A Multiplicative Extended Kalman Filter for Relative Rotorcraft Navigation

Robert C. Leishman
Timothy W. McLain
mclain@byu.edu

Follow this and additional works at: https://scholarsarchive.byu.edu/facpub

Part of the Mechanical Engineering Commons

Original Publication Citation

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/facpub/1299

This Report is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.
A Multiplicative Extended Kalman Filter for Relative Rotorcraft Navigation

Robert C. Leishman, Student Member, IEEE, Timothy W. McLain, Senior Member, IEEE.

Abstract—In this article we detail the fundamentals of a new approach to GPS-denied navigation for aerial vehicles in confined indoor environments. We depart from the common practice of navigating within a globally referenced map, and instead keep the position and yaw states relative to the current node in the map. The approach combines elements of graph SLAM with a multiplicative extended Kalman filter (MEKF). The filter provides quality state estimates at a fast rate and a graph SLAM algorithm maintains a pose graph. We provide specific details for the relative MEKF. We verify the relative estimation approach with hardware flight test results accompanied by comparisons to motion capture truth. We also provide flight results with estimates in the control loop.

Index Terms—Multiplicative EKF, Sensor Fusion, GPS-Denied Navigation, Quaternions, Quadrotor, Indoor Flight.

I. INTRODUCTION

Sensor fusion and localization are important aspects of navigation in unknown, GPS-denied environments, where global measurements and a priori map information are unavailable. Aerial flight using small aircraft like quadrotors is an additional challenge because of limited payload capacity for onboard sensors and computational resources, and the need to control the vehicle’s fast dynamics using the state estimates. Unlike ground robots, quadrotors cannot afford to pause in one place until complex algorithms finish processing and state estimates converge for navigation to continue. Accurate and fast state estimates are critical to maintain control of the vehicle and provide quality sensor information.

To date, successful implementations providing GPS-denied autonomous quadrotor flight, like those outlined in [1], [2], [3], [4], [5], [6], [7], [8], typically follow a general approach for localization and mapping. This approach, outlined in [1], enabled the first fully-autonomous quadrotor with the ability to fly using only onboard sensors in a priori unknown environments. Using this original approach the autonomous vehicle must: estimate motion using exteroceptive sensors, fuse motion estimates with other sensor information to form global state estimates, and employ some type of simultaneous localization and mapping (SLAM) to provide loop closure and global consistency.

Incremental motion of the vehicle is computed with an exteroceptive sensor and a motion estimation algorithm. The original approach for GPS-denied autonomous quadrotor flight utilized a planar laser scanner with a fast scan-matching algorithm [1]. A similar method is used in [3]. Planar laser-scanner implementations, however, require strict assumptions regarding the nature of the environment. Six-degree-of-freedom (6DoF) motion estimation using machine vision, as used in the approaches in [2], [4], [7], [9], is advantageous due to a camera sensor’s low cost, low power requirements, and light weight. Additionally, high-quality solutions can be obtained using fewer assumptions about the environment. Vision is, however, limited due to requirements for appropriate lighting and a sufficient number of features in the environment. Visual implementations usually employ a version of visual odometry (VO) [10], although a modified version of parallel tracking and mapping (PTAM) [11] is used in [6].

We note that the exteroceptive motion estimates are relative measurements between images or scans. Various approaches are employed to convert the relative measurements into the requisite global form. One method is to sum all previous motion estimates and form a pseudo-global measurement [1]. Another is to treat the measurement as an average velocity over the duration between sensor messages [4], [7]. Stochastic cloning [8], [9], [12] is a more valid option as the state is cloned to more appropriately handle the cross-correlations that are necessary to form a correct global measurement update.

Sensor fusion combines exteroceptive motion measurements with IMU information for improved global state estimates at the fast rate needed for feedback control of the quadrotor. Typically some form of an extended Kalman filter (EKF) is employed and the accelerometer and gyroscope measurements are used in the filter propagation step [1], [3], [4]. In contrast, the approach detailed in [7] uses a simple low-pass filter for sensor fusion. One difficulty noted in many of these approaches is the lack of a good velocity measurement to aid in the sensor fusion [13], [1], [3], [7]. This issue is discussed in [14] and an improved quadrotor dynamic model is shown to provide body-fixed velocity information using only accelerometer measurements.

Global state estimates of the vehicle will drift if only relative motion and IMU measurements are used. SLAM is a solution that provides the capability to maintain a map and to perform loop closure to eliminate drift and obtain globally consistent state estimates. The use of the pose graph and nonlinear optimization tools from the graph-SLAM paradigm [15], [16] and visual place recognition to find possible loop closures [17] is increasingly common. Graph SLAM provides flexibility in working with the sensor fusion and ease with which place recognition loop closures may be applied to the map. Visual place recognition algorithms recognize previously visited locations in the map using only visual information. Because of the need for globally metric information, a majority of the approaches require feedback from the SLAM algorithms as a measurement in the sensor fusion.

One problem with the general approach, which was noted in [5], is the requirement of globally metric information for the maps of the environment and states of the vehicle. Requiring global states incurs difficulties such as the need for additional states to incorporate relative position measurements [9], [8], waiting periods for global consistency [2], dependence on computationally-expensive optimization and loop closure feedback for localization [4], [7] and logic to accommodate large jumps in pose when loop closures are applied [8].

In [18], [19], the authors propose that a vehicle should navigate using a relative formulation of the vehicle state, rather than a global one. Similar to the general approach, they use a combination of graph SLAM and an EKF to provide mapping and sensor fusion. The map is a pose graph, with images from the onboard camera as key components of the nodes. The EKF provides estimates at the high rate required for feedback control of the vehicle. The difference is that the position and yaw states of the EKF are defined with respect to the current node in the map, rather than to a global origin. Relative state information affords many advantages, such as the ability to directly utilize relative exteroceptive measurements, elimination of required feedback to the filter from computationally-expensive SLAM algorithms, easy creation of map edges using the filter state and covariance, and flexible use of global information.

The main contribution of this article is the development of a multiplicative extended Kalman filter (MEKF) that uses an improved rotorcraft model [14] to provide relative, rather than global, state estimates. Another contribution is a more detailed development of the relative navigation approach originally described in [19] and the
Relative filtering approach described in [18]. Additionally, we verify the results of the relative estimation approach with hardware flight tests accompanied by comparisons to motion capture truth data. We also provide flight results with estimates in the control loop.

We utilize a hexacopter vehicle from MikroKopter, pictured in Figure 1, as the platform for experiments. An IMU, sonar altimeter, and front-facing RGB-D camera are the only sensors used for the sensor fusion and mapping. All computation is completed onboard using a small computer to perform the necessary processing.

![Mikrokopter hexacopter](image)

**Fig. 1.** The Mikrokopter hexacopter that we use to carry out the experiments. The only sensors we utilize are an altimeter, IMU, and front-facing RGB-D camera.

The remainder of the article is outlined as follows. We explain the approach to relative navigation in Section II. The model and MEKF are derived in Section III. The results are described in Section IV. And we summarize the work in Section V.

## II. RELATIVE NAVIGATION APPROACH

Relative navigation refers to navigation with respect to a local reference frame. We propose that the local frame change as the vehicle moves through the environment, establishing a topological representation of the world using a pose graph [20]. The changes in the local frame occur based on the needs of a robust VO algorithm, described in [21]. The algorithm is keyframe based. Instead of comparing consecutive images, each current image is compared to a reference image, called a keyframe, to obtain the 6DoF change in pose. New keyframes are declared when the vehicle has moved further than a predetermined threshold from the previous keyframe and the overlap between images becomes too small for reliable matching. The local coordinate frames, with respect to which the vehicle navigates, are derived from the keyframes.

The map in Figure 2 illustrates the relative topological approach. The VO algorithm initializes a keyframe at node 1 and an edge is added between the global frame and the node frame once this information is known. The filter estimates the position and yaw states of the vehicle with respect to the local coordinate frame at node 1 as the vehicle travels. When the VO requires a new keyframe to maintain good performance, a new keyframe and node are declared at pose 2. An edge is added to the map using the relative states and covariance in the MEKF. The navigation then continues with respect to node 2 by marginalizing out the old relative states and augmenting the state vector with new ones. This process continues as the vehicle moves through the environment, with new keyframes and nodes being declared as necessary and the MEKF changing the relative states each time a new keyframe is declared. As current images are compared to a keyframe, the position estimates will not drift as long as the keyframe does not change. A vector chain of edges connects the hexacopter to the global reference frame. Global position and yaw for the vehicle can be estimated by summing all the edges and the current state.

![Relative navigation](image)

**Fig. 2.** Relative navigation using nodes and edges. As the vehicle flies through the environment, nodes are created using the VO keyframes and edges are defined between them using the relative states of the MEKF. The vehicle state is relative to the current node (node 4). The global state of the vehicle can be found by summing the edges and the current state.

This relative navigation approach has several key advantages: direct use of sensor information for state updates, straightforward creation of map edges using the filter state and covariance, and flexible use of global information. We discuss below how each of these is helpful for autonomous aerial navigation.

Exteroceptive sensors provide relative information. In particular, the VO provides the change in 6DoF pose between the current and keyframe images. By expressing the VO result in the node coordinate frame, the position and attitude are updated directly in the filter. This simplification eliminates the need for additional states in the filter and does not require VO measurements to update the velocity states.

Defining edges between consecutive nodes is a simple matter of saving the relative portions of the state and covariance just before a new node is created. The covariances from each edge could be combined and then used for a confidence measure of the current global position. For example, a path planning algorithm might use the combined covariances of the edges to indicate when estimates have drifted sufficiently to warrant a planned loop closure.

Our proposed relative approach offers more flexibility than a globally-based method. A front-end subsystem provides the critical processes to keep the hexacopter flying, including the VO, sensor fusion, control, and obstacle avoidance. All the components in the front end are based in the relative coordinate frame explained above. A back-end subsystem will maintain globally consistent information, including a global map and high-level, global objectives when desired. The only information provided by the back end or by human operators are goal locations expressed in the current node reference frame. This is in contrast to other solutions that have been developed that require feedback from the computationally expensive visual place recognition and optimization components. Additional details on the architecture of the approach are found in [19].

This separation between the time-critical front end and globally consistent back end provides the flexibility mentioned. The system can fly reliably either with or without loop-closure constraints that constrain drift, and with or without global optimization. This is possible because the local navigation and control take place regardless of global changes within the map. Without loop closure it is clear that the map will drift and not remain globally consistent. The relative relationships between nodes, however, maintain locally consistent topological and metric relationships between saved locations. Therefore, the map could be traversed, even back to the starting location, by using these relative relationships. This is also true when loop closure...
A. Node Frame

The nodes in the pose graph contain keyframe images and they represent locations in the environment where those images are taken. Each node has a local coordinate frame defined, with respect to which the vehicle navigates through the environment. The local frame is formed using the global $\vec{n} \cdot \vec{e}$ plane and the body-fixed coordinate frame at the instant in time when the keyframe image is taken. This particular body frame is designated the reference body-fixed coordinate frame.

The node frame for the $j$-th node is defined by unit vectors in the front, right, and down directions: $\vec{f}_j$, $\vec{r}_j$, and $\vec{d}_j$. The $\vec{f}_j$ direction is parallel to the global $\vec{f}$ direction. The $\vec{f}_j$ direction is defined by the projection of the reference body $\vec{b}$ axis onto a plane parallel to the global $\vec{n} \cdot \vec{e}$ plane, as shown in Figure 3. The $\vec{r}_j$ direction is defined to make a right-hand coordinate system. This representation preserves the heading of the vehicle, which corresponds with the direction the vehicle was taken, and maintains a global down direction, which keeps the roll and pitch defined correctly.

The relationship between the keyframe and the local node frame is necessary for correctly expressing VO measurement updates. Points defined in the keyframe’s camera coordinate frame are expressed in the node coordinate frame using two rotations, $R^b_k$ and $R^e_k$, and a translation, $p^b$. The rotation $R^b_k$ and translation $p^b$ define the static transformation between the camera frame and the body-fixed frame that is obtained through calibration. $R^b_k$ rotates points from the camera frame into the body-fixed frame and $p^b$ are the coordinates of the camera focal point in the body-fixed coordinate frame. These values only change when the location or orientation of the camera or IMU changes on the platform.

The second rotation, $R^e_{ki}$, defines the relationship between the reference body-fixed frame and the node frame. The reference body-fixed frame is created by saving the roll and pitch angles of the vehicle, $\phi_j$ and $\theta_j$, at the instant in time the image is taken. The matrix $R^b_{ki} (\phi_j, \theta_j)$ rotates points from the body-fixed frame into the node $j$ coordinate frame. Combined together, the transformation $R^e_k (R^b_k p^{cr} + p^b)$ transforms points from the keyframe coordinate frame to the node coordinate frame.

B. Map

The map used in this work is a collection of nodes and edges in a relative topological pose graph. New nodes are created with each new keyframe. Edges are added between temporally and spatially consecutive keyframes.

A node is described, ultimately, by a keyframe RGB and depth image pair. Attached to the keyframe pair are the estimates of relative and global position and orientation, yet the image encodes the true instantaneous location of the vehicle. The node coordinate frame, defined above, enables navigation relative to the node.

Edges in the graph represent the estimated relative transformations between nodes. We currently only consider edges from the odometry, but we plan to soon include other constraints, such as those from visual recognition loop closures and intermittent GPS measurements. The odometry edges are created using the MEKF, based on the measurements from the VO algorithm. When a new node is received by the estimator from the VO, the old relative portions of the state and covariance are marginalized out and saved as the edge between the old and new nodes.

III. Multiplicative Extended Kalman Filter

In contrast to the work in [18], we now use unit-length quaternions to represent the attitude of the vehicle. A quaternion is parameterized by a scalar and a vector. We utilize the standard Hamiltonian form of the quaternion [23]. A unit-length quaternion represents an over-parameterization of the attitude and should not be used directly in an EKF framework because of singularity problems that can occur in the covariance matrix. Many generic estimation methods have been developed to handle this difficulty and are available and commonly used in practice [24]. We chose to derive a MEKF [25], [26] for our relative hexacopter system.

The MEKF is an indirect EKF, which means that the error in the state $\Delta x$ and the covariance of the error are maintained in the filter rather than the best estimate $\hat{x}$ and error covariance. The name Multiplicative was coined as the error in the quaternions is computed using a quaternion product $\otimes$ instead of subtraction (i.e., $\delta q(\theta) = q \otimes q^{-1}$) [24]. The three-dimensional error vector $\theta$, obtained from the error quaternion using a small angle approximation, is maintained in the error state. This results in a minimal representation that is safe to use in calculating the covariance. We also use a continuous-discrete representation [27] for our implementation of the MEKF.

In this implementation we do not directly maintain an estimate of the error state in the filter at each timestep and therefore do not maintain a true indirect filter. Instead, we maintain the best estimate $\hat{x}$ and the covariance $P$ of the error state $\Delta x$ in the filter. Details of the implementation are given below.

A. State Dynamics

We model the hexacopter with the nonlinear equations
\begin{align}
\dot{x} &= f(x, u), \\
y_i &= h_i(x, u), \quad 1 \leq i \leq p,
\end{align}
where $h_i$ is the $i$-th measurement function, and the vector $u$ represents the inputs that drive the evolution of the estimated states. The inputs to the model are simply the gyroscope measurements and the $b_z$ accelerometer
\begin{equation}
u = \begin{bmatrix} p_{gyro} & q_{gyro} & r_{gyro} & z_{accel} \end{bmatrix}^T.
\end{equation}
The true states \( x \) of the multicopter are defined with respect to the current node, node \( j \),
\[
    x = [p_n^T \ q_n^T \ v_n^T \ \beta^T \ \alpha^T \ q_n^T \ p_n^T]^T. \tag{4}
\]
The relative position vector \( p_n \), made up of \( f_j, r_j \) and \( d_j \), is the displacement of the body with respect to node \( j \). The quaternion \( q_n \), with components \( q_x, q_y, q_z \) and \( q_w \), expresses the attitude of the body frame with respect to the node frame. The component \( q_z \) is relative to the current node. \( v_n \) is the body frame velocity vector, composed of \( u, v \) and \( w \). The gyroscope bias vector is \( \beta \). We only estimate the accelerometer biases in the body \( b_0 \) and \( b_1 \) directions in \( \alpha \). The last two parameters in (4) represent the transformation from the body coordinate frame to the camera coordinate frame and can be optionally included in the state, \( q_n \) is the quaternion form of the frame transformation \( R_{bn} \), which rotates points from the camera coordinate frame into the body coordinate frame. \( q_n \) are the coordinates of the camera focal point expressed in the body coordinate frame. Once refinements to the calibration parameters are obtained, these estimates can be saved as constants and then removed from the state vector.

The estimated state vector is denoted \( \hat{x} \).

Error for all the components of the state, except the quaternions, are defined as
\[
    \delta a = a - \hat{a}. \tag{12}
\]

Error for a quaternion is defined by
\[
    \delta q = q \otimes q^{-1}. \tag{13}
\]

The attitude error vector is maintained in the error state and, like the other error components, is assumed to be small. The length of the error state is reduced by one for each quaternion when compared to the length of the true state. The 20-element relative error state is then
\[
    \Delta x = \left[ \delta p_n^T \ \delta \theta_n^T \ \delta v_n^T \ \delta \beta^T \ \delta \alpha^T \ \delta \delta \theta_n^T \ \delta \delta p_n^T \right]^T. \tag{15}
\]

The error dynamics, computed using (12), (13), and (5) through (11) are as follows. In the derivation we assumed that second-order effects are negligible. We also used the identities \( R(\delta q) \approx I_{3 \times 3} - [\delta \theta] \), where \( [\cdot] \) is the skew-symmetric matrix operator on a vector, and \( [y] x = -[x] y \) [25].

\[
    \delta \dot{p}_n = R(q_n^b) \delta v - R(q_n^b) (q_n^b) \delta \beta^b, \tag{16}
\]

\[
    \delta \dot{\theta}_n = -u_{(1:3)} \times \delta \theta_n - \beta - \eta_\omega, \tag{17}
\]

\[
    \delta \hat{v} = \delta v \times (u_{(1:3)} - \beta - \eta_\omega) + \hat{v} \times (-\delta \beta - \eta_\omega) + [R(q_n^b) \delta \eta_\omega] - \frac{1}{m} M \delta v \tag{18}
\]

\[
    \hat{\beta} = \eta_\beta \tag{19}
\]

\[
    \dot{\alpha} = \eta_\alpha \tag{20}
\]

\[
    \dot{q}_n^b = 0 \tag{21}
\]

\[
    \dot{p}_n^b = 0. \tag{22}
\]

Equations (16) through (22) are linearized and result in the following linear model
\[
    \Delta x = A \Delta x + B u, \tag{23}
\]
where \( A \) is the Jacobian of (16) through (22) with respect to the error state \( \Delta x \) and \( B \) is the Jacobian of (16) through (22) with respect to the input \( u \).

C. Prediction

As was mentioned above, in the filter we maintain the estimated state \( \hat{x} \) and the covariance \( \hat{P} \) of the error state (15). The estimated states are propagated forward by numerically integrating the nonlinear equations of motion, (5) through (11). The estimated covariance is propagated forward by numerically integrating the equation
\[
    \dot{\hat{P}} = A \hat{P} + \hat{P} A^\top + \gamma (B G B^\top + Q), \tag{24}
\]
where the matrices \( A \) and \( B \) are from (23), \( \gamma \) is a tuning parameter, \( G \) is a diagonal matrix of the measured covariance on the inputs (3) and \( Q \) is the process noise covariance which represents modeling error and disturbances.

In contrast to [18], some diagonal portions of \( Q \) are non-zero and must be found through hand tuning for the best performance. As we mentioned above, second-order terms were neglected in the
derivations of (16) through (18). This results in not being able to completely account for the error.

**D. Measurement Updates**

We update the filter with measurements of the form in (2). Specifically, altimeter, accelerometer, and VO position and orientation measurements are used. In this implementation we treat the VO measurements updates separately. This is possible because we account for the contribution of the rotational uncertainty in the position covariance when the covariances of the measurements are generated [21].

Each measurement update follows the same procedure. For generality, we will assume that the measurement is \( h \), the measurement from the model is \( \hat{h} \) and the Jacobian of the analytic residual \( \Delta h \) with respect to the error state is \( H \).

First we compute the residual \( \Delta h \)

\[
\Delta h = h - \hat{h}. \tag{25}
\]

The covariance of the residual \( S \) is computed using

\[
S = R_h + HPH^T, \tag{26}
\]

where \( R_h \) is the covariance on the measurement model \( \hat{h} \). The Kalman gain \( L \) is

\[
L = PH^T S^{-1}. \tag{27}
\]

The correction (or updated error state) \( \Delta \hat{x}^+ \) is computed as

\[
\Delta \hat{x}^+ = L \Delta h, \tag{28}
\]

where the \( ^+ \) notation denotes an updated variable. The covariance is updated using

\[
\hat{P}^+ = (I - LH) \hat{P}. \tag{29}
\]

After each measurement update, we need to use the correction (28) to update the current state estimate \( \hat{x} \). Using (12), we can update a component \( a \) of the state, which is not a quaternion, as

\[
a^+ = a + \delta a^+. \tag{30}
\]

To update the quaternions in the state, we create an error quaternion \( \delta q \) from the quaternion error vector \( \delta \theta \) and then use the quaternion product to compute the updated quaternion. First, we change the error vector into the vector portion of the error quaternion using

\[
\delta q_{vec}^+ = \frac{1}{2} \delta \theta^+. \tag{25}
\]

Next, we compute the unit-length error quaternion

\[
\delta q^+ = \sqrt{\frac{\delta q_{vec}^+}{1 - \delta q_{vec}^+} \delta q_{vec}}; \tag{27}
\]

or, if \( \left( \delta q_{vec}^+ \delta q_{vec} \right) > 1 \), using

\[
\delta q^+ = \frac{1}{\sqrt{1 + \delta q_{vec}^+ \delta q_{vec}}} \delta q_{vec}. \tag{28}
\]

Finally we can compute the new quaternion for the state estimate using the error quaternion \( \delta q \) and the estimate from the prediction step \( \hat{q} \)

\[
\hat{q}^+ = \delta q^+ \otimes \hat{q}. \tag{31}
\]

The measurement models for each sensor are outlined below. Each of these is used in the generalized method outlined in (25) through (31) to complete the measurement updates for the filter.

1) **Altimeter Measurement Model:** The altimeter provides a global measurement of the altitude of the vehicle, assuming flight near hover and a flat floor. We can obtain an estimate of the global altitude using the position in the current state \( \hat{p} \) and the global position \( \hat{p}_{node} \) of the current node with respect to which we are navigating. No rotational transformation is necessary as the global down and node down directions are parallel. To compute the Jacobian \( H_{alt} \) of the residual with respect to the error state (15), we must develop an analytical expression for the residual \( \Delta h = h_{alt} - \hat{h}_{alt} \). We have

\[
\begin{align*}
\hat{h}_{alt} &= p(3) + \hat{p}_{node}(3) + \eta_{alt} \\
\hat{h}_{alt} &= p(3) + \hat{p}_{node}(3),
\end{align*}
\]

for which the analytic residual is

\[
\begin{align*}
\Delta h_{alt} &= h_{alt} - \hat{h}_{alt} \\
\Delta h_{alt} &= p(3) + p_{node}(3) - (\hat{p}(3) + \hat{p}_{node}(3)) \\
\Delta h_{alt} &= \delta p(3) + \delta p_{node}(3) + \eta_{alt}. \tag{32}
\end{align*}
\]

The Jacobian of the residual is trivially

\[
H_{alt} = \begin{bmatrix} 0 & 0 & 1 & 0_{1 \times 3} & 0_{1 \times 3} & 0_{1 \times 2} & 0_{1 \times 3} & 0_{1 \times 3} \end{bmatrix}. \tag{33}
\]

The covariance of the measurement is \( E [\eta_{alt} \eta_{alt}^T] = R_{alt} \).

2) **Accelerometer Measurement Model:** The body \( \dot{b}_{x} \) and \( \dot{b}_{y} \) axis measurements are modeled as

\[
\begin{align*}
\hat{h}_{acc} &= -\frac{1}{m} M \delta v^b \alpha + \eta_{acc} \\
\hat{h}_{acc} &= -\frac{1}{m} M \delta v^b + \dot{\alpha}. \tag{34}
\end{align*}
\]

We can then calculate the Jacobian \( H_{acc} \):

\[
H_{acc} = \begin{bmatrix} 0_{3 \times 3} & -\frac{1}{m} M & 0_{3 \times 3} & 0_{3 \times 3} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \end{bmatrix}. \tag{35}
\]

3) **Visual Odometry Position Model:** Figure illustrates the chain of transformations involved in the measurement updates for the VO. The current pose in the state is represented by the transformation from the current node frame to the current body frame, \( q_n^b \) and \( p^n \). The calibration terms, \( q^c_n \) and \( p^c \), are found in two places since there is a reference body frame in between the node frame and the reference camera frame. The VO provides the transformation made up of \( q^c_{br} \) and \( p^c \).

Using Figure 4, we estimate the model of the VO translation \( p^c \) as

\[
\begin{align*}
\hat{h}_{vp} &= -L_b^c \left( q^c_n \right)^T p^b - L_n^c \left( q^b_n, q^c_n \right)^T p^n + L_{br}^c \left( q^m_n, q^c_n \right) p^m + \eta_{vp} \\
\hat{h}_{vp} &= -L_b^c \left( q^c_n \right)^T p^b - L_n^c \left( q^b_n, q^c_n \right)^T p^n + L_{br}^c \left( q^m_n, q^c_n \right) p^m + \eta_{vp}.
\end{align*}
\]

The analytic residual begins as

\[
\begin{align*}
\Delta h_{vp} &= -L_b^c \left( q^c_n \right)^T p^b - L_n^c \left( q^b_n, q^c_n \right)^T p^n + L_{br}^c \left( q^m_n, q^c_n \right) p^m + \eta_{vp} - \\
&\left( -L_b^c \left( q^c_n \right)^T p^b - L_n^c \left( q^b_n, q^c_n \right)^T p^n + L_{br}^c \left( q^m_n, q^c_n \right) p^m \right).\tag{36}
\end{align*}
\]
We can switch to a rotation matrix form and replace the truth terms with the estimate and error using (12). Note that the order of the rotation terms must switch when going from quaternions to rotation matrices as we are using the Hamilton representation of the quaternion.

\[
\Delta h_{vp} = -\mathcal{R}^{T}(\delta q^b_c)\mathcal{R}^{T}(q^c_b)\left(\delta p^b + \delta p^c\right) - \mathcal{R}^{T}(\delta q^b_c)\mathcal{R}^{T}(q^c_b)\mathcal{R}^{T}(\delta q^b_c)\left(\delta p^b + \delta p^c\right) + \mathcal{R}^{T}(\delta q^b_c)\mathcal{R}^{T}(q^c_b)\left(\delta p^b + \delta p^c\right) + \eta_{vp} + \mathcal{R}^{T}(q^c_b)\mathcal{R}(q^b_c)\mathcal{p}^b - \mathcal{R}^{T}(q^c_b)\mathcal{R}(q^b_c)\mathcal{p}^b - \mathcal{R}^{T}(q^c_b)\mathcal{R}(q^b_c)\left(\delta p^b + \delta p^c\right) + \eta_{vp}.
\]

Finally, after expanding, removing second-order terms, and simplifying using the approximation \(\mathcal{R}(\delta q^b_c) \approx (I - [\delta \theta^b_c])\) or \(\mathcal{R}^{T}(\delta q^b_c) \approx (I + [\delta \theta^b_c])\), we have

\[
\Delta h_{vp} = -\mathcal{R}^{T}(q^c_b)\delta p^b - [\delta \theta^b_c] \mathcal{R}(q^b_c)\mathcal{p}^b - \mathcal{R}^{T}(q^c_b)\mathcal{R}(q^b_c)\delta p^b + [\delta \theta^b_c] \mathcal{R}(q^b_c)\mathcal{p}^b.
\]

The Jacobian \(H_{vp}\) of the analytic residual is

\[
H_{vp} = \begin{bmatrix}
-\mathcal{R}^{T}(q^c_b)\mathcal{R}(q^b_c)\mathcal{p}^b & [\mathcal{R}^{T}(q^c_b)\mathcal{R}(q^b_c)]\p^b \mathcal{R}(q^b_c)\mathcal{R}(q^b_c)\mathcal{p}^b \\
0 & \mathcal{R}(q^b_c)\mathcal{R}(q^b_c)\mathcal{p}^b \\
0 & \mathcal{R}(q^b_c)\mathcal{R}(q^b_c)\mathcal{p}^b \\
\end{bmatrix}.
\]

4) Visual Odometry Orientation Model: Using Figure 4 we can also find a model for the view-matching rotation \(q^c_b\):

\[
h_{eq} = q^c_e + \eta_{eq} = q^c_b \otimes q^c_e \otimes q^b_e^{-1} + \eta_{eq},
\]

\[
h_{eq} = q^c_e \otimes q^c_e \otimes q^b_e \otimes q^b_e^{-1}.
\]

The analytic residual can be computed two ways, using subtraction [25] or multiplication [29]. We have found the methods to be roughly equivalent in practice, but we currently use the subtraction method.

The analytic model of the residual is

\[
\Delta h_{eq} = q^c_e + \eta_{eq} - q^c_e.
\]

If we multiply each side by the matrix \(\Xi^T(q^c_e)\), where

\[
\Xi^T(q^c_e) = \Xi^T(q^c_e) = [q^c_eI + [q^c_e, \cdot] - q^c_e],
\]

we can eliminate \(q^c_e\) (note that for any quaternion \(q^c_e\), \(\Xi^T(q^c_e)q^b_e = 0\)).

\[
\Delta h_{eq} = \Xi^T(q^c_e)\Delta h_{eq} = \Xi^T(q^c_e)(q^c_e + \eta_{eq}) = 0. \tag{40}
\]

Which, expanded out, is

\[
\Delta h_{eq} = \Xi^T(q^c_e) \left(\delta q^b_c \otimes q^b_c \otimes q^b_e \otimes \delta q^b_c \otimes q^b_e \otimes q^b_c^{-1} \otimes \delta q^b_c^{-1}\right).
\]

The analytic Jacobian of (41) is too large to be used in practice. We were able to calculate an approximate Jacobian at nominal values of the quaternions and use that with success. Note that using this method, we calculate the residual in the algorithm using (40) instead of (25), where \(q^c_e\) is computed using (39) and \(q^c_e\) is the measurement from the VO. Additionally, the covariance \(R_{eq}\) must be modified by \(\Xi^T(q^c_e)\), so that

\[
R_{eq} = \Xi^T(q^c_e)R_{eq}\Xi(q^c_e).
\]

5) Delayed View-Matching Updates: An additional challenge with the view-matching measurement updates is that they are delayed. The stochastic delay is due to the requisite image-processing time. Consequently, computing the position and orientation measurement updates requires a few additional steps. First, the state and covariance must first be restored to their values at the time the image was taken. Then the measurement updates can be applied in the normal fashion, (25) through (31). Finally, the state and covariance must be repropagated back to the current time by re-applying the prediction and measurement updates using the gyroscope, altimeter and accelerometer measurements at their respective timesteps. The state, covariance, IMU, and altimeter information are saved at each timestep to accommodate this requirement.

E. Augment and Marginalize the Relative State

When a new node is created by the view-matching algorithm, the relative portions of the state and covariance must change. This is slightly more difficult as we now use a quaternion to encode the orientation and the yaw is relative. However, the process is straightforward. We use the relationship between quaternions and Euler angles to adjust the state and covariance appropriately.

We can augment and marginalize simultaneously as we are not augmenting additional states or removing unnecessary ones, but are simply replacing the old relative states with new relative states. The states that change are the positions in \(\p^b\) and the yaw contained in \(q^b_c\). The positions are simply replaced with zeros. The quaternion in the state must maintain the same pitch and roll but the yaw must be zeroed out. This is done using the transformations to and from Euler angles. We use the current quaternion \(q^b_c\) to find the current pitch \(\hat{\phi}\) and roll \(\hat{\theta}\) angles.

\[
\hat{\phi} = \arctan\left(\frac{2(\hat{q}_{1w}\hat{q}_{2} + \hat{q}_{1}\hat{q}_{2})}{q_{2}^{2} + q_{2}^{2} - q_{2}^{2} - q_{2}^{2}}\right),
\]

\[
\hat{\theta} = \arcsin(2(q_{1w}\hat{q}_{2} - \hat{q}_{1}\hat{q}_{2})),
\]
and then we initialize the new quaternion using

\[
\delta q^+_x = \cos \left( \frac{\delta \theta}{2} \right) \sin \left( \frac{\delta \phi}{2} \right),
\]

\[
\delta q^+_y = \sin \left( \frac{\delta \theta}{2} \right) \cos \left( \frac{\delta \phi}{2} \right),
\]

\[
\delta q^+_z = -\sin \left( \frac{\delta \theta}{2} \right) \sin \left( \frac{\delta \phi}{2} \right),
\]

\[
\delta q^+_w = \cos \left( \frac{\delta \theta}{2} \right) \cos \left( \frac{\delta \phi}{2} \right).
\]

Initializing the covariance for the quaternion is slightly different, since the covariance of the quaternion is actually the covariance of the attitude error vector.

The error in yaw must be zeroed-out, but the error in roll and pitch must remain unchanged. Thus, we can use (42)-(47) to accomplish this, but we replace the quaternion \( \delta q^k \) with the error quaternion \( \delta q^\epsilon \).

Recall that the error quaternion is related to the attitude error vector through (14).

The equations for the new error quaternion given the old error quaternion are obtained by substituting (42) and (43) into (44)-(47).

We can simplify the equations by canceling out all the second-order terms, as the elements of \( \delta \theta \) are small:

\[
\delta q^+_x \approx \cos \left( \frac{\delta \theta}{2} \right) \sin \left( \frac{\delta \phi}{2} \right),
\]

\[
\delta q^+_y \approx \sin \left( \frac{\delta \theta}{2} \right) \cos \left( \frac{\delta \phi}{2} \right),
\]

\[
\delta q^+_z \approx -\sin \left( \frac{\delta \theta}{2} \right) \sin \left( \frac{\delta \phi}{2} \right).
\]

We need the Jacobian \( H_{\delta \theta} \) of the above three equations with respect to \( \delta \theta_x \), \( \delta \theta_y \) and \( \delta \theta_z \):

\[
H_{\delta \theta} = \begin{bmatrix}
\frac{\partial (\delta q^+_x)}{\partial (\delta \theta_x)} & \frac{\partial (\delta q^+_y)}{\partial (\delta \theta_x)} & \frac{\partial (\delta q^+_z)}{\partial (\delta \theta_x)} \\
\frac{\partial (\delta q^+_x)}{\partial (\delta \theta_y)} & \frac{\partial (\delta q^+_y)}{\partial (\delta \theta_y)} & \frac{\partial (\delta q^+_z)}{\partial (\delta \theta_y)} \\
\frac{\partial (\delta q^+_x)}{\partial (\delta \theta_z)} & \frac{\partial (\delta q^+_y)}{\partial (\delta \theta_z)} & \frac{\partial (\delta q^+_z)}{\partial (\delta \theta_z)}
\end{bmatrix}.
\]

We then modify the covariance \( \dot{P} \) using the overall Jacobian of the change in the error state

\[
H_A = \begin{bmatrix} 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 14} \end{bmatrix},
\]

and the following matrices

\[
C = H_A P \dot{H}_A^T,
\]

\[
T = H_A P,
\]

\[
L = PH_A^T.
\]

The matrix \( T : 6 \times 20 \) is used to overwrite the top six rows of the covariance. The matrix \( L : 20 \times 6 \) overwrites the left-most six columns of the covariance, and the matrix \( C : 6 \times 6 \) overwrites the top left corner, in that order. The remaining portions of the covariance are left the same as they are not relative.

IV. EXPERIMENTAL SETUP AND RESULTS

The results for three scenarios are presented. The first are of the relative MEKF running in real time during a manually-controlled flight. The second set are from an autonomous flight with the estimates in the control loop during a commanded hover. The third set of results are from an autonomous flight in which the hexacopter is following planned paths through an environment. All the flights were completed in a motion capture environment. The motion capture data is only used as the truth for the comparison of the results and to initialize the starting location of the vehicle for easier comparison between truth and estimates.

We utilize a MikroKopter hexacopter vehicle that carries a Intel Core i7 processor computer to do all the computations, as shown in Figure 1. The computer is running Ubuntu 12.04 Linux and all the applications are implemented in C++ and connected using the robot operating system (ROS) [30]. RGB-D imagery for the VO is provided by an ASUS Xtion Pro Live at 15 Hz. We also use a MicroStrain 3DM-GX3-15 IMU and a LV-MaxSonar-EZ3 sonar altimeter. Data is received from these sensors at 100 Hz and 40 Hz respectively.

A. Real-Time Results

Figures 5 through 8 illustrate the performance of the relative MEKF estimator compared to truth data from the motion capture system during a manually-controlled flight. Velocity truth was generated using a time difference of the 100 Hz motion capture position information, expressed in the body-fixed coordinate frame. The estimates were computed in real time, onboard the aircraft during the manual flight.
Fig. 6. Body-fixed frame side velocity $v$ truth and estimate comparison. Notice that there are no discontinuities, as the body-frame velocity is not relative. Results for the front and down body-fixed velocities are similar.

Fig. 7. The $x$ component of the quaternion $q^b_n$ truth and estimate comparison. There are not any discontinuities in this figure, as this portion of the quaternion is not relative. Results for the other portions of the quaternion are similar.

Fig. 8. A comparison between truth and estimates for the global value for east. This estimate is obtained by a vector sum of the edges and the current state at each time-step. This information is not required for the algorithms, but it is helpful in quantifying the quality of the estimates. Results for global north and down position are similar.

compared to truth data. We do have the unavoidable drift in yaw as we do not measure a global yaw angle. This could be corrected with the use of loop closures.

The composition of the relative states and the vector chain of edges to the origin provide the global estimate shown in Figure 8. At this point, no loop closure and optimization have been applied to these results. These are dead-reckoning results using a MEMs-grade IMU and odometry, through the VO. The estimates trend well with the truth information. The global information is not required in the filter and these results are provided for information only.

<table>
<thead>
<tr>
<th>State</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>front relative position ($f$)</td>
<td>0.044 (m)</td>
</tr>
<tr>
<td>right relative position ($r$)</td>
<td>0.048 (m)</td>
</tr>
<tr>
<td>down relative position ($d$)</td>
<td>0.029 (m)</td>
</tr>
<tr>
<td>x quaternion ($q_x$)</td>
<td>0.008 (rad)</td>
</tr>
<tr>
<td>y quaternion ($q_y$)</td>
<td>0.010 (rad)</td>
</tr>
<tr>
<td>z quaternion ($q_z$)</td>
<td>0.051 (rad)</td>
</tr>
<tr>
<td>w quaternion ($q_w$)</td>
<td>0.017 (rad)</td>
</tr>
<tr>
<td>forward body-fixed velocity ($u$)</td>
<td>0.086 (m/s)</td>
</tr>
<tr>
<td>side body-fixed velocity ($v$)</td>
<td>0.147 (m/s)</td>
</tr>
<tr>
<td>down body-fixed velocity ($w$)</td>
<td>0.057 (m/s)</td>
</tr>
<tr>
<td>global north position ($n$)</td>
<td>0.548 (m)</td>
</tr>
<tr>
<td>global east position ($e$)</td>
<td>0.325 (m)</td>
</tr>
<tr>
<td>global down position ($d$)</td>
<td>0.061 (m)</td>
</tr>
</tbody>
</table>

Figures 5 through 8 provide a sample visual confirmation of the quality of the results that the presented filter can produce. Table I below provides RMS error calculations for the state estimates over the same manual flight. These results confirm the performance of the states not shown in the figures along with those that are shown.

### B. Estimates in Control Feedback: Hover

The vehicle was commanded to hover at a spot one meter above the take-off location for these results. The controller is based off a previous design [31] with some modifications to account for the relative coordinate systems described above.

Table II provides the RMS errors of all the states and the global positions during the hover flight. Notice that the performance has improved compared to that of Table I. We believe that this is due to several reasons. The first is that the vehicle was commanded to hover and many more VO measurements compared to the same keyframe were made, which allowed more time for filter convergence. Secondly, since the control is based on the estimates, any deviations affect the control and cause motion excitation that increases the observability of that state; this is especially applicable for the attitude and velocity states. Finally, during the manual flight, faster speeds were obtained which could have contributed to more motion blur or other vision artifacts that could decrease the accuracy of the VO measurements.

Figure 9, created based on a similar figure in [32], demonstrates the results from the proposed navigation approach with the current state of the art in estimates-in-the-loop hover performance. Here we see that the proposed approach is close to state-of-the-art performance when compared to any sensor and performs better in north and east by a significant margin than either of the other methods that use a camera.
TABLE II
The RMS errors in the state from estimates produced during the estimates-in-the-loop hover flight.

<table>
<thead>
<tr>
<th>RMS Error for Estimates</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>front relative position ((f))</td>
<td>0.019 (m)</td>
</tr>
<tr>
<td>right relative position ((r))</td>
<td>0.037 (m)</td>
</tr>
<tr>
<td>down relative position ((d))</td>
<td>0.092 (m)</td>
</tr>
<tr>
<td>x quaternion ((q_x))</td>
<td>0.006 (rad)</td>
</tr>
<tr>
<td>y quaternion ((q_y))</td>
<td>0.009 (rad)</td>
</tr>
<tr>
<td>z quaternion ((q_z))</td>
<td>0.041 (rad)</td>
</tr>
<tr>
<td>w quaternion ((q_w))</td>
<td>0.017 (rad)</td>
</tr>
<tr>
<td>forward body-fixed velocity ((u))</td>
<td>0.071 (m/s)</td>
</tr>
<tr>
<td>side body-fixed velocity ((v))</td>
<td>0.061 (m/s)</td>
</tr>
<tr>
<td>down body-fixed velocity ((w))</td>
<td>0.051 (m/s)</td>
</tr>
<tr>
<td>global north position ((n))</td>
<td>0.154 (m)</td>
</tr>
<tr>
<td>global east position ((e))</td>
<td>0.099 (m)</td>
</tr>
<tr>
<td>global down position ((d))</td>
<td>0.049 (m)</td>
</tr>
</tbody>
</table>

Fig. 9. A comparison of the standard deviations of estimates-in-the-loop hover performance for several different platforms. This data is based on a similar figure published in [32] using material from [33], [3], [34], [32], [35], [36]. Standard deviations are displayed in the different north, east, and down directions, hence lower numbers denote better performance. Note that the proposed estimation approach is competitive with the state of the art in the north and east directions. Platforms are provided by either MikroKopter or Ascending Technologies.

C. Estimates in Control Feedback: Waypoint Path

Figures 10 through 14 demonstrate the results of some of the states during a flight with the state estimates in the control loop. The vehicle avoided obstacles by following paths generated by an onboard path planning algorithm. Goal locations for the vehicle were given as inputs from a remote operator.

Figure 10 provides a comparison of the true and estimated global 3D path of the vehicle. Recall that global information is not a necessary input to the filter. These dead-reckoning results, created using a MEMs-grade IMU and VO (through the MEKF), are provided for information only. In Figure 11 we see the results for the relative forward position state \(f\). There were 26 new nodes created during this autonomous flight. The body frame velocity \(u\) results are depicted in Figure 12. The estimates track the truth, even though the magnitude of the speed is small. The \(y\) and \(z\) components of the quaternion \(q_y\) are shown in Figures 13 and 14. The \(y\) quaternion roughly corresponds to the pitch angle of the hexacopter for this flight and the \(z\) component is close to the yaw angle of the vehicle. The yaw depicted in Figure 14 is the global yaw.

Fig. 10. A comparison between truth and estimates for the global 3D path. This estimate is obtained by a vector sum of the edges and the current state at each time-step. This information is not required for the algorithms, but it is helpful in quantifying the quality of the estimates. The green dots in the figure denote locations of the keyframes. The hexacopter maneuvers around some obstacles to achieve a goal location selected by a remote operator.

Fig. 11. Relative front position \(f\) truth and estimates. This data is from an autonomous flight with estimates in the control loop. The discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to jump to the new relative position. The global positions do not jump when new nodes are created. We express the global truth from the motion capture in the relative node coordinate frame for the comparison of these results. Results for the relative right and down positions are similar.

V. Conclusions

A relative, vision-based framework, like the approach described here, is an important step in furthering the capabilities of indoor aerial navigation. Current approaches that require globally referenced states often suffer deficiencies from the need for additional state elements to incorporate relative measurements, waiting periods to process global consistency, computation demands on ground stations, or schemes to accommodate large jumps in pose when loop closures are applied. Utilizing a relative approach allows more flexibility as the critical real-time processes of localization and control do not depend on computationally demanding optimization and loop-closure processes. Additionally, relative exteroceptive measurement updates are supported natively in the proposed MEKF and front-facing keyframes provide a rich source of information for path planning.

We have shown the viability and accuracy of the proposed relative MEKF through hardware results comparing estimates to truth
Fig. 12. Body-fixed frame forward velocity $u$ truth and estimate comparison. Notice that there are no discontinuities, as the body-frame velocity is not relative. Results for the right and down body-fixed velocities are similar.

Fig. 13. The $y$ component of the quaternion $q_y$, which is approximately the pitch angle of the hexacopter, comparison of truth and estimates. There are not any discontinuities, as this portion of the quaternion is not relative.

Fig. 14. The $z$ component of the quaternion $q_z$, truth and estimate comparison. This is approximately equal to the yaw angle for this flight. Note that there is some drift as we cannot measure the global yaw angle. We do not show the relative yaw angle in these results, even though that is what is actually tracked in the state vector.

information. We show that we can reliably estimate relative states, marginalize out states, augment new states when a new node is created, and smoothly estimate non-relative states during flights. The estimates are robust enough for utilization in feedback control. The demonstrated feedback-controlled hover results exceed the capabilities of contemporary vision-based approaches.

ACKNOWLEDGMENTS

This work was supported through the DoD SMART Scholarship program.

REFERENCES


