High-precision, consistent EKF-based visual–inertial odometry

Mingyang Li and Anastasios I. Mourikis

The International Journal of Robotics Research 2013 32: 690
DOI: 10.1177/0278364913481251

The International Journal of
Robotics Research
32(6) 690–711

DOI: 10.1177/0278364913481251
ijr.sagepub.com

High-precision, consistent EKF-based visual-inertial odometry

Mingyang Li and Anastasios I. Mourikis

Abstract

In this paper, we focus on the problem of motion tracking in unknown environments using visual and inertial sensors. We term this estimation task visual–inertial odometry *(VIO), in analogy to the well-known visual-odometry problem. We present a detailed study of extended Kalman filter (EKF)-based VIO algorithms, by comparing both their theoretical properties and empirical performance. We show that an EKF formulation where the state vector comprises a sliding window of poses (the multi-state-constraint Kalman filter (MSCKF)) attains better accuracy, consistency, and computational efficiency than the simultaneous localization and mapping (SLAM) formulation of the EKF, in which the state vector contains the current pose and the features seen by the camera. Moreover, we prove that both types of EKF approaches are* inconsistent*, due to the way in which Jacobians are computed. Specifically, we show that the observability properties of the EKF's linearized system models* do not *match those of the underlying system, which causes the filters to underestimate the uncertainty in the state estimates. Based on our analysis, we propose a novel, real-time EKF-based VIO algorithm, which achieves consistent estimation by (i) ensuring the correct observability properties of its linearized system model, and (ii) performing online estimation of the camera-to-inertial measurement unit (IMU) calibration parameters. This algorithm, which we term MSCKF 2.0, is shown to achieve accuracy and consistency higher than even an iterative, sliding-window fixed-lag smoother, in both Monte Carlo simulations and real-world testing.*

Keywords

vision-aided inertial navigation, visual-inertial odometry, extended Kalman filter consistency, visual-inertial SLAM

1. Introduction

This paper addresses the problem of tracking a vehicle's egomotion in GPS-denied environments, using an inertial measurement unit (IMU) and a monocular camera. Our focus is on estimating the pose of a vehicle moving in an unknown environment. Therefore, we do not assume that a feature map is available in advance, as in map-based localization methods e.g., (Wu et al., 2005; Trawny et al., 2007). Moreover, we do not aim at building such a map. Our goal is to estimate the vehicle trajectory only, using the inertial measurements and the observations of naturally occurring features. This task is similar to the well-known visual-odometry (VO) problem (Nister et al., 2004), with the added characteristic that an IMU is available. We thus term the approach *visual–inertial odometry* (VIO). VIO methods have attracted significant research interest, because they can either be used as stand-alone pose-tracking methods, or as part of larger localization systems. For instance, VIO estimates can be integrated with a loop-closure detection module to provide long-term, bounded-uncertainty localization (Mourikis and Roumeliotis, 2008; Jones and Soatto, 2011).

A key requirement for VIO algorithms is that their computational complexity remains bounded, both as a function of time and as a function of the size of the trajectory. Most present-day algorithms in this class are either extended Kalman filter (EKF)-based methods (Mourikis and Roumeliotis, 2007; Jones and Soatto, 2011; Kelly and Sukhatme, 2011), or methods utilizing iterative minimization over a window of states (Konolige and Agrawal, 2008; Dong-Si and Mourikis, 2011; Konolige et al., 2011). The latter are generally considered to be more accurate, as they employ re-linearization at each iteration to better deal with nonlinearities. However, the need for multiple iterations also incurs a high computational cost, compared to EKF-based

Department of Electrical Engineering, University of California, Riverside, CA, USA

Corresponding author:
Anastasios I. Mourikis, Department of Electrical Engineering, University of California, Riverside, Suite 343, Winston Chung Hall, Riverside, CA 92521, USA.
Email: mourikis@ee.ucr.edu

methods. Ideally, one would like to obtain accuracy similar to, or better than, that of iterative-minimization algorithms, but at the computational cost of an EKF algorithm. In this paper, we show how this can be achieved. Specifically, we carry out an in-depth analysis of EKF-based VIO, based on which we develop a novel real-time EKF algorithm. Our results show that this algorithm is more accurate than both existing EKF alternatives *and* iterative-minimization VIO.

As a starting point for our analysis, we compare the performance of two families of EKF-based VIO estimators: simultaneous localization and mapping (SLAM) and sliding-window algorithms. In the former class of methods, the filter state vector contains the current IMU pose as well as the features visible by the camera (Pinies et al., 2007; Jones and Soatto, 2011; Kleinert and Schleith, 2010), while in the latter the state vector contains only a sliding window of poses, and the feature measurements are used to apply probabilistic constraints between them (Diel et al., 2005; Mourikis and Roumeliotis, 2007). Out of this second class of methods, we focus on the multi-state-constraint Kalman filter (MSCKF) algorithm (Mourikis and Roumeliotis, 2007), which we show to be the maximum *a posteriori* (MAP) estimator up to linearization.

In this paper we show, through extensive Monte Carlo simulations emulating real-world datasets, that the MSCKF algorithm outperforms EKF-SLAM methods by a wide margin, in terms of accuracy, consistency, and computational efficiency. We attribute this primarily to the fact that the MSCKF makes no Gaussianity assumptions on the pdf of the features' positions, something that is required in EKF-SLAM. Having shown the advantages of the MSCKF over EKF-SLAM methods, we then focus on analyzing and further improving its performance. Specifically, our approach relies on improving the *consistency* of the MSCKF, which, in turn, also improves the accuracy of the estimates. As defined in (Bar-Shalom et al., 2001, Section 5.4), a recursive estimator is consistent when the estimation errors are zero-mean and have covariance matrix equal to that reported by the estimator.

We identify and address two key causes of inconsistency in the MSCKF. The first cause is related to a fundamental shortcoming of the EKF: we prove that, due to the way the EKF Jacobians are computed, even though the IMU's rotation about gravity (the yaw) is *not* observable in VIO (see, e.g., (Jones and Soatto, 2011; Kelly and Sukhatme, 2011; Martinelli, 2012)), it *appears to be* observable in the linearized system model used by the MSCKF, and the same occurs in EKF-SLAM. Thus, the estimator erroneously believes it has more information than it actually does, and reports a covariance matrix for the state that underestimates the actual one. The second cause of inconsistency is that, in most practical cases, the extrinsic calibration parameters (rotation and translation) between the camera and IMU are only known with finite precision. If (as is common practice) these parameters are assumed to be perfectly known, the unmodeled uncertainty will result in under-reporting of the state estimates' covariance.

To improve the consistency of the MSCKF, we address the two problems identified above. First, we show that a modification in the way in which the filter Jacobians are computed can restore the appropriate observability properties for the filter's linearized system model. We note that, as part of this modification, we derive a closed-form expression for the IMU's error-state transition matrix. This expression can be used in any case in which an IMU is used for estimation (e.g. not only in VIO, but also in GPS-aided INS), and to the best of our knowledge this is the first time such an expression has been proposed. In addition, to address the uncertainty in the knowledge of the camera-to-IMU transformation, we include these parameters in the MSCKF's state vector, so that they can be estimated online, along with the IMU state.

We term the modified MSCKF algorithm, which ensures the correct observability properties of its linearized system model and performs online calibration of the camera-to-IMU transformation, MSCKF 2.0. Our simulation and experimental results demonstrate that this novel algorithm shows substantial improvement in consistency compared to all of the existing EKF alternatives. Moreover, the algorithm outperforms the alternatives in terms of *accuracy*, since a more accurate representation of the uncertainty results in better state updates. More importantly, however, our results show that the MSCKF 2.0 obtains higher consistency and accuracy even than a comparable algorithm that uses sliding-window iterative minimization, which has much higher computational cost. This indicates that having a linearized system model with appropriate observability properties may be more important than using relinearization to better approximate the nonlinear measurement models.

2. Related work

The simplest (and most computationally efficient) approaches to VIO are *loosely coupled*, i.e. methods that process the IMU and image measurements separately. For instance, some methods first process the images for computing relative motion estimates between consecutive poses, and subsequently fuse these with the inertial measurements (Roumeliotis et al., 2002; Diel et al., 2005; Tardif et al., 2010; Weiss and Siegwart, 2011; Ma et al., 2012). Alternatively, IMU measurements can be processed separately for extracting rotation estimates, and fused in an image-based estimation algorithm (Oskiper et al., 2007; Konolige et al., 2011; Brockers et al., 2012). Separately processing the two sources of information leads to a reduction in computational cost, and as a result loosely coupled methods are typically suited for systems with very limited resources, such as MAVs (Brockers et al., 2012). However, this comes at the expense of information loss: for instance, using feature measurements for egomotion

estimation between pairs of images ignores the correlations between consecutive timesteps (Mourikis et al., 2007), and processing IMU measurements separately does not allow for optimal estimation of sensor biases.

In this work, we are therefore interested in *tightly coupled* methods, which directly fuse the visual and inertial data, thus achieving higher precision. As mentioned previously, these are either based on iterative minimization over a sliding window of states, or are EKF formulations.[1] The former methods (e.g., (Oskiper et al., 2007; Konolige and Agrawal, 2008; Dong-Si and Mourikis, 2011; Konolige et al., 2011; Lupton and Sukkarieh, 2012)), essentially implement bundle adjustment in a sliding window of states (Engels et al., 2006) with the addition of IMU measurements. The need for multiple iterations during minimization results in increased computational cost, however. In this paper, we show that a properly designed EKF estimator can attain performance *higher* than that of iterative minimization, at only a fraction of the computation.

To fuse the visual and inertial measurements, the most commonly used tightly-coupled EKF estimator is EKF-based SLAM, in which the current camera pose and feature positions are jointly estimated (Kim and Sukkarieh, 2007; Pinies et al., 2007; Jones and Soatto, 2011; Kleinert and Schleith, 2010; Kelly and Sukhatme, 2011). To keep the computational cost bounded in EKF-SLAM algorithms, features that move out of the camera's field of view must be removed from the state vector (Munguia and Grau, 2007). One disadvantage of EKF-SLAM is its computational complexity, cubic in the number of features in the state vector. When many features are visible (the common situation in images of natural scenes), EKF-SLAM's runtime can be unacceptably high (in fact, higher than that of iterative minimization in certain cases (Strasdat et al., 2010)).

To address this problem, the MSCKF algorithm was proposed as an alternative EKF-based VIO method (Mourikis and Roumeliotis, 2007; Shkurti et al., 2011). In contrast to EKF-SLAM, the MSCKF maintains a sliding window of poses in its state vector, and uses the feature measurements to impose constraints on these poses. This results in a computational complexity that is linear in the number of features, and thus the MSCKF is faster than EKF-SLAM. In this paper, we compare the MSCKF's *accuracy* and *consistency* to those of EKF-SLAM methods, and show that the MSCKF outperforms EKF-SLAM in these respects as well.

A key contribution of this work is the analysis and improvement of the consistency of EKF-based vision-aided inertial navigation. Past work on the consistency of 3D vision-based localization has primarily focused on the parameterization of feature positions. (Civera et al., 2008) showed that the Cartesian-coordinate (*XYZ*) parametrization results in severely non-Gaussian probability density functions (pdfs) for the features, and degrades accuracy and consistency. Therefore, an inverse-depth feature parametrization was proposed, which is better suited for the camera measurement model, and results in improved performance. Sola (2010) proposed an anchored homogeneous feature parametrization that was shown to further improve the filter's consistency. In our work, we compare all of the above parameterizations in VIO and show that, while the parameterization of (Sola, 2010) yields superior results to the alternatives, its performance is still worse than that of the MSCKF algorithm.

In this work, we take a different approach to exploring the consistency properties of EKF-based VIO. Specifically, our approach is motivated by recent work examining the relationship between the observability properties of the EKF's linearized system model and the filter's consistency, in the context of 2D SLAM (Huang et al., 2008, 2010). These works showed that, due to the way in which Jacobians are computed in the EKF, the robot's orientation appears to be observable in the linearized system model, while it is not in the actual, nonlinear system. As a result of this mismatch, the filter produces too small estimates for the uncertainty of the orientation estimates, and becomes inconsistent.

Our analysis in Section 4 shows that the same problem affects EKF-based VIO in 3D. This result first appeared in an earlier conference version of this paper (Li and Mourikis, 2012a). Moreover, similar results were independently derived in subsequent papers by (Hesch et al., 2012; Kottas et al., 2012). Compared with our earlier work, we here additionally (i) compare the performance of the MSCKF to EKF-SLAM based methods, (ii) show that the same, erroneous observability properties affect EKF-SLAM approaches, (iii) address the issue of inconsistency caused by inaccurate knowledge of the camera-to-IMU calibration, and (iv) present additional, large-scale simulation and experimental results.

3. EKF-based VIO

Consider a mobile platform, equipped with an IMU and a camera, moving with respect to a global coordinate frame, $\{G\}$. Our goal is to perform VIO, i.e. to track the position and orientation of the platform using inertial measurements and observations of naturally occurring point features, whose positions are not known *a priori*. To this end, we affix a coordinate frame $\{I\}$ to the IMU, and track the motion of this frame with respect to the global frame. In what follows, we first describe the parameterization we employ for the IMU state, and then discuss the two alternative tightly-coupled EKF formulations for VIO, and compare their performance.

3.1. IMU state parameterization

Following standard practice, the IMU state vector at time step ℓ is defined as the 16×1 vector:[2]

$$\mathbf{x}_{I_\ell} = \begin{bmatrix} {}^{I_\ell}_{G}\bar{\mathbf{q}}^{\mathrm{T}} & \mathbf{p}_\ell^{\mathrm{T}} & \mathbf{v}_\ell^{\mathrm{T}} & \mathbf{b}_{\boldsymbol{g}_\ell}^{\mathrm{T}} & \mathbf{b}_{\boldsymbol{a}_\ell}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{1}$$

where ${}^{I_\ell}_{G}\bar{\mathbf{q}}$ is the unit quaternion (Trawny and Roumeliotis, 2005) representing the rotation from the global frame to the IMU frame at time step ℓ, ${}^{G}\mathbf{p}_\ell$ and ${}^{G}\mathbf{v}_\ell$ are the IMU position

and velocity in the global frame, and $\mathbf{b}_{g_\ell}$ and $\mathbf{b}_{a_\ell}$ are the gyroscope and accelerometer biases.

To define the IMU error state, we use the standard additive error for the position, velocity, and biases (e.g. ${}^G\tilde{\mathbf{p}} = {}^G\mathbf{p} - {}^G\hat{\mathbf{p}}$). For the orientation error, out of the several possible options that exist, the preferable ones are those that (i) are local, so that singularities are avoided, and (ii) use a minimal, three-dimensional representation of the orientation error. To obtain a local parameterization, we define the orientation error based on the quaternion $\delta\bar{\mathbf{q}}$ that describes the difference between the true and estimated orientation. Specifically, we define

$$ {}^I_G\bar{\mathbf{q}} = {}^I_G\hat{\bar{\mathbf{q}}} \otimes \delta\bar{\mathbf{q}} \Rightarrow \delta\bar{\mathbf{q}} = {}^I_G\hat{\bar{\mathbf{q}}}^{-1} \otimes {}^I_G\bar{\mathbf{q}} \tag{2} $$

where $\otimes$ denotes quaternion multiplication. Intuitively, $\delta\bar{\mathbf{q}}$, is the (small) rotation that is needed to bring the estimated global frame to match the true one. To obtain a minimal representation for this rotation, we note that $\delta\bar{\mathbf{q}}$ can be written as

$$ \delta\bar{\mathbf{q}} = \begin{bmatrix} \frac{1}{2}{}^G\tilde{\boldsymbol{\theta}} \\ \sqrt{1 - \frac{1}{4}{}^G\tilde{\boldsymbol{\theta}}^{\mathrm{T}}{}^G\tilde{\boldsymbol{\theta}}} \end{bmatrix} \simeq \begin{bmatrix} \frac{1}{2}{}^G\tilde{\boldsymbol{\theta}} \\ 1 \end{bmatrix} \tag{3} $$

where ${}^G\tilde{\boldsymbol{\theta}}$ is a 3×1 vector describing the orientation errors about the three axes. With the above error definition, the IMU error state is defined as the 15×1 vector:

$$ \tilde{\mathbf{x}}_I = \begin{bmatrix} {}^G\tilde{\boldsymbol{\theta}}^{\mathrm{T}} & {}^G\tilde{\mathbf{p}}^{\mathrm{T}} & {}^G\tilde{\mathbf{v}}^{\mathrm{T}} & \tilde{\mathbf{b}}_{\mathbf{g}}^{\mathrm{T}} & \tilde{\mathbf{b}}_{\mathbf{a}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{4} $$

It is worth pointing out that our choice of the orientation-error parameterization is guided by the analysis of (Li and Mourikis, 2012a). That analysis showed that defining the orientation error based on the difference between the true and estimated *global* frame, as in (2)–(3), is preferable to defining it as the difference between the true and estimated *IMU* frame. Specifically, the latter choice (used, for example, by (Mourikis and Roumeliotis, 2007)) causes undesirable terms to appear in the observability matrix of the EKF's linearized system model.

3.2. EKF-based SLAM

In EKF-SLAM algorithms, the filter state vector contains the current IMU state, $\mathbf{x}_{I_\ell}$, and a representation of the feature positions. Thus, the filter state vector at time-step ℓ is defined as

$$ \mathbf{x}_\ell = \begin{bmatrix} \mathbf{x}_{I_\ell}^{\mathrm{T}} & \mathbf{f}_1^{\mathrm{T}} & \cdots & \mathbf{f}_{n_\ell}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{5} $$

where $\mathbf{f}_i$, $i = 1, \ldots, n_\ell$ are the features included in the state vector at time step ℓ. These could be parameterized in different ways. In this paper, we will consider the "traditional" *XYZ* coordinate parametrization, the inverse-depth parametrization (Civera et al., 2008), and the anchored homogeneous parametrization (Sola, 2010). These are the most commonly used parameterizations in practice, and the two latter ones specifically aim at increasing the filter's consistency and accuracy.

The EKF-SLAM equations are well known, and we therefore only briefly describe them here, to introduce the necessary notation. In standard practice, the IMU measurements are used to propagate the IMU state. To describe the way in which the errors in the propagated state estimate depend on the estimation errors at the previous time step, the EKF employs a linearized equation of the form:

$$ \tilde{\mathbf{x}}_{I_{\ell+1|\ell}} \simeq \boldsymbol{\Phi}_{I_\ell} \tilde{\mathbf{x}}_{I_{\ell|\ell}} + \mathbf{w}_{d_\ell} \tag{6} $$

where $\boldsymbol{\Phi}_I$ is the IMU error-state transition matrix, and $\mathbf{w}_{d_\ell}$ is a noise vector, with covariance matrix $\mathbf{Q}_{d_\ell}$. The filter's covariance matrix is also propagated according to:

$$ \mathbf{P}_{\ell+1|\ell} = \begin{bmatrix} \boldsymbol{\Phi}_{I_\ell}\mathbf{P}_{II_{\ell|\ell}}\boldsymbol{\Phi}_{I_\ell}^{\mathrm{T}} + \mathbf{Q}_{d_\ell} & \boldsymbol{\Phi}_{I_\ell}\mathbf{P}_{IF_{\ell|\ell}} \\ \mathbf{P}_{IF_{\ell|\ell}}^{\mathrm{T}}\boldsymbol{\Phi}_{I_\ell}^{\mathrm{T}} & \mathbf{P}_{FF_{\ell|\ell}} \end{bmatrix} $$

where $\mathbf{P}_{II_{\ell|\ell}}$ is the covariance matrix of the IMU state, $\mathbf{P}_{FF_{\ell|\ell}}$ is the covariance matrix of the features, and $\mathbf{P}_{IF_{\ell|\ell}}$ the cross-covariance between them.

Assuming a calibrated perspective camera, the observation of feature i at time step ℓ is described by the equation:[3]

$$ \mathbf{z}_{i\ell} = \mathbf{h}(\mathbf{x}_{I_\ell}, \mathbf{f}_i) + \mathbf{n}_{i\ell} = \begin{bmatrix} \frac{{}^{C_\ell}x_{f_i}}{{}^{C_\ell}z_{f_i}} \\ \frac{{}^{C_\ell}y_{f_i}}{{}^{C_\ell}z_{f_i}} \end{bmatrix} + \mathbf{n}_{i\ell} \tag{7} $$

where $\mathbf{n}_{i\ell}$ is the measurement noise vector, modeled as zero-mean Gaussian with covariance matrix $\sigma^2\mathbf{I}_2$, and the vector ${}^{C_\ell}\mathbf{p}_{f_i} = [{}^{C_\ell}x_{f_i} \quad {}^{C_\ell}y_{f_i} \quad {}^{C_\ell}z_{f_i}]^{\mathrm{T}}$ is the position of the feature with respect to the camera at time step ℓ:

$$ {}^{C_\ell}\mathbf{p}_{f_i} = {}^C_I\mathbf{R}\,{}^{I_\ell}_G\mathbf{R}\left({}^G\mathbf{p}_{f_i} - {}^G\mathbf{p}_{I_\ell}\right) + {}^C\mathbf{p}_I \tag{8} $$

with $\{{}^C_I\mathbf{R}, {}^C\mathbf{p}_I\}$ being the rotation and translation between the camera and the IMU. In EKF-SLAM, the feature observations are used directly for updating the state estimates. For this process, we employ the residual between the actual and expected feature measurement, and its linearized approximation:

$$ \begin{aligned} \mathbf{r}_{i\ell} &= \mathbf{z}_{i\ell} - \mathbf{h}(\hat{\mathbf{x}}_{I_{\ell|\ell-1}}, \hat{\mathbf{f}}_{i_{\ell|\ell-1}}) \\ &\simeq \mathbf{H}_{i\ell}\left(\hat{\mathbf{x}}_{\ell|\ell-1}\right)\tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{n}_{i\ell} \end{aligned} \tag{9} $$

where $\mathbf{H}_{i\ell}(\hat{\mathbf{x}}_{\ell|\ell-1})$ is the Jacobian matrix of $\mathbf{h}$ with respect to the filter state, evaluated at the state estimate $\hat{\mathbf{x}}_{\ell|\ell-1}$. This is a sparse matrix, containing non-zero blocks only at the locations corresponding to the IMU state (position and orientation) and the ith feature:

$$ \mathbf{H}_{i\ell}(\hat{\mathbf{x}}_{\ell|\ell-1}) = [\mathbf{H}_{I_{i\ell}}(\hat{\mathbf{x}}_{\ell|\ell-1}) \;\; \mathbf{0} \cdots \mathbf{H}_{\mathbf{f}_{i\ell}}(\hat{\mathbf{x}}_{\ell|\ell-1}) \cdots \mathbf{0}] \tag{10} $$

Once $\mathbf{r}_{i\ell}$ and $\mathbf{H}_{i\ell}$ have been computed, a Mahalanobis gating test is performed, and if successful the standard EKF update

equations are employed (Maybeck, 1982). Depending on the particular feature parameterization used, the exact form of the above Jacobians, as well as the "bookkeeping" required in the filter, will be slightly different.

In VIO, we must ensure that the computational cost of the algorithm remains bounded. To achieve this, features are removed from the state vector immediately once they leave the field of view of the camera. This of course means that the filter cannot process feature re-observations that occur when an area is re-visited. However, such "loop-closure" events do not need to be handled by a VIO algorithm: if desired, they can be handled by a separate algorithm running in parallel, as done for example in (Mourikis and Roumeliotis, 2008). In the "prototypical" VIO scenario, where the camera keeps moving in new areas, the EKF-SLAM algorithm described above will use all of the available feature information.

3.3. MSCKF

In contrast to EKF-SLAM, the MSCKF is an EKF algorithm that maintains in its state vector a sliding window of poses, and uses feature observations to impose probabilistic constraints between these poses (Mourikis and Roumeliotis, 2007). The state vector of the MSCKF at time step ℓ is defined as

$$\mathbf{x}_\ell = \begin{bmatrix} \mathbf{x}_{I_\ell}^{\mathrm{T}} & \boldsymbol{\pi}_{\ell-1}^{\mathrm{T}} & \boldsymbol{\pi}_{\ell-2}^{\mathrm{T}} & \cdots & \boldsymbol{\pi}_{\ell-N}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{11}$$

where $\boldsymbol{\pi}_i = [{}^{I_i}_{G}\bar{\mathbf{q}}^{\mathrm{T}} \quad {}^{G}\mathbf{p}_i^{\mathrm{T}}]^{\mathrm{T}}$, for $i = \ell - N, \ldots, \ell - 1$, are the IMU poses at the times the last N images were recorded.

During MSCKF propagation, the IMU measurements are used to propagate the IMU state estimate and the filter covariance matrix, similarly to EKF-SLAM. The difference lies in the way the feature measurements are used. Specifically, every time a new image is recorded by the camera, the MSCKF state and covariance are augmented with a copy of the current IMU pose, and the image is processed to extract and match features. Each feature is tracked until all of its measurements become available (e.g. until it goes out of the field of view), at which time an update is carried out using *all* of the measurements simultaneously.

To present the update equations, we consider the case where the feature f_i, observed from the N poses in the MSCKF state vector, is used for an update at time step ℓ. The first step of the process is to obtain an estimate of the feature position, ${}^{G}\hat{\mathbf{p}}_{f_i}$. To this end, we use all of the feature's measurements to estimate its position via Gauss–Newton minimization (Mourikis and Roumeliotis, 2007). Subsequently, we compute the residuals (for $j = \ell - N, \ldots, \ell - 1$):

$$\begin{aligned} \mathbf{r}_{ij} &= \mathbf{z}_{ij} - \mathbf{h}(\hat{\boldsymbol{\pi}}_{j|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i}) \qquad (12) \\ &\simeq \mathbf{H}_{\boldsymbol{\pi}_{ij}}(\hat{\boldsymbol{\pi}}_{j|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\,\tilde{\boldsymbol{\pi}}_{j|\ell-1} + \mathbf{H}_{\mathbf{f}_{ij}}(\hat{\boldsymbol{\pi}}_{j|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\,{}^{G}\tilde{\mathbf{p}}_{f_i} \\ &\quad + \mathbf{n}_{ij} \qquad (13) \end{aligned}$$

where $\tilde{\boldsymbol{\pi}}_{j|\ell-1}$ and ${}^{G}\tilde{\mathbf{p}}_{f_i}$ are the error of the current estimate for the jth pose and the error in the feature position respectively, and the matrices $\mathbf{H}_{\boldsymbol{\pi}_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$ are the corresponding Jacobians, evaluated using $\hat{\boldsymbol{\pi}}_{j|\ell-1}$ and ${}^{G}\hat{\mathbf{p}}_{f_i}$. At this point we note that, in the EKF algorithm, to be able to employ a measurement residual, $\mathbf{r}$, for a filter update, we must be able to write this residual in the form $\mathbf{r} \simeq \mathbf{H}\tilde{\mathbf{x}} + \mathbf{n}$, where $\tilde{\mathbf{x}}$ is the error in the state estimate, and $\mathbf{n}$ is a noise vector that is *independent* from $\tilde{\mathbf{x}}$. The residual in (13) does not have this form, as the feature position error ${}^{G}\tilde{\mathbf{p}}_{f_i}$ is correlated to both $\tilde{\boldsymbol{\pi}}_{j|\ell-1}$ and $\mathbf{n}_{ij}$ (this is because ${}^{G}\hat{\mathbf{p}}_{f_i}$ is computed as a function of $\hat{\boldsymbol{\pi}}_{j|\ell-1}$ and $\mathbf{z}_{ij}$, $j = \ell - N, \ldots, \ell - 1$). Therefore, in the MSCKF we proceed to remove ${}^{G}\tilde{\mathbf{p}}_{f_i}$ from the residual equations. For this purpose, we first form the vector containing the N residuals from all of the feature's measurements:

$$\mathbf{r}_i \simeq \mathbf{H}_{\boldsymbol{\pi}_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\,\tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{H}_{\mathbf{f}_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\,{}^{G}\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_i \tag{14}$$

where $\mathbf{r}_i$ and $\mathbf{n}_i$ are block vectors with elements $\mathbf{r}_{ij}$ and $\mathbf{n}_{ij}$, respectively, and $\mathbf{H}_{\boldsymbol{\pi}_i}$ and $\mathbf{H}_{\mathbf{f}_i}$ are matrices with block rows $\mathbf{H}_{\boldsymbol{\pi}_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$. Subsequently, we define the residual vector $\mathbf{r}_i^o = \mathbf{V}_i^{\mathrm{T}}\mathbf{r}_i$, where $\mathbf{V}_i$ is a matrix whose columns form a basis for the left nullspace of $\mathbf{H}_{\mathbf{f}_i}$. From (14), we obtain

$$\mathbf{r}_i^o = \mathbf{V}_i^{\mathrm{T}}\mathbf{r}_i \simeq \mathbf{H}_i^o(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\,\tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{n}_i^o \tag{15}$$

where $\mathbf{H}_i^o = \mathbf{V}_i^{\mathrm{T}}\mathbf{H}_{\boldsymbol{\pi}_i}$ and $\mathbf{n}_i^o = \mathbf{V}_i^{\mathrm{T}}\mathbf{n}_i$. Note that the residual vector $\mathbf{r}_i^o$ is now *independent* of the errors in the feature coordinates, and thus can be used for an EKF update. It should also be mentioned that, for efficiency, $\mathbf{r}_i^o$ and $\mathbf{H}_i^o$ are computed without explicitly forming $\mathbf{V}_i$ (Mourikis and Roumeliotis, 2007).

Once $\mathbf{r}_i^o$ and $\mathbf{H}_i^o$ are computed, we proceed to carry out a Mahalanobis gating test for the residual $\mathbf{r}_i^o$. Specifically, we compute

$$\gamma_i = (\mathbf{r}_i^o)^{\mathrm{T}}\left(\mathbf{H}_i^o\mathbf{P}_{\ell|\ell-1}(\mathbf{H}_i^o)^{\mathrm{T}} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{r}_i^o \tag{16}$$

and compare it against a threshold given by the 95th percentile of the χ^2 distribution with $2N - 3$ degrees of freedom ($2N-3$ is the number of elements in the residual vector $\mathbf{r}_i^o$). If the feature passes the test, we proceed to use $\mathbf{r}_i^o$ for an EKF update, together with the residuals of all other features that pass the gating test. After this update takes place, we remove from the sliding window those poses whose features have all been used for updates. An overview of the MSCKF is given in Algorithm 1.

3.4. Comparison of the MSCKF and EKF-SLAM approaches

We now compare the two VIO methods discussed in the preceding subsections. We start by noting that the MSCKF and EKF-SLAM make use of the *same* measurement information. Specifically, in Appendix B we prove that if the system

Algorithm 1 Multi-state-constraint Kalman filter (MSCKF)

Propagation: Propagate state vector and covariance matrix using IMU readings.

Update: when a new image is recorded,

- State augmentation: augment the state vector and state covariance matrix with the current IMU position and orientation.
- Image processing: extract corner features and perform feature matching.
- Update: for each feature whose track is complete, compute $\mathbf{r}_i^o$ and $\mathbf{H}_i^o$, and perform the Mahalanobis test. Use all features that pass the test for an EKF update.
- State management: remove from the state vector those IMU states for which all associated features have been processed.

model was linear-Gaussian, the MSCKF estimate for the current IMU pose would be the optimal MAP estimate. In the linear-Gaussian case, EKF-SLAM also yields the MAP estimate, since the Kalman filter is a MAP estimator (Kay, 1993). Thus, if the system models were linear-Gaussian, the EKF-SLAM and the MSCKF algorithms would yield the *same, optimal* estimates for the IMU pose. The differences in their performance arise due to fact that the actual measurement models are not linear, as discussed in what follows.

To test the performance of the methods with the actual nonlinear system models, we performed extensive Monte Carlo simulations. To obtain realistic simulation environments, we generated the simulation data based on real-world datasets. For each dataset, we first generated the platform angular velocity and linear acceleration by differentiating the ground-truth estimates obtained by high-precision GPS-INS during the actual experiment. Using this angular velocity and linear acceleration, we subsequently generated (i) the "ground truth" trajectory (position, velocity, orientation) by re-integration, and (ii) IMU measurements corrupted with noises and biases with characteristics identical to those of the actual sensors. In each Monte Carlo run, different realizations of the sensor noises and biases (which are modeled as white, zero-mean Gaussian noise and Gaussian random walk processes, respectively) were generated.

The feature tracks in the simulations were also generated with statistical characteristics matching those of the actual datasets. Specifically, by processing the images collected in the actual dataset, we modeled the distributions of (i) the number of features per image, (ii) the feature-track lengths, and (iii) the feature distance to the camera, at different parts of the trajectory. Then, in each simulation, feature tracks were generated by randomly sampling from these distributions, and the image measurements were corrupted by white, Gaussian noise, with standard deviation of one pixel, similarly to the actual data. In this way, the platform trajectory as well as the IMU and camera measurements have properties emulating those of a real dataset, which provides for more realistic testing.

For the results presented here, we used a 13 minute, 5.5 km long dataset, as the basis for the simulation. The dataset was collected with an ISIS IMU and a monocular camera, mounted on top of a vehicle driving in an urban environment. In each image, 100 features were tracked, on average, and the average track length was 4.5 frames (the feature-track distribution is similar to an exponential one, with many features tracked over short periods, and fewer features tracked longer). The algorithms compared are (i) the MSCKF, (ii) EKF-SLAM with inverse depth parametrization (IDP) (Civera et al., 2008), (iii) EKF-SLAM with the anchored homogeneous feature parametrization (AHP) (Sola, 2010), and (iv) EKF-SLAM with the "traditional" *XYZ* feature parametrization (*XYZ*). Our goal in these simulations is to examine both the accuracy and the consistency of the algorithms. Therefore, we collected statistics for the average normalized estimation error squared (NEES) for the IMU pose (position and orientation), as well as the root mean squared error (RMSE) for the IMU position and orientation. We note that for a consistent estimator the average pose NEES should be six (equal to the dimension of the pose error), while a larger NEES value indicates inconsistency.

In all of the SLAM algorithms, we wait until κ observations of a feature are available, prior to initializing it in the state vector. For this purpose, we maintain a sliding window of κ poses in the state vector, and when a feature has been observed κ times, all of the measurements of the feature are used concurrently to initialize the feature and its covariance. In our tests, we used the values $\kappa = 2$, $\kappa = 4$, and $\kappa = 6$. Even though for the IDP and AHP approaches it is not necessary to use multiple observations for initialization, our results show that this results in dramatically improved performance. We note that if a feature's track ends after fewer than κ observations, its measurements are processed with the MSCKF measurement model instead. In this way no measurements are discarded, and all the algorithms compared use the same feature observations for fairness.

Figure 1 shows the average NEES for the IMU pose, as well as the RMSE for the IMU position and orientation, averaged over 50 Monte Carlo trials. This plot corresponds to the case $\kappa = 4$ for the SLAM methods. Moreover, Table 1 provides the average NEES and RMSE values for all of the algorithms, and with different values of κ for the SLAM methods. Several interesting conclusions can be drawn from these results. The most important one is that the MSCKF algorithm outperforms all three EKF-SLAM VIO formulations, both in terms of accuracy (smaller RMSE) and in

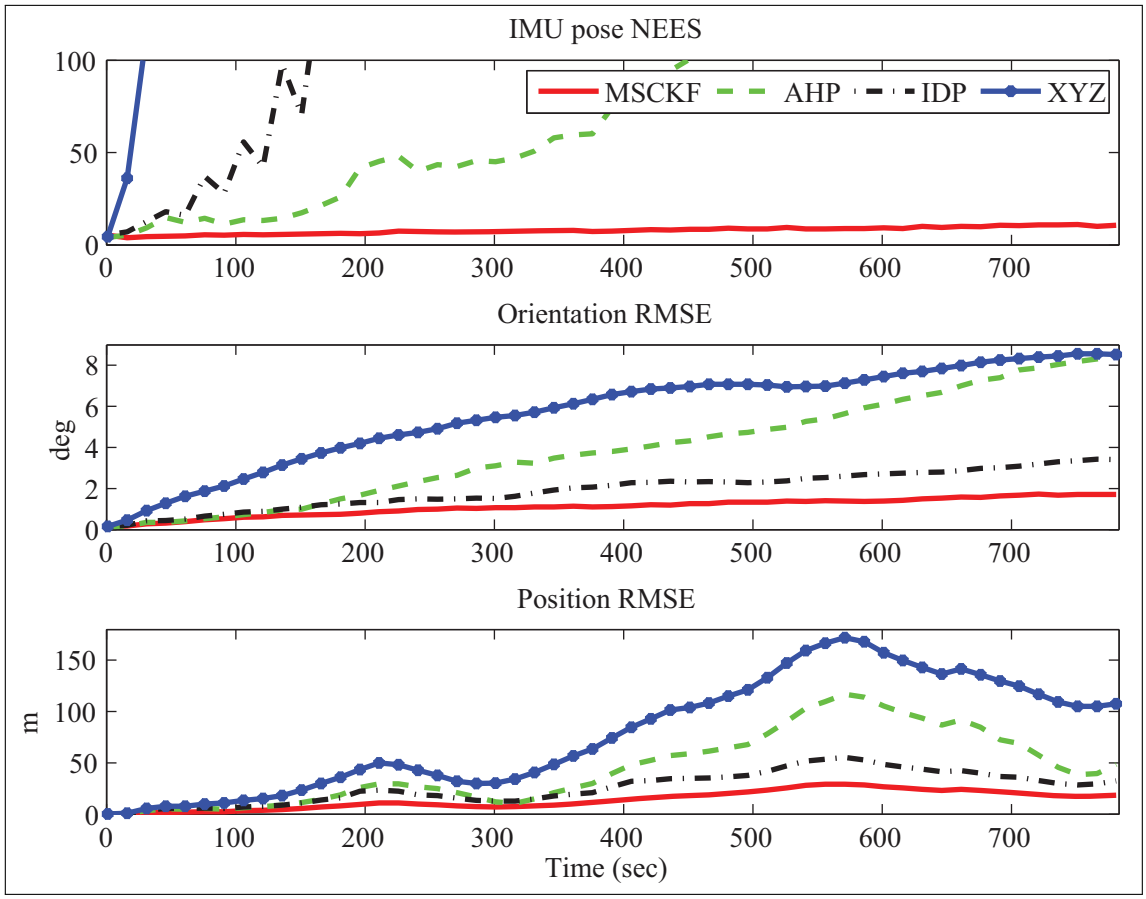

Fig. 1. The average NEES of the IMU pose and RMSE of the IMU position and orientation. The different lines correspond to the MSCKF (red solid line), the AHP (green dashed line), the IDP (black dashed-dotted line), and the *XYZ* (blue dotted line). Note that due to their large values, the curves for some of the EKF-SLAM methods are not fully visible. Table 1 presents the statistics for all curves.

Table 1. Simulation results: Performance metrics (IMU pose NEES, orientation RMSE, and position RMSE) for the MSCKF and the three EKF-SLAM algorithms with varying number of observations used for initialization. The values are averages over all Monte Carlo trials and all time steps (κ).

	$\kappa = 2$			$\kappa = 4$			$\kappa = 6$		
Algorithm	Position RMSE (m)	Orientation RMSE (°)	NEES	Position RMSE (m)	Orientation RMSE (°)	NEES	Position RMSE (m)	Orientation RMSE (°)	NEES
XYZ	N/A	N/A	N/A	78.447	5.609	4.9×10^3	53.469	3.974	1.3×10^3
IDP	69.502	3.731	2205.101	26.193	1.916	268.141	22.878	1.803	167.261
AHP	67.061	4.795	273.247	52.355	4.531	129.602	36.858	3.129	48.236
				Position RMSE (m)	Orientation RMSE (°)	NEES			
MSCKF				14.401	1.102	7.741			

terms of consistency (NEES closer to six). This is a result that we have consistently observed in all our tests, and that we attribute to two main reasons:

- First, all EKF-SLAM algorithms assume that the errors of the IMU state *and* feature positions are jointly Gaussian at each time step. However, due to the nonlinearity of the camera measurement model, this is not a good approximation, particularly for the *XYZ* parameterization (Civera et al., 2008). By intelligently choosing the feature parameterization, as AHP and IDP do, the accuracy and consistency of EKF-SLAM can be improved, as shown in these results. However, these algorithms still perform significantly worse than the MSCKF. Since

in the MSCKF the features are never included in the state vector, no assumptions on the feature errors' pdf are needed, thus avoiding a major source of inaccuracy.
- In EKF-SLAM, feature measurements are linearized and processed at *each* time step. By contrast, the MSCKF employs a "delayed linearization" approach: it processes each feature only when *all* of its measurements become available. This means that more accurate feature estimates are used in computing Jacobians, leading to more precise calculation of the Kalman gain and state corrections, and ultimately better accuracy.

Examining the different EKF-SLAM methods, we see that in accordance with previous results (Civera et al., 2008; Sola, 2010), the performance of the AHP and IDP parameterizations is significantly better than that of the *XYZ* parameterization. It should be mentioned that, for the *XYZ* parameterization, initializing features after only two observations is extremely unreliable: the estimator always fails, which does not allow us to obtain reliable statistics. This failure is characterized by a sequence of timesteps in which the filter corrections are very large (and erroneous), after which all residuals fail the Mahalanobis test, no filter updates occur, and the estimation errors increase rapidly. In fact, even if more observations are used for feature initialization, the *XYZ* parameterization still remains unreliable: for instance, when $\kappa = 4$, the EKF-SLAM with *XYZ* parametrization fails in 4% of the trials, if far-away features are discarded, and in approximately 70% of the trials if all features are kept. In the statistics reported in Table 1, only successful trials are used, to provide more meaningful statistics. Note that no failures were observed in the IDP SLAM, AHP SLAM, or MSCKF algorithms.

Moreover, we can observe that the use of more measurements for feature initialization (larger κ) leads to better performance, for all EKF-SLAM algorithms. The improvement as κ increases occurs because with more measurements, a better initial estimate for the feature is obtained, and thus the filter Jacobians become more accurate and the feature pdf closer to a Gaussian. Moreover, as κ increases, more features are in fact processed with the MSCKF measurement model, as a larger percentage of features is seen fewer than κ times. In this test, for example, when $\kappa = 6$ more than 50% of features are processed by the MSCKF update equations. Thus, as κ increases the EKF-SLAM algorithms essentially become "hybrids" between MSCKF and EKF-SLAM (Williams et al., 2011; Li and Mourikis, 2012b), and their performance approaches that of the pure MSCKF method.

In addition to the algorithms' estimation performance, it is also important to examine the computational efficiency of the different methods. For the tests performed above, the MSCKF's average runtime was 0.93 ms per update, while for the EKF-SLAM methods the average runtime was 1.54 ms for *XYZ*, 3.28 ms for IDP, and 4.45 ms for AHP, when $\kappa = 2$ (measured on a Core i7 processor at 2.66 GHz, with a single-threaded C++ implementation). These observed runtimes agree with the theoretical computational complexity of the algorithms: the MSCKF's computational cost per time step is *linear* in the number of features, as opposed to *cubic* for EKF-SLAM. Thus, we can conclude that due to the higher accuracy, consistency, robustness, and computational efficiency, the MSCKF is preferable to EKF-SLAM algorithms for VIO applications.

4. EKF consistency and relation to observability

In Table 1 we can see that the average IMU-pose NEES for the MSCKF in the simulation tests is 7.741, i.e. above the theoretically expected value of six for a consistent estimator. Moreover, Figure 1 shows that the NEES is gradually increasing over time, reaching an average of 10.6 in the last 100 s. These results show that MSCKF becomes *inconsistent*, albeit much less so than EKF-SLAM methods. This inconsistency can become significant in long trajectories, as demonstrated in the results of Section 8.2. In the remainder of the paper we focus on explaining the cause of the inconsistency, and proposing a solution to it, by examining the linearized system's observability properties.

To illustrate the main idea of our approach, consider a physical system described by general nonlinear models:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) + \mathbf{w} \tag{17}$$

$$\mathbf{z} = h(\mathbf{x}) + \mathbf{n} \tag{18}$$

where $\mathbf{x}$ is the system state, $\mathbf{u}$ is the control input, $\mathbf{z}$ are the measurements, and finally $\mathbf{w}$ and $\mathbf{n}$ are noise processes. To track the state vector $\mathbf{x}$ on a digital computer, discretization of the continuous-time system model is necessary. Furthermore, when an EKF is used for estimation, the filter equations (e.g. covariance propagation and update, gain computation) rely on a linearized version of the discrete-time model, which has the general form:

$$\tilde{\mathbf{x}}_{\ell+1} \simeq \mathbf{\Phi}_\ell \tilde{\mathbf{x}}_\ell + \mathbf{w}_\ell \tag{19}$$

$$\tilde{\mathbf{r}}_\ell \simeq \mathbf{H}_\ell \tilde{\mathbf{x}}_\ell + \mathbf{n}_\ell \tag{20}$$

where $\mathbf{\Phi}_\ell$ and $\mathbf{H}_\ell$ denote the error-state transition matrix and the measurement Jacobian matrix, respectively. Since the EKF equations are derived based on the linearized system model in (19)–(20), the observability properties of this model play a crucial role in determining the performance of the estimator. Ideally, one would like these properties to match those of the actual, nonlinear system in (17)–(18): if a certain quantity is unobservable in the actual system, its error should also be unobservable in the linearized model. If this is not the case, "fictitious" information about this quantity will be accumulated by the EKF, which will lead to inconsistency.

The observability properties of the nonlinear system in visual–inertial navigation have recently been studied in (Jones and Soatto, 2011; Kelly and Sukhatme, 2011;

Martinelli, 2012). Based on the analysis of these papers, it is now known that when a camera/IMU system moves in a general trajectory, in an environment with a known gravitational acceleration but *no* known features, four degrees of freedom are unobservable: (i) the three corresponding to the global position, and (ii) one corresponding to the rotation about the gravity vector (i.e. the yaw). In Section 6, we analyze the observability properties of the linearized system model employed in EKF-based VIO, by studying the nullspace of the observability matrix associated with (19)–(20):

$$\mathcal{O} \triangleq \begin{bmatrix} \mathbf{H}_k \\ \mathbf{H}_{k+1}\boldsymbol{\Phi}_k \\ \vdots \\ \mathbf{H}_{k+m}\boldsymbol{\Phi}_{k+m-1}\cdots\boldsymbol{\Phi}_k \end{bmatrix} \tag{21}$$

The nullspace of $\mathcal{O}$ describes the directions of the state space for which no information is provided by the measurements in the time interval $[k, k+m]$, i.e., the unobservable directions. For the linearized system to have observability properties analogous to the actual, nonlinear system, this nullspace should be of dimension four, and should contain the vectors corresponding to global translation and to rotation about gravity. However, our key result, proven in Section 6, is that this is not the case when the MSCKF (or an EKF-SLAM method) is employed for VIO. Specifically, due to the way in which the Jacobians are computed in the EKF, the global orientation *appears* to be observable in the linearized model, while it is not in the actual system. As a result of this mismatch, the filter produces too small values for the state covariance matrix (i.e. the filter becomes *inconsistent*), and this in turn degrades accuracy.

Note that, to study the nullspace of the matrix $\mathcal{O}$ in (21) for the VIO system, we must have an expression for the error-state transition matrix $\boldsymbol{\Phi}_\ell$, for $\ell = k, \ldots, k+m-1$. In turn, this requires an expression for the IMU's error-state transition matrix, defined in (6). Therefore, before proceeding with the observability analysis, we must derive an expression for this matrix. This is the focus of the next section.

5. Computing the IMU error-state transition matrix

Most existing methods for computing $\boldsymbol{\Phi}_I$ stem from the integration of the differential equation $\dot{\boldsymbol{\Phi}}_I(t, t_\ell) = \mathbf{F}(t)\,\boldsymbol{\Phi}_I(t, t_\ell)$, where $\mathbf{F}(t)$ is the Jacobian of the continuous-time system model of the IMU motion (Trawny and Roumeliotis, 2005). For instance, (Mourikis and Roumeliotis, 2007 and Tardif et al., 2010) employ Runge–Kutta numerical integration, (Weiss and Siegwart, 2011; Weiss et al, 2012) use a closed-form, approximate solution to the differential equation, while several algorithms (especially in the GPS-aided inertial navigation community) employ the simple approximation $\boldsymbol{\Phi}_I \simeq \mathbf{I} + \mathbf{F}\Delta t$ that is equivalent to using one-step Euler integration (e.g. (Foxlin, 2005; Farrell, 2008; Zachariah and Jansson, 2010; Lupton and Sukkarieh, 2012; Vu et al., 2012)). All of these methods for computing $\boldsymbol{\Phi}_I$ have the disadvantage that, being numerical in nature, they are not amenable to theoretical analysis. More importantly, however, when $\boldsymbol{\Phi}_I$ is computed numerically and/or approximately, we have no guarantees about its properties. As a result, if $\boldsymbol{\Phi}_I$ is computed in this fashion, we *cannot* guarantee that the observability matrix of the linearized VIO system (21) will have the desirable nullspace properties, a prerequisite for consistent estimation.

In what follows, we describe how the IMU error-state transition matrix can be computed in closed form, as a function of the state estimates. To this end, we first examine what motion information we can infer from the IMU data, and how this information can be used for state propagation. This will enable us to derive an expression for $\boldsymbol{\Phi}_I$ that holds independently of the particular method used to integrate the IMU signals. We note that the expression derived here can be employed in any estimation problem that uses IMU measurements for state propagation (e.g. GPS-aided inertial navigation, vision-aided inertial navigation, etc.).

5.1. What information do the IMU measurements provide?

The IMU's gyroscopes and accelerometers give sampled measurements of the following continuous-time signals:

$$\boldsymbol{\omega}_m(t) = {}^I\boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \mathbf{n}_\mathbf{r}(t) \tag{22}$$

$$\mathbf{a}_m(t) = {}^I_G\mathbf{R}(t)\left({}^G\mathbf{a}(t) - \mathbf{g}\right) + \mathbf{b}_a(t) + \mathbf{n}_\mathbf{a}(t) \tag{23}$$

where ${}^I\boldsymbol{\omega}(t)$ and ${}^G\mathbf{a}(t)$ denote the IMU angular rate and linear acceleration, respectively, $\mathbf{n}_\mathbf{r}(t)$ and $\mathbf{n}_\mathbf{a}(t)$ are white Gaussian noise processes, and $\mathbf{g}$ is the gravity vector expressed in the global frame.

Equation (22) shows that the IMU gyroscopes provide measurements of the rotational velocity, expressed in the IMU frame. Using these measurements, we can only infer the *relative* rotation of the IMU between two time instants. Moreover, (23) shows that the IMU accelerometers measure specific force, which includes both the body and gravitational acceleration, expressed in the local frame. These signals provide us with information about the velocity change expressed in the *local* IMU frame, and must be "gravity-compensated" before use for state propagation. In what follows, we momentarily assume that we have access to the continuous-time signals $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$ in the time interval $[t_\ell, t_{\ell+1}]$ (corresponding to the transition from timestep ℓ to $\ell+1$), and show how these signals can be used for state propagation. The effects of the discrete-time sampling of the IMU's signals are discussed in Section 5.2.

5.1.1. Gyroscope measurements The orientation of the IMU frame at time $t_{\ell+1}$ with respect to the IMU frame at t_ℓ (i.e., the relative rotation) can be computed by integrating

a differential equation, whose form depends on the selected representation of orientation. In the unit-quaternion representation, the relative rotation of the IMU between t_ℓ and $t_{\ell+1}$ is described by ${}^{I_{\ell+1}}_{I_\ell}\bar{\mathbf{q}}$. To compute an estimate of ${}^{I_{\ell+1}}_{I_\ell}\bar{\mathbf{q}}$ using $\boldsymbol{\omega}_m(t)$, we first obtain the estimated rotational velocity in $[t_\ell, t_{\ell+1}]$ as $\hat{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}_m(t) - \hat{\mathbf{b}}_g(t)$, and then integrate the differential equation:

$$ {}^{I_t}_{I_\ell}\dot{\hat{\bar{\mathbf{q}}}} = \frac{1}{2}\begin{bmatrix} -\lfloor\hat{\boldsymbol{\omega}}(t)\times\rfloor & \hat{\boldsymbol{\omega}}(t) \\ -\hat{\boldsymbol{\omega}}(t)^{\mathrm{T}} & 0 \end{bmatrix} {}^{I_t}_{I_\ell}\hat{\bar{\mathbf{q}}}, \quad t\in[t_\ell, t_{\ell+1}] \tag{24} $$

with initial condition ${}^{I_\ell}_{I_\ell}\hat{\bar{\mathbf{q}}} = [0 \quad 0 \quad 0 \quad 1]^{\mathrm{T}}$. The relative orientation estimate ${}^{I_{\ell+1}}_{I_\ell}\hat{\bar{\mathbf{q}}}$, computed from the above differential equation, can be employed for propagating the IMU *global* orientation estimate as follows:

$$ {}^{I_{\ell+1}}_{G}\hat{\bar{\mathbf{q}}} = {}^{I_{\ell+1}}_{I_\ell}\hat{\bar{\mathbf{q}}} \otimes {}^{I_\ell}_{G}\hat{\bar{\mathbf{q}}} \tag{25} $$

5.1.2. Accelerometer measurements Using $\mathbf{a}_m(t)$ and an estimate of the accelerometer bias, we can obtain an estimate of the IMU's acceleration in the global frame as (see (23)):

$$ {}^{G}\hat{\mathbf{a}}(t) = {}^{G}_{I}\hat{\mathbf{R}}(t)\left(\mathbf{a}_m(t) - \hat{\mathbf{b}}_{\mathbf{a}}(t)\right) + \mathbf{g} \tag{26} $$

Integrating this signal twice in the time interval $[t_\ell, t_{\ell+1}]$ gives the velocity and position propagation equations:

$$ \begin{aligned} {}^{G}\hat{\mathbf{v}}_{\ell+1} &= {}^{G}\hat{\mathbf{v}}_\ell + \int_{t_\ell}^{t_{\ell+1}} {}^{G}\hat{\mathbf{a}}(\tau)\, d\tau \\ &= {}^{G}\hat{\mathbf{v}}_\ell + \int_{t_\ell}^{t_{\ell+1}} {}^{G}_{I_\tau}\hat{\mathbf{R}}\left(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_{\mathbf{a}}(\tau)\right) d\tau + \mathbf{g}\Delta t \qquad (27) \\ &= {}^{G}\hat{\mathbf{v}}_\ell + {}^{G}_{I_\ell}\hat{\mathbf{R}}\,\hat{\mathbf{s}}_\ell + \mathbf{g}\Delta t \qquad (28) \end{aligned} $$

and

$$ \begin{aligned} {}^{G}\hat{\mathbf{p}}_{\ell+1} &= {}^{G}\hat{\mathbf{p}}_\ell + \int_{t_\ell}^{t_{\ell+1}} {}^{G}\hat{\mathbf{v}}(\tau)\, d\tau \\ &= {}^{G}\hat{\mathbf{p}}_\ell + {}^{G}\hat{\mathbf{v}}_\ell \Delta t + {}^{G}_{I_\ell}\hat{\mathbf{R}}\,\hat{\mathbf{y}}_\ell + \frac{1}{2}\mathbf{g}\Delta t^2 \qquad (29) \end{aligned} $$

where $\Delta t = t_{\ell+1} - t_\ell$, ${}^{G}_{I_\ell}\hat{\mathbf{R}} = {}^{G}_{I}\hat{\mathbf{R}}(t_\ell)$, and

$$ \hat{\mathbf{s}}_\ell = \int_{t_\ell}^{t_{\ell+1}} {}^{I_\ell}_{I_\tau}\hat{\mathbf{R}}\left(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_{\mathbf{a}}(\tau)\right) d\tau \tag{30} $$

$$ \hat{\mathbf{y}}_\ell = \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{s} {}^{I_\ell}_{I_\tau}\hat{\mathbf{R}}\left(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_{\mathbf{a}}(\tau)\right) d\tau\, ds \tag{31} $$

Note that the terms $\hat{\mathbf{s}}_\ell$ and $\hat{\mathbf{y}}_\ell$ depend *only* on the values of $\mathbf{a}_m(t)$ and $\boldsymbol{\omega}_m(t)$ in the time interval $[t_\ell, t_{\ell+1}]$, as well as on the IMU biases. These terms express the information provided by the IMU about the change in the IMU velocity and position in $[t_\ell, t_{\ell+1}]$. As shown in (28) and (29), to use $\hat{\mathbf{s}}_\ell$ and $\hat{\mathbf{y}}_\ell$ to propagate the global velocity and position estimates, we must express them in the global frame (via the rotation matrix ${}^{G}_{I_\ell}\hat{\mathbf{R}}$), and account for the gravitational acceleration.

We note that Lupton and Sukkarieh (2012) showed how the IMU measurements can be "pre-integrated", so that they can be used even without an initial guess for the state. While Lupton and Sukkarieh (2012) followed a reasoning similar to that presented here, we here go one step further, and use this analysis to obtain a closed-form expression for the error-state transition matrix.

5.2. Discrete-time IMU propagation

To derive Equations (24)–(25) and (28)–(31), it was assumed that the signals $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$ were available in the entire interval $[t_\ell, t_{\ell+1}]$. In practice, however, the IMU provides samples of $\mathbf{a}_m(t)$ and $\boldsymbol{\omega}_m(t)$ at the discrete times t_ℓ and $t_{\ell+1}$. To use these measurements for state propagation, it is necessary to employ additional assumptions about the time evolution of $\mathbf{a}_m(t)$ and $\boldsymbol{\omega}_m(t)$ between the two times for which samples are available. For instance, we can assume that these signals remain constant for the entire period (equal to their values at either t_ℓ or $t_{\ell+1}$), or that they change linearly between the sampled values. These assumptions will introduce approximations, which will be small if the sample rate is sufficiently high. We stress, however, that some approximation is unavoidable, since turning a continuous-time signal to a sampled one leads to loss of information.[4]

In what follows, we describe the integration approach followed in our implementation. In our work the IMU biases are modeled as random-walk processes, i.e. we model the continuous-time evolution of the biases by $\dot{\mathbf{b}}_{\mathbf{g}}(t) = \mathbf{n}_{\mathbf{wg}}(t)$ and $\dot{\mathbf{b}}_{\mathbf{g}}(t) = \mathbf{n}_{\mathbf{wa}}(t)$, where $\mathbf{n}_{\mathbf{wg}}$ and $\mathbf{n}_{\mathbf{wa}}$ are zero-mean white Gaussian noise processes, with covariance matrices $\mathbf{Q}_{\mathbf{wg}}$ and $\mathbf{Q}_{\mathbf{wa}}$. Therefore, during propagation the bias estimates remain constant: $\hat{\mathbf{b}}_{\mathbf{g}_{\ell+1}} = \hat{\mathbf{b}}_{\mathbf{g}_\ell}$ and $\hat{\mathbf{b}}_{\mathbf{a}_{\ell+1}} = \hat{\mathbf{b}}_{\mathbf{a}_\ell}$. To propagate the IMU pose in time, at time $t_{\ell+1}$ we use the IMU samples recorded at t_ℓ and $t_{\ell+1}$ and assume that the signals $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$ change linearly between these two time instants. With this assumption, we numerically integrate (24) using fourth-order Runge–Kutta to obtain ${}^{I_{\ell+1}}_{I_\ell}\hat{\bar{\mathbf{q}}}$, and propagate the IMU orientation using (25). For the position and velocity, we employ Equations (28) and (29), where the quantities $\hat{\mathbf{s}}_\ell$ and $\hat{\mathbf{y}}_\ell$ are computed using Simpson integration of (30) and (31).

5.3. Computing $\boldsymbol{\Phi}_{I_\ell}$

We now turn our attention to computing the IMU error-state transition matrix shown in (6), which can be done by direct linearization of the state-propagation equations (25), (28), and (29). For clarity, we here show the derivation of $\boldsymbol{\Phi}_{I_\ell}$ omitting the IMU biases, while the full result for the case where the biases are included in the state vector is shown in Appendix A. Starting with the orientation error, we note that the orientation-error definition in Equations (2)–(3) satisfies

$$ {}^{I}_{G}\mathbf{R} \simeq {}^{I}_{G}\hat{\mathbf{R}}\,\left(\mathbf{I}_3 - \lfloor {}^{G}\tilde{\boldsymbol{\theta}}\times\rfloor\right) \tag{32} $$

Moreover, the estimated rotation in the time interval $[t_\ell, t_{\ell+1}]$ is corrupted by an error due to the inaccuracy of the gyroscope measurements as well as the assumptions employed during integration. We define this error based on the expression ${}^{I_{\ell+1}}_{I_\ell}\bar{\mathbf{q}} = {}^{I_{\ell+1}}_{I_\ell}\hat{\bar{\mathbf{q}}} \otimes \delta\bar{\mathbf{q}}_{\Delta t}$, from which we obtain

$$ {}^{I_{\ell+1}}_{I_\ell}\mathbf{R} \simeq {}^{I_{\ell+1}}_{I_\ell}\hat{\mathbf{R}}\,(\mathbf{I} - \lfloor\tilde{\boldsymbol{\theta}}_{\Delta t}\times\rfloor) \tag{33} $$

where $\tilde{\boldsymbol{\theta}}_{\Delta\ell}$ is a 3×1 error vector. Substituting (33) and (32) into the expression relating the true rotation matrices, ${}^{I_{\ell+1}}_{G}\mathbf{R} = {}^{I_{\ell+1}}_{I_\ell}\mathbf{R}\,{}^{I_\ell}_{G}\mathbf{R}$, and removing second-order terms, we obtain the following linearized expression for the orientation-error propagation:

$$ {}^{G}\tilde{\boldsymbol{\theta}}_{\ell+1} \simeq {}^{G}\tilde{\boldsymbol{\theta}}_\ell + \hat{\mathbf{R}}_\ell^{\mathrm{T}}\tilde{\boldsymbol{\theta}}_{\Delta t} \tag{34} $$

where we used the shorthand notation ${}^{I_\ell}_{G}\hat{\mathbf{R}} = \hat{\mathbf{R}}_\ell$. For the velocity error, we linearize (28) using (32), to obtain the linearized error-propagation equation:

$$ {}^{G}\tilde{\mathbf{v}}_{\ell+1} \simeq -\lfloor\hat{\mathbf{R}}_\ell^{\mathrm{T}}\hat{\mathbf{s}}_\ell\times\rfloor\,{}^{G}\tilde{\boldsymbol{\theta}}_\ell + {}^{G}\tilde{\mathbf{v}}_\ell + \hat{\mathbf{R}}_\ell^{\mathrm{T}}\tilde{\mathbf{s}}_\ell \tag{35} $$

The term $\tilde{\mathbf{s}}_\ell = \mathbf{s}_\ell - \hat{\mathbf{s}}_\ell$ is the error in $\hat{\mathbf{s}}_\ell$, which depends only on the IMU measurement noise and the assumptions employed during integration. Similarly, for the position we obtain

$$ {}^{G}\tilde{\mathbf{p}}_{\ell+1} \simeq -\lfloor\hat{\mathbf{R}}_\ell^{\mathrm{T}}\hat{\mathbf{y}}_\ell\times\rfloor\,{}^{G}\tilde{\boldsymbol{\theta}}_\ell + {}^{G}\tilde{\mathbf{v}}_\ell\Delta t + {}^{G}\tilde{\mathbf{p}}_\ell + \hat{\mathbf{R}}_\ell^{\mathrm{T}}\tilde{\mathbf{y}}_\ell \tag{36} $$

where $\tilde{\mathbf{y}}_\ell = \mathbf{y}_\ell - \hat{\mathbf{y}}_\ell$. By combining (34), (35) and (36), we can now write

$$ \underbrace{\begin{bmatrix} {}^{G}\tilde{\boldsymbol{\theta}}_{\ell+1} \\ {}^{G}\tilde{\mathbf{p}}_{\ell+1} \\ {}^{G}\tilde{\mathbf{v}}_{\ell+1} \end{bmatrix}}_{\tilde{\mathbf{x}}_{I_{\ell+1}}} = \underbrace{\begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -\lfloor\hat{\mathbf{R}}_\ell^{\mathrm{T}}\hat{\mathbf{y}}_\ell\times\rfloor & \mathbf{I}_3 & \Delta t\mathbf{I}_3 \\ -\lfloor\hat{\mathbf{R}}_\ell^{\mathrm{T}}\hat{\mathbf{s}}_\ell\times\rfloor & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}}_{\boldsymbol{\Phi}_{I_\ell}} \underbrace{\begin{bmatrix} {}^{G}\tilde{\boldsymbol{\theta}}_\ell \\ {}^{G}\tilde{\mathbf{p}}_\ell \\ {}^{G}\tilde{\mathbf{v}}_\ell \end{bmatrix}}_{\tilde{\mathbf{x}}_{I_\ell}} + \underbrace{\begin{bmatrix} \hat{\mathbf{R}}_\ell^{\mathrm{T}}\tilde{\boldsymbol{\theta}}_{\Delta t} \\ \hat{\mathbf{R}}_\ell^{\mathrm{T}}\tilde{\mathbf{y}}_\ell \\ \hat{\mathbf{R}}_\ell^{\mathrm{T}}\tilde{\mathbf{s}}_\ell \end{bmatrix}}_{\mathbf{w}_{d_\ell}} \tag{37} $$

It is important to note that the above expression for $\boldsymbol{\Phi}_{I_\ell}$ has an intuitive explanation. We see that: (i) the diagonal block elements are all identity matrices, which shows that the errors in the IMU state at time t_ℓ "carry over" to the next time step, as expected; (ii) the velocity error, multiplied by Δt, affects the position error at time $t_{\ell+1}$; and (iii) the orientation error at time t_ℓ is multiplied by the "lever-arms" $\hat{\mathbf{R}}_\ell^{\mathrm{T}}\hat{\mathbf{y}}_\ell$ and $\hat{\mathbf{R}}_\ell^{\mathrm{T}}\hat{\mathbf{s}}_\ell$, causing accumulation of errors in position and velocity. To write the state transition matrix as a function of the state estimates, we solve (28) and (29) for $\hat{\mathbf{s}}_\ell$ and $\hat{\mathbf{y}}_\ell$, respectively, and substitute in (37) to obtain

$$ \boldsymbol{\Phi}_{I_\ell}(\hat{\mathbf{x}}_{I_{\ell+1}}, \hat{\mathbf{x}}_{I_\ell}) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \boldsymbol{\Phi}_{\mathbf{pq}}(\hat{\mathbf{x}}_{I_{\ell+1}}, \hat{\mathbf{x}}_{I_\ell}) & \mathbf{I}_3 & \Delta t\mathbf{I}_3 \\ \boldsymbol{\Phi}_{\mathbf{vq}}(\hat{\mathbf{x}}_{I_{\ell+1}}, \hat{\mathbf{x}}_{I_\ell}) & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{38} $$

$$ \boldsymbol{\Phi}_{\mathbf{pq}}(\hat{\mathbf{x}}_{I_{\ell+1}}, \hat{\mathbf{x}}_{I_\ell}) = -\lfloor({}^{G}\hat{\mathbf{p}}_{\ell+1} - {}^{G}\hat{\mathbf{p}}_\ell - {}^{G}\hat{\mathbf{v}}_\ell\Delta t - \frac{1}{2}\mathbf{g}\Delta t^2)\times\rfloor $$

$$ \boldsymbol{\Phi}_{\mathbf{vq}}(\hat{\mathbf{x}}_{I_{\ell+1}}, \hat{\mathbf{x}}_{I_\ell}) = -\lfloor({}^{G}\hat{\mathbf{v}}_{\ell+1} - {}^{G}\hat{\mathbf{v}}_\ell - \mathbf{g}\Delta t)\times\rfloor $$

We stress that this matrix is a closed-form function of the state estimates *only*. Thus, it can be employed regardless of the way in which the integration of (24), (30) and (31) is carried out.

Note that in different implementations of IMU propagation, the form of the equations being integrated may be different from those shown above (for example, for velocity propagation one may choose to numerically compute the integral in (27), instead of breaking it into two terms as in (28)). However, this does not change the nature of the information provided by the IMU measurements, and thus does not (in fact, *should* not) change the way in which the errors in a state estimate propagate to future estimates. This way is described by the matrix in (38), as discussed above.

6. Observability properties of the MSCKF system model

We now employ the closed-form expression for $\boldsymbol{\Phi}_{I_\ell}$ derived in the preceding section to analyze the observability properties of the MSCKF's system model. To simplify the presentation, we here carry out the analysis for the case where the IMU biases are not included in the estimated state vector. These biases are known to be observable (Jones and Soatto, 2011), and thus their inclusion would not change the key result of this analysis, which is the erroneous decrease in the dimension of the nullspace of the observability matrix.[5] The fact that the analysis also holds for the case where the biases are included in the state vector is demonstrated by the results presented in Sections 8 and 9. In the estimators used in all our simulations and experiments the biases are included in the state vector, as described in Section 3.1.

A direct analysis of the observability properties of the MSCKF's linearized system model is cumbersome, due to the form of the MSCKF equations (see, e.g., (15)). To simplify the analysis, we make use of the result of Appendix B, which shows that given a linear (or, equivalently, a linearized) model, the EKF-SLAM and MSCKF measurement equations are equivalent. This means that we can study the observability of the MSCKF's linearized model by studying the EKF-SLAM linearized model, but *using the MSCKF's linearization points*. Note that the MSCKF and EKF-SLAM linearize the measurements using different state estimates: in the MSCKF, a single estimate for each feature is used for computing all the Jacobians involving this feature (see (14)), in contrast to EKF-SLAM, where the current estimate is used at each iteration. The details of the observability analysis follow.

We consider an EKF-SLAM state vector containing the IMU orientation, position, and velocity, as well as the positions of the landmarks observed in the time interval $[k, k+m]$. For this state vector, the error-state transition matrix (using the MSCKF's linearization point) at time step ℓ is given by

$$ \boldsymbol{\Phi}_\ell(\hat{\mathbf{x}}_{I_{\ell+1|\ell}}, \hat{\mathbf{x}}_{I_{\ell|\ell}}) = \begin{bmatrix} \boldsymbol{\Phi}_{I_\ell}(\hat{\mathbf{x}}_{I_{\ell+1|\ell}}, \hat{\mathbf{x}}_{I_{\ell|\ell}}) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{3M\times 3M} \end{bmatrix} \tag{39} $$

where M is the number of landmarks. Turning to the feature measurements, we note that if feature i is processed at time step $\alpha_i + 1$, then in the MSCKF the corresponding Jacobians are evaluated with the state estimates computed using all measurements up to α_i, and the feature position estimate ${}^G\hat{\mathbf{p}}_{f_i}$ computed via triangulation. Thus, the measurement Jacobian we use in our analysis becomes (see (10) and (13)):

$$\begin{aligned}&\mathbf{H}_{i\ell}(\hat{\boldsymbol{\pi}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i})\\ &= [\mathbf{H}_{I_{i\ell}}(\hat{\boldsymbol{\pi}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i})\mathbf{0}\cdots\mathbf{H}_{\mathbf{f}_{i\ell}}(\hat{\boldsymbol{\pi}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i})\cdots\mathbf{0}]\end{aligned} \tag{40}$$

where the Jacobians of (7) with respect to the IMU pose and the feature position are given by

$$\mathbf{H}_{\mathbf{f}_{i\ell}}(\hat{\boldsymbol{\pi}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i}) = \mathbf{J}_{i\ell}(\hat{\boldsymbol{\pi}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i})\, {}^C_I\mathbf{R}\,\hat{\mathbf{R}}_{\ell|\alpha_i} \tag{41}$$

$$\begin{aligned}\mathbf{H}_{I_{i\ell}}(\hat{\boldsymbol{\pi}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i}) &= \mathbf{H}_{\mathbf{f}_{i\ell}}(\hat{\boldsymbol{\pi}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i})\\ &\quad \left[\lfloor\left({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{\ell|\alpha_i}\right)\times\rfloor \quad -\mathbf{I}_3 \quad \mathbf{0}_3\right]\end{aligned}$$

$$\mathbf{J}_{i\ell}(\hat{\boldsymbol{\pi}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i}) = \frac{1}{{}^{C_\ell}\hat{z}_{f_i}}\begin{bmatrix}1 & 0 & \frac{-{}^{C_\ell}\hat{x}_{f_i}}{{}^{C_\ell}\hat{z}_{f_i}}\\ 0 & 1 & \frac{-{}^{C_\ell}\hat{y}_{f_i}}{{}^{C_\ell}\hat{z}_{f_i}}\end{bmatrix} \tag{42}$$

with $[{}^{C_\ell}\hat{x}_{f_i} \quad {}^{C_\ell}\hat{y}_{f_i} \quad {}^{C_\ell}\hat{z}_{f_i}]^{\mathrm{T}}$ being the estimate of the feature position with respect to the camera:

$$\begin{bmatrix}{}^{C_\ell}\hat{x}_{f_i}\\ {}^{C_\ell}\hat{y}_{f_i}\\ {}^{C_\ell}\hat{z}_{f_i}\end{bmatrix} = {}^C_I\mathbf{R}\,\hat{\mathbf{R}}_{\ell|\alpha_i}\left({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{\ell|\alpha_i}\right) + {}^C\mathbf{p}_I \tag{43}$$

By substitution of (39) and (40) in (21), we can therefore study the observability properties of the linearized system model of the MSCKF. Before doing that, however, it is interesting to first examine what the observability matrix would look like in the "ideal" case where the *true* state estimates were used in computing all Jacobians.

6.0.1. "Ideal" observability matrix To derive the "ideal" observability matrix, we evaluate the state transition matrix as $\boldsymbol{\Phi}(\mathbf{x}_{I_{\ell+1}}, \mathbf{x}_{I_\ell})$ (see (39)), and evaluate the Jacobian matrix in (40) using the true states. Substituting these matrices in (21) yields the following result for the block row of the observability matrix corresponding to the observation of feature i at time step ℓ:

$$\check{\mathcal{O}}_{i\ell} = \check{\mathbf{M}}_{i\ell}\left[\check{\boldsymbol{\Gamma}}_{i\ell} \quad -\mathbf{I}_3 \quad -\Delta t_\ell\mathbf{I}_3 \quad \mathbf{0}_3 \quad \cdots \quad \mathbf{I}_3 \quad \cdots \quad \mathbf{0}_3\right] \tag{44}$$

$$\check{\mathbf{M}}_{i\ell} = \check{\mathbf{J}}_{i\ell}\, {}^C_I\mathbf{R}\,\mathbf{R}_\ell \tag{45}$$

$$\check{\boldsymbol{\Gamma}}_{i\ell} = \left\lfloor\left({}^G\mathbf{p}_{f_i} - {}^G\mathbf{p}_k - {}^G\mathbf{v}_k\Delta t_\ell - \frac{1}{2}\mathbf{g}\Delta t_\ell^2\right)\times\right\rfloor \tag{46}$$

In the above equations, Δt_ℓ denotes the time interval between time steps k and ℓ, and we have used the symbol "$\check{\ }$" to denote a matrix computed using the true state values. At this point, if we define the matrix $\mathbf{N}$ as

$$\mathbf{N} = \begin{bmatrix}\mathbf{0}_3 & \mathbf{g}\\ \mathbf{I}_3 & -\lfloor{}^G\mathbf{p}_k\times\rfloor\mathbf{g}\\ \mathbf{0}_3 & -\lfloor{}^G\mathbf{v}_k\times\rfloor\mathbf{g}\\ \mathbf{I}_3 & -\lfloor{}^G\mathbf{p}_{f_1}\times\rfloor\mathbf{g}\\ \mathbf{I}_3 & -\lfloor{}^G\mathbf{p}_{f_2}\times\rfloor\mathbf{g}\\ \vdots & \vdots\\ \mathbf{I}_3 & -\lfloor{}^G\mathbf{p}_{f_N}\times\rfloor\mathbf{g}\end{bmatrix} \tag{47}$$

it is easy to verify that $\check{\mathcal{O}}_{i\ell}\cdot\mathbf{N} = \mathbf{0}_{2\times4}$. Since this holds for any i and any ℓ (i.e. for all block rows of the observability matrix), we conclude that $\check{\mathcal{O}}\cdot\mathbf{N} = \mathbf{0}$. In addition, the four columns of $\mathbf{N}$ are linearly independent, which means that they form a basis for the nullspace of the observability matrix $\check{\mathcal{O}}$ (in (Li and Mourikis, 2011), we proved that no additional basis vectors can be found for the nullspace).

In other words, the above shows that the observability matrix, when all Jacobians are computed using the true states, has a nullspace of dimension four. Examining the physical interpretation of the basis, we see that the first three vectors correspond to global translation of the state vector, while the last column corresponds to rotations about gravity (i.e. the yaw) (Li and Mourikis, 2011). Thus, if we were able to evaluate all of the Jacobians using the true state estimates, the observability properties of the linearized system model would match those of the nonlinear system, as desired.

6.0.2. MSCKF observability matrix Using Equations (39) and (40) in (21), the block row of the observability matrix $\mathcal{O}$ corresponding to the observation of feature i at time step ℓ becomes

$$\mathcal{O}_{i\ell} = \mathbf{M}_{i\ell}\left[\boldsymbol{\Gamma}_{i\ell} + \Delta\boldsymbol{\Gamma}_{i\ell} \quad -\mathbf{I}_3 \quad -\Delta t_\ell\mathbf{I}_3 \quad \mathbf{0}_3 \quad \cdots \quad \mathbf{I}_3 \quad \cdots \quad \mathbf{0}_3\right] \tag{48}$$

where

$$\mathbf{M}_{i\ell} = \mathbf{J}_{i\ell}(\hat{\mathbf{x}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i})\, {}^C_I\mathbf{R}\,\hat{\mathbf{R}}_{\ell|\alpha_i} \tag{49}$$

$$\boldsymbol{\Gamma}_{i\ell} = \left\lfloor\left({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{k|k} - {}^G\hat{\mathbf{v}}_{k|k}\Delta t_\ell - \frac{1}{2}\mathbf{g}\Delta t_\ell^2\right)\times\right\rfloor \tag{50}$$

$$\Delta\boldsymbol{\Gamma}_{i\ell} = \lfloor{}^G\hat{\mathbf{p}}_{\ell|\ell-1} - {}^G\hat{\mathbf{p}}_{\ell|\alpha_i}\times\rfloor + \sum_{j=k+1}^{\ell-1}\left(\mathbf{E}_{\mathbf{p}}^j + \sum_{s=k+1}^{j}\mathbf{E}_{\mathbf{v}}^s\Delta t\right) \tag{51}$$

with

$$\mathbf{E}_{\mathbf{p}}^j = \lfloor{}^G\hat{\mathbf{p}}_{j|j-1} - {}^G\hat{\mathbf{p}}_{j|j}\times\rfloor \tag{52}$$

$$\mathbf{E}_{\mathbf{v}}^j = \lfloor{}^G\hat{\mathbf{v}}_{j|j-1} - {}^G\hat{\mathbf{v}}_{j|j}\times\rfloor \tag{53}$$

By comparing (48)–(51) with (44)–(46) we see that the structure of the observability matrix in both cases is similar.

The key difference is that when the Jacobians are evaluated using the state *estimates* instead of the true states, the "disturbance" term $\Delta\mathbf{\Gamma}_{i\ell}$ appears. While $\Delta\mathbf{\Gamma}_{i\ell}$ is quite complex, we can observe that it contains terms that depend on the corrections (e.g. ${}^G\hat{\mathbf{p}}_{j|j} - {}^G\hat{\mathbf{p}}_{j|j-1}$, ${}^G\hat{\mathbf{v}}_{j|j} - {}^G\hat{\mathbf{v}}_{j|j-1}$) that the filter applies at different time steps. Since these corrections are random, the term $\Delta\mathbf{\Gamma}_{i\ell}$ is a random one, and this "destroys" the special structure of the observability matrix. As a result, the property $\mathcal{O}_{i\ell} \cdot \mathbf{N} = \mathbf{0}$ does not hold.

It can be shown that the nullspace of $\mathcal{O}$ (i.e. the unobservable subspace) is now of dimension only three (Li and Mourikis, 2011). This nullspace is spanned by the first three column vectors (the first block column) of $\mathbf{N}$ in (47), which means that the global yaw *erroneously appears* to be observable. As a result, the MSCKF underestimates the uncertainty of the yaw. Since the yaw uncertainty affects the uncertainty of other state variables (e.g. the position), eventually the uncertainty of all states will be underestimated, and the estimator will be inconsistent. This helps to explain the results observed in the NEES plot of Figure 1.

It is important to point out that the incorrect observability properties of the linearized system model do not affect only the MSCKF algorithm. In Appendix C the observability matrix of the linearized model of EKF-SLAM is shown. This matrix has a nullspace of dimension three as well, similarly to the MSCKF. In fact, for the EKF-SLAM methods, the "disturbance" term appearing in the observability matrix contains additional terms due to the corrections in the feature position estimates. Such terms do not appear in the MSCKF, which uses only one estimate for each feature in all Jacobians.

7. MSCKF 2.0

In the preceding section, we proved that the linearized system model employed by the MSCKF has incorrect observability properties, causing inconsistency. In this section, we propose a simple solution to this problem. Moreover, we propose an extension of the basic MSCKF algorithm, which serves to improve the algorithm's performance in real-world scenarios. Specifically, in our analysis to this point it was assumed that the IMU-to-camera transformation (position and orientation) is perfectly known. In practice, this is typically not the case: while an estimate for the transformation may be known from a CAD plot or manual measurements, this is typically inexact. For example, the coordinate frames of the sensors are typically not perfectly aligned with the sensor housing, which makes manual measurements less useful. If the transformation is assumed to be perfectly known, even though the available estimates are not exact, both the consistency and the accuracy of the filter will deteriorate. To address this issue, we propose to include the camera-to-IMU transformation in the estimated state vector of the MSCKF.

7.1. Enforcing correct dimension of the unobservable subspace

As shown in Section 6, the fact that in the MSCKF *different estimates of the same states* are used for computing Jacobians leads to an infusion of "fictitious" information about the yaw. Specifically, the use of different estimates for the IMU position and velocity results in non-zero values for the disturbance terms $\Delta\mathbf{\Gamma}_{i\ell}$ (see (51)), which change the dimension of the nullspace of the observability matrix. To remove these $\Delta\mathbf{\Gamma}_{i\ell}$ terms, a simple solution is to ensure that only one estimate of each IMU position and velocity is used in all Jacobians involving it. A causal approach to achieve this is to always use the *first* available estimate for each state. Specifically, we compute the filter Jacobians as follows:

- Compute the IMU error-state transition matrix at timestep ℓ as
$$\mathbf{\Phi}^{\star}_{I_\ell}(\hat{\mathbf{x}}_{I_{\ell+1|\ell}}, \hat{\mathbf{x}}_{I_{\ell|\ell-1}}) \tag{54}$$
- Calculate measurement Jacobians as
$$\mathbf{H}^{\star}_{\mathbf{f}_{i\ell}} = \mathbf{J}_{i\ell}(\hat{\mathbf{x}}_{\ell|\alpha_i}, {}^G\hat{\mathbf{p}}_{f_i})\ {}^C_I\mathbf{R}\ \hat{\mathbf{R}}_{\ell|\alpha_i} \tag{55}$$
$$\mathbf{H}^{\star}_{I_{i\ell}} = \mathbf{H}^{\star}_{\mathbf{f}_{i\ell}}\left[\ \lfloor\left({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{\ell|\ell-1}\right)\times\rfloor \quad -\mathbf{I}_3 \quad \mathbf{0}_3\ \right] \tag{56}$$

As a result of the above changes, only the "propagated" estimates for the position and velocity (e.g. ${}^G\hat{\mathbf{p}}_{\ell|\ell-1}$ and ${}^G\hat{\mathbf{v}}_{\ell|\ell-1}$) are used in computing Jacobians. It is easy to show that with this change, the observability matrix regains the correct nullspace dimension, and thus the infusion of "fictitious" information for the yaw is avoided. We stress that we allow the state estimates to be updated normally; the only change we make to the MSCKF equations is that we do not use the updated estimates of the position and velocity in computing Jacobians. This change, which incurs *no* additional computational cost, substantially improves performance, as shown in the simulation and experimental results presented in Sections 8 and 9.

The idea of using the first estimates of all states to ensure the correct observability properties of the linearized system model can also be employed for EKF-SLAM VIO. In this case, in addition to the IMU position and velocity estimates, we must also use the same (first) estimate of each feature when computing all Jacobians involving it. As shown in Section 8, the resulting EKF-SLAM algorithms outperform the standard ones, yet cannot reach the accuracy or consistency of the MSCKF 2.0.

7.2. Camera-to-IMU calibration

To estimate the camera-to-IMU transformation in the MSCKF, we include the transformation parameters in the filter state vector. Specifically, we augment the IMU state by adding the pose of the camera with respect to the IMU, $\boldsymbol{\pi}_{CI} = \{{}^C_I\bar{\mathbf{q}}, {}^C\mathbf{p}_I\}$:

$$\mathbf{x}^{\star}_I = \begin{bmatrix} {}^I_G\bar{\mathbf{q}}^{\mathrm{T}} & {}^G\mathbf{p}^{\mathrm{T}} & {}^G\mathbf{v}^{\mathrm{T}} & \mathbf{b}_{\boldsymbol{g}}^{\mathrm{T}} & \mathbf{a}_{\boldsymbol{a}}^{\mathrm{T}} & {}^C_I\bar{\mathbf{q}}^{\mathrm{T}} & {}^C\mathbf{p}_I^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{57}$$

where we have used the symbol "⋆" to distinguish this state vector from the original IMU state in (1). During propagation, the estimates for the camera-to-IMU parameters as well as their covariance remain unchanged. For the updates, only minimal modifications of the MSCKF equations are required to account for the inclusion of $\boldsymbol{\pi}_{CI}$ in the state vector. Specifically, the linearized residual equations (13) for each feature measurement now become:

$$\mathbf{r}_{ij} = \mathbf{z}_{ij} - \mathbf{h}(\hat{\boldsymbol{\pi}}_{j|\ell-1}, \hat{\boldsymbol{\pi}}_{CI_{\ell|\ell-1}}, \hat{\mathbf{f}}_i) \tag{58}$$

$$\simeq \mathbf{H}_{ij}\tilde{\boldsymbol{\pi}}_{j|\ell-1} + \mathbf{H}_{C_{ij}}\tilde{\boldsymbol{\pi}}_{CI_{\ell|\ell-1}} + \mathbf{H}_{\mathbf{f}_{ij}}\tilde{\mathbf{f}}_i + \mathbf{n}_{ij} \tag{59}$$

for $j = \ell - N, \ldots, \ell - 1$, where $\mathbf{H}_{C_{ij}}$ is the Jacobian of the measurement with respect to the camera–IMU pose:

$$\mathbf{H}_{C_{ij}} = \mathbf{J}_{i\ell}\left[{}^{C}_{I}\hat{\mathbf{R}}_{\ell|\ell-1}\ \lfloor\hat{\mathbf{R}}_{\ell|\ell-1}({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{\ell|\ell-1})\times\rfloor\ \ \mathbf{I}_3\right]$$

The equations (59) can still be stacked to obtain an equation analogous to (14), as the error $\tilde{\boldsymbol{\pi}}_{CI_{\ell|\ell-1}}$ is now a part of the state vector. Thus, the MSCKF's method of removing the feature error to create a residual suitable for an EKF update (see (15)) can be applied with no further changes.

Estimating the camera-to-IMU transformation in the MSCKF framework offers two key advantages over alternative EKF-based algorithms for the same task: First, it can operate in unknown environments, with no *a priori* known features (in contrast to methods such as those of (Mirzaei and Roumeliotis, 2008 and Kelly et al., 2008)). Second, since it is based on the MSCKF, it shares all of the advantages of the MSCKF over SLAM-based methods (e.g., (Jones and Soatto, 2011; Kelly and Sukhatme, 2011)), as outlined in Section 3.4. For instance, its computational cost is significantly lower, and it is less sensitive to the nonlinear nature of the estimation problem. Moreover, based on the analysis of (Jones and Soatto, 2011; Kelly and Sukhatme, 2011), we know that the camera-to-IMU transformation is observable for general trajectories. Thus, by including it in the MSCKF state vector, we do not run the risk of introducing additional variables that may become "erroneously observable". We term the algorithm that uses the first estimates of each state in computing Jacobians and includes the IMU-to-camera calibration parameters in the state vector MSCKF 2.0.

8. Simulation results

In this section we present simulation results that illustrate the analysis presented in the preceding sections, and demonstrate the performance of the MSCKF 2.0 algorithm compared to alternatives. All of the simulation data is generated based on real-world datasets, as explained in Section 3.4, and all of the results reported are averages over 50 Monte Carlo trials.

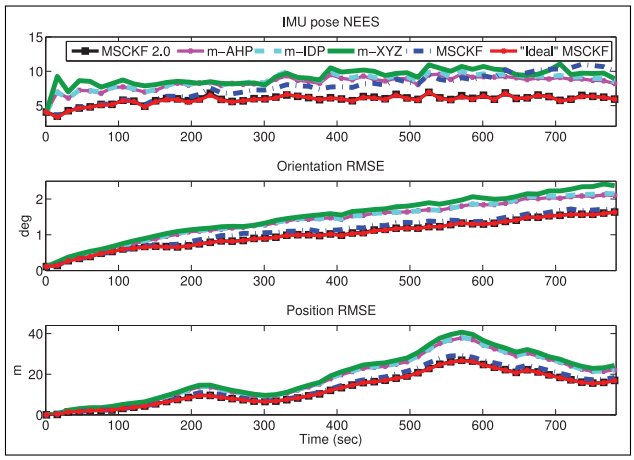

Fig. 2. Simulation results: the average position and orientation RMSE over 50 Monte-Carlo trials. The algorithms compared are the MSCKF (blue dash-dotted line), the "ideal" MSCKF (red line with x-marks), the MSCKF 2.0 (black line with squares), the m-*XYZ* SLAM (green solid line), the m-IDP SLAM (cyan dashed line), and the m-AHP SLAM (magenta line with "plus"-marks).

8.1. Comparison with EKF-based SLAM

We first compare the performance of the MSCKF and MSCKF 2.0 algorithms with EKF-SLAM based methods for VIO. For the results presented here, the camera-to-IMU calibration parameters are assumed to be known without uncertainty. In the results presented in Section 3.4, it was shown that the original MSCKF algorithm outperforms the standard EKF-SLAM algorithms using either the *XYZ*, IDP, or AHP feature parameterizations. We thus here focus on comparing the performance of the MSCKF-based algorithms to that of the "modified" EKF-SLAM versions, where the first estimate of each state is used in computing Jacobians to ensure the correct rank of the linearized system's observability matrix. These modified algorithms are identified as m-*XYZ*, m-IDP, and m-AHP. In addition, in this simulation we include an "ideal" MSCKF algorithm, in which the true IMU states and feature positions are used for computing all of the filter Jacobians. This algorithm (which is only realizable in a simulation environment), can serve as a benchmark of performance for the MSCKF-based methods. For the results presented here, exactly the same simulation data as in Section 3.4 are used, to facilitate comparison.

Figure 2 shows the average IMU pose NEES as well as the IMU position and orientation RMSE over time, for the three MSCKF-based methods, as well as for the three EKF-SLAM methods (with $\kappa = 4$ for the SLAM methods). Moreover, Table 2 provides average NEES and RMSE values for all of the algorithms (this table includes the results of Section 3.4 for easier comparison).

We can conclude that all of the "modified" algorithms, which use the first estimates of each state in Jacobian computation, outperform their counterparts that use the standard approach for Jacobian computation. Not only are these

Table 2. Average RMSE and NEES results for all the EKF-based VIO algorithms tested in the simulations.

	$\kappa = 2$			$\kappa = 4$			$\kappa = 6$		
Algorithm	Position RMSE (m)	Orientation RMSE (°)	NEES	Position RMSE (m)	Orientation RMSE (°)	NEES	Position RMSE (m)	Orientation RMSE (°)	NEES
XYZ	N/A	N/A	N/A	78.447	5.609	4.9×10^3	53.469	3.974	1.3×10^3
IDP	69.502	3.731	2205.101	26.193	1.916	268.141	22.878	1.803	167.261
AHP	67.061	4.795	273.247	52.355	4.531	129.602	36.858	3.129	48.236
m-*XYZ*	60.564	3.160	116.721	19.297	1.512	9.185	12.477	1.238	7.385
m-IDP	40.912	2.057	57.346	18.144	1.400	8.600	15.498	1.211	7.156
m-AHP	38.288	2.311	38.932	18.010	1.385	8.357	15.494	1.205	7.160

	Position RMSE (m)	Orientation RMSE (°)	NEES
MSCKF	14.401	1.102	7.741
MSCKF 2.0	12.840	1.008	5.890
"Ideal" MSCKF	12.720	1.001	5.816

algorithms more consistent (i.e. they have smaller NEES), but also more accurate (i.e. smaller RMSE). These results show that enforcing the correct observability properties of the linearized system is crucially important for the performance of *all* EKF-based VIO methods, and validate the analysis of Section 6. Despite the improvement that the modified EKF-SLAM algorithms offer, however, they are all less accurate than all of the MSCKF-based methods. This shows the advantages of the MSCKF approach to processing the feature measurements, which copes better with nonlinearities by not making Gaussian assumptions about the feature pdfs.

Additionally, we can observe that the performance of the MSCKF 2.0 algorithm is almost indistinguishable from that of the "ideal" MSCKF, both in terms of accuracy and consistency. This indicates that, as long as the correct observability properties are ensured, using slightly less accurate linearization points in computing the Jacobians does not significantly degrade the estimation performance. Based on the simulation results (and given that the "ideal" MSCKF is not realizable), we can conclude that the MSCKF 2.0 is the preferred VIO method among the EKF-based approaches considered.

8.2. Comparison with iterative-optimization methods

We next compare the performance of the MSCKF-based algorithms to that of an iterative-minimization-based method. Specifically, we use an information-form fixed-lag smoother (FLS), based on the work of Sibley et al. (2010) for comparison. This is a sliding-window bundle adjustment method that marginalizes older states to maintain a constant computational cost. The FLS is essentially the counterpart of the MSCKF within the class of iterative-minimization methods, which allows for a meaningful comparison. In our implementation, the sliding window contains a number of IMU poses as well as the features observed in these poses. The IMU measurements are used to provide the "process-model" information between the poses of the window, while the feature observations provide the "sensor-model" information (see Section 2.1 of (Sibley at al. 2010)). Every time a new image is recorded, Gauss–Newton minimization is employed to update the state estimates in the sliding window, and subsequently the oldest pose, and features that are no longer observed, are marginalized out. All of the methods tested (MSCKF, "ideal" MSCKF, MSCKF 2.0, and FLS) use a sliding window of the same length.

For these tests we employ a much longer dataset as our basis for generating simulated data. Specifically, we use the Cheddar Gorge dataset (Simpson et al., 2011), which involves a 29 km long trajectory, collected in 56 minutes of driving. For this dataset an Xsens IMU provided measurements at 100 Hz, and images were available at 20 Hz. In each image, 240 features were tracked on average, and the average track length was 4.1 frames (note that this is due to the fact that a very large percentage of features are tracked for short periods in this dataset, which involves a fast-moving vehicle).

Before examining the averaged results of all the Monte Carlo trials, it is interesting to examine the results of estimation for the rotation about gravity (the yaw) in a single trial. Figure 3 shows the estimation errors in the yaw for the four algorithms, as well as the $\pm 3\sigma$ envelopes computed using the reported covariance of each method (these are the reported 99.7% confidence regions). The most important observation here is that the reported standard deviation for both the MSCKF and the FLS fluctuates about a constant value, *as if* the yaw was observable. By contrast, in the "ideal" MSCKF and the MSCKF 2.0 algorithms, the standard deviation of the yaw increases over time, as theoretically expected, given that the yaw is unobservable. This figure clearly demonstrates the importance of the observability properties of the linearized system: when these do not match the properties of the underlying nonlinear

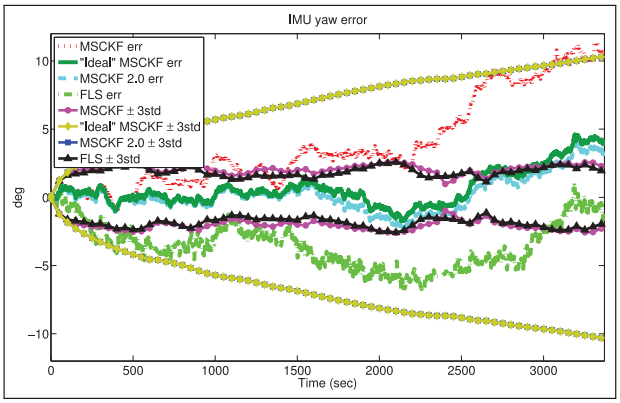

Fig. 3. IMU yaw errors and $\pm 3\sigma$ bounds in one representative trial of the Cheddar Gorge simulation. The yaw error for the MSCKF (dotted line – red), the "ideal" MSCKF (solid line – dark green), the MSCKF 2.0 (dashed line – cyan), and the FLS (dashdotted line – light green). The $\pm 3\sigma$ bounds for the MSCKF (circles – magenta), the "ideal" MSCKF (diamonds – yellow), the MSCKF 2.0 (squares – blue), and the FLS (triangles – black).

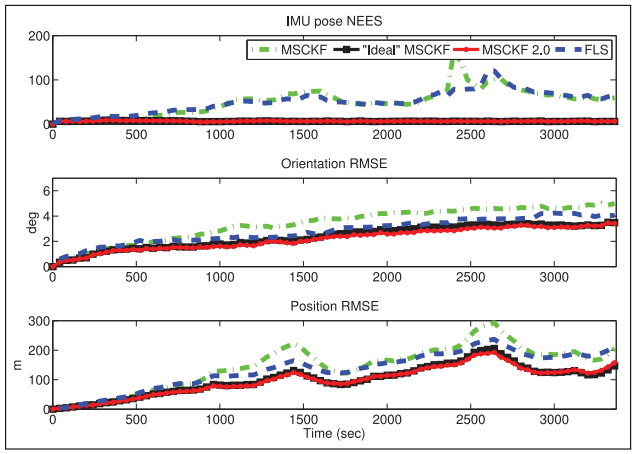

Fig. 4. Average NEES and RMSE over 50 Monte Carlo trials of the Cheddar Gorge simulation. The dotted green line corresponds to the MSCKF, the black line with squares to the "ideal" MSCKF, the red line with x-marks to the MSCKF 2.0, and the blue dashdotted line to the FLS.

system, the estimation results (e.g. reported uncertainty) exhibit fundamentally incorrect characteristics. We note here that the FLS also suffers from the same inconsistency problem, even though it employs iterative re-linearization, as shown in (Dong-Si and Mourikis, 2011).

The three subplots in Figure 4 show the average NEES for the IMU pose, as well as the RMSE for the IMU orientation and position, averaged over 50 Monte Carlo trials. Table 3 lists the average NEES and RMSE values for the four algorithms. First, we note that the performance of the MSCKF 2.0 is similar to that of the "ideal" MSCKF, and that both algorithms clearly outperform the standard MSCKF. These results once again show that by enforcing the correct observability properties, the filter's performance can be significantly improved. In addition, in this simulation environment, we see that the performance difference between the standard MSCKF and the MSCKF 2.0 is more pronounced than before. This is due to the fact that the Cheddar Gorge dataset is significantly longer (both in trajectory length and duration). As a result, more "spurious" information about the yaw is accumulated, due to the incorrect observability properties of the filter's linearized model. In turn, this causes a larger degradation in the estimates of the standard MSCKF.

More importantly though, we see that the MSCKF 2.0 (as well as the "ideal" MSCKF) attains substantially better accuracy and consistency even than the iterative FLS method. This occurs even though the latter uses approximately five times more computation time. The performance difference between the MSCKF 2.0 and the FLS demonstrates that (at least in the case examined here) having a linearized system model with appropriate observability properties is more important than using re-linearization to better approximate the nonlinear measurement models.

Table 3. Average NEES and RMSE for Figure 4.

	Position RMSE (m)	Orientation RMSE°	NEES
FLS	133.4	2.83	50.97
MSCKF	146.2	3.40	51.72
MSCKF 2.0	97.7	2.21	6.53
"Ideal" MSCKF	100.2	2.35	6.45

8.3. Performance of the online camera-to-IMU calibration

To test the performance of the online camera-to-IMU calibration, we conducted a second Monte Carlo simulation test based on the Cheddar Gorge dataset. In each Monte Carlo trial, the camera-to-IMU translation and orientation were set equal to known nominal values with the addition of random errors $\delta\mathbf{p}$ and $\delta\boldsymbol{\theta}$, respectively. In each trial, $\delta\mathbf{p}$ and $\delta\boldsymbol{\theta}$ were randomly drawn from zero-mean Gaussian distributions with standard deviations equal to $\sigma_p = 0.01$ m and $\sigma_\theta = 0.5^\circ$ along each axis, respectively. This setup models the scenario in which the transformation parameters are approximately, but not exactly, known (e.g. through manual measurement).

In this simulation, we compared the performance of four algorithms: (i) the MSCKF 2.0 algorithm with the online calibration enabled; (ii) the MSCKF 2.0 algorithm with the online calibration disabled, and assuming the camera-to-IMU transformation is equal to its nominal value. This will help to demonstrate the effect of incorrect transformation estimates on the estimator's accuracy and consistency. (iii) the m-AHP algorithm, with online camera-to-IMU calibration implemented (out of all of the EKF-SLAM algorithms

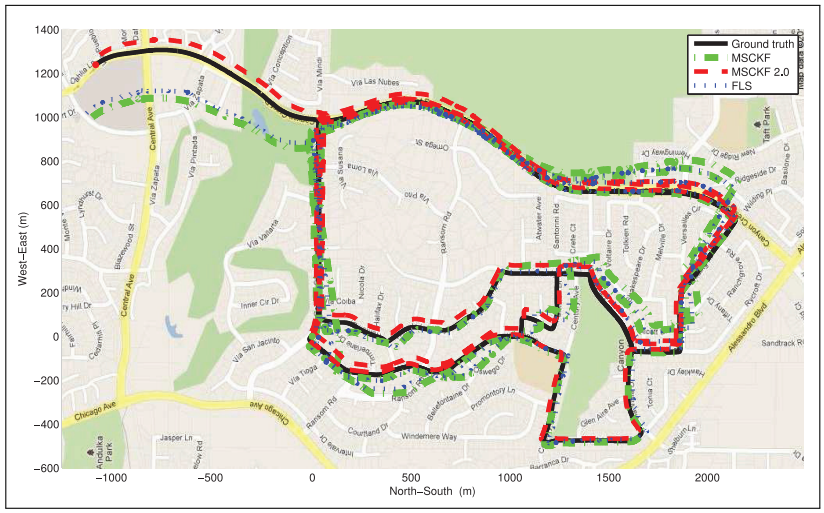

Fig. 5. Trajectory estimates plotted on a map of Canyon Crest, Riverside, CA. The initial vehicle position is shown by a green circle, and the end position by a red circle. The black solid line corresponds to the ground truth, the green dash-dotted to the MSCKF, the red dashed line to the MSCKF 2.0, and the blue dotted line to the FLS.

considered, the m-AHP is the one with the best performance, and thus is the "best-case scenario" for online calibration in the SLAM framework); and, finally, (iv) we ran the MSCKF 2.0 algorithm with perfectly known calibration, as a benchmark of performance (we term this the "precise" scenario).

Table 4 shows the results of Monte Carlo trials, listing separately the RMSE errors along the three axes (the x- and y-axes are parallel to the ground, while the z-axis is parallel to gravity). Three key observations can be made here. First, we observe that when the camera calibration is falsely assumed to be known (calibration "off"), the filter's accuracy and consistency are severely degraded, particularly along the z-axis. This happens even though the errors of the calibration parameters are relatively small in these simulations. Second, we can observe that the accuracy of the IMU pose estimates computed when the calibration is performed online with the MSCKF 2.0 is almost identical to the accuracy that is achieved with *a priori* perfectly known calibration. This is practically significant, as it indicates that more sophisticated (and time-consuming) calibration processes involving specialized equipment may not be required for most applications. Third, by comparing the performance of m-AHP to MSCKF 2.0, we observe that the SLAM-based approach attains lower accuracy and consistency for the IMU pose, as well as lower precision for the camera-to-IMU calibration. This result, which agrees with those of Section 8.1, demonstrates the advantage of performing the camera-to-IMU calibration in the MSCKF 2.0 framework.

9. Real-world experiment

We next describe the results of a real-world experiment, during which an IMU/camera platform was mounted on top of a car and driven on the streets of Riverside, CA. The sensors consisted of an Xsens MTi-G unit, and a PointGrey Bumblebee2 stereo pair (only a single camera's images are used). The IMU provided measurements at 100 Hz, while the camera images were stored at 20 Hz. For position ground truth we used a GPS–INS estimate of the trajectory. For feature extraction the Shi-Tomasi algorithm was used (Shi and Tomasi, 1994), and matching was carried out by normalized cross-correlation. On average, approximately 290 features were tracked per image. The experiment lasted about 37 minutes, during which the vehicle drove approximately 21.5 km. Some sample images from the experiment are shown in Figure 7.

Figure 5 shows the ground truth trajectory on a map of the area where the vehicle drove, as well as the estimates computed by three algorithms: the MSCKF, the FLS, and the proposed MSCKF 2.0. These are the three most accurate estimators tested, and we only present their results for clarity. Figure 6 plots the estimation error as well as the reported standard deviation of the yaw and the x–y position for the three algorithms. Similarly to what was observed in Figure 3, we see that the MSCKF 2.0 (plots on the left) offers a better characterization of the actual uncertainty. By contrast, the uncertainties of both the yaw and the IMU position are underestimated by the MSCKF and the FLS (plots on the right). The estimation errors for these algorithms are also significantly larger than those of the MSCKF 2.0. The elevation (altitude) estimates of the MSCKF 2.0 are also more accurate, having worst-case errors of 26 m, compared with 27 m for the MSCKF and 33 m for the FLS. The largest position error for the MSCKF 2.0 algorithm is approximately 58 m, which corresponds to only 0.28% of the travelled distance. In contrast, the trajectory estimates reported by the MSCKF and the FLS are much less accurate, with largest position errors of about 230 m and 202 m, respectively.

Table 4. Performance of the online camera–IMU calibration.

Transformation		Imprecise			precise
Estimator		m-AHP	MSCKF 2.0		
Calibration		on	on	off	N/A
IMU pose RMSE	x (m)	86.2	59.6	59.6	59.0
	y (m)	113.2	80.3	84.9	79.9
	z (m)	6.4	5.5	117.0	2.5
	roll (°)	0.11	0.10	0.15	0.10
	pitch (°)	0.12	0.11	0.15	0.10
	yaw (°)	3.27	2.26	2.26	2.25
IMU pose NEES		8.24	7.45	2591	7.02
Calibration RMSE	position (m)	0.03	0.03	N/A	N/A
	orientation (°)	0.07	0.05	N/A	N/A

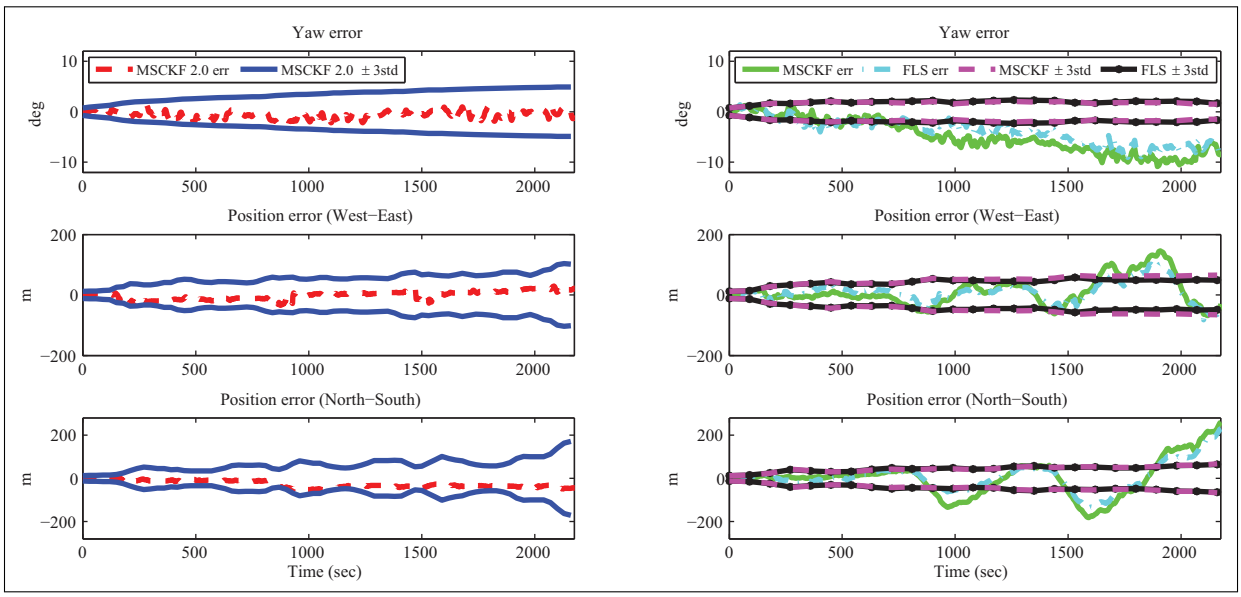

Fig. 6. Estimation errors for the three approaches. The left plots are the results for the MSCKF 2.0, and the right plots for the MSCKF and FLS.

Fig. 7. Sample images recorded during the experiment.

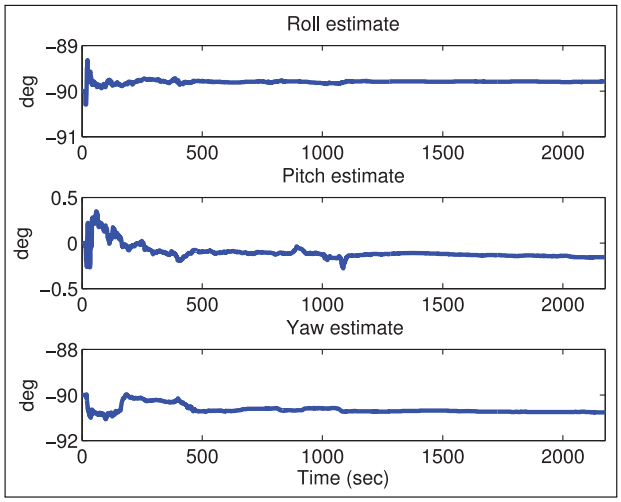

Fig. 8. Orientation estimates of the camera expressed in the IMU frame. Note that the estimation uses a quaternion representation, and the results are transformed to roll–pitch–yaw for visualization purposes only.

Since the precise IMU-to-camera parameters were not perfectly known, they were estimated online by the MSCKF 2.0 algorithm, using manual measurements for initialization. Figure 8 shows the orientation estimates between the IMU and the camera, where the roll, pitch, and yaw angles describe the camera orientation expressed in the IMU frame. The final standard deviation of the orientation estimates is $[0.008^\circ \quad 0.008^\circ \quad 0.039^\circ]$ about the three axes, while for the position we obtain [0.008 0.008 0.005] m, showing the high accuracy attainable by the online camera-to-IMU calibration process.

As a final remark, we note that in this experiment, the average processing time per update of the MSCKF 2.0 (including image processing and estimation) is 10 ms measured on a Core i7 processor at 2.66 GHz, with a single-threaded C++ implementation. Since the image period is 50 ms, the algorithm's performance is comfortably within the requirements for real-time operation.

10. Conclusion

In this paper, we have presented a detailed analysis of the properties and performance of different EKF-based VIO algorithms. We showed that the MSCKF algorithm attains better accuracy and consistency than EKF-based SLAM due to its less strict probabilistic assumptions and delayed linearization. In addition, we performed a rigorous study of the consistency properties of EKF-based VIO algorithms, and showed that the filters' linearized system models have incorrect observability properties, which result in inconsistency. To address this problem, we developed the MSCKF 2.0 algorithm, which uses a novel closed-form expression for the IMU error-state transition matrix and fixed linearization states to ensure the appropriate observability properties. Moveover, the MSCKF 2.0 algorithm is capable of performing online camera-to-IMU calibration. Extensive Monte Carlo simulations and real-world experimental testing provide strong validation of our theoretical analysis, and demonstrate that the proposed MSCKF 2.0 algorithm is capable of performing long-term, high-precision, consistent VIO in real time. In fact, the MSCKF 2.0 algorithm is shown to outperform even an iterative-minimization-based FLS, an algorithm with substantially higher computational requirements.

Funding

This work was supported by the National Science Foundation (grant number IIS-1117957), the UC Riverside Bourns College of Engineering, and the Hellman Family Foundation.

Notes

1. Note that hybrid approaches have also appeared (e.g. Mourikis and Roumeliotis, 2008), which use the estimates of the EKF as initial guesses for iterative minimization. Moreover, hybrid schemes that maintain both an EKF and a minimization-based estimator for robustness and/or efficiency have been proposed (Brockers et al., 2012; Weiss et al., 2012).
2. Throughout this paper, the preceding superscript (e.g. G in ${}^G\mathbf{p}_\ell$) denotes the frame of reference with respect to which quantities are expressed. ${}^A_B\mathbf{R}$ is the rotation matrix rotating vectors from frame $\{B\}$ to $\{A\}$, ${}^A_B\bar{\mathbf{q}}$ is the unit quaternion corresponding to the rotation matrix ${}^A_B\mathbf{R}$, $\lfloor \mathbf{c}\times \rfloor$ denotes the skew symmetric matrix corresponding to vector $\mathbf{c}$, $\mathbf{0}$ and $\mathbf{I}$ are the zero and identity matrices, respectively, $\hat{a}$ and $\tilde{a}$ represent the estimate and error of the estimate of a variable a respectively, and $\hat{a}_{i|j}$ is the estimate of variable a at time step i given measurements up to time step j.
3. We note that throughout this paper, we focus on the monocular-camera case, which is the more challenging one. However, our theoretical analysis and the MSCKF 2.0 algorithm are equally applicable to the case where a stereo pair is used for visual sensing.
4. Note that, if the signals are known to be band-limited, more advanced signal-reconstruction methods can be employed. However, this requires additional assumptions about the motion characteristics and/or the sensor, which are not always appropriate.
5. If we include in the state additional quantities that are known to be observable, this will augment the observability matrix (21) with additional, linearly independent, columns and will not affect the dimension of the nullspace of $\mathcal{O}$.

References

Bar-Shalom Y, Li XR and Kirubarajan T (2001) *Estimation with Applications to Tracking and Navigation*. New York: John Wiley & Sons.

Brockers R, Susca S, Zhu D and Matthies L (2012) Fully self-contained vision-aided navigation and landing of a micro air vehicle independent from external sensor inputs, volumn 8387.

Civera J, Davison A and Montiel J (2008) Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics* 24: 932–945.

Diel DD, DeBitetto P and Teller S (2005) Epipolar constraints for vision-aided inertial navigation. In: *IEEE Workshop on Motion and Video Computing*, Breckenridge, CO, pp. 221–228.

Dong-Si T and Mourikis AI (2011) Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 5655–5662.

Engels C, Stewenius H and Nister D (2006) Bundle adjustment rules. In: *Proceedings of the Photogrammetric Computer Vision Conference*, Bonn, Germany, pp. 266–271.

Farrell J (2008) *Aided navigation: GPS with high rate sensors*. New York: McGraw-Hill.

Foxlin E (2005) Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications* 25(6): 38–46.

Hesch JA, Kottas DG, Bowman SL, and Roumeliotis, SI (2012). Towards consistent vision-aided inertial navigation. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, Cambridge, MA.

Huang GP, Mourikis AI and Roumeliotis SI (2010) Observability-based rules for designing consistent EKF SLAM estimators. *The International Journal of Robotics Research* 29: 502–529.

Huang GP, Mourikis AI and Roumeliotis SI (2008) Analysis and improvement of the consistency of extended Kalman filter-based SLAM. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, CA, pp. 473–479.

Jones E and Soatto S (2011) Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research* 30: 407–430.

Kay SM (1993) *Fundamentals of Statistical Signal Processing, Vol. 1: Estimation Theory*. Englewood Cliffs, NJ: Prentice Hall.

Kelly J and Sukhatme G (2008) Fast relative pose calibration for visual and inertial sensors. In: *Proceedings of the International Symposium of Experimental Robotics*, Athens, Greece, pp. 515–524.

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

Kim J and Sukkarieh S (2007) Real-time implementation of airborne inertial-SLAM. *Robotics and Autonomous Systems* 55: 62–71.

Kleinert M and Schleith S (2010) Inertial aided monocular SLAM for GPS-denied navigation. In: *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Salt Lake City, UT, pp. 20–25.

Konolige K and Agrawal M (2008) FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics* 24: 1066–1077.

Konolige K, Agrawal M and Sola J (2011) Large-scale visual odometry for rough terrain. In: *Proceedings of the International Symposium of Robotics Research*, Flagstaff, AZ, pp. 201–212.

Kottas DG, Hesch JA, Bowman SL, and Roumeliotis, SI (2012). On the consistency of vision-aided inertial navigation. In *Proceedings of the International Symposium on Experimental Robotics*, Quebec City, Canada.

Li M and Mourikis AI (2011) *Consistency of EKF-based Visual–Inertial Odometry*. Technical report, University of California Riverside. http:// www.ee.ucr.edu/∼ mourikis/tech_reports/VIO.pdf.

Li M and Mourikis AI (2012a) Improving the accuracy of EKF-based visual-inertial odometry. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, St Paul, MN, pp. 828–835.

Li M and Mourikis AI (2012b). Optimization-based estimator design for vision-aided inertial navigation. *In Proceedings of Robotics: Science and Systems*, Sydney, Australia.

Lupton T and Sukkarieh S (2012) Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics* 28: 61–76.

Ma J, Susca S, Bajracharya M, Matthies L, Malchano M and Wooden D (2012) Robust multi-sensor, day/night 6-DOF pose estimation for a dynamic legged vehicle in GPS-denied environments. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, St Paul, MN, pp. 619–626.

Martinelli A (2012) Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics* 28: 44–60.

Maybeck PS (1979) *Stochastic Models, Estimation, and Control* (*Mathematics in Science and Engineering*, vol. 141-1). London: Academic Press.

Maybeck PS (1982) *Stochastic Models, Estimation and Control* (*Mathematics in Science and Engineering*, vol. 141-2). London: Academic Press.

Meyer CD (2000) *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Mirzaei FM and Roumeliotis SI (2008) A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics* 24: 1143–1156.

Mourikis AI and Roumeliotis SI (2007) A multi-state constraint Kalman filter for vision-aided inertial navigation. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy, pp. 3565–3572.

Mourikis AI and Roumeliotis SI (2008) A dual-layer estimator architecture for long-term localization. In: *Proceedings of the Workshop on Visual Localization for Mobile Platforms, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, pp. 3565–3572.

Mourikis AI, Roumeliotis SI and Burdick JW (2007) SC-KF mobile robot localization: A stochastic cloning-Kalman filter for processing relative-state measurements. *IEEE Transactions on Robotics* 23: 717–730.

Munguia R and Grau A (2007) Monocular SLAM for visual odometry. In: *Proceedings of the IEEE International Symposium on Intelligent Signal Processing*, Alcala Henares, Spain, pp. 1–6.

Nister D, Naroditsky O and Bergen J (2004) Visual odometry. In: *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Washington, DC, pp. 652–659.

Oskiper T, Zhiwei Z, Samarasekera S and Kumar R (2007) Visual odometry system using multiple stereo cameras and inertial measurement unit. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, pp. 1–8.

Pinies P, Lupton T, Sukkarieh S and Tardos J (2007) Inertial aiding of inverse depth SLAM using a monocular camera. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy, pp. 2797–2802.

Roumeliotis SI, Johnson AE and Montgomery JF (2002) Augmenting inertial navigation with image-based motion estimation. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, pp. 4326–4333.

Shi J and Tomasi C (1994) Good features to track. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Seattle, WA, pp. 593–600.

Shkurti F, Rekleitis I, Scaccia M and Dudek G (2011) State estimation of an underwater robot using visual and inertial information. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, pp. 5054–5060.

Sibley G, Matthies L and Sukhatme G (2010) Sliding window filter with application to planetary landing. *Journal of Field Robotics* 27: 587–608.

Simpson R, Cullip J and Revell J (2011) *The Cheddar Gorge data set*. Technical report, BAE Systems. http://openslam.org/misc/BAE_RSJCJR_2011.pdf

Sola J (2010) Consistency of the monocular EKF-SLAM algorithm for three different landmark parametrizations. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, AK, pp. 3513–3518.

Strasdat H, Montiel J and Davison A (2010) Real-time monocular SLAM: Why filter? In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, pp. 2657–2664.

Tardif J, George M, Laverne M, Kelly A and Stentz A (2010) A new approach to vision-aided inertial navigation. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taibei, China, pp. 4161–4168.

Trawny N, Mourikis AI, Roumeliotis SI, Johnson AE and Montgomery J (2007) Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks. *Journal of Field Robotics* 24: 357–378.

Trawny N and Roumeliotis S (2005) *Indirect Kalman Filter for 6D Pose Estimation*. Technical Report, University of Minnesota, Department of Computer Science and Engineering.

Vu A, Ramanandan A, Chen A, Farrell J and Barth M (2012) Real-time computer vision/DGPS-aided inertial navigation system for lane-level vehicle navigation. *IEEE Transactions on Intelligent Transportation Systems* 13: 899–913.

Weiss S, Achtelik M, Lynen S, Chli M and Siegwart R (2012) Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environment. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, St Paul, MN, pp. 957–964.

Weiss S and Siegwart R (2011) Real-time metric state estimation for modular vision–inertial systems. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 4531–4537.

Williams B, Hudson N, Tweddle B, Brockers R and Matthies L (2011) Feature and pose constrained visual aided inertial navigation for computationally constrained aerial vehicles. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 5655–5662.

Wu A, Johnson E and Proctor A (2005) Vision-aided inertial navigation for flight control. *AIAA Journal of Aerospace Computing, Information, and Communication* 2: 348–360.

Zachariah D and Jansson M (2010) Camera-aided inertial navigation using epipolar points. In: *Proceedings of the IEEE/ION Symposium on Position Location and Navigation*, Indian Wells, CA, pp. 303–309.

Appendix A

If the biases are included, the IMU error-state transition matrix is given by

$$\mathbf{\Phi}_{I_k} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{\Phi}_{\mathbf{qb}_g} & \mathbf{0}_3 \\ \mathbf{\Phi}_{\mathbf{pq}} & \mathbf{I}_3 & \Delta t\mathbf{I}_3 & \mathbf{\Phi}_{\mathbf{pb}_g} & \mathbf{\Phi}_{\mathbf{pa}} \\ \mathbf{\Phi}_{\mathbf{vq}} & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{\Phi}_{\mathbf{vb}_g} & \mathbf{\Phi}_{\mathbf{va}} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{60}$$

where

$$\begin{aligned}
\mathbf{\Phi}_{\mathbf{qb}_g} &= -\hat{\mathbf{R}}_\ell^{\mathrm{T}} \int_{t_\ell}^{t_{\ell+1}} {}^{I_\ell}_{I_\tau}\hat{\mathbf{R}}\, d\tau \\
\mathbf{\Phi}_{\mathbf{pb}_g} &= \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{w} \lfloor ({}^{G}\dot{\hat{\mathbf{v}}}_\tau - \mathbf{g}) \times \rfloor \hat{\mathbf{R}}_\ell^{\mathrm{T}} \int_{t_\ell}^{\tau} {}^{I_\ell}_{I_s}\hat{\mathbf{R}}\, ds\, d\tau\, dw \\
\mathbf{\Phi}_{\mathbf{pa}} &= -\hat{\mathbf{R}}_\ell^{\mathrm{T}} \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{\tau} {}^{I_\ell}_{I_s}\hat{\mathbf{R}}\, ds\, d\tau \\
\mathbf{\Phi}_{\mathbf{vb}_g} &= \int_{t_\ell}^{t_{\ell+1}} \lfloor ({}^{G}\dot{\hat{\mathbf{v}}}_\tau - \mathbf{g}) \times \rfloor \hat{\mathbf{R}}_\ell^{\mathrm{T}} \int_{t_\ell}^{\tau} {}^{I_\ell}_{I_s}\hat{\mathbf{R}}\, ds\, d\tau \\
\mathbf{\Phi}_{\mathbf{va}} &= -\hat{\mathbf{R}}_\ell^{\mathrm{T}} \int_{t_\ell}^{t_{\ell+1}} {}^{I_\ell}_{I_\tau}\hat{\mathbf{R}}\, d\tau
\end{aligned} \tag{61}$$

The detailed derivation of this result is provided in (Li and Mourikis, 2011).

Appendix B

We here prove that, in a linear-Gaussian system, the state estimate and covariance matrix computed by the MSCKF is the MAP estimate for the IMU pose. Since EKF-SLAM is also a MAP estimator, this means that the MSCKF and EKF-SLAM would be identical in a linear-Gaussian system. Due to limited space, we here provide an outline of the main steps of the proof, and the full details are provided in (Li and Mourikis, 2011).

Let us consider the following linear system:

$$\mathbf{x}_i = \mathbf{\Phi}_i \mathbf{x}_{i-1} + \mathbf{w}_{i-1} \tag{62}$$

$$\mathbf{z}_{ij} = \mathbf{H}_{\mathbf{x}_{ij}} \mathbf{x}_i + \mathbf{H}_{\mathbf{f}_{ij}} \mathbf{p}_{f_j} + \mathbf{n}_{ij} \tag{63}$$

where $\mathbf{x}_i$, $i = 0 \ldots N$ are the IMU states, $\mathbf{p}_{f_j}$, $j = 1 \ldots M$ are the feature positions, $\mathbf{w}_i$ and $\mathbf{n}_{ij}$ are zero-mean white Gaussian noise processes with covariance matrices $\mathbf{Q}_i$ and $\sigma^2\mathbf{I}_2$, respectively, and $\mathbf{\Phi}_i$, $\mathbf{H}_{\mathbf{x}_{ij}}$, and $\mathbf{H}_{\mathbf{f}_{ij}}$ are known matrices. We denote the vector containing all of the IMU states as $\mathbf{x} = \begin{bmatrix} \mathbf{x}_0^{\mathrm{T}} & \mathbf{x}_1^{\mathrm{T}} & \cdots & \mathbf{x}_N^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, the vector containing all of the feature positions as $\mathbf{f} = \begin{bmatrix} \mathbf{f}_1^{\mathrm{T}} & \mathbf{f}_2^{\mathrm{T}} & \cdots & \mathbf{f}_M^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, and the vector containing all measurements as

$$\mathbf{z} = \mathbf{H}_\mathbf{x}\mathbf{x} + \mathbf{H}_\mathbf{f}\mathbf{f} + \mathbf{n}$$

where $\mathbf{H}_\mathbf{x}$ and $\mathbf{H}_\mathbf{f}$, are matrices with block rows $\mathbf{H}_{\mathbf{x}_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$, respectively.

Using the prior estimate for the first state, $\hat{\mathbf{x}}_0$, as well as the state propagation equation, we can obtain an estimate for $\mathbf{x}$, which we denote by $\hat{\mathbf{x}}_s$, along with its covariance matrix $\mathbf{P}_s$. This estimate uses all of the information from the prior and the state propagation model. The MAP estimate for the entire state vector, which uses all the available information (prior, propagation model, and measurements) is given by

$$\begin{bmatrix}\hat{\mathbf{x}}_{\text{MAP}}\\ \hat{\mathbf{f}}_{\text{MAP}}\end{bmatrix} = \mathbf{\Lambda}^{-1}\begin{bmatrix}\mathbf{P}_s^{-1}\hat{\mathbf{x}}_s + \frac{1}{\sigma^2}\mathbf{H}_\mathbf{x}^\text{T}\mathbf{z}\\ \frac{1}{\sigma^2}\mathbf{H}_\mathbf{f}^\text{T}\mathbf{z}\end{bmatrix}$$

where $\mathbf{\Lambda}$ is the information matrix:

$$\mathbf{\Lambda} = \begin{bmatrix}\mathbf{P}_s^{-1} + \frac{1}{\sigma^2}\mathbf{H}_\mathbf{x}^\text{T}\mathbf{H}_\mathbf{x} & \frac{1}{\sigma^2}\mathbf{H}_\mathbf{x}^\text{T}\mathbf{H}_\mathbf{f}\\ \frac{1}{\sigma^2}\mathbf{H}_\mathbf{f}^\text{T}\mathbf{H}_\mathbf{x} & \frac{1}{\sigma^2}\mathbf{H}_\mathbf{f}^\text{T}\mathbf{H}_\mathbf{f}\end{bmatrix}$$

and $\mathbf{\Lambda}^{-1}$ is the covariance matrix of the MAP estimate. Using the standard properties of the inversion of a partitioned matrix, we can show that the estimate $\hat{\mathbf{x}}_{\text{MAP}}$ and its covariance matrix equal

$$\hat{\mathbf{x}}_{\text{MAP}} = \mathbf{P}_{\text{MAP}}\left(\mathbf{P}_s^{-1}\hat{\mathbf{x}}_s + \frac{1}{\sigma^2}\mathbf{H}_\mathbf{x}^\text{T}\left(\mathbf{I} - \mathbf{H}_\mathbf{f}\left(\mathbf{H}_\mathbf{f}^\text{T}\mathbf{H}_\mathbf{f}\right)^{-1}\mathbf{H}_\mathbf{f}^\text{T}\right)\mathbf{z}\right)$$

$$\mathbf{P}_{\text{MAP}} = \left(\mathbf{P}_s^{-1} + \frac{1}{\sigma^2}\mathbf{H}_\mathbf{x}^\text{T}\left(\mathbf{I} - \mathbf{H}_\mathbf{f}\left(\mathbf{H}_\mathbf{f}^\text{T}\mathbf{H}_\mathbf{f}\right)^{-1}\mathbf{H}_\mathbf{f}^\text{T}\right)\mathbf{H}_\mathbf{x}\right)^{-1}$$

On the other hand, in the MSCKF algorithm, if we use the IMU measurements to propagate the state estimates, and then employ the camera measurements for an update, the update is performed based on the residual:

$$\mathbf{r}_o \doteq \mathbf{V}^\text{T}(\mathbf{z} - \mathbf{H}_\mathbf{x}^\text{T}\hat{\mathbf{x}}_s) = \left(\mathbf{V}^\text{T}\mathbf{H}_\mathbf{x}\right)\tilde{\mathbf{x}}_s + \mathbf{n}_o \tag{64}$$

where $\mathbf{V}$ is a matrix whose columns form an orthonormal basis for the left nullspace of $\mathbf{H}_\mathbf{f}$, and $\mathbf{n}_o$ is a noise vector with covariance matrix $\sigma^2\mathbf{I}$. Using the Kalman filter equations, the state and covariance update can be written as

$$\hat{\mathbf{x}}_{\text{MSC}} = \hat{\mathbf{x}}_s + \mathbf{K}\mathbf{r}_o \tag{65}$$

$$\mathbf{P}_{\text{MSC}} = \left(\mathbf{P}_s^{-1} + \frac{1}{\sigma^2}\left(\mathbf{V}^\text{T}\mathbf{H}_\mathbf{x}\right)^\text{T}\left(\mathbf{V}^\text{T}\mathbf{H}_\mathbf{x}\right)\right)^{-1} \tag{66}$$

where $\mathbf{K}$ is the Kalman gain, which can be written as (Maybeck, 1979):

$$\mathbf{K} = \frac{1}{\sigma^2}\mathbf{P}_{\text{MSC}}\left(\mathbf{V}^\text{T}\mathbf{H}_\mathbf{x}\right)^\text{T} \tag{67}$$

Our goal is to show that $\hat{\mathbf{x}}_{\text{MSC}} = \hat{\mathbf{x}}_{\text{MAP}}$, and $\mathbf{P}_{\text{MSC}} = \mathbf{P}_{\text{MAP}}$. To this end, we note that the matrix $\mathbf{I} - \mathbf{H}_\mathbf{f}\left(\mathbf{H}_\mathbf{f}^\text{T}\mathbf{H}_\mathbf{f}\right)^{-1}\mathbf{H}_\mathbf{f}^\text{T}$ is the orthogonal projector onto the left nullspace of $\mathbf{H}_\mathbf{f}$, and thus $\mathbf{I} - \mathbf{H}_\mathbf{f}\left(\mathbf{H}_\mathbf{f}^\text{T}\mathbf{H}_\mathbf{f}\right)^{-1}\mathbf{H}_\mathbf{f}^\text{T} = \mathbf{V}\mathbf{V}^\text{T}$ (Meyer, 2000). Using this result, the equality $\mathbf{P}_{\text{MSC}} = \mathbf{P}_{\text{MAP}}$ follows immediately, and we can also write

$$\hat{\mathbf{x}}_{\text{MAP}} = \mathbf{P}_{\text{MSC}}\left(\mathbf{P}_s^{-1}\hat{\mathbf{x}}_s + \frac{1}{\sigma^2}\mathbf{H}_\mathbf{x}^\text{T}\mathbf{V}\mathbf{V}^\text{T}\mathbf{z}\right) \tag{68}$$

Substitution of (64) and (67) in (65) yields

$$\begin{aligned}\hat{\mathbf{x}}_{\text{MSC}} &= \hat{\mathbf{x}}_s + \frac{1}{\sigma^2}\mathbf{P}_{\text{MSC}}\left(\mathbf{V}^\text{T}\mathbf{H}_\mathbf{x}\right)^\text{T}\mathbf{V}^\text{T}(\mathbf{z} - \mathbf{H}_\mathbf{x}^\text{T}\hat{\mathbf{x}}_s)\\ &= \mathbf{P}_{\text{MSC}}\left(\left(\mathbf{P}_{\text{MSC}}^{-1} - \frac{1}{\sigma^2}\mathbf{H}_\mathbf{x}^\text{T}\mathbf{V}\mathbf{V}^\text{T}\mathbf{H}_\mathbf{x}^\text{T}\right)\hat{\mathbf{x}}_s + \frac{1}{\sigma^2}\mathbf{H}_\mathbf{x}^\text{T}\mathbf{V}\mathbf{V}^\text{T}\mathbf{z}\right)\end{aligned}$$

Showing that the last equation is equal to (68) follows immediately by use of (66).

Appendix C

In EKF-based SLAM, the current feature estimates are used for computing the measurement Jacobians at each time step. Thus, we have

$$\begin{aligned}\mathbf{H}_{I_{i\ell}} &= \mathbf{J}_{i\ell}\,{}^C_I\mathbf{R}\,\hat{\mathbf{R}}_{\ell|\ell-1}\left[\lfloor\left({}^G\hat{\mathbf{p}}_{f_{i|\ell-1}} - {}^G\hat{\mathbf{p}}_{\ell|\ell-1}\right)\times\rfloor\ -\mathbf{I}_3\ \mathbf{0}_3\right]\\ \mathbf{H}_{\mathbf{f}_{i\ell}} &= \mathbf{J}_{i\ell}\,{}^C_I\mathbf{R}\,\hat{\mathbf{R}}_{\ell|\ell-1}\end{aligned} \tag{69}$$

where the matrix $\mathbf{J}_{i\ell}$ is evaluated using the estimate:

$${}^{C_\ell}\hat{\mathbf{p}}_{f_{i|\ell-1}} = {}^C_I\mathbf{R}\,\hat{\mathbf{R}}_{\ell|\ell-1}\left({}^G\hat{\mathbf{p}}_{f_{i|\ell-1}} - {}^G\hat{\mathbf{p}}_{\ell|\ell-1}\right) + {}^C\mathbf{p}_I \tag{70}$$

Using these Jacobians, we obtain the block row of the observability matrix corresponding to the observation of feature i at time-step ℓ. This matrix has the same structure as (48), with

$$\begin{aligned}\mathbf{\Gamma}_{i\ell} &= \Big\lfloor {}^G\hat{\mathbf{p}}_{f_{i|\ell-1}} - {}^G\hat{\mathbf{p}}_{k|k} - {}^G\hat{\mathbf{v}}_{k|k}\Delta t_\ell - \frac{1}{2}\mathbf{g}\Delta t_\ell^2 \times \Big\rfloor\\ \Delta\mathbf{\Gamma}_{i\ell} &= \sum_{j=k+1}^{\ell-1}\left(\mathbf{E}_\mathbf{p}^j + \sum_{s=k+1}^{j}\mathbf{E}_\mathbf{v}^s\Delta t\right) + \left\lfloor\Delta^G\mathbf{p}_{f_i}\times\right\rfloor\\ \mathbf{M}_{i\ell} &= \mathbf{J}_{i\ell}\,{}^C_I\mathbf{R}\,\hat{\mathbf{R}}_{\ell|\ell}\end{aligned} \tag{71}$$

$$\Delta^G\mathbf{p}_{f_i} = \lfloor {}^G\hat{\mathbf{p}}_{f_{i|\ell-1}} - {}^G\hat{\mathbf{p}}_{f_{i|k-1}}\times\rfloor \tag{72}$$

$$\mathbf{E}_\mathbf{p}^j = \lfloor {}^G\hat{\mathbf{p}}_{j|j-1} - {}^G\hat{\mathbf{p}}_{j|j}\times\rfloor \tag{73}$$

$$\mathbf{E}_\mathbf{v}^j = \lfloor {}^G\hat{\mathbf{v}}_{j|j-1} - {}^G\hat{\mathbf{v}}_{j|j}\times\rfloor \tag{74}$$

Similarly to the MSCKF, the observability matrix of EKF-based SLAM also contains a disturbance term $\Delta\mathbf{\Gamma}_{i\ell}$, which decreases the dimension of the unobservable subspace.