Check for updates

Received: 18 March 2019 | Revised: 15 January 2020 | Accepted: 23 January 2020

DOI: 10.1002/navi.364

ORIGINAL ARTICLE

WILEY ION

Relative navigation of fixed-wing aircraft in GPS-denied environments

Gary Ellingson[1] | **Kevin Brink**[2] | **Tim McLain**[1]

[1]Mechanical Engineering, Brigham Young University, Provo, Utah, USA

[2]Munitions Directorate, Air Force Research Laboratory, Valparaiso, Florida, USA

Correspondence
Gary Ellingson, Mechanical Engineering, Brigham Young University, Provo, UT, USA.
Email: gary.ellingson@byu.edu

Funding information
Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry University Cooperative Research Center (IUCRC), Grant/Award Number: IIP-1650547; Utah Space Grant Consortium

Abstract

This work enables GPS-denied flight on fixed-wing UAS by accounting for fixed-wing-specific sensing requirements and using a methodology called relative navigation as an overarching framework. The development of an odometry-like, front-end, EKF-based estimator that utilizes only a monocular camera and an inertial measurement unit (IMU) is presented. The filter uses the measurement model of the multi-state-constraint Kalman filter. The filter also regularly resets its origin in coordination with the declaration of keyframe images. The keyframe-to-keyframe odometry estimates and their covariances are sent to a global back end that represents the global state as a pose graph. The back end is better suited to represent nonlinear uncertainties and incorporate opportunistic global constraints. We also introduce a method to account for front-end velocity bias in the back-end optimization. The paper provides simulation and hardware flight-test results of the front-end estimator and performs several back-end optimizations on the front-end data.

1 | INTRODUCTION

The capabilities of unmanned aircraft systems (UAS) have dramatically increased over the past decade. This expansion of capabilities has largely been enabled by the development, optimization, and miniaturization of traditional navigation methods, where Global Positioning System (GPS) measurements are fused with inertial measurements (GPS-INS). Figure 1 shows an example of a small, fixed-wing UAS. As UAS continue to get smaller and more advanced, they will be able to operate in confined spaces, in urban environments, and inside buildings. For UAS to continue to expand to more applications, they will require the ability to navigate when GPS is unavailable or unreliable. For example, many civil applications, including delivery and inspection services, require that UAS fly in close proximity to structures. Structures can reduce the accuracy and reliability of GPS signals. UAS can also be used in military applications for observing and prosecuting the enemy, but the threat of spoofing and jamming of GPS signals provides motivation for navigating without relying on GPS measurements.

Inertial measurements, by themselves, can be used to estimate the motion of a UAS, but sensor errors will accumulate and cause the estimates to drift. A UAS can be augmented with exteroceptive sensors, such as cameras or laser scanners, to measure the motion of the vehicle with respect to the surroundings. By fusing the inertial and exteroceptive measurements, the motion estimate will improve. In the absence of GPS or other global measurements, the global position and yaw angle are unobservable,[1-3] and the estimates will eventually diverge.

Sensor noise filtering and measurement fusion can take place in an extended Kalman filter (EKF). EKFs, used extensively on robots[4] and UAS[5,6] alike, account for both sensor errors and process uncertainty. They utilize a linear Gaussian representation of the state belief to take advantage of the computational convenience of a Kalman update but maintain the nonlinearity of the process propagation. This combination of properties performs well when

 © 2020 Institute of Navigation

FIGURE 1 This work enables GPS-denied flight of fixed-wing UAS. The method was tested on a modified STRIX StratoSurfer, a 1.5 m wingspan aircraft [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

errors remain small, such as when the availability of GPS measurements is used to regularly remove drift errors. The nonlinear nature of the process, however, causes the Gaussian representation of the belief to become inconsistent when errors are large due to the global states being unobservable and their estimates drifting from the true value. If a global measurement is received by an EKF after significant drift errors have accumulated, nonlinearities can make utilizing the measurement problematic. This causes over confidence, especially in states such as velocities and inertial measurement unit (IMU) biases.[7,8] This may result in large jumps in the estimate and, in severe cases, can even cause filter divergence. Methods to allow EKFs to handle sparse opportunistic global measurements are often ad hoc or cumbersome, including reinitializing the filter by shifting its origin, treating GPS as a relative sensor by transforming the measurement into a temporary coordinate frame,[8] or gating (and thus ignoring) the measurement.[9] Some methods simply avoid using an EKF when GPS measurements are intermittent.[10]

These observability and consistency problems have been addressed in recent years by the proposal of a new approach called relative navigation.[11,12] Relative navigation has been introduced as a solution for operating UAS when GPS is either unavailable or intermittent at best. It utilizes an EKF for front-end estimation relative to the local environment and a back-end optimization that combines the relative information to produce the global estimates. The complete architecture is shown in Figure 2. Dividing the architecture into a relative front end and a global back end provides important observability and computational advantages. The front end navigates with respect to a local frame where the states can remain observable and the Gaussian distribution can accurately represent uncertainty, thus enabling the computational advantage of an EKF to be utilized. The back end uses a pose graph that can accurately represent nonlinearities in heading and be robustly optimized when given additional constraints. These constraints arise from measurements that constrain the pose either to a global location, such as an opportunistic GPS measurement, or, as introduced in this paper, relative to non-stationary objects, such as other aircraft.

The majority of GPS-denied navigation research has been performed with multirotor UAS as the experimental platform. To effectively navigate fixed-wing UAS without GPS, additional considerations including aircraft dynamics, mission profiles, and sensing requirements must be taken into account. Prior work that has focused on fixed-wing UAS has often either made significant simplifying assumptions, including flat-Earth or Manhattan world environments, or imposed strict sensing requirements, such as a downward facing camera or distance measurements.[14,15] The ability of multirotor UAS to hover in place enables them to more easily fly in and around buildings and structure. This allows laser scans and other distance measurements to effectively measure the aircraft motion. For fixed-wing UAS that typically fly at high speeds and high above the ground, the vehicles are often at altitudes that exceed or approach the limit of the measurement range of depth sensors making them less effective for sensing aircraft motion.

Using the relative navigation framework as a guide, this work enables GPS-denied navigation of fixed-wing UAS by developing a tightly coupled, EKF-based, visual-inertial odometry (VIO). With the fixed-wing requirements in mind, we avoid the use of depth sensors, such as laser scanners and RGB-D cameras, and utilize only a monocular camera with no assumptions about the distance to observed features. By producing keyframe-based estimates of the change in pose, the front-end estimator enables the fixed-wing aircraft to utilize all the advantages of the global back end within the relative-navigation framework for GPS-denied navigation. This paper extends our previous efforts[16] where the concepts for the VIO filter and limited simulation results were initially presented. This paper provides a complete filter development and improved simulation results, as well as hardware, flight-demonstration results. Along with the flight-demonstration results, our efforts to mitigate calibration, timing, and initialization errors are discussed. Another contribution of this paper is in the back-end pose graph optimization. We provide a model of a slowly varying scale bias to account for both scale errors that arise from potentially unobservable velocity associated with straight-and-level flight and the correlation from one graph edge to the next. The full system localization is demonstrated by utilizing the front-end odometry together with various other opportunistic measurements that provide loop-closure-like constraints. Finally, to motivate future work, results are presented using measurements and constraints between cooperative aircraft

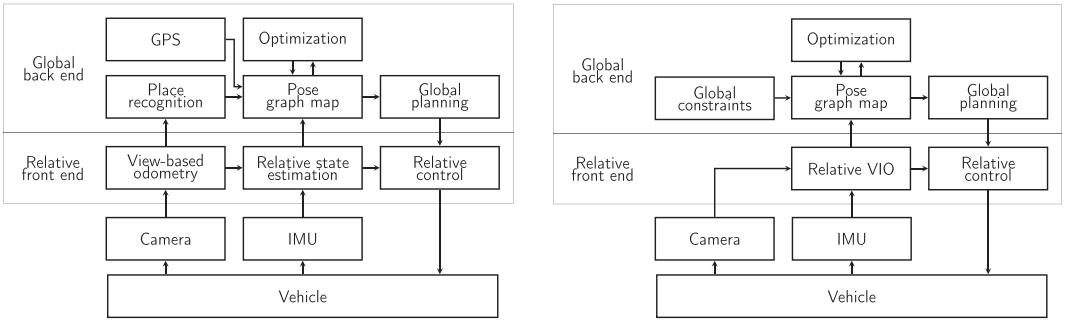

FIGURE 2 Relative navigation architecture for GPS-denied navigation. State estimation and control is performed relative to a locally declared coordinate frame. Global mission planning and localization are performed in the back end. Left: In previous work, a view-based odometry has been used to produce an odometry solution for use as a measurement in the filter.[12,13] Right: This work develops a fixed-wing, front-end estimator by removing the view-based odometry and making the estimator a tightly coupled, visual-inertial odometry. This work also provides a significant hardware flight-test experiment of the front-end estimator and provides a new method for improving the back-end optimization by modeling time-correlated scale bias caused by small velocity errors in front-end estimates

that demonstrate the potential of the proposed method for low-bandwidth, multi-vehicle cooperative localization.

2 | RELATED WORKS

This paper builds upon previous research in two main areas: The overarching framework draws from the relative navigation body of research, and the method for constructing the visual-inertial odometry uses the principles from the multi-state-constraint Kalman filter (MSCKF). Relevant research contributions in these areas are summarized in the subsequent sections.

2.1 | Relative navigation

Relative navigation is built on an elegant concept: At any point in time, an agent can have complete confidence in its position if, at that instant, it places its reference-frame origin at the vehicle center. An agent can further maintain good confidence of its local motion by observing the apparent motion of the local surroundings, even if the global position is unknown or is unobservable over large scales. As an example, a robot agent can set its initial position to zero and then localize around this initial origin, even though the origin's global position is arbitrary.

Relative navigation uses this concept in the front-end filter in a process called the relative-reset step. The reset step is closely related to the keyframe update of keyframe-based odometry methods. As the vehicle travels from the current origin, the front-end filter is able to reset the origin to the current location of the vehicle (the new coordinate frame being aligned with the heading of the vehicle but level with the local-level frame), where the reset coincides with the declaration of a keyframe image. Within the EKF, the covariance associated with the position and heading states can then be zeroed, and the states continue to evolve with respect to the newly declared reference frame. The state from just prior to the reset then forms a transformation from one reset to the next and, together with the associated covariance, is provided to the back end. The transformations form a directed pose graph, where each origin is a node (or node frame) and each transformation is an edge. Because the EKF operates only with respect to a local origin, it is observable, as well as consistent, by construction.[7] The uncertainty is regularly removed from the filter while a Gaussian is still able to accurately represent it, and nonlinearities are handled appropriately in the back-end graph.

The global position and heading are accounted for in the back end because it contains the keyframe-to-keyframe transformations as edges in a pose graph. The global pose, which is necessary for accomplishing a mission with a global goal, can be produced by combining, or composing, the transforms. Figure 3 demonstrates how the graph edges are able to represent the nonlinear coupling in $SE(3)$ better than a single pseudo-global state with a Gaussian uncertainty, especially when heading uncertainty is large.[7] The global localization is also improved when the back end is able to optimize the pose graph when it receives other constraints, such as opportunistic GPS measurements and place-recognition loop closures.[12,13] Graph optimization has been studied extensively, and computationally efficient methods are available[17,18] for performing these optimizations. Using these techniques, relative navigation deliberately avoids global updates to the front-end filter and thereby increases filter robustness.

The division of the front end and back end also provides additional benefits for scalable UAS operations. First, because the EKF of the front end implicitly draws on the Markov assumption (i.e., the current state and covariance completely represent the previous sequence of events and measurements), it essentially compresses

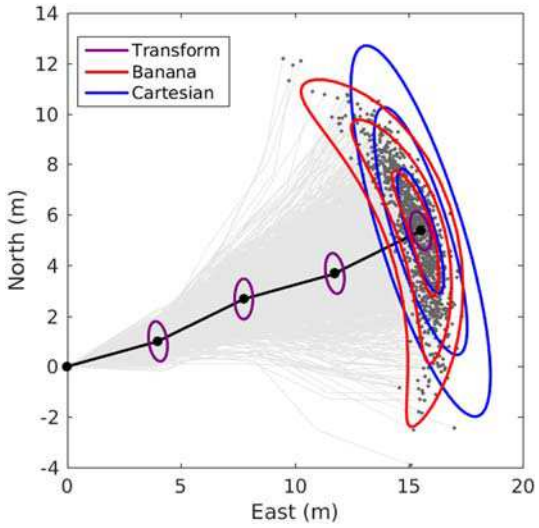

FIGURE 3 An example of a relative-navigation graph. Each edge provides a transformation from one keyframe (or node) to the next. Although the transformations and associated covariances (small ellipses) are linear, the graph is able to represent more complex, nonlinear uncertainties (such as the banana distribution of the Monte-Carlo samples) better than a single Gaussian (large ellipses).[3,7] This figure from Wheeler et al.[7] is reproduced with permission [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

the high-rate sensor information into edges that are published at a low frequency. This compression, effectively pre-marginalization of the graph factors, helps to make the back-end scale for long-duration flights. Also, as the back-end graph grows and the computation of optimization increases, the decoupling of the front end allows the graph optimization to be completed slower than real time if needed, while the front end is still providing full-rate state estimates necessary for vehicle control. Without providing empirical results, Brink[19] hypothesizes that these scalablility properties could be beneficial for multi-vehicle cooperative localization.

Prior to this work, the relative-navigation front end has relied on a loosely coupled VIO where the filter depends on a separate visual odometry algorithm, such as Zhang et al.,[20] and uses a complete odometry solution as a measurement input. This is depicted in Figure 2 with separate boxes for view-based odometry and relative-state estimation. The primary functions of the filter have been to perform the relative-reset step and fuse the odometry with inertial measurements.[21] The keyframe-based visual odometries have been responsible for maintaining visual overlap between the keyframe and the current image by declaring new keyframes regularly and thus have controlled when nodes are declared. They have, so far, resolved scale ambiguity by depending on sensors that measure distance, including laser scanners and RGB-D cameras.[12] Since these sensors are impractical for small, low-cost, fixed-wing UAS, a method that is capable of observing scale without them, such as a visual-inertial odometry, is ideal for this work.

2.2 | MSCKF

The MSCKF has had a significant impact on the VIO research field. Results have shown that it is capable of maintaining an accumulated error of less than 1% of the total distance traveled. It has also been proposed for a variety of applications including smart phones,[22] ground vehicles,[23] and spacecraft.[24] A recent comparison of the MSCKF to other VIO methods[25] shows that its accuracy and consistency performance remains comparable to the state-of-the-art while it is often computationally less expensive.

The work in Li and Mourikis[26] presented the MSCKF as a dual to EKF SLAM. When EKF SLAM is used as a VIO, the state vector includes states that evolve with the vehicle motion ($\mathrm{x}_{\mathrm{imu}}$) and is augmented with the image feature locations. The state vector x has the form:

$$\mathrm{x} = [\mathrm{x}_{\mathrm{imu}}^{\mathrm{T}} \quad f_0 \quad f_1 \quad \dots \quad f_k]^{\mathrm{T}},$$

where f_k is the location of feature k. Although EKF SLAM is relatively intuitive, several issues arise from the fact that the location of the feature is initially unknown by a scale factor and error is introduced when the state vector is augmented. Various modifications have been proposed, including delayed feature initialization[27] and inverse depth parameterization,[28] but the addition of initialization error to the state vector with every feature remains an issue with EKF-SLAM-based VIO.

The MSCKF avoids these issues by instead augmenting the state vector with the transformation to the camera at the instant each image is captured. The state vector is therefore defined as

$$\mathrm{x} = [\mathrm{x}_{\mathrm{imu}}^{\mathrm{T}} \quad \pi_0 \quad \pi_1 \quad \dots \quad \pi_k]^{\mathrm{T}}, \tag{1}$$

where π_k is the pose of image k. In this formulation, little additional error is added to the state vector during augmentation because the location of the image is well known and its error is correlated with error in $\mathrm{x}_{\mathrm{imu}}$. The state vector contains a time history of image poses that enables feature tracks to be used as measurements given a measurement model.[29] A given feature is tracked across a sequence of images, and once it leaves the camera field of view, the feature track is residualized as a single measurement-update step.

The residual is created by first performing a least-squares optimization to produce the three-dimensional location of the feature given the image poses. The optimized location of the feature is used to

produce the predicted pixel coordinates $\hat{z}$ that are subtracted from the measured pixel coordinates z to produce the residual r as

$$r \triangleq z - \hat{z} .$$

Because the feature location was optimized given both the feature pixel coordinates and the image poses in the state vector, the errors in the feature location are correlated with errors in the state vector. This correlation is removed by performing a projection of the residual onto the null space of the feature position. A linear approximation of the residual is produced by two Jacobians: $\boldsymbol{H}_x$, which accounts for the residual with respect to perturbations in the state vector, and $\boldsymbol{H}_f$, which approximates the residual with respect to perturbations in the feature location. The residual and Jacobians are fully defined in Appendix B1. These, with a noise term $\boldsymbol{\eta}$, can be written as

$$r \simeq H_x \tilde{x} + H_f \tilde{p}_n^f + \boldsymbol{\eta},$$

where $\tilde{x}$ and $\tilde{p}_n^f$ are the error in the state vector and position of the feature, respectively. The update is then performed by first projecting the residual, noise, and Jacobian H_x onto the null space of H_f, or

$$r_0 \triangleq A^T(z - \hat{z}) \simeq A^T H_x \tilde{x} + A^T \boldsymbol{\eta},$$

where A denotes the unitary matrix whose columns form the basis of the left null space of H_f. Finally, the projected residual r_0 and Jacobian H_0 are in an appropriate form

$$r_0 \simeq H_0 \tilde{x} + \boldsymbol{\eta}_0 ,$$

for use in the Kalman update, given that $H_0 = A^T H_x$ and $\boldsymbol{\eta}_0 = A^T \boldsymbol{\eta}$.

The MSCKF has also had several extensions and variations. The original work was extended in the publication of the MSCKF 2.0, which introduces a method for ensuring the state-transition matrix has accurate observability properties.[26] On-line camera calibration, including accounting for rolling-shutter, was introduced in Shelley[22] to improve accuracy. Several slightly different formulations of the state vector have been proposed. The work in Clement et al.,[23] for example, propagates the estimates using velocity commands and therefore avoids the need for acceleration bias terms. Formulations have used both continuous and fully discrete propagation steps with discrete measurement updates. Finally, to ensure computation remains tractable, several strategies have been proposed for regularly pruning camera poses from the state vector.[22,29]

3 | DEVELOPMENT

The MSCKF measurement model provides a method for constructing a VIO for a fixed-wing UAS because it does not make assumptions about the distance to image features and is both accurate and consistent, at least while nonlinearities due to heading uncertainty remain small. For it to function as a relative-navigation, front-end estimator, the original MSCKF, must be modified to include a reset step. There is some added complexity and some slight degradation in the filter's accuracy, compared to the original MSCKF inherent in this approach. The degradation is due to a small amount of information being lost every time a new node frame is declared. We argue that these changes and their benefits, specifically the improved robustness as well as the potential for a light-weight multi-agent back end, outweigh the disadvantages for many applications.

The development of the filter begins by completely defining the state vector in Equation (1). The pose of the vehicle body (b) consists of a quaternion q_n^b and a north-east-down position p_n^b with respect to the most recent node frame (n). The body of the aircraft is assumed to be centered at and axis aligned with the IMU. In contrast to other MSCKF implementations, the velocity is body-fixed v_b, meaning expressed in the body frame. The complete IMU state is

$$x_{imu} \triangleq \left[p_n^b \;\; q_n^b \;\; v_b \;\; \beta_w \;\; \beta_a \right]^T,$$

where the IMU acceleration and angular rate estimated bias are β_a and β_w, respectively. The transformation to the kth camera image i_k is its position and orientation in the node frame,

$$\pi_k \triangleq \left[p_n^{i_k} \;\; q_n^{i_k} \right]^T . \tag{2}$$

When an image is taken, these states are calculated from the current IMU state using

$$p_n^{i_k} = p_n^b + R^T(q_n^b) p_b^c$$
$$q_n^{i_k} = q_n^b \otimes q_b^c$$

where $\otimes$ is Hamiltonian-quaternion multiplication, (p_b^c, q_b^c) is the calibrated pose of the camera in the body frame, and $R(q_n^b)$ denotes the rotation matrix associated with q_n^b. It is important to note that the use of the quaternion for rotation requires the filter to use the error-state formulation and be multiplicative. In practice, this means that while the quaternion has four values, to be a minimal representation, the covariance for the same rotation is a three-by-three matrix of the rotational error uncertainty.[30,31]

The covariance $\boldsymbol{P}$ of the state vector consists of an upper left block that corresponds to the IMU state x_{imu}. With every image transformation that is added to the state vector, the covariance matrix is augmented as

$$\boldsymbol{P} \leftarrow \begin{bmatrix} \boldsymbol{P} & \boldsymbol{P}\boldsymbol{J}_\pi^T \\ \boldsymbol{J}_\pi \boldsymbol{P} & \boldsymbol{J}_\pi \boldsymbol{P} \boldsymbol{J}_\pi^T \end{bmatrix},$$

where the Jacobian $\boldsymbol{J}_\pi$ relates the current camera location to $\boldsymbol{x}_{imu}$ by accounting for the IMU-to-camera extrinsic

parameters. If $\lfloor \cdot \rfloor$ is the skew-symmetric matrix (defined in Appendix B1), $\boldsymbol{J}_\pi$ is defined as

$$\boldsymbol{J}_\pi \triangleq \begin{bmatrix} \mathrm{I}_{3\times3} & 0_{3\times3} & -\mathrm{R}^{\mathrm{T}}(\mathrm{q}_n^b)\lfloor \mathrm{p}_b^c \rfloor & 0_{3\times3} & 0_{3\times3} & 0_{3\times6k} \\ 0_{3\times3} & 0_{3\times3} & \mathrm{R}(\mathrm{q}_b^c) & 0_{3\times3} & 0_{3\times3} & 0_{3\times6k} \end{bmatrix},$$

where $0_{3\times3}$ is a 3 by 3 matrix of zeros and $\mathrm{I}_{3\times3}$ is the identity. These terms are important because the correlations from the images to the IMU states make the feature track measurements useful in removing error from $\mathrm{x}_{\mathrm{imu}}$.

The IMU states are propagated with every IMU measurement. The orientation and velocity are mechanized by integrating angular rate and acceleration measurements on the manifold. At each time step, a small amount of process noise is added to the covariance of the bias states to model a slow random walk and to the covariance of the integrated states to model sensor noise. The dynamics are modeled as

$$\begin{aligned} \dot{\boldsymbol{p}}_n^b &= \mathrm{R}^{\mathrm{T}}(\boldsymbol{q}_n^b)\mathrm{v}_b \\ \dot{\boldsymbol{q}}_n^b &= \frac{1}{2}\boldsymbol{q}_n^b \otimes \begin{bmatrix} w \\ 0 \end{bmatrix} \\ \dot{\boldsymbol{v}}_b &= \lfloor \mathrm{v}_b \rfloor w + \boldsymbol{R}(\boldsymbol{q}_n^b)\mathrm{g} + \mathrm{a} \\ \dot{\boldsymbol{\beta}} w &= \boldsymbol{\eta}_{\boldsymbol{\beta}_w} \\ \dot{\boldsymbol{\beta}} a &= \boldsymbol{\eta}_{\boldsymbol{\beta}_a} \\ \dot{\pi}_k &= 0_{7\times1}, \end{aligned}$$

where $\boldsymbol{\eta}_{\boldsymbol{\beta}_w}$ and $\boldsymbol{\eta}_{\boldsymbol{\beta}_a}$ are Gaussian noise processes for their respective states, $\boldsymbol{g}$ is the gravity vector, $\boldsymbol{w}$ is the angular velocity vector, and $\boldsymbol{a}$ is the acceleration vector. In practice, w and $\boldsymbol{a}$ are obtained by removing their respective bias estimates from the IMU measurements.

The measurement update, including the measurement Jacobians, is formulated to depend on the optimization producing the feature location in the node frame p_n^f. This is in contrast to prior work that has defined the optimizations in the global frame[22] or in the image frame where the feature was first observed.[29] The node frame was used because the filter state is relative to the most recent node and the majority of feature-track measurements are initialized on the keyframe image. An inverse depth parameterization of the feature location is used to perform Levenberg-Marquardt least squares and is defined in Appendix C1. The coordinate-frame transformations necessary for the optimization are depicted in Figure 4.

The relative-reset step consists of removing the heading portion of q_n^b as well as zeroing the position p_n^b. The uncertainty of the states is also removed by applying a projection to the covariance matrix.[21] The reset step is fully defined in Appendix D1. In prior relative navigation implementations,[12,13] the reset step was performed after the vehicle had traveled more than a specified distance or yawed more than a specified angle. Since these criteria are insufficient to ensure image overlap, this work makes the criteria for reset depend on the number of feature tracks that are maintained with the most recent keyframe. Once the feature tracker can no longer maintain nine common feature tracks, the reset is performed. This criteria is used to ensure that there is sufficient overlap between images and also that the number of feature correspondences is adequate to construct a complete transformation between the keyframe and the current image.[32] It has the added benefit of ensuring the reset does not happen sooner than necessary. The state vector is then purged of all image transformations π_k, and the current image becomes the next keyframe. The state vector is augmented with the keyframe image transformation π_0, and the keyframe is the first image i_0.

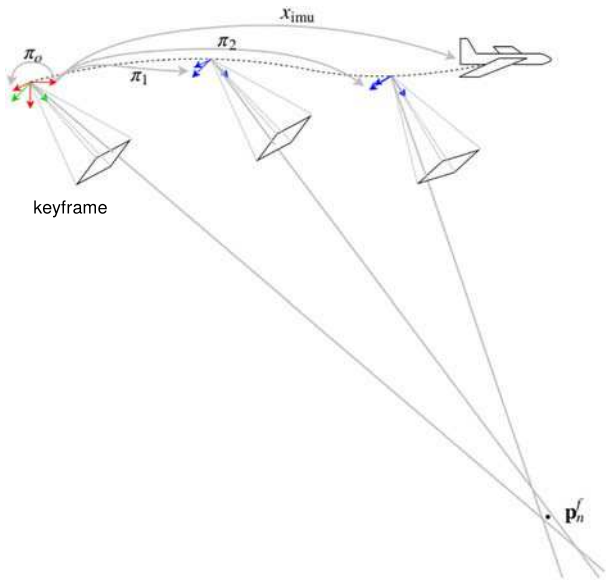

FIGURE 4 The location of a feature in the node frame p_n^f is obtained through a least-squares optimization. The flight path begins at the declaration of a new node frame. Also shown are the transformations from the node frame to the keyframe, to all other image frames, and to the current aircraft pose [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

4 | FRONT-END IMPLEMENTATION

The mathematical development of the filter, while essential, is insufficient without the myriad of implementation details necessary to run and test it. The following sections describe, in part, the simulation implementation details and our efforts to minimize and appropriately account for relevant sources of error that accompany running the filter on hardware.

4.1 | Filter

The feature tracker implemented a pyramidal KLT tracker[33-35] using C++ OpenCV libraries. The feature

tracker was responsible for informing the filter precisely when to augment the filter state as well as when to perform a reset step. When the feature tracker can no longer track a given feature, for example, if the feature goes out of view, the tracker provides to the filter the complete track as a measurement, consisting of the history of pixel coordinates for every image where it was observed.

Although it was initially developed in Python,[16] the filter was implemented in C++ and uses the Robot Operating System (ROS) for communication with sensors. The C++ implementation allowed the filter to run in real time and at full sensor rate, even on an embedded ARM processor.

4.2 | Simulation

The filter was first tested in a ROS/Gazebo simulation using the tools that are distributed with ROSplane.[36,37] The fidelity of the simulation was enhanced by simulating a small fixed-wing aircraft, including aircraft aerodynamics, flight characteristics, and sensors. The aircraft was flown in a realistic flight over a cityscape image appropriate for obtaining image features and testing a VIO algorithm.

In the simulation, sensor plug-ins were used to supply the filter with simulated camera images and IMU measurements from the aircraft. The IMU was oriented to be axis-aligned with the body of the aircraft, and noise and bias walk parameters were representative of an MPU-6050 IMU, based on models presented in Furrer et al. and Maybeck[38,39]. Feature tracks were obtained from the simulated camera image using the tracker described previously. An example of the simulated image and image feature tracks is shown in Figure 5. The camera was oriented facing forward and tilted 45° down from the longitudinal axis of the aircraft. The images were 640 by 480 pixels, and the camera had a 115° field of view.

4.3 | Hardware

The front-end filter was implemented on a small remotely piloted hobby-grade aircraft (Figure 1). The aircraft carried an Nvidia Jetson TX2 embedded computer. The use of the OpenCV CUDA functionality was utilized to perform image processing on the GPU. The use of the GPU freed the CPU to perform other tasks and reduced the tracker CPU load from 130% to 30% of a single processor core. The TX2 received images from a Point Grey Chameleon 3 USB camera and the acceleration and angular-rate gyroscope measurements from a thermally calibrated InertialSense IMU. This IMU is also a micro GPS-INS and is capable of producing a full navigation solution for truth comparison.

The hardware implementation introduced three sources of error that were not initially considered in the simulation: calibration error, timing error, and initialization error. Without addressing these errors, the filter would either diverge or give unsatisfactory performance. Figure 6 demonstrates the sensitivity of the estimator accuracy to these types of errors when they are deliberately introduced into the simulation without correcting for them as described in the following sections. The results in Figure 6 show that estimator performance was relativity insensitive to body (IMU) to camera position offset p_b^c calibration error, and therefore, the calibration error discussion below will focus primarily on body (IMU) to camera angular offset q_b^c error. Figure 7 further demonstrates that estimator performance is significantly degraded by these sources of error.

4.3.1 | Calibration error

Initial testing showed that a satisfactory calibration of the camera's intrinsic parameters can be performed prior to the flight. Error in the extrinsic parameters, specifically the body (IMU) to camera rotation angles, however, was detrimental to the filter performance. Since the transformation from the body to camera is used in the measurement-model calculation of the residual r and measurement Jacobians H_x and H_f (see Appendix B1), the error is correlated with every feature measurement, and it causes significant bias in the estimates. As shown in the lower left plot in Figure 6, position error increases as angular errors are added to the body-to-camera offset. With 0.8° of error or less, the RMS error appears to increase quadratically with calibration error. When the calibration error is greater than approximately 0.8°, the outlier rejection within the filter begins to prevent measurements from negatively affecting the estimates, and RMS error continues to increase approximately linear with additional calibration error.

This calibration error was accounted for, and removed in flight, by introducing the camera rotation to the state vector, making Equation (1) become

$$\mathrm{x} = [\mathrm{x}_{\mathrm{imu}}^{\mathrm{T}} \quad \mathrm{q}_b^c \quad \pi_0 \quad \pi_1 \quad \dots \quad \pi_k]^{\mathrm{T}},$$

where q_b^c is a quaternion of the rotation from the body (IMU) frame to camera frame. The covariance was initialized to a relatively large value and allowed to converge over time. We note that the introduction of q_b^c to the state vector makes the use of $\mathrm{q}_n^{i_k}$ in Equation (2) a non-minimum representation of the state because the camera pose includes the calibrated camera rotation. This slight mismodeling, and the fact that the camera calibration is both static and minimally observable in this case (monocular camera and unknown features), necessitates the use of partial updates[40] (defined in Appendix E1) to avoid inaccurate updates to q_b^c during in-flight calibration. The body-to-camera position offset p_b^c was not included in the online calibration for two reasons. First, the physical

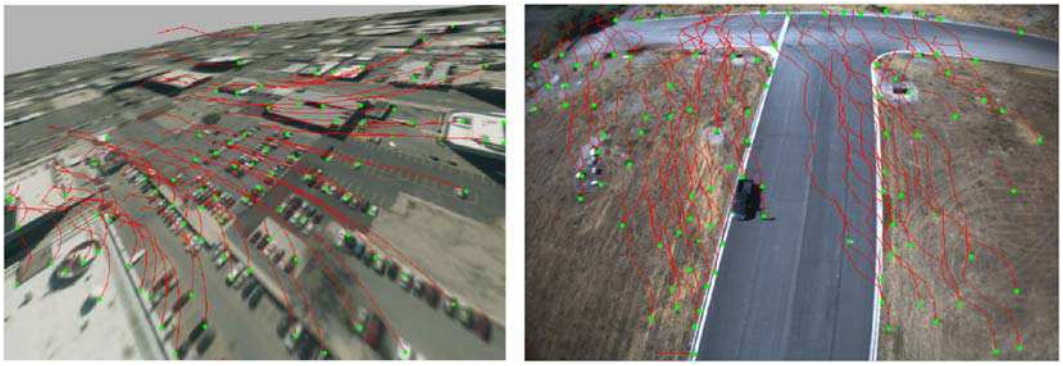

FIGURE 5 Left: Simulated camera image. Right: Real image from a flight test. Image features are shown as dots, and their track histories are also overlaid. Simulating the camera image rather than simulating the features improved the simulation fidelity [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

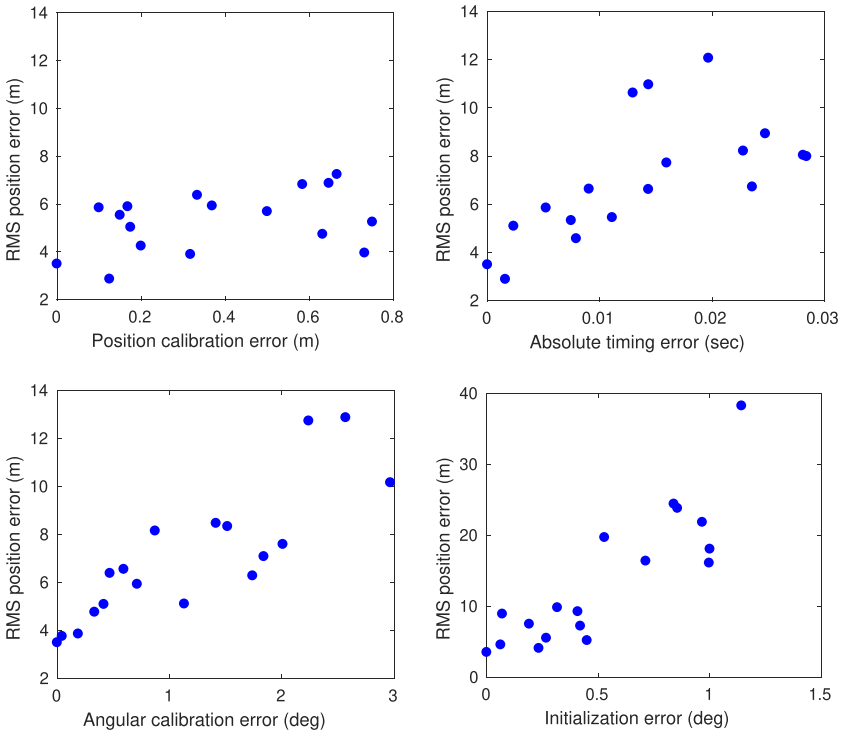

FIGURE 6 Using a simulation of an aircraft, the effect of unmodeled calibration, timing, and initialization errors (in the two left, upper right, and lower right plots, respectively) are shown on the root-mean-squared (RMS) position error for a 60 s flight and approximately 620 m trajectory length. Calibration error was introduced by adding both a position offset from the body frame (IMU) to camera frame (p_b^c) and an angular offset about a random axis to the rotation from the body frame (IMU) to camera frame (q_b^c). Timing error was added to the image-feature measurements. Initialization error was added to the initial roll and pitch estimate, but no error was added to the acceleration bias (see the discussion in Section 4.3.3) [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

distance was small (about 2 cm) compared to the baseline to observed features, making it nearly unobservable, and second, testing using the simulation environment showed the filter performance was insensitive to errors in p_b^c, as shown in Figure 6.

In general, the mathematical development provided in this paper, including the appendices, corresponds to the original state vector Equation (1). The inclusion of q_b^c as an estimate introduces only minor modifications and similar efforts are discussed in Shelley.[22]

4.3.2 | Timing error

The upper right plot in Figure 6 demonstrates that as little as 10 ms of timing error in the sensor measurements is enough to approximately double the RMS position error, and when the timing error was 20 ms or more, the timing error produced biases in the estimated motion that mostly corresponded with the direction of travel. This is due to image measurements, which are effectively position updates, being applied later than when they were produced. Since the TX2 was not running a real-time operating system, the filter depended on accurate time stamps on each measurement. The image measurements were prevented from receiving an accurate time stamp by significant and varying delays introduced while transferring the images from the camera to the computer. To overcome this delay, the camera was configured to provide a strobe pulse that coincided with the camera shutter. Each pulse

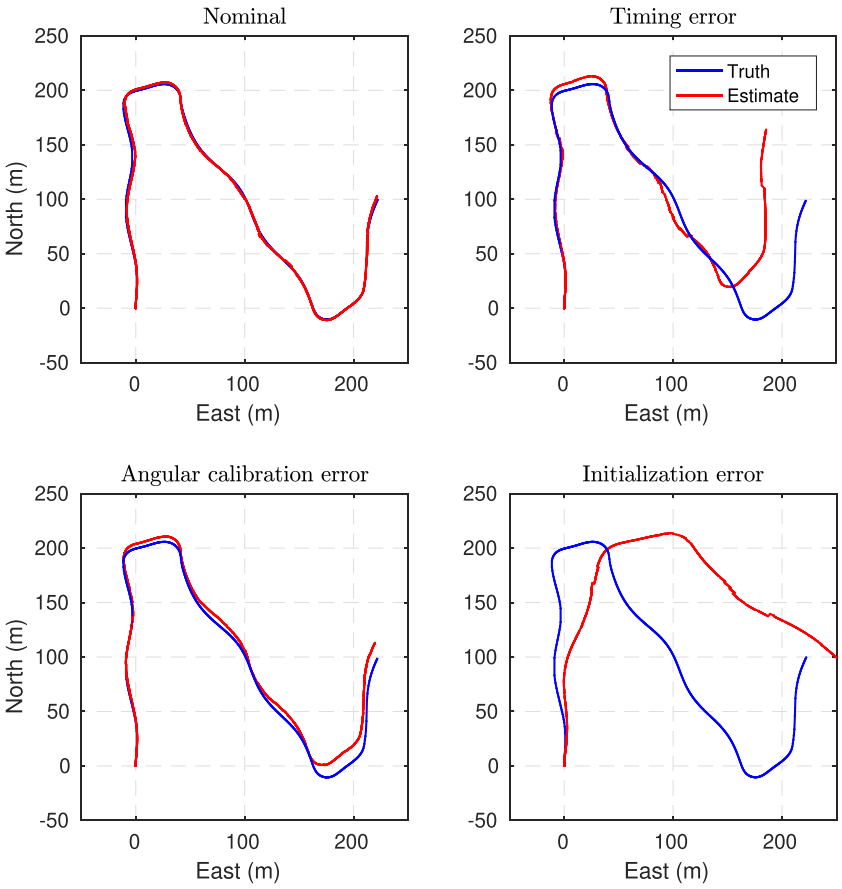

FIGURE 7 Unmodeled calibration, timing, and initialization errors cause unacceptably large pose estimation errors as shown in the lower left, upper right, and lower right plots compared to the nominal estimator performance shown in the upper left plot. The errors include 5° of body (IMU) to camera angular calibration error, 0.08 s of timing error, and 1.5° of initialization error and are representative of errors that could be expected in a hardware demonstration. These error sources produced root-mean-squared (RMS) position estimation errors of 36.9, 21.2, and 82.6 m in their respective tests [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

caused the InertialSense IMU to publish the current time stamp. Once the time stamps were received on the TX2 computer, they were added to a queue and used to re-stamp the images once they were fully transferred from the camera. Since both the IMU measurements and the GPS-INS truth navigation solution also originated from the InertialSense IMU, every necessary measurement was stamped relative to the same time reference. Because recombining time stamps with their corresponding images depended only on their order in the queue, this method was only reliable up to an image frame rate of 7 Hz.

Once the measurements were stamped with the correct time, they were used by the filter even though the images (and thus the feature track measurements) arrived after the IMU measurements. The filter uses the out-of-order measurement scheme described in Wheeler et al.,[12] where sensor measurements and filter state snapshots are stored in a priority queue. When an old measurement arrives, the filter rewinds to just before the new measurement, applies it, and then fast-forwards (and updates the snapshots) to the latest measurement. Because image measurements incur more delay and IMU measurements only propagate the IMU state x_{imu}, this method is computationally feasible and runs in real time.

4.3.3 | Initialization error

The MSCKF measurement model, including augmenting the state vector with the time-history of image transformation, performs well once the filter is converged to the true value but suffers when there are significant errors in the IMU state. This is particularly problematic during initialization. Assuming the aircraft is not moving when the filter starts, position and velocity can be initialized to zero with negligible covariance, and the angular-rate bias can be determined from the first few measurements. The filter must be initialized, however, to an unknown attitude q_n^b and acceleration bias $\boldsymbol{\beta}a$. These states cannot easily be sensed by measuring the gravity vector because attitude errors and acceleration bias are correlated. The lower right plot in Figure 6 shows how filter performance is sensitive to initial roll and pitch attitude error. In the simulation tests, the initial covariance for the attitude states did not accurately model the error. If the initialization errors are small and the flight is sufficiently long, the initialization errors often die out as the estimates converge, but not until after the odometry has accumulated error that can significantly reduce the accuracy of the navigation solution.

Our strategy for initialization of the filter (in the hardware flight tests only) included using the InertialSense

GPS-INS attitude to initialize the filter attitude and using its reported body-frame velocity as a measurement for the first 45 s to help the acceleration bias states converge. Using the velocity as a measurement was advantageous because the relative-reset step did not affect how the measurements were utilized by the filter. Conversely, using the reported position would have required transforming the measurement into the node frame using a potentially inaccurate attitude estimate. The use of the partial update[40] on acceleration bias and on attitude states improved the consistency of the filter and limited its confidence of the estimates. The partial update is detailed in Appendix E1.

5 | FRONT-END RESULTS

The filter was first tested in the high-fidelity fixed-wing simulation described above. Because the filter publishes the position relative to the most recent reset-step node frame, to plot and analyze the performance of the filter, the state must be put into the global frame or the truth must be put in the node frame. In Figure 8, the estimates are put into the global frame by composing current state with the previously published edge transforms, similar to a back-end graph. Figure 8 compares the front-end results for 60 s of three different simulated trajectories and shows that the results suffer when the aircraft flies straight and level, but improve as the aircraft turns.[41] This phenomenon is particularly important for fixed-wing aircraft because they often fly over greater distances to accomplish mission objectives. For the trajectory with the most turns, the total accumulated error is shown as less than 1% of the distance traveled, where the distance traveled is the integrated flight-path length.

Hardware flight results were also obtained by flying the aircraft in Figure 1 over a 6 km trajectory. The front-end estimates were produced on the aircraft in real time. The true flight trajectory and accumulated estimates are shown in Figure 9. Because the aircraft was flown by a remote pilot, the trajectory is only roughly straight and level, that is, other than during take-off and landing. The effects of the initialization error can also be seen in the first 100 s of the flight when the scale error is much greater. The filter estimates from 150 to 350 s are also shown to compare the performance after the estimates converge and the effects of initialization are minimized. The total accumulated error of the filter estimate from 150 to 350 s was approximately 2.5% of the distance traveled. The entire dataset was 388 s long, and the total accumulated error for the flight, corresponding to the trajectory labeled "front-end estimate" in Figure 9, was 5.3% of the distance traveled.

The results in Figure 10 show that the estimates track the true motion of the aircraft. The effect of the relative reset can be seen when position and heading angle abruptly return to zero, as previously defined. During a reset step, a new origin is declared at the position of the aircraft, ensuring both the true and estimated values return to zero. The estimated velocity and roll and pitch angles do not reset. The amount of time between resets varies depending on there being more than nine continuous feature tracks but generally is between one and seven seconds.

Figure 11 shows 3σ bounds around the relative error. The bounds are calculated from the square root of the diagonal terms of the covariance matrix $\mathbb{J}$. From these plots, it appears that the filter is consistent, and the uncertainty grows approximately linearly with the distance traveled. The effect of the relative-reset step can be seen as the error and 3σ bound both return to zero for the position and heading states. The filter also publishes its position and heading state (and associated covariance) from just prior to the reset to be used in a relative-navigation, back-end pose graph.

6 | BACK END

The results presented thus far, including those from both simulation and hardware flight tests, have been for the front-end estimator exclusively. We now briefly turn our attention to the back end with subsequent sections presenting results highlighting how the front end and back end work together to improve localization accuracy.

In all prior relative-navigation work, the global back-end graph optimization has assumed that the edges published from the front end have been statistically independent, meaning errors in one edge were uncorrelated with errors in all others. This assumption has worked well when the errors in the estimated linear velocities remained small due to direct depth measurements[13] or flying with s-turns to help velocity remain observable.[16] The assumption becomes less appropriate, however, when errors are more significant. In this paper, velocity error is more significant for a fixed-wing UAS flying straight and level over extended periods causing the forward velocity to be less observable[41] and therefore the edge errors to be more correlated. Further, because features tracks are discarded and reinitialized at each keyframe and associated relative-reset step, the velocity estimates are likely degraded. In this section, we introduce a method for modeling the correlation of the velocity error between edges in the back end. Modeling this correlation improves the ability of the back-end optimization to remove error when intermittent global measurements or other loop-closure-like constraints are available. The method is aligned with our belief that the back end provides value and flexibility for the GPS-denied localization problem and small errors intro-

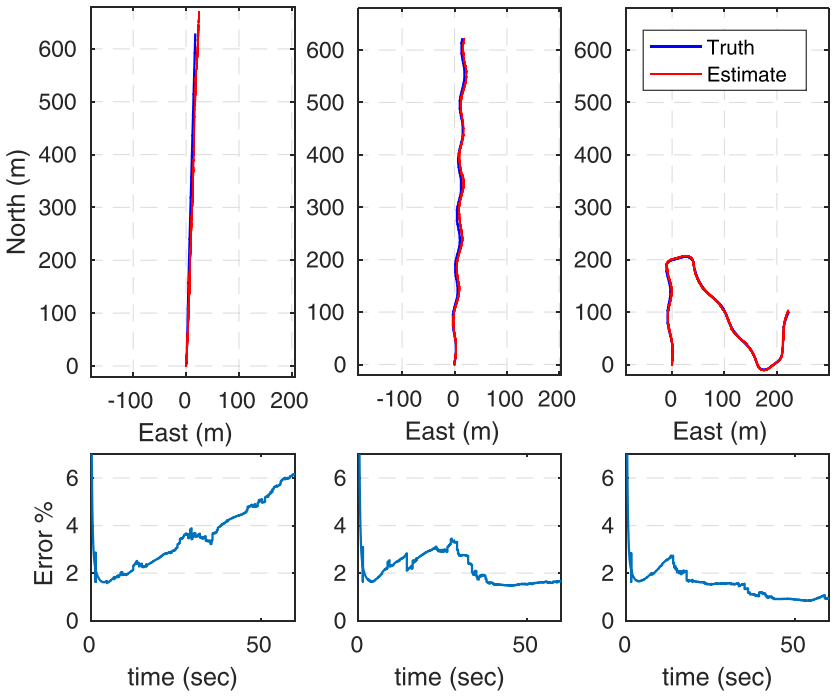

FIGURE 8 Top: Three simulation flight tests where the true path is compared to the accumulated estimate. Bottom: The error as a percent of the distance traveled is shown for the first 60 s of each flight. There are significant bias errors when the aircraft flies straight and level due to velocity being less observable for a monocular VIO. The estimates improve significantly when a non-straight trajectory is used, even with a slight sinusoidal s-turn. In the flight with the most deviation from straight, the accumulated error is ultimately less than 1% of the distance traveled [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

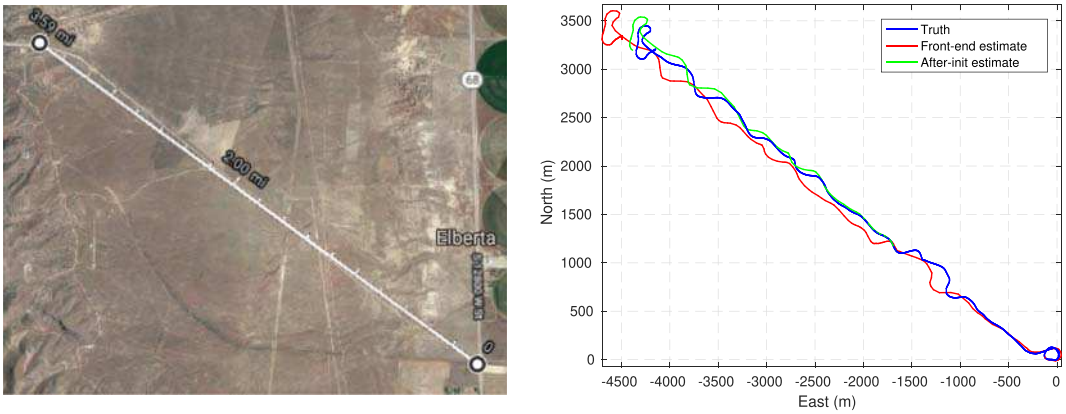

FIGURE 9 Left: The aircraft flight path during a manually flown flight test. Right: The true path is compared to the accumulated estimate. The estimates from 150 to 350 s are also shown to compare the result of removing most of the effects of the initialization errors. Other than the take-off and landing circles, the aircraft was flown approximately straight and level over a 6 km distance. Notice the scale bias in the estimates [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

duced by the relative reset are mitigated and offset by the back-end optimizations.

Velocity errors can be accounted for over a single edge by applying a scale bias to the published position. Because velocity errors persist through a relative-reset step, they are also correlated between consecutive edges. This correlation is similar to how gyro bias walks and is correlated over time.

In the back end, we model a two-dimensional slowly varying bias walk using trinary factors for edges (E_k), which are similar to IMU preintegration factors that account for IMU bias.[42] Each node variable N includes the global north and east position ($\mathrm{p} = [ne]^{\mathrm{T}}$) and global heading ($\psi$), and each bias variable B includes the scale bias ($b = [b_x b_y]^{\mathrm{T}}$). The factor is then defined by a loss function ℓ that effectively rotates the change in global position into the previous node frame, applies the bias scale, and then subtracts the measured odometry. The function is defined as

$$\ell(N_k, N_{k+1}, B_k, m) = \begin{bmatrix} (\cos(\psi_k)(n_{k+1} - n_k) + \sin(\psi_k)(e_{k+1} - e_k))b_x - m_x \\ (-\sin(\psi_k)(n_{k+1} - n_k) + \cos(\psi_k)(e_{k+1} - e_k))b_y - m_y \\ (\psi_{k+1} - \psi_k) - m_\psi \end{bmatrix},$$

where m is the measurement of the edge odometry published by the front end at each relative reset and

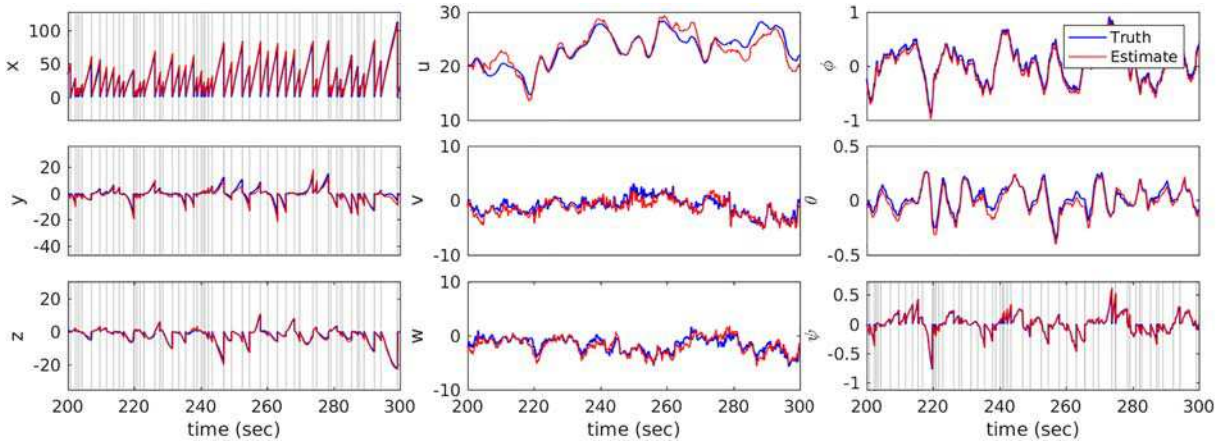

FIGURE 10 The true and estimate states of a UAS in flight relative to the most recent node. Position, velocity, and attitude states are shown from left to right, respectively. The relative reset associated with the declaration of new nodes are shown with vertical lines [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

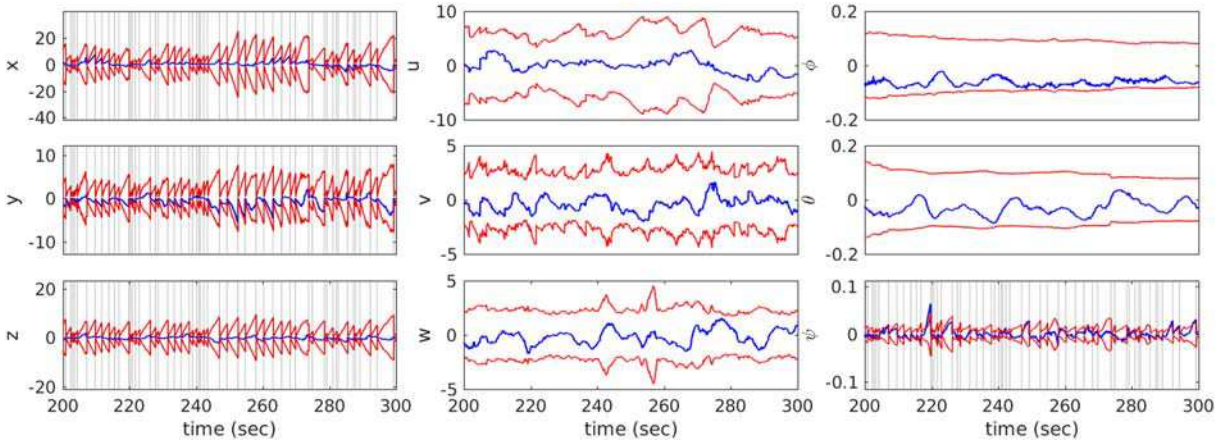

FIGURE 11 State error is shown for position, velocity, and attitude states (from left to right, respectively). The 3σ uncertainty bounds come from the square root of the diagonal terms of the covariance matrix. Sharp decreases in the position error bounds are due to the relative-reset steps that are also indicated by vertical lines [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

includes the change in position (m_x and m_y) and heading (m_ψ). The factor graph is shown in Figure 12 with the same depiction style as in Dellaert[17] and is implemented using GTSAM.

Unlike gyrocope bias, velocity errors are not a stochastic process. The autocorrelation of the velocity errors depends on the observability of the velocity and thus the flight trajectory of the aircraft, and the scale error correlation between edges depends on the time between resets. This means modeling the bias scale as a random walk is a simplification. In practice, the covariance of the binary factor (R) between bias variables (B_k) is hand tuned. In the results shown below, these factors use $\boldsymbol{\Sigma} = 0.0001\mathrm{I}_{2\times 2}$, where $\mathrm{I}_{2\times 2}$ is a 2 by 2 identity. This extension may be most relevant for fixed-wing aircraft using VIO but could also be applied to all previous relative-navigation work where velocity errors persist through the reset step.

Once the factors are defined, they can be added to the graph with connections to the appropriate variables. The variables in the graph are the global north, east, and yaw poses of the keyframe nodes (N) and the bias (B) at each odometry and are initialized appropriately. Finally, GTSAM provides functions to optimize the graph such that the loss of all the combined factors is minimized. The resulting graph, and thus the global state, is produced by optimizing after all the factors have been added. The full details of factor-graph optimization are extensive but can be ascertained from Dellaert,[17] Kummerle et al.,[18] Forster et al.,[42] and elsewhere.

Figure 13 shows several examples of the advantage of including these additional factors when there is significant scale bias in the estimates. When several GPS measurements are included in the optimization, the scale bias can cause the optimized trajectory between the measure-

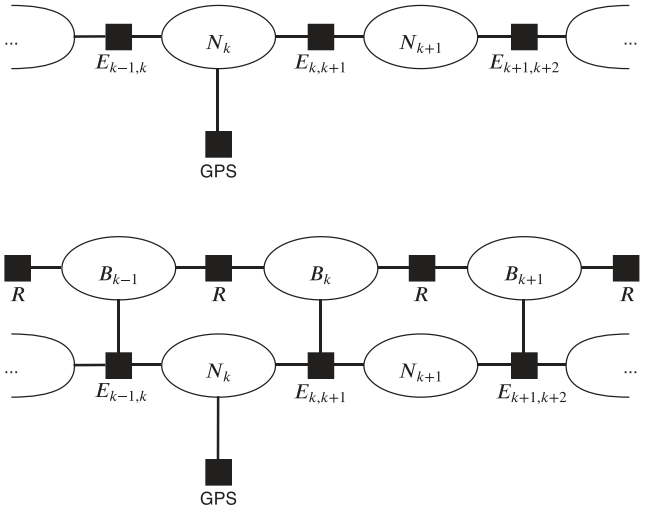

FIGURE 12 Factor graphs used in the global back end where values are ovals and measurement factors are squares. Top: Original graph where nodes (N) are connected by odometry edge factors (E) from the front-end filter and the edges are modeled as independent. GPS measurements or other global constraints can be opportunistically added as unary factors. Bottom: In our method, edges become a trinary factor, which also considers a bias scale variable (B). Because velocity errors persist through a relative-reset step and errors in the edges are correlated, the bias is modeled as a random walk through the use of binary factors (R) which are initialized as identity with small, non-zero covariance

ments to bulge out. This is because the optimizer prefers to add a small amount of heading change to each edge rather than reduce the distance between each edge transformation. Properly modeling the correlated scale error between edges with scale-bias factors enables the optimizer to remove the bulging and make the graph consistent with the true trajectory. The bulging effect is particularly dramatic, as seen in Figure 13, under conditions where the scale bias is large and GPS measurements are intermittent enough for a relatively large amount of heading uncertainty to accumulate. The figure demonstrates that when there is only one GPS measurement, the optimization cannot remove scale errors. Finally, it shows that when GPS measurements are frequent and regular, the need for scale-bias factors is mitigated since the GPS measurements correct for the scale bias before significant path errors accumulate.

The flight-test data used to construct Figure 13 included significant scale bias in the front-end estimates. As discussed previously, most of the bias occurred near the beginning of the trajectory due to initialization errors. The bias was up to a factor of approximately 1.2 and varied along the trajectory due to changes in observability of the forward velocity.

7 | FULL SYSTEM RESULTS

To demonstrate the value of the proposed relative front end, the full localization solution is produced in a single back-end graph using the published edges from pre-recorded hardware flight tests. The results shown in this section utilize a two-dimensional graph that is optimized post-process and in a single batch, although similar back-end architectures have been shown to work on single multirotor aircraft for both localization and navigation in near real time.[13] The back-end results simulated global measurements calculated from the reported states from the InertialSense GPS-INS that were used for truth comparison.

Figure 14 incorporates three simulated GPS measurements into the graph. The GPS measurements were added to the graph with a 0.32 m standard deviation error. These measurements help remove initialization errors and provide constraints to optimize the scale factors introduced previously. The results represent a mission profile where GPS is available until the aircraft enters an area where the GPS is spoofed or jammed or otherwise unavailable.

The results in Figure 15 also incorporate simulated global measurements, but in this example, the back end utilizes five distance measurements to two static features. The range measurements represent measurements to a distance-measuring-equipment (DME) transponder or similar fixed ground-based range station. The range measurements were simulated as having a 0.71 m standard deviation error. These results again show the ability of the back end to improve global accuracy. If a given range measurement had been used as an update to the EKF, however, it may have caused the filter to become inconsistent or even diverge, depending on the amount of uncertainty. By incorporating these inputs in the back end and not in the filter, this approach avoids the worst case scenario while still improving the localization.

Finally, the results in Figure 16 use relative inter-vehicle range measurements (with a simulated 0.71 m standard deviation) between aircraft flying in a small formation rather than from a stationary ground feature only. The measurements allow the aircraft to cooperatively localize. The three trajectories depicted are from separate flight tests of the test aircraft in Figure 1. Each trajectory includes significant initialization errors with bias scale. The center aircraft receives intermittent, simulated global position measurements such as those from GPS or a computationally expensive satellite-image-based place recognition system.[24] The results show that not only do the outside aircraft receive the benefit of the global measurements but also the relative position of the formation is maintained.

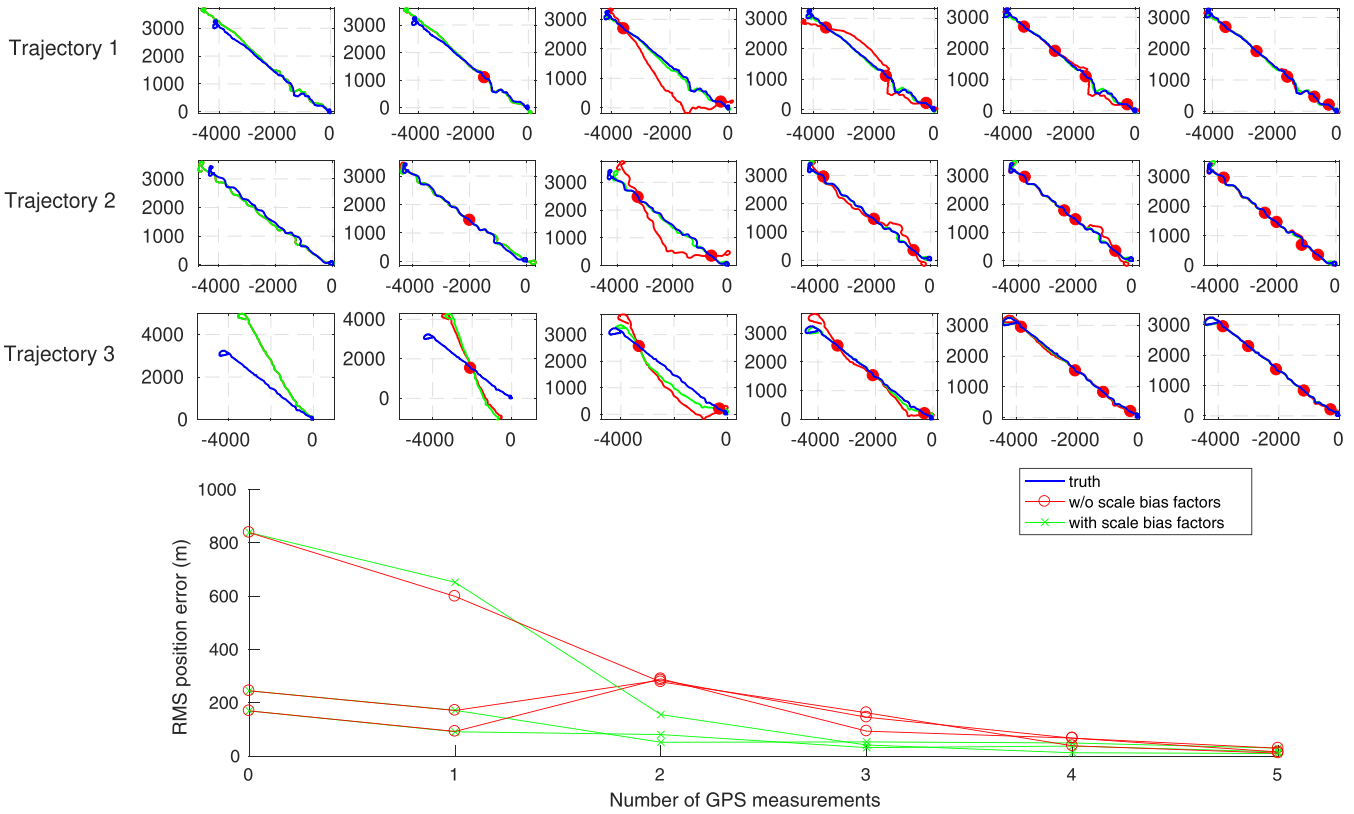

FIGURE 13 Back-end optimizations of three flight-test trajectories with and without scale-bias factors. The graphs were constructed from front-end estimates that included significant scale bias and used varying numbers of simulated intermittent GPS measurements. The upper plots show the optimized trajectories compared to the true paths. The lower plot shows the RMS position error of the optimized results as a function of the number of GPS measurements. The graphs that included the scale-bias factors were able to optimize the scale bias out of the estimates to match the true trajectory with greater accuracy than the graphs that did not include the additional factors [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

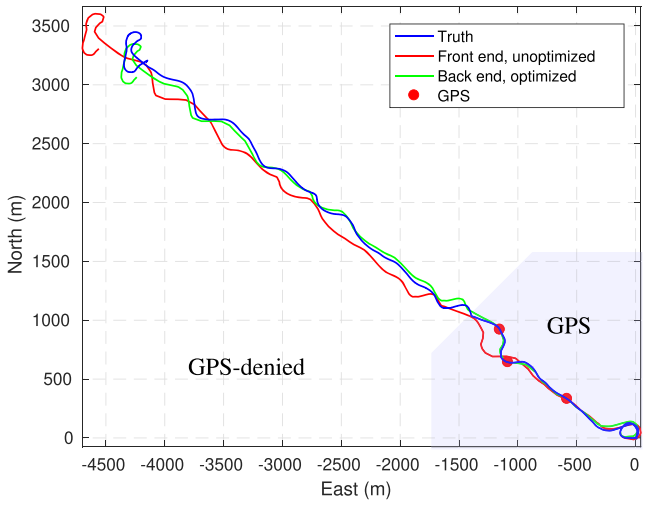

FIGURE 14 Back-end optimization of the flight-test graph. The unoptimized trajectory is the raw front-end estimates in a graph but before optimization (corresponding with the complete estimated trajectory in Figure 9). Three simulated GPS measurements were added to help with initialization. Scale-bias factors were used to remove the scale error of the estimated edges. The shaded background indicates areas where GPS was available [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

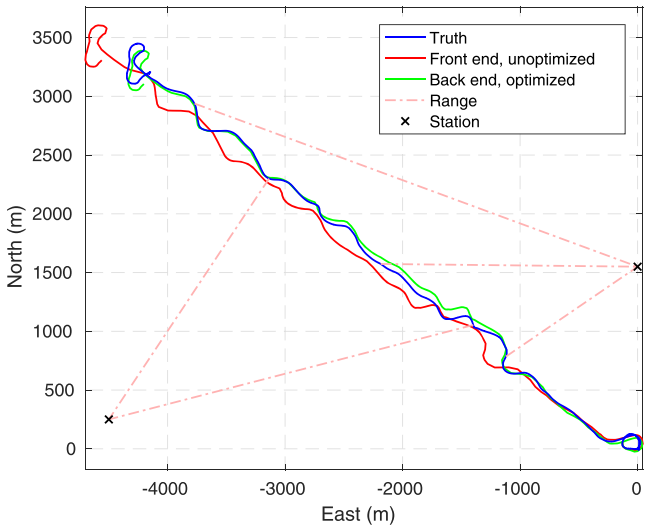

FIGURE 15 Back-end optimization of the flight-test graph includes five simulated range measurements to two static features or DME stations. The results improve significantly by removing initialization and scale errors [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

These later results do not account for several aspects of a full multi-vehicle cooperative solution, including the necessary communication links between vehicles. They do show that the proposed method holds promise for these scenarios. For example, a rough estimate of the total amount of sensor data processed is 5.8 GB for all three vehicles, where as the back-end graph is constructed from less

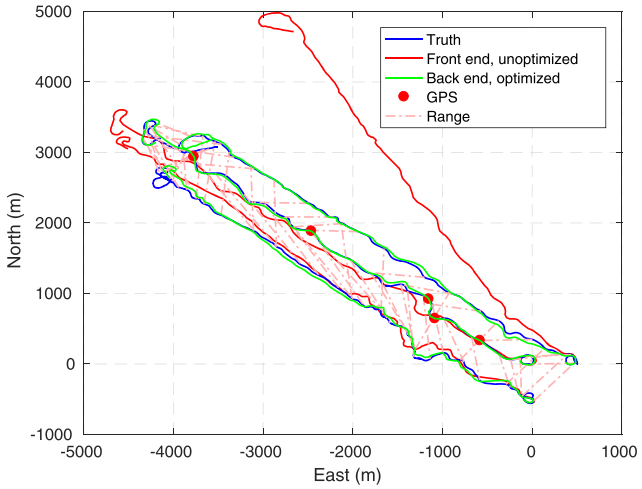

FIGURE 16 Back-end graph includes edges from three vehicles, or more accurately three flight tests, with starting location of the second and third vehicles artificially offset by 500 m south and 500 m east, respectively. The graph includes simulated inter-vehicle range measurements and five GPS measurements for the center vehicle. The localization accuracy of all vehicles improves and the relative position of the swarm is maintained [Color figure can be viewed at wileyonlinelibrary.com and www.ion.org]

than 0.15 MB of data. This suggests the potential for both the scalability of a multi-agent system as well as robustness to communication loss or delay.

8 | CONCLUSION

This paper has demonstrated a method for localizing a fixed-wing UAS in environments where GPS is either unavailable or unreliable. This work has used the relative navigation architecture, previously implemented for multirotor UAS, as a guide. The front-end filter depends on a camera and an IMU for sensing and has no other specific requirements. It uses a VIO approach to estimate the motion of the aircraft and regularly publish transformations that can be used in a back-end graph. The filter uses a modified MSCKF measurement model and the relative-reset step. The filter also makes no assumptions about the scale or distance to observed image features.

The filter was tested first in simulation. The simulation testing showed the filter accuracy is trajectory-dependent due to the lack of observability of the velocity in straight-and-level flight. In simulation, the total accumulated error is demonstrated as less than 1% of the distance traveled, provided there is sufficient turning to maintain observability of the velocity estimate. The front-end filter was also demonstrated in a hardware flight test. The implementation details of the flight test, including our efforts to account for calibration, timing, and initialization errors, were discussed. After the initialization errors were

removed, the filter was accurate and ultimately accumulated error of approximately 2.5% of the distance traveled.

The value of this approach can best be evaluated by considering the whole relative-navigation architecture. The estimates from the relative-navigation, front-end, VIO estimator are used in a back-end graph. The back-end graph is responsible for both representing and optimizing the global state, which is necessary for accomplishing a global mission. For the back end to more accurately utilize our front-end estimates, we introduced a scale-bias model to account for the correlation of the scale errors between edges.

This work has also demonstrated the use of the back end and graph optimization to incorporate other constraints, such as opportunistic, geo-referenced measurements. Such measurements can be problematic for a front-end EKF because large covariance and filter inconsistency can cause the update to produce large jumps in the state. Because the proposed relative front-end operates independently from the back end, jumps in the back-end state do not directly affect the control of the vehicle. The full system is able to operate over significant periods without global information and whenever it becomes available the system can seamlessly utilize it in the back end to improve localization.

Finally, we have shown the potential for the proposed method and the relative-navigation architecture to be used in multi-vehicle cooperative localization scenarios. The back-end graph is able to efficiently incorporate the odometry edges from multiple vehicles as well as relative inter-vehicle and global measurements. Cooperative, multi-vehicle localization will be explored in future work.

ACKNOWLEDGMENTS

This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry University Cooperative Research Center (IUCRC) under NSF award no. IIP-1650547 along with significant contributions from the AFRL Munitions Directorate and other C-UAS industry members. Gary Ellingson also received support from the Utah NASA Space Grant Consortium Fellowship.

ORCID

Gary Ellingson https://orcid.org/0000-0002-6278-4301

REFERENCES

1. Weiss S, Achtelik MW, Lynen S, Chli M, Siegwart R. Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. *2012 IEEE International Conference on Robotics and Automation (ICRA)*. Saint Paul, MN; May 2012:957-964.
2. Jones E, Vedaldi A, Soatto S. Inertial structure from motion with autocalibration. *Workshop on Dynamical Vision*. 2007;25.
3. Barfoot TD, Furgale PT. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*. 2014;30:679-693.
4. Thrun S, Burgard W, Fox D. *Probabilistic Robotics*. Cambridge, MA: MIT press; 2005.
5. Leishman RC, Macdonald JC, Beard RW, McLain TW. Quadrotors and accelerometers: State estimation with an improved dynamic model. *IEEE Control Systems Magazine*. 2014;34:28-41.
6. Beard RW, McLain TW. *Small Unmanned Aircraft: Theory and Practice*. Princeton, NJ: Princeton University Press; 2012.
7. Wheeler DO, Koch DP, Jackson JS, McLain TW, Beard RW. Relative navigation: A Keyframe-based approach for observable GPS-degraded navigation. *IEEE Control Systems*. 2018;38:30-48.
8. Shen S, Mulgaonkar Y, Michael N, Kumar V. Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV. *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China; May 2014:4974-4981.
9. Chambers A, Scherer S, Yoder L, et al. Robust multi-sensor fusion for micro aerial vehicle navigation in GPS-degraded/denied environments. *2014 American Control Conference*. Portland, OR; June 2014:1892-1899.
10. Chambers A, Achar S, Nuske S, et al. Perception for a river mapping robot. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA; September 2011:227-234.
11. Leishman RC, McLain TW, Beard RW. Relative navigation approach for vision-based aerial GPS-denied navigation. *Journal of Intelligent & Robotic Systems*. 2014;74(1–2):97-111.
12. Wheeler DO, Nyholm PW, Koch DP, et la. Relative navigation in GPS-degraded environments. *Encyclopedia of Aerospace Engineering*. 2016.
13. Wheeler DO, Koch DP, Jackson JS, et al. Relative Navigation of Autonomous GPS-Degraded Micro Air Vehicles. Available at https://scholarsarchive.byu.edu/facpub/1962/; 2017.
14. Leutenegger S, Siegwart RY. A Low-cost and fail-safe inertial navigation system for airplanes. *2012 IEEE International Conference on Robotics and Automation (ICRA)*. Saint Paul, MN; May 2012:612-618.
15. Lewis BP, Beard RW. A framework for visual return-to-home capability in GPS-denied environments. *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. Arlington, VA; June 2016:633-642.
16. Ellingson G, Brink K, McLain T. Relative visual-inertial odometry for fixed-wing aircraft in GPS-denied environments. *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. Monterey, CA: April 2018:786-792.
17. Dellaert F. Factor graphs and GTSAM: A hands-on introduction. Technical Report number GT-RIM-CO&R-2012-002; 2012.
18. Kummerle R, Grisetti G, Strasdat H, Konolige K, Burgard W. g2o: A general framework for graph optimization. *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*. Shanghai, China; May 2011:3607-3613.
19. Brink KM. Multi-agent relative pose estimation: Approaches and applications. *Proceedings Volume 10651 Open Architecture/Open Business Model Net-Centric Systems and Defense Transformation 2018*. Orlando, FL; May 2018:106510D.

20. Zhang J, Kaess M, Singh S. Real-time depth enhanced monocular odometry. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*. Chicago, IL; September 2014:4973-4980.
21. Koch DP, Wheeler DO, Beard R, McLain T, Brink KM. Relative Multiplicative Extended Kalman Filter for Observable GPS-Denied Navigation. Available at https://scholarsarchive.byu.edu/facpub/1963/. 2017.
22. Shelley MA. Monocular Visual Inertial Odometry on a Mobile Device. Master's thesis. Technischen Universitat Munchen, Munich, Germany, 2014.
23. Clement LE, Peretroukhin V, Lambert J, Kelly J. The Battle for filter supremacy: A comparative study of the multi-state constraint Kalman filter and the sliding window filter. *2015 12th Conference on Computer and Robot Vision (CRV)*. Halifax, NS; June 2015:23-30.
24. Mourikis AI, Trawny N, Roumeliotis SI, et al. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*. 2009;25:264-280.
25. Delmerico J, Scaramuzza D. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. *Memory*. 2018;10:20.
26. Li M, Mourikis AI. High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research*. 2013;32(6):690-711.
27. Forster C, Zhang Z, Gassner M, Werlberger M, Scaramuzza D. SVO: semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*. 2017;33(2):249-265.
28. Bloesch M, Burri M, Omari S, Hutter M, Siegwart R. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*. 2017;36(10):1053-1072.
29. Mourikis AI, Roumeliotis SI. A multi-state constraint Kalman filter for vision-aided inertial navigation. *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*. Roma, Italy; 2007:3565-3572.
30. Markley FL. Attitude error representations for Kalman filtering. *Journal of Guidance, Control, and Dynamics*. 2003;26(2):311-317.
31. Trawny N, Roumeliotis SI. Indirect Kalman Filter for 3D Attitude Estimation. University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep. 2; 2005.
32. Hartley RI. In defence of the eight-point algorithm. *Proceedings of the Fifth International Conference on Computer Vision*. Washington, DC; June 1995:1064-1070.
33. Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence*. Vancouver, BC; August 1981:674-679.
34. Tomasi C, Kanade T. Detection and tracking of point features. *Tech. Rep. CMU-CS-91-132*: Carnegie Mellon University; 1991.
35. Jianbo S, Tomasi C. Good features to track. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Seattle, WA; June 1994:593-600.
36. Ellingson G, McLain T. ROSplane: Fixed-wing autopilot for education and research. *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. Miami, FL; June 2017:1503-1507.
37. Jackson J, Ellingson G, McLain T. ROSflight: A lightweight, inexpensive MAV research and development tool. *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. Arlington, VA; June 2016:758-762.
38. Furrer F, Burri M, Achtelik M, Siegwart R. RotorS—A modular gazebo MAV simulator framework. In: Koubaa A, ed. *Robot Operation System (ROS)*. Vol 1. New York, NY: Springer; 2016:595-625.
39. Maybeck PS. *Stochastic Models, Estimation, and Control*, Vol. 3. Cambridge, MA: Academic Press; 1982.
40. Brink KM. Partial-update Schmidt–Kalman filter. *Journal of Guidance, Control, and Dynamics*. 2017;40(9):2214-2228.
41. Rutkowski A. The most accurate path from point A to point B is not necessarily a straight line. *AIAA Guidance, Navigation, and Control Conference*. Minneapolis, MN; August 2012:4761-4769.
42. Forster C, Carlone L, Dellaert F, Scaramuzza D. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. *Proceedings of the Robotics Science and Systems XI Conference*. Rome, Italy; July 2015.
43. Schmidt SF. Application of state-space methods to navigation problems. *Advances in Control Systems*. 1966:293-340.

How to cite this article: Ellingson G, Brink K, McLain T. Relative navigation of fixed-wing aircraft in GPS-denied environments. *NAVIGATION-US*. 2020;67:255–273. https://doi.org/10.1002/navi.364

APPENDIX A: MEASUREMENT JACOBIANS

This section defines the measurement Jacobians H_x and H_f that are necessary for the MSCKF measurement model. As described previously, the measurement is the pixel coordinates of a feature track. Thus, we begin by providing a camera projection function $h(p)$ to project a feature in the image frame onto a pixel coordinate while accounting for the camera matrix and distortion parameters. The projection function enables the construction of the predicted measurement and residual. Finally, we provide the partial derivatives to fully define the Jacobians.

If we first neglect distortion, the camera projection function consists of normalizing the feature position vector (in the image frame) p_i^f by the depth and multiplying it by the camera matrix K or

$$h(p_i^f) = K \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{A1}$$

where

$$p_i^f \triangleq \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix},$$

$$u \triangleq \frac{p_x}{p_z},$$

and

$$v \triangleq \frac{p_y}{p_z}.$$

Next, a camera distortion model that includes radial coefficients (k_1, k_2, and k_3) and tangential coefficients (t_1 and t_2) is applied to u and v using

$$r = u^2 + v^2$$

$$d_r = (1 + k_1 r + k_2 r^2 + k_3 r^3)$$
$$u' = d_r u + 2uvt_1 + (r + 2u^2)t_2$$
$$v' = d_r v + 2uvt_2 + (r + 2v^2)t_1 \ .$$

Finally, the projection function $h(\mathrm{p}_c^f)$ consists of substituting u' and v' for u and v, respectively, into Equation (A1).

Recall that the measurement z_k is the pixel location provided by the tracker for camera image i_k and the least-squares optimization produces p_n^f, the position of the feature in the node frame. The residual r can then be constructed by transforming the feature position to the appropriate camera image frame with

$$\mathrm{p}_{i_k}^f = \mathrm{R}_n^{i_k}[\mathrm{p}_n^f - \mathrm{p}_n^{i_k}]$$

and then projecting it using the projection function $h(\mathrm{p}_c^f)$. Thus, the residual for a single feature tracked over several images is

$$\mathrm{r} = \begin{bmatrix} \mathrm{z}_0 - h(\mathrm{R}_n^{i_0}[\mathrm{p}_n^f - \mathrm{p}_n^{i_0}]) \\ \mathrm{z}_1 - h(\mathrm{R}_n^{i_1}[\mathrm{p}_n^f - \mathrm{p}_n^{i_1}]) \\ \vdots \\ \mathrm{z}_k - h(\mathrm{R}_n^{i_k}[\mathrm{p}_n^f - \mathrm{p}_n^{i_k}]) \end{bmatrix} .$$

Finally, in constructing the measurement Jacobians, we define the partial derivative of the camera projection function as

$$\mathrm{J}_k \triangleq \frac{\partial h(\mathrm{p})}{\partial \mathrm{p}} \ ,$$

the skew-symmetric matrix for a vector as

$$\lfloor a \rfloor = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \ ,$$

and the partial derivative of the residual with respect to the image transformation π_k as

$$\mathrm{H}_{\pi_k} \triangleq \frac{\partial \mathrm{r}}{\partial \pi_k} = \left[-\mathrm{J}_k \mathrm{R}_n^{i_k} \ \mathrm{J}_k \lfloor \mathrm{R}_n^{i_k}[\mathrm{p}_n^f - \mathrm{p}_n^{i_k}] \rfloor \right] \ .$$

The measurement Jacobians are

$$\mathrm{H}_\mathrm{x} = \begin{bmatrix} 0_{2\times 15} & \mathrm{H}_{\pi_0} & 0_{2\times 6} & \dots & 0_{2\times 6} \\ 0_{2\times 15} & 0_{2\times 6} & \mathrm{H}_{\pi_1} & \dots & 0_{2\times 6} \\ \vdots & \vdots & & \ddots & \\ 0_{2\times 15} & 0_{2\times 6} & 0_{2\times 6} & \dots & \mathrm{H}_{\pi_k} \end{bmatrix}$$

and

$$\mathrm{H}_f = \begin{bmatrix} \mathrm{J}_0 \mathrm{R}_n^{i_0} \\ \mathrm{J}_1 \mathrm{R}_n^{i_1} \\ \vdots \\ \mathrm{J}_k \mathrm{R}_n^{i_k} \end{bmatrix} .$$

In practice, several tracks can be used in a single update by vertically stacking residuals r and the measurement Jacobians H_x and H_f.

APPENDIX B: FEATURE OPTIMIZATION

As part of the measurement model described previously, a least-squares optimization is performed to produce the position of the feature in the node frame (p_n^f) using the image transformations (π) and the pixel-coordinate measurements (z). This section defines the Levenberg-Marquardt least-squares optimization that is depicted in Figure 4.

For numerical stability, we optimize an inverse-depth parameterization of the feature position $\rho = [\frac{p_x}{p_z} \ \frac{p_y}{p_z} \ \frac{1}{p_z}]^\mathrm{T}$. Next, we define a function g that receives ρ in the node frame and the camera image pose and produces the feature position transformed into the image frame, or

$$\mathrm{p}_{i_k}^f = g(\rho_n^f, \pi_k) \triangleq \mathrm{R}(\mathrm{q}_n^{i_k}) \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \frac{1}{p_z} \mathrm{R}(\mathrm{q}_n^{i_k}) \mathrm{p}_n^{i_k} \ .$$

The position of the feature can be projected into pixel coordinates of the image using the camera projection matrix and distortion parameters defined in Equation (A1) of the previous section.

We now set up the formal optimization problem

$$\min_{\rho} \quad f(\rho_n^f) = \sum_{k=0}^{n} \left[\mathrm{z}_k - h(g(\rho_n^f, \pi_k)) \right]^2 \ .$$

Finally, after the optimization is completed, the position of the feature p_n^f is extracted from ρ_n^f.

APPENDIX C: RESET STEP

This section describes the process for performing a relative-reset step. The relative-reset step is performed as the position and heading state estimates and their uncertainties are removed from the front-end filter and a new local origin is declared. We discuss first removing the estimate and then the uncertainty from the covariance matrix.

Removing the position from the state vector is performed by simply applying zeroes to the position vector or

$$\mathrm{p}_n^b \leftarrow 0_{3\times 1} \ .$$

The orientation of the body in the node frame is represented by a quaternion q_n^b. Removing the heading from the quaternion is non-intuitive, however, and we instead decompose it to Euler angles, remove the heading, and finally reconstruct the quaternion. Using the common aircraft attitude representation of roll ϕ, pitch θ, and yaw ψ as the active 3-2-1 Euler angles and a Hamiltonian quater-

nion, the decomposition is

$$\phi = \text{atan}\left(\frac{2q_0q_x + 2q_yq_z}{q_z^2 - q_x^2 - q_y^2 + q_0^2}\right),$$
$$\theta = \text{asin}\left(2q_0q_y - 2q_xq_z\right),$$
$$\psi = \text{atan}\left(\frac{2q_0q_z + 2q_xq_y}{q_x^2 - q_y^2 - q_z^2 + q_0^2}\right).$$

The new quaternion is constructed from the roll and pitch angles and zero for yaw ($\psi = 0$) by applying equations

$$q_x = \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2},$$
$$q_y = \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2},$$
$$q_z = \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2},$$
$$q_0 = \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2}.$$

When removing the uncertainty from the covariance matrix P, we only consider the IMU portion of the state or x_{imu}, as the augmented camera image transforms π_k are removed from the state vector during the reset. This is done by constructing a projection matrix N and applying it to P using

$$\text{P} \leftarrow \text{NP}_{15\times15}\text{N}^{\text{T}}$$

where

$$\text{N} \triangleq \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times6} \\ 0_{3\times3} & \text{I}_{3\times3} & 0_{3\times3} & 0_{3\times6} \\ 0_{3\times3} & 0_{3\times3} & \text{N}_q & 0_{3\times6} \\ 0_{6\times3} & 0_{6\times3} & 0_{6\times3} & \text{I}_{6\times6} \end{bmatrix}$$

and

$$\text{N}_q \triangleq \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos^2\phi & -\cos\phi\sin\phi \\ 0 & -\cos\phi\sin\phi & \sin^2\phi \end{bmatrix}.$$

APPENDIX D: PARTIAL UPDATE

The partial-update Schmidt-Kalman filter (PSKF) was introduced in Brink.[40] The PSKF generalizes the classic EKF update step and offers a simple and effective approach to improve the EKF's consistency and robustness when estimating problematic and mildly-observable filter states. It is an extension of the core concept behind the Schmidt-Kalman filter[43] resulting in the ability to reweight the classic filter update to apply anywhere from 0% to 100% of the nominal EKF update for each state at each update step.

Unlike a Schmidt-Kalman filter, which applies a zero update to so-called *nuisance* states and full updates to all other states, the partial updates can be applied both to static nuisance states, as well as classic *full* states. The partial update is performed by first calculating the full Kalman update using

$$\text{K} = \text{P}^-\text{H}^{\text{T}}(\text{HP}^-\text{H}^{\text{T}} + \text{R})^{-1}$$
$$\hat{\text{x}}^+ = \hat{\text{x}}^- + \text{K(r)}$$
$$\text{P}^+ = (\text{I} - \text{KH})\text{P}^-.$$

The state and covariance is then partially updated with

$$\hat{x}_i \leftarrow \gamma_i\hat{x}_i^- + (1-\gamma_i)\hat{x}_i^+$$
$$P_{ij} \leftarrow \gamma_i\gamma_jP_{ij}^- + (1-\gamma_i\gamma_j)P_{ij}^+$$

where γ_i is from a user defined $\gamma = [\gamma_0 \dots \gamma_n]^{\text{T}}$ and chosen such that $\gamma_i \in [0,1]$. The value $1-\gamma_i$ can be thought of as the percentage of the full update applied to state i. For example, $\gamma_i = 0$ implies the full EKF update is applied to state i while $\gamma_i = 1$ implies that state is simply *considered*. Anything in between would result in a partial update of the state. Generally less observable, slowly time-varying states should receive a lower percentage of the full update, while more observable states with higher process noise or uncertainty growth rates would receive larger (or full) updates.

The partial-update approach was shown to increase filter robustness to large uncertainties in camera to IMU calibration example in Brink.[40] In our filter, the partial update is applied on the acceleration bias β_1, body attitude q_n^b, and camera to IMU rotation q_b^c states. The results were obtained with γ values of 0.9, 0.9, and 0.97, respectively (and 0 for all other filter states), implying that 10%, 10%, and 3% (and 100%) of the nominal updates were applied to the respective states at each measurement update step.