Lecture Notes in Computer Science*, pages 67–74.

Schweighofer, G., Segvic, S., and Pinz, A. (2008). Online/realtime structure and motion for general camera models. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 1–6.

Shen, S., Mulgaonkar, Y., Michael, N., and Kumar, V. (2013). Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In *Robotics: Science and Systems (RSS)*.

Stewénius, H., Nistér, D., Oskarsson, M., and Åström, K. (2005). Solutions to minimal generalized relative pose problems. In *OMNIVIS*.

Strasdat, H., Davison, A., Montiel, J., and Konolige, K. (2011). Double window optimisation for constant time visual slam. In *IEEE International Conference on Computer Vision (ICCV)*.

Tribou, M. (2014). *Relative Pose Estimation Using Non-overlapping Multicamera Clusters*. PhD thesis, University of Waterloo.

Weiss, S., Achtelik, M. W., Lynen, S., Achtelik, M. C., Kneip, L., Chli, M., and Siegwart, R. (2013). Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics (JFR)*, 30(5):803–831.