2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Tightly-Coupled Monocular Visual-Inertial Fusion for Autonomous Flight of Rotorcraft MAVs

Shaojie Shen, Nathan Michael, and Vijay Kumar

***Abstract*— There have been increasing interests in the robotics community in building smaller and more agile autonomous micro aerial vehicles (MAVs). In particular, the monocular visual-inertial system (VINS) that consists of only a camera and an inertial measurement unit (IMU) forms a great minimum sensor suite due to its superior size, weight, and power (SWaP) characteristics. In this paper, we present a tightly-coupled nonlinear optimization-based monocular VINS estimator for autonomous rotorcraft MAVs. Our estimator allows the MAV to execute trajectories at 2 m/s with roll and pitch angles up to 30 degrees. We present extensive statistical analysis to verify the performance of our approach in different environments with varying flight speeds.**

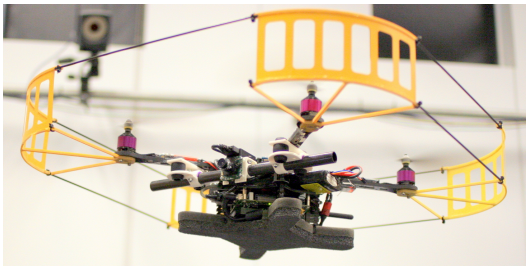


Fig. 1. Our quadrotor experimental platform equipped with an Intel NUC computer, a MEMS IMU, and a camera.

I. Introduction

Sensor-equipped micro aerial vehicles (MAVs) with autonomous flight capability are ideal platforms for missions in complex and confined environments due to its small size, superior mobility, and minimum operator workload. While it is obvious that reducing the size of the MAV will make it more suitable for operations in confined environments, it also poses tighter SWaP constrains. As the size scales down, a monocular visual-inertial system (VINS) that consists of a camera and a low cost IMU becomes the only viable setup due to its ultra light weight and small footprint. In fact, monocular VINS is the minimum sensor suite that allows both autonomous flight and sufficient environment awareness.

In this work, we propose a tightly-coupled optimization-based monocular VINS estimator that enables autonomous flight of rotorcraft MAVs in unstructured and unknown environments. This work builds on and is a substantial improvement from our recent work on linear initialization of monocular VINS [1]. We identify the contribution of this work as threefold: 1) The development of a tightly-coupled nonlinear sliding window optimization framework for optimal fusion of IMU and monocular camera measurements; 2) The integration of initialization and marginalization schemes from our recent work into a complete monocular VINS estimator with analysis of choices of parameters; and 3) Online experiments of autonomous flight through a variety of environments with statistical analysis of estimator and flight performance.

To the best of our knowledge, we are the first to demonstrate autonomous flight with a monocular VINS sensor suite using a tightly-coupled fusion approach. We are able to achieve autonomous tracking of highly dynamical trajectories with a maximum velocity of 2 m/s and maximum roll/pitch angles close to 30 degrees. Statistical analysis (Sect. VI) suggests good performance with different speeds and in different environments.

Next, we discuss literatures in similar fields in Sect. II. In Sect. III, we recap the basis of initialization of monocular VINS. The derivation of the tightly-coupled optimization framework is presented in Sect. IV. This is followed by a discuss of a two-way marginalization scheme for handling degenerate motions that are unique to monocular VINS (Sect. V). Sect. VI first discusses hardware setup and software implementation details, then presents multiple online experiments with performance analysis.

II. Related Work

Solutions to VINS with either monocular or stereo cameras has been proposed using filtering frameworks [2]–[9] and with graph-based optimization/bundle adjustment frameworks [10]–[12]. Filtering-based approaches may achieve faster processing due to its continuous marginalization of past states, but early fix of linearization points may results in sub-optimal results. Graph-based approaches benefit from iterative re-linearization but they usually demand more computational resources. With proper marginalization, a constant complexity sliding window graph-based framework can be obtained [10]. Conditioning is also a popular method among the computer vision community to achieve constant computation complexity [13, 14].

S. Shen is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. eeshaojie@ust.hk

N. Michael is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. nmichael@cmu.edu

V. Kumar is with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA. kumar@grasp.upenn.edu

We gratefully acknowledge support from ARL grant W911NF-08-2-0004, ONR grants N00014-07-1-0829 and N00014-09-1-1051, and NSF grant IIP-1113830.

We can also categorize VINS solutions as loosely coupled [2, 3] or tightly coupled [4]–[9, 11, 12]. Loosely coupled approaches utilize independent vision processing modules such as PTAM [13] for up-to-scale pose estimation, and integrate results from the vision module with IMU for scale estimation. Tightly coupled approaches usually lead to better estimation results due to the direct and systematic fusion of feature and IMU measurements.

When it comes to VINS with only one camera, there is a unique challenge in scale ambiguity due to degenerate motion. It is well known that in order to render the scale observable, accelerations in at least two axes are required [5, 6, 15]. However, for a rotorcraft MAV, degenerate motions such as hovering and constant velocity motions are unavoidable. The hover case can be addressed via conditioning in a keyframe-based loosely coupled approach [13], or with a last-in-first-out (LIFO) scheme in a tightly-coupled sliding window approach [9].

The monocular VINS sensor suite has been used for autonomous flight. In [2, 3], a monocular SLAM framework is used as the main vision processing pipeline. In conjunction with a loosely coupled filtering framework, these approaches successfully enable a quadrotor to fly autonomously with a downward-facing camera. However, due to the lack of direct scale measurement, these approaches relies on the assumption of slowly-varying or good initialization of the visual scale. This can be difficult to enforce during fast motions at low-altitudes with potentially rapid changes in the observed features.

III. On-the-Fly Initialization

A good initialization point is required for solving the highly nonlinear monocular VINS system. In practice, however, this initialization point is usually hard to obtain due to the fact that the metric scale of the monocular VINS system is not directly observable. In order to initialize the metric scale, motions that consist of nonzero acceleration are required. For a MAV, this often results in unknown and nontrivial initial velocity and attitude (gravity vector). As such, we require that the system is capable of on-the-fly initialization to recover all critical states such as velocity, gravity vector, and depth of features. The problem can be formulated as solving two sets of linear systems and it is discussed in our earlier work [1]. Here we briefly recap the rationale behind our approach.

We begin by defining notations. We consider $(\cdot)^w$ as the earth's inertial frame, $(\cdot)^b$ as the current IMU body frame, $(\cdot)^k$ as the camera frame while taking the k^{th} image, Note that IMU usually runs at a higher rate than the camera, and that multiple IMU measurements may exist in the interval $[k, k+1]$. We assume that the camera and the IMU are pre-calibrated such that the camera optical axis is aligned with the z-axis of the IMU. $\mathbf{p}_Y^X$, $\mathbf{v}_Y^X$, and $\mathbf{R}_Y^X$ are 3D position , velocity, and rotation of frame Y with respect to frame X. In particular, $\mathbf{p}_t^X$ represents the position of the body frame at time t with respect to frame X. Similar conversion follows for other parameters. $\mathbf{g}^w = [0,\, 0,\, g]^{\mathrm{T}}$ is the gravity vector in the world frame, and $\mathbf{g}^k$ is the earth's gravity vector expressed in the body frame of the k^{th} image.

Given two time instants (corresponding to two image frames), the IMU propagation model for position and velocity, expressed in the world frame, can be written as:

$$\begin{aligned}\mathbf{p}_{k+1}^w &= \mathbf{p}_k^w + \mathbf{v}_k^w \Delta t + \iint_{t\in[k,k+1]} \left(\mathbf{R}_t^w \mathbf{a}_t^b - \mathbf{g}^w\right) dt^2 \\ \mathbf{v}_{k+1}^w &= \mathbf{v}_k^w + \int_{t\in[k,k+1]} \left(\mathbf{R}_t^w \mathbf{a}_t^b - \mathbf{g}^w\right) dt\end{aligned} \tag{1}$$

where $\mathbf{a}_t^b$ is the linear acceleration in the body frame, Δt is the time difference between k and $k+1$. It can be seen that the rotation between the world frame and the body frame is required in order to propagate the states with IMU measurements. This rotation can only be determined if the initial attitude of the vehicle is known, which is not the case during the initialization phase of monocular VINS. However, as suggested in [16], if the reference frame of the IMU propagation model is attached to the first pose of the system (i.e. the first pose that we are trying to estimate), (1) can be rewritten as:

$$\begin{aligned}\mathbf{p}_{k+1}^0 &= \mathbf{p}_k^0 + \mathbf{R}_k^0 \mathbf{v}_k^k \Delta t - \mathbf{R}_k^0 \mathbf{g}^k \Delta t^2/2 + \mathbf{R}_k^0 \boldsymbol{\alpha}_{k+1}^k \\ \mathbf{v}_{k+1}^{k+1} &= \mathbf{R}_k^{k+1} \mathbf{v}_k^k - \mathbf{R}_k^{k+1} \mathbf{g}^k \Delta t + \mathbf{R}_k^{k+1} \boldsymbol{\beta}_{k+1}^k \\ \mathbf{g}^{k+1} &= \mathbf{R}_k^{k+1} \mathbf{g}^k\end{aligned} \tag{2}$$

where $\boldsymbol{\alpha}_{k+1}^k$ and $\boldsymbol{\beta}_{k+1}^k$ can be obtained solely with IMU measurements within $[k, k+1]$, $\mathbf{R}_k^0$ is the change in rotation since the first pose (or since the 0^{th} image), and $\mathbf{R}_{k+1}^k$ is the incremental rotation between two images. By decoupling optimization process of rotation and other quantities, it is possible to recover all initial states in the monocular VINS system in a linear fashion. More specifically, rotation can be obtained by solving a linear system that incorporate short-term integration of gyroscope measurements and relative epipolar constraints. After the rotation is fixed, all other IMU states ($\mathbf{p}_k^0$, $\mathbf{v}_k^k$, $\mathbf{g}^k$), as well as depth of features, can also be solved linearly. We refer readers to [1] for details of this on-the-fly initialization process.

IV. Tightly-Coupled Nonlinear Optimization

After the monocular VINS is initialized with the linear approach [1], we are able to use a tightly-coupled nonlinear optimization framework to jointly optimize both the translation and rotation components of the system.

A. Formulation

We use a tightly-coupled, sliding window graph-based formulation for nonlinear optimization to achieve both high accuracy and maintain bounded computation complexity. The *full state* vector is defined as (the transpose is ignored for the simplicity of presentation):

$$\begin{aligned}\mathcal{X} &= \left[\mathbf{x}_n^0,\, \mathbf{x}_{n+1}^0,\, \cdots \mathbf{x}_{n+N}^0,\, \lambda_0,\, \lambda_{m+1},\, \cdots \lambda_{m+M}\right] \\ \mathbf{x}_k^0 &= \left[\mathbf{p}_k^0,\, \mathbf{v}_k^k,\, \mathbf{q}_k^0\right] \\ \mathbf{p}_0^0 &= [0,\, 0,\, 0]\,, \quad \mathbf{q}_0^0 = [0,\, 0,\, 0,\, 1]\end{aligned}$$

where $\mathbf{x}_k^0$ is the k^{th} *camera state* that consists of the pose with respect to the first camera pose, as well as the body frame velocity. We use quaternions ($\mathbf{q} = [q_x,\, q_y,\, q_z,\, q_w]$) to represent rotation in order to avoid singularities. The Hamilton notation is used for quaternions. Note the dimension of the state vector is not the same as the dimension of the degree-of-freedom of the system. The three-dimensional rotation is over-parameterized by the four-dimensional quaternion. N is the number of camera states in the sliding window, M is the number of all features that have sufficient parallax within the sliding window. n and m are starting indexes of states in the sliding window. λ_l is the depth of the l^{th} point feature from its first observation.

We aim to find a configuration of the state parameters that produce the maximum a posteriori estimate by minimizing the sum of the Mahalanobis norm of all measurement errors:

$$\min_{\mathcal{X}} \left\{ (\mathbf{b}_p - \mathbf{\Lambda}_p \mathcal{X}) + \sum_{k \in \mathcal{D}} \left\| r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k,\, \mathcal{X}) \right\|_{\mathbf{P}_{k+1}^k}^2 + \sum_{(l,j) \in \mathcal{C}} \left\| r_{\mathcal{C}}(\hat{\mathbf{z}}_l^j,\, \mathcal{X}) \right\|_{\mathbf{P}_l^j}^2 \right\} \tag{3}$$

where $\mathbf{\Lambda}_p$ is the prior, $\mathcal{D}$ and $\mathcal{C}$ are indexes of the set of IMU and camera measurements, with the corresponding residuals defined as $r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k,\, \mathcal{X})$ and $r_{\mathcal{C}}(\hat{\mathbf{z}}_l^j,\, \mathcal{X})$. These quantities will be will be derived in Sect. IV-B and Sect. IV-C respectively.

Although the residuals for position, velocity, and feature depth can be easily defined:

$$\begin{aligned} \mathbf{p} &= \hat{\mathbf{p}} + \delta\mathbf{p} \\ \mathbf{v} &= \hat{\mathbf{v}} + \delta\mathbf{v} \\ \lambda &= \hat{\lambda} + \delta\lambda \end{aligned} \tag{4}$$

the residual for rotation is more involved. Similar to [11], we use the perturbation of the tangent space of the rotation manifold as the minimum-dimensional representation of the rotation residual. The error quaternion term $\delta\mathbf{q}$ is defined as the small difference between the estimated and the true quaternions:

$$\mathbf{q} = \hat{\mathbf{q}} \otimes \delta\mathbf{q}, \quad \delta\mathbf{q} \approx \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix} \tag{5}$$

where $\otimes$ is the quaternion multiplication operator. From this we can use the three-dimensional error vector $\delta\boldsymbol{\theta}$ as the representation of rotation residual. Similarly, we can write the error term in the form of rotation matrix:

$$\mathbf{R} \approx \hat{\mathbf{R}} \cdot (\mathbb{I} + \lfloor \delta\boldsymbol{\theta} \times \rfloor) \tag{6}$$

where $\lfloor \delta\boldsymbol{\theta} \times \rfloor$ is the skew-symmetric matrix from $\delta\boldsymbol{\theta}$.

Following this definition, we operate on the error state representation during the optimization:

$$\begin{aligned} \delta\mathcal{X} &= \left[\delta\mathbf{x}_n^0,\, \delta\mathbf{x}_{n+1}^0,\, \cdots \delta\mathbf{x}_{n+N}^0,\, \delta\lambda_m,\, \delta\lambda_{m+1},\, \cdots \delta\lambda_{m+M}\right] \\ \delta\mathbf{x}_k^0 &= \left[\delta\mathbf{p}_k^0,\, \delta\mathbf{v}_k^k,\, \delta\boldsymbol{\theta}_k^0\right] \end{aligned}$$

We linearize the cost function (3) with respect to $\delta\mathcal{X}$, and iteratively minimize the cost of the resulting linear system. Given the current best state estimates $\hat{\mathcal{X}}$, we have:

$$\min_{\delta\mathcal{X}} \left\{ \left(\mathbf{b}_p - \mathbf{\Lambda}_p \hat{\mathcal{X}}\right) + \sum_{k \in \mathcal{D}} \left\| r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k,\, \hat{\mathcal{X}}) + \mathbf{H}_{k+1}^k \delta\mathcal{X} \right\|_{\mathbf{P}_{k+1}^k}^2 + \sum_{(l,j) \in \mathcal{C}} \left\| r_{\mathcal{C}}(\hat{\mathbf{z}}_l^j,\, \hat{\mathcal{X}}) + \mathbf{H}_l^j \delta\mathcal{X} \right\|_{\mathbf{P}_l^j}^2 \right\} \tag{7}$$

where $\mathbf{H}_{k+1}^k$ and $\mathbf{H}_l^j$, which will also be defined in Sect. IV-B and Sect. IV-C, are the Jacobians of $r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k,\, \hat{\mathcal{X}})$ and $r_{\mathcal{C}}(\hat{\mathbf{z}}_l^j,\, \hat{\mathcal{X}})$ with respect to $\delta\mathcal{X}$, respectively. The system (7) can be rewritten and solved as:

$$(\mathbf{\Lambda}_p + \mathbf{\Lambda}_{\mathcal{D}} + \mathbf{\Lambda}_{\mathcal{C}})\, \delta\mathcal{X} = (\mathbf{b}_p + \mathbf{b}_{\mathcal{D}} + \mathbf{b}_{\mathcal{C}}) \tag{8}$$

after which the state estimates can be updated as:

$$\hat{\mathcal{X}} = \hat{\mathcal{X}} \oplus \delta\mathcal{X} \tag{9}$$

where $\oplus$ is the compound operator that has the form of simple addition for position, velocity, and feature depth as in (4), but is formulated as quaternion multiplication for rotations as in (5).

B. IMU Measurement Model

We now present the formulation for the IMU measurement $\hat{\mathbf{z}}_{k+1}^k$, the measurement covariance matrix $\mathbf{P}_{k+1}^k$, the residual $r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k,\, \mathcal{X})$, and the measurement Jacobian $\mathbf{H}_{k+1}^k$.

It should be first noted that since there are multiple accelerometer and gyroscope measurements between two images, the IMU measurement $\hat{\mathbf{z}}_{k+1}^k$ is a composition of multiple IMU readings.

$$\hat{\mathbf{z}}_{k+1}^k = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{k+1}^k \\ \hat{\boldsymbol{\beta}}_{k+1}^k \\ \hat{\mathbf{q}}_{k+1}^k \end{bmatrix} = \begin{bmatrix} \iint_{t \in [k,k+1]} \hat{\mathbf{R}}_t^k \hat{\mathbf{a}}_t^b dt^2 \\ \int_{t \in [k,k+1]} \hat{\mathbf{R}}_t^k \hat{\mathbf{a}}_t^b dt \\ \int_{t \in [k,k+1]} \mathbf{\Omega}(\hat{\boldsymbol{\omega}}_t^b) \hat{\mathbf{q}}_t^k dt \end{bmatrix} \tag{10}$$

where

$$\begin{aligned} \hat{\mathbf{a}}_t^b &= \mathbf{a}_t^b + {}^a\mathbf{n}_t \\ \hat{\boldsymbol{\omega}}_t^b &= \boldsymbol{\omega}_t^b + {}^\omega\mathbf{n}_t \end{aligned}$$

are accelerometer and gyroscope measurements that are corrupted with additive noise, and

$$\mathbf{\Omega}(\hat{\boldsymbol{\omega}}_t^b) = \frac{1}{2} \begin{bmatrix} -\lfloor \hat{\boldsymbol{\omega}}_t^b \times \rfloor & \hat{\boldsymbol{\omega}}_t^b \\ -\hat{\boldsymbol{\omega}}_t^{b\mathrm{T}} & \mathbf{0} \end{bmatrix}$$

$\hat{\mathbf{R}}_t^k$ can be derived from $\hat{\mathbf{q}}_t^k$. Again, while error terms for $\hat{\boldsymbol{\alpha}}_{k+1}^k$ and $\hat{\boldsymbol{\beta}}_{k+1}^k$ are still additive, since $\hat{\mathbf{q}}_{k+1}^k$ is over-parameterized, we define its error terms as the perturbation from the true value:

$$\mathbf{q}_{k+1}^k \approx \hat{\mathbf{q}}_{k+1}^k \otimes \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}_{k+1}^k \\ 1 \end{bmatrix} \tag{11}$$

With the approximated rotation matrix composition of the error term (6), we can derive the continuous-time linearized

dynamics of the error terms from (10) and (11):

$$\begin{bmatrix}\delta\dot{\boldsymbol{\alpha}}\\ \delta\dot{\boldsymbol{\beta}}\\ \delta\dot{\boldsymbol{\theta}}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbb{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & -\hat{\mathbf{R}}_t^k\lfloor\hat{\mathbf{a}}_t^b\times\rfloor\\ \mathbf{0} & \mathbf{0} & -\lfloor\hat{\boldsymbol{\omega}}_t^b\times\rfloor\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\alpha}_t^k\\ \delta\boldsymbol{\beta}_t^k\\ \delta\boldsymbol{\theta}_t^k\end{bmatrix} + \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ -\hat{\mathbf{R}}_t^k & \mathbf{0}\\ \mathbf{0} & -\mathbb{I}\end{bmatrix}\begin{bmatrix}{}^a\mathbf{n}_t\\ {}^\omega\mathbf{n}_t\end{bmatrix} = \mathbf{F}_t\delta\mathbf{z}_t^k + \mathbf{G}_t\mathbf{n}_t$$

from which we can derived the first-order discrete-time covariance update equation in order to recursively compute $\mathbf{P}_{k+1}^k$ with the initial covariance $\mathbf{P}_k^k = \mathbf{0}$:

$$\mathbf{P}_{t+\delta t}^k = (\mathbb{I}+\mathbf{F}_t\delta t)\cdot\mathbf{P}_t^k\cdot(\mathbb{I}+\mathbf{F}_t\delta t)^{\mathrm{T}} + (\mathbf{G}_t\delta t)\cdot\mathbf{Q}_t\cdot(\mathbf{G}_t\delta t)^{\mathrm{T}} \tag{12}$$

where $t\in[k,k+1]$, and δt is the time between two IMU measurements, and $\mathbf{Q}_t$ is the covariance matrix for IMU measurements.

Following (2) and (11), we can now define the measurement residual $r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X}) = \left[\delta\boldsymbol{\alpha}_{k+1}^k,\, \delta\boldsymbol{\beta}_{k+1}^k,\, \delta\boldsymbol{\theta}_{k+1}^k\right]^{\mathrm{T}}$ as:

$$\begin{bmatrix}\delta\boldsymbol{\alpha}_{k+1}^k\\ \delta\boldsymbol{\beta}_{k+1}^k\\ \delta\boldsymbol{\theta}_{k+1}^k\end{bmatrix} = \begin{bmatrix}\mathbf{R}_0^k\left(\mathbf{p}_{k+1}^0 - \mathbf{p}_k^0 + \mathbf{g}^0\frac{\Delta t^2}{2}\right) - \mathbf{v}_k^k\Delta t - \hat{\boldsymbol{\alpha}}_{k+1}^k\\ \mathbf{R}_0^k\left(\mathbf{R}_{k+1}^0\mathbf{v}_{k+1}^{k+1} + \mathbf{g}^0\Delta t\right) - \mathbf{v}_k^k - \hat{\boldsymbol{\beta}}_{k+1}^k\\ 2\left[\hat{\mathbf{q}}_{k+1}^{k^{-1}}\otimes\mathbf{q}_k^{0^{-1}}\otimes\mathbf{q}_{k+1}^0\right]_{xyz}\end{bmatrix} \tag{13}$$

where the gravity vector of the first camera state $\mathbf{g}^0$ is solved by the linear initialization [1], and $(\cdot)_{xyz}$ extracts the vector part of a quaternion.

Using (6) and the fact that $\mathbf{R}_{k+1}^0 = \mathbf{R}_k^0\cdot\mathbf{R}_{k+1}^k$, we can obtain the following by ignoring higher order terms:

$$\delta\boldsymbol{\theta}_{k+1}^0 = \hat{\mathbf{R}}_0^{k+1}\cdot\hat{\mathbf{R}}_k^0\cdot\delta\boldsymbol{\theta}_k^0 + \delta\boldsymbol{\theta}_{k+1}^k,$$

which provides a simple linearized form of the propagation of rotation error terms. As such, the Jacobian of the IMU measurement residual with respect to the error state can be obtained as:

$$\begin{aligned}\mathbf{H}_{k+1}^k &= \begin{bmatrix}\frac{\partial r_{\mathcal{D}}}{\partial\delta\mathbf{x}_k} & \frac{\partial r_{\mathcal{D}}}{\partial\delta\mathbf{x}_{k+1}}\end{bmatrix}\\ \frac{\partial r_{\mathcal{D}}}{\partial\delta\mathbf{x}_k} &= \begin{bmatrix}-\mathbf{R}_0^k & -\Delta t\mathbb{I} & \lfloor\mathbf{R}_0^k\cdot(\mathbf{p}_{k+1}^0 - \mathbf{p}_k^0 + \mathbf{g}^0\frac{\Delta t^2}{2})\times\rfloor\\ \mathbf{0} & -\mathbb{I} & \lfloor\mathbf{R}_0^k\cdot(\mathbf{R}_{k+1}^0\mathbf{v}_{k+1}^{k+1} + \mathbf{g}^0\Delta t)\times\rfloor\\ \mathbf{0} & \mathbf{0} & -\mathbf{R}_0^{k+1}\cdot\mathbf{R}_k^0\end{bmatrix}\\ \frac{\partial r_{\mathcal{D}}}{\partial\delta\mathbf{x}_{k+1}} &= \begin{bmatrix}\mathbf{R}_0^k & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{R}_0^k\cdot\mathbf{R}_{k+1}^0 & -\mathbf{R}_0^k\cdot\mathbf{R}_{k+1}^0\lfloor\mathbf{v}_{k+1}^{k+1}\times\rfloor\\ \mathbf{0} & \mathbf{0} & \mathbb{I}\end{bmatrix}\end{aligned} \tag{14}$$

Equations (10), (12), (13), and (14) define all required quantities to specify the IMU measurement model.

C. Camera Measurement Model

The formulation of the camera measurement model is straightforward. The feature measurement is the observation of the feature in the normalized image plane: $\hat{\mathbf{z}}_l^j = \left[\hat{u}_l^j,\, \hat{v}_l^j\right]^{\mathrm{T}}$. The residual term is the reprojection error, which is defined as:

$$\begin{aligned}r_{\mathcal{C}}(\hat{\mathbf{z}}_l^j, \mathcal{X}) &= \begin{bmatrix}\frac{{}^f x_l^j}{{}^f z_l^j} - \hat{u}_l^j\\ \frac{{}^f y_l^j}{{}^f z_l^j} - \hat{v}_l^j\end{bmatrix}\\ \mathbf{f}_l^j &= \begin{bmatrix}{}^f x_l^j\\ {}^f y_l^j\\ {}^f z_l^j\end{bmatrix} = \mathbf{R}_0^j\left(\mathbf{p}_i^0 - \mathbf{p}_j^0 + \lambda_l\mathbf{R}_i^0\mathbf{u}_l^i\right)\end{aligned} \tag{15}$$

where $\mathbf{u}_l^i = [u_l^i,\, v_l^i,\, 1]^{\mathrm{T}}$ is the first observation of the feature and it is considered as noiseless. The residual covariance is the feature measurement noise matrix $\mathbf{P}_l^j$.

The Jacobian can be obtained by utilizing (6) and applying the chain rule on (15):

$$\begin{aligned}\mathbf{H}_l^j &= \begin{bmatrix}\frac{1}{{}^f z_l^j} & 0 & -\frac{{}^f x_l^j}{{}^f z_l^{j^2}}\\ 0 & \frac{1}{{}^f z_l^j} & -\frac{{}^f y_l^j}{{}^f z_l^{j^2}}\end{bmatrix}\begin{bmatrix}\frac{\partial\mathbf{f}_l^j}{\partial\delta\mathbf{x}_i} & \frac{\partial\mathbf{f}_l^j}{\partial\delta\mathbf{x}_j} & \frac{\partial\mathbf{f}_l^j}{\partial\delta\lambda_l}\end{bmatrix}\\ \frac{\partial\mathbf{f}_l^j}{\partial\delta\mathbf{x}_i} &= \begin{bmatrix}\mathbf{R}_0^j & \mathbf{0} & \mathbf{R}_0^j\cdot\mathbf{R}_i^0\left\lfloor\lambda_l\mathbf{u}_l^i\times\right\rfloor\end{bmatrix}\\ \frac{\partial\mathbf{f}_l^j}{\partial\delta\mathbf{x}_j} &= \begin{bmatrix}-\mathbf{R}_0^j & \mathbf{0} & \left\lfloor\mathbf{R}_0^j\left(\mathbf{p}_i^0 - \mathbf{p}_j^0 + \lambda_l\mathbf{R}_i^0\mathbf{u}_l^i\right)\times\right\rfloor\end{bmatrix}\\ \frac{\partial\mathbf{f}_l^j}{\partial\delta\lambda_l} &= \mathbf{R}_0^j\cdot\mathbf{R}_i^0\mathbf{u}_l^i\end{aligned} \tag{16}$$

Equations (15) and (16) define all required quantities to specify the camera measurement model.

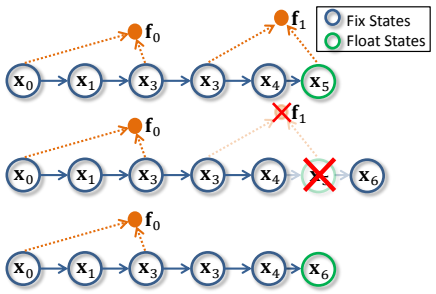
V. Handling Scale Ambiguity via Two-Way Marginalization

For MAV applications, due to limited onboard computational resources and the requirement of real-time processing for feedback control, we have to bound the complexity of the estimator by selectively marginalizing out camera states and features from the sliding window. However, due to the well known acceleration excitation requirement [5, 6, 15] for scale observability for monocular VINS, a naive strategy that always marginalize the oldest state may result in unobservable scale in degenerate motions such as hovering or constant velocity motions.

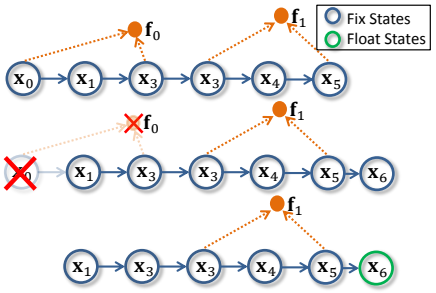
For hovering, as proved in [9], if the vehicle first undergoes generic motions with sufficient acceleration excitation, then enters hovering, the scale observability can be preserved by using a last-in-first-out (LIFO) sliding window scheme. [9] performs state-only measurement update during hovering, and covariance is updated only once as the vehicle exits hovering. However, this approach will lead to pessimistic covariance as during hovering, the covariance can grow arbitrary big, while the actual estimation error is bounded.

For constant velocity motions, the scale is unobservable as old states that correspond to generic motions will eventually be removed from the sliding window due to computation constraints. However, we can still perform scale propagation by marginalizing old states and correct scale drifting when the platform resumes generic motions at a later time.

Based on this discussion, we propose to use a two-way marginalization scheme to bound the computation cost and



(a)



(b)

Fig. 2. Fig 2(a) shows the structure of the full state before, during, and after marginalizing a recent camera state ($\mathbf{x}_5$) after a newer camera state $\mathbf{x}_6$ is added. Similar marginalization process of the oldest camera state is shown in Fig. 2(b).

to handle degenerate motions. This is the adaption of the similar approach proposed in our earlier work [1] into the nonlinear optimization framework. We add a new camera state to the sliding window if the time between two camera states Δt is larger than δ. Note that we do not have a notion of spatial keyframes as in vision-only approaches [13] due to the requirement of bounding the error in the integrated IMU measurements between two camera states. We then select whether to remove the oldest or more recent camera states based on a parallax test. As shown in Fig. 2, a recent state is identified as *fixed* only if it has sufficient parallax to the previous state. Otherwise it will be removed in the next marginalization. We refer the readers to [1] for details of this selection process. We construct a new prior based on all measurements related to the removed states

$$\mathbf{\Lambda}_p = \mathbf{\Lambda}_p + \sum_{k \in \mathcal{D}^-} \mathbf{H}_{k+1}^{k^T} \mathbf{P}_{k+1}^{k^{-1}} \mathbf{H}_{k+1}^{k} + \sum_{(l,j) \in \mathcal{C}^-} \mathbf{H}_{l}^{j^T} \mathbf{P}_{l}^{j^{-1}} \mathbf{H}_{l}^{j} \tag{17}$$

where $\mathcal{D}^-$ and $\mathcal{C}^-$ are sets of removed IMU and camera measurements respectively. The marginalization can the be carried out via Schur Complement [10].

Intuitively, our approach will keep removing the recent camera states if the vehicle has small or no motion. Keeping older camera states in this case will preserve acceleration information that is necessary to recover the scale, while still storing all information provided by the marginalized states as prior. On the other hand, if the vehicle moves at a fast constant speed, older camera states will be removed and converted into priors for scale propagation.

VI. Experimental Results

The configuration of the experimental platform, which is based on the Pelican quadrotor from Ascending Technologies, GmbH, is shown in Fig. 1. It is natively equipped with an AutoPilot board consisting of an IMU and a user-programmable ARM7 microcontroller. The main computation unit onboard is an Intel NUC with a 1.8 GHz Core i3 processor with 8 GB of RAM and a 120 GB SSD. The only additions to onboard sensing are a mvBlueFOX-MLC200w grayscale HDR camera with standard lens that capture 752×480 images at 25 Hz, and a Microstrain 3DM-GX2 IMU. The use of an additional IMU is not for getting better IMU measurements, but for assembling a sensor suite that is rigidly configured, vibration isolated, and with good time synchronization. The total mass of the platform is 1.28kg, which leads to a thrust to weight ratio of approximately two. The entire algorithm is developed in C++ using ROS as the interfacing robotics middleware. Planning, trajectory generation and control methodologies are adapted from our earlier work [17].

A. Real-Time Implementation

Although the onboard camera captures images at 25 Hz, it is both computationally infeasible and unnecessary to perform the optimization at such a high rate. The system starts with one camera state in the sliding window, and a fixed number of corner features detected in that camera image. Features are tracked in the high-rate image sequence until the next camera state is added to the sliding window (Sect. V). The pre-integrated IMU measurement (Sect. IV-B) is also computed as new camera state is added.

We utilize multi-thread implementation to achieve real-time operation. Three threads run concurrently. The first thread is the image processing front end that we just described. The second thread is the main VINS optimizer (Sect. IV) and the marginalization module (Sect. V). Finally, due to computation constraint, our VINS system runs at 10 Hz with approximate processing latency of 35 ms. This is not sufficient for autonomous control of MAVs. We therefore implement a third thread to propagate the latest solution from the optimizer forward using the high-rate IMU measurements. The output of this thread is used directly as the feedback for the trajectory tracking controller [17]. The computation time of each components in our system is shown in Table. I. It suggests that our algorithm is able to run real-time onboard.

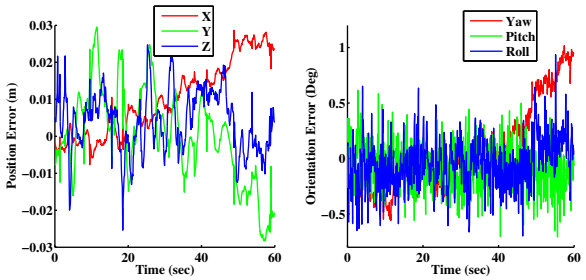
B. Implementation Details and Choice of Parameters

We maintain $N = 30$ camera states and $M = 200$ features in the sliding window. These choices reflects maximum utilization of the available computation power. For each incoming image, we try to detect 400 features with a minimum

Module	Time (ms)	Rate (Hz)	Thread
Feature Tracking	5	25	1
Add New Camera State	9	10	1
Nonlinear Optimization	35	10	2
Marginalization	17	10	2
IMU Forward Propagation	1	100	3

TABLE I

COMPUTING TIME OF DIFFERENT SYSTEM MODULES WITH 30 CAMERA STATES AND 200 FEATURES.



(a) Position Error (b) Orientation Error

Fig. 3. Hover performance of the MAV using onboard state estimates for feedback control comparing with ground truth. Note that there is almost no drift in any direction and the errors are less than 3 cm and 1 degree.

separation of 30 pixels using Shi-Tomasi corners [18] and track them using the KLT tracker [19]. The enforcement of feature separation is to avoid numerical issues due to poorly distributed features. At this point, we apply RANSAC with epipolar constraints for outlier rejection with a threshold of 1 pixel.

We enforce each feature to reach a parallax of at least 30 pixels (distance in the image frame after rotation compensation) before its added into the sliding window for depth estimation. The choice of the minimum parallax is for avoiding numerical issues in triangulation of noisy feature observations with small baseline. During the two-way marginalization (Sect. V), all features that are first observed in the marginalized frame are also marginalized and removed. All features measurements that are made in the removed frame are also marginalized. We marginalize and remove additional feature if the total number of features goes beyond 200.

We determine the addition of frames to the sliding window based on $\delta = 0.1s$, which implies that the sliding window estimator runs at 10 Hz. We choose the camera state to be marginalized based on a parallax threshold of $\varepsilon = 30$ pixels. To deal with motions that is mostly rotation, we additionally force to remove the oldest camera state if the total number of matched features drops below 50.

In the nonlinear optimization, we propagate camera states using pre-integrated IMU measurements and triangulate the depth of each newly added features using current pose estimates. Although we assume all measurements are corrupted by Gaussian noise, we use the Huber kernel [20] for increased robustness outliers. The cutoff threshold for the Huber kernel is set to be 2 pixels. In practice, gross outliers will result in very low weights due to the Huber kernel, which may lead to singularities in dimensions correspond to poorly observed features. In this case, we remove features if their conditional information, which correspond to diagonal terms in the information matrix ($\mathbf{\Lambda}_{\mathcal{C}}$ in (8)), drops below 1. This threshold mainly depends on the stability of the underlying matrix solver, it can be set to some small positive number as far as numerical issues are avoided.

While it is straightforward to include bias parameters in the optimization, we experimentally found that the bias for our IMU is almost always zero. For the sake of saving computation power, we do not include bias terms in the optimization.

C. Autonomous Hovering

One major issue of using a monocular VINS sensor suite for autonomous MAVs is the lack of direct measurement of metric scale. We use this experiment to highlight the accuracy of the overall system, with highlights on our two-way marginalization (Sect. V) scheme. After the on-the-fly initialization is completed, the MAV is commanded to hover using its onboard state estimates for feedback control. As shown in Fig 3, since the two-way marginalization always removes newer camera states while hovering, previous feature observations that have sufficient parallax, as well as IMU measurements that have sufficient accelerations are kept. As such, almost-drift-free estimates in full 6 degree-of-freedom are obtained. Very minor drift can be found in the error plot. This is due to small drifts in the KLT tracker, which may be avoided with descriptor-based feature matching. However, we defer this implementation as future work. During hovering, the onboard position estimate compares well with the ground truth with standard deviation of $\{0.0099, 0.0124, 0.0081\}$ meters. The precision of hovering using such onboard estimates for feedback control has the standard deviation of $\{0.0282, 0.0307, 0.0161\}$ meters.

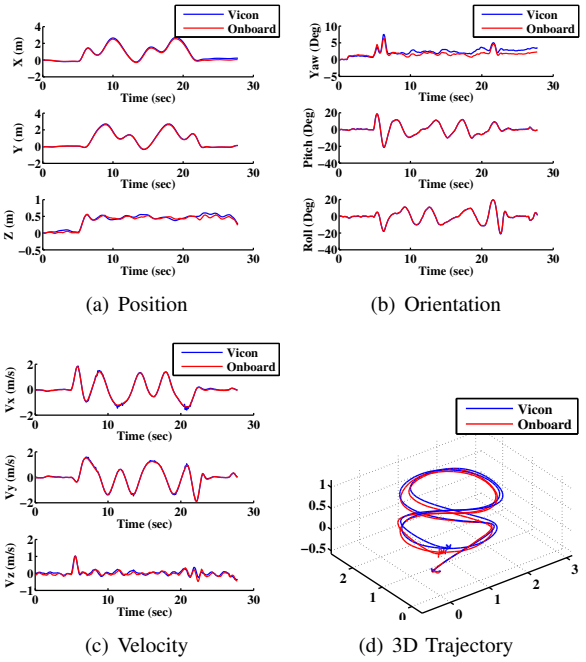
D. Autonomous Trajectory Tracking

In this experiment, we test the performance of autonomous trajectory tracking while using the onboard state estimate for feedback control. The MAV is commanded to track a figure eight pattern with each circle being 0.9 meters in radius, as shown in Fig. 4(d). We conduct multiple trials of the same desired path but with different velocities. The desired velocity is given as a parameter for the minimum jerk trajectory generator [17], although the actual velocity problem varies during the flight. Important performance metrics are shown in Table. II. With different desired velocity, the maximum roll and pitch angles increases due to higher linear acceleration. The time required to complete the path is shortened accordingly. Meanwhile, we observe almost no difference in position and yaw drifting, which suggests that our approach is able to handle fast motions without

Trial	1	2	3	4	5	6	7
Duration (sec)	35.88	27.20	25.50	25.19	20.77	18.14	17.84
Max Velocity (m/s)	1.035	1.271	1.391	1.397	1.694	1.953	1.955
Max Roll/Pitch (deg)	6.047	9.089	11.51	13.34	20.71	26.19	28.04
Position Drift (m)	0.073	0.141	0.039	0.091	0.127	0.078	0.162
Yaw Drift (deg)	0.140	1.227	0.622	0.825	1.756	0.901	1.174
Velocity Standard Deviation (m/s)	0.055	0.053	0.062	0.066	0.068	0.067	0.063
Roll/Pitch Standard Deviation (deg)	0.160	0.190	0.251	0.204	0.311	0.332	0.262

TABLE II

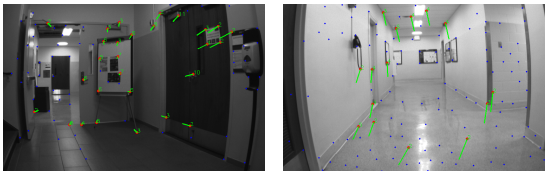
PERFORMANCE OF AUTONOMOUS TRAJECTORY TRACKING EXPERIMENTS WITH VARYING SPEEDS. THE VELOCITY AND ROLL/PITCH STANDARD DEVIATIONS ARE SHOWN AS MAXIMUM VALUES BETWEEN ALL DIMENSIONS.



(a) Position (b) Orientation

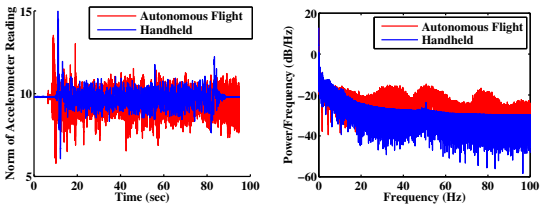
(c) Velocity (d) 3D Trajectory

Fig. 4. Performance of autonomous trajectory tracking using onboard monocular VINS estimates for feedback control. The maximum speed is 2 m/s, while the maximum roll/pitch angle is 28 degrees. The onboard velocity and attitude estimates track the ground truth accurately. Small drifts position and yaw, which are known to be unobservable, are observed.



(a) (b)

Fig. 5. Images from the onboard camera during autonomous flight in indoor environments. Red dots represent features that have valid depth from sliding window optimization, green lines shows the tracking of such features. Blue dots are features that are being tracked but have not been added into the optimization. The criteria for choosing features is discussed in Sect. VI-B.



(a) (b)

Fig. 6. Comparison of IMU noise characteristics in both time and frequency domains for in flight and handheld cases. Plots shows the variation of the magnitude of the IMU measurement across all axes. Note how the mechanical vibrations worsen IMU performance while in flight.

degeneration of performance. We do observe a slight increase in the standard deviations in velocity and roll and pitch, however, we do not observe any downgrade in terms of flight performance. In Fig. 4, we highlight the fastest trial with a maximum velocity of 2 m/s and maximum roll/pitch angles close to 30 degrees. We note that due to numerical differentiation, the Vicon ground truth is actually noisier than the onboard state estimate as shown at 12 sec and 21 sec in Fig. 4(c). As such, the velocity error is expected to be smaller than the values reported on the plot.

E. Autonomous Flight in Indoor Environments

In this experiment, we show autonomous flight in the hallway of a typical office building using the proposed monocular VINS state estimates for feedback control. Fig. 5 shows images from the onboard camera during the experiment. Since no ground truth is available, we run multiple trials to verify the accuracy, consistency, and repeatability of the proposed approach. As shown in Fig. 7, we conduct two autonomous flight and one handheld experiments. Statistics of each trial is shown in Table. III.

During autonomous flight experiments, the MAV is stabilized using onboard state estimates. The operator have the freedom of sending waypoints to the MAV, from which a minimum jerk trajectory will be generated and tracked by the vehicle. The operator may also control the velocity of the MAV directly via the kinematic controller [17]. In both autonomous flight experiments, the MAV was able to fly stably and return to its starting position with only a small drift in estimated poses.

We do note that the estimation accuracy in autonomous flight is slightly worse than the handheld experiment. This is due to much noisier IMU measurements caused by mechan-

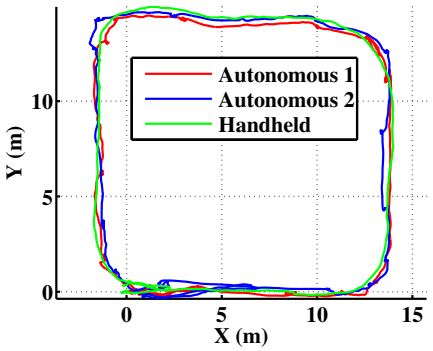


Fig. 7. Trajectories of two autonomous flight and one handheld experiments. The onboard estimator is able to deliver repeatably results in the sense that the drift in all trials are similar, and the scale estimate is correct. Keep in mind that the visual scale is not directly observable in the monocular VINS setup, a consistent scale implies correct scale estimation. Videos of experiments can be found at `http://www.ee.ust.hk/~eeshaojie/ICRA2015.mp4`.

Trial	Autonomous 1	Autonomous 2	Handheld
Duration (sec)	281.4	303.6	78.72
Distance (m)	81.39	87.52	63.38
Position Drift (m)	0.634	0.944	0.364
Position Drift %	0.78%	1.08%	0.58%
Yaw Drift (deg)	1.83	2.703	1.79
Yaw Drift %	0.51%	0.75%	0.50%

TABLE III

STATISTICS OF AUTONOMOUS FLIGHT AND HANDHELD EXPERIMENTS.

ical vibrations from the motors while flying. A comparison in both time and frequency domains of IMU measurements during both in flight and handheld cases is shown in Fig. 6. In all trials, the onboard state estimator behaves consistently in the sense that the overall trajectory is well aligned. Note that since the visual scale is not directly observable in the monocular VINS setting, consistent scale in all trials indicates that the scale is correctly estimated.

VII. CONCLUSION

In this work, we propose a tightly-coupled monocular VINS estimator that enables autonomous flight of a rotorcraft MAV in unknown and unstructured environments. Our approach optimizes a fixed history of vehicle states as well as environment features using nonlinear optimization. We also pay special attentions to the initialization of monocular VINS and the handling of degenerate motions. We present online experimental results in different environments with a maximum flight velocity of 2 m/s and maximum roll/pitch angles close to 30 degrees. Statistical analysis is presented to verify the performance.

It is well known that for monocular VINS, motions that are perpendicular to the camera optical axis are required for feature triangulation. We are therefore interested in studying the relation between motion and feature triangulation and investigate into active control strategies for guaranteed obstacle detection and avoidance with monocular VINS.

REFERENCES

[1] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Initialization-free monocular visual-inertial estimation with application to autonomous MAVs," in *Proc. of the Intl. Sym. on Exp. Robot.*, Marrakech, Morocco, 2014.

[2] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Saint Paul, MN, May 2012, pp. 957–964.

[3] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, "Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robot. Autom. Mag.*, vol. 21, no. 3, 2014.

[4] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Roma, Italy, Apr. 2007, pp. 3565–3572.

[5] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *Intl. J. Robot. Research*, vol. 30, no. 1, pp. 56–79, Jan. 2011.

[6] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *Intl. J. Robot. Research*, vol. 30, no. 4, pp. 407–430, Apr. 2011.

[7] D. G. Kottas, J. A. Hesch, S. L. Bowman, and S. I. Roumeliotis, "On the consistency of vision-aided inertial navigation," in *Proc. of the Intl. Sym. on Exp. Robot.*, Quebec, Canada, June 2012.

[8] M. Li and A. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *Intl. J. Robot. Research*, vol. 32, no. 6, pp. 690–711, May 2013.

[9] D. Kottas, K. Wu, and S. Roumeliotis, "Detecting and dealing with hovering maneuvers in vision-aided inertial navigation systems," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Tokyo, Japan, Nov. 2013, pp. 3172–3179.

[10] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *J. Field Robot.*, vol. 27, no. 5, pp. 587–608, Sept. 2010.

[11] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial SLAM using nonlinear optimization," in *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, June 2013.

[12] V. Indelman, A. Melim, and F. Dellaert, "Incremental light bundle adjustment for robotics navigation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Tokyo Japan, Nov. 2013, pp. 1952–1959.

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

[14] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. of the IEEE Intl. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 2320–2327.

[15] A. Martinelli, "Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," vol. 28, no. 1, pp. 44–60, Feb. 2012.

[16] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 61–76, Feb. 2012.

[17] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor," in *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, 2013.

[18] J. Shi and C. Tomasi, "Good features to track," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Seattle, WA, June 1994, pp. 593–600.

[19] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, Vancouver, Canada, Aug. 1981, pp. 24–28.

[20] P. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 2, pp. 73–101, 1964.