Robotics and Autonomous Systems 131 (2020) 103577

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

Visual-inertial teach and repeat

Matías Nitsche [a,*], Facundo Pessacg [a], Javier Civera [b]

[a] Instituto de Investigación en Ciencias de la Computación (ICC-UBA-CONICET). Buenos Aires, Argentina
[b] Instituto de Investigación en Ingeniería de Aragón (I3A). Universidad de Zaragoza, Spain

ARTICLE INFO

Article history:
Available online 13 June 2020

Keywords:
UAV
Embedded
Navigation
Visual-inertial
Stereo
Teach-and-replay
Relative

ABSTRACT

Teach and Repeat (T&R) refers to the technology that allows a robot to autonomously follow a previously traversed route, in a natural scene and using only its onboard sensors. In this paper we present a Visual-Inertial Teach and Repeat (VI-T&R) algorithm that uses stereo and inertial data and targets Unmanned Aerial Vehicles with limited on-board computational resources. We propose a tightly-coupled relative formulation of the visual-inertial constraints that is tailored to the T&R application. In order to achieve real-time operation on limited hardware, we reduce the problem to motion-only visual-inertial Bundle Adjustment. In the repeat stage, we detail how to generate a trajectory and smoothly follow it with a constantly changing relative frame. The proposed method is validated in simulated environments, using real sensor data from the public EuRoC dataset, and using our own robotic setup and closed-loop control. Our experimental results demonstrate high accuracy and real-time performance both on a standard desktop system and on a low-cost Odroid X-U4 embedded computer.

1. Introduction

Mimicking a certain trajectory which has been previously followed by a mobile sensor platform is a desirable robotic capability with clear applications; such as structure inspection, environment monitoring or sample-return missions in planetary exploration. This problem is commonly referred to as *teach and repeat* (T&R) navigation.

Localization should be estimated from the robot's internal sensors in most of these potential application scenarios, as reliable external positioning (e.g., with GPS) might not always be possible. Simultaneous Localization and Mapping (SLAM) is a widely adopted technology for GPS-denied navigation [1], allowing to estimate the robot pose for use in the control loop. However, SLAM typically uses an *absolute* formulation, where all robot poses and landmark positions are referred to a single privileged reference frame (generally the initial robot pose). This choice imposes global map consistency, which cannot be generally guaranteed due to the inevitable drift of the pose estimation. Thus, most solutions expect frequent loop-detection followed by global pose-graph relaxation, which becomes very costly for large-scale scenarios and is in general not suitable for long-distance navigation.

If the expectation of optimal path-planning is abandoned, global map consistency is actually not required [2]. In other words, only local consistency is really necessary. In fact, it is possible to employ a *relative* formulation of the problem, where poses are expressed as a transformation relative to other nearby pose and landmarks are referred to the initial observing coordinate frame. As a result, constant-time loop-closing becomes possible, obtaining a more efficient approach [3].

The computational cost of the navigation solution is of particular importance for payload-limited hardware platforms, such as small aerial robots. For this reason, relative localization approaches become an attractive solution. Regarding the sensor, the visual-inertial combination is a very convenient choice for agile and small robots, due to its low power demand, cost, size and weight. Inertial data can capture brisk motions at high rate, while visual data refer to external features and hence reduce drift.

Motivated by these two aspects, in this work we present an efficient visual-inertial teach and repeat (VI-T&R) method from stereo and inertial data. Our main contribution is a relative tightly-coupled keyframe-based formulation of the problem, the first one in the literature to our knowledge. We also propose several adaptations for resource-constrained hardware and demonstrate that our approach can be run in small aerial vehicles with limited computational resources.

This paper builds upon our previous work [4], generalizing the approach on such paper. Our estimated state now includes a sliding window of the most recent robot poses. Furthermore, in this work we estimate both the gyroscope and accelerometer biases, whereas [4] only considered the former one. Finally, we properly marginalize poses falling outside of the window instead

* Corresponding author.
E-mail address: mnitsche@dc.uba.ar (M. Nitsche).

https://doi.org/10.1016/j.robot.2020.103577
0921-8890/

of simply discarding past measurements. We also report an extensive evaluation of our proposal including experimental results in simulation environments, in the EuRoC dataset, and in our own robotic platform with closed-loop control.

2. Related work

The literature on T&R navigation has mostly focused on terrestrial robots equipped with either laser [5–7] or visual sensors [8–13]. Works that address other types of locomotion such as underwater [14] or aerial vehicles, as in our case, are scarcer.

An early work for the case of aerial vehicles is [15], where a proof-of-concept T&R navigation system is presented. In this work a downward-looking camera is used for pose tracking, based on a planar floor assumption. In following work [16], the T&R method is adapted for the case of fixed-wing aerial platforms. However, the computation is done offline and requires a Graphics Processing Unit (GPU). Moreover, closed-loop control and trajectory planning are left as future work by the authors. More recently, a fully working visual-only T&R method is proposed for solving the emergency return of an aerial vehicle in [17]. The proposed method is able to track the robot motion at high speeds using a stereo camera mounted on an active gimbal. As before, to achieve real-time operation, a GPU is employed to extract and match image features.

A VI-T&R approach for aerial robots has been recently proposed in [18], involving tightly-coupled and loosely-coupled estimators running together. Real-time performance is demonstrated, but using a powerful Intel i7 processor. The authors of this work state that, in order to navigate autonomously, loop closing and global Bundle Adjustment are needed between the teach and repeat phases. Gao et al. [19] also present a VI-T&R system, focusing on optimal trajectory generation for autonomous path-following in small unmanned aerial vehicles. However, part of the computation is delegated to an off-board laptop computer. And also, they aim to achieve global map consistency based on loop closure, which may not be possible during long-distance operation and exploratory trajectories.

To address the problem of expensive global Bundle Adjustment in SLAM, Sibley [20] proposed a Relative Bundle Adjustment (RBA) framework where a relative formulation was used. Under this approach, Bundle Adjustment operates over a graph of relative poses with landmarks specified in relation to these poses, thus defining a *manifold*. The main benefit of this approach is that a constant-time solution of the full maximum-likelihood estimate can be obtained incrementally. Furthermore, loop closure also becomes constant-time, which is a considerable benefit in contrast to methods based on an absolute formulation of the problem. This framework was then used as the basis of a visual SLAM method known as R-SLAM [3] targeting large-scale mapping.

The relative formulation of the localization problem has been adopted at various degrees in several other works in the SLAM literature. In [21], while following the usual approach of referencing poses w.r.t the initial coordinate frame, the benefit of expressing landmark positions relative to the observation frame is highlighted. Moreover, authors argue in favor of not fixing the initial pose of state estimation window (typically done to remove the gauge freedom of the solution) for the purpose of avoiding an unbounded growth of the uncertainty of the states, which introduces linearization errors. Sola et al. [22] show the benefits of the so-called *landmark anchoring*, consisting on referring landmarks to the local reference frame at initialization. Similarly, De Palézieux et al. [23] employ the concept of *anchor nodes* for the same purpose, referencing all information to the initial pose of the estimation window. Robocentric estimation approaches (e.g., [24]) refer all landmarks and poses to the current robot frame. In [25] visual-inertial estimation over a purely-relative map is presented. The benefit of a relative formulation is also identified by Eckenhoff et al. [26], where they propose to switch to a relative formulation of the problem before marginalization for the purpose of improving the estimator consistency. This same idea is followed in ICE-BA [27].

However, while all these works follow a relative formulation, the state optimization is solved by first transforming from a relative representation to an absolute equivalent one within a local frame. The motivation for such strategy is argued to be performance or simplicity, as a fully relative approach is considered too expensive. Also, under relative schemes state uncertainties are not usually considered. In terms of improving performance, Moreno et al. [28] propose a generalization of RBA which allows to express the problem with different degree of locality, thus representing both fully absolute and relative cases in the extremes.

In contrast to the aforementioned works in the literature, we fully embrace the relative formulation of the problem and do not resort to absolute representations in intermediate steps. This allows us to handle state uncertainty within local coordinate frames in a straightforward manner. Moreover, we present a tightly-coupled approach for stereo visual-inertial estimation in relative terms and demonstrate a performance that is sufficient for closed-loop control, even on a low-power single-board computer. From this algorithm we are able to build a robust VI-T&R method for unmanned aerial vehicle navigation, running fully on-board.

3. Method overview

The proposed T&R framework is based on a relative Visual-Inertial Odometry (VIO). In the context of the T&R problem, during the *teach* phase, VIO is employed to build a map of recent robot poses and three-dimensional landmarks. This map takes the form of a graph, with nodes corresponding to keyframes where landmarks are initialized and edges representing the relative transforms. During the *repeat* phase, this same VIO system runs in parallel to a map-localization task, which relates the current VIO map with the prior map built during the teach phase. A smooth trajectory is then built from the nearby portion of the prior map which is suitable for closed-loop path-following.

In the following sections we first define the VIO problem and its solution by means of a maximum a posteriori (MAP) estimator. Then, we describe the map-localization scheme employed during the repeat phase of this framework. Finally, we present the algorithm for trajectory generation and following.

4. Relative visual-inertial odometry

Under a relative formulation, the VIO state to be estimated includes the current robot pose as well as the inertial states, such as linear velocity, gravity and biases. In contrast to an absolute formulation of the problem, the current robot pose is not expressed by means of a transformation between a unique coordinate system (the origin of the map, set during system initialization) and the robot coordinate system, but as a *small* transformation to a nearby keyframe. Similarly, all other variables are expressed in their local coordinate system.

4.1. State definition

The state to be estimated is defined as a sliding-window involving n recent robot poses and associated variables:

$$\mathbf{x} = \left\{ \left\{ \mathbf{T}_i^{i+1} \right\}_{i=1\dots n-1}, \left\{ \mathbf{v}_i, \hat{\mathbf{g}}_i, \mathbf{b}_i^g, \mathbf{b}_i^a \right\}_{i=1\dots n} \right\} \tag{1}$$

where

$$\mathbf{T}_i^{i+1} = \begin{bmatrix} \mathbf{R}_i^{i+1} & \mathbf{t}_i^{i+1} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathrm{SE}(3)$$

stands in general for the relative motion between two consecutive keyframes denoted as i and $i+1$. In particular, n corresponds to the most recent robot frame (not yet marked as keyframe). Thus, $\mathbf{T}_{n-1}^{n}$ denotes the motion between the last keyframe and the current frame. $\mathbf{R}_i^{i+1} \in \mathrm{SO}(3)$ represents the relative rotation, and $\mathbf{t}_i^{i+1} \in \mathbb{R}^3$ the translation between the origins of the respective local camera frames. $\mathbf{v}_i \in \mathbb{R}^3$ is the ith camera velocity in its local frame, $\hat{\mathbf{g}}_i \in \mathrm{S}^2$ is the normalized gravity vector of the ith keyframe expressed in local frame coordinates, and $\mathbf{b}_i^g, \mathbf{b}_i^a \in \mathbb{R}^3$ are, respectively, the gyroscope and accelerometer biases.

Note that, differently from other visual-inertial estimators, we include the gravity direction as part of the state. Under the traditional absolute formulations for visual-inertial state estimation, this variable is removed from the state by aligning the global reference frame to the gravity direction during system initialization. The gravity vector in such global frame is reduced to the canonical vector $\mathbf{g} = \begin{bmatrix} 0 & 0 & -G \end{bmatrix}^\top$. But, as a consequence, the rest of the variables in the state are tied to a gravity-aligned absolute frame. In turn, by including the gravity vector in the local camera frames we remove this dependence and all states are relative. As an additional benefit, it becomes possible to explicitly re-estimate the gravity direction during normal system operation and thus avoid coupling gravity and absolute orientation errors. In order to improve the observability of the state (in particular of the accelerometer bias), we assume a known gravity magnitude (G) and only optimize the gravity direction.

Notice that our state does not have gauge-freedom problems, which for the absolute visual-inertial case corresponds to the four degrees of freedom of the absolute 3D position and yaw angle.

Finally, observe that our state does not include the landmark coordinates, as we triangulate them from the stereo pairs and do not optimize their position. In other words, we perform motion-only bundle adjustment. The reasons for this choice are to reduce the dimensionality of the state and hence the computational footprint of our VIO, and to avoid concurrent access to the state by the tracking and local mapping threads. As a result, we are able to run the VIO estimator in a single thread.

4.2. Maximum a posteriori estimation

The MAP estimation of our relative visual-inertial state $\mathbf{x}$, given a measurement vector $\mathbf{z}$ composed of q independent observations $\mathbf{z} = (\mathbf{z}_1^\top, \ldots, \mathbf{z}_t^\top, \ldots, \mathbf{z}_q^\top)^\top$ and some prior information over the state $p(\mathbf{x})$, is the vector $\hat{\mathbf{x}}$ that maximizes the probability function $p(\mathbf{x}|\mathbf{z})$. By applying Bayes' rule and assuming independent observations, $p(\mathbf{x}|\mathbf{z})$ can be written as:

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})} = \prod_{t=1}^{q} \frac{p(\mathbf{z}_t|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z}_t)}$$

The MAP estimate $\hat{\mathbf{x}}$ can be formulated as:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \prod_{t=1}^{q} p(\mathbf{z}_t|\mathbf{x})p(\mathbf{x})$$

where we omit the denominator $p(\mathbf{z}) = \prod_{t=1}^{q} p(\mathbf{z}_t)$ as it does not depend on $\mathbf{x}$. We can rewrite the previous expression by applying $\ln(\cdot)$ as:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ln\left(\prod_{t=1}^{q} p(\mathbf{z}_t|\mathbf{x})p(\mathbf{x})\right) = \tag{2}$$

$$= \arg\min_{\mathbf{x}} \sum_{t=1}^{q} \ln(p(\mathbf{z}_t|\mathbf{x})) + \ln(p(\mathbf{x})) \tag{3}$$

Assuming Gaussian-distributed variables, the MAP state estimation is:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{t=1}^{q} \left\| h_t(\mathbf{x}) \boxminus \hat{\mathbf{z}}_t \right\|_{\boldsymbol{\Omega}_t}^2 + \left\| \mathbf{x} \boxminus \check{\mathbf{x}} \right\|_{\boldsymbol{\Omega}_{\mathbf{x}}}^2$$

where $\mathbf{z}_t = h_t(\mathbf{x}) \boxplus \mathbf{w}$ represents the measurement model, with $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}_t^{-1})$, and $\mathbf{x} \sim \mathcal{N}(\check{\mathbf{x}}, \boldsymbol{\Omega}_{\mathbf{x}}^{-1})$ represents prior information. Note that we use the generic operators $\boxplus$ and $\boxminus$, which allow us to operate with variables on a manifold. Finally, we can write the previous expression in a more compact form as:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{u=1}^{r} \|\mathbf{e}_u(\mathbf{x})\|_{\boldsymbol{\Omega}_u}^2 = \arg\min_{\mathbf{x}} c(\mathbf{x}) \tag{4}$$

where $\mathbf{e}_u(\mathbf{x})$ is the error corresponding to the observation $\mathbf{z}_u$ and $c(\cdot)$ is the total cost function, composed of the sum of the individual squared residuals.

In particular, for our visual-inertial state estimation, the cost function is composed of three terms

$$\begin{aligned} c(\mathbf{x}) = &\sum_{i=1}^{n} \sum_{k \in K_i} \left\| \mathbf{e}_{i,k}^V \right\|_{(\mathbf{Q}_{i,k}^V)^{-1}}^2 + \\ &\sum_{i=1}^{n} \left\| \mathbf{e}_i^I \right\|_{(\mathbf{Q}_i^I)^{-1}}^2 + \left\| \mathbf{e}^P \right\|_{(\mathbf{Q}^P)^{-1}}^2 \end{aligned} \tag{5}$$

$\mathbf{e}_{i,k}^V$ is the visual residual that corresponds to the observation of landmark $\boldsymbol{\ell}_k$ from the ith keyframe (where n is the number of keyframes and K_i the set of landmark indices observed from keyframe i), $\mathbf{e}_i^I$ is the inertial residual corresponding to the motion from keyframe i to keyframe $i+1$, and $\mathbf{e}^P$ the residual related to prior information coming from the marginalization of previous states. $\mathbf{Q}_{i,k}^V$, $\mathbf{Q}_i^I$, $\mathbf{Q}^P$ are the covariance matrices of the noise associated to each observation type.

4.2.1. Visual residual

The visual residual is the stereo reprojection error, i.e., the geometric distance between a landmark projection and its corresponding image feature, on both left and right images.

In our relative formulation, we identify the kth landmark of the map as ${}^j\boldsymbol{\ell}_k \in \mathbb{R}^3$, where its coordinates are expressed in frame j. When initializing a new landmark, frame j will correspond to the frame of the left camera of the stereo pair used to triangulate its position. On the other hand, the left–right image feature pair detected in image frame i which matches the image projection of landmark ${}^j\boldsymbol{\ell}_k$ in i is denoted as $\mathbf{y}_{i,k} \in \mathbb{R}^4$. The visual residual is then defined as:

$$\mathbf{e}_{i,k}^V = \pi\left(g\left(\mathbf{T}_i^j, {}^j\boldsymbol{\ell}_k\right)\right) - \mathbf{y}_{i,k} \tag{6}$$

where g transforms ${}^j\boldsymbol{\ell}_k$ to the reference frame of the left camera of the ith frame:

$$g\left(\mathbf{T}_i^j, {}^j\boldsymbol{\ell}_k\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{T}_i^j \begin{bmatrix} {}^j\boldsymbol{\ell}_k \\ 1 \end{bmatrix} = {}^i\boldsymbol{\ell}_k$$

and $\pi : \mathbb{R}^3 \to \mathbb{R}^4$ is the stereo-projection function:

$$\pi({}^i\boldsymbol{\ell}_k) = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & {}^l c_v & 0 \\ f_u & 0 & c_u & -f_u b \\ 0 & f_v & c_v & 0 \end{bmatrix} \frac{1}{z} \begin{bmatrix} {}^i\boldsymbol{\ell}_k \\ 1 \end{bmatrix}$$

where ${}^i\boldsymbol{\ell}_k = [x, y, z, 1]^\top$, b is the stereo baseline, and f_u, f_v, c_u, c_v are the intrinsic parameters of rectified stereo camera pair. Note that since the image rectification process is not perfect, vertical coordinates of the image feature $\mathbf{y}_{i,k}$ in each camera may differ slightly and thus we include both in (6).

Finally, to obtain $\mathbf{T}_i^j$, we compose the chain of relative transforms from keyframe i to j as

$$\mathbf{T}_i^j = \mathbf{T}_i^{i+1}\mathbf{T}_{i+1}^{i+2}\dots\mathbf{T}_{j-1}^{j} \tag{7}$$

This chain might involve transforms which are not actively estimated anymore (i.e. they have been already marginalized). For these, we use their last estimated value before marginalization.

The covariance matrix associated to the visual residual is defined as:

$$\mathbf{Q}_{i,k}^V = {}^i\mathbf{Y}_k + \mathbf{J}_\pi \mathbf{R}_i^j \boldsymbol{\Lambda}_k \mathbf{R}_j^{i\,\top} \mathbf{J}_\pi^\top \tag{8}$$

where ${}^i\mathbf{Y}_k = \sigma_f^2 \mathbf{I}_{4\times4}$ relates to the noise of the stereo observation $\mathbf{y}_{i,k}$, $\mathbf{J}_\pi$ is the Jacobian of the projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^4$, $\mathbf{R}_i^j$ is the accumulated rotation from i to j and ${}^j\boldsymbol{\Lambda}_k \in \mathbb{R}^{3\times3}$ is the covariance matrix of landmark ${}^j\boldsymbol{\ell}_k$ (see Section 4.4.1). Notice that, $\mathbf{Y}_{i,k}$ models the noise relative to feature extraction and matching in the current frame i, while $\mathbf{J}_\pi \mathbf{R}_i^j \boldsymbol{\Lambda}_k \mathbf{R}_j^{i\,\top} \mathbf{J}_\pi^\top$ refers to the projection of the triangulation uncertainty $\boldsymbol{\Lambda}_k$ from its original coordinate frame j to the observation frame i. Since landmark positions are not re-estimated in the proposed approach, it is beneficial to consider their uncertainty during minimization [10,29], which will be ultimately reflected in the uncertainty of the estimated states.

4.2.2. Inertial residual

Our inertial residual between keyframes i and $i+1$, based on the preintegration described on Forster et al. [30], is

$$\mathbf{e}_i^I = \begin{bmatrix} \mathbf{e}_{\boldsymbol{\Delta}\mathbf{R}_i} & \mathbf{e}_{\boldsymbol{\Delta}\mathbf{v}_i} & \mathbf{e}_{\boldsymbol{\Delta}\mathbf{p}_i} & \mathbf{e}_{\mathbf{b}_i^g} & \mathbf{e}_{\mathbf{b}_i^a} & \mathbf{e}_{\hat{\mathbf{g}}_i} \end{bmatrix}^\top$$

where each of the individual errors are as follows:

$$\mathbf{e}_{\boldsymbol{\Delta}\mathbf{R}_i} = \log\left(\left(\boldsymbol{\Delta}\tilde{\mathbf{R}}_i^{i+1}(\bar{\mathbf{b}}_i^g) \exp\left(\frac{\partial \boldsymbol{\Delta}\bar{\mathbf{R}}_i^{i+1}}{\partial \mathbf{b}^g} \delta\mathbf{b}^g \right)^\wedge \right)^\top \mathbf{R}_i^{i+1} \right)^\vee$$

$$\mathbf{e}_{\boldsymbol{\Delta}\mathbf{v}_i} = (\mathbf{R}_i^{i+1}\mathbf{v}_{i+1} - \mathbf{v}_i) - G\hat{\mathbf{g}}^i \Delta t - \left(\boldsymbol{\Delta}\tilde{\mathbf{v}}_{i,i+1}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \frac{\partial \boldsymbol{\Delta}\bar{\mathbf{v}}_{i,i+1}}{\partial \mathbf{b}^g}\delta\mathbf{b}^g + \frac{\partial \boldsymbol{\Delta}\bar{\mathbf{v}}_{i,i+1}}{\partial \mathbf{b}^a}\delta\mathbf{b}^a \right)$$

$$\mathbf{e}_{\boldsymbol{\Delta}\mathbf{t}_i} = \mathbf{t}_i^{i+1} - \mathbf{v}_i\Delta t - \frac{1}{2}G\hat{\mathbf{g}}_i\Delta t^2 - \left(\boldsymbol{\Delta}\tilde{\mathbf{p}}_{i,i+1}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \frac{\partial \boldsymbol{\Delta}\bar{\mathbf{p}}_{i,i+1}}{\partial \mathbf{b}^g}\delta\mathbf{b}^g + \frac{\partial \boldsymbol{\Delta}\bar{\mathbf{p}}_{i,i+1}}{\partial \mathbf{b}^a}\delta\mathbf{b}^a \right)$$

$$\mathbf{e}_{\mathbf{b}_i^g} = \mathbf{b}_{i+1}^g - \mathbf{b}_i^g$$

$$\mathbf{e}_{\mathbf{b}_i^a} = \mathbf{b}_{i+1}^a - \mathbf{b}_i^a$$

$$\mathbf{e}_{\hat{\mathbf{g}}_i} = \log\left(\hat{\mathbf{g}}_i - \mathbf{R}_i^{i+1}\hat{\mathbf{g}}_{i+1}\right)^\vee$$

The first three errors $\mathbf{e}_{\boldsymbol{\Delta}\mathbf{R}_i}$, $\mathbf{e}_{\boldsymbol{\Delta}\mathbf{v}_i}$ and $\mathbf{e}_{\boldsymbol{\Delta}\mathbf{p}_i}$ refer to the agreement between the estimated relative rotation, velocity and translation and the preintegrated inertial measurements between keyframes i and $i+1$. The preintegration of the IMU measurements between the discrete time instants when such keyframes where captured, K_i and K_{i+1}, is as follows

$$\begin{aligned} \boldsymbol{\Delta}\tilde{\mathbf{R}}_i^{i+1}(\bar{\mathbf{b}}_i^g) &= \prod_{k=K_i}^{K_{i+1}-1} \exp\left(\tilde{\boldsymbol{\omega}}_k - \bar{\mathbf{b}}_i^g \Delta t^\wedge\right) \\ \boldsymbol{\Delta}\tilde{\mathbf{v}}_{i,i+1}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) &= \\ &= \sum_{k=K_i}^{K_{i+1}-1} \boldsymbol{\Delta}\tilde{\mathbf{R}}_i^k(\bar{\mathbf{b}}_i^g)\ \left(\tilde{\mathbf{a}}_k - \bar{\mathbf{b}}_i^a\right)\Delta t \\ \boldsymbol{\Delta}\mathbf{p}_i^{i+1} &= \sum_{k=K_i}^{K_{i+1}-1} \Big(\boldsymbol{\Delta}\mathbf{v}_{i,k}\Delta t + \\ &\quad + \frac{1}{2}\boldsymbol{\Delta}\tilde{\mathbf{R}}_i^k(\bar{\mathbf{b}}_i^g)\left(\tilde{\mathbf{a}}_k - \bar{\mathbf{b}}_i^a\right)\Delta t^2 \Big) \end{aligned} \tag{9}$$

where $\tilde{\boldsymbol{\omega}}_k$ and $\tilde{\mathbf{a}}_k$ are the measurements from the gyroscope and accelerometer respectively.

As the gyroscope and accelerometer biases $\mathbf{b}^g$ and $\mathbf{b}^a$ are initially unknown and drift over time, they have to be estimated. In order to account for the effect of these changes on the preintegrated terms from Eq. (9), Forster et al. [30] formulates them as a function of $\mathbf{b}^g$ and $\mathbf{b}^a$ and applies first order corrections ($\frac{\partial}{\partial \mathbf{b}^g}\delta\mathbf{b}^g$ and $\frac{\partial}{\partial \mathbf{b}^a}\delta\mathbf{b}^a$) during minimization.

In contrast to Forster et al. [30], we do not need to compute the relative transform from keyframe i to $i+1$, as it is directly estimated as $\mathbf{T}_i^{i+1}$ in our state. Velocities, positions and gravity are also referred to their local frames, which further simplifies the residual definitions. Also, we include $\mathbf{r}_{\hat{\mathbf{g}}_i}$ related to the gravity direction, which models the expected consistency between its change between keyframes i and $i+1$ and the corresponding orientation change $\mathbf{R}_i^{i+1}$ during that interval.

The covariance matrix $\mathbf{Q}^I$ for the inertial residual is computed incrementally during integration as shown in [31]. For the gravity residual term, we simply use a very low uncertainty value in order to express a strong constraint.

4.2.3. Prior residual

We include a third residual term in our cost function, that corresponds to a prior in the underlying MAP estimation. This residual takes the following form

$$\left\|\mathbf{e}^P\right\|^2_{(\mathbf{Q}^P)^{-1}} = \left\|\mathbf{x} \boxminus \hat{\mathbf{x}}\right\|^2_{(\mathbf{Q}^P)^{-1}} \tag{10}$$

modeling a Gaussian prior $\mathcal{N}(\hat{\mathbf{x}}, \mathbf{Q}^P)$ on the state. In practice there will be different factors connected to individual variables which compose the complete state $\mathbf{x}$, such as the first gravity vector and bias terms of the estimation window.

A generalization of this residual is also included which corresponds to the result of marginalization process and will take the following form:

$$\left\|\mathbf{e}^P\right\|^2 = \left\|\mathbf{A}_m(\mathbf{x} \boxminus \hat{\mathbf{x}}) + \mathbf{b}_m\right\|^2 \tag{11}$$

where, again, this residual may not necessarily be defined over the whole state $\mathbf{x}$ but over the subset affected by marginalization (known as the Markov blanket). In Appendix B we show how to arrive to the previous expression and how $\mathbf{A}_m$, $\mathbf{b}_m$, $\hat{\mathbf{x}}$ are obtained.

4.3. Local parametrizations

Following recent approaches [32], in our VIO we optimize variables in their local parametrizations. In the following sections we detail the parametrization of relative transforms as well as of the gravity direction vector and define the appropriate operators and their Jacobians.

4.3.1. Transform parametrization

We optimize relative transforms in their corresponding tangent space, in minimal coordinates. In particular, we choose the right $\boxplus$ and $\boxminus$ operators [32], which is consistent with the majority of related works:

$$\mathbf{T} \boxplus \boldsymbol{\delta} = \mathbf{T} \exp\left(\boldsymbol{\delta}^{\wedge}\right)$$
$$\mathbf{T_1} \boxminus \mathbf{T_2} = \log\left(\mathbf{T_2}^{-1}\mathbf{T_1}\right)^{\vee}$$

where exp and log are, respectively, the exponential and logarithmic maps for SE(3). Jacobians of these operators with respect to the perturbations in minimal coordinates can be obtained by applying the chain rule (see Appendix C).

Since we are interested in obtaining not only the maximum-a-posteriori (MAP) estimates but also their uncertainty, we model transforms as Gaussian variables $\mathbf{T}$ with:

$$\mathbf{T} = \bar{\mathbf{T}} \exp\left(\boldsymbol{\delta}^{\wedge}\right) \tag{12}$$

where $\bar{\mathbf{T}} \in SE(3)$ is the mean transform and $\boldsymbol{\delta} \in \mathbb{R}^6 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is a local perturbation. To operate with uncertain transforms, we follow the pose composition, inversion and point transformation defined in [33]. However, since Barfoot [33] employs a left-perturbation model for uncertain poses, we perform the required adaptations to follow the right-perturbation model of Eq. (12), as presented in Appendix A.

4.3.2. Gravity vector parametrization

In this framework, the gravity vector direction is naturally expressed as an element of $\mathbb{R}^3$. However, direction (unit) vectors belong to the S^2 manifold, which has only two degrees of freedom. As S^2 does not form a Lie group, we follow the parametrization proposed in [34] which we briefly add here for completeness[1].

Given two unit vectors $\hat{\mathbf{g}}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^{\top} \in \mathbb{R}^3, i \in 1, 2$ and a local perturbation $\boldsymbol{\delta} \in \mathbb{R}^2$, the $\boxplus$ and $\boxminus$ operators are defined as:

$$\hat{\mathbf{g}}_1 \boxplus \boldsymbol{\delta} = \mathbf{R}_{\hat{\mathbf{g}}_1} \exp(\boldsymbol{\delta})$$
$$\hat{\mathbf{g}}_2 \boxminus \hat{\mathbf{g}}_1 = \log(\mathbf{R}_{\hat{\mathbf{g}}_1}^{\top} \hat{\mathbf{g}}_2)$$

The exponential and logarithmic mappings are defined as follows

$$\exp(\boldsymbol{\delta}) = \begin{bmatrix} \cos(\|\boldsymbol{\delta}\|) \\ \operatorname{sinc}(\|\boldsymbol{\delta}\|)\boldsymbol{\delta} \end{bmatrix}$$

$$\log \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} = \begin{cases} \operatorname{atan2}(0, w)\mathbf{e}_1 & \mathbf{v} = \mathbf{0} \\ \operatorname{atan2}(\|\mathbf{v}\|, w)\dfrac{\mathbf{v}}{\|\mathbf{v}\|} & \mathbf{v} \neq \mathbf{0} \end{cases}$$

where $\mathbf{e}_1 = [1\ 0]^{\top}$ is the first unit vector, $w \in \mathbb{R}$, $\mathbf{v} \in \mathbb{R}^2$ and

$$\mathbf{R}_{\hat{\mathbf{g}}_i} = \begin{bmatrix} x_i & -r & 0 \\ y_i & x_i \cos\alpha & -\sin\alpha \\ z_i & x_i \sin\alpha & \cos\alpha \end{bmatrix},$$
$$\alpha = \operatorname{atan2}(z_i, y_i), r = \sqrt{y_i^2 + z_i^2}$$

4.4. Initialization

In order to bootstrap our visual-inertial state estimator, it is necessary to address the initialization of its different parts, as some of them will not be observable until enough observations have been gathered. Our initialization process will start with a window size of $n = 2$, where a visual-only estimation will be first performed, for the purpose of initializing the inertial states (velocity, gravity and IMU bias terms), after which full VIO estimation is initiated.

[1] Note that [34] erroneously defines $r = \sqrt{x^2 + z^2}$

4.4.1. Landmark initialization

For landmark initialization we employ stereo triangulation. First, salient image features are extracted in the left and right images and matched by nearest-neighbor search using binary descriptors. In general, during keyframe creation, previously seen landmarks are not refined (and used for camera tracking instead) and only unseen ones are instantiated. Since the stereo image-pair is already rectified, stereo correspondences are only sought in adjacent image rows (i.e. along the epipolar lines). This accelerates the search and further reduces false positives.

Left–right feature positions (x_l, y_l) and (x_r, y_r) of successful stereo matches are then used to triangulate the landmark position $\boldsymbol{\ell}$ in the left camera coordinates. We obtain the landmark covariance matrix $\Lambda \in \mathbb{R}^{3\times3}$ by linear propagation of the triangulation function $\pi^{-1} : \mathbb{R}^3 \to \mathbb{R}^3$:

$$\pi^{-1}(\begin{bmatrix} x_l & x_r & y_l \end{bmatrix}^{\top}) = \frac{b}{d}\begin{bmatrix} x_l & y_l & 1 \end{bmatrix}^{\top}$$
$$d = x_l - x_r$$

where we only consider the vertical coordinate of the feature in the left camera for simplicity. The landmark covariance is then:

$$\Lambda = \mathbf{J}_{\pi^{-1}} \mathbf{Y} (\mathbf{J}_{\pi^{-1}})^{\top}$$

where

$$\mathbf{J}_{\pi^{-1}} = \frac{b}{d}\begin{bmatrix} -\frac{x_r}{d} & \frac{x_l}{d} & 0 \\ -\frac{y_l}{d} & \frac{y_l}{d} & 1 \\ -\frac{1}{d} & \frac{1}{d} & 0 \end{bmatrix}$$

and $\mathbf{Y} = \sigma_f \mathbf{I}^{3\times3}$ is the covariance matrix of the noise related to feature extraction and matching.

4.4.2. Inertial states initialization

With the initial keyframe and its set of 3D landmarks, for subsequent stereo pairs and IMU measurements we run a visual-only estimation followed by a simplified visual-inertial estimator with some fixed states. We address initialization as a visual-inertial alignment problem, similarly to Lin et al. [35].

The visual estimator uses stereo residuals $\mathbf{e}^V$ in the cost function and a state composed only of the relative transforms $\mathbf{T}_i^{i+1}$. After this, a second estimator is run with the full state and it is initialized in two stages: first, $\mathbf{v}_i$, $\hat{\mathbf{g}}_i$, $\mathbf{b}_i^g$ are estimated leaving $\mathbf{T}_i^{i+1}$ fixed (with the value from the previous visual estimator) as well as $\mathbf{b}_i^a$ which is set to zero. During this stage, the preintegrated velocities $\boldsymbol{\Delta}\mathbf{v}_{i,i+1}$ and translation $\boldsymbol{\Delta}\mathbf{p}_{i,i+1}$ are omitted in $\mathbf{e}_{\boldsymbol{\Delta}\mathbf{v}_i}$ and $\mathbf{e}_{\boldsymbol{\Delta}\mathbf{t}_i}$ which allows $\mathbf{v}_i$ to be estimated from visual observations alone and the IMU states to be initialized. From the first and second keyframes, $\mathbf{b}^g$ is initialized and kept fixed in the second stage, where $\mathbf{b}_i^a$ initial value is estimated. As sensor excitation may not be enough to achieve observability, a zero-mean prior over $\mathbf{b}_0^a$ is introduced.

Initialization is completed after a user-defined number of keyframes. While it would be more appropriate to measure state convergence before finishing initialization, since we introduce keyframes only after certain angular or linear motion, this process works in practice. Our stereo setup simplifies the initialization, a monocular one would require richer motions before the scale and the inertial states converge.

4.5. Tracking

The camera pose is tracked as follows. Every time a new stereo pair is captured, feature extraction, description and stereo matching is performed. Successfully stereo-matched features are then

matched against a set of landmark candidates that are expected to be visible from the current camera pose. For details on this *active matching* approach see Section 4.5.1. These landmark-to-feature correspondences are then inserted into the optimization as visual factors. Furthermore, as IMU measurements arrive, they are stored in a buffer (using a separate thread, which loss of IMU readings) as well as used to perform state prediction at IMU rate (see Section 4.5.3). A buffer is used in order to handle the reception delay of images with respect to IMU measurements, since it is typical that the former will arrive later. After every new image-reception, the IMU preintegration interval spanning the current reference keyframe and the active frame is updated using the corresponding IMU measurements stored in the buffer. Tracking then proceeds by minimizing the cost in Eq. (5), which results in an updated estimate for the state $\mathbf{x}$ that includes the latest sensor data.

4.5.1. Active-matching

In order to track the robot w.r.t. to the map, the first step is to determine a set of candidate landmarks to be matched. This set is built from all landmarks triangulated from stereo pairs within the estimation window. Notice that landmarks from marginalized states are removed and cannot be re-visited again. Hence, the size of the optimization window is a design parameter that should be carefully chosen depending on the desired behavior: a smaller window is optimized in less time but landmarks are removed at high rates. This increases the chance of landmark duplication, increasing also the errors associated to drift. Larger windows have larger computational footprints, but the larger context in the optimization window and larger landmark tracks lead to higher accuracy.

The proposed feature-to-landmark matching strategy follows an active approach where the camera pose is first predicted to the current time frame based on the latest sensor data and then both the pose and landmark position uncertainty are used to obtain high-probability areas in the current image where feature correspondences should be found. In contrast to many works where this matching step only considers a fixed image region around landmark projections, this active-matching approach will naturally extend or reduce the search area size depending on the actual pose uncertainty. Thus, matches are established under a probabilistic setting, based on a given user-defined confidence interval instead, which removes the need to hand-tune a pixel-based search region size.

Given a set of candidate landmarks, active-matching then consists in projecting their 3D position and covariance matrix into left and right images. To obtain the 2D landmark projection from its 3D coordinates, the same kinematic-chain procedure of Eq. (7) is used. For covariance projection, Eq. (8) is extended to consider the uncertainty of all transforms involved in the kinematic chain, as in [10]. Thus, for a given landmark ${}^{j}\boldsymbol{\ell}$, referenced from keyframe j, and associated covariance matrix $\boldsymbol{\Phi}$ we wish to obtain its 2D projection $\check{\mathbf{y}}$ into frame i and its covariance $\check{\mathbf{Y}}$, which is obtained as:

$$\check{\mathbf{y}} = \pi\left(g\left(\mathbf{T}_i^j, {}^{j}\boldsymbol{\ell}\right)\right)$$

$$\check{\mathbf{Y}} = \mathbf{J}_\pi\left(\mathbf{A}\boldsymbol{\Sigma}_i^j\mathbf{A}^\top + \mathbf{R}_i^j\boldsymbol{\Phi}\mathbf{R}_j^i\right)\mathbf{J}_\pi^\top$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_i^j & -\mathbf{R}_i^j({}^{i}\boldsymbol{\ell})^\wedge \end{bmatrix}$$

and $\boldsymbol{\Sigma}_i^j$ is the covariance of $\mathbf{T}_i^j$. For details of these operations please refer to Appendix A.

From $\check{\mathbf{Y}}$ and a given confidence interval (we choose 95%), 2D ellipses can be obtained for the left and right projections, which encompass high-probability areas around each landmark projection where matching features are sought. To efficiently filter features enclosed into the uncertainty ellipses, we first build a list of axis-aligned bounding-boxes from the ellipses. From features inside the bounding-box, a match is established if its image descriptor is close enough to the landmark's last observed descriptor and if the Mahalanobis distance between projection and feature is less than a threshold, as in:

$$(\mathbf{y} - \tilde{\mathbf{y}})(\mathbf{Y} + \hat{\mathbf{Y}})(\mathbf{y} - \tilde{\mathbf{y}})^\top < \chi_4^2 \tag{13}$$

with $\mathbf{Y}$ defined as in (8) and selecting the appropriate χ_4^2 threshold for the chosen confidence interval.

4.5.2. Keyframe creation

Keyframes are inserted periodically based on a set of conditions. Whenever a new keyframe is introduced, new landmarks (not matched to any candidate landmark) are initialized as explained in Section 4.4.1.

A keyframe will be inserted if several thresholds on the motion w.r.t. the most recent keyframe are exceeded, which ensures sufficient sensor excitation between keyframes and also controls the sampling density of the robot motion during the teach phase. We also introduce a new keyframe when the ratio of successfully tracked landmarks is below a given threshold. This ratio will be low whenever the camera moves and observes the scene from a different perspective and also whenever the camera rotates and faces a new area.

While previous conditions are sufficient to trigger keyframe insertion, we enforce a necessary condition of sufficient tracked landmarks for the new keyframe: the sum of both successfully tracked landmarks and new ones (from stereo-matched but not map-matched features) should be a above a given number. This ensures that the new keyframe will only be introduced when sufficient visual information is present, which helps to achieve stable state estimation.

If the sufficient but not the necessary conditions are met (e.g., the robot moves into a poorly lit area with very few visible features), the system will operate solely on predicted state information until all conditions are satisfied (i.e., the robot moves again into a sufficiently illuminated area).

4.5.3. State prediction

The map-based tracking previously described produces a state estimate at camera-rate. However, in order to control a highly dynamic platform a higher rate would be desirable. For this reason, we generate a state prediction at IMU rate, based on the latest inertial data. While in the long term these predictions will drift, for relatively short periods of time they can be quite accurate. Moreover, by considering the state covariance and appropriately propagating it, it is also possible to know the covariance for the predicted state. This is particularly useful for the previously described active matching procedure.

The state prediction consists in obtaining new mean values ${}^{*}\mathbf{T}_{n-1}^n$, ${}^{*}\mathbf{v}_n$, ${}^{*}\hat{\mathbf{g}}_n$ of the most recent relative transform, linear velocity and gravity direction respectively, based on the most recent IMU readings $\boldsymbol{\omega}_t, \mathbf{a}_t$. In the general case (estimator fully initialized), these predictions are produced as follows:

$${}^{*}\mathbf{T}_{n-1}^n = \mathbf{T}_{n-1}^n\boldsymbol{\Delta}\mathbf{T}$$

$${}^{*}\mathbf{v}_n = \boldsymbol{\Delta}\mathbf{R}^\top\left(\mathbf{v}_n + \boldsymbol{\Delta}\mathbf{v} + \hat{\mathbf{g}}_n G\Delta t\right)$$

$${}^{*}\hat{\mathbf{g}}_n = \boldsymbol{\Delta}\mathbf{R}^\top\hat{\mathbf{g}}_n$$

where

$$\boldsymbol{\Delta}\mathbf{T} = \begin{bmatrix} \boldsymbol{\Delta}\mathbf{R} & \boldsymbol{\Delta}\mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

$$\boldsymbol{\Delta}\mathbf{R} = \exp\left((\boldsymbol{\omega}_t - \mathbf{b}_n^g)\Delta t^{\wedge}\right)$$
$$\boldsymbol{\Delta}\mathbf{v} = (\mathbf{a}_t - \mathbf{b}_n^a)\Delta t$$
$$\boldsymbol{\Delta}\mathbf{p} = \tfrac{1}{2}\,\boldsymbol{\Delta}\mathbf{v}\Delta t + \mathbf{v}_n\Delta t + \tfrac{1}{2}\,\hat{\mathbf{g}}_n G\Delta t^2$$

The corresponding covariances can be obtained by first-order propagation, as detailed in Appendix A.

During initialization, the IMU cannot be used yet for state prediction since bias terms are unknown. The prediction in this case is simply defined as the same state as in the previous timestep, with an artificially inflated time-dependent covariance. While this is a very crude approximation, it is in practice sufficient for this stage. Once $\mathbf{b}^g$ is already estimated (but while $\mathbf{b}^a$ is not), $\boldsymbol{\Delta}\mathbf{R}$ can be used to predict the rotational component of $\mathbf{T}_{n-1}^n$ and to apply the corresponding rotation to $\mathbf{v}_n$ and $\hat{\mathbf{g}}_n$.

4.6. Marginalization

Our VIO is based on non-linear optimization over a sliding-window, and whenever new variables are introduced to the state, old ones are removed to keep a fixed window size. When variables are removed, measurement factors are also removed. If these factors were simply discarded, the result of the estimation would not be a MAP solution and would only consider short-term information. Marginalization is used in order to obtain a prior distribution over the remaining state while summarizing the information of the deleted factors.

Our strategy consists in the marginalization of the oldest velocity, gravity, biases and transform from $\mathbf{x}$ whenever the state is augmented with a new keyframe. Marginalizing these states in a relative VIO is relatively simple thanks to the choice of representing the local gravity at each keyframe. While this may seem redundant, if a single common gravity vector were to be estimated instead, it would need to be transformed repeatedly to a new frame after its local frame is marginalized. This would require the transformation of all already marginalized connected factors, which in principle would require the ability to re-linearize past information or employing another non-trivial approach.

In the context of non-linear state estimation, marginalized residuals are summarized by means of a new factor that is built using first-order approximations. This is known to be a source of errors and might lead to inconsistent state estimates (i.e., there is an artificial information gain). In particular, for long-term absolute pose estimation, this approximation can cause problems. Under a global formulation, it is usual to fix the initial pose in order to address the inherent gauge freedom. This leads to an increasing state uncertainty which accumulates these linear-approximation errors over time.

Leutenegger et al. [36] approached this issue by not fixing the initial pose. This was found to be more efficient by Zhang et al. [37], while also requiring to apply a transformation to the covariance matrices to obtain a geometrically meaningful result. Alternatively, Eckenhoff et al. [26] show that by switching to a relative representation before marginalization, the gauge freedom is eliminated. In contrast to these works our state is directly defined in relative terms, benefiting from the absence of gauge freedom problem and allowing us a more natural representation of the uncertainty.

5. Navigation

The *repeat* phase of the proposed VI-T&R navigation involves a series of tasks. First, the robot needs to be localized with respect to the prior map (built during *teach* phase). Then, a smooth trajectory is generated using the keyframes of the *teach* map that are closest to the current robot pose. Finally, a closed-loop controller is used to follow this trajectory.

During the *repeat* phase, the same VIO module used during *teach* is run. However, in this case the VIO map is not stored and is only used to relate the current robot motion to the *teach* map. The pose of the VIO module is used by the closed-loop controller, while re-localization in the *teach* map is run independently at a lower rate. This decoupling allows the robot to navigate even in case of tracking loss with respect to the *teach* map. When the robot is re-localized in the *teach* map after tracking loss, the use of relative transforms prevent the estimated pose to suffer a large correction. Such large corrections are typical in SLAM and can have a negative effect in the control loop.

5.1. Reference map localization

Under the proposed relative formulation, localizing against the reference map boils down to finding a relative transform $\mathbf{T}_{i^*}^i$ between the latest VIO keyframe $i = n-1$ and the closest keyframe i^* in the reference map. In this manner, the current robot pose can be related to the reference map by chaining this transform with $\mathbf{T}_{n-1}^n$, the most recent relative transform of the VIO state.

To obtain $\mathbf{T}_{i^*}^i$, we perform visual-only localization against the reference map by minimizing the reprojection error between observations in keyframe i and the corresponding landmarks observed in keyframe i^*, using the following cost function:

$$J(\mathbf{x}) = \sum_{k=1}^{m_i} \left\| \mathbf{e}_{i,k}^V \right\|^2_{(\mathbf{Q}_{i,k}^V)^{-1}} + \tag{14}$$
$$\left\| \mathbf{T}_{i^*}^i \ominus \check{\mathbf{T}}_{i^*}^i \right\|^2_{(\mathbf{Q}_{\check{\mathbf{T}}_{i^*}^i})^{-1}} \tag{15}$$

where $\check{\mathbf{T}}_{i^*}^i$ is a prior for the unknown transform. At the bootstrapping of the *repeat* stage (and when tracking is lost), keyframe i^* is unknown. We do place recognition using Bag-of-Words [38] (BoW) descriptors, between the last keyframe i and the complete set of keyframes in the *teach* stage. For this purpose, during the teach phase, we store a BoW representation for each keyframe. The prior $\check{\mathbf{T}}_{i^*}^i$ is obtained by PnP, robustified using RANSAC, from the nearest-neighbor feature correspondences between the candidate keyframes extracted by BoW matching.

In subsequent steps, after a new keyframe $i+1$ is inserted, the prior is first updated to represent the pose of keyframe $i+1$ w.r.t. keyframe i^* as $\check{\mathbf{T}}_{i^*}^i \mathbf{T}_i^{i+1}$. The keyframe $(i+1)^*$ closest to the keyframe $i+1$ in the *local map* starting from i^* is set as the new map reference node. The new prior $\check{\mathbf{T}}_{(i+1)^*}^i$ is then obtained as $\mathbf{T}_{(i+1)^*}^{i^*} \check{\mathbf{T}}_{i^*}^{i+1}$. Finally, Eq. (15) is used to obtain the resulting $\mathbf{T}_{(i+1)^*}^{i+1}$. Landmark matches between $i+1$ and $(i+1)^*$ are established by projecting landmarks observed in $(i+1)^*$ to $i+1$, using $\check{\mathbf{T}}_{(i+1)^*}^{i+1}$, and running the active matching procedure described in Section 4.5.1.

If a minimum number of correspondences fails to be established, indicating a badly predicted prior or that an unmapped area is being explored, relocalization based on BoW is triggered. Until successfully relocalized, $\check{\mathbf{T}}_{(i+1)^*}^{i+1}$ is used as the localization result (and thus, as input to the closed-loop controller), resulting in purely predicted map-to-VIO pose.

5.2. Trajectory generation

Once localized with respect to the desired path, a target trajectory is built to allow for smooth path following. Under a relative formulation, since the reference frame i^* will be constantly changing, the trajectory will need to be recomputed. This presents the challenge of building a new trajectory without introducing discontinuities which would impede smooth path following.

To address this, we proceed as follows. We first build a locally Euclidean map of poses centered around i^* by chaining the corresponding relative transforms of the prior map. We then use these poses and linear and angular velocities associated with each keyframe (as estimated during the *teach* phase) as constraints for a polynomial trajectory generator [39]. The trajectory is obtained in the four-dimensional space of x, y, z and yaw, which correspond to the controllable degrees of freedom of a multirotor platform. For the desired arrival time at each keyframe we use the travel time relative to the initial pose of this local map, as observed during the teach phase. The trajectory built in this manner should be very similar to the one followed during the *teach* run. It is also possible to repeat the path faster by scaling the desired arrival time or by tuning the aggressiveness of the polynomial generator.

The resulting trajectory can then be sampled with a *time cursor* t to obtain linear and angular position, velocity and acceleration setpoints for a closed-loop controller. The difficulty here lies in that, whenever the reference keyframe i^* is redefined, t cannot be used to sample the new trajectory, since $t = 0$ now represents a different pose. As the travel times between every pair of keyframes of the path are known, we can obtain the new cursor t' over the new trajectory as $t - t_0$, where t_0 is the arrival time of the initial node of the new trajectory measured w.r.t. the old trajectory's initial node. As a result we effectively map the previous cursor t expressed with respect to the previous trajectory to the new cursor t' measured with respect to the new trajectory.

Furthermore, in order to handle possible discontinuities between previous and current trajectories, we build the local map from both past and future poses (up to a distance), which implies that discontinuities would only appear at the extremes of the trajectories. Since a new trajectory is computed after a new keyframe is inserted, the trajectory around the cursor t is always locally smooth.

5.3. Closed-loop control

From the map localization result and the generated trajectory, we can compute a path-tracking error between the current robot pose (referenced to the nearby portion of the map), velocity and acceleration and the current setpoint and its derivatives sampled from the trajectory. For the case of a multirotor platform, these setpoints can be used as input to a proportional controller in order to achieve the trajectory tracking. In this work, for the case of experiments performed in simulation, we employ the position-velocity tracking controller from Lee et al. [40]. For real-world experiments, the position and velocity setpoints are fed into the proportional controller of the internal autopilot, which combines a PID position-velocity controller on top of a separate attitude controller [41].

In terms of the relative formulation, the closed-loop control presents a challenge since most controllers expect the current state and setpoint as absolute quantities reference to a fixed coordinate frame. From these, an error is internally computed and used to generate the control outputs. Since the pose error can be actually considered a relative transform between current and desired pose and since the proposed framework directly obtains this information, in this context it is more appropriate to employ a controller which directly expects the control error, i.e. the relative pose. For this reason, we modify the controller from Lee et al. [40] to include this possibility. Similarly, we directly employ the gravity vector expressed in the trajectory's reference as part of the controller equations, instead of the canonical vector which would be used when working under a gravity-aligned reference frame.

On the other hand, since performing these modifications to the autopilot of the multirotor platform used for real-world experiments can be particularly challenging, we opted for an alternate option. Since the controller expects all estimates referred to the absolute coordinate frame, we simply obtain an absolute pose by combining the relative transforms estimated by the VIO module. While this pose will drift over time, it will only be used to compute an error with respect to the desired pose, which will be similarly obtained. Thus, the control error will be in general sufficiently accurate for closed-loop control.

6. Experimental results

In this section we analyze the performance of the proposed VI-T&R navigation method. We first focus on the localization accuracy of the relative visual-inertial estimation, and of the map-based localization during the repeat phase. Then, we analyze the computational cost of different steps of the VI-T&R method, particularly when run on a resource-constrained hardware platform. Finally, we validate the complete VI-T&R navigation approach by performing closed-loop control experiments both in simulated and in real-world environments.

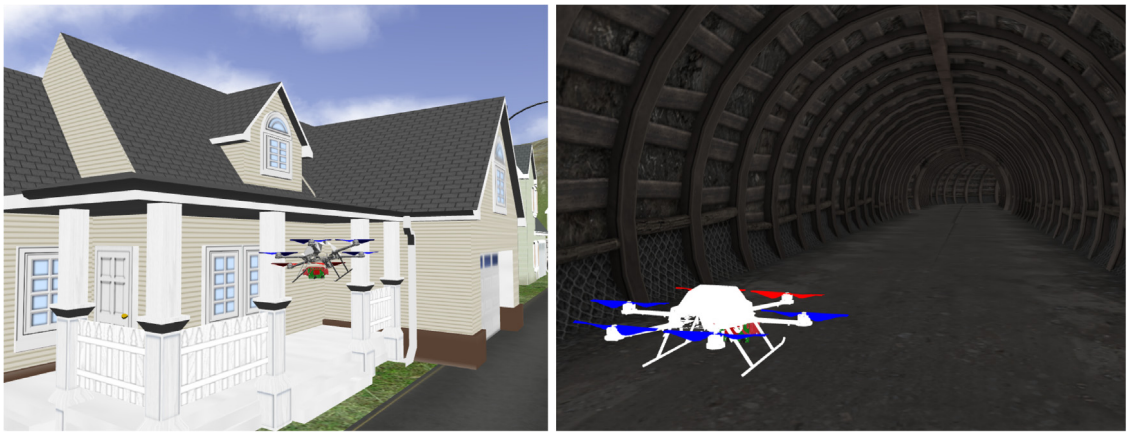
6.1. Experimental setup

In our experiments we use two distinct computing platforms: a single-board Odroid XU-4 ARM computer, featuring eight cores running at 2 GHz, and a modern desktop Intel i7-4790 computer, with eight cores running at 3.6 GHz. The former is chosen since it can be carried on board of a small multirotor vehicle, while the latter allows to establish an expected performance baseline without significant hardware constraints.

The VI-T&R system was implemented in C++ using the Robot Operating System (ROS) framework, coupled with Sophus [42] for Lie algebra calculations, OpenCV for feature extraction and image processing and Ceres [43] for the non-linear solver. On the Odroid computer, we employ an NEON optimized feature extraction and description library [44].

While the framework allows different image feature detector and description algorithms to be used, we experimentally found that Good Features To Track algorithm (GFTT) by Jianbo Shi and Tomasi [45] detects robust features which can be tracked longer while also being homogeneously distributed in the image. Furthermore, it depends on a *relative* feature saliency threshold instead of an absolute value (as most other detection algorithms) which results in robustness to image contrast changes. The main limitation to GFTT is that it is computationally expensive on embedded platforms. For this reason, on the Odroid computer we have chosen to use a NEON optimized implementation of the FAST feature extractor. For feature description we chose the ORB descriptor which can be computed efficiently. On the Odroid computer we further simplify this computation by obtaining the descriptor only along the upright image patch orientation, which thus is equivalent to that of BRIEF algorithm but using ORB's descriptor pattern.

As a simulation tool we used the rotorS package for Gazebo [46], choosing an AscTec Firefly hexacopter platform with a stereo VI sensor, configured at 20 frames per second with a resolution of 752 $\times$ 480 px and an IMU rate of 200 Hz. For real-world experiments we employ a custom built quadrotor platform, based on the Pixhawk autopilot, which carries the Odroid XU-4 computer and a MyntEye stereo-camera which integrates an internal Inertial Measurement Unit hardware-synchronized to the camera clock. The MyntEye is configured to run at the same rate and image-resolution as our simulated experiments.



(a) `outdoor` environment (b) `tunnel_practice` environment

Fig. 1. Simulation environments.

6.2. VIO localization

In this section we evaluate the performance of our relative VIO module. For this purpose, we run the VI-T&R system in *teach* mode on the EuRoC MAV dataset [47]. This dataset contains sensor data captured with an Astec Firefly hex-rotor equipped with a visual-inertial sensor while flying in two different environments.

In order to analyze the VIO localization accuracy we measure the relative pose error (RPE) [48] for increasing subsequence lengths, up to a given distance. For each subsequence we compose the relative transforms and compare the final pose with the relative pose obtained from the dataset ground truth. We do not report absolute pose errors, since it is of no relevance for T&R navigation: T&R approaches require only local consistency within the portion of the map used in the repeat phase. We choose a maximum subsequence length of 4 m since this is a local area of sufficient size to run map-based localization and trajectory generation during the repeat phase.

We show experiments on the two platforms described in Section 6.1, in order to assess the effect of limited computational resources on real-time tracking. Our experiments were done in real time: stereo pairs will be skipped if the computation for the previous one did not finish. We also tested the case of non hardware-accelerated GFTT/ORB on Odroid, to establish the impact on tracking quality resulting from the use of the simpler FAST detector.

For these (and all subsequent experiments) we have set an estimation window of size $n = 5$ which was found to give a good balance between tracking performance and estimation quality.

In Fig. 2 we show the results from this experiment. In general, there is no significant loss on the tracking performance when it is run on the Odroid XU-4 instead of the Intel i7. The biggest differences can be observed in the `V` series of EuRoC, as these are more challenging.

In terms of overall localization accuracy, we can observe a relative translation error between 1% to 4%. While such errors can be thought of as higher than some other VIO systems, it should be highlighted that we are using a much simpler hardware than those works, and our approach demonstrates real-time closed-loop on-board control on a small aerial robot.

6.3. Bias and gravity estimation

To further assess the performance of the proposed VIO, we also evaluate the accuracy of the IMU bias and gravity estimation. We present two sets of experiments in *teach* mode. First, we run the system on the EuRoC dataset (as in the previous section) and compare the IMU bias estimated against the available ground-truth. Second, since ground-truth gravity is not available for the EuRoC dataset, we perform a series of simulations where both bias and gravity can be evaluated. We use the rotorS framework, that provides a realistic simulation of the IMU by including appropriate IMU noise and bias drift.

From all of our experiments, we choose two subsequences of both `MH` and `V` series of the EuRoC dataset as a representative sample. Fig. 3 shows the estimation of the three components of the accelerometer and gyroscope biases over time. Observe how the estimation is close to the ground truth value, displayed as an orange line. As before we overlay 3σ bounds from our uncertainty estimation, to assess the consistency of the estimation.

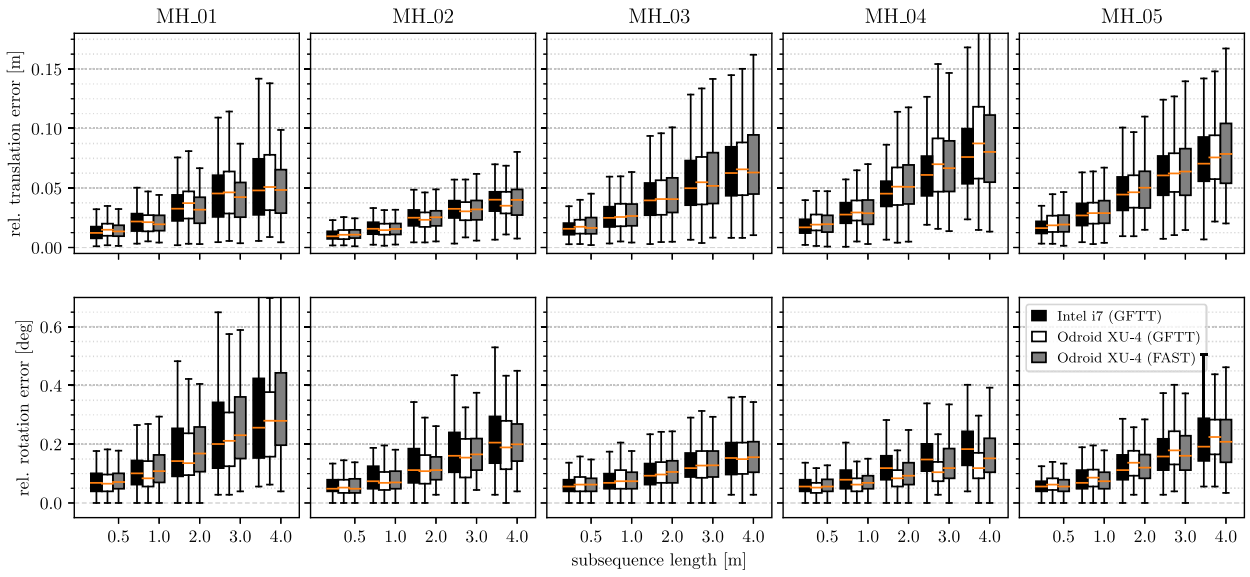
Observe that the bias estimation converge, in general, to the expected value within the 3σ uncertainty region. It should be remarked that, for the `V` series, the convergence is slower and in some cases it is not even reached (such as for accelerometer bias for `V1_01`). Again, this difference can be explained by the challenging conditions of this series of EuRoC sequences.

For our simulation experiments, we evaluate the bias estimation accuracy as before, and also the components of the local gravity vector. We performed these analysis using two different simulations: a shorter one (68s) which includes varied motions in different directions and rotations and a longer one (255s) which mostly consists of long periods of approximately constant velocity over long distances with only a few changes in motion direction. The results are presented in Fig. 4.

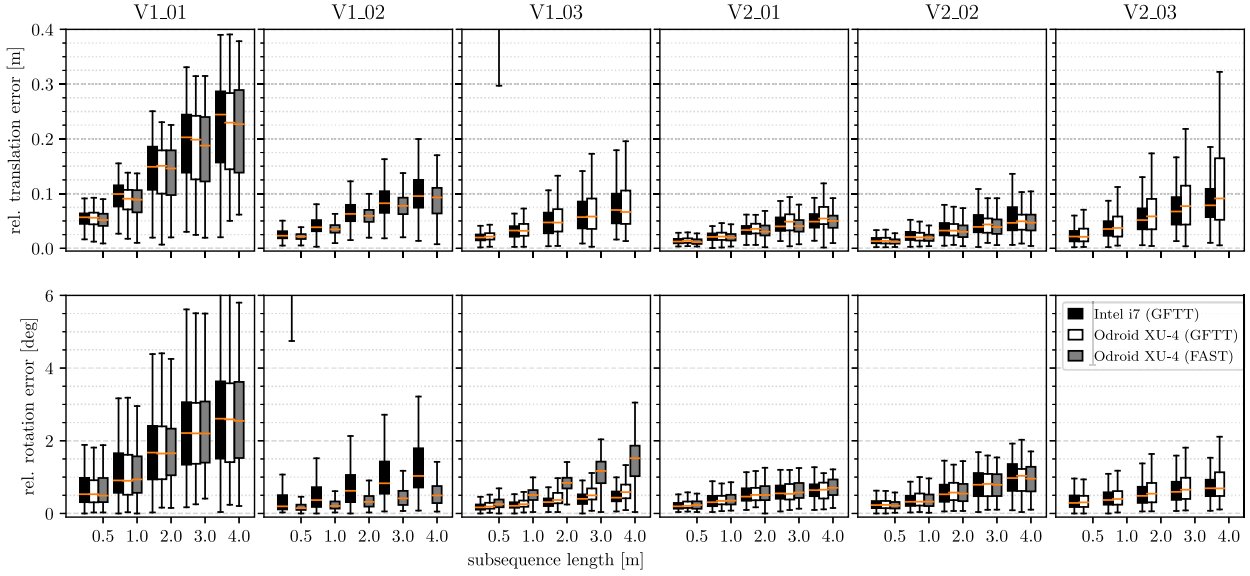
Observe that in these experiments the bias is accurately estimated and the estimated gravity follows closely the ground-truth with low uncertainty. Moreover, the second simulated experiment serves to verify the correct IMU bias terms even when operating for longer periods of time where bias drift could be an issue.

6.4. Map localization

In this section we analyze the results of localizing the robot against a previously built map as it is performed during the repeat phase. Specifically, we compare the estimated relative transform between the active VIO map and the prior map built in the *teach* phase against the ground-truth transform. For this experiment we again used the EuRoC dataset, where we take advantage of the partial overlap between different sequences of the dataset in the two environments: we run VIO on one sequence in *teach* mode and we run the *repeat* localization using a different sequence.



(a) MH sequences



(b) V sequences

Fig. 2. Relative Pose Error of VIO estimation Error measured on EuRoC dataset for Intel i7 and Odroid computers. Missing or out of range data signifies tracking failure.

Since the EuRoC dataset sequences do not fully overlap (ie. the robot sometimes goes over areas not visited during the *teach* phase), this experiments also allow to analyze the ability of the proposed method to handle large deviations with respect to the *teach* sequences. In these cases, when a recent VIO keyframe cannot be localized against a nearby map keyframe, the last successful localization result (map-to-VIO relative transform) is still chained with the VIO localization result from that point on. Thus, during these *exploratory* periods, map-localization can be considered only a prediction since it is not yet corrected with recent visual information (i.e., a map match to current image features).

In Fig. 5 we present the translation error for the VIO-to-map relative transform, for two pairs of teach-repeat experiments using pairs of EuRoC sequences with sufficient overlap. As in the previous section, we perform the experiment on both hardware platforms. We distinguish in these results measurements which are obtained from successful map matches with the last VIO keyframe (in green) from those which re-use last result and simply chain VIO tracking results (in red). Furthermore, we overlay 3σ uncertainty region to indicate the method confidence in the result. It can be seen that for time periods where only a prediction is obtained the uncertainty grows until relocalized to the map when the error also drops again.

A number of observations can be done from these results. First, we can see the translation error of the estimated relative transform is generally within 5 to 10 cm for both hardware platforms. When analyzing in closer detail, for the MH_01/MH_02 combination, at $t = 30$ the system running on Odroid XU-4 fails to localize and the error is higher. Similarly, the error increases between $t = 40$ and $t = 50$ in MH_04/MH_05 Odroid XU-4. In any case, in both situations map tracking is quickly re-established once relocalization is successful. For purely *exploratory* parts, observe for example the results between times $t = 40$ to $t = 60$

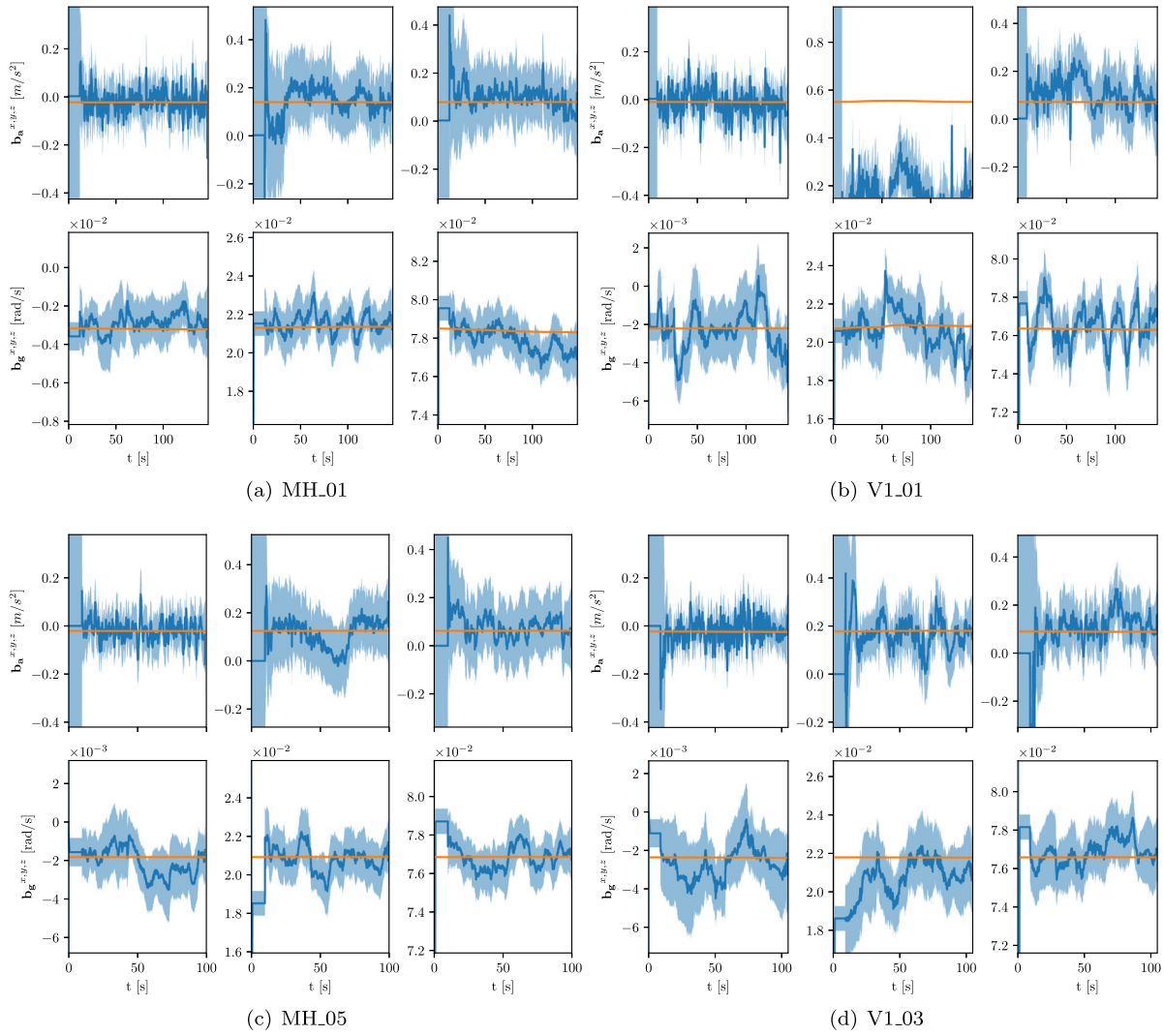


Fig. 3. Accelerometer and gyroscope bias estimation over time, measured over EuRoC dataset MH and V sequences. Orange line corresponds to ground-truth bias value whereas blue corresponds to estimation (dark blue: mean value, light blue: 3σ uncertainty region).

and $t = 75$ to $t = 85$ in MH_01/MH_02. In both cases, map-based localization is only predicted from VIO and uncertainty thus increases accordingly while still maintaining a low error of around 10 cm.

Another interesting case is the tracking failure at $t = 45$ for the MH_04/MH_05 case, with subsequent relocalization at around $t = 55$. Here the error does not only increases considerably, but it is also outside the confidence region. When inspecting the corresponding timeframe in MH_05 we observe that it corresponds to almost full darkness with only few distant visible objects.

In general, the results in this section indicate that map-based localization is of sufficient accuracy for trajectory following in real-time on an embedded platform such as the Odroid XU-4. Furthermore, localization appears to be robust to several challenging situations.

6.5. Computational cost

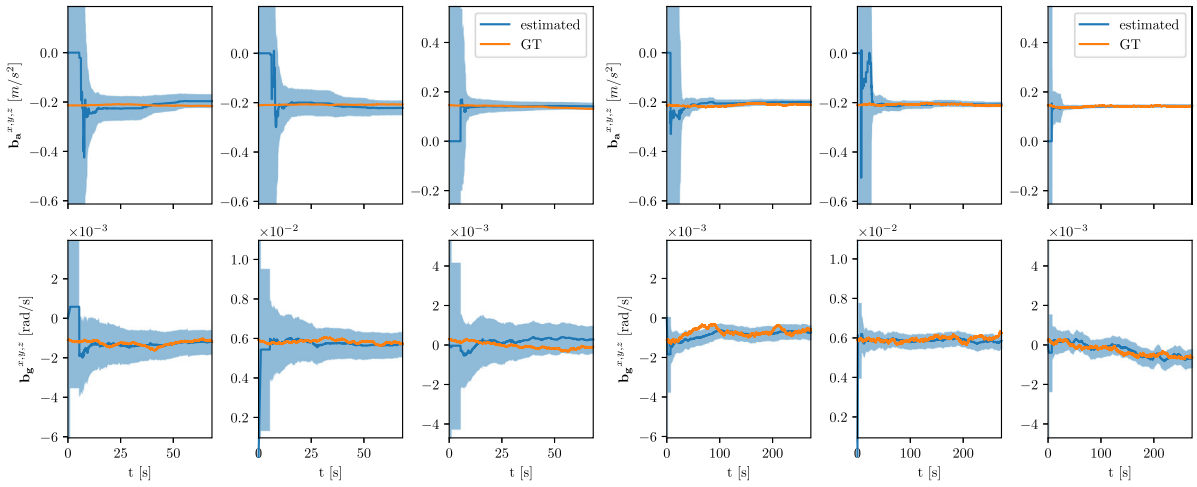
We report the running time of the main steps of our algorithm in both hardware platforms and, as before, we include the case of GFTT and hardware-accelerated FAST detector on Odroid XU-4. We also measure times in both teach and repeat modes to analyze the performance of the VIO task running alone (as done during *teach*) and in series with the map-based localization (as done in *repeat*).

Fig. 6(a) reports the average running time of the main steps of the VIO task which involve: a) feature extraction and description (on left and right images, which is performed in parallel), b) active-matching (see 4.5.1), c) VIO map tracking , d) solver minimization and covariance computation, e) marginalization and f) keyframe addition.

From this first result, we can observe that feature extraction and solver are the most computationally demanding tasks. For the GFTT case in particular, observe that there is approximately a seven-fold increase in runtime between Intel i7 and Odroid X-U4. On the contrary, we can also see that the hardware-accelerated FAST algorithm running on Odroid X-U4 is not only faster than GFTT on the same platform, but also on Intel i7. This is a significant difference which results in a reduction of the total VIO time of about 50%.

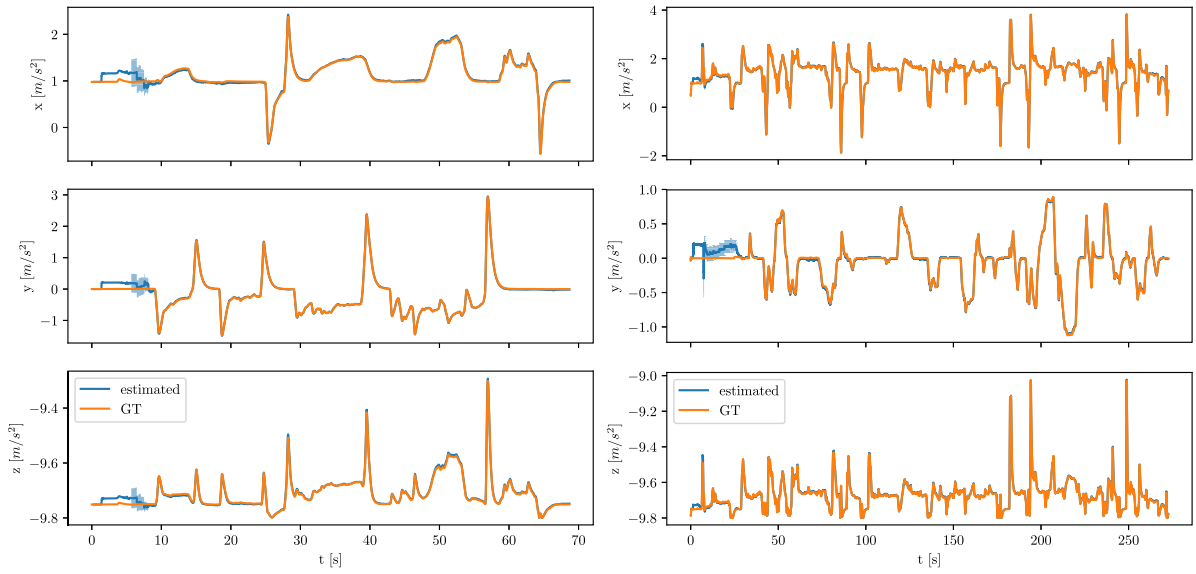
In general terms, the average frame rate of the VIO task results in around 27 Hz for Intel i7 while for Odroid X-U4 it is approximately 11 Hz and 6 Hz for FAST and GFTT respectively.

In Fig. 6(b) we report the complete tracking time for the VI-T&R system in *repeat* mode. In this case, feature extraction and VIO are those presented in 6(a) while the other steps involved in *repeat* phase include tracking of the prior map and global relocalization (see 5.1), the latter being run only when tracking is not successful. For this case we see that the running time of



(a) Accelerometer and gyroscope bias, experiment #1

(b) Accelerometer and gyroscope bias, experiment #2



(c) Gravity vector, experiment #1

(d) Gravity vector, experiment #2

Fig. 4. Accelerometer, gyroscope bias and gravity estimation over time, measured over simulated environments. Orange line corresponds to ground-truth value whereas blue corresponds to estimation (dark blue: mean value, light blue: 3σ uncertainty region).

these extra steps is smaller than those of feature extraction and VIO task. However, on Odroid X-U4, map-based localization cost becomes significant when compared to the Intel i7 case. As a whole, this amounts to an average tracking rate of 25 Hz for Intel i7 and 9 Hz and 5 Hz for Odroid X-U4 with FAST and GFTT detector, respectively.

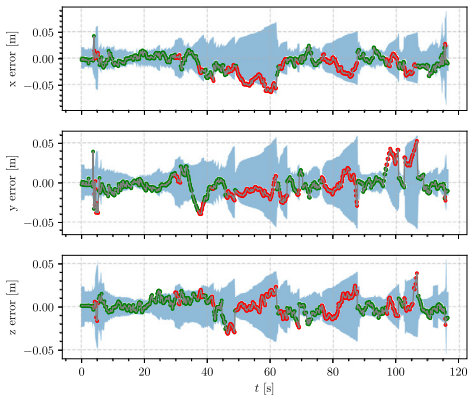
Given a camera frame-rate of 20 Hz we can see that the Intel i7, in average, would be capable of processing all frames while the Odroid X-U4 would incur in a considerable image-loss (around 50% for VIO). However, as we are not losing inertial measurements and also maintain an IMU-rate state prediction, we can still perform accurate estimation at high rate. Of course, as the UAV moves faster and performs more aggressive motion, image-tracking will incur in higher estimation errors and would eventually fail. However, we did not encounter this problem in practice.

6.6. Navigation

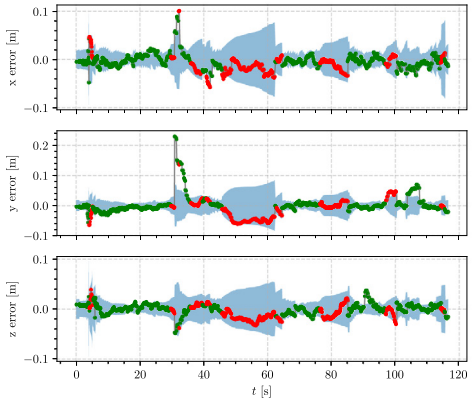
While previous experiments aim to verify that the VI-T&R performance is sufficient for navigation in resource-constrained UAVs, in this section we experimentally validate the accuracy in repeating a previously taught trajectory. We perform these experiments partly in simulation and partly on a real aerial platform. Simulations allow us to focus on the VI-T&R system performance against ground truth. Still, since simulations might not capture all the detail of real-world scenarios and platforms (e.g., IMU noise and realistic images) we include qualitative results on a real platform to validate the system.

For the aforementioned experiments we first run our system in *teach* mode while manually flying the aerial robot. Then, we perform several passes in *repeat* mode. We report qualitative views of all of our experiments, and quantitative assessment of the path following errors for the simulation cases.

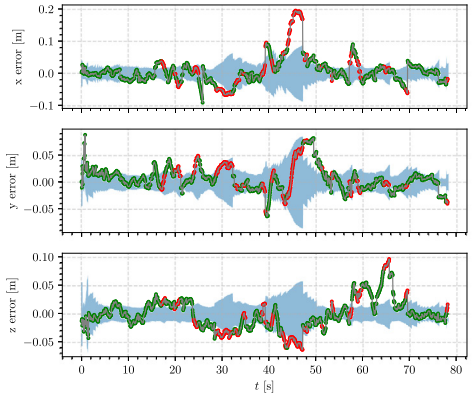
Fig. 7 shows our experiments, using the rotorS simulator. We used different setups: `outdoors` environment included in the simulator and `tunnel_practice_1`, `tunnel_practice_2`



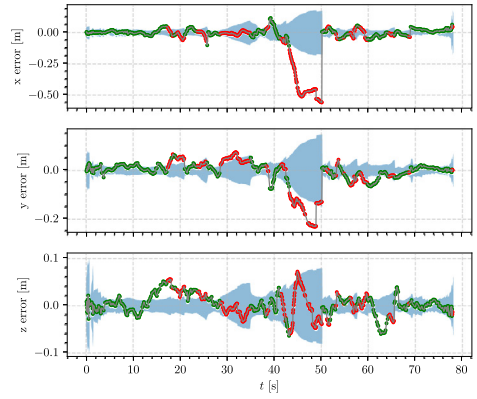
(a) Teach: MH_01, Repeat: MH_02. Intel i7



(b) Teach: MH_01, Repeat: MH_02. Odroid XU-4



(c) Teach: MH_05, Repeat: MH_04. Intel i7



(d) Teach: MH_05, Repeat: MH_04. Odroid XU-4

Fig. 5. Map-based localization error on EuRoC dataset. Green indicate successful map-based localization while red correspond to predicted value based on short-term VIO localization result. Shaded blue regions are 3σ uncertainty bounds.

environments from DARPA's Subterranean Challenge [49] support package (see Fig. 1). For `outdoors` we performed two inspection-like runs where the robot flies around a building. For the other scenarios we simply followed the tunnel in an arbitrary fashion for a longer period of time. Thus, while the first case involves fast motions and rich visual information the second involves long and mostly straight motion in a self-similar environment.

When analyzing the performance of the trajectory following experiments performed in simulation a maximum error of approximately 20 to 50 cm can be observed. Moreover, this error does not change significantly when comparing the shorter and longer sets of experiments. On the other hand, it is likely that by improving the tuning constants of the position-velocity controller included with the rotorS simulator this error could be reduced. This is also evidenced by the fact that in the last two experiments there is an initial control error in the y (longitudinal) direction but not on the other two axis. In other words, the vehicle lags behind the setpoint until it is able to eventually *catch up*.

For the real-world experimentation, we present in Fig. 8 the perceived control error for each of the five repeat runs. In these figures we shade the time interval where the robot is under control of the T&R system: since takeoff and landing maneuvers are difficult to safely handle under autonomous operation without GPS assistance we manually fly these portions of the path. Since no ground-truth information is available for these tests, we show external views of the robot during these experiments in Fig. 9 and Fig. 10, to aid in the qualitative analysis of the path-following behavior.

With the obtained results we can indeed observe that the robot is able to smoothly follow the trajectory under guidance of the T&R method proposed in this work, running in real-time on the embedded Odroid X-U4 installed on the aerial robot. It is also possible to observe that there is some controller delay which likely indicates better tuning of the low-level controller, which for this experiment is part of the Pixhawk autopilot.

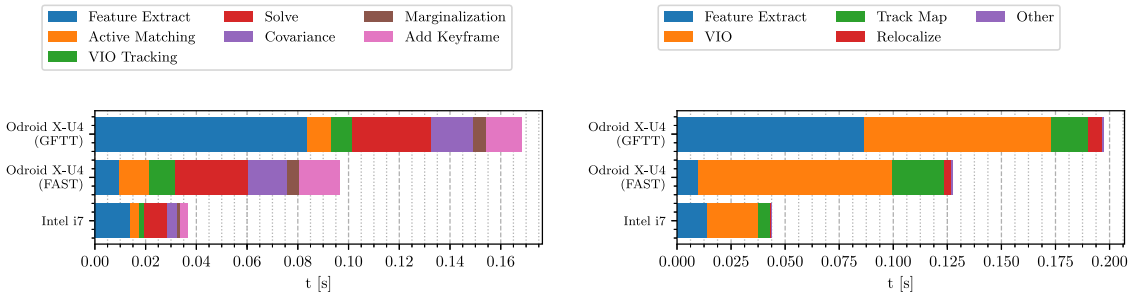
In more general terms, this experiment also demonstrates in practice the ability to build a map by fusing visual and inertial data on board a resource constrained hardware platform suitable for small unmanned aerial vehicles as well as to solve both localization against this map, trajectory generation and following.

7. Conclusions

We have presented a full VI-T&R system based on tightly-coupled fusion of visual stereo information and inertial data, suitable for small unmanned aerial robots in applications such as 3D inspection or surveillance. The method was tested on the challenging EuRoC dataset and demonstrated closed-loop performance in simulation and in real-world experiments.

The resulting precision of the system running on the low-power Odroid X-U4 computer was shown to be comparable to that obtained with a more powerful Intel i7 computer which does not suffer from frame-loss. By employing a relative formulation of the localization problem and by seeking only local consistency, we achieve a system capable of running at a rate of about 10 Hz on the embedded computer when coupled with a hardware-accelerated feature processing pipeline. Furthermore, we verified convergence of inertial states under this formulation, which involves distinguishing the gravity vector, and thus empirically verify the observability of the problem.

In order to validate our claim of achieving real-time operation suitable for UAV navigation we first test our system in simulated environments consisting of both short and long-distance navigation scenarios. We then performed experiments on a real robotic platform with the system running fully on-board and demonstrated path-following behavior of a previously learned path.



(a) Teach (b) Repeat

Fig. 6. Average running times of our VI-T&R, for Intel i7 and Odroid XU-4, for *teach* (using `MH_01` EuRoC) and *repeat* (using `MH_01` and `MH_02` EuRoC).

Given the various components that conform the proposed VI-T&R system, we can identify several streams of future work. First, we wish to extend our relative VIO to support a map-adjustment task which would potentially increase the accuracy of the method and further reduce resulting pose uncertainty. Second, since one of the main benefits of the relative formulation is the ability to naturally represent loops in the underlying graph-based map, we plan to add the ability for loop detection during teach stage. This also opens the possibility to perform navigation decisions over a network of paths, which further extends the application scenarios of our approach. Finally, we consider introducing optimizations to the solver minimization and covariance computation steps to further decrease computation time.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research was supported by UBACYT project 20020170200397BA (University of Buenos Aires, Argentina), PICT project 2015-3167 (Ministry of Science and Technology, Argentina), project PGC2018-096367-B-I00 (Ministry of Science and Innovation, Spain) and Grupo DGA-T45-17R/FSE (Aragón regional government).

Appendix A. Uncertain transform operations

In this section we obtain the appropriate definition for uncertain pose composition, inversion and uncertain point transformation, following a right-perturbation model (see Eq. (12)). For simplicity we obtain second order approximation of covariance matrices.

A.1. Pose composition

Given two uncertain transforms $\mathbf{T}_1, \mathbf{T}_2$, we wish to obtain their composition $\mathbf{T} = \mathbf{T}_1, \mathbf{T}_2$. Applying definition (12) we have:

$$\begin{aligned}\mathbf{T}_1\mathbf{T}_2 &= \bar{\mathbf{T}}_1 \exp\left(\boldsymbol{\xi}_1^{\wedge}\right)\bar{\mathbf{T}}_2 \exp\left(\boldsymbol{\xi}_2^{\wedge}\right) = \\ &= \bar{\mathbf{T}}_1\bar{\mathbf{T}}_2 \exp\left(\mathrm{Adj}\left(\bar{\mathbf{T}}_2^{-1}\right)\boldsymbol{\xi}_1^{\wedge}\right)\exp\left(\boldsymbol{\xi}_2^{\wedge}\right) \\ &= \bar{\mathbf{T}}\exp\left(\boldsymbol{\xi}^{\wedge}\right)\end{aligned}$$

where Adj () is the adjoint of SE(3). Applying the Baker-Campbell-Hausdorff formula [33], we can perform a second-order approximation to obtain the covariance matrix $\boldsymbol{\Sigma}$ of the transformation $\mathbf{T}$ as:

$$\begin{aligned}\boldsymbol{\Sigma} &= \mathrm{E}\left[\boldsymbol{\xi}\boldsymbol{\xi}^{\top}\right] \simeq \\ &\simeq \mathrm{E}\left[\boldsymbol{\xi}_1'\boldsymbol{\xi}_1'^{\top} + \boldsymbol{\xi}_2\boldsymbol{\xi}_2^{\top}\right] \simeq \\ &\simeq \boldsymbol{\Sigma}_1' + \boldsymbol{\Sigma}_2 = \mathrm{Adj}\left(\bar{\mathbf{T}}_2^{-1}\right)\boldsymbol{\Sigma}_1\mathrm{Adj}\left(\bar{\mathbf{T}}_2^{-1}\right)^{\top} + \boldsymbol{\Sigma}_2\end{aligned}$$

where we used $\boldsymbol{\xi}_1' = \mathrm{Adj}\left(\bar{\mathbf{T}}_2^{-1}\right)\boldsymbol{\xi}_1$ as a shorthand.

A.2. Pose inverse

Given an uncertain transform $\mathbf{T}$ we wish to obtain its inverse:

$$\mathbf{T}^{-1} = \left(\bar{\mathbf{T}}\exp\left(\boldsymbol{\xi}^{\wedge}\right)\right)^{-1} = \exp\left(-\boldsymbol{\xi}^{\wedge}\right)\bar{\mathbf{T}}^{-1}$$

The corresponding covariance matrix is then:

$$\boldsymbol{\Sigma}_{-1} = \mathrm{Adj}\left(\bar{\mathbf{T}}\right)\ \boldsymbol{\Sigma}\ \mathrm{Adj}\left(\bar{\mathbf{T}}\right)^{\top}$$

A.3. Point transformation

Given a homogeneous point $\mathbf{x} = \begin{bmatrix}\boldsymbol{\epsilon} & \eta\end{bmatrix} \in \mathbb{R}^4$, we model its uncertainty by $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{D}\boldsymbol{\zeta}$ where $\boldsymbol{\zeta} \in \mathbb{R}^3 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Phi})$ is a perturbation vector and $\mathbf{D} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}^{\top} \in \mathbb{R}^{4\times 3}$. In this case we wish to obtain the result of transforming $\mathbf{x}$ by $\mathbf{T}$:

$$\begin{aligned}\mathbf{T}\mathbf{x} &= \bar{\mathbf{T}}\exp\left(\boldsymbol{\xi}^{\wedge}\right)(\bar{\mathbf{x}} + \mathbf{D}\boldsymbol{\zeta}) \simeq \\ &\simeq \bar{\mathbf{T}}(\mathbf{I} + \boldsymbol{\xi}^{\wedge} + \frac{1}{2}\boldsymbol{\xi}^{\wedge}\boldsymbol{\xi}^{\wedge})(\bar{\mathbf{x}} + \mathbf{D}\boldsymbol{\zeta})\end{aligned}$$

where we can apply the following property from Barfoot [33]:

$$\begin{aligned}\boldsymbol{\xi}^{\wedge}\mathbf{x} &= \mathbf{x}^{\odot}\boldsymbol{\xi} \\ \mathbf{x}^{\odot} &= \begin{bmatrix}\eta\mathbf{I} & -\boldsymbol{\epsilon}^{\wedge} \\ \mathbf{0}^{\top} & \mathbf{0}^{\top}\end{bmatrix}\end{aligned}$$

resulting in

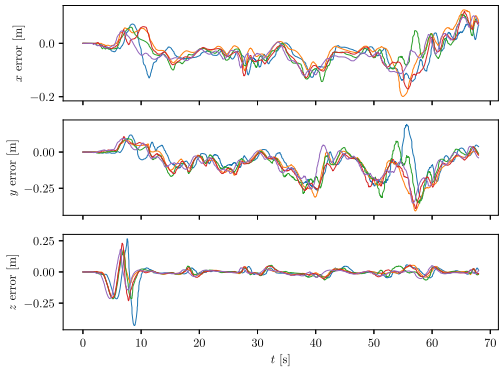
$$\begin{aligned}\mathbf{T}\mathbf{x} &\simeq \bar{\mathbf{T}}\bar{\mathbf{x}} + \bar{\mathbf{T}}\mathbf{D}\boldsymbol{\zeta} + \bar{\mathbf{T}}\boldsymbol{\xi}^{\wedge}\bar{\mathbf{x}} = \\ &= \bar{\mathbf{T}}\bar{\mathbf{x}} + \bar{\mathbf{T}}\mathbf{D}\boldsymbol{\zeta} + \bar{\mathbf{T}}\bar{\mathbf{x}}^{\odot}\boldsymbol{\xi}\end{aligned}$$

where we also dropped the higher order terms. We can then write the covariance matrix of the (inhomogeneous) transformed point as:

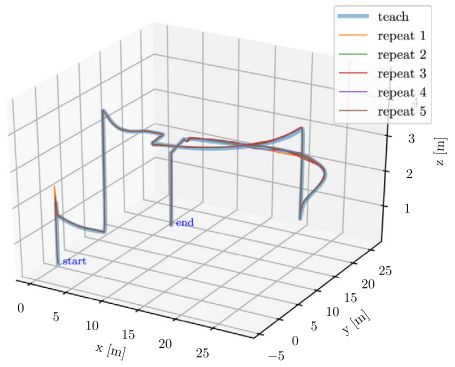
$$\begin{aligned}\boldsymbol{\Phi}' &= \mathbf{D}^{\top}\left(\bar{\mathbf{T}}\bar{\mathbf{x}}^{\odot}\boldsymbol{\Sigma}\left(\bar{\mathbf{T}}\bar{\mathbf{x}}^{\odot}\right)^{\top} + \bar{\mathbf{T}}\mathbf{D}\boldsymbol{\Phi}\left(\bar{\mathbf{T}}\mathbf{D}\right)^{\top}\right)\mathbf{D} = \\ &= \begin{bmatrix}\bar{\mathbf{R}} & -(\bar{\mathbf{R}}\boldsymbol{\epsilon}^{\wedge})\end{bmatrix}\boldsymbol{\Sigma}\begin{bmatrix}\bar{\mathbf{R}}^{\top} \\ -(\bar{\mathbf{R}}\boldsymbol{\epsilon}^{\wedge})^{\top}\end{bmatrix} + \bar{\mathbf{R}}\boldsymbol{\Phi}\bar{\mathbf{R}}^{\top}\end{aligned}$$

Appendix B. Marginalization

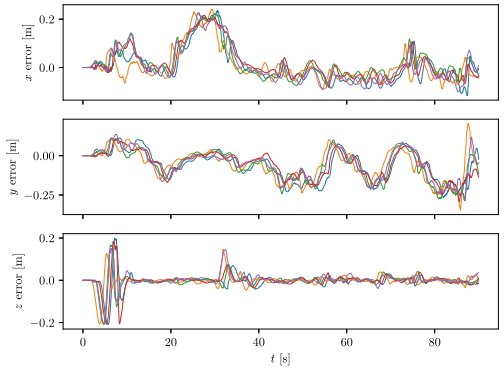
For the purpose of completeness, we here summarize the equations underlying the marginalization process in the context of the MAP estimation. Recall from Eq. (4) the we wish to obtain



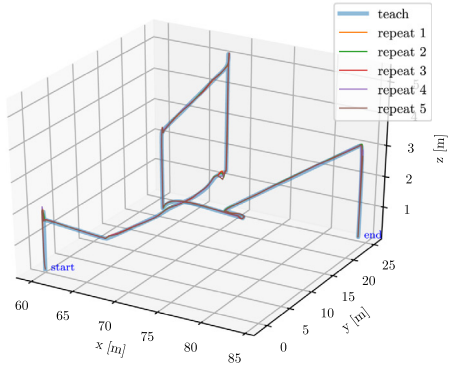
(a) `outdoor` experiment #1 control error



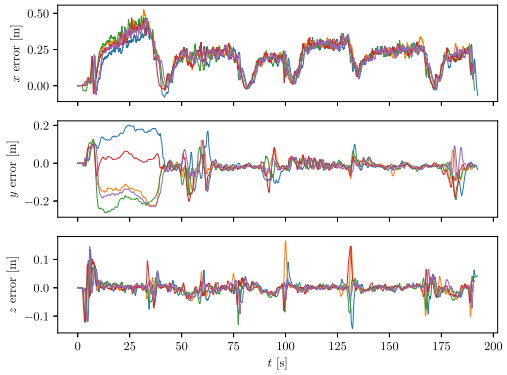
(b) `outdoor` experiment #1 trajectories



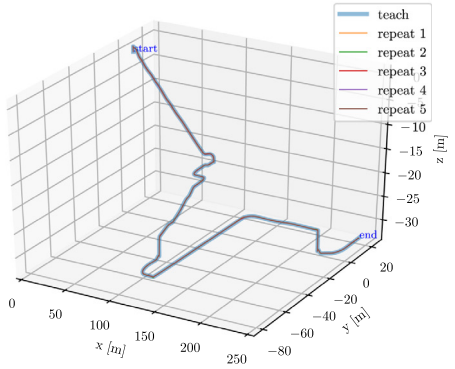
(c) `outdoor` experiment #2 control error



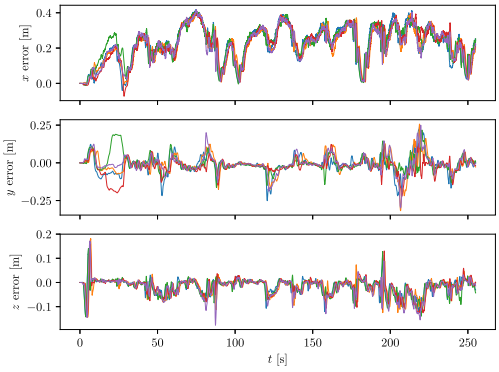
(d) `outdoor` experiment #2 trajectories



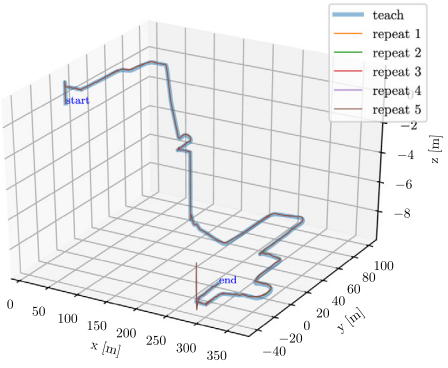
(e) `tunnel_practice_1` experiment control error



(f) `tunnel_practice_1` experiment trajectories

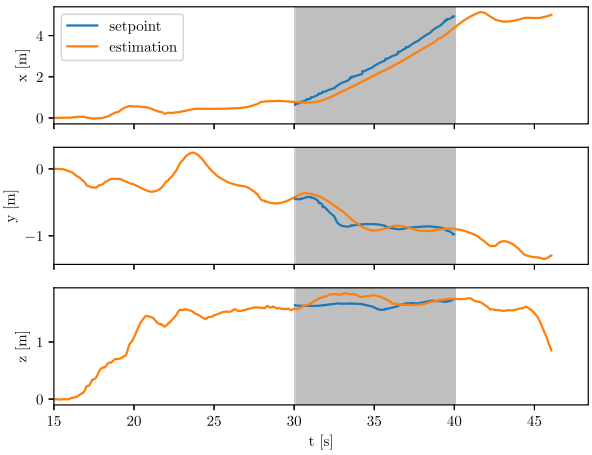


(g) `tunnel_practice_2` experiment control error

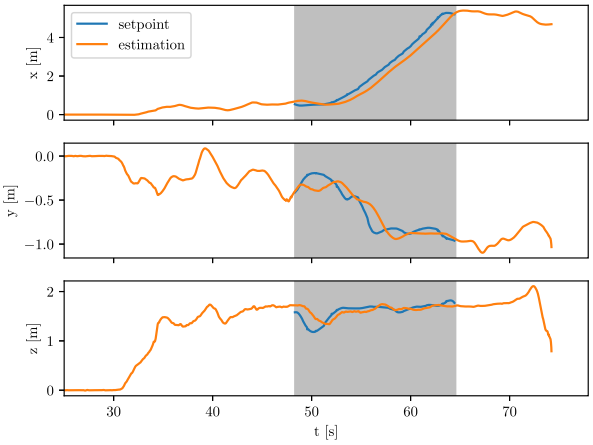


(h) `tunnel_practice_2` trajectories

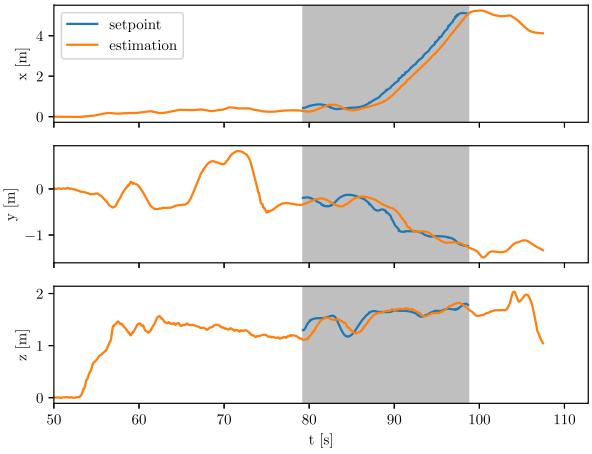
Fig. 7. Trajectory following experiments under simulation.



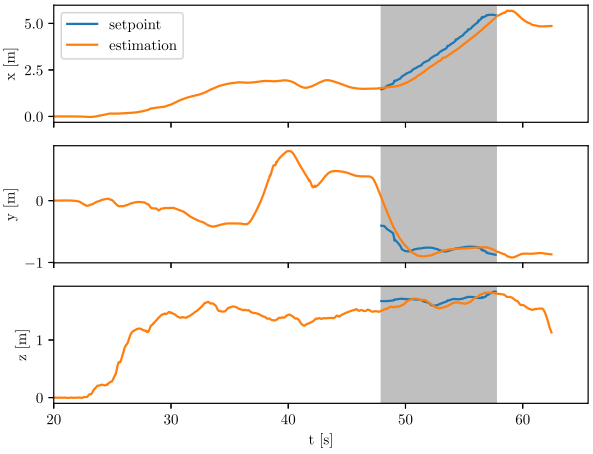
(a) Control error for repeat run #1



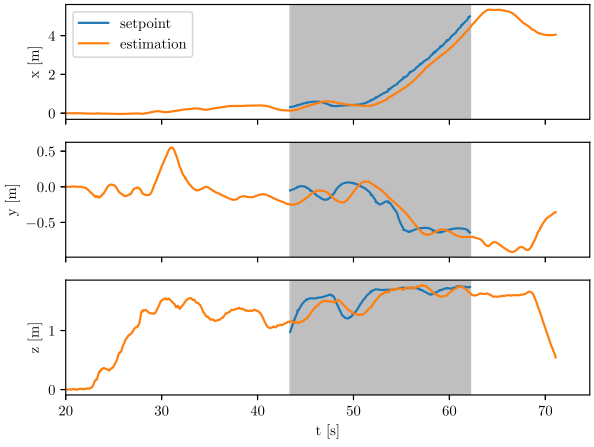
(b) Control error for repeat run #2



(c) Control error for repeat run #3



(d) Control error for repeat run #4



(e) Control error for repeat run #5

Fig. 8. Trajectory following experiments in real-world conditions.

a solution to a minimization problem via Least-Squares. Due to the non-linearity of $c(\mathbf{x})$, this problem is solved iteratively as:

$$\boldsymbol{\Delta}\hat{\mathbf{x}} = \underset{\boldsymbol{\Delta x}}{\arg\min} \sum_i \left\| \mathbf{e}_i(\hat{\mathbf{x}}) + \mathbf{J}_i(\boldsymbol{\Delta}\hat{\mathbf{x}}) \right\|^2_{\boldsymbol{\Omega}_i} \tag{B.1}$$

where

$$\mathbf{J}_i = \left. \frac{\partial \mathbf{e}_i(\hat{\mathbf{x}} \boxplus \boldsymbol{\Delta x})}{\partial \boldsymbol{\Delta x}} \right|_{\boldsymbol{\Delta x}=0} \tag{B.2}$$

Each iteration involves obtaining a new estimate $\hat{\mathbf{x}}^+$ from $\hat{\mathbf{x}}^-$, obtained in the previous step, applying the increment $\boldsymbol{\Delta}\hat{\mathbf{x}}$ as:

$$\hat{\mathbf{x}^+} = \hat{\mathbf{x}^-} \boxplus \boldsymbol{\Delta}\hat{\mathbf{x}} \tag{B.3}$$

(B.1) can be solved by means of the following system of equations:

$$\left(\sum_i \mathbf{J}_i^\top \boldsymbol{\Omega}_i \mathbf{J}_i \right) \boldsymbol{\Delta x} = - \sum_i \mathbf{J}_i^\top \boldsymbol{\Omega}_i \mathbf{e}_i(\hat{\mathbf{x}}) \tag{B.4}$$

Fig. 9. External view during the teach phase of the experiment using experimental aerial robot. In red we highlight the portion of the path that was later autonomously repeated.

which can also be expressed as:

$$\boldsymbol{\Lambda}\Delta\mathbf{x} = -\mathbf{b} \tag{B.5}$$

We will define a partition of the complete state as $\mathbf{x} = \{\mathbf{x}_m, \mathbf{x}_b, \mathbf{x}_r\}$, where $\mathbf{x}_m$ is the set of variables to be marginalized, $\mathbf{x}_b$ is the *Markov blanket* of $\mathbf{x}_m$ (the subset of $\mathbf{x}$ with factors connected to $\mathbf{x}_m$) and $\mathbf{x}_r$ the rest of $\mathbf{x}$ which is not related to $\mathbf{x}_m$. We can rewrite cost function $c(\mathbf{x})$ as:

$$c(\mathbf{x}_m, \mathbf{x}_b, \mathbf{x}_r) = c_r(\mathbf{x}_b, \mathbf{x}_r) + c_m(\mathbf{x}_m, \mathbf{x}_b) \tag{B.6}$$

As a result, Eq. (4) can be formulated as:

$$\begin{aligned}\arg\min_{\mathbf{x}} c(\mathbf{x}) &= \arg\min_{\mathbf{x}_b,\mathbf{x}_r} \left(\arg\min_{\mathbf{x}_m} c(\mathbf{x}_m, \mathbf{x}_b, \mathbf{x}_r) \right) = \\ &\arg\min_{\mathbf{x}_b,\mathbf{x}_r} \left(c_r(\mathbf{x}_b, \mathbf{x}_r) + \arg\min_{\mathbf{x}_m} c_m(\mathbf{x}_m, \mathbf{x}_b) \right)\end{aligned}$$

The goal of the marginalization is then to approximate the right term (which depends on $\mathbf{x}_m$) from known values of $\mathbf{x}_b$. For this purpose, we can apply a second-order Taylor approximation over c_m obtaining:

$$\begin{aligned}c_m(\mathbf{x}_m, \mathbf{x}_b) \simeq c_m(\hat{\mathbf{x}}_m, \hat{\mathbf{x}}_b) + \begin{bmatrix} \mathbf{g}_m \\ \mathbf{g}_b \end{bmatrix} \begin{bmatrix} \mathbf{x}_m \boxminus \hat{\mathbf{x}}_m \\ \mathbf{x}_b \boxminus \hat{\mathbf{x}}_b \end{bmatrix} + \\ \frac{1}{2} \begin{bmatrix} \mathbf{x}_m \boxminus \hat{\mathbf{x}}_m \\ \mathbf{x}_b \boxminus \hat{\mathbf{x}}_b \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\Lambda}_{mm} & \boldsymbol{\Lambda}_{bm} \\ \boldsymbol{\Lambda}_{bm} & \boldsymbol{\Lambda}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{x}_m \boxminus \hat{\mathbf{x}}_m \\ \mathbf{x}_b \boxminus \hat{\mathbf{x}}_b \end{bmatrix}\end{aligned} \tag{B.7}$$

From the previous expression we can write $\mathbf{x}_m$ as a function of $\mathbf{x}_b$ as:

$$\mathbf{x}_m = \hat{\mathbf{x}}_m - \boldsymbol{\Lambda}_{mm}^{-1}(\mathbf{g}_m + \boldsymbol{\Lambda}_{bm}^{-1}(\mathbf{x}_b \boxminus \hat{\mathbf{x}}_b)) \tag{B.8}$$

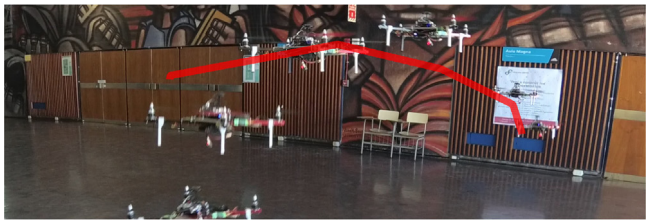
Substituting this expression in (B.7), we arrive at the following approximation for the minimization of c_m:

$$\begin{aligned}\arg\min_{\mathbf{x}_m} c_m(\mathbf{x}_m, \mathbf{x}_b) \simeq \\ \arg\min_{\mathbf{x}_m} c_{marg}(\mathbf{x}_m, \mathbf{x}_b) = \zeta + \mathbf{g}_{marg}^\top(\mathbf{x}_b \boxminus \tilde{\mathbf{x}}_b) + \\ \frac{1}{2}(\mathbf{x}_b \boxminus \tilde{\mathbf{x}}_b)\boldsymbol{\Lambda}_{marg}(\mathbf{x}_b \boxminus \tilde{\mathbf{x}}_b)^\top\end{aligned}$$

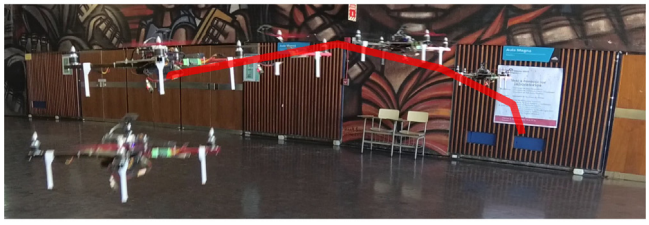
where

$$\begin{aligned}\mathbf{g}_{marg}^\top &= \mathbf{g}_b^\top - \boldsymbol{\Lambda}_{bm}\boldsymbol{\Lambda}_{mm}^{-1}\mathbf{g}_m^\top \\ \boldsymbol{\Lambda}_{marg} &= \boldsymbol{\Lambda}_{bb} - \boldsymbol{\Lambda}_{bm}\boldsymbol{\Lambda}_{mm}^{-1}\boldsymbol{\Lambda}_{bm}\end{aligned} \tag{B.9}$$

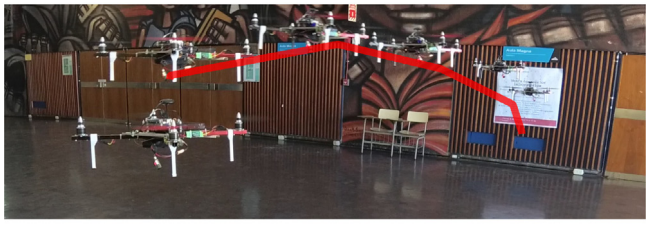
The resulting cost function c_{marg} can then be included as part of Eq. (4), where the new state will now only be $\mathbf{x} = \{\mathbf{x}_b, \mathbf{x}_r\}$.



(a) Trajectory following during repeat phase experiment #1

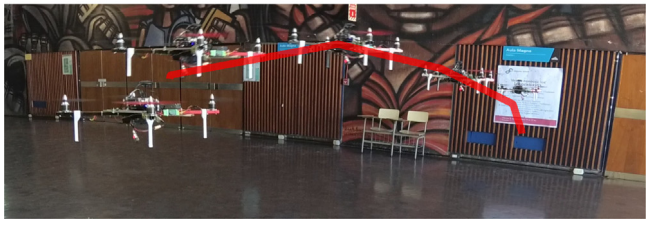


(b) Trajectory following during repeat phase experiment #2



(c) Trajectory following during repeat phase experiment #3

(d) Trajectory following during repeat phase experiment #4



(e) Trajectory following during repeat phase experiment #5

Fig. 10. External view of the aerial robot during the trajectory following experiment. The line marked in red corresponds to the approximate expected path to be followed as defined during teach phase.

Since c_{marg} contains both linear and quadratics terms, we can rewrite it as a single quadratic factor:

$$c_{marg}(\mathbf{x}_b) = \frac{1}{2}\left\|\mathbf{A}_m(\mathbf{x}_b \boxminus \hat{\mathbf{x}}_b) + \mathbf{b}_m\right\|_2^2 \tag{B.10}$$

where

$$\begin{aligned}\mathbf{A}_m^\top\mathbf{A}_m &= \boldsymbol{\Lambda}_{marg} \\ \mathbf{A}_m^\top\mathbf{b}_m &= \mathbf{g}_{marg}\end{aligned} \tag{B.11}$$

The equivalence of minimizing (B.10) and (B.11) can be shown as follows:

$$c_{marg}(\mathbf{x}_b) = \frac{1}{2}\left\|\mathbf{A}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b) + \mathbf{b}_m\right\|_2^2 =$$

$$= \frac{1}{2}\left(\mathbf{A}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b) + \mathbf{b}_m\right)^\top \left(\mathbf{A}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b) + \mathbf{b}_m\right) =$$
$$= \frac{1}{2}\left((\mathbf{x}_b - \hat{\mathbf{x}}_b)^\top \mathbf{A}_m^\top + \mathbf{b}_m^\top\right)\left(\mathbf{A}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b) + \mathbf{b}_m\right) =$$
$$= \frac{1}{2}(\mathbf{x}_b - \hat{\mathbf{x}}_b)^\top \mathbf{A}_m^\top \mathbf{A}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b) +$$
$$+ \frac{1}{2}(\mathbf{x}_b - \hat{\mathbf{x}}_b)^\top \mathbf{A}_m^\top \mathbf{b}_m +$$
$$+ \frac{1}{2}\mathbf{b}_m^\top \mathbf{A}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b) + \frac{1}{2}\mathbf{b}_m^\top \mathbf{b}_m =$$
$$= \frac{1}{2}(\mathbf{x}_b - \hat{\mathbf{x}}_b)^\top \boldsymbol{\Lambda}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b) +$$
$$+ \frac{1}{2}\left(\underbrace{\mathbf{b}_m^\top \mathbf{A}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b)}_{\mathbf{r}}\right)^\top +$$
$$+ \frac{1}{2}\underbrace{\mathbf{b}_m^\top \mathbf{A}_m(\mathbf{x}_b - \hat{\mathbf{x}}_b)}_{\mathbf{r}} + \frac{1}{2}\mathbf{b}_m^\top \mathbf{b}_m =$$
$$= \frac{1}{2}\left\|\mathbf{x}_b - \hat{\mathbf{x}}_b\right\|^2_{\boldsymbol{\Lambda}_m} + \frac{1}{2}\mathbf{r} + \frac{1}{2}\mathbf{r}^\top + \frac{1}{2}\mathbf{b}_m^\top \mathbf{b}_m =$$
$$= \frac{1}{2}\left\|\mathbf{x}_b - \hat{\mathbf{x}}_b\right\|^2_{\boldsymbol{\Lambda}_m} + \mathbf{r} + \frac{1}{2}\mathbf{b}_m^\top \mathbf{b}_m =$$
$$= \frac{1}{2}\left\|\mathbf{x}_b - \hat{\mathbf{x}}_b\right\|^2_{\boldsymbol{\Lambda}_m} + \mathbf{g}_{marg}^\top(\mathbf{x}_b - \tilde{\mathbf{x}}_b) + \frac{1}{2}\mathbf{b}_m^\top \mathbf{b}_m$$

where, since $\frac{1}{2}\mathbf{b}_m^\top \mathbf{b}_m$ is a constant term, it does not change the result of the minimization.

Appendix C. Pose parametrization Jacobians

For the Jacobian of $\boxplus$ operator w.r.t. the perturbation vector, for simplicity we use the Sophus library [42] implementation (which is obtained via symbolic computation). On the other hand, the Jacobian of $\boxminus$ is here presented in detail since this is often overlooked and its essential in order to implement a marginalization or prior factor. We require the Jacobian of this operator w.r.t. to its two operands. In the first case, we want to obtain the Jacobian of

$$\mathbf{T}_1 \exp\left(\boldsymbol{\delta}_1{}^\wedge\right) \boxminus \mathbf{T}_2 = \log\left(\mathbf{T}_2^{-1}\mathbf{T}_1 \exp\left(\boldsymbol{\delta}_1{}^\wedge\right)\right)^\vee$$

with respect to $\boldsymbol{\delta}_1$. To simplify the calculation, we will define $\mathbf{T} = \mathbf{T}_2^{-1}\mathbf{T}_1$ which allows to employ the following approximation:

$$\log\left(\mathbf{T}\exp\left(\boldsymbol{\delta}^\wedge\right)\right)^\vee \simeq \log(\mathbf{T})^\vee + \mathcal{J}_r^{-1}\left(\log(\mathbf{T})^\vee\right)\boldsymbol{\delta} \tag{C.1}$$

As a result, the desired Jacobian is now:

$$\frac{\partial \log\left(\mathbf{T}\exp\left(\boldsymbol{\delta}_1{}^\wedge\right)\right)^\vee}{\partial \boldsymbol{\delta}_1} =$$
$$= \frac{\partial \log(\mathbf{T})^\vee + \mathcal{J}_r^{-1}\left(\log(\mathbf{T})^\vee\right)\boldsymbol{\delta}_1}{\partial \boldsymbol{\delta}_1} =$$
$$= \mathcal{J}_r^{-1}\left(\log(\mathbf{T})^\vee\right)$$

For the opposite case, we have:

$$\mathbf{T} \boxminus \left(\mathbf{T}_2 \exp\left(\boldsymbol{\delta}_2{}^\wedge\right)\right) =$$
$$= \log\left(\left(\mathbf{T}_2 \exp\left(\boldsymbol{\delta}_1{}^\wedge\right)\right)^{-1}\mathbf{T}_1\right)^\vee =$$
$$= -\log\left(\mathbf{T}_1^{-1}\mathbf{T}_2 \exp\left(\boldsymbol{\delta}_1{}^\wedge\right)\right)^\vee \simeq$$
$$\simeq -\left(\log\left(\mathbf{T}^{-1}\right)^\vee + \mathcal{J}_r^{-1}\left(\log\left(\mathbf{T}^{-1}\right)^\vee\right)\boldsymbol{\delta}_2\right)$$

which results in:

$$\frac{\partial\, \mathbf{T} \boxminus \left(\mathbf{T}_2 \exp\left(\boldsymbol{\delta}_2{}^\wedge\right)\right)}{\partial \boldsymbol{\delta}_2} = \mathcal{J}_r^{-1}\left(\log\left(\mathbf{T}^{-1}\right)^\vee\right)$$

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J.J. Leonard, Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, IEEE Trans. Robot. 32 (6) (2016) 1309–1332.

[2] R. Brooks, Visual map making for a mobile robot, in: ICRA, 1985, http://dx.doi.org/10.1109/ROBOT.1985.1087348.

[3] C. Mei, G. Sibley, M. Cummins, P. Newman, I. Reid, RSLAM: A system for large-scale mapping in constant-time using stereo, IJCV 94 (2) (2011) 198–214, http://dx.doi.org/10.1007/s11263-010-0361-7.

[4] M. Nitsche, F. Pessacg, J. Civera, Visual-inertial teach & repeat for aerial robot navigation, in: 2019 European Conference on Mobile Robots (ECMR), IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/ECMR.2019.8870926.

[5] P. Krüsi, et al., Lighting-invariant adaptive route following using iterative closest point matching, JFR 32 (4) (2015) 534–564, http://dx.doi.org/10.1002/rob.21524.

[6] C. Sprunk, G.D. Tipaldi, A. Cherubini, W. Burgard, Lidar-based teach-and-repeat of mobile robot trajectories, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 3144–3149.

[7] C. McManus, P. Furgale, B. Stenning, T.D. Barfoot, Lighting-invariant visual teach and repeat using appearance-based lidar, J. Field Robotics 30 (2) (2013) 254–287.

[8] P. Furgale, T.D. Barfoot, Visual teach and repeat for long-range rover autonomy, J. Field Robotics (2006) (2010) 1–27, http://dx.doi.org/10.1002/rob.

[9] C.J. Ostafew, A.P. Schoellig, T.D. Barfoot, Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 176–181.

[10] M. Paton, K. MacTavish, M. Warren, T.D. Barfoot, Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat, in: IROS, 2016, http://dx.doi.org/10.1109/IROS.2016.7759303.

[11] J. Dequaire, C.H. Tong, W. Churchill, I. Posner, Off the beaten track: Predicting localisation performance in visual teach and repeat, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016, pp. 795–800.

[12] L. Clement, J. Kelly, T.D. Barfoot, Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery, J. Field Robotics 34 (1) (2017) 74–97.

[13] T. Krajník, F. Majer, L. Halodová, T. Vintr, Navigation without localisation: Reliable teach and repeat based on the convergence theorem, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 1657–1664.

[14] P. King, A. Vardy, A.L. Forrest, Teach-and-repeat path following for an autonomous underwater vehicle, J. Field Robotics 35 (5) (2018) 748–763.

[15] A. Pfrunder, A.P. Schoellig, T.D. Barfoot, A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera, in: CRV, 2014, http://dx.doi.org/10.1109/CRV.2014.40.

[16] M. Warren, M. Paton, K. MacTavish, A.P. Schoellig, T.D. Barfoot, Towards visual teach and repeat for GPS-denied flight of a fixed-wing UAV, in: M. Hutter, R. Siegwart (Eds.), Field and Service Robotics, 2018, pp. 481–498.

[17] M. Warren, M. Greeff, B. Patel, J. Collier, A.P. Schoellig, T.D. Barfoot, There's no place like home: Visual teach and repeat for emergency return of multirotor UAVs during GPS failure, IEEE Robot. Autom. Lett. 4 (1) (2019) 161–168, http://dx.doi.org/10.1109/LRA.2018.2883408, http://arxiv.org/abs/1809.05757. https://ieeexplore.ieee.org/document/8544011/.

[18] M. Fehr, T. Schneider, M. Dymczyk, J. Sturm, R. Siegwart, Visual-inertial teach and repeat for aerial inspection, 2018, arXiv:1803.09650.

[19] F. Gao, L. Wang, K. Wang, W. Wu, B. Zhou, L. Han, S. Shen, Optimal trajectory generation for quadrotor teach-and-repeat, IEEE Robot. Autom. Lett. 4 (2) (2019) 1493–1500, http://dx.doi.org/10.1109/LRA.2019.2895110.

[20] G. Sibley, Relative bundle adjustment, Electron. Notes Theor. Comput. Sci. 220 (3) (2009) 1–26, http://dx.doi.org/10.1016/j.entcs.2008.11.016.

[21] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, P. Furgale, Keyframe-based visual–inertial odometry using nonlinear optimization, IJRR 34 (3) (2015) 314–334, http://dx.doi.org/10.1177/0278364914554813.

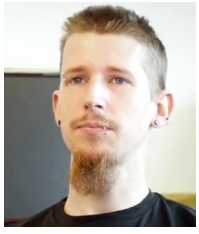
[22] J. Sola, T. Vidal-Calleja, J. Civera, J.M.M. Montiel, Impact of landmark parametrization on monocular EKF-SLAM with points and lines, Int. J. Comput. Vis. 97 (3) (2012) 339–368.

[23] N. De Palézieux, T. Nägeli, O. Hilliges, Duo-VIO: Fast, light-weight, stereo Inertial Odometry, IROS (2016) http://dx.doi.org/10.1109/IROS.2016.7759350.

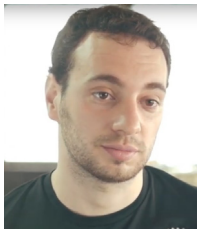
[24] J. Civera, O.G. Grasa, A.J. Davison, J. Montiel, 1-point RANSAC for extended kalman filtering: Application to real-time structure from motion and visual odometry, J. Field Robotics 27 (5) (2010) 609–631.

[25] N. Keivan, A. Patron-Perez, G. Sibley, Asynchronous adaptive conditioning for visual-inertial SLAM, STAR 109 (2016) 309–321, http://dx.doi.org/10.1007/978-3-319-23778-7_21.

[26] K. Eckenhoff, L. Paull, G. Huang, Decoupled, consistent node removal and edge sparsification for graph-based SLAM, IEEE International Conference on Intelligent Robots and Systems 2016-Novem (2016) 3275–3282, http://dx.doi.org/10.1109/IROS.2016.7759505.
[27] H. Liu, M. Chen, G. Zhang, H. Bao, Y. Bao, ICE-BA: Incremental, consistent and efficient bundle adjustment for visual-inertial SLAM, Cvpr (2018) 1974–1982, http://dx.doi.org/10.1109/CVPR.2018.00211.
[28] F.-A. Moreno, J.-L. Blanco, J. Gonzalez-Jimenez, A constant-time SLAM back-end in the continuum between global mapping and submapping: application to visual stereo SLAM, Int. J. Robot. Res. 35 (9) (2016) 1036–1056, http://dx.doi.org/10.1177/0278364915619238, arXiv:arXiv:1508.04886v1.
[29] H. Strasdat, J.M. Montiel, A.J. Davison, Visual SLAM: Why filter? Image Vis. Comput. 30 (2) (2012) 65–77, http://dx.doi.org/10.1016/j.imavis.2012.02.009.
[30] C. Forster, L. Carlone, F. Dellaert, D. Scaramuzza, On-manifold preintegration for real-time visual-inertial odometry, IEEE Trans. Robot. 33 (1) (2017) 1–21, http://dx.doi.org/10.1109/TRO.2016.2597321, arXiv:1512.02363.
[31] C. Forster, L. Carlone, F. Dellaert, D. Scaramuzza, Supplementary material to IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation, Robot.: Sci. Syst. XI (3) (2015) 1–10, http://dx.doi.org/10.15607/RSS.2015.XI.006, arXiv:1512.02363.
[32] J. Solà, J. Deray, D. Atchuthan, A micro Lie theory for state estimation in robotics, 2018, arXiv:1812.01537.
[33] T.D. Barfoot, State Estimation for Robotics, 12-08-2017 ed., 2017.
[34] C. Hertzberg, R. Wagner, U. Frese, L. Schröder, Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds, Inf. Fusion 14 (1) (2013) 57–77, http://dx.doi.org/10.1016/j.inffus.2011.08.003, arXiv:1107.1119.
[35] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, S. Shen, Autonomous aerial navigation using monocular visual-inertial fusion, J. Field Robotics 35 (1) (2018) 23–51, http://dx.doi.org/10.1002/rob.21732.
[36] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, R. Siegwart, Keyframe-based visual-inertial SLAM using nonlinear optimization, Proc. Robot.: Sci. Syst. (2013).
[37] Z. Zhang, G. Gallego, D. Scaramuzza, On the Comparison of Gauge Freedom Handling in Optimization-based Visual-Inertial State Estimation, 2018, pp. 1–8.
[38] R. Mur-Artal, J.D. Tardós, Fast Relocalisation and Loop Closing in Keyframe-Based SLAM, Technical Report.
[39] C. Richter, A. Bry, N. Roy, Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments, in: Springer Tracts in Advanced Robotics, vol. 114, 2016, pp. 649–666, http://dx.doi.org/10.1007/978-3-319-28872-7_37.
[40] T. Lee, M. Leok, N. McClamroch, Control of complex maneuvers for a quadrotor UAV using geometric methods on SE (3), 2010, pp. 1–32, http://dx.doi.org/10.1002/asjc.0000, arXiv preprint arXiv:1003.2005. arXiv:arXiv:1003.2005v4.
[41] D. Brescianini, M. Hehn, R. D'Andrea, Nonlinear Quadrocopter Attitude Control. Technical Report, Technical Report, Eidgenössische Technische Hochschule Zürich, Departement Maschinenbau und Verfahrenstechnik, Zürich, 2013, http://dx.doi.org/10.3929/ethz-a-009970340.
[42] H. Strasdat, S. Lovegrove, Sophus, 2019, https://github.com/strasdat/Sophus.
[43] S. Agarwal, K. Mierle, et al., Ceres solver, 2019, http://ceres-solver.org.
[44] C. Chatfield, Real-time feature extraction on the raspberry pi and other arm processors supporting NEON, 2018, https://github.com/0xfaded/pislam.
[45] Jianbo Shi, Tomasi, Good features to track, in: 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 593–600, http://dx.doi.org/10.1109/CVPR.1994.323794.
[46] F. Furrer, M. Burri, M. Achtelik, R. Siegwart, RotorS—A modular gazebo MAV simulator framework, Stud. Comput. Intell. 625 (2016) 595–625, http://dx.doi.org/10.1007/978-3-319-26054-9_23.
[47] Burri, et al., The EuRoC micro aerial vehicle datasets, Int. J. Robot. Res. 35 (10) (2016) 1157–1163.
[48] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: CVPR, IEEE, 2012, pp. 3354–3361, http://dx.doi.org/10.1109/CVPR.2012.6248074, arXiv:1612.07695.
[49] Open Source Robotics Foundation, DARPA Subterranean challenge, 2019, https://bitbucket.org/osrf/subt/wiki/Home.



Matías Nitsche received the Ph.D. degree in Computer Science in 2016 at the University of Buenos Aires, Argentina. He is currently a Research Assistant at the National Council of Scientific and Technological Research (CONICET), Argentina. His research is focused on solving localization and autonomous navigation of Unmanned Aerial Vehicles using stereo-vision and inertial sensors.



Facundo Pessacg was born in Rio Gallegos, Argentina, in 1990. He received his master degree in Physics in 2015 from the University of Buenos Aires, with a master thesis subject on industrial robots. He is currently a full time Ph.D. student at the University of Buenos Aires. His research interests include computer vision, visual active SLAM methods, loop closure and robotics.

Javier Civera was born in Barcelona, Spain, in 1980. He received the Industrial Engineering degree in 2004 and the Ph.D. degree in 2009, both from the University of Zaragoza in Spain. He is currently an Associate Professor at the University of Zaragoza, where he teaches courses in computer vision, machine learning and control engineering. He has participated in and leaded several EU-funded, national and technology transfer projects related with vision and robotics and has been funded for research visits to Imperial College (London) and ETH (Zürich). He has coauthored more than 45 publications in top conferences and journals, receiving more than 4,000 references (GoogleScholar). He has served as Associate Editor at IEEE T-ASE, IEEE RA-L, IEEE ICRA and IEEE/RSJ IROS. Currently, his research interests are in the use of multi-view geometry and machine learning to produce robust and real-time visual mapping technologies for robotics, wearables and AR applications.